```
LLL                 IIIIIIIII     BBBBBBBBBBBB     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
LLL                 IIIIIIIII     BBBBBBBBBBBB     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
LLL                 IIIIIIIII     BBBBBBBBBBBB     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
LLL                    III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III        BBBBBBBBBBBB     RRRRRRRRRRRR           TTT        LLL
LLL                    III        BBBBBBBBBBBB     RRRRRRRRRRRR           TTT        LLL
LLL                    III        BBBBBBBBBBBB     RRRRRRRRRRRR           TTT        LLL
LLL                    III        BBB         BBB  RRR   RRR              TTT        LLL
LLL                    III        BBB         BBB  RRR   RRR              TTT        LLL
LLL                    III        BBB         BBB  RRR   RRR              TTT        LLL
LLL                    III        BBB         BBB  RRR      RRR           TTT        LLL
LLL                    III        BBB         BBB  RRR      RRR           TTT        LLL
LLL                    III        BBB         BBB  RRR      RRR           TTT        LLL
LLLLLLLLLLLLLLL     IIIIIIIII     BBBBBBBBBBBB     RRR         RRR        TTT        LLLLLLLLLLLLLLL
LLLLLLLLLLLLLLL     IIIIIIIII     BBBBBBBBBBBB     RRR         RRR        TTT        LLLLLLLLLLLLLLL
LLLLLLLLLLLLLLL     IIIIIIIII     BBBBBBBBBBBB     RRR         RRR        TTT        LLLLLLLLLLLLLLL
```

```
LL              IIIIII    BBBBBBBB    PPPPPPPP    KK      KK    AAAAAA    RRRRRRRR      IIIIII    TTTTTTTTTT
LL              IIIIII    BBBBBBBB    PPPPPPPP    KK      KK    AAAAAA    RRRRRRRR      IIIIII    TTTTTTTTTT
LL                II      BB      BB  PP      PP  KK      KK  AA      AA  RR      RR      II          TT
LL                II      BB      BB  PP      PP  KK      KK  AA      AA  RR      RR      II          TT
LL                II      BB      BB  PP      PP  KK    KK    AA      AA  RR      RR      II          TT
LL                II      BB      BB  PP      PP  KK    KK    AA      AA  RR      RR      II          TT
LL                II      BBBBBBBB    PPPPPPPP    KKKKKK      AA      AA  RRRRRRRR        II          TT
LL                II      BBBBBBBB    PPPPPPPP    KKKKKK      AA      AA  RRRRRRRR        II          TT
LL                II      BB      BB  PP          KK    KK    AAAAAAAAAA  RR  RR          II          TT
LL                II      BB      BB  PP          KK    KK    AAAAAAAAAA  RR  RR          II          TT
LL                II      BB      BB  PP          KK      KK  AA      AA  RR    RR        II          TT          ....
LL                II      BB      BB  PP          KK      KK  AA      AA  RR    RR        II          TT          ....
LLLLLLLLLL      IIIIII    BBBBBBBB    PP          KK      KK  AA      AA  RR      RR    IIIIII        TT          ....
LLLLLLLLLL      IIIIII    BBBBBBBB    PP          KK      KK  AA      AA  RR      RR    IIIIII        TT          ....

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
```

