Electronic components and materials

PHILIPS

laboratory report

central application laboratory CAB
eindhoven - the netherlands

number:	EDP 7612	date:	06.01.1977
project:	6631	pages:	S 1 ; A ; R 16

title LOGARITHMIC ROUTINE FOR THE 2650 MICROPROCESSOR

author E.D. van Veldhuizen

summary This report describes a possible routine for calculating the $^{e}\log$ (ln) of BCD floating point numbers by means of the CORDIC algorithm. The format of the floating point numbers (input number as well as result) is the same as used in the arithmetic routines described in EDP 7603 by A.F. Craens. These arithmetic routines are used in the log routine and must be present in the program store to execute the log function.

E.D. van Veldhuizen

Advies Octrooi d.d. 31 jan. 1977	X AV	GV		B		BL
Opgave Mamo d.d. 31 jan. 1977	AV	X GV	X SP	B		BL
Datum: 11 jan. 1977	Mamo					

CONTENTS: PAGE

1. INTRODUCTION

The e log of a BCD floating point number is calculated using the CORDIC algorithm as described in references nr. 2 and nr. 3.

The interfaces to the program which uses the log routine are such, that calling the routine gives as little overhead as possible in time as well as in program store.

Also the status of the calling program is saved.

In describing the function, the algorithm is given in BASIC. Boundaries and scaling of input numbers are indicated. Accuracy and timing are given to indicate the performance of the routine.

Finally the actual program is described in flowcharts and given in assembly language.

2. REFERENCES

2.1. BCD Floating point arithmetic for the 2650 microprocessor by A.F. Craens. EDP 7603.

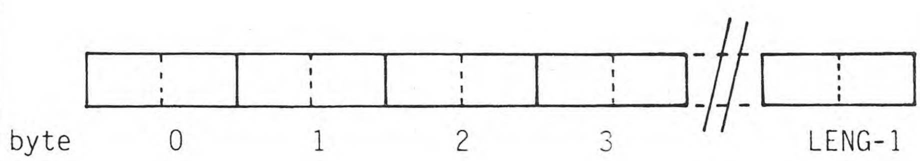
2.2. The CORDIC trigonometric computing Technique by J.E. Volders IRE transactions on electronic computers, September 1959.

2.3. Use decimal CORDIC for generation of many transconducted functions by H. Schmidt and A. Bogacki. EDN, February 20, 1973.

3. INTERFACES

3.1. Format

The format of the BCD floating point numbers as given in reference nr. 1, is as follows:



Byte 0: sign of exponent (+=H'000, (-=H'F0')
1: exponent (absolute, 2 digits)
2: sign of mantissa (H'00' or H'F0')
3: two most significant digits of mantissa
LENG-1: two least significant digits of mantissa

The length of the floating point number can be set by defining the LENG constant in the BCD arithmetic floating point package. The length of the mantissa will be (LENG-3) * 2 digits. The exponent always has two digits. The input number should always be left-normalized.

3.2. Input/Output variables

When calling the log routine, R0 and R1 of the selected bank should contain the address of byte 0 of the input number. R0 holds the most significant part and R1 the least significant part of the address.
R2 and R3 should contain the address of byte 0 of the result location. R2 holds the most significant part and R3 the least significant part of the address.
Register select is don't care.

3.3. PSW and registers

When entering the log routine the current PSL is saved and before returning, it will be restored including the condition code. PSU is not affected.

The registers belonging to the selected bank and R0 are changed and are not restored.
The registers of the bank which was not selected are not affected.

3.4. Error signalling

When an error occurs, because a negative or zero number appears as an operand, a jump is made to location ERR. However, no return from subroutine has been made before the jump to ERR.
The location ERR has to be defined in the main program.

4: FUNCTION

When the log routine is entered, input and output addresses as present in R0, R1, R2 and R3 are stored and the Program Status Lower is saved.
Then the input number is scaled up or down to bring it in the region between +0.1 and +1. For each multiplication by 10 (increment of exponent by 1), ln 10 is subtracted from the result output.
For each division by 10 (decrement of exponent by 1), ln 10 is added to the result output.
Also the number is screened whether it is negative, zero or one. If negative or zero, the routine jumps to location "ERR", which should be defined by the main program, and no return from subroutine is made.
If the number is 1 (after scaling in fact. 0.1), ln. 10 is subtracted from the result output and a return is made.

