```
FFFFFFFFFFFFFFF     000000000     RRRRRRRRRRRR     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
FFFFFFFFFFFFFFF     000000000     RRRRRRRRRRRR     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
FFFFFFFFFFFFFFF     000000000     RRRRRRRRRRRR     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
FFF              000         000  RRR         RRR  RRR         RRR        TTT        LLL
FFF              000         000  RRR         RRR  RRR         RRR        TTT        LLL
FFF              000         000  RRR         RRR  RRR         RRR        TTT        LLL
FFF              000         000  RRR         RRR  RRR         RRR        TTT        LLL
FFF              000         000  RRR         RRR  RRR         RRR        TTT        LLL
FFF              000         000  RRR         RRR  RRR         RRR        TTT        LLL
FFFFFFFFFFFF     000         000  RRRRRRRRRRRR     RRRRRRRRRRRR           TTT        LLL
FFFFFFFFFFFF     000         000  RRRRRRRRRRRR     RRRRRRRRRRRR           TTT        LLL
FFFFFFFFFFFF     000         000  RRRRRRRRRRRR     RRRRRRRRRRRR           TTT        LLL
FFF              000         000  RRR   RRR        RRR   RRR              TTT        LLL
FFF              000         000  RRR   RRR        RRR   RRR              TTT        LLL
FFF              000         000  RRR   RRR        RRR   RRR              TTT        LLL
FFF              000         000  RRR      RRR     RRR      RRR           TTT        LLL
FFF              000         000  RRR      RRR     RRR      RRR           TTT        LLL
FFF              000         000  RRR      RRR     RRR      RRR           TTT        LLL
FFF                 000000000     RRR         RRR  RRR         RRR        TTT        LLLLLLLLLLLLLLL
FFF                 000000000     RRR         RRR  RRR         RRR        TTT        LLLLLLLLLLLLLLL
FFF                 000000000     RRR         RRR  RRR         RRR        TTT        LLLLLLLLLLLLLLL
```

