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

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 = 010910
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.

```
<table>
<thead>
<tr>
<th>Location A-</th>
<th>Location B-</th>
</tr>
</thead>
<tbody>
<tr>
<td>HT TT TH</td>
<td>HT TH TT</td>
</tr>
</tbody>
</table>

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.
COMPUTER PROGRAM PAL 000201 A

/CHEW—CONVERT ANY BCD TO BINARY-
/DIRECT PRECISION PAL 000201 A

CHEW

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 CONTENTS OF LCC AFTER JMS CHEW INS

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 ADDRES OF LEAST SIG. BCD NO.
TAD I ADRS
DCA HOLD+1 /STORE LEAST SIG. BCD NO.
CCA 3 / NOW CLEAR THESE TWO LOCATIONS FOR
DCA 4 /SUMMATION OF ANSWER
TAD HOLD /MOST SIG. OF BCD NO. IN ACC

DO, CUM+1 /VALUE IN ACC. TO COMMON STORAGE

DCA TEMPL/ STORE IT + GET IT BACK

TAD TEMPLY

DCA TEMPL /STORE RCTATED #
SNL /IS THERE A BIT
JMP INCJ / NO INCREMENT COUNTERS
TAD I TABLE /YES ADD ONE OF THE BIT VALUES

AGN,

CCA CUM+1 / PUT IT IN LEAST SIG OF MULTIPLICATIONS
ISZ CHK4 / M#M+1
TAD CHK4 / GET M
TAD CHK3 / M+(-L)
SZA / IS M#L
JMP MLTPY / NO GC MULTPLY

CLL CLA
TAD 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. LOCATION
RAL / ADD
TAD CUM / MOST SIG
TAD 3 / HALVES
CCA 3 / STORE FOR POSSIBLE EXIT

DCA CUM / CLEAR FOR RESULT OF NEXT MULTIPLY
DCA CHK4 / SET M#0
JMP INCJ / GO INCREMENT COUNTERS

MLTPY, JMS MULT /GO TO MULTIPLY BY 12 SUBROUTINE
COMPUTER PROGRAM

INCJ, ISZ TABLE / STORE #STORE+1
INCJ, ISZ CHK2 / J#J+1#0
INCJ, JMP INC1 / NO INCREMENT I

PAGE BRK
TAD 164 / -4 IN ACC
DCA CHK2 / RESET J TO -4
TAD RESET
DCA TABLE / PUT STORE BACK TO INITIAL VALUE
IAC / +1 IN ACC
TAD CHK3 / L+1
DCA CHK3 / L#L+1
INC1, JMP +4 / NO, GO SEE IF I EQUALS -14 OCTAL.
INC1, ISZ CHEW / INCREMENT RETURN LOC TO MAIN PROG
C L L / CLEAR FOR EXIT
INC1, JMP I CHEW / JUMPS BACK TO MAIN PROG PAST %LOC
TAD CHK1 / ADD I TO ACC
TAD FRN / ADD +14 OCTAL TO ACC
SZA CLA / DOES I#-12
JMP DG / NO, GO BACK AND CHECK 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 CL L / 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, CL L
TAD CUM+1 / ADD LEAST--
TAD NOW+1 / SIG. PARTS.
DCA CUM+1
R AL / ADD ANY OVERFLOW IN LINK--
TAD C UM / TO THE SUM OF THE MOST SIGS.
TAD NOW /--
DCA C UM
ISZ COUNT / IF ZERO, NO. HAS BEEN ADDED TO ITSELF--
JMP REPEAT /12 OCTAL TIMES.
C L L
JMP I MUL T / EXIT TO MAIN PROGRAM
NEG24, -30 / OCTAL NO.
CHK1, 0 / COUNTER I-SEF FLOWCHART
CHK2, 0 / COUNTER J
CHK3, 0 / COUNTER L
CHK4, 0 / COUNTER M
COMPRESS

TEMPY,  0
ADRS,  0 / LOCATION OF BCD# 
CUM,   0 / HOLDS RESULTS OF MULTIPLY(X,Y)
TABLE, STORE / WHERE X=4,2,2, OR Y=011

RESET, STORE / MEANS TO GET STORE#STORE+1 + TO RESET IT
FRTN, 14 / MEANS TO RESET START ADDRS OF BITS.
STORE, 4 / OCTAL NO.
    2 / OCTAL BIT VALUES OF THE 4-BIT GROUPS---
    2 / MAKING UP AN INDIVIDUAL BCD DIGIT---
    1 / STARTING WITH THE LEFTMOST BIT---

NOW,  0 / VALUE AND PROCEEDING RIGHT.

NEG11, 7767
COUNT, 0

PAGEBRK
PALSE
$