For the purpose of the algorithm as given below, a table T of 8 constants has been loaded and forms a part of the program. For each item of the table applies:
T(I) = tanh (T) where T initially is 0.1 and for the next I T = T/10.
The actual log calculation is done by means of the following algorithm.

```
        Input number is A.
1000    X  = A-1
        Y  = A+1
        U  = 0
        Z  = 0.1
        J9 = 22
        N  = 8
        For I = 1 to N
        For J = 1 to J9
        If Y > 0 then go to P
        If X < 0 then go to Q
3000  S U = U - T(I)
        X1 = X + Y * Z
        Y1 = Y + X * Z
        Go to R
4000  P If X < 0 go to S
5000  Q U = U + T(I)
        X1 = X - Y * Z
        Y1 = Y - X * Z
6000  R X = X1
        Y = Y1
        Next J
        Z = Z/10
        J9 = 9
        Next I
        Output = U+U
        End
```

This algorithm follows from the description given in reference 3. It is implemented in assembly language as indicated in the flowcharts.

First the operand is brought in the range of 0.1 ⟶ 1.0.

Each time the exponent is increased by 1,ln 10 is subtracted from the output result.

If the exponent is decreased, ln 10 is added to the output result. Addition or subtraction of U and T(I) is determined by the signs of X and Y being equal or not.

5. PERFORMANCE

5.1. Accuracy

The accuracy of the calculated result has been investigated by taking 100 log calculations (log+00+01000000 ---- log+01+10000000).

On average the error is less than 1 in 3.10^6.

Worst case the error could be 1 in 4.10^3.

5.2. Timing

The execution time of the log routine ranges from 4 ms (log+01+10000000) to about 6 seconds (log+90+99999999).

The average execution time is about 2 - 2.5 seconds.

Accuracy and timing have been measured with 8-digit mantiosa.

With less digits the accuracy will detoriate but the timing will improve.