FILEID**COMR50ASC

```
  CCCCCCCC    000000    MM      MM  RRRRRRRR    5555555555    000000      AAAAAA      SSSSSSSS    CCCCCCCC
  CCCCCCCC    000000    MM      MM  RRRRRRRR    5555555555    000000      AAAAAA      SSSSSSSS    CCCCCCCC
CC          00      00  MMMM  MMMM  RR      RR  55            00      00  AA      AA  SS          CC
CC          00      00  MMMM  MMMM  RR      RR  55            00      00  AA      AA  SS          CC
CC          00      00  MM  MM  MM  RR      RR  555555        00    0000  AA      AA  SS          CC
CC          00      00  MM  MM  MM  RR      RR  555555        00    0000  AA      AA  SS          CC
CC          00      00  MM      MM  RRRRRRRR          55    00  00  00  AA      AA    SSSSSS    CC
CC          00      00  MM      MM  RRRRRRRR          55    00  00  00  AA      AA    SSSSSS    CC
CC          00      00  MM      MM  RR  RR              55  0000    00  AAAAAAAAAA          SS  CC
CC          00      00  MM      MM  RR  RR              55  0000    00  AAAAAAAAAA          SS  CC
CC          00      00  MM      MM  RR    RR    55      55  00      00  AA      AA          SS  CC              ....
CC          00      00  MM      MM  RR    RR    55      55  00      00  AA      AA          SS  CC              ....
  CCCCCCCC    000000    MM      MM  RR      RR    555555      000000    AA      AA  SSSSSSSS      CCCCCCCC      ....
  CCCCCCCC    000000    MM      MM  RR      RR    555555      000000    AA      AA  SSSSSSSS      CCCCCCCC      ....

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            .TITLE  COM$R50ASC      ; FORTRAN COMPATIBILITY - RADIX 50 TO ASCII CONVERSI
0000            .IDENT  /1-003/         ; File: COMR50ASC.MAR  Edit: SBL1003
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    ; FACILITY: FORTRAN COMPATIBILITY LIBRARY
0000    ;++
0000    ; ABSTRACT:
0000    ;
0000    ;       R50ASC is a FORTRAN callable procedure to convert radix-50 strings
0000    ;       into ASCII Hollerith strings.
0000    ;
0000    ;--
0000    ;
0000    ; VERSION: 0
0000    ;
0000    ; HISTORY:
0000    ;
0000    ; AUTHOR:
0000    ;       Peter Yuo, 9-Sep-77: Version 0
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ;
0000    ;
```

```
0000            .SBTTL  HISTORY         ; Detailed Current Edit History
0000
0000
0000    ; Edit History for Version 01 of R50ASC
0000    ;
0000
0000    ; 0-03  Correct constant values for EDIV instructions
0000    ; 00-06 - Change file name to COMR50ASC.MAR, PSECT to F4PCOMPAT.  TNH 5-Jan-78
0000    ; 00-07 - Bug fix: don't use no_output_char for input count!  JMT 13-Feb-78
0000    ; 1-001 - Update version number and copyright notice.  JBS 16-NOV-78
0000    ; 1-002 - Add "_" to PSECT directive.  JBS 21-DEC-78
0000    ; 1-003 - Fix bug where largest possible RAD50 value was considered
0000    ;           invalid.  SBL 6-May-80
```

```
                    0000     64          .SBTTL  DECLARATIONS
                    0000     65
                    0000     66 ;
                    0000     67 ; INCLUDE FILES:
                    0000     68 ;
                    0000     69
                    0000     70 ;
                    0000     71 ; EXTERNAL SYMBOLS:
                    0000     72 ;
                    0000     73
                    0000     74 ;
                    0000     75 ; MACROS:
                    0000     76 ;
                    0000     77
                    0000     78 ;
                    0000     79 ; PSECT DECLARATIONS:
                    0000     80 ;
                00000000     81          .PSECT  _F4PCOMPAT$CODE PIC,USR,CON,REL,LCL,SHR,EXE,RD,NOWRT
                    0000     82
                    0000     83 ;
                    0000     84 ; EQUATED SYMBOLS:
                    0000     85 ;
                    0000     86
        00000004    0000     87          no_output_char  = 4                 ; no_output_char.rw.r
        00000008    0000     88          radix50_array   = 8                 ; radix50_array.rbu.ra
        0000000C    0000     89          ascii_array     = 12                ; ascii_array.wbu.ra
        00000640    0000     90          L_50_POWER_2    = ^X0a16 + ^X640
                    0000     91                                              ; 50(octal)**2
                    0000     92
                    0000     93 ;
                    0000     94 ; OWN STORAGE:
                    0000     95 ;
                    0000     96
                    0000     97 ASCII_TAB:
                20  0000     98          .BYTE   ^040                        ; 0 - space
                41  0001     99          .BYTE   ^0101                       ; 1 - 'A'
                42  0002    100          .BYTE   ^0102                       ; 2 - 'B'
                43  0003    101          .BYTE   ^0103                       ; 3 - 'C'
                44  0004    102          .BYTE   ^0104                       ; 4 - 'D'
                45  0005    103          .BYTE   ^0105                       ; 5 - 'E'
                46  0006    104          .BYTE   ^0106                       ; 6 - 'F'
                47  0007    105          .BYTE   ^0107                       ; 7 - 'G'
                48  0008    106          .BYTE   ^0110                       ; 10 - 'H'
                49  0009    107          .BYTE   ^0111                       ; 11 - 'I'
                4A  000A    108          .BYTE   ^0112                       ; 12 - 'J'
                4B  000B    109          .BYTE   ^0113                       ; 13 - 'K'
                4C  000C    110          .BYTE   ^0114                       ; 14 - 'L'
                4D  000D    111          .BYTE   ^0115                       ; 15 - 'M'