Table of Contents

```
0000     1          .TITLE  LIB$$PACK_ARITH         PACKED ARITHMETIC
0000     2          .IDENT  /1-009/                 ; FILE:LIBPKARIT.MAR EDIT:MDL1009
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  ;       VERSION: 1
0000    30  ;
0000    31  ;       HISTORY:
0000    32  ;
0000    33  ;       AUTHOR:
0000    34  ;               Linda E. Benson         1-August-1981
0000    35  ;
0000    36  ;       MODIFIED BY:
0000    37  ;
0000    38  ;
0000    39  ;
```

```
0000     41          .SBTTL  HISTORY         ; Detailed current edit history
0000     42
0000     43 ;
0000     44 ; Edit history for Version 1 of PACKARITH
0000     45 ;
0000     46 ; 1-001 Original LEB 01-Aug-81
0000     47 ; 1-002 Change shared external reference to G^ and added PSECT declarations
0000     48 ;       STR$DATA and _STR$CODE.  RNH 25-Sep-81
0000     49 ; 1-003 Changed PSECT declarations to use LIB$ prefixes instead of STR$
0000     50 ;       prefixes.  Also changed code to now return a status instead of
0000     51 ;       signalling.  Supplied default info on parameters to the routines
0000     52 ;       and changed the order of some of the input parameters to conform
0000     53 ;       to the calling standard.  LEB 29-DEC-81
0000     54 ; 1-004 Fixed the ASHP in LIB$$CVT_STR_PACK_R9 so it doesn't use overlapping
0000     55 ;       operands.  A similar fix was made to LIB$$MUL_PACK_R10.  LEB 1-APR-82.
0000     56 ; 1-005 Added LIB$$UNPACK_SD_R8 and LIB$$CVT_AP_P entry points.  RNH 21-Jan-82
0000     57 ;       (date of January was date code was written - but not signed in until
0000     58 ;        BASIC V2.0 would ship).
0000     59 ; 1-006 Changed all occurrences of the following instruction format -
0000     60 ;       (MOVAQ -8[R8],R8) to (MOVAQ @#-8[R8],R8) to get around the problem
0000     61 ;       of getting incorrect results when the module is pulled into a
0000     62 ;       shareable image.  LEB 1-DEC-82
0000     63 ; 1-007 Changed CLRQ instruction in routine LIB$$MUL_PACK_R10 to a
0000     64 ;       MOVP instruction so that a valid packed digit can be ensured
0000     65 ;       during further arithmetic operations.  While this worked fine
0000     66 ;       on most VAX processors, we encountered different results on
0000     67 ;       uVAX due to the CLRQ usage.  LEB 25-Feb-84
0000     68 ; 1-008 Fixed bug in LIB$$SUB_PACK_R8. STAN 15-Jun-1984
0000     69 ; 1-009 Moved packed decimal constants into a code psect and eliminated OWN
0000     70 ;       storage to allow for demand-zero linker compression.  MDL 6-Jul-1984
```

```
                        72          .SBTTL  DECLARATIONS
                        73
                        74 ;
                        75 ; INCLUDE FILES
                        76 ;
                        77
                        78 ;
                        79 ; EXTERNAL SYMBOLS
                        80 ;
                        81
                        82          .DSABL          GBL                             ;\Prevent undefined symbols
                        83                                                          ;/from being automatically g
                        84
                        85          .EXTRN          LIB$STOP                        ; Signal a condition and sto
                        86          .EXTRN          LIB$_INVARG                     ; Invalid argument
                        87          .EXTRN          SS$_NORMAL                      ; Successful completion
                        88          .EXTRN          LIB$AB_CVTTP_U
                        89          .EXTRN          LIB$AB_CVTPT_U
                        90
                        91 ;
                        92 ; MACROS
                        93 ;
                        94 ;        NONE
                        95 ;
                        96
                        97 ;
                        98 ; PSECT DECLARATIONS
                        99 ;
                       100
                       101 ;
                       102 ; EQUATED SYMBOLS
                       103 ;
                       104
                       105 ;
                       106 ; CODE PSECT
                       107 ;
                       108
                 00000000  109          .PSECT  _LIB$CODE       RD, NOWRT, EXE, SHR, PIC
                       110
                       111 ;
                       112 ; CONSTANTS
                       113 ;
                       114
0C 00 00 00 00 00 00 00 01  0000  115 RADIX:          .PACKED         1000000000000000
   9C 99 99 99 99 99 99 99  0009  116 RADIX_M_1:      .PACKED         999999999999999
   0C 00 00 00 00 00 00 00  0011  117 ZERO:           .PACKED         000000000000000
                         5C  0019  118 FIVE:           .PACKED         5
                         1C  001A  119 ONE:            .PACKED         1
```

```
                        .SBTTL  LIB$$CVT_STR_PACK_R9    Converts a string to packed array

                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       This routine converts a string of decimal digits to an array of
                ;       packed decimal values.  Each entry in the array contains 15 digits.
                ;
                ; FORMAL PARAMETERS:
                ;   addr.rnu.r          R6      Address of string of decimal digits
                ;   num_digs.rl.v       R7      Number of digits in string
                ;   num_ents.rl.v       R8      Number of entries in array
                ;   retn_str.wp.r       R9      Address of packed decimal array
                ;
                ; IMPLICIT INPUTS:
                ;       -NONE
                ;
                ; ROUTINE VALUE:
                ;       -NONE
                ;
                ; COMPLETION CODES:
                ;       -NONE
                ;
                ; MACROS:
                ;       -NONE
                ;
                ; SIDE EFFECTS:
                ;       -NONE
                ;--
```

```
                                 001B   152 ;+
                                 001B   153 ; For reference:
                                 001B   154 ;                    R6      Address of string
                                 001B   155 ;                    R7      Number of digits in string
                                 001B   156 ;                    R8      Number of entries in array
                                 001B   157 ;                    R9      Address of packed decimal array
                                 001B   158 ;-
                                 001B   159
                                 001B   160 LIB$$CVT_STR_PACK_R9::
                      5E   08   C2  001B   161          SUBL2   #8, SP                  ; allocate 8 bytes for a temp.
                  57  FF A647   9E  001E   162          MOVAB   -1(R6)[R7], R7
