

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      111111      000000      333333
PPPPPPPP      AAAAAA      SSSSSSSS      111111      000000      333333
PP           PP  AA        AA  SS          11          00          00      33          33
PP           PP  AA        AA  SS          11          00          00      33          33
PP           PP  AA        AA  SS          11          00          00      33          33
PP           PP  AA        AA  SS          11          00          00      33          33
PPPPPPPP      AA        AA  SSSSSS      11          00          00      33          33
PPPPPPPP      AA        AA  SSSSSS      11          00          00      33          33
PP           AAAAAAAAAA      SS          11          00          00      33          33
PP           AAAAAAAAAA      SS          11          00          00      33          33
PP           AA        AA  SS          11          00          00      33          33
PP           AA        AA  SS          11          00          00      33          33
PP           AA        AA  SSSSSSSS      111111      000000      333333      .....
PP           AA        AA  SSSSSSSS      111111      000000      333333      .....

```

```

LL           111111      SSSSSSSS
LL           111111      SSSSSSSS
LL           11          SS
LL           11          SS
LL           11          SS
LL           11          SS
LL           11          SSSSSS
LL           11          SSSSSS
LL           11          SS
LL           11          SS
LL           11          SS
LL           11          SS
LLLLLLLLLLLL 111111      SSSSSSSS
LLLLLLLLLLLL 111111      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 OUTPUT ; 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 V1.2 -- JANUARY 1981 **
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 08Jan80: 1) Allow output of 31 character scalar values in
0000 47 : PASSWRITESCAL.
0000 48 : 2) Bugfix in PASSPUTBIN. Compiler was calling PASSWRITEOK
0000 49 : twice under some circumstances.
0000 50 : Paul Hohensee
0000 51 : 05May80: Fix PASSWRITESCAL to force output even if specified
0000 52 : field width <= 0.
0000 53 : Paul Hohensee
0000 54 :
0000 55 : 16Oct80: Change PRN CRLF so that lines are printed:
0000 56 : <LF> <text> <CR>
0000 57 : Susan Azibert

```

```

0000 58 :
0000 59 :      13Jan81: Change all tests of status returns from RMS to
0000 60 :             BLBC RO,label and BLBS RO,label from CMPL RO,#RMS$NORMAL;
0000 61 :             BNEQ label, etc.
0000 62 :             Paul Hohensee
0000 63 :
0000 64 :      13Jan81: Change scalar output so element name is right truncated
0000 65 :             for short field width, right justified and blank filled
0000 66 :             for long field width.
0000 67 :
0000 68 :      28Aug81: Use General addressing mode. Steve Lionel.
0000 69 :

```

```

0000 70 : *****
0000 71 : *****
0000 72 : **
0000 73 : **
0000 74 : **          SECTION 3          **
0000 75 : **          OUTPUT PROCEDURES  **
0000 76 : **
0000 77 : **
0000 78 : **
0000 79 : *****
0000 80 : *****

```

```

0000 81 :
0000 82 :
0000 83 : For any file variable the following storage is assumed:
0000 84 :

```

```

0000 85 :
0000 86 : FSB:  :-----:
0000 87 :       : POINTER  :
0000 88 :       :-----:
0000 89 :       : STATUS WORD :
0000 90 :       :-----:
0000 91 :       : LAST      :
0000 92 :       :-----:
0000 93 :       : LINELIMIT :
0000 94 :       :-----:
0000 95 :       : LINECOUNT :
0000 96 :       :-----:
0000 97 :       : RECORD NUMBER :
0000 98 : RAB:  :-----:
0000 99 :       : 44(HEX) BYTES :
0000 100 :       : .             :
0000 101 :       : .             :
0000 102 :       :-----:
0000 103 : FAB:  :-----:
0000 104 :       : 50(HEX) BYTES :
0000 105 :       : .             :
0000 106 :       : .             :
0000 107 :       :-----:
0000 108 : NAM:  :-----:
0000 109 :       : 38(HEX) BYTES :
0000 110 :       : .             :
0000 111 :       : .             :
0000 112 :       :-----:
0000 113 :
0000 114 :

```

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

```

0000 115 : -----
0000 116 :
0000 117 : Macro options
0000 118 :
0000 119 :     .DSABL  GBL           ; no undefined references
0000 120 :     .ENABL  FPT           ; rounded arithmetic
0000 121 :
0000 122 : External references
0000 123 :
0000 124 :     .EXTRN  PASSIOERROR
0000 125 :     .EXTRN  PASSWRITEOK
0000 126 :     .EXTRN  PASSBUFFEROVER
0000 127 :     .EXTRN  PASSWITELN
0000 128 :
0000 129 :
0000 130 :     .EXTRN  FOR$CNV_OUT_D
0000 131 :     .EXTRN  FOR$CNV_OUT_E
0000 132 :     .EXTRN  FOR$CNV_OUT_F
0000 133 :     .EXTRN  FOR$CNV_OUT_I
0000 134 :     .EXTRN  FOR$CNV_OUT_O
0000 135 :     .EXTRN  FOR$CNV_OUT_Z
0000 136 :
0000 137 : Provide definitions of system values
0000 138 :
0000 139 :     $DSCDEF                ; string descriptor definitions
0000 140 :     $FABDEF
0000 141 :     $RABDEF
0000 142 :     $RMSDEF                ; for status code checking
0000 143 :
0000 144 : PASCAL compiler constants
0000 145 :
0000 146 : Note: The constants below with the names 'PASSC_XXXXX' are
0000 147 : used in the PASCAL compiler with the names 'XXXXX'. If the
0000 148 : values in the compiler are altered then the values below
0000 149 : must be altered accordingly.
0000 150 :
0000 151 :     PASSC_DFLTRECSI = 257;      ; default buffer size
0000 152 :     PASSC_NIL = 0              ; NIL pointer
0000 153 :     PASSC_TRUE = 1            ; TRUE
0000 154 :     PASSC_FALSE = 0          ; FALSE
0000 155 :     PASSC_NOCARR = 0         ; no carriage control
0000 156 :     PASSC_CARRIAGE = 1       ; FORTRAN carriage control
0000 157 :     PASSC_LIST = 2           ; LIST carriage control
0000 158 :     PASSC_PRN = 3            ; PRN carriage control
0000 159 :
0000 160 : PRN carriage control constants
0000 161 :
0000 162 :     PRN_CRLF = ^X8D01        ; PRN carriage control constant
0000 163 :                               ; for <LF> <text> <CR>
0000 164 :     PRN_NULL = ^X0000       ; PRN carriage control constant
0000 165 :                               ; for no carriage control
0000 166 :
0000 167 : File status block constants
0000 168 :
00000018 0000 169 :     FSB$C_BLN = ^X18         ; FSB block length
0000 170 :     FSB$V_OPEN = 5
0000 171 :     FSB$V_EOF = 1

```