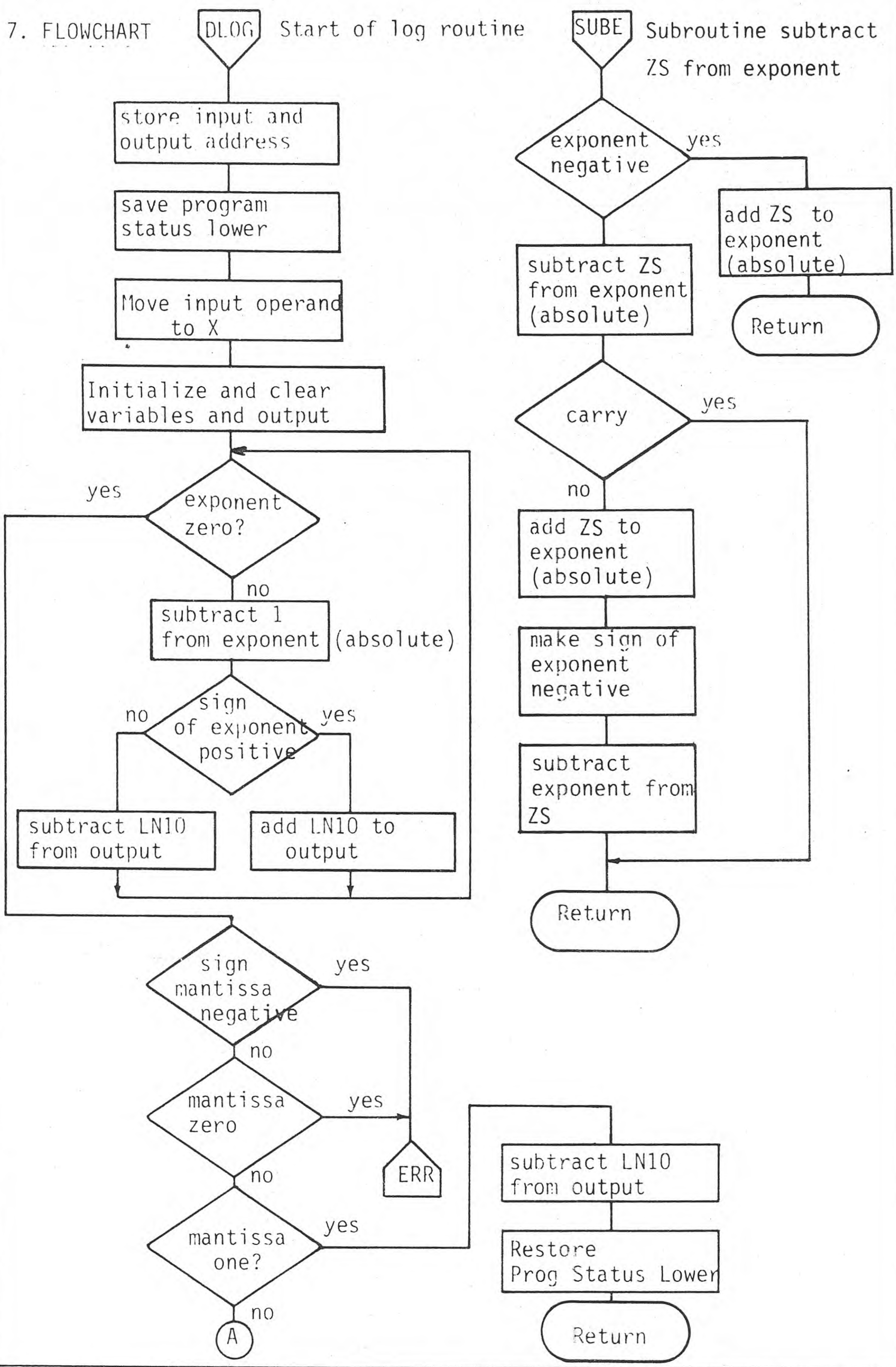
6. REMARKS

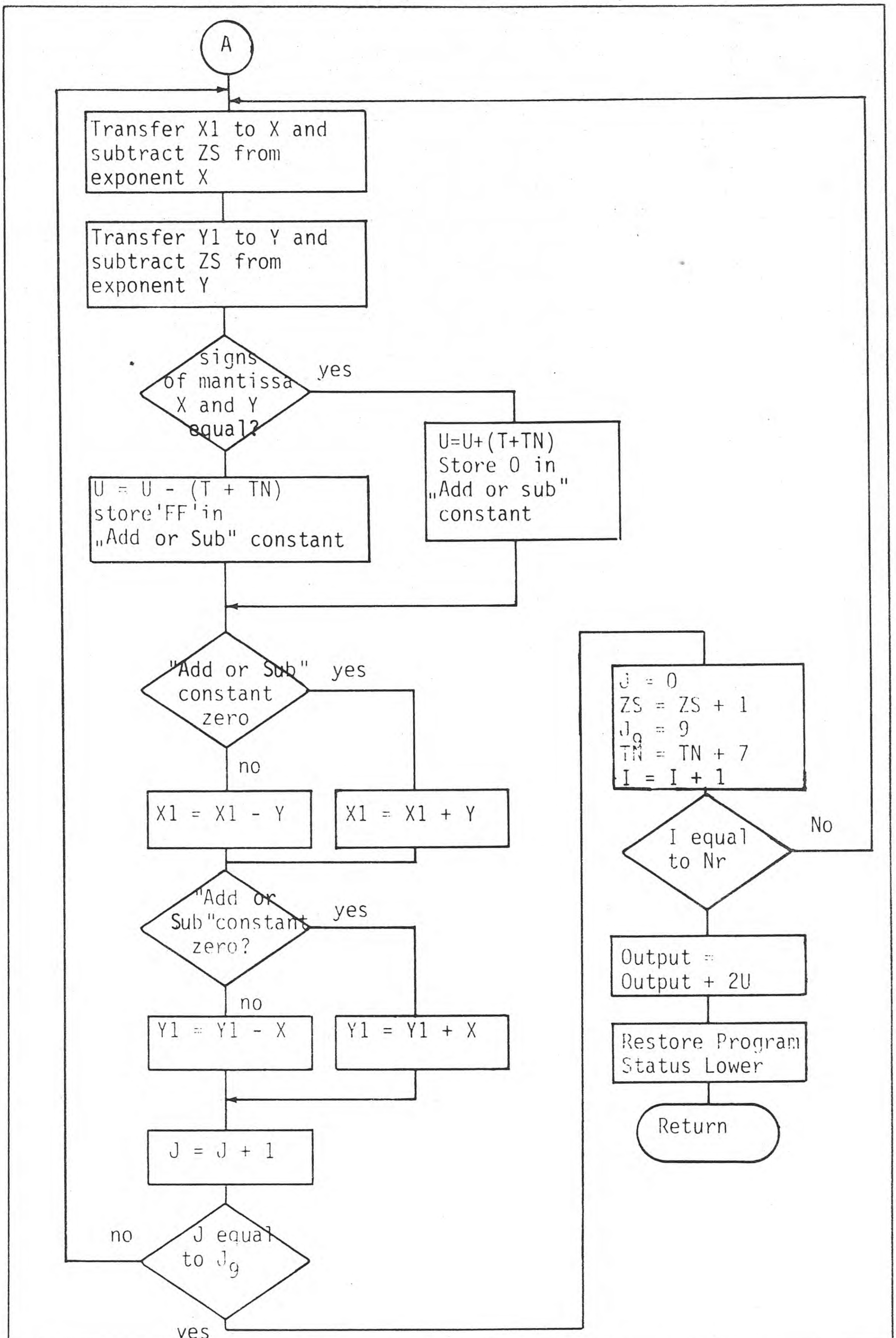
The CORDIC algorithm and existing floating point routines were used on request from Signetics. The present routine was made as an exercise to get some indication about execution time and accuracy. As is shown in section 5, this approach does not seem to be very suitable.