                  58  FF A948   7E  0023   163          MOVAQ   -1(R9)[R8], R8
              0002 56   0F   57   F1  0028   164 1$:      ACBL    R7, #15, R6, 2$
                           18   11  002E   165          BRB     4$
              0004 59   08   58   F1  0030   166 2$:      ACBL    R8, #8, R9, 3$
                      5E   08   C0  0036   167          ADDL2   #8, SP                  ; deallocate temp space.
                           05   0039   168          RSB
 0F   00000000'GF    F1 A6   0F   26  003A   169 3$:      CVTTP   #15, -15(R6), G^LIB$AB_CVTTP_U, #15, -8(R9)
                        F8 A9        0044
                           E0   11  0046   170          BRB     1$
                                 0048   171
                      57   56   C2  0048   172 4$:      SUBL    R6, R7
                      57   10   C0  004B   173          ADDL    #16, R7
 0F   00000000'GF    F1 A6   57   26  004E   174          CVTTP   R7, -15(R6), G^LIB$AB_CVTTP_U, #15, (R9)
                           69        0058
                  57   0F   57   C3  0059   175          SUBL3   R7, #15, R7
   6E   0F   00   69   0F   57   F8  005D   176          ASHP    R7, #15, (R9), #0, #15, (SP)
                      69   6E   7D  0064   177          MOVQ    (SP), (R9)
                           05   11  0067   178          BRB     6$
                69    A4 AF   0F   34  0069   179 5$:      MOVP    #15, ZERO, (R9)
              FFF5 59   08   58   F1  006E   180 6$:      ACBL    R8, #8, R9, 5$
                      5E   08   C0  0074   181          ADDL2   #8, SP                  ; deallocate temp space.
                           05   0077   182          RSB
                                 0078   183
```

```
0078   185          .SBTTL  LIB$$CALC_D_R7          Calculates normalization factor
0078   186
0078   187 ;++
0078   188 ; FUNCTIONAL DESCRIPTION:
0078   189 ;
0078   190 ;       This routine calculates the normalization factor, d = int(b/(v1+1))
0078   191 ;       NOTE: This routine returns R0 = 1 is d = 1 and 0 otherwise.
0078   192 ;
0078   193 ; FORMAL PARAMETERS:
0078   194 ;    addr_v1.rp.r       R6      Address of v1
0078   195 ;    addr_d.rp.r        R7      Address of d
0078   196 ;
0078   197 ; IMPLICIT INPUTS:
0078   198 ;       -NONE
0078   199 ;
0078   200 ; ROUTINE VALUE:
0078   201 ;       -NONE
0078   202 ;
0078   203 ; COMPLETION CODES:
0078   204 ;       -NONE
0078   205 ;
0078   206 ; MACROS:
0078   207 ;       -NONE
0078   208 ;
0078   209 ; SIDE EFFECTS:
0078   210 ;       -NONE
0078   211 ;--
```

```
                                0078   213 ;+
                                0078   214 ; For reference:
                                0078   215 ;                   R6      Address of v1
                                0078   216 ;                   R7      Address of d
                                0078   217 ;-
                                0078   218
                                0078   219 LIB$$CALC_D_R7::
                        5E  0C  C2  0078   220          SUBL    #12, SP                         ; Allocate space on stack for
                                    007B   221                                                  ;   intermediate results
6E  10  66  0F  9B AF   01  21  007B   222          ADDP6   #1, ONE, #15, (R6), #16, (SP)   ; RESULT = v1+1
   0F  FF76 CF  10  6E  10  27  0083   223          DIVP    #16, (SP), #16, RADIX, #15, (R7); (R7) = d, R0 = 0
                            67  008B
        67  0F  8A AF   01  37  008C   224          CMPP4   #1, ONE, #15, (R7)              ; Check for d = 1
                        03  12  0092   225          BNEQ    1$                              ; d neq 1, return status = 0
                        50  01  90  0094   226          MOVB    #1, R0                          ; Return status = 1
                        5E  0C  C0  0097   227 1$:       ADDL    #12, S                          ; Deallocate stack
                                05  009A   228          RSB
```

```
009B    230          .SBTTL  LIB$$CALC_Q_R9          COMPUTE A QUOTIENT DIGIT
009B    231
009B    232 ;++
009B    233 ; FUNCTIONAL DESCRIPTION:
009B    234 ;
009B    235 ; This routine calculates an approximation to single quotient digit.
009B    236 ; The initial computation may not be accurate enough.  In the latter
009B    237 ; case, the initial value is "adjusted" by one.  The adjustment process
009B    238 ; will only be attempted twice.
009B    239 ;
009B    240 ; FORMAL PARAMETERS:
009B    241 ;    addr_v1.rp.r       R6      Address of v(1)
009B    242 ;    addr_uj.rp.r       R7      Address of u(j)
009B    243 ;    flag.rl.v          R8      flag - A non-zero value indicates that the
009B    244 ;                               divisor, B, consists of only one 15 digit chunk
009B    245 ;                               and consequently an abbreviated algorithm for
009B    246 ;                               computing q(j) can be used.
009B    247 ;
009B    248 ;    addr_qj.wp.r       R9      Address of q(j)
009B    249 ;
009B    250 ; IMPLICIT INPUTS:
009B    251 ;       -NONE
009B    252 ;
009B    253 ; ROUTINE VALUE:
009B    254 ;       -NONE
009B    255 ;
009B    256 ; COMPLETION CODES:
009B    257 ;       -LIB$_INVARG
009B    258 ;       -SS$_NORMAL
009B    259 ;
009B    260 ; MACROS:
009B    261 ;       -NONE
009B    262 ;
009B    263 ; SIDE EFFECTS:
009B    264 ;       -NONE
009B    265 ;--
```

```
                                009B   267
                                009B   268 ;+
                                009B   269 ; The following is the equation that will calculate
                                009B   270 ; a quotient digit.
                                009B   271 ; q = (u(j)*RADIX + u(j+1))/v(1);
                                009B   272 ; For reference:
                                009B   273 ;                R6      Address of v(1)
                                009B   274 ;                R7      Address of u(j)
                                009B   275 ;                R8      flag - A non-zero value indicates that the divisor, B,
                                009B   276 ;                               consists of only one 15 digit chunk, and conse-
                                009B   277 ;                               quently an abbreviated algorithm for computing
                                009B   278 ;                               q(j) can be used.
                                009B   279 ;                R9      Address of q(j)
                                009B   280
                                009B   281 LIB$$CALC_Q_R9::
                     5E   20 C2 009B   282         SUBL    #32, SP                         ; Allocate stack space for
                                009E   283                                                 ;    intermediate calculations