```

0000 172 : FSBSV_EOLN = 2
0000 173 : FSBSV_GET = 3
0000 174 : FSBSV_TXT = 4 ; textfile flag
0000 175 : FSBSV_RDLN = 0 ; last access READLN
0000 176 : FSBSV_DIR = 6 ; direct access flag
0000 177 : FSBSV_PUT = 7
0000 178 : FSBSV_INT = 8 ; internal flag
0000 179 : FSBSV_PRMT = 9 ; prompt flag
0000 180 : FSBSV_OUTPUT = 10 ; OUTPUT file flag
0000 181 : FSBSV_ACTIN = 11 ; actual input flag
0000 182 : FSBSV_DELZ = 30 ; delete file if empty
0000 183 : FSBSV_INC = 31 ; included file flag
0000 184 : FSBSB_CC = 6 ; carriage control byte offset
0000 185 : FSBSM_OPEN = ^X0020
0000 186 : FSBSM_EOF = ^X0002
0000 187 : FSBSM_EOLN = ^X0004
0000 188 : FSBSM_GET = ^X0008
0000 189 : FSBSM_PRMT = ^X0200
0000 190 : FSBSM_PUT = ^X00000080
0000 191 : FSBSM_TXT = ^X0010
0000 192 : FSBSM_RDLN = ^X0001
0000 193 : FSBSM_DIR = ^X00000040
0000 194 : FSBSM_INT = ^X00000100
0000 195 : FSBSM_OUTPUT = ^X0400
0000 196 : FSBSM_ACTIN = ^X0800
0000 197 : FSBSM_DELZ = ^X40000000
0000 198 : FSBSM_INC = ^X80000000
0000 199 : FSBSL_CNT = 16 ; line count (textfiles)
0000 200 : FSBSL_INC = 20 ; %INCLUDE block address
0000000C 0000 201 : FSBSL_LIM = 12 ; linelimit
00000008 0000 202 : FSBSL_LST = 8 ; last word offset
0000 203 : FSBSL_PFSB = 20 ; related file FSB for prompting
0000 204 : ; for INPUT, has address of OUTPUT FSB
0000 205 : ; for OUTPUT, has address of INPUT FSB
0000 206 : ; (shares storage with include address
0000 207 : ; and direct access record
0000 208 : ; buffer address)
0000 209 : FSBSL_REC = 20 ; record buffer address for
0000 210 : ; direct access (shares storage
0000 211 : ; with include address and related
0000 212 : ; file FSB)
0000 213 : FSBSL_STA = 4 ; status word offset
0000 214 :
0000 215 : Character constants
0000 216 :
00000020 0000 217 : TAB = ^X09
0000 218 : SPACE = ^X20
0000000C 0000 219 : DOLLAR = ^X24
0000002A 0000 220 : FORMFEED = ^XC
0000 221 : STAR = ^X2A
0000 222 : PLUS = ^X2B
0000 223 : MINUS = ^X2D
0000 224 : POINT = ^X2E
00000030 0000 225 : ZERO = ^X30
00000031 0000 226 : ONE = ^X31
0000 227 : NINE = ^X39
0000 228 : AA = ^X41

```

```

0000 229 : DD = ^X44
0000 230 : EE = ^X45
0000 231 : ZZ = ^X5A
0000 232 : UNDERSCORE = ^X5F
0000 233 : AA_SMALL = ^X61
0000 234 : ZZ_SMALL = ^X7A
0000 235 :
0000 236 :
00000000 237 : .PSECT _PASSCODE, PIC,EXE,SHR,NOWRT
0000 238 :
0000 239 : *****
0000 240 : *
0000 241 : * PASSPUTBIN *
0000 242 : * pas$putbinary *
0000 243 : *
0000 244 : *****
0000 245 :
0000 246 : Argument offsets
0000 247 :
0000 248 : AP ; number of arguments (1)
00000004 0000 249 : FSB_DISP = 04 ; FSB address
0000 250 :
00000000'GF 00C0 0000 251 : .ENTRY PASSPUTBIN,^M<R6,R7>
6C FA 0002 252 : CALLG (AP),G^PASSWRITEOK
02 11 0009 253 : brb newent
00C0 000B 254 : .entry pas$putbinary,^m<r6,r7>
56 04 AC D0 000D 255 newent:
57 18 56 C1 0011 256 : MOVL FSB_DISP(AP),R6 ; R6 = address of FSB
04 A7 02 CA 0015 257 : ADDL3 R6,#FSB$C_BLN,R7 ; R7 = address of RAB
05 50 E9 001E 258 : $PUT RAB=R7
1E A7 00 90 0022 259 : BICL2 #RAB$M_TPT,RAB$R_ROP(R7); clear TPT bit
04 0025 260 : BLBC R0,910$ ; branch if error
0029 261 : MOVB #RAB$C_SEQ,RAB$B_RAC(R7); make sure sequential
002A 262 : RET
002A 263 :
002A 264 : Write error
002A 265 :
002A 266 910$:
7E 78 A7 9A 002A 267 : PUSHL R0
70 A7 DD 002C 268 : MOVZBL <RAB$C_BLN+FAB$B_FNS>(R7),-(SP)
00000000'GF 03 FB 0030 269 : PUSHL <RAB$C_BLN+FAB$R_FNA>(R7)
003A 270 : CALLS #3,G^PASSIOERROR
003A 271 :
0000003A 272 :
003A 273 : .PSECT _PASSCODE, PIC,EXE,SHR,NOWRT
003A 274 :
003A 275 : *****
003A 276 : *
003A 277 : * PASSPUTTXT *
003A 278 : *
003A 279 : *****
003A 280 :
003A 281 : Increments the file pointer. If the pointer is positioned at the last
003A 282 : position at entry time then the buffer has overflowed.
003A 283 :
003A 284 : Argument offsets
003A 285 :

```

```

00000004 003A 286 : AP ; number of arguments (1)
00000004 003A 287 : FSB_DISP = 04 ; FSB address
00000004 003A 288 :
00000000'GF 000C 003A 289 : .ENTRY PASS$PUTTXT,^M<R2,R3>
52 04 AC D0 003C 290 CALLG (AP),G^PASS$WRITEOK
53 18 52 C1 0043 291 MOVL FSB_DISP(AP),R2 ; R2 = address of FSB
08 A2 62 D1 0047 292 ADDL3 R2,#FSB$C_BLN,R3 ; R3 = address of RAB
00000000'GF 07 19 004B 293 CMPL (R2),FSB$_LST(R2)
6C FA 004F 294 BLSS 190$ ; branch if ok
00000000'GF 6C FA 0051 295 CALLG (AP),G^PASS$BUFFEROVER ; buffer overflow
005B 296 190$:
62 D6 005B 297 INCL (R2)
04 005A 298 RET
005B 299 :
00000000'GF 005B 300 :
005B 301 : .PSECT _PASS$CODE, PIC,EXE,SHR,NOWRT
005B 302 :
005B 303 : *****
005B 304 : *
005B 305 : * PASS$WRITECHAR *
005B 306 : *
005B 307 : *****
005B 308 :
005B 309 : Writes a character to the file buffer. If the field width is less
005B 310 : than or equal to zero then zero field width is used (ie. no output).
005B 311 :
005B 312 : Argument offsets
005B 313 :
00000004 005B 314 :
00000008 005B 315 AP ; number of arguments (4)
0000000C 005B 316 FSB_DISP = 04 ; FSB address
00000010 005B 317 CHR_DISP = 08 ; character value (low order byte)
005B 318 FLD_DISP = 12 ; field width (by value)
005B 319 NOT_DISP = 16 ; (not used)
007C 005B 320 :
56 04 AC D0 005D 321 : .ENTRY PASS$WRITECHAR,^M<R2,R3,R4,R5,R6>
00000000'GF 56 DD 0061 322 MOV_ FSB_DISP(AP),R6 ; R6 = address of FSB
0C AC D5 0063 323 PUSHL R6
028 15 006A 324 CALLS #1,G^PASS$WRITEOK
006D 325 TSTL FLD_DISP(AP) ; check field width
006F 326 BLEQ 199$ ; exit if zero field width
006F 327 :
006F 328 : Check if enough room
50 08 A6 66 C3 006F 329 SUBL3 (R6),FSB$_LST(R6),R0 ; R0 = number of bytes left
50 0C AC D1 0074 330 CMPL FLD_DISP(AP),R0
09 15 0078 331 BLEQ 110$
00000000'GF 56 DD 007A 332 PUSHL R6
0C AC D7 007C 333 CALLS #1,G^PASS$BUFFEROVER ; buffer overflow
0083 334 110$:
0C AC 20 00 B6 00 2C 0083 335 DECL FLD_DISP(AP)
63 08 AC 01 28 0086 336 MOVCS #0,@(R6),#SPACE,FLD_DISP(AP),@(R6); blank fill
66 53 D0 008D 337
008F 338 MOVCS #1,CHR_DISP(AP),(R3) ; put character
0094 339 MOVL R3,(R6) ; update file pointer
0097 340 199$:
04 0097 341 RET
0098 341 :

```



