DECUS

PROGRAM LIBRARY

DECUS NO.	8-93
TITLE	CHEW - Convert Any BCD to Binary, Double Precision
AUTHOR	Louis O. Cropp
COMPANY	Sandia Corporation Albuquerque, New Mexico
DATE	September 14, 1967
FORMAT	

Although this program has been tested by the contributor, no warranty, express or implied, is made by the contributor, Digital Equipment Computer Users Society or Digital Equipment Corporation as to the accuracy or functioning of the program or related program material, and no responsibility is assumed by these parties in connection therewith.

CHEW - CONVERT ANY BCD TO BINARY-
DOUBLE PRECISION

DECUS Program Library Write-up DECUS No. 8-93

ABSTRACT

This subroutine converts a double precision (6 digit) unsigned-integral-binary coded decimal (BCD) number with bit values of 4, 2, 2, and 1 to its integral-positive-binary equivalent in two computer words. It is possible to change the bit values to any desired values and thereby convert any BCD number to binary.

REQUIREMENTS

A. Standard PDP-8 or PDP-8/S

B. Core storage - 0109_{10}

C. Locations 3 and 4 on page zero must be temporarily available for use by this subroutine

D. Location 164 must contain a -4. Location 166 must contain a -6.

SUBROUTINES USED

None

RESTRICTIONS

None

USAGE

Enter this subroutine with a JMS CHEW. The first location following the JMS must contain the address of the most significant part of the BCD number to be converted. Return to the main program will be at JMS+ 2 with the accumulator and link clear. The results are temporarily stored as follows:

C (Location 3) # most significant portion of answer

C (Location 4) # least significant portion of answer

COMMENTS

This subroutine assumes that the number to be converted to binary is a binary coded decimal (BCD) number occupying two 12-bit words. The subroutine then searches by continually rotating the words left starting with the most significant half of the BCD number. When a bit is found, its bit value is multiplied by 12 (octal) the proper number of times. The basis for the conversion routine is that

```
100,000 (decimal) equals 12 (octal) exponent 5
 10,000 (decimal) equals 12 (octal) exponent 4
  1,000 (decimal) equals 12 (octal) exponent 3
    100 (decimal) equals 12 (octal) exponent 2
     10 (decimal) equals 12 (octal) exponent 1
      1 (decimal) equals  1 (octal)
```

Therefore, if we have a BCD number with the bit values equal to 4, 2, 2, and 1, and the following bit construction in two words

```
1000 0000 0000 # most significant half
0000 0000 0000 # least significant half
```

This subroutine multiplies 4 by 12 (octal) five times since the bit is in the hundred thousands place, and adds the resultant binary number by double precision addition to the binary values of any other bits that may be present in the two-word BCD number.

The bit values used by this subroutine can be changed by placing the desired bit values (in octal) in the four locations called store in the program listing, starting with the leftmost bit value in a 4-bit group and proceeding to the right to that the rightmost bit value is contained in store+3 in this manner, any form of BCD number can be converted to binary.

The program expects to find two words full of BCD coded bits as follows.

```
               HT  TT  TH
Location A-  /,,,/,,,/,,,/  -most significant half
Location B-  /,,,/,,,/,,,/  -least significant half
               H   T   U
```

Where HT, TT, TH, H, T, and U represent BCD digits in the hundred thousands, ten thousands, thousands, hundreds, tens, and units places respectively.

Locations A and B must be sequential. If only a 12-bit BCD number (i.e., one word) were to be converted to binary, location A would have to be filled with zeros with the data in the following location. In other words, BCD words ranging in length from 1 to 6 digits can be converted to binary if they are right justified in a double precision word upon entry to this subroutine.

The biggest double precision BCD number (i.e., 999,999) is equal to 3641077 (octal) which does not fill the sign bit in a double precision word. This means that any positive-double precision number in BCD can be translated to a positive-double precision number in binary.

The maximum execution time for this subroutine is approximately 5.31 milliseconds.