                4E  000E    112          .BYTE   ^0116                       ; 16 - 'N'
                4F  000F    113          .BYTE   ^0117                       ; 17 - 'O'
                50  0010    114          .BYTE   ^0120                       ; 20 - 'P'
                51  0011    115          .BYTE   ^0121                       ; 21 - 'Q'
                52  0012    116          .BYTE   ^0122                       ; 22 - 'R'
                53  0013    117          .BYTE   ^0123                       ; 23 - 'S'
                54  0014    118          .BYTE   ^0124                       ; 24 - 'T'
                55  0015    119          .BYTE   ^0125                       ; 25 - 'U'
                56  0016    120          .BYTE   ^0126                       ; 26 - 'V'
```

```
57 0017        .BYTE   ^0127                     ; 27 - 'W'
58 0018        .BYTE   ^0130                     ; 30 - 'X'
59 0019        .BYTE   ^0131                     ; 31 - 'Y'
5A 001A        .BYTE   ^0132                     ; 32 - 'Z'
24 001B        .BYTE   ^044                      ; 33 - '$'
2E 001C        .BYTE   ^056                      ; 34 - '.'
3F 001D        .BYTE   ^077                      ; 35 - unused
30 001E        .BYTE   ^060                      ; 36 - '0'
31 001F        .BYTE   ^061                      ; 37 - '1'
32 0020        .BYTE   ^062                      ; 40 - '2'
33 0021        .BYTE   ^063                      ; 41 - '3'
34 0022        .BYTE   ^064                      ; 42 - '4'
35 0023        .BYTE   ^065                      ; 43 - '5'
36 0024        .BYTE   ^066                      ; 44 - '6'
37 0025        .BYTE   ^067                      ; 45 - '7'
38 0026        .BYTE   ^070                      ; 46 - '8'
39 0027        .BYTE   ^071                      ; 47 - '9'
   0028
   0028  ;
   0028  ; REGISTER USAGE:
   0028  ;
   0028  ; R0:R1 - current word for processing ( use quadword in order to use EDIV)
   0028  ; R2 - holds remainder for EDIV
   0028  ; R3 - CHARS_REM, remaining number of output chars to produce
   0028  ; R4 - NEXT_INPUT_POSITION, next radix-50 input position
   0028  ; R5 - NEXT_OUTPUT_POSITION, next ASCII output position
   0028  ;
```

```
          0028          .SBTTL  R50ASC - RADIX 50 TO ASCII CONVERSION ROUTINE
          0028
          0028    ;++
          0028    ; FUNCTIONAL DESCRIPTION:
          0028    ;
          0028    ;       Algoritmic steps:
          0028    ;       1) Initialization
          0028    ;          CHARS_REM = no_output_char.rw.r
          0028    ;          NEXT_INPUT_POSITION = radix50_array.rbu.ra
          0028    ;          NEXT_OUTPUT_POSITION = ascii_array.wbu.ra
          0028    ;       2) If CHARS_REM =< 0 then DONE
          0028    ;       3) Get current word for processing
          0028    ;       4) If overflow then (output '???', go to step 6)
          0028    ;       5) Do conversion
          0028    ;          [Note] n is a word value, so longword division will not produce
          0028    ;          negative results
          0028    ;          a. (Let n = a2*50**2 + a1 * 50 + a0)
          0028    ;             Divide n by 50**2. (n = q1*50**2 + r1)
          0028    ;             ASCII_TAB[q1] is 1st ASCII char to output
          0028    ;          b. Divide r1 by 50. (r1 = q2*50 + r2)
          0028    ;             ASCII_TAB[q2] is 2nd ASCII char to output
          0028    ;             ASCII_TAB[r2] is 3rd ASCII char to output
          0028    ;       6) CHARS_REM = CHARS_REM -3
          0028    ;       7) Go back to step 2
          0028    ;
          0028    ; CALLING SEQUENCE:
          0028    ;
          0028    ;       CALL R50ASC (no_output_char.rw.r, radix50_array.rbu.ra,
          0028    ;                    ascii_array.rbu.ra)
          0028    ;
          0028    ;
          0028    ; INPUT PARAMETERS:
          0028    ;
          0028    ;       no_output_char.rw.r     ; the max number of output chars to produce
          0028    ;       radix50_array.rbu.ra    ; address of radix-50 input location
          0028    ;
          0028    ; IMPLICIT INPUTS:
          0028    ;       NONE
          0028    ;
          0028    ; OUTPUT PARAMETERS:
          0028    ;
          0028    ;       ascii_array.wbu,ra      ; addres of ascii output location
          0028    ;
          0028    ; IMPLICIT OUTPUTS:
          0028    ;       NONE
          0028    ;
          0028    ; COMPLETION CODES:
          0028    ;       NONE
          0028    ;
          0028    ; SIDE EFFECTS:
          0028    ;       NONE
          0028    ;
          0028    ;--
          0028
          0028
          0028
    403C  0028          .ENTRY  R50ASC, ^M<IV, R2, R3, R4, R5>
```

```
                                  002A                                                 ; standard call-by-reference entry
                                  002A                                                 ; enable integer overflow
                                  002A  ;
                                  002A  ; Initialization
                                  002A  ;
                                  002A
                 53   04 BC   3C  002A                 MOVZWL  @no_output_char(AP), R3 ; R3 = CHARS_REM
                 54   08 AC   D0  002E                 MOVL    radix50_array(AP), R4   ; R4 = NEXT_INPUT_POSITION
                 55   0C AC   D0  0032                 MOVL    ascii_array(AP), R5     ; R5 = NEXT_OUTPUT_POSITION
                         51   D4  0036                 CLRL    R1                      ; R1 always zero
                                  0038
                                  0038  ;
                                  0038  ; If CHARS_REM <= 0, then DONE
                                  0038  ;