   6E  1E  00  67    0F   0F F8 009E   284         ASHP    #15, #15, (R7), #0, #30, (SP)   ; (SP) = u(j)*radix
     6E  1E  08 A7   0F   20    00A5   285         ADDP4   #15, 8(R7), #30, (SP)           ; RESULT = u(j)*radix + u(j+1)
           67  66    0F   35    00AB   286         CMPP3   #15, (R6), (R7)                 ; Compare u(j) and V(1)
                     08   12    00AF   287         BNEQ    1$                              ;
         69  FF53 CF   0F   34    00B1   288         MOVP    #15, RADIX_M_1, (R9)            ; q = radix - 1
                     07   11    00B7   289         BRB     2$
   69  0F  6E  1E  66   0F   27 00B9   290 1$:     DIVP    #15, (R6), #30, (SP), #15, (R9) ; (R9) = q
                     58   D5    00C0   291 2$:     TSTL    R8                              ; see if adjustment is needed
                     6D   12    00C2   292         BNEQ    RETURN                          ; R8 = 0 means no adjustment
                                00C4   293
10 AE  1E  66  0F   69   0F   25 00C4   294         MULP    #15, (R9), #15, (R6), #30, 16(SP)
                                00CC   295                                                 ; 16(SP) = V1 = q*v(1)
     6E  1E  10 AE  1E   22    00CC   296         SUBP4   #30, 16(SP), #30, (SP)          ; (SP) = U1 =
                                00D2   297                                                 ;    u(j)*radix+u(i+1)-V1
10 AE  1E  00  6E  1E   0F   F8 00D2   298         ASHP    #15, #30, (SP), #0, #30, 16(SP) ; 16(SP) = U2 = U1*radix
   10 AE  1E  10 A7   0F   20    00DA   299         ADDP4   #15, 16(R7), #30, 16(SP)        ; 16(SP) = U2+u(j+2) = U3
6E  1E  08 A6  0F   69   0F   25 00E1   300         MULP    #15, (R9), #15, 8(R6), #30, (SP); (SP) = V2 = q*v(2)
     10 AE  1E  6E  1E   22    00E9   301         SUBP4   #30, (SP), #30, 16(SP)          ; 16(SP) = U3 - V2
                     40   18    00EF   302         BGEQ    RETURN
                                00F1   303
                                00F1   304 ;
                                00F1   305 ; Quotient digit must be adjusted.
                                00F1   306 ;
                                00F1   307
   69  0F  FF24 CF   01   22    00F1   308         SUBP4   #1, ONE, #15, (R9)              ; (R9) = q - 1
6E  1E  00  66  0F   0F   F8 00F8   309         ASHP    #15, #15, (R6), #0, #30, (SP)   ; (SP) = v(1)*radix
     6E  1E  08 A6  0F   20    00FF   310         ADDP4   #15, 8(R6), #30, (SP)           ; (SP) = v(1)*radix+v(2)
     10 AE  1E  6E  1E   20    0105   311         ADDP4   #30, (SP), #30, 16(SP)          ; 16(SP) = new test value
                     24   18    010B   312         BGEQ    RETURN
                                010D   313
                                010D   314 ;
                                010D   315 ; Quotient digit must be adjusted again.
                                010D   316 ;
                                010D   317
   69  0F  FF08 CF   01   22    010D   318         SUBP4   #1, ONE, #15, (R9)              ; (R9) = q - 1
6E  1E  00  66  0F   0F   F8 0114   319         ASHP    #15, #15, (R6), #0, #30, (SP)   ; (SP) = v(1)*radix
     6E  1E  08 A6  0F   20    011B   320         ADDP4   #15, 8(R6), #30, (SP)           ; (SP) = v(1)*radix+v(2)
     10 AE  1E  6E  1E   20    0121   321         ADDP4   #30, (SP), #30, 16(SP)          ; 16(SP) = new test value
                     08   18    0127   322         BGEQ    RETURN
                                0129   323
```

```
                        0129   324 ;
                        0129   325 ; If adjustment needs to be made more than twice, something is wrong.
                        0129   326 ;
                        0129   327
     50  00000000'8F    D0  0129   328 ERR:    MOVL    #LIB$_INVARG,R0
                        05  0130   329         RSB
                            0131   330
           5E    20     C0  0131   331 RETURN: ADDL    #32, SP                         ; Deallocate stack space
     50  00000000'8F    D0  0134   332         MOVL    #SS$_NORMAL,R0
                        05  013B   333         RSB
                            013C   334
```

```
                        .SBTTL  LIB$$ADJUST_Q_R9

                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       This routine is used to adjust the intermediated results of the divi-
                ;       sion algorithm.  In particular when the difference of v(j)v(j+1)...
                ;       v(j+m) and q*u(1)u(2)...u(m) is negative, this routine is called to
                ;       subtract one from the value of q and add u(1)u(2)...u(m) to the
                ;       previously mentioned difference to get a positive result.
                ;
                ; FORMAL PARAMETERS:
                ;    b_chunks.rl.v      R6      Number of chunks in B ( i.e. the value of m)/
                ;                               Maximum value of index
                ;    addr_u1.rp.r       R7      Addr of u(j)
                ;    addr_v1.rp.r       R8      Addr of v(1)
                ;    addr_q.mp.r        R9      Addr of q / carry flag
                ;
                ; IMPLICIT INPUTS:
                ;       -NONE
                ;
                ; ROUTINE VALUE:
                ;       -NONE
                ;
                ; COMPLETION CODES:
                ;       -NONE
                ;
                ; MACROS:
                ;       -NONE
                ;
                ; SIDE EFFECTS:
                ;       -NONE
                ;--
```

```
                                013C   370 ;+
                                013C   371 ; For reference:
                                013C   372 ;                   R6      Number of chunks in B ( i.e. the value of m) / Maximum
                                013C   373 ;                           value of index
                                013C   374 ;                   R7      Addr of u(j+1)
                                013C   375 ;                   R8      Addr of v(1)
                                013C   376 ;                   R9      Addr of q / carry flag
                                013C   377 ;-
                                013C   378
                                013C   379 LIB$$ADJUST_Q_R9::
      69   0F    FED9 CF    01   22  013C   380          SUBP4   #1, ONE, #15, (R9)      ; Decrement q
                           59   D4  0143   381          CLRL    R9                      ; Set carry flag to 0
           56   FFFFFFF8 9F46   7E  0145   382          MOVAQ   @#-8[R6], R6            ; Set upper index value
                           11   11  014D   383          BRB     2$
                     0D 59   00   E5  014F   384 1$:      BBCC    #0, R9, 2$              ; Check and clear carry flag
     6746  0F    FEC2 CF    01   20  0153   385          ADDP4   #1, ONE, #15, (R7)[R6]  ; Add in carry from last iteration
