


```

FFFFFFFFF  000000  RRRRRRRR  IIIIII  NN  NN  IIIIII  DDDDDDDD  EEEEEEEEE  SSSSSSSS
FFFFFFFFF  000000  RRRRRRRR  IIIIII  NN  NN  IIIIII  DDDDDDDD  EEEEEEEEE  SSSSSSSS
FF          00      00  RR      RR  II      II      II      DD      DD  EE      SS
FF          00      00  RR      RR  II      II      II      DD      DD  EE      SS
FF          00      00  RR      RR  II      II      II      DD      DD  EE      SS
FF          00      00  RR      RR  II      II      II      DD      DD  EE      SS
FFFFFFFFF  00      00  RRRRRRRR  II      II      II      DD      DD  EEEEEEE  SSSSSS
FFFFFFFFF  00      00  RRRRRRRR  II      II      II      DD      DD  EEEEEEE  SSSSSS
FF          00      00  RR  RR  II      II      II      DD      DD  EE      SS
FF          00      00  RR  RR  II      II      II      DD      DD  EE      SS
FF          00      00  RR  RR  II      II      II      DD      DD  EE      SS
FF          00      00  RR  RR  II      II      II      DD      DD  EE      SS
FF          000000  RR      RR  IIIIII  NN  NN  IIIIII  DDDDDDDD  EEEEEEEEE  SSSSSSSS
FF          000000  RR      RR  IIIIII  NN  NN  IIIIII  DDDDDDDD  EEEEEEEEE  SSSSSSSS

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
LLLLLLLLLL  IIIIII  SSSSSSSS
LLLLLLLLLL  IIIIII  SSSSSSSS

