```
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        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
```

```
**FILE**ID**LIBADDX                                    F  7
```

```
LL            IIIIII    BBBBBBBB      AAAAAA    DDDDDDDD    DDDDDDDD    XX      XX
LL            IIIIII    BBBBBBBB      AAAAAA    DDDDDDDD    DDDDDDDD    XX      XX
LL             II       BB      BB  AA      AA  DD      DD  DD      DD  XX      XX
LL             II       BB      BB  AA      AA  DD      DD  DD      DD  XX      XX
LL             II       BB      BB  AA      AA  DD      DD  DD      DD    XX  XX
LL             II       BB      BB  AA      AA  DD      DD  DD      DD    XX  XX
LL             II       BBBBBBBB    AA      AA  DD      DD  DD      DD      XX
LL             II       BBBBBBBB    AA      AA  DD      DD  DD      DD      XX
LL             II       BB      BB  AAAAAAAAAA  DD      DD  DD      DD    XX  XX
LL             II       BB      BB  AAAAAAAAAA  DD      DD  DD      DD    XX  XX
LL             II       BB      BB  AA      AA  DD      DD  DD      DD  XX      XX      ....
LL             II       BB      BB  AA      AA  DD      DD  DD      DD  XX      XX      ....
LLLLLLLLLL    IIIIII    BBBBBBBB    AA      AA  DDDDDDDD    DDDDDDDD    XX      XX      ....
LLLLLLLLLL    IIIIII    BBBBBBBB    AA      AA  DDDDDDDD    DDDDDDDD    XX      XX      ....

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
```

```
    (2)       56        DECLARATIONS
    (3)       98        LIB$ADDX - Addition of infinite precision integers
    (4)      215        LIB$SUBX - Subtraction of infinite precision integers
```

```
0000            .TITLE  LIB$ADDX - Add infinite precision integers
0000            .IDENT  /1-006/                 ; File: LIBADDX.MAR  Edit: PDG1006
0000
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
0000
0000    ;++
0000    ; FACILITY: General Utility Library
0000    ;
0000    ; ABSTRACT:
0000    ;
0000    ;       Routines for performing addition and subtraction on
0000    ;       integers of arbitrary length.
0000    ;
0000    ; ENVIRONMENT: User Mode, AST Reentrant
0000    ;
0000    ;--
0000    ; AUTHOR: Steven B. Lionel, CREATION DATE: 17-NOV-1978
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ; Steven B. Lionel, : VERSION 01
0000    ; 1-001 - Original
0000    ; 1-002 - Corrected an error in a comment.  JBS 14-DEC-78
0000    ; 1-003 - Add "_" to PSECT directive.  JBS 21-DEC-78
0000    ; 1-004 - Make default loop count 1 instead of 2.  A value of 2
0000    ;         makes it loop 3 times!  SBL 25-MAR-1980
0000    ; 1-005 - Use register temp in loop to allow for overlap.  SBL 13-June-1980
0000    ; 1-006 - Allow length of zero or one.  Recognize overflow.  Recognize length
0000    ;         of -2**31 as an error.  Made compare with (AP) unsigned.  Make sure
0000    ;         that C bit is 0 before entering loop.  PDG 9-Aug-81
0000    ;--
```

```
                    0000          .SBTTL  DECLARATIONS
                    0000     ;
                    0000     ; INCLUDE FILES:
                    0000     ;
                    0000
                    0000     ;
                    0000     ; EXTERNAL DECLARATIONS:
                    0000     ;
                    0000          .DSABL  GBL                       ; Prevent undeclared symbols
                    0000                                            ; from being declared global
                    0000
                    0000          .EXTRN  SS$_NORMAL                ; Normal successful completion
                    0000          .EXTRN  SS$_INTOVF                ; Integer overflow error
                    0000          .EXTRN  LIB$_INVARG               ; Invalid argument to function
                    0000     ;
                    0000     ; MACROS:
                    0000     ;
                    0000
                    0000     ;
                    0000     ; EQUATED SYMBOLS:
                    0000     ;
          00000004  0000          addend = 4              ; Address of addend array
          00000008  0000          augend = 8              ; Address of augend array
          0000000C  0000          sum = 12                ; Address of sum (result) array
                    0000
          00000004  0000          minuend = 4             ; Address of minuend array
          00000008  0000          subtrahend = 8          ; Address of subtrahend array
          0000000C  0000          difference = 12         ; Address of difference array
                    0000
          00000010  0000          length = 16             ; Address of length in longwords
                    0000
                    0000     ;
                    0000     ; OWN STORAGE:
                    0000     ;
                    0000
                    0000     ;
                    0000     ; PSECT DECLARATIONS:
                    0000     ;
                00000000          .PSECT _LIB$CODE PIC, USR, CON, REL, LCL, SHR, -
                    0000                          EXE, RD, NOWRT, LONG
                    0000
```

```
0000          .SBTTL  LIB$ADDX - Addition of infinite precision integers
0000    ;++
0000    ; FUNCTIONAL DESCRIPTION:
0000    ;
0000    ;       LIB$ADDX performs addition of arbitrary length integers.  The
0000    ;       values to be added are located in arrays of longwords: the
0000    ;       higher addresses holding the more significant parts of the values.