Therefore is is recommended to investigate some other possibilities (other algorithms, binary calculation and conversion to B.C.D. etc.).

../GS

7. FLOWCHART


DLOG
Start of log routine
store input and output address
save program status lower
Move input operand to X
Initialize and clear variables and output
exponent zero?
yes
no
subtract 1 from exponent
(absolute)
sign of exponent positive
no
yes
subtract LN10 from output
add LN10 to output
sign mantissa negative
yes
no
mantissa zero
yes
no
ERR
mantissa one?
yes
no
A
subtract LN10 from output
Restore Prog Status Lower
Return
SUBE
Subroutine subtract ZS from exponent
exponent negative
yes
add ZS to exponent (absolute)
Return
subtract ZS from exponent (absolute)
carry
yes
no
add ZS to exponent (absolute)
make sign of exponent negative
subtract exponent from ZS
Return


A
Transfer X1 to X and subtract ZS from exponent X
Transfer Y1 to Y and subtract ZS from exponent Y
signs of mantissa X and Y equal?
yes
U=U+(T+TN) Store 0 in „Add or sub" constant
U = U - (T + TN) store 'FF' in „Add or Sub" constant
"Add or Sub" constant zero
yes
no
X1 = X1 - Y
X1 = X1 + Y
"Add or Sub"constant zero?
yes
no
Y1 = Y1 - X
Y1 = Y1 + X
J = J + 1
no
J equal to J9
yes
J = 0
ZS = ZS + 1
J9 = 9
TN = TN + 7
I = I + 1
I equal to Nr
No
Output = Output + 2U
Restore Program Status Lower
Return

date:

2650 MICROPROCESSOR	ref.no.

Program title	BCD logarithmic routine
Function	Calculating the $^{e}\log$ of a BCD floating point number Ln (input operand)= output result
Parameters: input	The address of byte 0 of the input operand is in R0 and R1. The address of byte 0 of the output results is in R2 and R3.
output	Result of $^{e}\log$ in location pointed to by R2, R3.
Requirements: hardware	None
software	Availability of BCD floating point arithmetic package.

Modified in 2650								Max. Subroutine Nesting Level	1
Registers	R0 [X]	R1 [X]	R2 [X]	R3 [X]	R1' [ ]	R2' [ ]	R3' [ ]	Assembler	
PSU	F [ ]	II [ ]	SP [ ]					Memory requirement (bytes)	
PSL	CC [ ]	IDC [ ]	RS [ ]	WC [ ]	OVF [ ]	COM [ ]	C [ ]	RAM	38
								ROM	582

Central application laboratory CAB Eindhoven - The Netherlands

PHILIPS

```
TWIN ASSEMBLER VER 2.0    BCD LOGARITHM ROUTINE  (EDP 7612)    PAGE 0015

LINE ADDR  OBJECT  E SOURCE

0584                  *
0585                  * EDMV 761123-1400
0586                  ***************************
0587                  * BCD LOGARITHMIC ROUTINE *
0588                  ***************************
0589                  *
0590                  * THIS ROUTINE CALCULATES THE FLOG OF AN INPUT
0591                  * NUMBER AND STORES THE RESULT IN AN OUTPUT AREA
0592                  *
0593                  * CALLING NAME IS FLOG
0594                  * INPUT  ADDRESS MUST BE GIVEN IN R0,R1
0595                  * OUTPUT ADDRESS MUST BE GIVEN IN R2,R3
0596                  *
0597                  * THE ROUTINE USES BCD FLOATING POINT PACKAGE (REPORT EDP7603)
0598                  * FORMAT OF INPUT AND OUTPUT NUMBER ARE AS REQUIRED
0599                  * FOR THIS PACKAGE
0600                  *
0601                  * IF THE INPUT IS NEGATIVE OR ZERO, A BRANCH IS
0602                  * MADE TO LOCATION ERR
0603                  *
0604                  ***********************************************************
0605 07D9                    ORG     H'800
0606                  *
0607 0008             NR     EQU     8          * NUMBER OF OUTER LOOPS
0608                  *
0609                  * TABLE OF CONSTANTS *
0610                  *
0611 0800 00000010    T      DATA    H'00.00.00.10.03.25.75'
     0804 032575
0612 0807 F0010010           DATA    H'F0.01.00.10.00.03.33'
     080B 000333
0613 080E F0020010           DATA    H'F0.02.00.10.00.00.03'
     0812 000003
0614 0815 00000000           DATA    H'00.00.00.00.01.00.00'
     0819 010000
0615 081C 00000000           DATA    H'00.00.00.00.00.10.00'
     0820 001000
0616 0823 00000000           DATA    H'00.00.00.00.00.01.00'
     0827 000100
0617 082A 00000000           DATA    H'00.00.00.00.00.00.10'
     082E 000010
0618 0831 00000000           DATA    H'00.00.00.00.00.00.01'
     0835 000001
0619 0838 00010010    ONE    DATA    H'00.01.00.10.00.00.00'
     083C 000000
0620 083F 00000000    U      DATA    H'00.00.00.00.00.00.00'
     0843 000000
0621 0846 00010023    LN10   DATA    H'00.01.00.23.02.58.51'
     084A 025851
```