```

B
C
O
D
E
F
I
L
E
I
D
F
O
R
I
N
I
D
E
S
N
1
5

(2) 56
(5) 151
(8) 249
(11) 352

DECLARATIONS
FOR\$INI_DES1_R2
FOR\$INI_DES2_R3
FOR\$INI_DESC_R6

! One dimension array desc. init.
! Two dimension array desc. init.
! N dimension array desc. init.


```
0000 1 .TITLE FOR$INI_DES ; FORTRAN array descriptor initialization
0000 2 .IDENT /1-021/ ; File: FORINIDES.MAR Edit: SBL1021
0000 3
0000 4 :
0000 5 :*****
0000 6 :*
0000 7 :* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 :* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 :* ALL RIGHTS RESERVED.
0000 10 :*
0000 11 :* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 :* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 :* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 :* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 :* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 :* TRANSFERRED.
0000 17 :*
0000 18 :* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 :* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 :* CORPORATION.
0000 21 :*
0000 22 :* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 :* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24 :*
0000 25 :*
0000 26 :*****
0000 27 :
0000 28 :++
0000 29 : FACILITY: FORTRAN support library
0000 30 :
0000 31 : ABSTRACT:
0000 32 :
0000 33 : FORTRAN support routines to initialize variable
0000 34 : dimensioned array descriptors at FORTRAN procedure entry time.
0000 35 : For speed the important special cases of 1 and 2 dimensions
0000 36 : are provided for as separate routines.
0000 37 :
0000 38 : ENVIRONMENT: Any access mode--usually user mode, AST re-entrant
0000 39 :
0000 40 : AUTHOR: Thomas N. Hastings, CREATION DATE: 1-Apr-77
0000 41 :
0000 42 : MODIFIED BY:
0000 43 : Thomas N. Hastings, 2-Apr-77: VERSION 01
0000 44 : 01 - original version
0000 45 : 01-15 - Call LIB$STOP directly. TNH 20-Dec-77
0000 46 : 1-016 - Update copyright notice. JBS 16-NOV-78
0000 47 : 1-017 - Remove FOR_ADJARRDIM - not needed. JBS 16-DEC-78
0000 48 : 1-018 - Add "" to the PSECT directives. JBS 22-DEC-78
0000 49 : 1-019 - Error FOR$ ADJARRDIM now becomes continuable. SBL 25-May-1979
0000 50 : 1-020 - Add index multiply for datatype DC (D COMPLEX*16). The compiler
0000 51 : doesn't use octaword context addressing for DC, but can for GC.
0000 52 : SBL 6-July-1979
0000 53 : 1-021 - Use general mode addressing. SBL 30-Nov-1981
0000 54 :--
```



```
0000 56 .SBTTL DECLARATIONS
0000 57
0000 58 :
0000 59 : INCLUDE FILES:
0000 60 :
0000 61 :
0000 62 : EXTERNAL SYMBOLS:
0000 63 :
0000 64 .DSABL GBL ; Declare all externals
0000 65 .EXTRN FOR$ ADJARRDIM ; 32-bit condition value for signal
0000 66 .EXTRN LIB$SIGNAL ; SIGNAL error
0000 67
0000 68 :
0000 69 : MACROS:
0000 70 :
0000 71 $DSCDEF ; define string descriptor symbols
0000 72 :
0000 73 : EQUATED SYMBOLS:
0000 74 :
0000 75
FFFFF8 0000 76 DSC$L_LOW_INDX = DSC$W_LENGTH-8 ; place to store lowest index
0000 77 ; for use by FORTRAN with INDEX instr.
FFFFFC 0000 78 DSC$L_HIGH_INDX = DSC$L_LOW_INDX+4 ; place to store highest index
0000 79 ; for use by FORTRAN with INDEX instr.
0000 80 ; Note: the above are defined here, rather than in SRMDEF, since
0000 81 ; they are not part of the calling standard between modules.
0000 82 ; They are only a convention between the Common OTS and a single
0000 83 ; FORTRAN module.
0000 84
0000 85 :
0000 86 : OWN STORAGE:
0000 87 :
0000 88 none
0000 89 :
0000 90 :
0000 91 : PSECT DECLARATIONS:
0000 92 :
0000 93
00000000 94 .PSECT _FOR$CODE PIC,SHR,LONG,EXE,NOWRT
0000 95 ; program section for FOR$ code
0000 96
```



```
0000 98
0000 99 : C.10.5 Array Descriptor (DSC$K_CLASS_A)
0000 100
0000 101 : An array descriptor consists of 3 contiguous blocks. The first block
0000 102 : contains the descriptor prototype information and is part of every
0000 103 : array descriptor. The second and third blocks are optional. If the
0000 104 : third block is present then so is the second.
0000 105
0000 106 : A complete array descriptor has the form:
0000 107
0000 108 :
0000 109 :
0000 110 :
0000 111 :
0000 112 :
0000 113 :
0000 114 :
0000 115 :
0000 116 :
0000 117 :
0000 118 :
0000 119 :
0000 120 :
0000 121 :
0000 122 :
0000 123 :
0000 124 :
0000 125 :
0000 126 :
0000 127 :
0000 128 :
0000 129 :
0000 130 :
0000 131 :
0000 132 :
0000 133 :
0000 134 :
0000 135 :
0000 136 :
0000 137 :
0000 138 :
0000 139 :
0000 140 :
0000 141 :
0000 142 :
0000 143 :
0000 144 : PSECT DECLARATIONS:
0000 145 :
0000 146 :
00000000 147 : .PSECT _FOR$CODE PIC,SHR,LONG,EXE,NOWRT
0000 148 : ; program section for FOR$ code
0000 149
```