0000    ;
0000    ;       The number of longwords to be added is given in the optional
0000    ;       argument "length".  If this is not specified, the default is
0000    ;       2, or quadword addition.
0000    ;
0000    ;       The sum is placed in the array addressed by the third argument.
0000    ;       Any two or all three of the first three arguments may be the same.
0000    ;       If overflow occurs, the function value returned is SS$_INTOVF.
0000    ;
0000    ; CALLING SEQUENCE:
0000    ;
0000    ;       status.wlc.v = LIB$ADDX (addend.rl.ra, augend.rl.ra, sum.wl.ra
0000    ;                               [, length.rl])
0000    ;
0000    ; INPUT PARAMETERS:
0000    ;
0000    ;       addend  - The address of an array of longwords.  The array
0000    ;                 contains a multiple precision integer, with the
0000    ;                 bits increasing in significance with increasing
0000    ;                 addresses.
0000    ;
0000    ;       augend  - The address of an array of longwords.  The array
0000    ;                 contains a multiple precision integer, with the
0000    ;                 bits increasing in significance with increasing
0000    ;                 addresses.
0000    ;
0000    ;       length  - Optional.  The length in longwords of the arrays to
0000    ;                 be added.  The length may be zero or greater.  If
0000    ;                 not, error LIB$_INVARG is returned.
0000    ;
0000    ; IMPLICIT INPUTS:
0000    ;
0000    ;       NONE
0000    ;
0000    ; OUTPUT PARAMETERS:
0000    ;
0000    ;       sum     - The address of an array of longwords.  The sum of the
0000    ;                 addend and augend is placed in this array.
0000    ;
0000    ; IMPLICIT OUTPUTS:
0000    ;
0000    ;       NONE
0000    ;
0000    ; FUNCTION VALUE:
0000    ; COMPLETION CODES:
0000    ;
0000    ;       SS$_NORMAL      - Successful completion
0000    ;       SS$_INTOVF      - Integer overflow - sum is correct except for
0000    ;                         the sign bit which is lost.
0000    ;       LIB$_INVARG     - Invalid argument.  Length is negative.
```

```
                        0000  ;                                    The sum is not changed.
                        0000  ;
                        0000  ; SIDE EFFECTS:
                        0000  ;
                        0000  ;       NONE
                        0000  ;
                        0000  ;--
                        0000
                  0004  0000          .ENTRY LIB$ADDX, ^M<R2>          ; Disable integer overflow
                        0002
                        0002  ;+
                        0002  ;       Set up R0 as the count of longwords remaining.  If length
                        0002  ;       is not specified, use the default of 2 (quadword addition).
                        0002  ;       If length is negative, return error LIB$_INVARG.
                        0002  ;-
                        0002
        50    01    7D  0002          MOVQ    #1, R0                   ; Default two longwords
              51    D7  0005          DECL    R1                       ; Initialize index
        6C    04    91  0007          CMPB    #4, (AP)                 ; Is length present?