```

0098 342 :
00000098 343 : .PSECT _PASSCODE, PIC,EXE,SHR,NOWRT
0098 344 :
0098 345 : *****
0098 346 : *
0098 347 : * PASSWRITESTR *
0098 348 : *
0098 349 : *****
0098 350 :
0098 351 : Writes a string right justified with blank fill on the designated
0098 352 : file. If the field width is smaller than the string length the string
0098 353 : is truncated on the right.
0098 354 :
0098 355 : Argument offsets
0098 356 :
0098 357 :
00000004 0098 358 : AP ; number of arguments (4)
00000008 0098 359 : FSB_DISP = 04 ; FSB address
0000000C 0098 360 : STR_DISP = 08 ; string address
00000010 0098 361 : FLD_DISP = 12 ; field width (by value)
0098 362 : LEN_DISP = 16 ; string length (by value)
0098 363 :
0098 364 : .ENTRY PASSWRITESTR,^M<R2,R3,R4,R5,R7>
52 04 AC D0 009A 364 : MOVL FSB_DISP(AP),R2 ; R2 = address of FSB
00000000'GF 52 DD 009E 365 : PUSHL R2
0C AC D5 00A0 366 : CALLS #1,G^PASSWRITEOK
45 15 00A7 367 : TSTL FLD_DISP(AP)
0098 368 : BLEQ 199$ ; exit if field width <= 0
0098 369 :
0098 370 : Check if passing string value or address
0098 371 :
04 10 AC D1 00AC 372 : CMPL LEN_DISP(AP),#4
0098 373 : BLEQ 100$
57 08 AC D0 00B0 373 : MOVL STR_DISP(AP),R7 ; R7 = address of string
04 11 00B6 375 : BRB 101$
0098 376 : 100$:
57 08 AC DE 00B8 377 : MOVAL STR_DISP(AP),R7 ; R7 = address of string
0098 378 : 101$:
50 08 A2 62 C3 00BC 379 : SUBL3 (R2),FSB$L_LST(R2),R0
50 0C AC D1 00C1 380 : CMPL FLD_DISP(AP),R0
0098 381 : BLEQ 105$
0098 382 : PUSHL R2
00000000'GF 01 FB C0C9 383 : CALLS #1,G^PASSBUFFEROVER ; buffer overflow
0098 384 : 105$:
54 0C AC 10 AC C3 00D0 385 : SUBL3 LEN_DISP(AP),FLD_DISP(AP),R4; R4 = number of bytes to pad
0098 386 : BGTRU 110$ ; branch if padding required
00 B2 67 0C AC 28 00D6 386 : MOVC3 FLD_DISP(AP),(R7),@ (R2) ; write width characters
0098 387 : BRB 111$
0098 388 :
0098 389 : 110$:
00 B2 54 20 00 B2 00 2C 00E0 389 : MOVC5 #0,@ (R2),#SPACE,R4,@ (R2); blank fill
0098 390 : MOVC3 LEN_DISP(AP),(R7),(R3) ; write string
0098 391 :
0098 392 : 111$:
04 BC 53 D0 00ED 393 : MOVL R3,@FSB_DISP(AP) ; update pointers
0098 394 :
0098 395 : 199$:
0098 396 : RET
0098 397 :
000000F2 398 : .PSECT _PASSCODE, PIC,EXE,SHR,NOWRT

```

```

00F2 399 :
00F2 400 :
00F2 401 : *****
00F2 402 : *          *
00F2 403 : *  PASSWRITESCAL  *
00F2 404 : *          *
00F2 405 : *****
00F2 406 : Write out a scalar value on the designated text file. If the field
00F2 407 : width is less than that required for the value, the value is left truncated
00F2 408 : If the field width is greater than that required for the value, the
00F2 409 : value is right justified with blank fill.
00F2 410 :
00F2 411 : Argument offsets
00F2 412 :
00F2 413 :
00000004 00F2 414 : AP ; number of arguments (4)
00000008 00F2 415 : FSB_DISP = 04 ; FSB address
0000000C 00F2 416 : SCA_DISP = 08 ; scalar value (by value)
00000010 00F2 417 : FLD_DISP = 12 ; field width (by value)
00000014 00F2 418 : NAM_DISP = 16 ; namelist address
00F2 419 : MAX_DISP = 20 ; maximal ordinal value of
00F2 420 : scalar (by value)
00F2 421 :
00F2 422 : Constants
00000020 00F2 423 : namelen = 32 ; length in bytes of one entry in
00F2 424 : name list.
00F2 425 :
00FC 00F2 426 : .ENTRY PASSWRITESCAL, ^M<R2,R3,R4,R5,R6,R7>
56 04 AC DO 00F4 427 : MOVL FSB_DISP(AP),R6 ; R6 = address of FSB
00000000 GF 56 DD 00F8 428 : PUSHL R6
57 20 08 AC C5 0101 429 : CALLS #1,G^PASSWRITEOK
57 10 AC C0 0106 430 : MULL3 SCA_DISP(AP),#namelen,R7
010A 431 : ADDL2 NAM_DISP(AP),R7 ; R7 = scalar name address
010A 432 :
010A 433 : Calculate scalar name length and check for bounds
08 AC D5 010A 434 :
14 AC 08 AC D1 010D 435 : TSTL SCA_DISP(AP)
67 20 20 3A 0116 436 : BLSS 900$
51 20 50 C3 011F 437 : CMPL SCA_DISP(AP),MAX_DISP(AP)
011A 438 : BGTR 900$
011E 439 : LOCC #SPACE,#namelen,(R7)
011E 440 : SUBL3 R0,#namelen,R1
011E 441 :
011E 442 : Call PASSWRITESTR to actually write the value to the buffer
011E 443 :
04 51 DD 011E 444 : PUSHL R1 ; pass name length
04 AC DD 0120 445 : PUSHL FLD_DISP(AP) ; pass field width
04 51 D1 0123 446 : CMPL R1,R4 ; pass by value or reference
04 15 0126 447 : BLEQ 110$
57 DD 0128 448 : PUSHL R7 ; by reference
02 11 012A 449 : BRB 111$
012C 450 110$:
67 DD 012C 451 : PUSHL (R7) ; by value
012E 452 111$:
FF62 CF 04 AC DD 012E 453 : PUSHL FSB_DISP(AP)
04 FB 0131 454 : CALLS #4,PASSWRITESTR
04 0136 455 : RET

```