4	DTYPE	LENGTH
POINTER		
DIMCT	AFLAGS	Reserved
ARSIZE		

:Descriptor

Block 1 - Prototype

A0
M1
...
M(n-1)
Mn

Block 2 - Multipliers

L1
U1
...
Ln
Un

Block 3 - Bounds


```
0000 151 .SBTTL FOR$INI_DES1_R2 ! One dimension array desc. init.
0000 152
0000 153 :++
0000 154 : FUNCTIONAL DESCRIPTION:
0000 155 :
0000 156 : Initialize one-dimension array descriptor entries based on the lower
0000 157 : and upper subscripts limits. The LOW_INDX and HIGH_INDX
0000 158 : entries are FORTRAN specific for in-line subscript checking
0000 159 : using the INDEX instruction and are outside the standardized
0000 160 : array descriptor passed between modules.
0000 161 :
0000 162 : CALLING SEQUENCE:
0000 163 :
0000 164 : MOVA address_of_descriptor, R0
0000 165 : JSB FOR$INI_DES1_R2
0000 166 :
0000 167 : INPUT PARAMETERS:
0000 168 :
0000 169 : R0 = address of array descriptor
0000 170 :
0000 171 : IMPLICIT INPUTS:
0000 172 :
0000 173 : length of each data element - LENGTH
0000 174 : data type of each element - DTYPE
0000 175 : address of first allocated entry - POINTER
0000 176 : lower subscript limit - L1
0000 177 : upper subscript limit - U1
0000 178 :
0000 179 : OUTPUT PARAMETERS:
0000 180 :
0000 181 : NONE
0000 182 :
0000 183 : IMPLICIT OUTPUTS:
0000 184 :
0000 185 : multiplier - M1
0000 186 : array size in bytes - ARSIZE
0000 187 : highest linear index - HIGH_INDX
0000 188 : lowest linear index - LOW_INDX
0000 189 : address of A(0) - A0
0000 190 :
0000 191 : COMPLETION CODES:
0000 192 :
0000 193 : NONE
0000 194 :
0000 195 : SIDE EFFECTS:
0000 196 :
0000 197 : NONE
0000 198 :
0000 199 :--
```



```
0000 201
0000 202 ; Register usage:
0000 203 :
0000 204 : R0 = address of descriptor
0000 205 : R1 = length of data element in bytes (if string, length of string)
0000 206 : R2 = lower limit on subscript
0000 207 :
00000018 0000 208 L_L1 = DSC$L_M1+4 ; lower limit (L1) for one-dim
0000001C 0000 209 L_U1 = L_L1+4 ; upper limit (U1) for one-dim
0000 210
0000 211
0000 212 FOR$INI_DES1_R2::
51 60 3C 0000 213 -MOVZWL DSC$W_LENGTH(R0), R1 ; R1 = length of data element
0003 214
52 1C A0 18 A0 C3 0003 215 SUBL3 L_L1(R0), L_U1(R0), R2 ; R2 = U1 - L1
52 D6 0009 216 INCL R2 ; R2 = M1 = U1 - L1 + 1
2F 15 000B 217 BLEQ ERROR ; error if 0 or negative
14 A0 52 D0 000D 218 MOVL R2, DSC$L_M1(R0) ; store M1
0C A0 51 52 C5 0011 219
0016 220 MULL3 R2, R1, DSC$L_ARSIZE(R0) ; store array size in bytes
0016 221
F8 A0 18 A0 7D 0016 222
001B 223 MOVQ L_L1(R0), DSC$L_LOW_INDX(R0) ; store low and high linear index
001B 224 ; i.e., L1 and U1
52 18 A0 51 C5 001B 225