                           03   1C  015B   386          BVC     2$                      ; Branch if no carry
                        59 01   90  015D   387          MOVB    #1, R9                  ; Set carry flag
      6746   0F    6846   0F   20  0160   388 2$:      ADDP4   #15, (R8)[R6], #15, (R7)[R6]
                                0167   389                                             ; Add u(m-i) to v(j+m-i)
                           03   1C  0167   390          BVC     3$                      ; Branch if no carry
                        59 01   90  0169   391          MOVB    #1, R9                  ; Set carry flag
  FFD9 56   FFFFFFF8 8F   00   F1  016C   392 3$:      ACBL    #0, #-8, R6, 1$         ; Loop until all chunks are processed
                              05   0176   393          RSB
                                0177   394
```

```
0177   396          .SBTTL  LIB$$MUL_PACK_R10         MULTIPLY PACKED STRINGS
0177   397 ;++
0177   398 ; FUNCTIONAL DESCRIPTION:
0177   399 ;
0177   400 ;       This routine multiplies an array of 15 digit packed decimal values
0177   401 ;       by a 15 digit packid decimal multiplier and stores the result in an
0177   402 ;       array of 15 digit packed decimal values.  The size of the result
0177   403 ;       array is either the same as the size of the multiplicand or one entry
0177   404 ;       larger, depending on the passed parameters
0177   405 ;
0177   406 ;               product <-- multiplicand * multiplier
0177   407 ;
0177   408 ; FORMAL PARAMETERS:
0177   409 ;    mulpr.rp.r          R6      Addr of the packed multiplier
0177   410 ;    mulpd.rp.r          R7      Addr of the packed multiplicand array
0177   411 ;    len_mulpd.rl.v      R8      Length of packed multiplicand array
0177   412 ;    len_prod.rl.v       R9      Length of packed product array
0177   413 ;    prod.wp.r           R10     Addr of the packed product array
0177   414 ;
0177   415 ; IMPLICIT INPUTS:
0177   416 ;       -NONE
0177   417 ;
0177   418 ; ROUTINE VALUE:
0177   419 ;       -NONE
0177   420 ;
0177   421 ; COMPLETION CODES:
0177   422 ;       -NONE
0177   423 ;
0177   424 ; MACROS:
0177   425 ;       -NONE
0177   426 ;
0177   427 ; SIDE EFFECTS:
0177   428 ;       -NONE
0177   429 ;--
```

```
                                0177   431 ;+
                                0177   432 ; For reference:
                                0177   433 ;                   R6      Addr of the packed multiplier
                                0177   434 ;                   R7      Addr of the packed multiplicand array
                                0177   435 ;                   R8      Length of packed multiplicand array
                                0177   436 ;                   R9      Length of packed product array
                                0177   437 ;                   R10     Addr of the packed product array
                                0177   438 ;-
                                0177   439
                                0177   440 LIB$$MUL_PACK_R10::
                          5E    20  C2  0177   441          SUBL    #32, SP                 ; Allocate stack space for intermediate
                                0179   442                                                  ;   results
                          59    58  C2  017A   443          SUBL    R8, R9                  ; R9 = difference in product and
                                017D   444                                                  ;       multiplicand array lengths
                58  FFFFFFF8 9F48  7E  017D   445          MOVAQ   a#-8[R8], R8            ; R8 = largest permissible index value
                6E   FE87 CF    0F  34  0185   446          MOVP    #15,ZERO,(SP)           ; Set CARRY = 0 for first iteration
          1E  6748   0F    66    0F  25  018B   447 1$:      MULP    #15, (R6), #15, (R7)[R8], #30, 8(SP)
                                0192
                                0194   448                                                  ; 8(SP) = the product of the ith entry
                                0194   449                                                  ;         of the multiplicand and the
                                0194   450                                                  ;         multiplier.
          08 AE   1E    6E    0F  20  0194   451          ADDP4   #15, (SP), #30, 8(SP)   ; Add in carry from previous iteration
             6A48   10 AE    0F  34  019A   452          MOVP    #15, 16(SP), (R10)[R8]  ; Store low order 15 digits of this
                                01A0   453                                                  ;   iteration in the ith entry of the
                                01A0   454                                                  ;   product array
                      10 AE   0C  90  01A0   455          MOVB    #^X0C, 16(SP)           ; Add a low order zero and plus sign to
                                01A4   456                                                  ;   to high order digits
    0F    00    08 AE   10    FF 8F  F8  01A4   457          ASHP    #-1, #16, 8(SP), #0, #15, 24(SP)
                                   18 AE   01AC
                      6E    18 AE   7D  01AE   458          MOVQ    24(SP), (SP)
                                01B2   459                                                  ; CARRY = High 15 digits of this
                                01B2   460                                                  ;         iteration
  FFCF 58   FFFFFFF8 8F    00    F1  01B2   461          ACBL    #0, #-8, R8, 1$         ; Loop until all entries in the multi-
                                01BC   462                                                  ;   plicand array have been processed
                          59    D5  01BC   463          TSTL    R9                      ; Check if the carry from the last iter-
                                01BE   464                                                  ;   ation should be returned with the
                                01BE   465                                                  ;   product
                          05    13  01BE   466          BEQL    2$                      ;
                   6A48   6E    0F  34  01C0   467          MOVP    #15, (SP), (R10)[R8]    ; Return carry digits
                          5E    20  C0  01C5   468 2$:      ADDL    #32, SP                 ; Deallocate stack space
                                05  01C8   469          RSB                             ; Return
                                01C9   470
```

```
                        .SBTTL  LIB$$SUB_PACK_R8        SUBTRACT PACKED STRINGS

                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       This routine subtracts one array of packed decimal values from another.
                ;       The lengths of the input strings are assumed to be equal.