8222 031 76361

Electronic components and materials

```
TWIN ASSEMBLER VER 2.0    BCD LOGARITHM ROUTINE  FOR 7612      PAGE 0016

LINE ADDR  OBJECT  E SOURCE

                     * VARIABLES *
                     *
     084D            I       RES    1         * INNER LOOP COUNTER
     084E            I9      RES    1         * NR OF CYCLES (INITIALLY 22)
     084F            I       RES    1         * OUTER LOOP COUNTER
     0850            TN      RES    1         * NEXT TABLE ITEM
     0851            ZS      RES    1         * EXPONENT DECREMENT NR
     0852            ADS     RES    1         * TEST LOC  ADD OR SUBTR
     0853            PSL     RES    1         * STORE LOCATION PSL
     0854            X       RES    7
     085B            X1      RES    7
     0862            Y       RES    7
     0869            Y1      RES    7
     0870            OUTR    RES    2
                     *
                     * SUBROUTINE LOAD POINTERS
                     * PTR1 FROM R0 R1.  PTR2 FROM R2 R3
                     * SPT2 = PTR1 + 2
                     *
     0872 CC0440     STPT    STRA R0 PTR1
     0875 CD0441             STRA R1 PTR1+1
     0878 CE0442             STRA R2 PTR2
     087B CF0443             STRA R3 PTR2+1
     087E CE0444             STRA R2 SPT2
     0881 8702               ADDI,R3 2
     0883 CF0445             STRA R3 SPT2+1
     0886 17                 RETC,UN
                     *
                     * SUBROUTINE SUBTRACT ZS FROM EXPONENT
                     * R2 CONTAINS SIGN. R3 CONTAINS EXPONENT
                     * CONDITION CODE ACCORDING TO CONTENTS OF R2
                     *
     0887 1807       SUBE    BCTR,Z    EPOS     JUMP IF EXPONENT POSITIVE
     0889 8766               ADDI R3   H'66     IF NEGATIVE ADD ZS
     088B 8F0851             ADDA,R3   ZS             TO EXPONENT
     088E 97                 DAR R3
     088F 17                 RETC,UN
     0890 AF0851     EPOS    SUBA R3   ZS       EXPONENT POSITIVE
     0893 97                 DAR,R3             SUBTRACT Z FROM EXPONENT
     0894 B501               TPSL      H'01     TEST CARRY
     0896 14                 RETC,R1            IF CARRY  RETURN
     0897 8766               ADDI,R3   H'66     CALCULATE ORIGINAL EXPONENT
     0899 8F0851             ADDA,R3   ZS
     089C 97                 DAR,R3
     089D 06F0               LODI R2   H'F0'    CHANGE SIGN OF EXPONENT
     089F 0C0851             LODA,R0   ZS       SUBTRACT EXPONENT FROM ZS
     08A2 A3                 SUBZ      R3
     08A3 C3                 STRZ      R3
     08A4 17                 RETC,UN
```

```
TWIN ASSEMBLER VER 2.0    BCD LOGARITHM ROUTINE (EDP 7612)    PAGE 0017

LINE ADDR  OBJECT  E SOURCE

                   * START LOGARITHMIC ROUTINE
                   *
     08A5 CC0862   DLOG   STRA,R0 Y          * STORE INPUT ADDRESS
     08A8 CD0863          STRA,R1 Y+1
     08AB CE0870          STRA,R2 OUTP       * STORE OUTPUT ADDRESS
     08AE CF0871          STRA,R3 OUTP+1
     08B1 13              SPSL               * SAVE PROGAM STAT. LOWER
     08B2 CC0857          STRA,R0 PSL
     08B5 0507            LODI,R1 LENG       * MOVE INPUT TO X
     08B7 0D0862   MVIN   LODA,R0 *Y,R1,-      AS TEMPORARY STORE
     08BA CD6854          STRA,R0 X,R1
     08BD 5978            BRNR,R1 MVIN
                   * INITIALIZE VARIABLES
     08BF 20              EORZ    R0
     08C0 CC084D          STRA,R0 I          * I = 0
     08C3 CC084F          STRA,R0 J          * J = 0
     08C6 CC0850          STRA,R0 TN         * TN = 0
     08C9 0401            LODI,R0 1
     08CB CC0851          STRA,R0 TS         * TS = 1
     08CE 0416            LODI,R0 22
     08D0 CC084E          STRA,R0 IS
     08D3 0707            LODI,R3 LENG       * CLEAR U
     08D5 20              EORZ    R0
     08D6 CF483F   CLRU   STRA,R0 U,R3,-
     08D9 5B7B            BRNR,R3 CLRU
                   *
     08DB 0507            LODI,R1 LENG       * CLEAR OUTPUT
     08DD 20              EORZ    R0
     08DE CD0870   CLRO   STRA,R0 *OUTP,R1 -
     08E1 597B            BRNR,R1 CLRO
                   *
                   * DECREASE EXPONENT TILL 00