10 A0 04 A0 52 C3 0020 226 MULL3 R1, L_L1(R0), R2 ; R2 = low * length
0026 227 SUBL3 R2, DSC$A_POINTER(R0), DSC$A_A0(R0) ; A0 = POINTER - low * length
0E 02 A0 91 0026 228
07 13 002A 229 CMPB DSC$B_DTYPE(R0), #DSC$K_DTYPE_T ; is this CHARACTER?
0D 02 A0 91 002C 230 BEQL 10$ ; branch if yes
01 13 0030 231 CMPB DSC$B_DTYPE(R0), #DSC$K_DTYPE_DC ; is this D Complex?
05 0032 232 BEQL 10$ ; branch if yes
0033 233 RSB ; return if not
F8 A0 52 D0 0033 234
FC A0 51 C4 0037 235 10$: MOVL R2, DSC$L_LOW_INDX(R0) ; store scaled low index by length
05 003B 236 MULL R1, DSC$L_HIGH_INDX(R0) ; scale high index by length
003C 237 RSB ; return
003C 238
003C 239 ;+
003C 240 ; Here on dimensioning error.
003C 241 ; SIGNAL error FOR$_ADJARRDIM (93='ADJUSTABLE ARRAY DIMENSION ERROR')
003C 242 ; -
003C 243
00000000'8F DD 003C 244 ERROR: PUSHL #FOR$_ADJARRDIM ; FORTRAN error #
00000000'GF 01 FB 0042 245 CALLS #1, G*LIB$SIGNAL ; SIGNAL error
05 0049 246 RSB ; In case it comes back
004A 247
```



```

004A 249 .SBTTL FOR$INI_DES2_R3 ! Two dimension array desc. init.
004A 250
004A 251 :++
004A 252 : FUNCTIONAL DESCRIPTION:
004A 253 :
004A 254 : Initialize two-dimension array descriptor entries based on the lower
004A 255 : and upper subscripts limits and the address of the
004A 256 : first allocated entry. The LOW_INDX and HIGH_INDX
004A 257 : entries are FORTRAN specific for in-line subscript checking
004A 258 : using the INDEX instruction and are outside the standardized
004A 259 : array descriptor passed between modules.
004A 260 :
004A 261 : CALLING SEQUENCE:
004A 262 :
004A 263 : MOVA address_of_descriptor, R0
004A 264 : JSB FOR$INI_DES2_R3
004A 265 :
004A 266 : INPUT PARAMETERS:
004A 267 :
004A 268 : R0 = address of array descriptor
004A 269 :
004A 270 : IMPLICIT INPUTS:
004A 271 :
004A 272 : length of each data element - LENGTH
004A 273 : data type of each element - DTYPE
004A 274 : address of first allocated entry - POINTER
004A 275 : lower 1st subscript limit - L1
004A 276 : upper 1st subscript limit - U1
004A 277 : lower 2nd subscript limit - L2
004A 278 : upper 2nd subscript limit - U2
004A 279 :
004A 280 : OUTPUT PARAMETERS:
004A 281 :
004A 282 : NONE
004A 283 :
004A 284 : IMPLICIT OUTPUTS:
004A 285 :
004A 286 : 1st dimension multiplier - M1
004A 287 : 2nd dimension multiplier - M2
004A 288 : array size in bytes - ARSIZE
004A 289 : highest linear index - HIGH_INDX
004A 290 : lowest linear index - LOW_INDX
004A 291 : address of A(0) - A0
004A 292 :
004A 293 : COMPLETION CODES:
004A 294 :
004A 295 : NONE
004A 296 :
004A 297 : SIDE EFFECTS:
004A 298 :
004A 299 : NONE
004A 300 :
004A 301 :--

