DATA GENERAL
CORPORATION
Southboro,
Massachusetts 01772
(617) 485-9100

PROGRAM

Double Precision Binary to ASCII Decimal

TAPES

ASCII Source: 090-000036

ABSTRACT

This routine converts a double precision, two's complement binary number to a string of ASCII characters representing its equivalent signed decimal value.

 093-000032-01

1. REQUIREMENTS

1.1 Memory

1K or larger alterable memory.

1.2 Equipment

NOVA central processor.

1.3 External Subroutines

A user supplied routine for accepting the ASCII output characters (see 2.3) must be provided.

1.4 Other

None.

2. OPERATING PROCEDURE

2.1 Calling Sequence

```
JSR  .DBD
return
```

2.2 Input Format

A double precision, two's complement integer is passed in AC1 (high order) and AC2 (low order).

2.3 Output Format

The output is an ASCII character string of the form

+DDDDDDDDDD(null)

or -DDDDDDDDDD(null)

The user must supply a routine that accepts these output characters. The address of this routine must be stored by the user in location 41 of page zero. The characters will be passed one at a time, right adjusted in AC$\emptyset$ (bit 8 = $\emptyset$).

Twelve characters will be passed; the sign, most significant digit, . . ., least significant digit, null (all Ø). The user routine need only save AC3 (if used). Return to the conversion routine should be made by a JMP Ø,3.

2.4 Error Returns

None.

2.5 State of Active Registers upon Exit

ACØ remains unchanged. AC1, AC2, AC3, and Carry are destroyed.

2.6 Cautions to User

None.

3. DISCUSSION

3.1 Algorithms

The sign of the result is determined and passed to the user output routine. Conversion is then made on the absolute value of the input. The principle of the algorithm is to determine how many 1,ØØØ,ØØØ,ØØØ's, 1ØØ,ØØØ,ØØØ's, . . ., 1's are contained in the number. This can be calculated by successively subtracting the appropriate double precision power of ten from the original value until the result is negative. Each subtraction that gives a result greater than or equal to zero causes octal 6Ø (ASCII Ø) to be incremented. When the result becomes negative, its previous value is restored, the ASCII digit is passed to the user, and the next lower power of ten is used to compute the next digit. After 1Ø**Ø is used, the conversion is complete.

3.2 Limitations and Accuracy

The routine is exact for all 32-bit, two's complement numbers. The range, therefore, is

$$-2,147,483,648 \leq N < +2,147,483,648$$

3.3 Size and Timing

The routine is 112 (octal) words in length.

Execution time is approximately

1.Ø61 + N * .Ø47 milliseconds

where N is the sum of the digits of the result. For example, the number 156,8Ø4,319 requires

1.Ø61 + 37 * .Ø47 = 2.8 milliseconds.

3.4 References

Write-up 093-000026-00 describes single precision binary to decimal which uses the same basic algorithm.

3.5 Flow Diagrams

None.

4. EXAMPLES AND APPLICATIONS

The ASCII source of .DBD is provided with the NOVA software. If a user routine requires double precision binary to decimal, the tape should be edited into the user's source.