```

0137 456 :
0137 457 900$:
7E 83A4 8F 3C 0137 458 MOVZWL #^X83A4,-(SP)
7E 0090 C6 9A 013C 459 MOVZBL <FSB$C_BLN+RAB$C_BLN+FAB$B_FNS>(R6),-(SP)
00000000'GF 03 FB 0141 460 PUSHL <FSB$C_BLN+RAB$C_BLN+FAB$L_FNA>(R6)
014C 461 CALLS #3,G^PASSIOERROR
014C 462 :
014C 463 :
0000014C 464 .PSECT _PASSCODE, PIC,EXE,SHR,NOWRT
014C 465 :
014C 466 *****
014C 467 *
014C 468 * PASSWRITEINT *
014C 469 *
014C 470 *****
014C 471 :
014C 472 : Writes an integer right justified in the designated field width.
014C 473 : If the field width is less than the minimum the minimum field width is
014C 474 : used. If the integer won't fit in the designated field width then the
014C 475 : amount needed is used. If the field width used will overflow the buffer
014C 476 : a runtime error occurs.
014C 477 :
014C 478 : Argument offsets
014C 479 :
014C 480 :
00000004 014C 481 AP ; number of arguments (4)
00000008 014C 482 FSB_DISP = 04 ; FSB address
0000000C 014C 483 INT_DISP = 08 ; integer value
00000010 014C 484 FLD_DISP = 12 ; field width (by value)
014C 485 NOT_DISP = 16 ; (not used)
014C 486 :
014C 487 : Other constants
00000001 014C 488 IMINP = 1 ; minimum field width for
014C 489 ; positive integers
00000002 014C 490 IMINN = 2 ; minimum field width for
014C 491 ; negative integers
00000014 014C 492 IMAX = 20 ; maximum field width needed
014C 493 ; for integers
014C 494 :
56 04 AC 03FC 014C 495 .ENTRY PASSWRITEINT,^M<R2,R3,R4,R5,R6,R7,R8,R9>
00000000'GF 01 FB 0152 496 MOVL FSB_DISP(AP),R6 ; R6 = address of FSB
015B 497 PUSHL R6
015B 498 CALLS #1,G^PASSWRITEOK
015B 499 :
015B 500 : Make room for descriptor on stack
015B 501 :
5E 08 C2 015B 502 SUBL2 #DSC$C_S_BLN,SP
58 5E D0 015E 503 MOVL SP,R8 ; R8 = address of descriptor
0161 504 :
0161 505 : Check for minimum field width (1 for positive, 2 for negative)
0161 506 :
53 0C AC D0 0161 507 MOVL FLD_DISP(AP),R3 ; R3 = field width
0165 508 TSTL INT_DISP(AP) ; test sign of value
0168 509 BLSS 110$
016A 510 ; positive value
53 01 D1 016A 511 CMPL #IMINP,R3 ; use at least minimum
016D 512 BLEQ 120$

```

```

PA
SY
$$
$$
S1
S2
S2
S3
S3
S4
S4
S5
S6
CH
DI
DOI
DS
DS
DS
EM
FAI
FAI
FAI
FAI
FLI
FM
FM
FOI
FOI
FOI
FOI
FOI
FOI
FOI
FOI
FOI
FOI
FOI
IM
IM
IN
LEI
MA
NAI
NAI
NEI
NO
ON
ON
OV
PA
PA
PA
PA
PA
PA

```

```

53 01 D0 016F 513      MOVL  #IMINP,R3
   08 11 0172 514      BRB   120$
   53 02 D1 0174 515      110$: ; negative value
   03 15 0177 516      CMPL  #IMINN,R3      ; use at least minimum
   53 02 D0 0179 518      BLEQ  120$
   01 017C 519      MOVL  #IMINN,R3
   017C 520      120$: ; R3 = field width
   017C 521      : Convert number to character string
   017C 522      :
57 08 A6 66 C3 017C 523      SUBL3 (R6),FSB$L_LST(R6),R7 ; R7 = number of bytes left in line
   57 53 D1 0181 524      CMPL  R3,R7
   09 15 0184 525      BLEQ  125$
00000000'GF 56 DD 0186 526      PUSHL R6
   01 FB 0188 527      CALLS #1,G^PASS$BUFFEROVER ; buffer overflow
   018F 528      125$:
   68 53 B0 018F 529      MOVW  R3,DSC$W_LENGTH(R8) ; pass field width
   04 A8 66 D0 0192 530      MOVL  (R6),DSC$A_POINTER(R8) ; pass buffer address
   58 DD 0196 531      PUSHL R8 ; pass descriptor address
   08 AC DD 0198 532      PUSHL INT_DISP(AP)
00000000'GF 02 FB 019B 533      CALLS #2,G^FOR$CNV_OUT_I
   05 50 E9 01A2 534      BLBC  R0,130$
   66 53 C0 01A5 535      ADDL2 R3,(R6) ; update file pointer
   36 11 01A8 536      BRB   199$ ; exit, conversion succeeded
   01AA 537      :
   01AA 538      : Bad conversion; use a larger buffer and try again
   01AA 539      :
   01AA 540      130$:
   68 14 B0 01AA 541      MOVW  #IMAX,DSC$W_LENGTH(R8) ; pass buffer length
   5E 14 C2 01AD 542      SUBL2 #IMAX,SP ; make room for buffer on stack
   59 5E D0 01B0 543      MOVL  SP,R9
   04 A8 59 D0 01B3 544      MOVL  R9,DSC$A_POINTER(R8) ; pass buffer address
   58 DD 01B7 545      PUSHL R8 ; pass descriptor address
   08 AC DD 01B9 546      PUSHL INT_DISP(AP)
00000000'GF 02 FB 01BC 547      CALLS #2,G^FOR$CNV_OUT_I
   1B 50 E9 01C3 548      BLBC  R0,910$
   69 14 20 3B 01C6 549      SKPC  #SPACE,#IMAX,(R9) ; skip leading spaces
   01CA 550      ; R0 = number of remaining
   01CA 551      ; characters
   01CA 552      ; R1 = address of remaining
   01CA 553      ; characters
   57 50 D1 01CA 554      CMPL  R0,R7 ; check if enough room
   09 15 01CD 555      BLEQ  140$
00000000'GF 56 DD 01CF 556      PUSHL R6
   01 FB 01D1 557      CALLS #1,G^PASS$BUFFEROVER ; buffer overflow
   01D8 558      140$:
   00 B6 61 50 28 01D8 559      MOVC3 R0,(R1),a(R6) ; move string to output buffer
   66 53 D0 01DD 560      MOVL  R3,(R6) ; update file pointer
   01E0 561      199$:
   04 01E0 562      RET
   01E1 563      :
   01E1 564      : Output conversion error
   01E1 565      :
   01E1 566      910$:
7E 83A4 8F 3C 01E1 567      MOVZWL #^X83A4,-(SP)
7E 0090 C6 9A 01E6 568      MOVZBL <FSB$C_BLN+RAB$C_BLN+FAB$B_FNS>(R6),-(SP)
   0088 C6 DD 01EB 569      PUSHL <FSB$C_BLN+RAB$C_BLN+FAB$L_FNA>(R6)

```