```



```
004A 303
004A 304 ; Register usage:
004A 305
004A 306 R0 = address of descriptor
004A 307 R1 = length of data element in bytes (if string, length of string)
004A 308 R2 = lower limit on subscript
004A 309 R3 = upper limit on subscript
004A 310
0000001C 004A 311 L_L1 = DSC$M2+4 ; lower limit 1 for 2-dim
00000020 004A 312 L_U1 = L_L1+4 ; upper limit 1 for 2-dim
00000024 004A 313 L_L2 = L_L1+8 ; lower limit 2 for 2-dim
00000028 004A 314 L_U2 = L_L1+12 ; upper limit 2 for 2-dim
004A 315
004A 316
004A 317 FOR$INI_DES2_R3::
51 60 3C 004A 318 MOVZWL DSC$W_LENGTH(R0), R1 ; R1 = length of data element
004D 319
52 20 A0 1C A0 C3 004D 320
52 52 D6 004D 321
52 E5 15 0053 322
53 28 A0 24 A0 C3 0055 323
53 53 D6 005D 324
53 DB 15 005F 325
14 A0 52 7D 0061 326
0065 327
0C A0 53 52 C4 0065 328
0C A0 53 51 C5 0068 329
006D 330
006D 331
53 52 28 A0 C5 006D 332
53 53 20 A0 C0 0072 333
0076 334
52 24 A0 C4 0076 335
52 1C A0 C0 007A 336
F8 A0 52 7D 007E 337
0082 338
10 A0 04 A0 52 C4 0082 339
0085 340
008B 341
0E 02 A0 91 008B 342
07 13 008F 343
0D 02 A0 91 0091 344
01 13 0095 345
05 0097 346
0098 347
F8 53 51 C4 0098 348 10$: MULL R1, R3 ; R3 = scaled high by length
F8 A0 52 7D 0098 349 ; store scaled low and high index
05 009F 350 RSB ; return

SUBL3 L_L1(R0), L_U1(R0), R2 ; R2 = U1 - L1
INCL R2 ; R2 = M1 = U1 - L1 + 1
BLEQ ERROR ; error if 0 or negative
SUBL3 L_L2(R0), L_U2(R0), R3 ; R3 = U2 - L2
INCL R3 ; R3 = M2 = U2 - L2 + 1
BLEQ ERROR ; error if multiplier is 0 or negative
MOVQ R2, DSC$M1(R0) ; store M1 and M2

MULL R2, R3 ; R3 = M1 * M2
MULL3 R1, R3, DSC$M_ARSIZE(R0) ; store array size in bytes
; = data type length * M1 * M2

MULL3 L_U2(R0), R2, R3 ; R3 = M1 * U2
ADDL L_U1(R0), R3 ; R3 = high index = U2*M1 + U1

MULL L_L2(R0), R2 ; R2 = M1*L2
ADDL L_L1(R0), R2 ; R2 = low index = L2*M1 + L1
MOVQ R2, DSC$LOW_INDX(R0) ; store low and high linear index

MULL R1, R2 ; R2 = low * length
SUBL3 R2, DSC$A_POINTER(R0), DSC$A_A0(R0) ; A0 = POINTER - low * length

CMPB DSC$B_DTYPE(R0), #DSC$K_DTYPE_T ; is this CHARACTER?
BEQL 10$ ; branch if yes
CMPB DSC$B_DTYPE(R0), #DSC$K_DTYPE_DC ; is this D Complex?
BEQL 10$ ; branch if yes
RSB ; return if not
```



```
00A0 352 .SBTTL FOR$INI_DESC_R6 ! N dimension array desc. init.
00A0 353
00A0 354
00A0 355 :++
00A0 356 : FUNCTIONAL DESCRIPTION:
00A0 357 : Initialize n-dimension array descriptor entries base on the lower
00A0 358 : and upper subscripts limits and the address of the
00A0 359 : first allocated entry. The LOW_INDX and HIGH_INDX
00A0 360 : entries are FORTRAN specific for in-line subscript checking
00A0 361 : using the INDEX instruction and are outside the standardized
00A0 362 : array descriptor passed between modules.
00A0 363
00A0 364 : CALLING SEQUENCE:
00A0 365
00A0 366 : MOVA address_of_descriptor, R0
00A0 367 : JSB FOR$INI_DESC_R3
00A0 368
00A0 369 : INPUT PARAMETERS:
00A0 370
00A0 371 : R0 = address of array descriptor
00A0 372
00A0 373 : IMPLICIT INPUTS:
00A0 374
00A0 375 : length of each data element - LENGTH
00A0 376 : data type of each element - DTYPE
00A0 377 : address of first allocated entry - POINTER
00A0 378 : lower 1st subscript limit - L1
00A0 379 : upper 1st subscript limit - U1