                   * ADD ( IF EXP POSITIVE) OR
                   * SUBTRACT (IF EXP NEGATIVE)
                   * LN10 TO OR FROM OUTP
                   *
     08E3 7508     DECE   CPSL    WC         * WITHOUT CARRY
     08E5 0501            LODI,R1 1          * LOAD INDEX FOR EXPONENT
     08E7 0D6854          LODA,R0 X,R1       * LOAD EXPONENT
     08EA 1828            BCTR,Z  CONT       * IF ZERO CONTINUE
     08EC A401            SUBI,R0 1          * SUBTRACT 1 FROM EXPONENT
     08EE 94              DAR,R0             * DECIMAL ADJUST
     08EF CD6854          STRA,R0 X,R1       * STORE DECREASED EXPONENT
     08F2 0C0870          LODA,R0 OUTP       * LOAD POINTERS FOR
     08F5 0D0871          LODA,R1 OUTP+1       ADDING OR SUBTRACTING
     08F8 CC0446          STRA,R0 PTRR         LN10 TO OR FROM OUTP
     08FB CD0447          STRA,R1 PTRR+1
     08FF 0609            LODI,R2 <LN10
     0901 074F            LODI,R3 >LN10
     0903 3F0972          BSTA,UN STPT
```

```
LINE ADDR  OBJECT  E SOURCE

0905 0C0854             LODA,R0 X       * LOAD SIGN EXPONENT
0908 1805               BCTR,Z  LADD    * IF POSITIVE ADD
090A 3F0469             BSTA,UN DSUB
090D 1B54               BCTR,UN DECE
090F 3F0470     LADD    BSTA,UN DADD
0912 1B4F               BCTR,UN DECE
0914 20         CONT    EORZ    R0      * MAKE SIGN OF
0915 CC0854             STRA,R0 X         EXPONENT POSITIVE
                *
                * TEST MANTISSA FOR NEGATIVE, ZERO OR ONE
                * IF NEGATIVE OR ZERO, JUMP TO ERR
                * IF MANTISSA IS ONE ( NORMALIZED INPUT
                *  THEN IS 0.1), SUBTRACT LN10 FROM OUTPUT
                *  AND RETURN
                *
0918 0C0856             LODA,R0 X+2     * TEST SIGN MANTISSA
091B 9C07CB             BCFA,Z  ERR     * IF NEG, JUMP TO ERR
091E 0507               LODI,R1 LENG
0920 0D4854     TSTZ    LODA,R0 X,R1,-  * LOAD DIGIT OF MANTISSA
0923 9C092E             BCFA,Z  TSTO    * IF NOT ZERO, TEST FOR 1
0926 E503               COMI,R1 3       * END OF MANTISSA?
0928 9C0920             BCFA,EQ TSTZ    * IF NOT, KEEP TESTING
092B 1F07CB             BCTA,UN ERR     * JUMP TO ERR
092E E503       TSTO    COMI,R1 3       * FIRST BYTE OF MANTISSA?
0930 9C0953             BCFA,EQ CTN     * IF NOT, CONTINUE
0933 E410               COMI,R0 H'10'   * ONE?
0935 9C0953             BCFA,EQ CTN     * IF NOT, CONTINUE
0938 0C0870             LODA,R0 OUTP    * LOAD POINTERS
093B 0D0871             LODA,R1 OUTP+1
093E CC0446             STRA,R0 PTRR
0941 CD0447             STRA,R1 PTRR+1
0944 0608               LODI,R2 <LN10
0946 0746               LODI,R3 >LN10
0948 3F0872             BSTA,UN STPT
094B 3F0469             BSTA,UN DSUB    * OUTPUT - LN10
094E 0C0853             LODA,R0 PSL     * RESTORE PROGRAM STATUS
0951 93                 LPSL
0952 17                 RETC,UN         * RETURN
0953 0408       CTN     LODI,R0 <X      * LOAD POINTER NORMALIZED
0955 0554               LODI,R1 >X        INPUT
0957 0608               LODI,R2 <ONE    * LOAD POINTER CONSTANT 1
0959 0738               LODI,R3 >ONE
095B 3F0872             BSTA,UN STPT
095E 0408               LODI,R0 <X1     * LOAD POINTER X1
0960 055B               LODI,R1 >X1
0962 CC0446             STRA,R0 PTRR
0965 CD0447             STRA,R1 PTRR+1
0968 3F0469             BSTA,UN DSUB    * X1 = INPUT - 1
```

```
TWIN ASSEMBLER VER 2.0    BCD LOGARITHM ROUTINE (EDP 7612)    PAGE 0019

LINE ADDR  OBJECT  E SOURCE

0773 096B 0408              LODI,R0 <Y1       * LOAD POINTER Y1
0774 096D 0569              LODI,R1 >Y1
0775 096F CC0446            STRA,R0 PTRR
0776 0972 CD0447            STRA,R1 PTRR+1
0777 0975 3F0470            BSTA,UN DADD      * Y1 = INPUT + 1
0778                *
0779                * X = X1 * 10^(EXP-ZS)
0780                *
0781 0978 7508      LOOP    CPSL    WC        * WITHOUT CARRY
0782 097A 0706              LODI,R3 LENG-1    * MOVE X1 TO X
0783 097C 0F685B    MOVX    LODA,R0 X1,R3
0784 097F CF6854            STRA,R0 X,R3
0785 0982 FB78              BDRR,R3 MOVX
0786 0984 0F085C            LODA,R3 X1+1      * EXPONENT
0787 0987 0E085B            LODA,R2 X1        * SIGN OF EXPONENT
0788 098A 3F0887            BSTA,UN SUBE      * DECREMENT EXPONENT X