```

00000000'GF 03 FB 01EF 570 CALLS #3,G^PASSIOERROR
                01F6 571 :
                01F6 572 :
000001F6 573 : .PSECT _PASSCODE, PIC,EXE,SHR,NDWRT
                01F6 574 :
                01F6 575 : *****
                01F6 576 : *
                01F6 577 : * PASSWRITEDOUBE *
                01F6 578 : *
                01F6 579 : *****
                01F6 580 :
                01F6 581 : Write out a double precision number in 'E' format. A minimum
                01F6 582 : field width of FMIN is used.
                01F6 583 :
                01F6 584 : Argument offsets
                01F6 585 :
                01F6 586 : AP ; number of arguments (4)
00000004 01F6 587 : FSB_DISP = 04 ; FSB address
00000008 01F6 588 : DOB_DISP = 08 ; double number (by reference)
0000000C 01F5 589 : FLD_DISP = 12 ; field width (by value)
                01F6 590 :
                01F6 591 : Other constants
                01F6 592 :
00000008 01F6 593 : FMIN = 8 ; minimum field width
                01F6 594 :
                003C 01F6 595 : .ENTRY PASSWRITEDOUBE,^M<R2,R3,R4,R5>
                01F8 596 :
                01F8 597 : Make room for descriptor and double precision value on stack
                01F8 598 :
                01F8 599 : SUBL2 #<DSC$C_S_BLN+8>,SP
08 A1 5E 10 C2 01FB 600 : MOVL SP,R1 ; R1 = descriptor address
                51 5E D0 01FE 601 : MOVD @DOB_DISP(AP),DSC$C_S_BLN(R1); put value on stack
                08 BC 70 01FE 601 : MOVL #1,R5 ; set flag
                55 01 D0 0203 602 : BRW PASSWREALE ; jump to common code
                0010 31 0206 603 :
                0209 604 :
                0209 605 : *****
                0209 606 : *
                0209 607 : * PASSWRITEREAL *
                0209 608 : *
                0209 609 : *****
                0209 610 :
                0209 611 : Write a real number in 'E' format. A minimum field width of EMIN is
                0209 612 : used.
                0209 613 :
                0209 614 : Argument offsets
                0209 615 :
                0209 616 : AP ; number of arguments (4)
00000004 0209 617 : FSB_DISP = 04 ; FSB address
00000008 0209 618 : REL_DISP = 08 ; real number (by value)
0000000C 0209 619 : FLD_DISP = 12 ; field width (by value)
00000010 0209 620 : NOT_DISP = 16 ; (not used)
                0209 621 :
                0209 622 : Other constants
                0209 623 :
00000008 0209 624 : EMIN = 08 ; minimum field width
                0209 625 :
                003C 0209 626 : .ENTRY PASSWRITEREAL,^M<R2,R3,R4,R5>

```



```

0275 684 :
0275 685 : 910$:
7E 83A4 8F 3C 0275 686 MOVZWL #^X83A4,-(SP)
7E 0090 C2 9A 027A 687 MOVZBL <FSB$C_BLN+RAB$_BLN+FAB$B_FNS>(R2),-(SP)
00000000 GF 03 DD 027F 688 PUSHL <FSB$C_BLN+RAB$_BLN+FAB$L_FNA>(R2)
0283 689 CALLS #3,G^PASSIOERROR
028A 690 :
028A 691 :
0000 028A 692 .PSECT _PASSCODE, PIC,EXE,SHR,NOWRT
028A 693 :
028A 694 : *****
028A 695 : * *
028A 696 : * PASSWRITEDOUBF *
028A 697 : * *
028A 698 : *****
028A 699 :
028A 700 : Writes out a double number in fixed format.
028A 701 :
028A 702 : Argument offsets
028A 703 :
028A 704 : AP ; number of arguments (4)
00000004 028A 705 FSB_DISP = 04 ; FSB address
00000008 028A 706 DOB_DISP = 08 ; double value (by reference)
0000000C 028A 707 FLD_DISP = 12 ; field width (by value)
00000010 028A 708 DIG_DISP = 16 ; digits to right of decimal
028A 709 : ; point (by value)
028A 710 :
028A 711 : Other constants
028A 712 :
00000003 028A 713 FMIN = 3 ; minimum field width
0000002A 028A 714 FMAX = 42 ; maximum field width
028A 715 :
007C 028A 716 .ENTRY PASSWRITEDOUBF,^M<R2,R3,R4,R5,R6>
028C 717 :
028C 718 : Make room for descriptor and double precision value on stack
028C 719 :
028C 720 :
028C 720 SUBL2 #<DSC$C_S_BLN+8>,SP
028F 721 MOVL SP,R1 ; R1 = address of descriptor
0292 722 MOVD @DOB_DISP(AP),DSC$C_S_BLN(R1); put value on stack
0297 723 BRW PASS$WREALF ; jump to common code
029A 724 :
029A 725 : *****
029A 726 : * *
029A 727 : * PASSWRITEREALF *
029A 728 : * *
029A 729 : *****
029A 730 :
029A 731 : Writes out a real number in fixed format.
029A 732 :
029A 733 : Argument offsets
029A 734 :
029A 735 : AP ; number of arguments (4)
00000004 029A 736 FSB_DISP = 4 ; FSB address
00000008 029A 737 REL_DISP = 08 ; real number (by value)
0000000C 029A 738 FLD_DISP = 12 ; field width
00000010 029A 739 DIG_DISP = 16 ; digits to right of decimal
029A 740 : ; point (by value)

```