00A0 380 : lower 2nd subscript limit - L2
00A0 381 : upper 2nd subscript limit - U2
00A0 382
00A0 383 : lower nth subscript limit - Ln
00A0 384 : upper nth subscript limit - Un
00A0 385
00A0 386 : OUTPUT PARAMETERS:
00A0 387
00A0 388 : NONE
00A0 389
00A0 390 : IMPLICIT OUTPUTS:
00A0 391
00A0 392 : 1st dimension multiplier - M1
00A0 393
00A0 394 : nth dimension multiplier - Mn
00A0 395 : highest linear index - HIGH_INDX
00A0 396 : lowest linear index - LOW_INDX
00A0 397 : address of A(0) - A0
00A0 398
00A0 399 : COMPLETION CODES:
00A0 400
00A0 401 : NONE
00A0 402
00A0 403 : SIDE EFFECTS:
00A0 404
00A0 405 : NONE
00A0 406
00A0 407 :--
```



```
00A0 409 : Register usage:
00A0 410 :
00A0 411 : R0 = address of descriptor
00A0 412 : R1 = length of data element in bytes (if string, length of string)
00A0 413 : R2 = Mi then partial product * Mi + Li
00A0 414 : R3 = accumulated product of Mi, then partial product * Mi + Ui
00A0 415 : R4 = pointer to multipliers (M1...Mn)
00A0 416 : R5 = pointer to lower/upper subscript limits (L1, U1, ..., Ln, Un)
00A0 417 : R6 = no. of dimensions
00A0 418 :
00A0 419 FOR$INI_DESC_R6::
54 51 60 3C 00A0 420 MOVZWL DSC$W_LENGTH(R0), R1 ; R1 = length of data element
56 14 A0 DE 00A3 421 MOVAL DSC$L_M1(R0), R4 ; R4 = adr. of M1
56 0B A0 9A 00A7 422 MOVZBL DSC$B_DIMCT(R0), R6 ; R6 = dimension count
55 64 46 DE 00AB 423 MOVAL (R4)[R6], R5 ; R5 = adr. of L1
53 01 D0 00AF 424 MOVL #1, R3 ; R3 = initial array size in bytes
00B2 425
52 85 85 C3 00B2 426 10$: SUBL3 (R5)+, (R5)+, R2 ; R2 = Ui - Li
52 82 D6 00B6 427 INCL R2 ; R2 = Mi = Ui - Li + 1
84 52 D0 00B8 428 BLEQ ERROR ; error if 0 or negative
53 52 C4 00BA 429 MOVL R2, (R4)+ ; store Mi
EF 56 F5 00BD 430 MULL R2, R3 ; R3 = accumulated product of Mi
0C A0 53 51 C5 00C0 431 SOBGTR R6, 10$ ; loop for each dimension
00C3 432
00C8 433 MULL3 R1, R3, DSC$L_ARSIZE(R0) ; store array size in bytes
00C8 434 ; = data type length * M1 * M2
54 04 C2 00C8 435
56 0B A0 9A 00CB 436 SUBL #4, R4 ; R4 = address Mn, so skip Mn next loop
52 75 7D 00CF 437 MOVZBL DSC$B_DIMCT(R0), R6 ; R6 = no. of dimensions
OC 11 00D2 438 MOVQ -(R5), R2 ; R2/R3 = Ln/Un
00D4 439 BRB 30$ ; skip Mn
53 74 C4 00D4 440
53 75 C0 00D7 441 20$: MULL -(R4), R3 ; R3 = partial product * Mi
52 64 C4 00DA 442 ADDL -(R5), R3 ; R3 = partial product * Mi + Ui
52 75 C0 00DD 443 MULL (R4), R2 ; R2 = partial product * Mi
F1 56 F5 00DD 444 ADDL -(R5), R2 ; R2 = partial product * Mi + Li
00E0 445 30$: SOBGTR R6, 20$ ; loop no. of dimensions-1
00E3 446
F8 A0 52 7D 00E3 447 MOVQ R2, DSC$L_LOW_INDX(R0) ; store low and high linear index
00E7 448
10 A0 04 A0 52 C3 00E7 449 MULL R1, R2 ; R2 = low * length
00EA 450 SUBL3 R2, DSC$A_POINTER(R0), DSC$A_A0(R0) ; A0 = POINTER - low * length
00F0 451
0E 02 A0 91 00F0 452 CMPB DSC$B_DTYPE(R0), #DSC$K_DTYPE_T ; is this CHARACTER?
07 13 00F4 453 BEQL 40$ ; branch if yes
0D 02 A0 91 00F6 454 CMPB DSC$B_DTYPE(R0), #DSC$K_DTYPE_DC ; is this D Complex?
01 13 00FA 455 BEQL 40$ ; branch if yes
05 00FC 456 RSB ; return if not
00FD 457
F8 A0 53 51 C4 00FD 458 40$: MULL R1, R3 ; R3 = scaled high by length
0100 459 MOVQ R2, DSC$L_LOW_INDX(R0) ; store scaled low and high index
0104 460 RSB ; return
0105 461 .END
```