0789 098D CE0854            STRA,R2 X         * STORE SIGN
0790 0990 CF0855            STRA,R3 X+1       * STORE EXPONENT
0791                *
0792                * Y = Y1 * 10^(EXP-ZS)
0793                *
0794 0993 0706              LODI,R3 LENG-1    * MOVE Y1 TO Y
0795 0995 0F6869    MOVY    LODA,R0 Y1,R3
0796 0998 CF6862            STRA,R0 Y,R3
0797 099B FB78              BDRR,R3 MOVY
0798 099D 0F086A            LODA,R3 Y1+1      * EXPONENT
0799 09A0 0E0869            LODA,R2 Y1        * SIGN OF EXPONENT
0800 09A3 3F0887            BSTA,UN SUBE      * DECREMENT EXPONENT Y
0801 09A6 CE0862            STRA,R2 Y         * STORE SIGN
0802 09A9 CF0863            STRA,R3 Y+1       * STORE EXPONENT
0803                *
0804                * U = U +/- T(I)
0805                *
0806 09AC 0408              LODI,R0 <U        * LOAD POINTERS
0807 09AE 053F              LODI,R1 >U
0808 09B0 CC0446            STRA,R0 PTRR
0809 09B3 CD0447            STRA,R1 PTRR+1
0810 09B6 0608              LODI,R2 <T        * NEXT CONSTANT FROM TABLE
0811 09B8 0F0850            LODA,R3 TN
0812 09BB 8700              ADDI,R3 >T
0813 09BD 3F0872            BSTA,UN STPT
0814                * DETERMINE ADDITION OR SUBTRACTION
0815 09C0 0C0856            LODA,R0 X+2       * COMPARE SIGNS OF MANTISSAE
0816 09C3 EC0864            COMA,R0 Y+2
0817 09C6 9806              BCFR,EQ SUBM      * IF NOT EQUAL, JUMP TO SUBM
0818 09C8 3F0470            BSTA,UN DADD      * OTHERWISE ADD
0819 09CB 20                EORZ    R0
0820 09CC 1B05              BCTR,UN CTN1      * CONTINUE
0821 09CE 3F0469    SUBM    BSTA,UN DSUB
0822 09D1 04FF              LODI,R0 H'FF'
0823 09D3 CC0852    CTN1    STRA,R0 ADS       * STORE BRANCH CONSTANT
```

```
TWIN ASSEMBLER VER 2.0    BCD LOGARITHM ROUTINE (EDP 7612)    PAGE 0020

LINE ADDR  OBJECT  E SOURCE

                     * X1 = X1 +/- Y
                     *
     09D6 0408              LODI,R0 <X1       * LOAD POINTERS
     09D8 055B              LODI,R1 >X1
     09DA CC0446            STRA,R0 PTRR
     09DD CD0447            STRA,R1 PTRR+1
     09E0 0608              LODI,R2 <Y
     09E2 0762              LODI,R3 >Y
     09E4 3F0872            BSTA,UN STPT
                     * DETERMINE ADDITION OR SUBTRACTION
     09E7 0C0852            LODA,R0 AOS       * LOAD BRANCH CONSTANT
     09EA 9805              BCFR,Z  ADDM      * IF NOT ZERO, JUMP TO ADDM
     09EC 3F0469            BSTA,UN DSUB
     09EF 1B03              BCTR,UN CTN2      * CONTINUE
     09F1 3F0470     ADDM   BSTA,UN DADD
                     *
                     * Y1 = Y1 +/- X
                     *
     09F4 0408       CTN2   LODI,R0 <Y1       * LOAD POINTERS
     09F6 0569              LODI,R1 >Y1
     09F8 CC0446            STRA,R0 PTRR
     09FB CD0447            STRA,R1 PTRR+1
     09FE 0608              LODI,R2 <X
     0A00 0754              LODI,R3 >X
     0A02 3F0872            BSTA,UN STPT
                     * DETERMINE ADDITION OR SUBTRACTION
     0A05 0C0852            LODA,R0 AOS       * LOAD BRANCH CONSTANT
     0A08 9805              BCFR,Z  ADDN      * IF NOT ZERO, JUMP TO ADDN
     0A0A 3F0469            BSTA,UN DSUB
     0A0D 1B03              BCTR,UN CTN3      * CONTINUE
     0A0F 3F0470     ADDN   BSTA,UN DADD
                     *
                     * COMMON PART
                     *
     0A12 7508       CTN3   CPSL    WC        * WITHOUT CARRY
     0A14 0C084D            LODA,R0 I         * I = I + 1
     0A17 8401              ADDI,R0 1
     0A19 CC084D            STRA,R0 I
     0A1C EC084E            COMA,R0 I9        * COMPARE WITH I9
     0A1F 9C0978            BCFA,EQ LOOP      * IF LESS LOOP
     0A22 20                EORZ    R0        * RESTORE I TO 0
     0A23 CC084D            STRA,R0 I
     0A26 0C0851            LODA,R0 ZS        * ADD 1 TO ZS
     0A29 8401              ADDI,R0 1
     0A2B CC0851            STRA,R0 ZS
     0A2E 0409              LODI,R0 9         * I9 = 9
     0A30 CC084E            STRA,R0 I9
     0A33 0C0850            LODA,R0 TN        * ADD 7 TO TN FOR NEXT CONSTANT
     0A36 8407              ADDI,R0 7
     0A38 CC0850            STRA,R0 TN
```