```

04 A1 66 D0 02F8 798      MOVL      (R6),DSC$A_POINTER(R1)
      00 DD 02FC 799      PUSHL     #0                ; scale factor
      54 DD 02FE 800      PUSHL     R4                ; digits in fraction
      51 DD 0300 801      PUSHL     R1                ; string descriptor
00000000'GF 08 A1 DF 0302 802      PUSHAL   DSC$C_S_BLN(R1)   ; value
      04 FB 0305 803      CALLS    #4,G^FOR$CNV_OUT_F
      05 50 E9 030C 804      BLBC     R0,130$
      66 53 C0 030F 805      ADDL2    R3,(R6)           ; update the file pointer
      3E 11 0312 806      BRB      199$
      0314 807      ;
      0314 808      ; Bad conversion; use a buffer of subfield+overflowsize and try again
      0314 809      ;
      0314 810      ;
      53 51 5E D0 0314 811      MOVL     SP,R1              ; R1 = descriptor address
      54 2A C1 0317 812      ADDL3    #FMAX,R4,R3       ; R3 = new buffer size
      5E 53 C2 031B 813      SUBL2    R3,SP             ; make room on stack
      61 53 B0 031E 814      MOV'     R3,DSC$W_LENGTH(R1)
04 A1 5E D0 0321 815      MOVL     SP,DSC$A_POINTER(R1) ; buffer address
      00 DD 0325 816      PUSHL     #0
      54 DD 0327 817      PUSHL     R4                ; digits in fraction
      51 DD 0329 818      PUSHL     R1                ; descriptor address
00000000'GF 08 A1 DF 032B 819      PUSHAL   DSC$C_S_BLN(R1)   ; value address
      04 FB 032E 820      CALLS    #4,G^FOR$CNV_OUT_F
      1B 50 E9 0335 821      BLBC     R0,910$
      6E 53 20 3B 0338 822      SKPC     #SPACE,R3,(SP)    ; skip leading blanks
      55 50 D1 033C 823      CML     R0,R5
      09 15 033F 824      BLEQ    140$
00000000'GF 56 CD 0341 825      PUSHL     R6
      01 FB 0343 826      CALLS    #1,G^PASS$BUFFEROVER ; buffer overflow
      034A 827      ;
00 B6 61 50 28 034A 828      MOV3     R0,(R1),@ (R6)    ; store string
      66 53 D0 034F 829      MOVL     R3,(R6)           ; update file pointer
      0352 830      ;
      04 0352 831      RET
      0353 832      ;
      0353 833      ; Output conversion error
      0353 834      ;
      0353 835      ;
      7E 83A4 8F 3C 0353 836      MOVZWL   #^X83A4,-(SP)
      7E 0090 C6 9A 0358 837      MOVZBL   <FSB$C_BLN+RAB$C_BLN+RAB$B_FNS>(R6),-(SP)
00000000'GF 0088 C6 DD 035D 838      PUSHL    <FSB$C_BLN+RAB$C_BLN+RAB$L_FNA>(R6)
      03  FB 0361 839      CALLS    #3,G^PASSIOERROR
      0000 0368 840      .PSECT   _PASS$CODE,      PIC,EXE,SHR,NOWRT
      0368 841      ;
      0368 842      ; *****
      0368 843      ; *
      0368 844      ; * PASSWRITEHEX *
      0368 845      ; *
      0368 846      ; *****
      0368 847      ;
      0368 848      ; Writes out a longword in hexadecimal form. Leading zeros are printed
      0368 849      ; up to eight places.
      0368 850      ;
      0368 851      ; Argument offsets
      0368 852      ;
      0368 853      ;
00000004 0368 854      AP                ; number of arguments (4)
      FSB_DISP = 04      ; FSB address

```

```

00000008 0368 855 VAL_DISP = 08 ; value to be printed
0000000C 0368 856 FLD_DISP = 12 ; field width (by value)
00000010 0368 857 NOT_DISP = 16 ; (not used)
0368 858
0368 859 : Other constants
0368 860
00000008 0368 861 HMAX = 8 ; maximum zero fill field width
00000010 0368 862 OVERFLOWSIZE = 16 ; maximum overflow buffer size
0368 863
00FC 0368 864 .ENTRY PASS$WRITEHEX,^M<R2,R3,R4,R5,R6,R7>
56 04 AC D0 036A 865 MOVL FSB_DISP(AP),R6 ; R6 = address of FSB
56 DD 036E 866 PUSHL R6
000000C7'GF 01 FB 0370 867 CALLS #1,G^PASS$WRITEOK
0C AC D5 0377 868 TSTL FLD_DISP(AP)
65 15 037A 869 BLEQ $40 ; exit if field width <= 0
037C 870
037C 871 : Make room for descriptor on stack
037C 872
5E 08 C2 037C 873 SUBL2 #DSC$C_S_BLN,SP
52 5E D0 037F 874 MOVL SP,R2 ; R2 = address of descriptor
62 0C AC B0 0382 875 MOVW FLD_DISP(AP),DSC$W_LENGTH(R2); store length
04 A2 66 D0 0386 876 MOVL (R6),DSC$A_POINTER(R2) ; store buffer address
52 DD 038A 877 PUSHL R2
08 AC DD 038C 878 PUSHL VAL_DISP(AP)
00000000'GF 02 FB 038F 879 CALLS #2,G^FOR$CNV_OUT_Z
66 0C AC C0 0396 880 ADDL2 FLD_DISP(AP),(R6)
04 B2 2A 91 039A 881 CMPB #STAR,@DSC$A_POINTER(R2); test overflow
03 13 039E 882 BEQL $25
0035 31 03A0 883 BRW $35
03A3 884 $25:
66 0C AC C2 03A3 885 SUBL2 FLD_DISP(AP),(R6) ; restore pointer
62 10 B0 03A7 886 MOVW #OVERFLOWSIZE,DSC$W_LENGTH(R2)
5E 10 C2 03AA 887 SUBL2 #OVERFLOWSIZE,SP
57 5E D0 03AD 888 MOVL SP,R7
04 A2 57 D0 03B0 889 MOVL R7,DSC$A_POINTER(R2)
52 DD 03B4 890 PUSHL R2
08 AC DD 03B6 891 PUSHL VAL_DISP(AP)
00000000'GF 02 FB 03B9 892 CALLS #2,G^FOR$CNV_OUT_Z
54 67 10 20 3B 03C0 893 SKPC #SPACE,#OVERFLOWSIZE,(R7); skip blanks
54 50 0C AC C3 03C4 894 SUBL3 FLD_DISP(AP),R0,R4
51 54 C0 03C9 895 ADDL2 R4,R1
00 B6 61 0C AC 28 03CC 896 MOVW3 FLD_DISP(AP),(R1),@(R6) ; deposit string
66 0C AC C0 03D2 897 ADDL2 FLD_DISP(AP),(R6) ; fix up pointer
09 11 03D6 898 ERB $40
54 08 CE 03D8 899 $35:
000004C6'EF 16 03DB 900 MNEGL #HMAX,R4
04 03E1 901 JSB ZERO_FILL_R3
03E2 902 $40:
03E2 903
03E2 904
000003E2 905 .PSECT _PASS$CODE, PIC,EXE,SHR,NOWRT
03E2 906
03E2 907
03E2 908
03E2 909
03E2 910
03E2 911
*****
*
* PASS$WRITEHEXD
*
*****

```