                ;               difference <-- difference - subtrahend
                ;       NOTE:  This routine returns R0 = 1 if the difference is less than zero
                ;               and 0 otherwise
                ;
                ; FORMAL PARAMETERS:
                ;    num_digs.rl.v      R6      Number of 15 digits chunks minus one in subtrahend
                ;    addr_mind.rp.r     R7      Address of minuend (difference) array
                ;    addr_subd.rp.r     R8      Address of subtrahend array
                ;
                ; IMPLICIT INPUTS:
                ;       -NONE
                ;
                ; ROUTINE VALUE:
                ;       -NONE
                ;
                ; COMPLETION CODES:
                ;       -NONE
                ;
                ; MACROS:
                ;       -NONE
                ;
                ; SIDE EFFECTS:
                ;       -NONE
                ;--
```

```
                          01C9    504 ;+
                          01C9    505 ; For reference:
                          01C9    506 ;                   R6      Number of 15 digits chunks minus one in subtrahend
                          01C9    507 ;                   R7      Address of minuend (difference) array
                          01C9    508 ;                   R8      Address of subtrahend array
                          01C9    509 ;-
                          01C9    510
                          01C9    511 LIB$$SUB_PACK_R8::
        56   00000000 9F46   7E   01C9    512             MOVAQ   a#0[R6], R6             ; Compute largest index value
                       0C   11   01D1    513             BRB     2$                      ; Not necessary to check for borrow
                          01D3    514                                                     ;    on first iteration
                       50   D5   01D3    515 1$:         TSTL    R0                      ; Check for borrow on last iteration
                       08   13   01D5    516             BEQL    2$                      ; Branck if no borrow
  6746   0F   FE3E CF   01   22   01D7    517             SUBP4   #1, ONE, #15, (R7)[R6]  ; Process borrow.
     6746   0F   6846   0F   22   01DF    518 2$:         SUBP4   #15, (R8)[R6], #15, (R7)[R6]
                          01E6    519                                                     ; Subtract one 15 digit chunk.  Note
                          01E6    520                                                     ;   that R0 = 0 after execution.
                       02   1D   01E6    521             BVS     3$                      ; Overflow could be set if we borrowed
                          01E8    522                                                     ; a 1 from a 0 and then further
                          01E8    523                                                     ; subtracted a 9. This results in a -0
                          01E8    524                                                     ; (with the N bit cleared and V bit set).
                          01E8    525                                                     ; In that case, we still want to borrow.
                       0A   18   01E8    526             BGEQ    4$                      ; Branch if no borrow is necessary
  6746   0F   FE11 CF   10   20   01EA    527 3$:         ADDP4   #16, RADIX, #15, (R7)[R6]
                          01F2    528                                                     ; Adjust last diference to make it >= 0
                          01F2    529                                                     ; Note that ADDP4 sets R) to 0.
                       50   D6   01F2    530             INCL    R0                      ; Set borrow flag (R0 <- 1)
FFD5 56   FFFFFFF8 8F   00   F1   01F4    531 4$:         ACBL    #0, #-8, R6, 1$         ; Loop until all chunks are processed
                       05   01FE    532             RSB
                          01FF    533
```

```
                        .SBTTL  LIB$$ROUND_R7           Rounds packed decimal array

                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       This routine rounds an array of packed decimal values to a given
                ;       value of significant digits
                ;
                ; FORMAL PARAMETERS:
                ;   addr_dig.rp.r       R6      Address of 15 digit chunk in which rounding
                ;                               occurs
                ;   position.rl.v       R7      Position within low order chunk at which the
                ;                               rounding occurs
                ;
                ; IMPLICIT INPUTS:
                ;       -NONE
                ;
                ; ROUTINE VALUE:
                ;       -NONE
                ;
                ; COMPLETION CODES:
                ;       -NONE
                ;
                ; MACROS:
                ;       -NONE
                ;
                ; SIDE EFFECTS:
                ;       -NONE
                ;--
```

```
                                01FF    565 ;+
                                01FF    566 ; For reference:
                                01FF    567 ;                   R6      Address of 15 digit chunk in which rounding occurs
                                01FF    568 ;                   R7      Position within low order chunk at which the
                                01FF    569 ;                           rounding occurs
                                01FF    570 ;-
                                01FF    571
                                01FF    572 LIB$$ROUND_R7::
                        5E  08  C2  01FF    573             SUBL    #8, SP                          ; Allocate stack space for
                                    0202    574                                                     ;    intermediate results
   0F   00  FE11 CF     01  57  F8  0202    575             ASHP    R7, #1, FIVE, #0, #15, (SP)     ; Get value to add to
                                6E      020A
                                    020B    576                                                     ;    array to effect rounding
            66  0F      6E  0F  20  020B    577             ADDP4   #15, (SP), #15, (R6)            ; Add rounding value to array
                                0C  1C  0210    578             BVC     2$                      ; If no carry, we are done
                        56  08  C2  0212    579 1$:         SUBL    #8, R6                          ; Get addr of next higher chunk
        66  0F  FE00 CF     01  20  0215    580             ADDP4   #1, ONE, #15, (R6)              ; Propagate carry
                                F4  1D  021C    581             BVS     1$                      ; If necessary continue to
                                    021E    582                                                     ;    propagate carry