                                  0038
                                  0038  AGAIN:
                         53   D5  0038                 TSTL    R3
                         3E   15  003A                 BLEQ    DONE
                                  003C
                                  003C  ;
                                  003C  ; Get current word for processing
                                  003C  ;
                                  003C
                    50   84   3C  003C                 MOVZWL  (R4)+, R0               ; R0/R1 = current word for processing
                                  003F                                                 ; use quadword for EDIV
                                  003F
                                  003F  ;
                                  003F  ; If overflow then output '???'
                                  003F  ;
                                  003F
           0000F9FF 8F   50   D1  003F                 CMPL    R0, #^O174777           ; 174777(octal) is largest radix 50
                                  0046                                                 ; value in a word
                         07   15  0046                 BLEQ    GOOD                    ; branch if value is good
           50   0000B9E5 8F   D0  0048                 MOVL    #^O134745, R0           ; RAD50 for '???'
                                  004F
                                  004F  ;
                                  004F  ; Do actual conversion
                                  004F  ;
                                  004F
                         53   D7  004F  GOOD:          DECL    R3                      ; dec byte count
                         27   19  0051                 BLSS    DONE                    ; and quit if it was 0
 50   52   50   00000640 8F   7B  0053                 EDIV    #L_50_POWER_2, R0, R2, R0
                                  005C                                                 ; divide current word (n) in R0/R1 by
                                  005C                                                 ; 50(octal)**2,
                                  005C                                                 ; n = q1 * 50(octal)**2 + r1
                                  005C                                                 ; R2 = q1, R0 = r1
               85   A0 AF42   90  005C                 MOVB    ASCII_TAB[R2], (R5)+    ; output corresponding ascii char
                         53   D7  0061                 DECL    R3                      ; dec byte count again
                         15   19  0063                 BLSS    DONE                    ; and branch if it was 0
          50   52   50   28   7B  0065                 EDIV    #^O50, R0, R2, R0       ; q1 = q2 * 50(octal) + r2
                                  006A                                                 ; R0/R1 = q1, R2 = q2, R0 = r2
               85   92 AF42   90  006A                 MOVB    ASCII_TAB[R2], (R5)+    ; output 2nd char
                         53   D7  006F                 DECL    R3                      ; last byte for this RAD50 word
                         07   19  0071                 BLSS    DONE                    ; done if it was 0
               85   89 AF40   90  0073                 MOVB    ASCII_TAB[R0], (R5)+    ; output 3rd char
                         BE   11  0078                 BRB     AGAIN                   ; get next input RAD50 word
                                  007A
```

```
     007A   263 ;
     007A   264 ; DONE
     007A   265 ;
 04  007A   266 DONE:   RET
     007B   267
     007B   268         .END
```

```
AGAIN            00000038 R      01
ASCII_ARRAY    = 0000000C
ASCII_TAB        00000000 R      01
DONE             0000007A R      01
GOOD             0000004F R      01
L_50_POWER_2   = 00000640
NO_OUTPUT_CHAR = 00000004
R50ASC           00000028 RG     01
RADIX50_ARRAY  = 00000008

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

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
. ABS  .                        00000000  (      0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
_F4PCOMPAT$CODE                 0000007B  (    123.)  01 (  1.)    PIC   USR   CON   REL   LCL   SHR   EXE   RD   NOWRT NOVEC BYTE

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

Phase                    Page faults    CPU Time       Elapsed Time
-----                    -----------    --------       ------------
Initialization                   29    00:00:00.10    00:00:01.11
Command processing              105    00:00:00.51    00:00:05.65
Pass 1                           70    00:00:00.71    00:00:02.52
Symbol table sort                 0    00:00:00.01    00:00:00.01
Pass 2                           57    00:00:00.48    00:00:02.52
Symbol table output               2    00:00:00.02    00:00:00.02
Psect synopsis output             3    00:00:00.02    00:00:00.02
Cross-reference output            0    00:00:00.00    00:00:00.00
Assembler run totals            268    00:00:01.85    00:00:11.86

The working set limit was 900 pages.
3027 bytes (6 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 9 non-local and 0 local symbols.
268 source lines were read in Pass 1, producing 10 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$:COMR50ASC/OBJ=OBJ$:COMR50ASC MSRC$:COMR50ASC/UPDATE=(ENH$:COMR50ASC)
```

0178 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

COMASSIGN
LIS

FORIOSDEF
SDL

FORFMT
REQ

FOROPN
REQ

COMR50ASC
LIS

COMERRTST
LIS

FDLTRFVEC
LIS

FORPAR
SDL

FORLIB
REQ

COMERRSET
LIS

COMIRAD50
LIS

FORPROLOG
REQ

FORRTL

FORRCE
SDL

FORMACROS
REQ

FORRTL
MAP

COMFDBSET
LIS

FORERR
SDL

FORDEF
FOR

COMEST
REQ

FORNML
REQ

COMCLOSE
LIS