              28    1A  000A          BGTRU   2$                       ; No, use default (C=0 if branch taken)
           10 AC    D5  000C          TSTL    length(AP)               ; Is length omitted? (C=0)
              23    13  000F          BEQL    2$                       ; Yes, use default
     50    10 BC    D0  0011          MOVL    @length(AP), R0          ; R0 contains length (C=0 (unchanged))
              04    15  0015          BLEQ    0$                       ; Branch if negative or zero length
              50    D7  0017          DECL    R0                       ; Subtract one (C=0, since R0 was > 0)
              19    11  0019          BRB     2$                       ; Jump into loop
                        001B
                        001B  ;       User gave negative or zero length
                        001B
              2E    13  001B  0$:     BEQL    EXIT                     ; Do nothing if length = 0
 50    00000000'8F  D0  001D          MOVL    #LIB$_INVARG, R0         ; Error
                    04  0024          RET                              ; Return to caller
                        0025
                        0025  ;+
                        0025  ;       Addition loop
                        0025  ;-
                        0025
   52    04 BC41    D0  0025  1$:     MOVL    @addend(AP)[R1], R2      ; Do addition in a temp because augend
   52    08 BC41    D8  002A          ADWC    @augend(AP)[R1], R2      ; and sum may overlap and because ADWC
   0C BC41    52    D0  002F          MOVL    R2, @sum(AP)[R1]         ; is a two-operand instruction
     ED 51    50    F2  0034  2$:     AOBLSS  R0, R1, 1$               ; Loop till done
                        0038
                        0038  ;+
                        0038  ;       Now, add one more time, this time preserving the overflow flag
                        0038  ;-
   52    08 BC41    D0  0038          MOVL    @augend(AP)[R1], R2
0C BC41    04 BC41  D0  003D          MOVL    @addend(AP)[R1], @sum(AP)[R1]
   0C BC41    52    D8  0044          ADWC    R2, @sum(AP)[R1]
                        0049
                        0049  ;+
                        0049  ;       Test for overflow and return with proper condition
                        0049  ;-
                        0049
              05    1D  0049  ATEST:  BVS     AOVFL                    ; Integer overflow
     50    00'8F    9A  004B  EXIT:   MOVZBL  #SS$_NORMAL, R0          ; Return success
                    04  004F          RET
                        0050
```

```
                50   0000'8F   3C  0050    212 AOVFL:  MOVZWL  #SS$_INTOVF, R0            ; Integer overflow
                               04  0055    213         RET                                ; Return failure
```

```
0056            .SBTTL  LIB$SUBX - Subtraction of infinite precision integers
0056     ;++
0056     ; FUNCTIONAL DESCRIPTION:
0056     ;
0056     ;       LIB$SUBX performs subtraction of arbitrary length integers.  The
0056     ;       values to be subtracted are located in arrays of longwords: the
0056     ;       higher addresses being the higher precision parts of the values.
0056     ;
0056     ;       The number of longwords to be subtracted is given in the optional
0056     ;       argument "length".  If this is not specified, the default is
0056     ;       2, or quadword subtraction.
0056     ;
0056     ;       The difference is placed in the array addressed by the third argument.
0056     ;       Any two or all three of the first three arguments may be the same.
0056     ;       If overflow occurs, the function value returned is SS$_INTOVF.
0056     ;
0056     ; CALLING SEQUENCE:
0056     ;
0056     ;       status.wlc.v = LIB$SUBX (minuend.rl.ra, subtrahend.rl.ra,
0056     ;                                difference.wl.ra [, length.rl])
0056     ;
0056     ; INPUT PARAMETERS:
0056     ;
0056     ;       minuend         - The address of an array of longwords.  The array
0056     ;                         contains a multiple precision integer, with the
0056     ;                         bits increasing in significance with increasing
0056     ;                         addresses.
0056     ;
0056     ;       subtrahend      - The address of an array of longwords.  The array
0056     ;                         contains a multiple precision integer, with the
0056     ;                         bits increasing in significance with increasing
0056     ;                         addresses.
0056     ;
0056     ;       length          - Optional.  The length in longwords of the arrays to
0056     ;                         be subtracted.  The length must be greater than one.  If
0056     ;                         not, error LIB$_INVARG is returned.
0056     ;
0056     ; IMPLICIT INPUTS:
0056     ;
0056     ;       NONE
0056     ;
0056     ; OUTPUT PARAMETERS:
0056     ;
0056     ;       difference      - The address of an array of longwords.  The subtrahend is
0056     ;                         subtracted from the minuend, and the result is placed
0056     ;                         in this array.
0056     ;
0056     ; IMPLICIT OUTPUTS:
0056     ;
0056     ;       NONE
0056     ;
0056     ; FUNCTION VALUE:
0056     ; COMPLETION CODES:
0056     ;
0056     ;       SS$_NORMAL      - Successful completion
0056     ;       SS$_INTOVF      - Integer overflow - difference is correct except for
0056     ;                         the sign bit which is lost.
```

```
                               0056   ;       LIB$_INVARG     - Invalid argument.  Length is negative.
                               0056   ;                         The difference is not changed.
                               0056   ;
                               0056   ; SIDE EFFECTS:
                               0056   ;
                               0056   ;       NONE
                               0056   ;
                               0056   ;--
                               0056
                         0004  0056           .ENTRY LIB$SUBX, ^M<R2>         ; Disable integer overflow
                               0058
                               0058   ;+
                               0058   ;       Set up R0 as the count of longwords remaining.  If length
                               0058   ;       is not specified, use the default of 2 (quadword subtraction).