                        5E  08  C0  021E    583 2$:         ADDL    #8, SP                          ; Deallocate stack space
                                05  0221    584             RSB
```

```
                        .SBTTL  LIB$$CVT_PACK_STR_R8    Converts a packed array to string

                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       This routine converts an array of packed decimal values to a string
                ;       of decimal digits.  Each entry in the array contains 15 digits.
                ;
                ; FORMAL PARAMETERS:
                ;    array.rp.ra        R6      Address of packed decimal array
                ;    array_ent.rl.v     R7      Number of entries in array
                ;    ret_string.wt.r    R8      Address of return string
                ;
                ; IMPLICIT INPUTS:
                ;       -NONE
                ;
                ; ROUTINE VALUE:
                ;       -NONE
                ;
                ; COMPLETION CODES:
                ;       -NONE
                ;
                ; MACROS:
                ;       -NONE
                ;
                ; SIDE EFFECTS:
                ;       -NONE
                ;--
```

```
                     0222   616 ;+
                     0222   617 ; For reference:
                     0222   618 ;                    R6      Address of packed decimal array
                     0222   619 ;                    R7      Number of entries in array
                     0222   620 ;                    R8      Address of return string
                     0222   621 ;-
                     0222   622
                     0222   623 LIB$$CVT_PACK_STR_R8::
          53   57    0F   C5  0222   624          MULL3   #15, R7, R3
               53    58   C0  0226   625          ADDL    R8, R3              ; R3 = R8 + 15*R7
     57   FFFFFFF8 9F47   7E  0229   626          MOVAQ   @#-8[R7], R7
0F   00000000'GF   6647   0F   24  0231   627 1$:      CVTPT   #15, (R6)[R7], G^LIB$AB_CVTPT_U, #15, -15(R3)
                     F1 A3         023B
  FFEA 57   FFFFFFF8 8F   00   F1  023D   628          ACBL    #0, #-8, R7, 1$ ;\Decr counter of # of chunks
                           0247   629                                          ;/of 15 packed digits
                        05 0247   630          RSB
```

```
0248    632          .SBTTL  LIB$$UNPACK_SD_R8       Unpacks SD descripter
0248    633
0248    634 ;++
0248    635 ; FUNCTIONAL DESCRIPTION:
0248    636 ;
0248    637 ;       This rountine unpacks the information in an SD descripter and converts
0248    638 ;       the packed decimal string to a leading separate string
0248    639 ;
0248    640 ; FORMAL PARAMETERS:
0248    641 ;
0248    642 ;       INPUT:
0248    643 ;          desc.rp.r            R4      Address of SD descripter
0248    644 ;       OUTPUT:
0248    645 ;          length.ww.r          R5      Address of LENGTH parameter
0248    646 ;          exp.wb.r             R6      Address of EXP parameter
0248    647 ;          sign.wb.r            R7      Address of SIGN parameter
0248    648 ;       BOTH:
0248    649 ;          pointer.ml.r         R8      Address of pointer to temp storage/
0248    650 ;                                       Address of pointer to return string
0248    651 ;
0248    652 ; IMPLICIT INPUTS:
0248    653 ;       -NONE
0248    654 ;
0248    655 ; ROUTINE VALUE:
0248    656 ;       -NONE
0248    657 ;
0248    658 ; COMPLETION CODES:
0248    659 ;       -NONE
0248    660 ;
0248    661 ; MACROS:
0248    662 ;       -NONE
0248    663 ;
0248    664 ; SIDE EFFECTS:
0248    665 ;       -NONE
0248    666 ;--
```

```
                     0248   668 ;+
                     0248   669 ; For reference:
                     0248   670 ;
                     0248   671 ;       R4      Address of SD descripter
                     0248   672 ;       R5      Address of LENGTH parameter
                     0248   673 ;       R6      Address of EXP parameter
                     0248   674 ;       R7      Address of SIGN parameter
                     0248   675 ;       R8      Address of pointer to temp storage/
                     0248   676 ;               Address of pointer to return string
                     0248   677 ;-
                     0248   678
                     0248   679 LIB$$UNPACK_SD_R8::
                 67  94  0248   680         CLRB    (R7)                    ; Set SIGN to 0
00 B8  64  04 B4  64  08  024A   681         CVTPS   (R4), @4(R4), (R4), @(R8); Convert packed to separate
                    03  18  0251   682         BGEQ    1$                      ; Branch if >= 0
                 67  01  90  0253   683         MOVB    #1, (R7)                ; Set SIGN to 1
        01 A3  64  30  3B  0256   684 1$:      SKPC    #^X30, (R4), 1(R3)      ; Skip over leading zeros and sign
               65  50  B0  025B   685         MOVW    R0, (R5)                ; Return actual length
               68  51  D0  025E   686         MOVL    R1, (R8)                ; Return addr of first non-zero digit
            66  08 A4  90  0261   687         MOVB    8(R4), (R6)             ; Return EXP = SCALE
                    05  0265   688         RSB
                     0266   689
```

```
0266   691          .SBTTL  LIB$$CVT_AP_P_R8        Converts packed array to single value
0266   692
0266   693 ;++
0266   694 ; FUNCTIONAL DESCRIPTION:
0266   695 ;
0266   696 ;       This routine converts an array of packed decimal values to a single
0266   697 ;       value specified by and SD descripter.  This routine may signal packed
0266   698 ;       overflow.
0266   699 ;
0266   700 ; FORMAL PARAMETERS:
0266   701 ;
0266   702 ;       INPUT:
0266   703 ;          rnd_trunc.rb.v       R3      Round/trunc indicator
0266   704 ;          arrary.ma.v          R5      Address of packed decimal array
0266   705 ;          length.mb.v          R6      Numver of elements in arrary
0266   706 ;          scale.mw.v           R7      Shift factor for obtaining result
0266   707 ;          sign.rb.v            R8      Sign indicator 1 for - and 0 for +
0266   708 ;       OUTPUT:
0266   709 ;          desc.rp.r            R4      Address of SD descripter
0266   710 ;
0266   711 ; IMPLICIT INPUTS:
0266   712 ;       -NONE
0266   713 ;
0266   714 ; ROUTINE VALUE:
0266   715 ;       -NONE
0266   716 ;
0266   717 ; COMPLETION CODES:
0266   718 ;       -NONF
0266   719 ;
0266   720 ; MACROS:
0266   721 ;       -NONE
0266   722 ;
0266   723 ; SIDE EFFECTS:
0266   724 ;       -NONE
0266   725 ;--
```

```
                                0266   727 ;+
                                0266   728 ; For reference:
                                0266   729 ;
                                0266   730 ;       R3      Round/trunc indicator