```

03E2 912 :
03E2 913 : Write out a double precision value (quadword) in hexadecimal form.
03E2 914 : Leading zeros up to 16 places are printed
03E2 915 :
03E2 916 : Argument offsets
03E2 917 :
03E2 918 :
00000004 03E2 919 : AP ; number of arguments (4)
00000008 03E2 920 : FSB_DISP = 04 ; FSB address
0000000C 03E2 921 : VAR_DISP = 08 ; value address
00000010 03E2 922 : FLD_DISP = 12 ; field width by value
03E2 923 : NOT_DISP = 16 ; (not used)
03E2 924 :
03E2 925 : Other constants
00000008 03E2 926 : HMAX = 8 ; maximum field for leading zeros
0000 03E2 927 : .ENTRY PASS$WRITEHEXD,^M<>
50 0C AC 08 C3 03E4 928 : SUBL3 #HMAX,FLD_DISP(AP),R0 ; R0 = field width low bytes
06 14 03E9 929 : BGTR 110$
50 0C AC D0 03EB 930 : MOVL FLD_DISP(AP),R0
16 11 03EF 931 : BRB 111$
03F1 932 :
03F1 933 : Print low order longword
03F1 934 :
03F1 935 : 110$:
50 04 08 AC DD 03F1 936 : PUSHL #0
50 04 08 AC DD 03F3 937 : PUSHL R0
06 14 03F5 938 : ADDL3 VAR_DISP(AP),#4,R0
FF64 CF 04 AC DD 03FA 939 : PUSHL (R0) ; low order longword
50 08 D0 03FC 940 : PUSHL FSB_DISP(AP)
04 04 FB 03FF 941 : CALLS #4,PASS$WRITEHEX
50 08 D0 0404 942 : MOVL #HMAX,R0 ; field width high bytes
0407 943 :
0407 944 : Print R0 digits of high order longword
0407 945 :
0407 946 : 111$:
00 DD 0407 947 : PUSHL #0
50 DD 0409 948 : PUSHL R0
08 BC DD 040B 949 : PUSHL @VAR_DISP(AP)
04 AC DD 040E 950 : PUSHL FSB_DISP(AP)
FF52 CF 04 FB 0411 951 : CALLS #4,PASS$WRITEHEX
04 0416 952 : RET
0417 953 :
0417 954 :
0000 0417 955 : .PSECT _PASS$CODE, PIC,EXE,SHR,NOWRT
0417 956 :
0417 957 : *****
0417 958 : * PASS$WRITEOCT *
0417 959 : *
0417 960 : *****
0417 961 :
0417 962 :
0417 963 : Argument offsets
0417 964 :
0417 965 :
00000004 0417 966 : AP ; number of arguments (4)
00000008 0417 967 : FSB_DISP = 04 ; FSB address
0000000C 0417 968 : VAL_DISP = 08 ; value to be printed
: FLD_DISP = 12 ; field width

```

```

00000010 0417 969      NOT_DISP = 16                ; (not used)
          0417 970      ;
          0417 971      ; Other constants
          0417 972      ;
0000000B 0417 973      OMAX = 11                ; maximum field for leading zeros
00000014 0417 974      OVERFLOWSIZE = 20        ; overflow buffer size
          0417 975      ;
          00FC 0417 976      .ENTRY PASS$WRITEOCT, ^M<R2,R3,R4,R5,R6,R7>
56 04 AC D0 0419 977      MOVL FSB_DISP(AP),R6      ; R6 = address of FSB
          56 DD 041D 978      PUSHL R6
00000000'GF 01 FB 041F 979      CALLS #1,G^PASS$WRITEOK
          OC AC D5 0426 980      TSTL FLD_DISP(AP)
          65 15 0429 981      BLEQ $43              ; exit if field width <= 0
          042B 982      ;
          042B 983      ; Make room for descriptor on stack
          042B 984      ;
          5E 08 C2 042B 985      SUBL2 #DSC$C_S_BLN,SP
          52 5E D0 042E 986      MOVL SP,R2
62 0C AC B0 0431 987      MOVW FLD_DISP(AP),DSC$W_LENGTH(R2); store length
04 A2 66 D0 0435 988      MOVL (R6),DSC$A_POINTER(R2) ; store buffer address
          52 DD 0439 989      PUSHL R2
          08 AC DD 043B 990      PUSHL VAL_DISP(AP)
00000000'GF 02 FB 043E 991      CALLS #2,G^FOR$CNV_OUT_0
          66 0C AC C0 0445 992      ADDL2 FLD_DISP(AP),(R6)
04 B2 2A 91 0449 993      CMPB #STAR,@DSC$A_POINTER(R2); test overflow
          03 13 044D 994      BEQL $55
          0035 31 044F 995      BRW $65
          0452 996      $55:
66 0C AC C2 0452 997      SUBL2 FLD_DISP(AP),(R6) ; restore pointer
          62 14 B0 0456 998      MOVW #OVERFLOWSIZE,DSC$W_LENGTH(R2)
          5E 14 C2 0459 999      SUBL2 #OVERFLOWSIZE,SP
04 A2 57 5E D0 045C 1000     MOVL SP,R7
          52 DD 0463 1002     PUSHL R2
          08 AC DD 0465 1003     PUSHL VAL_DISP(AP)
00000000'GF 02 FB 0468 1004     CALLS #2,G^FOR$CNV_OUT_0
          67 14 20 3B 046F 1005     SKPC #SPACE,#OVERFLOWSIZE,(R7); skip blanks
54 50 0C AC C3 0473 1006     SUBL3 FLD_DISP(AP),R0,R4
          51 54 C0 0478 1007     ADDL2 R4,R1
00 B6 61 0C AC 28 047B 1008     MOVCL3 FLD_DISP(AP),(R1),@(R6) ; deposit string
          66 0C AC C0 0481 1009     ADDL2 FLD_DISP(AP),(R6) ; fix up pointer
          09 11 0485 1010     BRB $43
          0487 1011     $65:
          54 08 CE 0487 1012     MNEGL #OMAX,R4
000004C6'EF 16 048A 1013     JSB ZERO_FILL_R3
          04 0490 1014     $43:
          0491 1015     ;
          0491 1016     ;
00000491 0491 1017     .PSECT _PASS$CODE,          PIC,EXE,SHR,NOWRT
          0491 1018     ;
          0491 1019     ; *****
          0491 1020     ; *
          0491 1021     ; * PASS$WRITEOCT *
          0491 1022     ; *
          0491 1023     ; *****
          0491 1024     ;
          0491 1025     ; Write out a double precision value (quadword) in octal format.

```