The program listing follows.

```
                           /CHEW-CCNVERT ANY BCD TO BINARY-
                           /DOUBLE PRECISICN PAL 00021 A
CHEW,     0
          CLA CLL
          TAD NEG24
          DCA CHK1         /SET COUNTER I#-30 CCTAL
          TAD 164          /-4 IN ACC.
          CCA CHK2         / SET J#-4
          TAD 166          /-6 IN ACC.
          DCA CHK3         / SET L#-6
          DCA CHK4         / SET M#C
          TAD I CHEW       / GET CCNTENTS OF LCC AFTER JMS CHEW INS
                                         T
          CCA ADRS         / STORE LCC OF MOST SIG OF BCD#
          TAD I ADRS       / GET MOST SIG OF BCD# IN ACC
          DCA HOLD         /STORE MOST SIG. OF BCD NO.
          ISZ ADRS         /GET TO ADDRS OF LEAST SIG.BCD NO.
          TAD I ADRS
          CCA HOLD+1       /STORE LEAST SIG.BCD NO.
          CCA 3            / NOW CLEAR THESE TWO LOCATIONS FOR
          DCA 4            /SUMMATICN OF ANSWER
          TAD HOLD         /MOST SIG. OF BCC NO. IN ACC
MID,      DCA TEMPY        /VALUE IN ACC.TO COMMON STORAGE
DO,       CLA CLL          / STORE IT + GET IT BACK
          TAD TEMPY
          RAL
          DCA TEMPY        /STORE RCTATED #
          SNL              /IS THERE A BIT
          JMP INCJ         / NO INCREMENT CCUNTERS
          TAD I TABLE      /YES ADD CNE OF THE BIT VALUES
          CCA CUM+1        / PUT IT IN LEAST SIG OF MULTIPLICATIONS
AGN,      ISZ CHK4         / M#M+1
          TAD CHK4         / GET M
          TAC CHK3         / M+(-L)
          SZA              / IS M#L
          JMP MLTPY        / NO GC MULTIPLY
          CLL CLA
          TAC CUM+1        / YES GET LEAST SIG RESULT OF MULTIPLY
          TAD 4            /ADD IT TO LEAST SIG.OF BINARY NO.
          DCA 4            /STORE RESULT IN BINARY NO.LOCATICN
          RAL              /  ADD
          TAC CUM          /  MOST SIG
          TAD 3            /HALVES
          CCA 3            /STORE FCR POSSIBLE EXIT
          DCA CUM          /CLEAR FCR RESULT OF NEXT MULTIPLY
          DCA CHK4         / SET M#0
          JMP INCJ         / GO INCREMENT CCUNTERS
MLTPY,    JMS MULT         /GO TO MULTIPLY BY 12 SUBROUTINE
```

```
                JMP AGN
INCJ,           ISZ TABLE       / STORE#STORE+1
                ISZ CHK2        / J#J+1%0
                JMP INCI        / NO INCREMENT I
        PAGEBRK
                TAD 164         /-4 IN ACC
                DCA CHK2        / RESET J TO -4
                TAD RESET
                DCA TABLE       / PUT STCRE BACK TO INITIAL VALUE
                IAC             /+1 IN ACC
                TAD CHK3        / L+1
                DCA CHK3        / L#L+1
INCI,           ISZ CHK1        / I#I+1#0
                JMP .+4         /NO.GO SEE IF I EQUALS -14 OCTAL.
                ISZ CHEW        / INCREMENT RETURN LOC TO MAIN PROG
                CLL             /CLEAR FOR EXIT
                JMP I CHEW      / JUMPS BACK TC MAIN PROG PAST %LOC
                TAD CHK1        / ADD I TO ACC
                TAD FRTN        /ADD +14 OCTAL TO ACC
                SZA CLA         / DOES I#-12
                JMP DO          / NO,GO BACK AND LOOK FOR ANOTHER BIT
                TAD HOLD+1      /YES.GET LEAST SIG.BCD NO. IN ACC
                JMP MID         / PUT LEAST SIG OF BCD#IN TEMPY
MULT,           0               /SUBROUTINE TO MULTIPLY THE VALUE
                CLA CLL         /IN CUM BY 12 OCTAL.
                TAD NEG11       /SET COUNTER TO---
                DCA COUNT       /NIMUS 11 OCTAL.
                TAD CUM+1       /RESTORE STARTING VALUE SO IT CAN---
                DCA NOW+1       /BE ADDED TO ITSELF 12 TIMES(OCTAL--
                TAD CUM         /AND STILL RETAIN THE RUNNING TOTAL--
                DCA NOW         /IN CUM AND CUM+1.
REPEAT,         CLL
                TAD CUM+1       /ADD LEAST---
                TAD NOW+1       /SIG.PARTS.
                DCA CUM+1
                RAL             /ADD ANY OVERFLOW IN LINK---
                TAD CUM         /TO THE SUM OF THE MOST SIGS.
                TAD NOW
                DCA CUM
                ISZ COUNT       /IF ZERO,NO.HAS BEEN ADDED TO ITSELF---
                JMP REPEAT      /12 OCTAL TIMES.
                CLL
                JMP I MULT      /EXIT TO MAIN PROGRAM
NEG24,          -30             /OCTAL NO.
CHK1,           0               / COUNTER I-SEE FLOWCHART
CHK2,           0               / COUNTER J
CHK3,           0               / COUNTER L
CHK4,           0               / COUNTER M
```

```
        TEMPY,     0
        ADRS,      0                  / LOCATION OF BCD#
        CUM,       0                  / HOLDS RESULTS OF MULTIPLY(X.12Y)
                   0                  / WHERE X#4,2,2,OR1 Y#1THRU5
        TABLE,     STORE              / MEANS TO GET STORE#STORE+1 + TO RESET
                                                IT
        RESET,     STORE              /MEANS TO RESET START ADDRS OF BITS.
        FRTN,      14                 /OCTAL NO.
        STORE,     4                  /OCTAL BIT VALUES OF THE 4-BIT GROUPS---
                   2                  /MAKING UP AN INDIVIDUAL BCD DIGIT---
                   2                  /STARTING WITH THE LEFTMOST BIT---
                   1                  /VALUE AND PROCEEDING RIGHT.
        HOLD,      0
                   0
        NOW,       0
                   0
        NEG11,     7767
        COUNT,     0
               PAGEBRK
                   PAUSE
        $
```