                                0266   731 ;       R4      Address of SD descripter
                                0266   732 ;       R5      Pointer to packed decimal array
                                0266   733 ;       R6      Number of elements in arrary
                                0266   734 ;       R7      Shift factor for obtaining result
                                0266   735 ;       R8      Sign indicator 1 for - and 0 for +
                                0266   736 ;-
                                0266   737
                                0266   738 LIB$$CVT_AP_P_R8::
                    5E   28     C2 0266   739         SUBL    #40, SP                 ; Allocate 10 longwords on the stack
                      24 AE     DC 0269   740         MOVPSL  36(SP)                  ; Save current PSL
              24 AE   FF7F 8F   AA 026C   741         BICW    #^XFF7F, 36(SP)         ; Save current DV bit
                      0080 8F   B9 0272   742         BICPSW  #^X80                   ; Turn off decimal overflow reporting
          56    FFFFFFF8 9F46   7E 0276   743         MOVAQ   @#-8[R6], R6            ; Initialize offset into array
                         6E   D4 027E   744         CLRL    (SP)                    ; Initialize overflow indicator
    1F    53    6546     0F   57 F8 0280   745         ASHP    R7, #15, (R5)[R6], R3, #31, 4(SP)
                      04 AE          0287
                                0289   746                                         ; Shift 1st chunk and store
                    56   08     C2 0289   747         SUBL    #8, R6                  ; Decrement number of chunks
                         2E   19 028C   748         BLSS    4$                      ; Branch if none left
                    57   0F     C0 028E   749 1$:     ADDL    #15, R7                 ; Adjust shift count
    1F    00    6546     0F   57 F8 0291   750         ASHP    R7, #15, (R5)[R6], #0, #31, 20(SP)
                      14 AE          0298
                                029A   751                                         ; Shift next chunk and store
                         03   1C 029A   752         BVC     2$                      ; Branch if no overflow
                         6E   01 90 029C   753         MOVB    #1, (SP)                ; Set overflow indicator to 1
     04 AE    1F    14 AE     1F   20 029F   754 2$:     ADDP    #31, 20(SP), #31, 4(SP) ; Add current 15 digit chunk to result
                         03   1C 02A6   755         BVC     3$                      ; Branch if no overflow
                         6E   01 90 02A8   756         MOVB    #1, (SP)                ; Set overflow indicator to 1
  FFD9 56    FFFFFFF8 8F     00   F1 02AB   757 3$:     ACBL    #0, #-8, R6, 1$         ; Branch if more chunks left
                         58   95 02B5   758         TSTB    R8                      ; Check for correct sign
                         03   13 02B7   759         BEQL    4$                      ; Branch if result should be positive
                      13 AE   96 02B9   760         INCB    19(SP)                  ; Change sign to negative
  64    00    04 AE    1F     00   F8 02BC   761 4$:     ASHP    #0, #31, 4(SP), #0, (R4), @4(R4)
                      04 B4          02C3
                                02C5   762                                         ; Store result as indicated by descriptor
                         03   1C 02C5   763         BVC     5$                      ; Branch if no overflow
                         6E   01 90 02C7   764         MOVB    #1, (SP)                ; Set overflow indicator to 1
                      24 AE   B5 02CA   765 5$:     TSTW    36(SP)                  ; Check if caller had DV bit set
                         03   13 02CD   766         BEQL    6$                      ; Branch if overflow reporting not on
                    50   6E   D0 02CF   767         MOVL    (SP), R0                ; Move overflow indicator to R0
                      24 AE   B8 02D2   768 6$:     BISPSW  36(SP)                  ; Restore DV flag
                    5E   28   C0 02D5   769         ADDL    #40, SP                 ; Deallocate stack storage
                         05   02D8   770         RSB                             ; Return
                                02D9   771
                                02D9   772         .END
```

```
ERR                             00000129 R      01
FIVE                            00000019 R      01
LIB$$ADJUST_Q_R9                0000013C RG     01
LIB$$CALC_D_R7                  00000078 RG     01
LIB$$CALC_Q_R9                  0000009B RG     01
LIB$$CVT_AP_P_R8                00000266 RG     01
LIB$$CVT_PACK_STR_R8            00000222 RG     01
LIB$$CVT_STR_PACK_R9            0000001B RG     01
LIB$$MUL_PACK_R10               00000177 RG     01
LIB$$ROUND_R7                   000001FF RG     01
LIB$$SUB_PACK_R8                000001C9 RG     01
LIB$$UNPACK_SD_R8               00000248 RG     01
LIB$AB_CVTPT_U                  ********   X    00
LIB$AB_CVTTP_U                  ********   X    00
LIB$STOP                        ********   X    00
LIB$_INVARG                     ********   X    00
ONE                             0000001A R      01
RADIX                           00000000 R      01
RADIX_M_1                       00000009 R      01
RETURN                          00000131 R      01
SS$_NORMAL                      ********   X    00
ZERO                            00000011 R      01
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
_LIB$CODE	000002D9 (729.)	01 (1.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.03	00:00:02.09
Command processing	107	00:00:00.31	00:00:04.98
Pass 1	95	00:00:01.05	00:00:02.86
Symbol table sort	0	00:00:00.01	00:00:00.16
Pass 2	145	00:00:00.91	00:00:05.94
Symbol table output	4	00:00:00.02	00:00:00.16
Psect synopsis output	2	00:00:00.01	00:00:00.01
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	384	00:00:02.34	00:00:16.20

```
The working set limit was 1050 pages.
11995 bytes (24 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 22 non-local and 28 local symbols.
772 source lines were read in Pass 1, producing 9 object records in Pass 2.
C pages of virtual memory were used to define 0 macros.
```

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

Macro library name                          Macros defined
------------------                          --------------
_$255$DUA28:[SYSLIB]STARLET.MLB;2                    0

0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=(GLOBAL,TRACEBACK)/LIS=LIS$:LIBPKARIT/OBJ=OBJ$:LIBPKARIT MSRC$:LIBPKARIT/UPDATE=(ENH$:LIBPKARIT)
```

LIBLOCC
LIS

LIBLUN
LIS

LIBMOVC3
LIS

LIBMOVTC
LIS

LIBPOLYF
LIS

LIBINSQTI
LIS

LIBINTOVE
LIS

LIBLEXICA
LIS

LIBPKARIT
LIS

LIBMATCHC
LIS

LIBPOLYD
LIS

LIBLOOKUP
LIS

LIBLPLINE
LIS

LIBMOVC5
LIS

LIBLEN
LIS

LIBMSG
LIS

LIBINSV
LIS

LIBMATCA
LIS

LIBMOVTUC
LIS