Central application laboratory CAB Eindhoven - The Netherlands

```
TWIN ASSEMBLER VER 2 0    BCD LOGARITHM ROUTINE (EDP 7612)    PAGE 0021

LINE ADDR  OBJECT  E SOURCE

0876 0A3B 0C084F            LODA,R0 I         * ADD 1 TO I
0877 0A3E 8401              ADDI,R0 1
0878 0A40 CC084F            STRA,R0 I
0879 0A43 E408              COMI,R0 NR        * COMPARE WITH NR
0880 0A45 9C0973            BCFA,EQ LOOP      * IF LESS THEN LOOP
0881                   *
0882                   *
0883                   *  OUTP = OUTP + 2U
0884                   *
0885 0A48 0C0870            LODA,R0 OUTP      * LOAD POINTERS
0886 0A4B 0D0871            LODA,R1 OUTP+1      FOR ADDING U TWO
0887 0A4E CC0446            STRA,R0 PTRR        TIMES TO OUTP
0888 0A51 CD0447            STRA,R1 PTRR+1
0889 0A54 0608              LODI,R2 <U
0890 0A56 073F              LODI,R3 >U
0891 0A58 3F0872            BSTA,UN STPT
0892 0A5B 3F0470            BSTA,UN DADD      * ADD
0893 0A5E 3F0470            BSTA,UN DADD      * ADD
0894 0A61 0C0853            LODA,R0 PSL       * RESTORE PROGRAM STATUS
0895 0A64 93                LPSL
0896 0A65 17                RETC,UN           * RETURN
0897                   *
0898 0A00                   END     A

 TOTAL ASSEMBLY ERRORS = 0000
```