FOR\$INI_DES
Symbol table

L 16
; FORTRAN array descriptor initializatio 15-SEP-1984 23:53:19 VAX/VMS Macro V04-00 Page 10
6-SEP-1984 10:56:25 [FORRTL.SRC]FORINIDES.MAR;1 (12)

DSC\$A_A0 = 00000010
DSC\$A_POINTER = 00000004
DSC\$B_DIMCT = 0000000B
DSC\$B_DTYPE = 00000002
DSC\$K_DTYPE_DC = 0000000D
DSC\$K_DTYPE_T = 0000000E
DSC\$L_ARSIZE = 0000000C
DSC\$L_HIGH_INDXX = FFFFFFFFC
DSC\$L_LOW_INDXX = FFFFFFFF8
DSC\$L_M1 = 00000014
DSC\$L_M2 = 00000018
DSC\$W_LENGTH = 00000000
ERROR = 0000003C R 02
FOR\$INI_DES1_R2 = 00000000 RG 02
FOR\$INI_DES2_R3 = 0000004A RG 02
FOR\$INI_DESC_R6 = 000000A0 RG 02
FOR\$ ADJARRDIM ***** X 00
LIB\$SIGNAL ***** X 00
L_L1 = 0000001C
L_L2 = 00000024
L_U1 = 00000020
L_U2 = 00000028

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

PSECT name	Allocation	PSECT No.	Attributes															
. ABS .	00000000 (0.)	00 (0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE					
\$ABS\$	00000000 (0.)	01 (1.)	NOPIC	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE					
_FOR\$CODE	00000105 (261.)	02 (2.)	PIC	USR	CON	REL	LCL	SHR	EXE	RD	NOWRT	NOVEC	LONG					

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

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.09	00:00:01.06
Command processing	120	00:00:00.49	00:00:02.93
Pass 1	143	00:00:02.29	00:00:07.16
Symbol table sort	0	00:00:00.18	00:00:00.43
Pass 2	89	00:00:00.99	00:00:03.38
Symbol table output	4	00:00:00.03	00:00:00.03
Psect synopsis output	2	00:00:00.02	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	389	00:00:04.10	00:00:15.03

The working set limit was 1050 pages.
12011 bytes (24 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 143 non-local and 6 local symbols.
461 source lines were read in Pass 1, producing 10 object records in Pass 2.
8 pages of virtual memory were used to define 7 macros.

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

Macro library name

Macros defined

_S255\$DUA28:[SYSLIB]STARLET.MLB;2

4

190 GETS were required to define 4 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=(GLOBAL,TRACEBACK)/LIS=LISS\$:FORINIDES/OBJ=OBJ\$:FORINIDES MSRC\$:FORINIDES/UPDATE=(ENH\$:FORINIDES)

0180 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