5. PROGRAM LISTING

A listing of .DBD follows. No origin is given in the source, enabling the tape to be edited anywhere within a user routine.

```
                ; BINARY TO DECIMAL ASCII CONVERT
                ; CONVERTS A DOUBLE PRECISION, TWO'S COMPLEMENT NUMBER
                ; TO AN ASCII DECIMAL CHARACTER STRING

                ; INPUT:         D IN AC1, AC2 (HIGH, LOW)

                ; OUTPUT:        ASCII CHARACTER STRING, TERMINATED BY A
                ;                NULL WORD.
                ;                CHARACTERS PASSED RIGHT ADJUSTED,
                ;                BIT 8 = 0, IN AC0 TO USER
                ;                ROUTINE WHOSE ADDRESS MUST BE
                ;                STORED IN LOCATION 41 OF PAGE 0

                ;                STRING OF FORM:
                ;                        +NNNNNNNNNN(NULL)
                ;                OR      -NNNNNNNNNN(NULL)

                ; CALLING SEQUENCE
                ;       JSR     .DBD
                ;       RETURN

                ; DESTROYED:    AC1, AC2, AC3, CARRY
                ; UNCHANGED:    AC0

00000 054054 .DBD:  STA 3,.FD03     ; SAVE RETURN
00001 040053        STA 0,.FD00     ; SAVE AC0
00002 020101        LDA 0,.FD30     ; POINT TO HIGH ORDER POWER IN
                                    ; TABLE
00003 040106        STA 0,.FD12
00004 020107        LDA 0,.FD20     ; ASSUME "+"
00005 125113        MOVL# 1,1,SNC
00006 000013        JMP .FD99       ; YES, WAS POSITIVE
00007 150404        NEG 2,2,SZR     ; NO, NEGATIVE
00010 124001        COM 1,1,SKP
00011 124400        NEG 1,1
00012 020110        LDA 0,.FD21     ; GET "-"
00013 044103 .FD99: STA 1,.FD10     ; SAVE ABS(NUMBER)
00014 050104        STA 2,.FD10+1
00015 006041        JSR @.FD40      ; PUT OUT SIGN OR DIGIT
00016 024103        LDA 1,.FD10     ; RESTORE ABS(NUMBER)
00017 030104        LDA 2,.FD10+1
00020 020111        LDA 0,.FD22     ; GET OCTAL 57
00021 040105        STA 0,.FD11     ; COUNT IT UP IN STORAGE
00022 034106        LDA 3,.FD12     ; CURRENT POINTER TO POWER OF
                                    ; 10 TABLE


00023 021401 .FD98: LDA 0,1,3       ; LOW ORDER WORD
00024 101005        MOV 0,0,SNR     ; TEST FOR END OF TABLE
00025 000050        JMP .FD97       ; DONE
00026 112420        SUBZ 0,2
00027 021400        LDA 0,0,3       ; HIGH ORDER WORD
00030 101003        MOV 0,0,SNC
00031 106001        ADC 0,1,SKP
00032 106400        SUB 0,1
00033 010105        ISZ .FD11       ; COUNT UP DIGIT
00034 125113        MOVL# 1,1,SNC   ; TEST FOR <0
00035 000023        JMP .FD98       ; KEEP SUBTRACTING
```

```
---
???

00036 021401            LDA 0,1,3       ; RESTORE POSITIVE VALUE
00037 113022            ADDZ 0,2,SZC
00040 125400            INC 1,1
00041 021400            LDA 0,0,3
00042 107000            ADD 0,1
00043 175400            INC 3,3         ; BUMP AC3 TO NEXT TABLE ENTRY
00044 175400            INC 3,3
00045 054106            STA 3,.FD12
00046 020105            LDA 0,.FD11     ; GET DIGIT
00047 000013            JMP .FD99       ; PUT IT OUT


00050 006041 .FD97:     JSR 0.FD40      ; PUT OUT NULL
00051 020053            LDA 0,.FD00     ; RESTORE AC0


00052 002054            JMP 0.FD03      ; RETURN

00053 000000 .FD00:     0               ; SAVE AC0
00054 000000 .FD03:     0               ; SAVE RETURN

00055 035632 .FD05:     35632           ; 10**9
00056 145000            145000
00057 002765            2765            ; 10**8
00060 160400            160400
00061 000230            230             ; 10**7
00062 113200            113200
00063 000017            17              ; 10**6
00064 041100            41100
00065 000001            1               ; 10**5
00066 103240            103240
      000012            .RDX 10
00067 000000            0               ; 10**4
00070 023420            10000
00071 000030            0               ; 10**3
00072 001750            1000
00073 000000            0               ; 10**2
00074 000144            100
00075 000000            0               ; 10**1
00076 000012            10
00077 000000            0               ; 10**0
00100 000001            1
00101 000055 .FD30:     .FD05   ; POINTER TO CONVERSION TABLE
00102 000000            0       ; END OF TABLE INDICATION
      000010            .RDX 8

      000002 .FD10:     .BLK 2          ; SAVE CURRENT DOUBLE WORD
00105 000000 .FD11:     0               ; COUNT UP DIGIT WORD
00106 000000 .FD12:     0               ; POINTER TO POWER OF TEN ENTRY

00107 000053 .FD20:     "+              ; ASCII "+"
00110 000055 .FD21:     "-              ; ASCII "-"
00111 000057 .FD22:     57              ; ASCII "0" -1


      000041 .FD40=41                   ; PAGE 0 PUT CHARACTER ADDRESS
```