```

0491 1026 : Leading zeros up to twenty-two places are printed.
0491 1027 :
0491 1028 : Argument offsets
0491 1029 :
0491 1030 : AP ; number of arguments (4)
00000004 0491 1031 : FSB_DISP = 04 ; FSB address
00000008 0491 1032 : VAR_DISP = 08 ; value address
0000000C 0491 1033 : FLD_DISP = 12 ; field width by value
00000010 0491 1034 : NOT_DISP = 16 ; (not used)
0491 1035 :
0491 1036 : Other constants
0491 1037 :
0000000B 0491 1038 : OMAX = 11 ; maximum field for leading zeros
0491 1039 :
0000 0491 1040 : .ENTRY PASSWRITEOCTD,^M<>
50 0C AC 0B C3 0493 1041 : SUBL3 #OMAX,FLD_DISP(AP),R0 ; R0 = field width low bytes
06 14 0498 1042 : BGTR 110$
50 0C AC D0 049A 1043 : MOVL FLD_DISP(AP),R0
16 11 049F 1044 : BRB 111$
04A0 1045 :
04A0 1046 : Print low order longword
04A0 1047 :
04A0 1048 : 110$:
00 DD 04A0 1049 : PUSHL #0
50 DD 04A2 1050 : PUSHL R0
50 04 08 AC C1 04A4 1051 : ADDL3 VAR_DISP(AP),#4,R0
60 DD 04A9 1052 : PUSHL (R0) ; low order longword
04 AC DD 04AB 1053 : PUSHL FSB_DISP(AP)
FF64 CF 04 FB 04AE 1054 : CALLS #4,PASSWRITEOCT
50 0B D0 04B3 1055 : MOVL #OMAX,R0 ; field width high bytes
04B6 1056 :
04B6 1057 : Print R0 digits of high order longword
04B6 1058 :
04B6 1059 : 111$:
00 DD 04B6 1060 : PUSHL #0
50 DD 04B8 1061 : PUSHL R0
08 BC DD 04BA 1062 : PUSHL @VAR_DISP(AP)
04 AC DD 04BD 1063 : PUSHL FSB_DISP(AP)
FF52 CF 04 FB 04C0 1064 : CALLS #4,PASSWRITEOCT
04 04C5 1065 : RET
04C6 1066 :
04C6 1067 :
0000 04C6 1068 : .PSECT _PASSCODE, PIC,EXE,SHR,NOWRT
04C6 1069 :
04C6 1070 : *****
04C6 1071 : * ZERO_FILL_R3 *
04C6 1072 : *
04C6 1073 : *****
04C6 1074 :
04C6 1075 :
04C6 1076 : JSB routine to zero-fill octal and hex output
04C6 1077 :
04C6 1078 :
04C6 1079 : ZERO_FILL_R3: ; entry point
52 0C AC CE 04C6 1080 : MREGL FLD_DISP(AP),R2 ; get length
54 52 D1 04CA 1081 : CMPL R2,R4
03 18 04CD 1082 : BGEQ $30

```



04 1E A3	00	EO	051A	1140	BBS	#FAB\$V FTN,FAB\$B_RAT(R3),20\$; check for carriage control
	0C	DD	051F	1141	PUSHL	#FORMFEED ; not FORTRAN
	02	11	0521	1142	BRB	30\$
			0523	1143		
	31	DD	0523	1144	PUSHL	#NONE ; FORTRAN
			0525	1145		
	52	DD	0525	1146	PUSHL	R2 ; FSB address
FB2F CF	04	FB	0527	1147	CALLS	#4,PASSWRITECHAR
	52	DD	052C	1148	PUSHL	R2
00000000'EF	01	FB	052E	1149	CALLS	#1,PASSWRITELN ; terminate line
		04	0535	1150	RET	; return
			0536	1151		
			0536	1152		
			0536	1153		
			0536	1154		
					.END	

```

$$TMP1      = 00000001
$$TMP2      = 00000057
$10         = 000004D6 R      02
$20         = 000004E8 R      02
$25         = 000003A3 R      02
$30         = 000004D2 R      02
$35         = 000003D8 R      02
$40         = 000003E1 R      02
$43         = 00000490 R      02
$55         = 00000452 R      02
$65         = 00000487 R      02
CHR_DISP    = 00000008
DIG_DISP    = 00000010
DOB_DISP    = 00000008
DSCSA_POINTER = 00000004
DSCSC_S_BLN = 00000008
DSCSW_LENGTH = 00000000
EMIN        = 00000008
FABSB_FNS   = 00000034
FABSB_RAT   = 0000001E
FABSL_FNA   = 0000002C
FABSV_FTN   = 00000000
FLD_DISP    = 0000000C
FMAX        = 0000002A
FMIN        = 00000003
FOR$CNV_OUT_D ***** X 00
FOR$CNV_OUT_E ***** X 00
FOR$CNV_OUT_F ***** X 00
FOR$CNV_OUT_I ***** X 00
FOR$CNV_OUT_O ***** X 00
FOR$CNV_OUT_Z ***** X 00
FORMFEED    = 0000000C
FSB$C_BLN   = 00000018
FSB$S_LIM   = 0000000C
FSB$S_LST   = 00000008
FSB_DISP    = 00000004
HMAX        = 00000008
IMAX        = 00000014
IMINN       = 00000002
IMINP       = 00000001
INT_DISP    = 00000008
LEN_DISP    = 00000010
MAX_DISP    = 00000014
NAMELEN     = 00000020
NAM_DISP    = 00000010
NEWENT      = 0000000D R      02
NOT_DISP    = 00000010
OMAX        = 00000008
ONE         = 00000031
OVERFLOWSIZE = 00000014
PASSBUFFEROVER ***** X 00
PASSIOERROR ***** X 00
PASSLINELIMIT = 000004E9 RG 02
PASSPAGE    = 000004F5 RG 02
PASSPUTBIN  = 00000000 RG 02
PASSPUTBINARY = 00000008 RG 02
PASSPUTTXT  = 0000003A RG 02
    
```

```

PASSWREAL   00000219 R      02
PASSWREALF  000002A7 R      02
PASSWRITECHAR 0000005B RG 02
PASSWRITEDOUBE 000001F6 RG 02
PASSWRITEDOUBF 0000028A RG 02
PASSWRITEHEX  00000368 RG 02
PASSWRITEHEXD 000003E2 RG 02
PASSWRITEINT  0000014C RG 02
PASSWRITELN   ***** X 00
PASSWRITEOCT  00000417 RG 02
PASSWRITEOCTD 00000491 RG 02
PASSWRITEOK   ***** X 00
PASSWRITEREAL 00000209 RG 02
PASSWRITEREALF 0000029A RG 02
PASSWRITESCAL 000000F2 RG 02
PASSWRITESTR  00000098 RG 02
RAB$B_RAC     = 0000001E
RAB$C_BLN     = 00000044
RAB$C_SEQ     = 00000000
RAB$S_RBF     = 00000028
RAB$S_RPF     = 00000004
RAB$M_TPT     = 00000002
REL_DISP      = 00000008
SCA_DISP      = 00000008
SPACE         = 00000020
STAR         = 0C00002A
STR_DISP      = 00000008
SYSPUT        ***** G 02
VAL_DISP      = 00000008
VAR_DISP      = 00000008
ZERO         = 00000030
ZERO_FILL_R3 000004C6 R      02
    
```



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

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 ( 0.)	00 ( 0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$AB\$\$	00000000 ( 0.)	01 ( 1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
_PAS\$CODE	00000536 ( 1334.)	02 ( 2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

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

Phase	Page faults	CPU Time	Elapsed Time
Initialization	34	00:00:00.08	00:00:00.50
Command processing	135	00:00:00.45	00:00:01.24
Pass 1	252	00:00:08.14	00:00:16.17
Symbol table sort	0	00:00:00.87	00:00:00.91
Pass 2	194	00:00:02.86	00:00:04.53
Symbol table output	12	00:00:00.07	00:00:00.09
Psect synopsis output	3	00:00:00.03	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	633	00:00:12.50	00:00:23.47

The working set limit was 1500 pages.  
49931 bytes (98 pages) of virtual memory were used to buffer the intermediate code.  
There were 40 pages of symbol table space allocated to hold 726 non-local and 40 local symbols.  
1154 source lines were read in Pass 1, producing 61 object records in Pass 2.  
14 pages of virtual memory were used to define 12 macros.

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

Macro library name	Macros defined
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	9

772 GETS were required to define 9 macros.

There were no errors, warnings or information messages.

MACRO/DISABLE=TRACE/LIS=LIS\$:PASIO3/OBJ=OBJ\$:PASIO3 MSRC\$:PASIO3/UPDATE=(ENH\$:PASIO3)