                               0058   ;       If length is negative, return error LIB$_INVARG.
                               0058   ;-
                               0058
             50    01     7D   0058           MOVQ    #1, R0                  ; Default two longwords
                   51     D7   005B           DECL    R1                      ; Initialize index
             6C    04     91   005D           CMPB    #4, (AP)                ; Is length present?
                   28     1A   0060           BGTRU   2$                      ; No, use default (C=0 if branch taken)
                10 AC     D5   0062           TSTL    length(AP)              ; Is length omitted? (C=0)
                   23     13   0065           BEQL    2$                      ; Yes, use default
          50    10 BC     D0   0067           MOVL    @length(AP), R0         ; R0 contains length (C=0 (unchanged))
                   04     15   006B           BLEQ    0$                      ; Branch if negative or zero length
                   50     D7   006D           DECL    R0                      ; Subtract one (C=0, since R0 was > 0)
                   19     11   006F           BRB     2$                      ; Jump into loop
                               0071
                               0071   ;       User gave negative or zero length
                               0071
                   D8     13   0071   0$:     BEQL    EXIT                    ; Do nothing if length = 0
    50    00000000'8F     D0   0073           MOVL    #LIB$_INVARG, R0        ; Error
                          04   007A           RET                             ; Return to caller
                               007B
                               007B   ;+
                               007B   ;       Subtraction loop
                               007B   ;-
                               007B
        52    04 BC41     D0   007B   1$:     MOVL    @minuend(AP)[R1], R2            ; Do subtraction in a temp
                               0080                                                   ; because subtrahend and
        52    08 BC41     D9   0080           SBWC    @subtrahend(AP)[R1], R2         ; difference may overlap and
        0C BC41   52      D0   0085           MOVL    R2, @difference(AP)[R1]         ; because SBWC is a 2 operand
                               008A                                                   ; instruction
           ED 51    50    F2   008A   2$:     AOBLSS  R0, R1, 1$                      ; Loop till done
                               008E
                               008E   ;+
                               008E   ;       Now subtract one final time, this time preserving overflow
                               008E   ;-
                               008E
        52    08 BC41     D0   008E           MOVL    @subtrahend(AP)[R1], R2
   0C BC41    04 BC41     D0   0093           MOVL    @minuend(AP)[R1], @difference(AP)[R1]
        0C BC41   52      D9   009A           SBWC    R2, @difference(AP)[R1]
                               009F
                               009F   ;+
                               009F   ;       Test for overflow and return with proper condition
                               009F   ;-
                   AF     1D   009F           BVS     AOVFL                   ; Test for overflow
```

```
                              00A1   329
         50   00'8F      9A   00A1   330          MOVZBL  #SS$_NORMAL, R0          ; Return success
                         04   00A5   331          RET
                              00A6   332
                              00A6   333          .END
```

```
ADDEND          = 00000004
AOVFL             00000050 R     01
ATEST             00000049 R     01
AUGEND          = 00000008
DIFFERENCE      = 0000000C
EXIT              0000004B R     01
LENGTH          = 00000010
LIB$ADDX          00000000 RG    01
LIB$SUBX          00000056 RG    01
LIB$_INVARG       ********   X   00
MINUEND         = 00000004
SS$_INTOVF        ********   X   00
SS$_NORMAL        ********   X   00
SUBTRAHEND      = 00000008
SUM             = 0000000C

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

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
_LIB$CODE                       000000A6  (  166.)  01 (  1.)    PIC   USR   CON   REL   LCL   SHR   EXE   RD  NOWRT NOVEC LONG

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

Phase                    Page faults    CPU Time       Elapsed Time
-----                    -----------    --------       ------------
Initialization                   29     00:00:00.05    00:00:01.01
Command processing              117     00:00:00.35    00:00:02.30
Pass 1                           72     00:00:00.48    00:00:04.92
Symbol table sort                 0     00:00:00.00    00:00:00.00
Pass 2                           72     00:00:00.40    00:00:02.57
Symbol table output               2     00:00:00.01    00:00:00.40
Psect synopsis output             3     00:00:00.00    00:00:00.01
Cross-reference output            0     00:00:00.00    00:00:00.00
Assembler run totals            297     00:00:01.30    00:00:11.29

The working set limit was 750 pages.
3941 bytes (8 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 15 non-local and 6 local symbols.
333 source lines were read in Pass 1, producing 14 object records in Pass 2.
0 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$:LIBADDX/OBJ=OBJ$:LIBADDX MSRC$:LIBADDX/UPDATE=(ENH$:LIBADDX)
```

0203 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY