Floating-Point Library V1.2 for TurboForth V1.2

Low-Level Support Functions Reference Guide

by

Lee Stewart

Table of Contents

1 Introduction

The low-level floating-point library (FPLTF) presented herein supplies all of the low-level functions necessary to replace the similar GPL (Graphics Programming Language) and XML (Executable Machine Language) functions in the console of the TI-99/4A computer. Along with the high-level TurboForth (TF) library described in the *Floating-Point Library V1.2 for TurboForth V1.2: TurboForth Words*, the FPLTF provides full floating-point functionality to TurboForth version 1.2.

The FPLTF is written in TMS9900 assembler and adapted by Lee Stewart from the MDOS L10 Math Library to run on the TI-99/4A. It occupies 5740 bytes starting at 2000h in CPU RAM space.

The functions in the low-level library are enumerated below:

Floating-Point Math Functions		
Number	Name	Description
0 (00h)	FCOMP	Floating-Point Compare
1 (01h)	FSUB	Floating-Point Subtract
2 (02h)	FADD	Floating-Point Add
3 (03h)	FMULT	Floating-Point Multiply
4 (04h)	FDIV	Floating-Point Divide
5 (05h)	PWR	Floating-Point Power
6 (06h)	EXP	Floating-Point e^x
7 (07h)	LOG	Floating-Point ln(x)
8 (08h)	SQR	Floating-Point sqr(x)
9 (09h)	COS	Floating-Point cos(x)
10 (0Ah)	SIN	Floating-Point sin(x)
11 (0Bh)	TAN	Floating-Point tan(x)
12 (0Ch)	ATN	Floating-Point atn(x)
13 (0Dh)	GRI	Floating-Point Greatest Integer
14 (0Eh)	CFI	Convert Floating-Point to Integer
15 (0Fh)	CIF	Convert Integer to Floating-Point
16 (10h)	CSINT	Convert String to Integer
17 (11h)	CSN	Convert String to Floating-Point
18 (12h)	CNS	Convert Floating-Point to String

The FPLTF uses the Floating Point Accumulator (FAC) at 834Ah and the Argument Register (ARG) at 835Ah, both of which are 8 bytes long. A number of other Scratchpad (8300h – 83FFh) locations are also used, so that portion of Scratchpad memory is saved/restored as required by the TurboForth words that use the FPLTF.

The TMS9900 assembly language source code (ALC) for the library is presented at the end of this document.

2 Acknowledgements

The author would like to thank the following for their help with the development of this library:

- Tim Tesch for the source code of the MDOS L10 Floating Point Library (FPL) and encouragement and permission in this effort to adapt it to do the heavy lifting for the TurboForth Floating Point Library.
- Beery Miller for permission to use the MDOS L10 FPL and to quote his *GenRef v1.00 MDOS Reference Guide: Math Library*.
- 9640News and all of the MDOS contributors who developed the TMS9900 code for the MDOS L10 FPL, which we adapted for use with TurboForth on the TI-99/4A.
- Mark Wills for developing the tagged object code loader for Turboforth, which made development of FPLTF less painful.

3 Loading the Low-Level Floating-Point Library

Loading the low-level floating-point library is the first order of business when the TurboForth Floating-Point Library (blocks 54 – 65) is loaded. It loads the memory image of the FPL from blocks 66 – 71 to 2000h.

4 Notes on Radix 100 Notation

TurboForth floating-point math routines use radix-100 format for floating-point numbers. The term "radix" is used in mathematics to mean "number base". We will use "radix 100" to describe the base-100 or centimal number system and "radix 10" to describe the base-10 or decimal number system. Radix-100 format is the same format used by the XML and GPL routines in the TI-99/4A console. Each floating-point number is stored in 8 bytes (4 cells) with a sign bit, a 7-bit, excess-64 (64-biased) integer exponent of the radix (100) and a normalized, 7-digit (1 radix-100 digit/byte) significand for a total of 8 bytes per floating point number. The signed, radix-100 exponent can be -64 to +63. (Keep in mind that the exponent is for radix-100 notation. Those same exponents radix 10 would be -128 to +126.) The exponent is stored in the most significant byte (MSB) biased by 64, *i.e.*, 64 is added to the actual exponent prior to storing, *i.e.*, -64 to +63 is stored as 0 to 127.

The significand (significant digits of the number) must be normalized, *i.e.*, if the number being represented is not zero, the MSB of the significand must always contain the first non-zero (significant) radix-100 digit, with the radix exponent of such a value that the radix point immediately follows the first digit. This is essentially scientific notation for radix 100. Each byte contains one radix-100 digit of the number, which, of course, means that each byte can have a value from 0 to 99 (0 to 63h) except for the first byte of a non-zero number, which must be 1 to 99. It is easy to view a radix-100 number as a radix-10 number by representing the radix-100 digits as pairs of radix-10 digits because radix 100 is the square of radix 10. In the following list of largest and smallest possible 8-byte floating point numbers, the radix-100 representation is on the left with spaces between pairs of radix-100 digits. The radix-16 (hexadecimal) internal representation of each byte of the number is also shown:

- Largest positive floating point number [hexadecimal: 7F 63 63 63 63 63 63]:

$$99.99999999999 \times 100^{63} = 99.999999999999 \times 10^{126}$$
$$= 9.9999999999999 \times 10^{127}$$

- Largest negative floating point number [hexadecimal: 80 9D 63 63 63 63 63 63]:

$$-99.99999999999 \times 100^{63} = -99.999999999999 \times 10^{126}$$
$$= -9.9999999999999 \times 10^{127}$$

- Smallest positive floating point number [hexadecimal: 00 01 00 00 00 00 00 00]:

$$01.00000000000 \times 100^{-64} = 1.000000000000 \times 10^{-128}$$

- Smallest negative floating point number [hexadecimal: FF FF 00 00 00 00 00 00]:

$$-01.00000000000 \times 100^{-64} = -1.000000000000 \times 10^{-128}$$

The only difference in the internal storage of positive and negative floating point numbers is that only the first word (2 bytes) of negative numbers is negated or complemented (two's complement).

A floating point zero is represented by zeroing only the first word. The remainder of the floating point number does not need to be zeroed for the number to be treated as zero for all floating point calculations.

5 Calling the Functions

The calling program must consider the following:

- Input parameters for calling a function—
  - R0 = function number.
  - R1 = address of any number expected:
    - This is usually the reserved spot on the FP stack for a floating-point number in 8-byte, radix-100 format.
    - For string-conversion functions, this may be the FP stack or **`HERE`** (for **`>F`**).
    - This actually can be any RAM address, but for TurboForth is usually one of the above addresses.
  - R2 =
    - the address of the string buffer (usually **`_fpstr`**) if required for input or output (first byte is the length of the string) or
    - address of second floating-point number in 8-byte, radix-100 format if two numbers are expected.
  - R7 = option 1 (only used for CNS to convert a floating-point number to a string).
  - R8 = option 2 (only used for CNS).

  - R9 = option 3 (only used for CNS).

- Output parameters upon return—
  - Status will be in the status register (ST) immediately upon return.
  - R0 = error code or 0.
  - R1 = address (usually on the FP stack) of any returned number in 8-byte, radix-100 format. This is the same address to which the input R1 address points.
  - R2 = the address of the string buffer (usually **`_fpstr`** for TurboForth) for any returned string. will have used R1 as a pointer to the string buffer (first byte of string buffer will have the string length). This is the same address to which the input R2 address points.
- Call the function with—
  - **`BLWP @>2000`** (TMS9900 assembly language);
  - **`$2000 @@ BLWP,`** (TurboForth assembly code).

6 Functions

6.1 Parameters

6.1.1 Floating-Point Numbers

A floating-point parameter is indicated as *float* and is in 8-byte radix-100 format.

6.1.2 Integers

An integer parameter is indicated as *integer* and is in 16-bit format.

6.1.3 Strings

A string parameter is indicated as *string* and is expected to contain the length byte as the first character of the string buffer.

6.2 FCOMP—Floating-Point Compare

Input:

R0 = 0000h

R1 = Pointer to $float_1$

R2 = Pointer to $float_2$

Output:

Status Register = AG set if $float_2 > float_1$

EQ set if $float_2 = float_1$

6.3 FSUB—Floating-Point Subtract

Input:

R0 = 0001h

R1 = Pointer to $float_1$

R2 = Pointer to $float_2$

Output:

R0 = Error Code

R1 = Pointer to $float_2 - float_1$

6.4 FADD—Floating-Point Add

Input:

R0 = 0002h

R1 = Pointer to $float_1$

R2 = Pointer to $float_2$

Output:

R0 = Error Code

R1 = Pointer to $float_2 + float_1$

6.5 FMULT—Floating-Point Multiply

Input:

R0 = 0003h

R1 = Pointer to $float_1$

R2 = Pointer to $float_2$

Output:

R0 = Error Code

R1 = Pointer to $float_2 * float_1$

6.6 FDIV—Floating-Point Divide

Input:

R0 = 0004h

R1 = Pointer to $float_1$

R2 = Pointer to $float_2$

Output:

R0 = Error Code

R1 = Pointer to $float_2 \div float_1$

6.7 PWR—Floating-Point Power

Input:

R0 = 0005h

R1 = Pointer to $float_1$

R2 = Pointer to $float_2$

Output:

R0 = Error Code

R1 = Pointer to $float_2^{float_1}$

6.8 EXP—Floating-Point e^x

Input:

R0 = 0006h

R1 = Pointer to *float*

Output:

R0 = Error Code

R1 = Pointer to e^{float}

6.9 LOG—Floating-Point ln(*x*)

Input:

R0 = 0007h

R1 = Pointer to *float*

Output:

R0 = Error Code

R1 = Pointer to ln(*float*)

6.10 SQR—Floating-Point $\sqrt{x}$

Input:

R0 = 0008h

R1 = Pointer to *float*

Output:

R0 = Error Code

R1 = Pointer to $\sqrt{float}$

6.11 COS—Floating-Point cos(*x*)

Input:

R0 = 0009h

R1 = Pointer to *float*

Output:

R0 = Error Code

R1 = Pointer to cos(*float*)

6.12 SIN—Floating-Point sin(*x*)

Input:

R0 = 000Ah

R1 = Pointer to *float*

Output:

R0 = Error Code

R1 = Pointer to sin(*float*)

6.13 TAN—Floating-Point tan(*x*)

Input:

R0 = 000Bh

R1 = Pointer to *float*

Output:

R0 = Error Code

R1 = Pointer to tan(*float*)

6.14 ATN—Floating-Point atan(*x*)

Input:

R0 = 000Ch

R1 = Pointer to *float*

Output:

R0 = Error Code

R1 = Pointer to atan(*float*)

6.15 GRI—Floating-Point floor(*x*)

Input:

R0 = 000Dh

R1 = Pointer to *float*

Output:

R0 = Error Code

R1 = Pointer to floor(*float*)

6.16 CFI—Convert Floating-Point to Integer

Input:

R0 = 000Eh

R1 = Pointer to *float*

Output:

R0 = Error Code

R1 = Pointer to *integer*

6.17 CIF—Convert Integer to Floating-Point

Input:

R0 = 000Fh

R1 = Pointer to *integer*

Output:

R0 = Error Code

R1 = Pointer to *float*

6.18 CSINT—Convert String to Integer

Input:

R0 = 0010h

R1 = Pointer to result

R2 = Pointer to *string*

Output:

R0 = Error Code

R1 = Pointer to *integer*

R2 = Pointer to number of digits used

6.19 CSN—Convert String to Floating-Point

Input:

R0 = 0011h

R1 = Pointer to result

R2 = Pointer to *string*

Output:

R0 = Error Code

R1 = Pointer to *float*

6.20 CNS—Convert Floating-Point to String

Input:

R0 = 0012h

R1 = Pointer to *float*

R2 = Pointer to *string*

R7 = $option_1$

- bit_0: 0 = free format (ignore $option_2$ & $option_3$)
  1 = fixed format ($option_2$ & $option_3$ are field sizes)
- bit_1: 1 = explicit sign
- bit_2: 1 = '+' instead of space for positive sign
- bit_3: 1 = E-notation
- bit_4: 1 = extended E-notation (bit_3 = 1 also required)

R8 = $option_2$ (places left of decimal point, including explicit sign)

R9 = $option_3$ (places right of decimal point, including decimal point)

For fixed format, $option_2$ and $option_3$ do not include the 4 places for E-notation (E±nn) or the 5 places for extended E-notation (E±nnn).

Output:

R0 = Error Code

R2 = Pointer to *string*

7 TMS9900 Assembly Source Code

This listing is the output from Asm994a, the assembler packaged with the TI-99/4A emulator Win994a by Cory Burr (*http://www.99er.net/win994a.shtml*). You will notice that this is v3.010, not the one packaged with the latest Win994a package, which was v3.009. The JH bug was fixed in v3.010, but never repackaged so it is only available from (???).

```
Asm994a TMS99000 Assembler - v3.010

               * Asm994a Generated Register Equates
               *
      0000 0000 R0       EQU     0
      0000 0001 R1       EQU     1
      0000 0002 R2       EQU     2
      0000 0003 R3       EQU     3
      0000 0004 R4       EQU     4
      0000 0005 R5       EQU     5
      0000 0006 R6       EQU     6
      0000 0007 R7       EQU     7
      0000 0008 R8       EQU     8
      0000 0009 R9       EQU     9
      0000 000A R10      EQU     10
      0000 000B R11      EQU     11
      0000 000C R12      EQU     12
      0000 000D R13      EQU     13
      0000 000E R14      EQU     14
      0000 000F R15      EQU     15
               *
   1           ********************************************************************************
   2           *
   3           *  WRITTEN: 06/05/1987
   4           *  UPDATED: (too many times...) PaulC
   5           *
   6           *  FILE:    HDS1.MDOS.L10.MATHS
   7           *
   8           *  NAME:    TRINSIC FUNCTIONS
   9           *           & FLOATING POINT
  10           *
  11           ********************************************************************************
  12           *  12/2012: Modified by Lee Stewart to run in the TI-99/4A
  13           *  in place of the console's GPL/XML floating point routines.
  14           *  This particular instance is designed to be called from TurboForth v1.2+.
  15           ********************************************************************************
  16
  17  0000 834A FAC    EQU  >834A       **les**
  18  0000 835C ARG    EQU  >835C       **les**
  19  0000 83A0 WSG    EQU  >83A0       **les**  FLOATING POINT WORKSPACE
  20           *
  21           *    DEF'S IN THIS COMPILE
  22           *
  23  0000 2000        DEF  FPLLNK      **les** for entry with BLWP
  24           *
  25           *    NOW FOR SOME EQUATES
  26           *
  27  0000 8300 PAD    EQU >8300       **les**
  28  0000 8354 FDVSR  EQU  FAC+>000A         FLOAT DIVISOR
  29  0000 836C EXP    EQU  ARG+16
  30  0000 836E SIGN   EQU  EXP+2
  31  0000 0003 ERROV  EQU  >03                INTEGER OVERFLOW CODE
  32  0000 8000 SGNBIT EQU  >8000
  33  0000 3000 X3000  EQU  >3000
  34  0000 2B00 XPLUS  EQU  >2B00
  35  0000 2D00 XMINUS EQU  >2D00
  36  0000 3900 X3900  EQU  >3900
  37  0000 2E00 XDOT   EQU  >2E00
  38
  39                   AORG >2000        **les** where we want to load library for TurboForth
  40           *
  41  2000 83A0 FPLLNK  DATA WSG,FPMLIB    entry point for BLWP; programmer can use   **les**
```

```
41  2002 2008
42            **les**                       BLWP @FPLLNK or
43            **les**                       BLWP @>2000 (above AORG sets this address)
44            *
45  2004 071D BADM1  SETO *R13              **les** set caller's R0 to error
46  2006 0380        RTWP
47            *
48  2008 C0DD FPMLIB MOV  *R13,R3           **les** save caller's math opcode
49  200A 1613        JNE  MATH2             this is status from the mov *r13,r3
50            *
51                   PAGE                   do a floating compare pretty quick
52  200C C16D OCOMP  MOV  @2(R13),R5
52  200E 0002
53  2010 C1ED        MOV  @4(R13),R7
53  2012 0004
54  2014 8D57        C    *R7,*R5+          COMPARE THE FIRST WORDS
55  2016 160B        JNE  OCOMRT            DONE COMPARING IF NOT EQUAL
56  2018 C1B7        MOV  *R7+,R6           SIGN OF THE NUMBERS
57  201A 1309        JEQ  OCOMRT            NUMBERS ARE ZERO AND EQUAL
58  201C 1503        JGT  OCOM01            BOTH NEGATIVE
59  201E C185        MOV  R5,R6
60  2020 C147        MOV  R7,R5
61  2022 C1C6        MOV  R6,R7
62  2024 8D77 OCOM01 C    *R7+,*R5+         BOTH POSITIVE
63  2026 1603        JNE  OCOMRT            CONTINUE COMPARING UNITL UNEQUAL
64  2028 8D77        C    *R7+,*R5+         OR END OF NUMBER
65  202A 1601        JNE  OCOMRT
66  202C 8557        C    *R7,*R5           THE LAST !
67  202E 02CF OCOMRT STST R15               EXIT AS SPECIFIED
68  2030 0380        RTWP
69  2032 04E0 MATH2  CLR  @FXNTYP     **les** 0 = FP return in caller's R1
69  2034 20D8
70  2036 04DD        CLR  *R13              ZERO OUT ERROR CODE
71  2038 04E0        CLR  @FAC+10
71  203A 8354
72  203C 0283        CI   R3,18
72  203E 0012
73  2040 1BE1        JH   BADM1             invalid opcode
74  2042 0A13        SLA  R3,1
75  2044 C123        MOV  @MATH##(R3),R4   **les** math opcode address to R4
75  2046 209C
76            *
77  2048 0283        CI   R3,14*2
77  204A 001C
78  204C 1B11        JH   DOIT              conversions...
79            *
80  204E C06D        MOV  @2(R13),R1        arg1  **les** caller's R1 to FAC
80  2050 0002
81  2052 0202        LI   R2,FAC            put in fac
81  2054 834A
82  2056 06A0        BL   @R1$2
82  2058 2092
83  205A 0283        CI   R3,5*2
83  205C 000A
84  205E 1B08        JH   DOIT              monadic function
85            *
86  2060 C06D        MOV  @4(R13),R1     **les** arg2:  caller's R1 to ARG
86  2062 0004
87            *
88  2064 0202        LI   R2,ARG            put in arg
88  2066 835C
89  2068 06A0        BL   @R1$2
89  206A 2092
90  206C 0720        SETO @FXNTYP     **les** -1 = FP return in caller's R2
90  206E 20D8
91            *
92  2070 0694 DOIT   BL   *R4
93            *
94  2072 DB60 TRINRT MOVB @FAC+10,@1(R13)   return error code
94  2074 8354
94  2076 0001
95  2078 02CF        STST R15               routines do an RT to return float from FAC
96            *
97            **les** result from FAC to caller's destination
```

```
 98           ********ENSURE WE ACTUALLY HAVE A DESTINATION ADDRESS*********
 99  207A 0201        LI   R1,FAC
 99  207C 834A
100  207E C0AD        MOV  @2(R13),R2      monadic fxn return address in caller's R1
100  2080 0002
101  2082 C0E0        MOV  @FXNTYP,R3      2-parm function?
101  2084 20D8
102  2086 1302        JEQ  TRINR2          no
103  2088 C0AD        MOV  @4(R13),R2      yes; return address is in caller's R2
103  208A 0004
104  208C 06A0 TRINR2 BL   @R1$2
104  208E 2092
105           *
106  2090 0380        RTWP                   return to caller...
107           *
108           **les**-----A space-saving routine below for copying a floating-point number.
109           *
110  2092 CCB1 R1$2   MOV  *R1+,*R2+
111  2094 CCB1        MOV  *R1+,*R2+
112  2096 CCB1        MOV  *R1+,*R2+
113  2098 C491        MOV  *R1,*R2
114  209A 045B        RT
115           *
116           **les**-----A space-saving routine above for copying a floating-point number.
117           *
118           *      dispatch table for opcodes
119           *
120  209C 200C MATH## DATA OCOMP,FSUB,FADD,FMULT,FDIV,PWR$$
120  209E 2140
120  20A0 2144
120  20A2 2244
120  20A4 2380
120  20A6 2C50
121  20A8 2E0A        DATA EXP$$,LOG$$,SQR$$,COS$$,SIN$$,TAN$$
121  20AA 2F74
121  20AC 3180
121  20AE 323E
121  20B0 325C
121  20B2 332E
122  20B4 3390        DATA ATN$$,GRI$$
122  20B6 341A
123  20B8 2B4A        DATA CFI$$
124  20BA 2BF6        DATA CIF,CSINT$,CSN,CNS
124  20BC 2AB4
124  20BE 2960
124  20C0 24FC
125           *
126           *  NOW FOR SOME REQUIRED DATA
127           *
128  20C2 0003 CW03   DATA 3
129  0000 20C5 CBH08  EQU  $+1
130  20C4 0008 CW08   DATA 8
131  20C6 0080 CW128  DATA 128
132  20C8 0010 CW16   DATA 16
133  20CA 0064 CW100  DATA 100
134  20CC 0520 ERRNIP DATA >0520
135  20CE 06A0 ERRLOG DATA >06A0
136  20D0 0460 ERRSQR DATA >0460
137  20D2 0A   CBHA   BYTE >0A
138  20D3 80   CBH80  BYTE >80
139  20D4 32   CBD50  BYTE 50
140  20D5 0000        EVEN
141           *
142           *  NOW FOR SOME REQUIRED AREAS IN MEMORY
143           *
144  20D6 0000 EXTRTN DATA 0
145           * FAC11  DATA 0       **les**     save FAC+11                  **les**
146           * FAC12  DATA 0       **les**     save FAC+12                  **les**
147           * FAC13  DATA 0       **les**     save FAC+13                  **les**
148  20D8 0000 FXNTYP DATA 0       **les**  fxn type: 0 = 1 FP; -1 = 2 FPs  **les**
149  20DA 0000 SAVCSN DATA 0
150  20DC 0000 SAVR12 DATA 0
151  20DE 0000 SAVR13 DATA 0
152  20E0 0000 WSM6   DATA 0
```

```
153  20E2 0000 WSM8   DATA 0
154  20E4 0000 WSM10  DATA 0
155  20E6 0000 P$     DATA 0
156  20E8 0000 OE$    DATA 0
157  20EA 0000 Q$     DATA 0
158  20EC 0000 STK1   DATA 0,0,0,0
158  20EE 0000
158  20F0 0000
158  20F2 0000
159  20F4 0000 STK2   DATA 0,0,0,0
159  20F6 0000
159  20F8 0000
159  20FA 0000
160  20FC 0000 STK3   DATA 0,0,0,0
160  20FE 0000
160  2100 0000
160  2102 0000
161  2104 0000 STK4   DATA 0,0,0,0
161  2106 0000
161  2108 0000
161  210A 0000
162            FORBUF
163  210C 0000 PLYBUF DATA 0,0,0,0
163  210E 0000
163  2110 0000
163  2112 0000
164  2114 0000 PLWBUF DATA 0,0,0,0
164  2116 0000
164  2118 0000
164  211A 0000
165            *
166            * UN-COMMENT THE FOLLOWING WHEN USING TI-99/4A ASSEMBLER (WITH CORRECT DSK#):
167            *
168            *       COPY "DSK2.FLOAT"
169            *       PAGE
170            *
171            **les** Modified CNS and CSNN below were renamed with appended "_TF" because,
172            **les**    initially, some changes were unique to TurboForth.  That uniqueness
173            **les**    has since evaporated, but the names were retained.
174            *
175            *       COPY "DSK2.CNS_TF"
176            *       PAGE
177            *       COPY "DSK2.CSNN_TF"
178            *       PAGE
179            *       COPY "DSK2.CFI"
180            *       PAGE
181            *       COPY "DSK2.CIF"
182            *       PAGE
183            *       COPY "DSK2.TRINSIC1"
184            *       PAGE
185            *       COPY "DSK2.TRINSIC2"
186            *       END
               *
               *       COPY    'C:\Users\Lee\Downloads\TI-99-4A\TurboForth\MDOS\L10\FLOAT.a99'
               *
  1            *===========================================================================
  2            *
  3            *    FILE:  WDS1.156.FPIN
  4            *
  5            *    WHAT:  FLOATING POINT INTERFACE
  6            *
  7            * VERSION:  1.0 - 03/02/86    BASE LINE FROM 99/4A
  8            *
  9            *===========================================================================
 10            FCOMP
 11  211C 0207 FCOMP1 LI   R7,ARG
 11  211E 835C
 12  2120 0205 FCOMP7 LI   R5,FAC            ENTRY FOR COMPARE NUM R7*
 12  2122 834A
 13  2124 8D57        C    *R7,*R5+          COMPARE THE FIRST WORDS
 14  2126 160B        JNE  FCOMRT            DONE COMPARING IF NOT EQUAL
 15  2128 C1B7        MOV  *R7+,R6           SIGN OF THE NUMBERS
 16  212A 1309        JEQ  FCOMRT            NUMBERS ARE ZERO AND EQUAL
 17  212C 1503        JGT  FCOM01            BOTH NEGATIVE
```

```
18  212E C185        MOV  R5,R6
19  2130 C147        MOV  R7,R5
20  2132 C1C6        MOV  R6,R7
21  2134 8D77 FCOM01 C    *R7+,*R5+         BOTH POSITIVE
22  2136 1603        JNE  FCOMRT            CONTINUE COMPARING UNITL UNEQUAL
23  2138 8D77        C    *R7+,*R5+         OR END OF NUMBER
24  213A 1601        JNE  FCOMRT
25  213C 8557        C    *R7,*R5           THE LAST !
26  213E 045B FCOMRT RT                     EXIT AS SPECIFIED
27                   PAGE
28  2140 0520 FSUB   NEG  @FAC             SAVE RETURN ADDRESS
28  2142 834A
29            FADD
30  2144 C1E0 FADD1  MOV  @ARG,R7          IS ARGUMNET ZERO ?
30  2146 835C
31  2148 130B        JEQ  FADD02           YES NO CHANGE TO FAC
32  214A C220        MOV  @FAC,R8          IS FAC ZERO
32  214C 834A
33  214E 1609        JNE  FADD03           NO, GO ADD FAC TO ARG
34  2150 0201 AGTOFC LI   R1,ARG           YES, MOVE ARG TO FAC
34  2152 835C
35  2154 0202        LI   R2,FAC
35  2156 834A
36            **les**  Can't use this instruction to replace the next 4 because we   **les**
37            **les**  would need to save/restore R11, which negates the advantage!! **les**
38                  ; BL   @R1$2           move 4 words from *R1 to *R2               **les**
39            *
40  2158 CCB1        MOV  *R1+,*R2+
41  215A CCB1        MOV  *R1+,*R2+        NEXT VICTIM
42  215C CCB1        MOV  *R1+,*R2+
43  215E C491        MOV  *R1,*R2          DONE ?
44            *
45  2160 045B FADD02 RT                    EXIT TO GPL WITH STATUS
46            **les**                           ^^^???
47  2162 29C8 FADD03 XOR  R8,R7            SIGN DIFFERENCE
48  2164 0203        LI   R3,FAC
48  2166 834A
49  2168 0206        LI   R6,ARG
49  216A 835C
50  216C 0753        ABS  *R3              TAKE ABSOULTE VALUES OF FAC
51  216E 0756        ABS  *R6              AND ARG
52  2170 8DB3        C    *R3+,*R6+
53  2172 1513        JGT  FADD05           IS OKAY
54  2174 1109        JLT  FADD2B
55  2176 8DB3        C    *R3+,*R6+
56  2178 1510        JGT  FADD05
57  217A 1106        JLT  FADD2B
58  217C 8DB3        C    *R3+,*R6+
59  217E 150D        JGT  FADD05
60  2180 1103        JLT  FADD2B
61  2182 8593        C    *R3,*R6
62  2184 140A        JHE  FADD05
63  2186 1002        JMP  FADD21
64
65  2188 0643 FADD2B DECT R3
66  218A 0646        DECT R6
67  218C C013 FADD21 MOV  *R3,R0
68  218E CCD6        MOV  *R6,*R3+
69  2190 CD80        MOV  R0,*R6+
70  2192 0283        CI   R3,FAC+8
70  2194 8352
71  2196 16FA        JNE  FADD21
72  2198 2A07        XOR  R7,R8
73  219A 04C5 FADD05 CLR  R5               HANDY ZERO
74  219C 04E0        CLR  @FAC+8           CLEAR GUARD DIGITS FOR FAC
74  219E 8352
75  21A0 04E0        CLR  @ARG+8           AND ARG
75  21A2 8364
76  21A4 D808        MOVB R8,@SIGN         SAVE THE RESULT SIGN
76  21A6 836E
77  21A8 D1A0        MOVB @FAC,R6          FAC EXP TO R3(R6)
77  21AA 834A
78  21AC 0986        SRL  R6,8
79  21AE C806        MOV  R6,@EXP          USE FAC EXP AS RESULT EXP
```

```
 79  21B0 836C
 80  21B2 D805        MOVB R5,@FAC           CLEAR HIGH BYTE OF FAC TO CHECK
 80  21B4 834A
 81  21B6 7820        SB   @ARG,@WSG+13      FOR OVERFLOW
 81  21B8 835C
 81  21BA 83AD
 82  21BC 0286        CI   R6,7              SMALLER NUMBER TOO SMALL TO
 82  21BE 0007
 83  21C0 1540        JGT  FADD15            AFFECT THE SUM ?
 84            *                             YES, RETURN WITH LARGER NUMBER IN FAC
 85  21C2 C006        MOV  R6,R0             EXPONENT DIFFERENCE
 86  21C4 0208        LI   R8,1*256          1 FOR BYTE OPERATION
 86  21C6 0100
 87  21C8 0209        LI   R9,100*256        100 FOR BYTE OPERATION
 87  21CA 6400
 88  21CC 0205        LI   R5,FAC+9          POINTER TO LOW BYTE OF BIG NUM
 88  21CE 8353
 89  21D0 0206        LI   R6,ARG+9          AND LOW BYTE OF SMALLER NUMBER
 89  21D2 8365
 90  21D4 6180        S    R0,R6             ADJ ARG POINTER TO ALIGN RADIX
 91  21D6 C100        MOV  R0,R4             ADD/SUBTRACT LOOP COUNTER IS
 92  21D8 0224        AI   R4,-9             BYTES LEFT IN SMALLER NUMBER
 92  21DA FFF7
 93  21DC C047        MOV  R7,R1             TWO NUMBERS HAVE SAME SIGN
 94  21DE 1120        JLT  FADD11            NO SUBTRACT THEM
 95  21E0 B556 FADD06 AB   *R6,*R5           YES ADD A BYTE OF SMALL TO LARGE
 96  21E2 9255        CB   *R5,R9            IS SUM LARGER THAN RADIX
 97  21E4 1A03        JL   FADD07            THEN CONTINUE TO NEXT BYTE
 98  21E6 7549        SB   R9,*R5            SUBTRACT RADIX FROM THIS BYTE
 99  21E8 B948        AB   R8,@-1(R5)        AND ADD CARRY TO NEXT BYTE
 99  21EA FFFF
100  21EC 0605 FADD07 DEC  R5                TO NEXT HIGHER BIG NUMBER
101  21EE 0606        DEC  R6                AND NEXT HIGHER SMALLER NUMBER
102  21F0 0584        INC  R4                IF NOT ALL SIGNIF BYTES OF SMALL
103  21F2 11F6        JLT  FADD06            ADDED, THEN CONTINUE
104  21F4 1002        JMP  FADD09            ELSE PROPAGATE CARRY
105  21F6 0605 FADD08 DEC  R5                WAS LARGER POINT TO NEXT BYTE
106  21F8 B548        AB   R8,*R5            ADD CARRY TO NEXT BYTE
107  21FA 7549 FADD09 SB   R9,*R5            SUBTRACT RADIX FROM NEXT BYTE
108  21FC 15FC        JGT  FADD08            DONE IF REACHED ONE BYTE
109           *                             SMALLER THAN RADIX
110  21FE 13FB        JEQ  FADD08            CONTIUE IF RESULT = RADIX
111  2200 B549        AB   R9,*R5            RADIX SUBTRACTED ONCE TOO OFTEN
112  2202 D060        MOVB @FAC,R1           CARRY OUT OF HIGH ORDER RESULT ?
112  2204 834A
113  2206 130B        JEQ  FADD10            NO ROUNT RESULT
114  2208 05A0        INC  @EXP              YES, INCREMENT EXPONENT
114  220A 836C
115  220C 0201        LI   R1,FAC+8
115  220E 8352
116  2210 0202        LI   R2,9
116  2212 0009
117  2214 D851 FADD30 MOVB *R1,@1(R1)        SHIFT FAC ONE BYTE
117  2216 0001
118  2218 0601        DEC  R1
119  221A 0602        DEC  R2
120  221C 16FB        JNE  FADD30
121  221E 1072 FADD10 JMP  ROUN1
122  2220 7556 FADD11 SB   *R6,*R5           SUBTRACT A BYTE OF SMALL FROM BIG
123  2222 1504        JGT  FADD12
124  2224 1303        JEQ  FADD12
125  2226 B549        AB   R9,*R5
126  2228 7948        SB   R8,@-1(R5)
126  222A FFFF
127  222C 0605 FADD12 DEC  R5
128  222E 0606        DEC  R6
129  2230 0584        INC  R4
130  2232 11F6        JLT  FADD11
131  2234 1003        JMP  FADD14
132  2236 B549 FADD13 AB   R9,*R5
133  2238 0605        DEC  R5
134  223A 7548        SB   R8,*R5
135  223C D115 FADD14 MOVB *R5,R4
136  223E 11FB        JLT  FADD13
```

```
137  2240 1046        JMP  NORMAL
138
139  2242 1077 FADD15 JMP  PACKUP
140                   PAGE
141            *
142            *    FLOATING POINT MULTIPLICATION
143            *
144  2244 0203 FMULT  LI   R3,FAC            IF FAC IS ZERO
144  2246 834A
145  2248 0205        LI   R5,ARG
145  224A 835C
146  224C C213        MOV  *R3,R8            IF FAC IS ZERO
147  224E 1346        JEQ  FMULZR            THEN RESULT IS ZERO
148  2250 2A15        XOR  *R5,R8            COMPUT RESULT SIGN
149  2252 0755        ABS  *R5               IF ARG IS ZERO
150  2254 1343        JEQ  FMULZR            THEN ZERO FAC AND RETURN
151  2256 0753        ABS  *R3               TAKE ABS VALUE OF FAC
152  2258 04C9        CLR  R9                TO ZERO LOW BYTE OF RESULT EXP
153  225A D253        MOVB *R3,R9            RESULT EXP = FAC EXP
154  225C B255        AB   *R5,R9            PLUS ARG EXP
155  225E 06C9        SWPB 9                     >06C9
156  2260 0229        AI   R9,-63            SUBTRACT BIAS
156  2262 FFC1
157  2264 C809        MOV  R9,@EXP
157  2266 836C
158  2268 D808        MOVB R8,@SIGN          SAVE TIL NORMAL, ROUND
158  226A 836E
159  226C 0205        LI   R5,FAC+8           LOW ORDER DIGITS
159  226E 8352
160  2270 04F5 FMCLR  CLR  *R5+              WILL BE
161  2272 0285        CI   R5,FAC+16          FORMED
161  2274 835A
162  2276 16FC        JNE  FMCLR             HERE
163  2278 0205        LI   R5,FAC+8           BYTES IN FAC+1
163  227A 8352
164  227C 0605 FMUL02 DEC  R5                CHANGE SIGNIFICANT BYTE COUNT
165            *                             FOR LAST ZERO BYTE
166  227E D015        MOVB *R5,R0            IF NEXT FAC BYTE IS ZERO
167  2280 13FD        JEQ  FMUL02            THEN DECREMENT COUNT FOR IT
168  2282 0207        LI   R7,8              COUNT SIGNIFICANT BYTES IN ARG
168  2284 0008
169  2286 0607 FMUL03 DEC  R7                DECREMENT FOR ZERO BYTE
170  2288 D027        MOVB @ARG(R7),R0       IF THIS BYTE OF ARG IS ZERO
170  228A 835C
171  228C 13FC        JEQ  FMUL03            THEN DECREMENT COUNT
172  228E 04C0        CLR  R0                MPY/DIV WORK REGISTER
173  2290 3880        MPY  R0,R2             CURRENT RESULT HIGH BYTE
174  2292 C185        MOV  R5,R6
175  2294 0208        LI   R8,WSG+1           RB(R0)
175  2296 83A1
176  2298 0209        LI   R9,100            RADIX
176  229A 0064
177  229C C107 FMUL04 MOV  R7,R4             INNER LOOP CTR = BYTES IN ARG
178  229E A187        A    R7,R6             RESULT PTR TO END OF NEXT PARTIAL PROD
179  22A0 D815        MOVB *R5,@WSG+7         RB(R3) IS NEXT DIGIT OF FAC
179  22A2 83A7
180  22A4 D543        MOVB R3,*R5            CLEAR FAC DIGIT FOR NEXT PARTIAL
181  22A6 D624 FMUL05 MOVB @ARG(R4),*R8      GET NEXT DIGIT OF ARG
181  22A8 835C
182  22AA 3803        MPY  R3,R0             AND MPY IT
183  22AC D816        MOVB *R6,@WSG+5         TO CORRESPONDING PARITIAL PROD
183  22AE 83A5
184            *                             DIGIT IN RB(R2)
185  22B0 A042        A    R2,R1             ADD IN NEXT PARTIAL PROD DIGIT
186  22B2 3C09        DIV  R9,R0             CONVERT PRODUCT TO RADIX DIGIT
187            *                             AND CARRY
188  22B4 D5A0        MOVB @WSG+3,*R6         STORE NEW RESULT DIGIT IN FAC
188  22B6 83A3
189  22B8 0606        DEC  R6                POINT TO NEXT HIGHER BYTE OF RESULT
190  22BA B598        AB   *R8,*R6           ADD IN CARRY TO NEXT HIGHER BYTE
191  22BC 0604        DEC  R4                IF ALL ARG DIGITS NOT DONE
192  22BE 15F3        JGT  FMUL05            THEN CONTINUE
193  22C0 0606        DEC  R6                POINT TO START OF NEXT PARTIAL
194  22C2 0605        DEC  R5                IF FAC DIGITS REMAIN
```

```
195  22C4 0285        CI   R5,FAC
195  22C6 834A
196  22C8 15E9        JGT  FMUL04            THEN CONTINUE
197  22CA 04E0 FMEND  CLR  @FAC+10           no error to report
197  22CC 8354
198            *
199            *    NORMALIZE THE NUMBER
200            *
201  22CE 0201 NORMAL LI   R1,-9             NUMBER OF BYTES IN FAC INCLUDING
201  22D0 FFF7
202            *                             GUARD DIGITS
203  22D2 D0A1 NORM01 MOVB @FAC+10(R1),R2     IS NEXT BYTE OF FAC NON-ZERO
203  22D4 8354
204  22D6 1607        JNE  NORM02            YES SHIFT REST LEFT
205  22D8 0581        INC  R1                NO ALL BYTES ZERO
206  22DA 11FB        JLT  NORM01            YES LOOK AT NEXT BYTE
207
208  0000 22DC FZERO  EQU  $
209  22DC 04E0 FMULZR CLR  @FAC              INSTALL FLOATING ZERO
209  22DE 834A
210  22E0 04E0        CLR  @FAC+2            CLEAR BASIC TYPE CODE
210  22E2 834C
211  22E4 1033        JMP  STEX              AND EXIT WITH STATUS
212  22E6 C001 NORM02 MOV  R1,R0             NUMBER OF NON-ZERO BYTES
213  22E8 0220        AI   R0,9              FIRST BYTE NON-ZERO
213  22EA 0009
214  22EC 130B        JEQ  ROUN1             YES, FINISH
215  22EE 6800        S    R0,@EXP           NO, ADJUST EXPONENT FOR SHIFT
215  22F0 836C
216  22F2 0202        LI   R2,FAC+1          POINT OT FIRST BYTE OF FAC
216  22F4 834B
217  22F6 DCA1 NORM03 MOVB @FAC+10(R1),*R2+   MOVE NON-ZERO BYTE
217  22F8 8354
218  22FA 0581        INC  R1                IF NON-ZERO BYTES REMAIN
219  22FC 11FC        JLT  NORM03            THEN MOVE ANOTHER BYTE
220  22FE DC81 NORM04 MOVB R1,*R2+           MOVE A ZERO
221  2300 0600        DEC  R0                LAST BYTE DONE
222  2302 15FD        JGT  NORM04            NO CONTINUE
223
224            ROUND
225            ROUN1
226  2304 9820        CB   @FAC+8,@CBD50      IS ROUNDING NECESSARY
226  2306 8352
226  2308 20D4
227  230A 1A13        JL   PACKUP            NO PUT EXPONENT BACK
228  230C 0201        LI   R1,7              ROUND UP, GET NUMBER OF BYTES NEEDED
228  230E 0007
229  2310 0202 ROUNUP LI   R2,1*256          ONE FOR BYTE INSTRUCTION
229  2312 0100
230  2314 0200        LI   R0,100*256        SAME
230  2316 6400
231  2318 B842 ROUN02 AB   R2,@FAC(R1)       ADD ONE TO A BYTE OF FAC
231  231A 834A
232  231C 9021        CB   @FAC(R1),R0       IF BYTE NOT GREATER THAN RADIX
232  231E 834A
233  2320 1A08        JL   PACKUP            THEN PUT EXPONENT IN FAC
234  2322 7840        SB   R0,@FAC(R1)       BRING DIGIT BACK IN RANGE
234  2324 834A
235  2326 0601        DEC  R1                IF CARRY PAST HIGH BYTE OF FAC
236  2328 15F7        JGT  ROUN02            THEN CARRY TO NEXT HIGHER BYTE
237  232A 05A0        INC  @EXP              FRACTION HAS OVERFLOWED
237  232C 836C
238  232E D802        MOVB R2,@FAC+1         MAKE THE HIGH BYTE A ONE
238  2330 834B
239
240  2332 8820 PACKUP C    @EXP,@CW128
240  2334 836C
240  2336 20C6
241  2338 140D        JHE  OVEXP1
242  233A D820        MOVB @EXP+1,@FAC        PUT EXPONENT IN FAC
242  233C 836D
242  233E 834A
243  2340 D0A0        MOVB @SIGN,R2
243  2342 836E
```

```
244  2344 0542        INV  R2               IF SIGN IS NEGATIVE
245  2346 1102        JLT  PACK01
246  2348 0520        NEG  @FAC             THEN INVERT FIRST WORD
246  234A 834A
247
248           PACK01
249           STEX
250           STEX01
251  234C 045B        RT
252                   PAGE
253  0000 234E DIVZER EQU  $
254  234E 0209 FDIV01 LI   R9,>0200         DIVIDE BY ZERO ERROR CODE
254  2350 0200
255  2352 1005        JMP  BIGFLT           LARGEST MAGNITUDE WITH SIGN
256
257           OVEXP
258  2354 D0A0 OVEXP1 MOVB @EXP,R2          IS EXPONENT NEGATIVE
258  2356 836C
259  2358 11C1        JLT  FZERO            YES, RETURN ZERO @@ underflow!!!
260           OV
261  235A 0209 OV1    LI   9,>0100          ERROR CODE
261  235C 0100
262  0000 235C ERROVF EQU  $-2
263  235E 0200 BIGFLT LI   R0,->7F63        HIGH WORD OF LARGEST POSITIVE VALUE
263  2360 809D
264  2362 D0A0        MOVB @SIGN,R2         IS FAC NEGATIVE
264  2364 836E
265  2366 1101        JLT  BIGF01           YES @@pc fix...from jlt bigflt
266  2368 0500        NEG  R0
267  236A 0202 BIGF01 LI   R2,FAC           GET POINTER TO FAC
267  236C 834A
268  236E CC80        MOV  R0,*R2+          PUT APPROPRIATE HIGH WORD IN FAC
269  2370 0200        LI   R0,>6363         GET 99'S
269  2372 6363
270  2374 CC80        MOV  R0,*2+           PUT IN FAC
271  2376 CC80        MOV  R0,*2+           ....
272  2378 C480        MOV  R0,*2            ...
273  237A D809 ERRXI1 MOVB R9,@FAC+10        PLACE ERROR CODE
273  237C 8354
274  237E 10E6        JMP  STEX             NO ROUTINE SPECIFIED, RETURN
275                   PAGE
276  2380 0203 FDIV   LI   R3,FAC           POINTER TO FAC
276  2382 834A
277  2384 C213        MOV  *R3,R8           GET DIVISOR FIRST WORD
278  2386 0200        LI   R0,ARG           POINTER TO ARG
278  2388 835C
279  238A 2A10        XOR  *R0,R8           COMPUTE SIGN OF QUOTIENT
280  238C D808        MOVB R8,@SIGN         SAVE IT
280  238E 836E
281  2390 0753        ABS  *R3              ABSOLUTE DIVISOR
282  2392 13DD        JEQ  FDIV01           CANT BE ZERO
283  2394 0750        ABS  *R0              IS DIVIDED ZERO
284  2396 13A2        JEQ  FMULZR           YES RESULT IS ZERO
285  2398 D250        MOVB *R0,R9           GET DIVIDEND EXPONENT
286  239A 7253        SB   *R3,R9           SUBTRACT EXPONENTS TO GET
287           *                             QUOTIENT EXPONENT
288  239C 0889        SRA  R9,8             GET DIFFERENCE IN LOW BYTE
289  239E 0229        AI   R9,64            ADD BIAS TO EXPONENT
289  23A0 0040
290  23A2 C809        MOV  R9,@EXP          AND SAVE RESULT
290  23A4 836C
291           *                             MOVE FAC TO DIVISOR STORAGE
292  23A6 0204        LI   R4,FAC+10        WHERE TO MOVE IT TO
292  23A8 8354
293  23AA 0205        LI   R5,ARG+8
293  23AC 8364
294
295  23AE CD33        MOV  *R3+,*R4+        MOVE A WORD
296  23B0 CD33        MOV  *R3+,*R4+
297  23B2 CD33        MOV  *R3+,*R4+        MOVE A WORD
298  23B4 C513        MOV  *R3,*R4
299  23B6 04F5        CLR  *R5+
300  23B8 04F5        CLR  *R5+
301  23BA 04F5        CLR  *R5+
```

```
302  23BC 04D5        CLR  *R5               JUST MOVE AND CLEAR
303  23BE 04C4        CLR  R4
304  23C0 D804        MOVB R4,@ARG           CLEAR EXTRA HIGH BYTE OF
304  23C2 835C
305            *                            DIVIDEND
306  23C4 0205        LI   R5,WSG+1           GET POINTERS INTO MULTIPLY
306  23C6 83A1
307  23C8 0206        LI   R6,WSG+3
307  23CA 83A3
308  23CC 0207        LI   R7,100            RADIX
308  23CE 0064
309  23D0 04C2        CLR  R2                CLEAR HIGH BYTE OF WHERE V1 WILL BE
310  23D2 D820        MOVB @FDVSR+1,@WSG+5    GET V1 IN RB(R2)
310  23D4 8355
310  23D6 83A5
311  23D8 0282        CI   R2,49             HAS VI ALREADY BEEN NORMALIZED
311  23DA 0031
312  23DC 151E        JGT  FDIV06            YES PROCEED WITH DIVISION
313  23DE 0582        INC  R2                NO COMPUTE V1+1
314  23E0 04C3        CLR  R3                GET RADIX IN R REGS FOR DIV
315  23E2 C107        MOV  R7,R4             GET RADIX
316  23E4 3CC2        DIV  R2,R3             COMPUTE MULTIPLER
317            *                            INT(100/(V1+1))
318  23E6 0209        LI   R9,FDVSR+8
318  23E8 835C
319  23EA 0204 FDVLP  LI   R4,8              GET NUMBER OF BYTES IN DIVIDEND+1
319  23EC 0008
320  23EE 0604 FDIV04 DEC  R4                IGNORE ZERO BYTE AT END
321  23F0 0609        DEC  R9
322  23F2 D019        MOVB *R9,R0            IS NEXT HIGHER ORDER BYTE ZERO
323  23F4 13FC        JEQ  FDIV04            YES KEEP LOOKING FOR NON ZERO
324  23F6 04C0        CLR  R0                CLEAR CARRY AND LOW ORDER BYTE
325  23F8 C080 FDIV05 MOV  R0,R2             SAVE CARRY FROM LAST BYTE
326  23FA D559        MOVB *R9,*R5           GET NEXT BYTE OF DIVIDEND
327  23FC 3803        MPY  R3,R0             MULTIPLY THIS BYTE FOR MULTIPLER
328  23FE A042        A    R2,R1             ADD IN CARRY FROM PREVIOUS BYTE
329  2400 3C07        DIV  R7,R0             CNVRT TO RADIX DIGIT AND CARRY
330  2402 D656        MOVB *R6,*R9           PUT RESULT BYTE IN DIVIDEND
331  2404 0609        DEC  R9
332  2406 0604        DEC  R4                LOOP UNTIL ALL DIVIDEND BYTES
333  2408 15F7        JGT  FDIV05            NO CONTINUE MULTIPLICATION
334  240A 0289        CI   R9,FDVSR
334  240C 8354
335  240E 1603        JNE  FDVLPA
336  2410 0209        LI   R9,ARG+8
336  2412 8364
337  2414 10EA        JMP  FDVLP
338  2416 D815 FDVLPA MOVB *R5,@ARG          YES PUT CARRY OUT OF HIGH BYTE
338  2418 835C
339
340  241A 0206 FDIV06 LI   R6,8              NUMBER DIVISOR BYTES+1
340  241C 0008
341  241E 0606 FDIV07 DEC  R6                COMPUT NUBER OF SIGNIF BYTES
342            *                            IN DIVISOR
343  2420 D026        MOVB @FDVSR(R6),R0     GET NEXT HIGHER ORDER BYTE
343  2422 8354
344  2424 13FC        JEQ  FDIV07            IGNORE IF IT IS ZERO
345  2426 04C7        CLR  R7                CLR HIGH BYTE OF WHERE V1 WILL BE
346  2428 D820        MOVB @FDVSR+1,@WSG+15   RB(R7) IS V1
346  242A 8355
346  242C 83AF
347  242E C207        MOV  R7,R8             COPY VI TO COMPUTE
348  2430 3A20        MPY  @LW100,R8         COMPUT 100*V1
348  2432 24EE
349  2434 D820        MOVB @FDVSR+2,@WSG+17   GET V2
349  2436 8356
349  2438 83B1
350  243A A248        A    R8,R9             COMPUTE 100*V1+V2
351  243C 0205        LI   R5,-9             COMPUTE NINE BYTES OF QUOTIENT
351  243E FFF7
352  2440 020F        LI   R15,ARG           PTR TO HIGH BYTE OF DIVIDEND
352  2442 835C
353  2444 04C2 FDIV08 CLR  R2                CLEAR HIGH BYTE OF WHERE U(J) WILL BE
354  2446 D81F        MOVB *R15,@WSG+5        RB(R2) IS U(J)
```

```
354  2448 83A5
355  244A 38A0        MPY  @LW100,R2         COMPUTE 100*U(J)
355  244C 24EE
356  244E 04C0        CLR  R0                WHERE U(J+1) WILL BE
357  2450 D82F        MOVB @1(R15),@WSG+1    GET U(J+1)
357  2452 0001
357  2454 83A1
358  2456 A0C0        A    R0,R3             100*U(J)+U(J+1)
359  2458 3C87        DIV  R7,R2             GET Q AND REMAINDER
360  245A 38E0        MPY  @LW100,R3         100*REMAINDER
360  245C 24EE
361  245E D82F        MOVB @2(R15),@WSG+1    U(J+2)
361  2460 0002
361  2462 83A1
362  2464 A100        A    R0,R4             100*REM +U(J+2)
363  2466 C002        MOV  R2,R0             GET Q FOR THE TEST
364  2468 3808        MPY  R8,R0             COMPUTE V2*Q
365  246A 0282        CI   R2,100            DOES Q=100
365  246C 0064
366  246E 1302        JEQ  FDIV09            YES MAKE Q=99
367  2470 6044        S    R4,R1             NO COMPUTE V2*Q-(100*REM+U(J+2))
368  2472 1003        JMP  FDIV11            GO CHECK IF ITS IN RANGE
369
370  2474 6044 FDIV09 S    R4,R1             COMPUTE V2*Q-(100*REM+U(J+2))
371  2476 0602 FDIV10 DEC  R2                DEC Q
372  2478 6049        S    R9,R1             COMPUTE ABOVE FOR NEW Q
373  247A 15FD FDIV11 JGT  FDIV10            IF Q IS TOO BIG MAKE IT SMALLER
374  247C C082        MOV  R2,R2             IS Q ZERO82
375  247E 1329        JEQ  FDIV16            YES DO NO SUBTRACTING
376  2480 04C3        CLR  R3                CLEAR CARRY INTO FIRST BYTE
377  2482 C106        MOV  R6,R4             GET DIVISOR LOOP COUNT
378  2484 A3C6        A    R6,R15            TO LOW ORDER BYTE OF DIVIDEND
379  2486 C0C0 FDIV12 MOV  R0,R3             SAVE CARRY FROM PREVIOUS BYTE
380  2488 D824        MOVB @FDVSR(R4),@WSG+1  GET NEXT BYTE OF DIVISOR
380  248A 8354
380  248C 83A1
381  248E 3802        MPY  R2,R0             MPY BYTE OF DIVISOR BY QUOTIENT
382  2490 A043        A    R3,R1             ADD IN CARRY FROM LAST DIVISOR
383  2492 3C20        DIV  @LW100,R0         CONVERT RESULT TO A RADIX DIGIT
383  2494 24EE
384  2496 77E0        SB   @WSG+3,*R15        SUBTRACT PRODUCT BYTE FROM DIVIDEND
384  2498 83A3
385  249A 1504        JGT  FDIV13            IS RESULT POSITIVE
386  249C 1303        JEQ  FDIV13            OR ZERO ?
387  249E B7E0        AB   @LB100,*R15       NO ADD RADIX BACK
387  24A0 24EF
388  24A2 0580        INC  R0                INCREMENT PRODUCT CARRY
389  24A4 060F FDIV13 DEC  R15               POINT TO NEXT BYTE OF DIVIDEND
390  24A6 0604        DEC  R4                SUBTRACT ALL BYTES OF DIVISOR
391  24A8 15EE        JGT  FDIV12            NO CONTINUE
392  24AA 77E0        SB   @WSG+1,*R15        YES SUBTRACT CARRY FROM DIVISOR
392  24AC 83A1
393  24AE 1511        JGT  FDIV16            HIGH ORDER HIGHEST ORDER
394  24B0 1310        JEQ  FDIV16            DIVIDEND BYTE NEGATIVE RESULT
395  24B2 0602        DEC  R2                DEC Q WAS ONE TOO BIG
396  24B4 C106        MOV  R6,R4             GET ADD-BACK LOOP COUNT
397  24B6 A3C6        A    R6,R15            POINT TO LOW ORDER BYTE OF DIVIDEND
398  24B8 B7E4 FDIV14 AB   @FDVSR(R4),*R15   ADD BYTE OF DIVISOR TO DIVIDEND
398  24BA 8354
399  24BC 981F        CB   *R15,@LB100       RESULT LARGER THAN RADIX
399  24BE 24EF
400  24C0 1A05        JL   FDIV15            NO RESULT IS CORRECT
401  24C2 77E0        SB   @LB100,*R15       YES SUBTRACT RADIX
401  24C4 24EF
402  24C6 BBE0        AB   @LB1,@-1(R15)     ADD ONE FOR CARRY TO HIGH BYTE
402  24C8 24F3
402  24CA FFFF
403  24CC 060F FDIV15 DEC  R15               TO NEXT HIGHER BYTE OF DIVIDEND
404  24CE 0604        DEC  R4                DONE NOTHING IN ALL BYTES OF DIVIDEND
405  24D0 15F3        JGT  FDIV14            NO ADD IN THE NEXT ONE
406  24D2 D960 FDIV16 MOVB @WSG+5,@FAC+10(R5)  PUT AWAY NEXT QUOTIENT BYTE
406  24D4 83A5
406  24D6 8354
407  24D8 058F        INC  R15               HIGH ORDER OF NEXT SIGNIF DVDND
```

```
408  24DA 0585        INC  R5               COPUTE ALL NECESSARY BYTES
409  24DC 11B3        JLT  FDIV08           NO CONTINUE
410  24DE 0460        B    @FMEND           YES NORMAILIZE AND CONTINUE
410  24E0 22CA
                *
                *       COPY    'C:\Users\Lee\Downloads\TI-99-4A\TurboForth\MDOS\L10\CNS_TF.a99'
                *
  1             *
  2             *   PAUL HERE IS THE NEW CNS ROUTINE WHICH SHOULD FIX A BUNCH OF PROBLEMS
  3             *      PLEASE NOTE THE FORMAT OF THE CALL.  IF THIS IS DIFFERENT
  4             *      THAN WHAT YOU EXPECT (OR OTHERS) PLEASE TELL ME ABOUT IT
  5             *
  6             *      INPUT R0 = 18
  7             *            R1 = FLOAT1
  8             *            R2 = ADDRESS of STRING output buffer; 1st byte = length
  9             *            R7 = OPTION 1  (for TurboForth 1.2)
 10             *            R8 = OPTION 2  (for TurboForth 1.2)
 11             *            R9 = OPTION 3  (for TurboForth 1.2)
 12             *
 13             *      OUTPUT R0 = ERROR CODE
 14             *             R2 = ADDRESS of STRING output buffer; 1st byte = length
 15             *
 16             *-------------------------------------------------------------------------
 17             *
 18             *      WRITTEN: 07/17/1985
 19             *               TESTED AND MODIFIED 10/21/1985
 20             *      FILE:    WDS1.136.CNS
 21             *
 22             *      NAME:    CONVERT NUMBER TO STRING
 23             *
 24             *
 25             *       CCCCC    N     N    SSSSS
 26             *      C     C   NN    N   S     S
 27             *      C         N N   N   S
 28             *      C         N  N  N    SSSSS
 29             *      C         N   N N         S
 30             *      C     C   N    NN   S     S
 31             *       CCCCC    N     N    SSSSS
 32             *
 33             *
 34             *
 35             *-------------------------------------------------------------------------
 36             *
 37             *      CONVERT THE NUMBER IN THE FAC TO A STRING
 38             *      AS WAS CALLED BY BASIC
 39             *
 40             *      BL   @CNS
 41             *           FAC11        0 FOR FREE FORMAT(FAC12-FAC13 IGNORED)
 42             *                        BIT 0 ON FOR FIXED FORMAT
 43             *                        BIT 1 ON FOR AN EXPLICIT SIGN
 44             *                        BIT 2 ON TO OUTPUT THE SIGN OF A POS.
 45             *                        NO. AS A PLUS SIGN('+') INSTEAD OF A
 46             *                           SPACE (BIT 1 MUST ALSO BE ON)
 47             *                        BIT 3 ON FOR E-NOTATION OUTPUT
 48             *                           : BIT 4 ALSO ON FOR EXTENDED E-NOTATION
 49             *           FAC12 AND FAC13 SPECIFY THE FIELD SIZE
 50             *           FAC12        NUMBER OF PLACES IN THE FIELD TO THE
 51             *                        LEFT OF THE DECIMAL POINT INCLUDING AN
 52             *                        EXPLICIT SIGN AND EXCLUDING THE DECIMAL
 53             *                        POINT.
 54             *           FAC13        NUMBER OF PLACES IN THE FIELD TO THE
 55             *                        RIGHT OF THE DECIMAL POINT INCLUDING THE
 56             *                        DECIMAL POINT.
 57             *           FAC12 AND FAC13 EXCLUDE THE 4 POSITIONS FOR THE
 58             *                        EXPONENT IF BIT 3 IS ON.
 59             *
 60             *      ON INPUT USE THE FOLLOWING REGISTERS
 61             *           R0 CONTAINS FORMAT OPTIONS
 62             *           R1 CONTAINS NUMBER OF PLACES IN THE FIELD TO THE LEFT
 63             *                        OF THE DECIMAL POINT INCLUDING AN EXPLICIT
 64             *                        SIGN AND EXCLUDING THE DECIMAL POINT.
 65             *           R2 CONTAINS NUMBER OF PLACES IN THE FIELD TO THE RIGHT OF
 66             *                        THE DECIMAL POINT INCLUDING THE DECIMAL POINT
 67             *           R1 AND R2 EXCLUDE THE 4 POSITIONS FOR THE EXPONENT IF
```

```
 68           *                       BIT 3 IS ON
 69           *
 70           *      OTHER REGISTER USAGE
 71           *           R8   POINTS TO NUMBER TO BE CONVERTED
 72           *           R6   POINTS TO OUTPUT FIELD
 73           *           R10  POINTS TO END OF NUMBER (= TO FAC+8)
 74           *
 75           *      ERRORS: THE FIELD HAS MORE THAN 14 SIGNIFICANT DIGITS
 76           *
 77           CH0002
 78  24E2 0002 LWCNS  DATA >0002
 79  24E4 0004 LWCNP  DATA >0004
 80  24E6 0008 LWCNE  DATA >0008
 81  24E8 0010 LWCNF  DATA >0010
 82           *
 83           *    INTEGER POWER OF TEN TABLE
 84           *
 85  24EA 2710 CNSITT DATA 10000
 86  24EC 03E8        DATA 1000
 87  24EE 00   LW100  BYTE 0
 88  24EF 64   LB100  BYTE 100
 89  24F0 00   LW10   BYTE 0
 90  24F1 0A   LB10   BYTE 10
 91  24F2 00          BYTE 0
 92  24F3 01   LB1    BYTE 1
 93           *
 94           *    CONSTANTS
 95           *
 96  24F4 20   LBSPC  BYTE ' '
 97  24F5 2A   LBAST  BYTE '*'
 98  24F6 2E   LBPER  BYTE '.'
 99  24F7 45   LBE    BYTE 'E'
100  24F8 30   LBZER  BYTE '0'
101  24F9 59   CBH59  BYTE >59
102  24FA 63   CBH63  BYTE >63
103           *===============================================================================
104           *
105           *    CONVERT NUMBER TO STRING
106           *
107           *===============================================================================
108  24FB 0000        EVEN     *>>> Assembler Auto-Generated <<<
109  24FC C80D CNS    MOV  R13,@SAVR13
109  24FE 20DE
110  2500 C20D        MOV  R13,R8                SET UP POINTERS
111           *
112           **les** This is convoluted!! R8 is also used to reference the passed params!
113           *
114           * Move number to FAC
115           *
116  2502 C06D        MOV  @2(R13),R1      get pointer to number
116  2504 0002
117  2506 0202        LI   R2,FAC
117  2508 834A
118  250A 06A0        BL   @R1$2           move 4 words from *R1 to *R2    **les**
118  250C 2092
119           *
120  250E C02D        MOV  @14(R13),R0     GET OPT #1                          **les**
120  2510 000E
121  2512 C06D        MOV  @16(R13),R1     GET OPT #2                          **les**
121  2514 0010
122  2516 C0AD        MOV  @18(R13),R2     GET OPT #3                          **les**
122  2518 0012
123           *
124           *
125  251A 0206        LI   R6,FAC+11              GET LOCATION OF DESTINATION
125  251C 8355
126           **les**        ^^^^^^  String placed here with leading space [next instr]
127  251E DDA0        MOVB @LBSPC,*R6+
127  2520 24F4
128  2522 0203        LI   R3,'-'*256
128  2524 2D00
129  2526 0760        ABS  @FAC                   IS NUMBER POSITIVE?
129  2528 834A
130  252A 1107        JLT  CNS01                  IS NEGATIVE
```

```
131  252C 0203        LI   R3,' '*256            SO MASK IS EITHER + OR " "
131  252E 2000
132  2530 2420        CZC  @LWCNP,R0             IS IT REQUIRED SIGN?
132  2532 24E4
133  2534 1302        JEQ  CNS01                 NO
134  2536 0203        LI   R3,'+'*256            YES, REQUIRED SIGN OF NUMBER
134  2538 2B00
135  253A DD83 CNS01  MOVB R3,*R6+
136  253C C000        MOV  R0,R0                 IS IT FREE FORM FLOATING OUTPUT?
137  253E 166E        JNE  CNSX                  NO, IS FORMATTED
138            *
139            *    FREE FORMAT FLOATING OUTPUT
140            *
141  2540 C120        MOV  @FAC,R4
141  2542 834A
142  2544 1606        JNE  CNSF1
143  2546 DDA0        MOVB @LBZER,*R6+
143  2548 24F8
144  254A D5A0        MOVB @LWCNS,*R6
144  254C 24E2
145  254E 0460        B    @CNSMLS               FINISH UP
145  2550 2888
146
147  2552 06A0 CNSF1  BL   @CNSTEN               GET BASE 10 EXPONENT, IS NO.
147  2554 279A
148            *                                 LESS THAN ONE?
149  2556 1110        JLT  CNSF02                YES-IT CANNOT BE PRINTED AS AN INT.
150  2558 028D        CI   R13,9                 NO-IS NUMBER TOO BIG TO PRINT AS INT.?
150  255A 0009
151  255C 150D        JGT  CNSF02                YES-SO ROUND NO. FOR E-NOTATION OUTPUT
152  255E D020        MOVB @FAC,R0               NO-CHECK IF THE NUMBER IS AN INT.
152  2560 834A
153            *                                 GET EXP. HIGH BYTE IS STILL ZERO
154  2562 06C0        SWPB R0
155  2564 0220        AI   R0,FAC-62
155  2566 830C
156  2568 D070 CNSF01 MOVB *R0+,R1               IS NEXT BYTE OF FRACTION ZERO
157  256A 1606        JNE  CNSF02                NO-PRINT NUM IN FIXED FORMAT
158  256C 0280        CI   R0,FAC+8              HAVE WE REACHED THE END OF NUMBER
158  256E 8352
159  2570 1AFB        JL   CNSF01                NOT YET
160  2572 04E0        CLR  @WSM10                SET INTEGER TYPE OUTPUT
160  2574 20E4
161  2576 1013        JMP  CNSF05
162
163  2578 0201 CNSF02 LI   R1,5                  ASSUME ROUNDING FOR E-NOTATION
163  257A 0005
164  257C 028D        CI   R13,9                 IS NO. TOO BIG FOR FIXED POINT OUTPUT?
164  257E 0009
165  2580 150A        JGT  CNSF04                YES-ROUND FOR E-NOTATION
166  2582 028D        CI   R13,-4                NO-IS NUMBER TOO SMALL FOR FIXED POINT
166  2584 FFFC
167  2586 1107        JLT  CNSF04                YES ROUND FOR E-NOTATION OUTPUT
168  2588 0201        LI   R1,9                  FORCE R1 TO = 9
168  258A 0009
169  258C 028D        CI   R13,-2                NO-WILL NO, BE PRINTED WITH MAX. NO. OF
169  258E FFFE
170            *                                 FIXED FORMAT SIGNIFICANT DIGITS?
171  2590 1502        JGT  CNSF04                YES-ROUND ACCORDINGLY
172  2592 0581        INC  R1                    NO-ROUND NUMBER FOR MAX. SIGNIFICANT
173            *                                 DIGITS (R1=10)
174  2594 A04D        A    R13,R1                THAT CAN BE [RINTED FOR THIS NUMBER
175  2596 06A0 CNSF04 BL   @CNSRND               ROUND NO. ACCORDINGLY ROUNDING CAN
175  2598 2726
176  259A 0720        SETO @WSM10                CHANGE THE EXPONENT AND SO THE PRINT FORMAT
176  259C 20E4
177            *                                 TO BE USED, SET NON-INTEGER PRINT FLAG
178  259E 028D CNSF05 CI   R13,9                 DECIDE WHICH PRINT FORMAT TO USE
178  25A0 0009
179  25A2 152B        JGT  CNSG                  TOO BIG FOR FIXED FORMAT
180  25A4 028D        CI   R13,-6                USE E-NOTATION NUMBER IN RANGE FOR MAX.
180  25A6 FFFA
181            *                                 FIXED POINT DIGITS?
182  25A8 1515        JGT  CNSF08                YES-USE FIXED FORMAT OUTPUT
```

```
183  25AA 028D        CI   R13,-10               NO-NO. TOO SMALL FOR FIXED FORMAT?
183  25AC FFF6
184  25AE 1125        JLT  CNSG                  YES-USE E-NOTATION OUTPUT
185               *                             NO- NO. OF SIGNIFICANT DIGITS WILL
186               *                             DETERMINE FIXED FORMAT OUTPUT OR NOT
187  25B0 0200        LI   R0,FAC+8              GET POINTER TO LAST BYTE
187  25B2 8352
188  25B4 0203        LI   R3,4                  4=15-11 GET NO. OF DIGITS+2-EXPONENT
188  25B6 0004
189  25B8 A0C7        A    R7,R3                 SCALE FACTOR TAKE INTO ACCOUNT A LEADING
190               *                             ZERO IN FAC+1
191  25BA 0643 CNSF06 DECT R3                    DECREMENT SIG DIGIT COUNT FOR LAST ZERO
192               *                             BYTE
193  25BC 0600        DEC  R0                    POINT TO NEXT MOST SIG BYTE OF FAC
194  25BE D050        MOVB *R0,R1                IS NEXT BYTE ALL ZERO?
195  25C0 13FC        JEQ  CNSF06                YES-CONTINUE LOOKING FOR LEAST SIG BYTE
196               *                             NO-FOUND THE LEAST SIG BYTE, THIS LOOP
197               *                             WILL ALWAYS TERMINATE SINCE FAC+1 NEVER 0
198  25C2 04C0        CLR  R0                    TAKE INTO ACCOUNT IF THE LEAST SIG BYTE IS
199               *                             DIVISIBLE BY TEN
200  25C4 0981        SRL  R1,8
201  25C6 3C20        DIV  @LW10,R0              DIVIDE LSB BY 10
201  25C8 24F0
202  25CA C041        MOV  R1,R1                 IS THE REMAINDER ZERO?
203  25CC 1601        JNE  CNSF07                NO-SIGNIFICANT DIGIT COUNT IS CORRECT
204  25CE 0603        DEC  R3                    YES-LS BYTE HAS A TRAILING 0
205  25D0 8343 CNSF07 C    R3,R13                TOO MANY SIG DIGITS FOR FIXED FORMAT?
206  25D2 1513        JGT  CNSG                  YES-USE E-NOTATION
207               *
208               *    FREE FORMAT FIXED POINT AND INTEGER
209               *    FLOATING OUTPUT
210               *
211  25D4 0203 CNSF08 LI   R3,12
211  25D6 000C
212  25D8 6347        S    R7,R13
213  25DA 1106        JLT  CNSF12
214  25DC 0204        LI   R4,3
214  25DE 0003
215  25E0 A10D        A    R13,R4
216  25E2 06A0 CNSF10 BL   @CNSDIG
216  25E4 27B6
217  25E6 1006        JMP  CNSF11
218
219  25E8 0700 CNSF12 SETO R0
220  25EA 600D        S    R13,R0
221  25EC 06A0        BL   @CNSPER
221  25EE 2876
222  25F0 04C4        CLR  R4
223  25F2 10F7        JMP  CNSF10
224
225  25F4 06A0 CNSF11 BL   @CNSUTR
225  25F6 28CC
226  25F8 100B        JMP  CNSG01
227               *
228               *    FREE FORMAT E-NOTATION FLOATING OUTPUT
229               *
230  25FA 0203 CNSG   LI   R3,8
230  25FC 0008
231  25FE 0204        LI   R4,3
231  2600 0003
232  2602 6107        S    R7,R4
233  2604 06A0        BL   @CNSDIG
233  2606 27B6
234  2608 06A0        BL   @CNSUTR
234  260A 28CC
235  260C 06A0        BL   @CNSEXP
235  260E 27FE
236  2610 0460 CNSG01 B    @CNSMLS
236  2612 2888
237               *
238               *   FIELD TOO BIG, SO LENGTH IS 0 ERROR
239               *
240  2614 04C0 FLDBIG CLR  R0       LENGTH IS ZERO
241  2616 04C6        CLR  R6       POINTER IS ZERO
```

```
242  2618 0460        B    @CNSS02
242  261A 293E
243            *
244            *    FIXED FORMAT OUTPUT
245            *
246  261C C102 CNSX   MOV  R2,R4
247  261E A101        A    R1,R4
248  2620 0284        CI   R4,14
248  2622 000E
249  2624 15F7        JGT  FLDBIG         FIELD SIZE IS TOO LARGE
250  2626 2420        CZC  @LWCNE,R0
250  2628 24E6
251  262A 1606        JNE  CNSX01
252  262C 0283        CI   R3,'-'*256
252  262E 2D00
253  2630 1303        JEQ  CNSX01
254  2632 2420        CZC  @LWCNS,R0
254  2634 24E2
255  2636 1306        JEQ  CNSX02
256  2638 0601 CNSX01 DEC  R1
257  263A 1504        JGT  CNSX02
258  263C 0283        CI   R3,'-'*256
258  263E 2D00
259  2640 1301        JEQ  CNSX02
260  2642 04C1        CLR  R1
261  2644 C801 CNSX02 MOV  R1,@WSM6
261  2646 20E0
262  2648 1110        JLT  CNSJ04
263  264A 0602        DEC  R2
264  264C 1501        JGT  CNSX03
265  264E 04C2        CLR  R2
266  2650 C802 CNSX03 MOV  R2,@WSM8
266  2652 20E2
267  2654 C101        MOV  R1,R4
268  2656 A102        A    R2,R4
269  2658 1308        JEQ  CNSJ04
270            *
271            *    FIXED FORMAT FLOATING OUTPUT
272            *
273  265A 06A0        BL   @CNSTEN
273  265C 279A
274  265E 2420        CZC  @LWCNE,R0
274  2660 24E6
275  2662 1645        JNE  CNSK
276            *
277            *    FIXED FORMAT FLOATING F-FORMAT
278            *
279  2664 880D        C    R13,@WSM6
279  2666 20E0
280  2668 1102        JLT  CNSJ00
281  266A 0460 CNSJ04 B    @CNSAST
281  266C 2908
282  266E C04D CNSJ00 MOV  R13,R1
283  2670 A042        A    R2,R1
284  2672 0281        CI   R1,-1
284  2674 FFFF
285  2676 112A        JLT  CNSVZR
286  2678 06A0        BL   @CNSRND
286  267A 2726
287  267C 6347        S    R7,R13
288  267E 110D        JLT  CNSJ01
289  2680 0700        SETO R0
290            *
291            *
292            *
293  2682 A020        A    @WSM6,R0
293  2684 20E0
294  2686 600D        S    R13,R0
295  2688 06A0        BL   @CNSZER
295  268A 2880
296  268C 0203        LI   R3,3
296  268E 0003
297  2690 A0CD        A    R13,R3
298  2692 C103        MOV  R3,R4
```

```
299  2694 A0E0        A    @WSM8,R3
299  2696 20E2
300  2698 1011        JMP  CNSJ02
301  269A C0E0 CNSJ01 MOV  @WSM8,R3
301  269C 20E2
302  269E 1316        JEQ  CNSVZR
303  26A0 C020        MOV  @WSM6,R0
303  26A2 20E0
304  26A4 0580        INC  R0
305  26A6 06A0        BL   @CNSZER
305  26A8 2880
306  26AA C306        MOV  R6,R12
307  26AC 0700        SETO R0
308  26AE 600D        S    R13,R0
309  26B0 06A0        BL   @CNSPER
309  26B2 2876
310  26B4 A0CD        A    R13,R3
311  26B6 0223        AI   R3,3
311  26B8 0003
312  26BA 04C4        CLR  R4
313  26BC 06A0 CNSJ02 BL   @CNSDIG
313  26BE 27B6
314  26C0 C028        MOV  @18(R8),R0      **les** for TurboForth
314  26C2 0012
315  26C4 1601        JNE  CNSJ03
316  26C6 D700        MOVB R0,*R12
317  26C8 0460 CNSJ03 B    @CNSCHK
317  26CA 28E2
318              *
319              *    FIXED FORMAT OUTPUT OF ZERO
320              *
321  26CC C020 CNSVZR MOV  @WSM6,R0
321  26CE 20E0
322  26D0 0580        INC  R0
323  26D2 06A0        BL   @CNSZER
323  26D4 2880
324  26D6 C306        MOV  R6,R12
325  26D8 C028        MOV  @18(R8),R0      **les** for TF
325  26DA 0012
326  26DC 1302        JEQ  CNSV01
327  26DE 06A0        BL   @CNSPER
327  26E0 2876
328  26E2 C028 CNSV01 MOV  @14(R8),R0      **les** for TF
328  26E4 000E
329  26E6 2420        CZC  @LWCNE,R0
329  26E8 24E6
330  26EA 13EE        JEQ  CNSJ03
331  26EC 1019        JMP  CNSK01
332              *
333              *    FIXED FORMAT FLOATING E-FORMAT
334              *
335  26EE C160 CNSK   MOV  @FAC,R5
335  26F0 834A
336  26F2 1603        JNE  CNSK1
337  26F4 04C7        CLR  R7
338  26F6 04CD        CLR  R13
339  26F8 10E9        JMP  CNSVZR
340  26FA A042 CNSK1  A    R2,R1
341  26FC 0601        DEC  R1
342  26FE 06A0        BL   @CNSRND
342  2700 2726
343  2702 C0E0        MOV  @WSM6,R3
343  2704 20E0
344  2706 6343        S    R3,R13
345  2708 058D        INC  R13
346  270A 60C7        S    R7,R3
347  270C 05C3        INCT R3
348  270E C103        MOV  R3,R4
349  2710 A0E0        A    @WSM8,R3
349  2712 20E2
350  2714 06A0        BL   @CNSDIG
350  2716 27B6
351  2718 C028        MOV  @18(R8),R0      **les** for TF
351  271A 0012
```

```
352  271C 1601        JNE  CNSK01
353  271E 0606        DEC  R6
354  2720 06A0 CNSK01 BL   @CNSEXP
354  2722 27FE
355  2724 10D1        JMP  CNSJ03
356            *
357            *
358            *
359            *
360            *
361  2726 C28B CNSRND MOV  R11,R10
362  2728 6341        S    R1,R13
363  272A 6047        S    R7,R1
364  272C 0811        SRA  R1,1
365  272E 05C1        INCT R1
366  2730 0203        LI   R3,49*256
366  2732 3100
367  2734 081D        SRA  R13,1
368  2736 1702        JNC  CNSR01
369  2738 0203        LI   R3,4*256
369  273A 0400
370  273C 0281 CNSR01 CI   R1,7
370  273E 0007
371  2740 152B        JGT  CNSR05
372  2742 0207        LI   R7,FAC
372  2744 834A
373  2746 04CC        CLR  R12
374  2748 D357        MOVB *R7,R13
375  274A D10D        MOVB R13,R4
376  274C 098D        SRL  R13,8
377  274E A1C1        A    R1,R7
378  2750 B5C3        AB   R3,*R7
379  2752 D160        MOVB @FAC,R5
379  2754 834A
380  2756 0985        SRL  R5,8
381  2758 C805        MOV  R5,@EXP
381  275A 836C
382  275C D805        MOVB R5,@SIGN
382  275E 836E
383  2760 06A0        BL   @ROUNUP          USES ONLY R0-R2,R11 FOR RETURN
383  2762 2310
384  2764 04C1        CLR  R1
385  2766 0287        CI   R7,FAC+1
385  2768 834B
386  276A 1603        JNE  CNSR02
387  276C 9120        CB   @FAC,R4
387  276E 834A
388  2770 160E        JNE  CNSR03
389  2772 0283 CNSR02 CI   R3,4*256
389  2774 0400
390  2776 160C        JNE  CNSR04
391  2778 04C0        CLR  R0
392  277A 06C1        SWPB R1
393  277C D057        MOVB *R7,R1
394  277E 06C1        SWPB R1
395  2780 3C20        DIV  @LW10,R0
395  2782 24F0
396  2784 3820        MPY  @LW10,R0
396  2786 24F0
397  2788 06C1        SWPB R1
398  278A D5C1        MOVB R1,*R7
399  278C 06C1        SWPB R1
400  278E 0587 CNSR03 INC  R7
401  2790 DDC1 CNSR04 MOVB R1,*R7+           ZERO NEXT BYTE OF FAC
402  2792 0287        CI   R7,FAC+8          DONE ZEROING THE RESR OF THE FAC
402  2794 8352
403  2796 1AFC        JL   CNSR04            NO - CONTINUE ZEROING THE REST
404  2798 C2CA CNSR05 MOV  R10,R11           YES RESTORE RETURN ADDRESS
405            *
406            *    GET BASE TEN EXPONENT OF THE NUMBER IN FAC
407            *
408            *
409  279A 020D CNSTEN LI   R13,->4000        NEGATIVE BIAS
409  279C C000
```

```
410  279E B360        AB   @FAC,R13          GET BASE 1 HUNDRED EXPONENT OF FAC
410  27A0 834A
411  27A2 087D        SRA  R13,7             MULTIPLY IT BY TWO AND PUT IT IN LOW BYTE
412  27A4 04C7        CLR  R7                THE HIGH BIT OF FAC+1 IS ALWAYS OFF
413  27A6 9820        CB   @FAC+1,@CBHA      IS FIRST DIGIT OF FAC ONE DECIMAL
413  27A8 834B
413  27AA 20D2
414  27AC 1102        JLT  CNST01
415  27AE 058D        INC  R13               YES - BASE TEN EXPONENT IS EVEN
416  27B0 0587        INC  R7                THIS MAKES THE BASE TEN EXPON ODD
417  27B2 C34D CNST01 MOV  R13,R13           SET STATUS BITS TO REFLECT BASE TEN EXPO
418  27B4 045B CNSDRT RT
419            *
420            *
421            *
422            *
423  27B6 C28B CNSDIG MOV  R11,R10
424  27B8 04E0        CLR  @FAC+8
424  27BA 8352
425  27BC 04C1        CLR  R1
426  27BE 0202        LI   R2,FAC+1
426  27C0 834B
427  27C2 06A0        BL   @CNSD03
427  27C4 27E6
428  27C6 04C0 CNSD01 CLR  R0
429  27C8 06C1        SWPB R1
430  27CA D072        MOVB *R2+,R1
431  27CC 06C1        SWPB R1
432  27CE 3C20        DIV  @LW10,R0
432  27D0 24F0
433  27D2 06A0        BL   @CNSD02
433  27D4 27DC
434  27D6 C001        MOV  R1,R0
435  27D8 020B        LI   R11,CNSD01
435  27DA 27C6
436  27DC 0220 CNSD02 AI   R0,'0'
436  27DE 0030
437  27E0 06C0        SWPB R0
438  27E2 DD80        MOVB R0,*R6+
439  27E4 06C0        SWPB R0
440  27E6 0604 CNSD03 DEC  R4
441  27E8 1603        JNE  CNSD04
442  27EA C306        MOV  R6,R12
443  27EC DDA0        MOVB @LBPER,*R6+
443  27EE 24F6
444            *
445            *    VSPTR IS AT CPU ADDRESS >6E    **les** ???
446            *    MAKE SURE NOT TO DESTROY IT    **les** ???
447            *
448  27F0 0286 CNSD04 CI   R6,FAC+30
448  27F2 8368
449  27F4 1402        JHE  CNSD06
450  27F6 0603        DEC  R3
451  27F8 15DD        JGT  CNSDRT
452  27FA D583 CNSD06 MOVB R3,*R6
453
454  27FC 045A        B    *R10
455            *
456            *
457            *
458  27FE C28B CNSEXP MOV  R11,R10
459  2800 C80C        MOV  R12,@SAVR12     SAVE R12 TEMPORARILY
459  2802 20DC
460  2804 DDA0        MOVB @LBE,*R6+
460  2806 24F7
461  2808 0200        LI   R0,'-'*256
461  280A 2D00
462  280C 074D        ABS  R13
463  280E 1102        JLT  CNSE01
464  2810 0200        LI   R0,'+'*256
464  2812 2B00
465  2814 DD80 CNSE01 MOVB R0,*R6+
466  2816 0200        LI   R0,LW100
466  2818 24EE
```

```
467  281A 028D        CI   R13,100
467  281C 0064
468  281E 110B        JLT  CNSE02
469  2820 C328        MOV  @14(R8),R12      **les** for TF
469  2822 000E
470  2824 1303        JEQ  CNSE04
471  2826 2720        CZC  @LWCNF,R12
471  2828 24E8
472  282A 1605        JNE  CNSE02
473  282C DDA0 CNSE04 MOVB @LBAST,*R6+
473  282E 24F5
474  2830 DDA0        MOVB @LBAST,*R6+
474  2832 24F5
475  2834 100E        JMP  CNSE03
476
477  2836 06A0 CNSE02 BL   @CNSI01
477  2838 2860
478  283A C028        MOV  @14(R8),R0      **les** for TF
478  283C 000E
479  283E 1303        JEQ  CNSE05
480  2840 2420        CZC  @LWCNF,R0
480  2842 24E8
481  2844 1606        JNE  CNSE03
482  2846 0226 CNSE05 AI   R6,-3
482  2848 FFFD
483  284A DDA6        MOVB @01(R6),*R6+
483  284C 0001
484  284E DDA6        MOVB @01(R6),*R6+
484  2850 0001
485  2852 D5A0 CNSE03 MOVB @LW10,*R6
485  2854 24F0
486  2856 C320        MOV  @SAVR12,R12  RESTORE R12
486  2858 20DC
487  285A 045A        B    *R10
488            *
489            *
490            *    CONVERT AN UNSIGNED INTEGER INTO AN ACSII STRING
491            *
492            *
493  285C 0200 CNSINT LI   R0,CNSITT  **les** This instr is never executed!!
493  285E 24EA
494  2860 04CC CNSI01 CLR  R12
495  2862 3F30        DIV  *R0+,R12
496  2864 022C        AI   R12,'0'
496  2866 0030
497  2868 06CC        SWPB R12
498  286A DD8C        MOVB R12,*R6+
499  286C 0280        CI   R0,CNSITT+10
499  286E 24F4
500  2870 1AF7        JL   CNSI01
501  2872 D58C        MOVB R12,*R6
502  2874 045B        RT
503            *
504            *
505            *
506  2876 DDA0 CNSPER MOVB @LBPER,*R6+
506  2878 24F6
507  287A 1002        JMP  CNSZER
508  287C DDA0 CNSZ01 MOVB @LBZER,*R6+
508  287E 24F8
509  2880 0600 CNSZER DEC  R0
510  2882 15FC        JGT  CNSZ01
511  2884 D580        MOVB R0,*R6
512  2886 045B        RT
513            *
514            *    SUPPRESS LEADING ZEROS
515            *
516  2888 020B CNSMLS LI   R11,CNSSTR        ENTRY TO FINISH UP NUMBER AFTERWARD
516  288A 2934
517  288C 0206 CNSLEA LI   R6,FAC+12         GET POINTER TO SIGN
517  288E 8356
518  2890 D056        MOVB *R6,R1            GET SIGN
519  2892 DDA0 CNSL01 MOVB @LBSPC,*R6+       PUT A SPACE WHERE THE ZERO OS SIGN WAS
519  2894 24F4
```

```
520  2896 9816        CB   *R6,@LBZER        IS THE NEXT BYTE ZERO
520  2898 24F8
521  289A 13FB        JEQ  CNSL01            YES SUPPRESS IT
522  289C D016        MOVB *R6,R0            NO IS THIS THE END OF THE NUMBER
523  289E 130F        JEQ  CNSL02            YES PUT THE ZERO BACK
524  28A0 9800        CB   R0,@LBE           NO IS THIS THE START OF THE EXPONENT
524  28A2 24F7
525  28A4 130C        JEQ  CNSL02            YES PUT THE ZERO BACK IN
526  28A6 9800        CB   R0,@LBPER         NO IS THIS THE DECMIAL POINT
526  28A8 24F6
527  28AA 160C        JNE  CNSL03            NO PUT THE SIGN BACK IN
528  28AC C028        MOV  @14(R8),R0        YES IS FREE FORMAT OUTPUT (for TF)      **les**
528  28AE 000E
529  28B0 1609        JNE  CNSL03            NO THEN PUT THE SIGN BACK IN FIX FORMAT
530  28B2 D026        MOVB @01(R6),R0        YES ANY DIGITS TO RIGHT OF DECMINAL POINT
530  28B4 0001
531  28B6 1303        JEQ  CNSL02            NO END OF NUMBER PUT ZERO BACK IN
532  28B8 9800        CB   R0,@LBE           DOES EXPONENT START AFTER DECMINAL POINT
532  28BA 24F7
533  28BC 1603        JNE  CNSL03            NO PUT THE SIGN BACK
534  28BE 0606 CNSL02 DEC  R6                YES POINT BACK TO WHERE THE ZERO WAS
535  28C0 D5A0        MOVB @LBZER,*R6        PUT THE ZERO BACK IN THE NUMBER
535  28C2 24F8
536  28C4 0606 CNSL03 DEC  R6                POINT BACK TO WHERE THE SIGN WILL GO
537  28C6 D581        MOVB R1,*R6            PUT THE SIGN BACK IN THE BUFFER
538  28C8 045B        RT
539            *
540            *    REMOVE TRAILING ZEROS
541            *
542  28CA 0606 CNSU01 DEC  R6
543  28CC 9826 CNSUTR CB   @-01(R6),@LBZER
543  28CE FFFF
543  28D0 24F8
544  28D2 13FB        JEQ  CNSU01
545  28D4 C820        MOV  @WSM10,@WSM10
545  28D6 20E4
545  28D8 20E4
546  28DA 1601        JNE  CNSU02
547  28DC C18C        MOV  R12,R6
548            *
549            *
550            *
551  28DE D583 CNSU02 MOVB R3,*R6
552  28E0 045B        RT
553            *
554            *    SET UP POINTER TO THE BEGINNING OF A FIXED FORMAT
555            *    FIELD AND SEE IF THE FIELD IS LARGE ENOUGH, AND
556            *    FINISH UP.
557            *
558  28E2 06A0 CNSCHK BL   @CNSLEA
558  28E4 288C
559  28E6 C18C        MOV  R12,R6
560  28E8 61A8        S    @16(R8),R6        **les** for TF
560  28EA 0010
561  28EC 9826        CB   @-01(R6),@LBSPC
561  28EE FFFF
561  28F0 24F4
562  28F2 160A        JNE  CNSAST
563  28F4 C028        MOV  @14(R8),R0        **les** for TF
563  28F6 000E
564  28F8 2420        CZC  @LWCNS,R0
564  28FA 24E2
565  28FC 131B        JEQ  CNSSTR
566  28FE 9816        CB   *R6,@LBSPC
566  2900 24F4
567  2902 1318        JEQ  CNSSTR
568  2904 9056        CB   *R6,R1
569  2906 1316        JEQ  CNSSTR
570            *
571            *
572            *
573  2908 C028 CNSAST MOV  @14(R8),R0     OPTION 1                                    **les**
573  290A 000E
574  290C C068        MOV  @16(R8),R1     OPTION 2                                    **les**
```

```
574  290E 0010
575  2910 A068        A    @18(R8),R1     OPTION 2 + OPTION 3 (# of places, excl. exp.) **les**
575  2912 0012
576  2914 2420        CZC  @LWCNE,R0
576  2916 24E6
577  2918 1305        JEQ  CNSA01
578  291A 8C71        C    *R1+,*R1+
579  291C 2420        CZC  @LWCNF,R0
579  291E 24E8
580  2920 1301        JEQ  CNSA01
581  2922 0581        INC  R1
582  2924 0206 CNSA01 LI   R6,FAC+12
582  2926 8356
583  2928 C006        MOV  R6,R0
584  292A DC20 CNSA02 MOVB @LBAST,*R0+
584  292C 24F5
585  292E 0601        DEC  R1
586  2930 15FC        JGT  CNSA02
587  2932 D401        MOVB R1,*R0
588            *
589            *    FINISH UP. COMPUTE THE LENGTH OF THE
590            *               STRING AND RETURN THE STRING TO THE USER
591            *                                    AND THE STRING LENGTH
592            *
593  2934 C006 CNSSTR MOV  R6,R0
594  2936 D070 CNSS01 MOVB *R0+,R1
595  2938 16FE        JNE  CNSS01
596  293A 0600        DEC  R0
597  293C 6006        S    R6,R0        LENGTH IN R0, STRING ADDRESS IN R6
598  293E C360 CNSS02 MOV  @SAVR13,R13
598  2940 20DE
599  2942 04DD        CLR  *R13             assume no error
600  2944 C16D        MOV  @4(R13),R5   caller's R2 pointer to result string  **les**
600  2946 0004
601  2948 06C0        SWPB R0           get string length to high byte        **les**
602  294A DD40        MOVB R0,*R5+      string length byte to dest byte       **les**
603            **les**       ^^^^---to 1st byte of dest string; inc to next byte
604  294C 0980        SRL  R0,8         get string length back to low byte    **les**
605  294E 1602        JNE  CNSEX1
606  2950 071D        SETO *R13             ERROR
607  2952 1003        JMP  CNSS03           CNSS03
608
609            *
610            CNSEX1
611            *
612            **les***  Copy string to destination
613            *
614  2954 DD76 CNSS04 MOVB *R6+,*R5+  <---starts at 2nd byte of destination  **les**
615  2956 0600        DEC  R0
616  2958 16FD        JNE  CNSS04
617
618  295A C01D CNSS03 MOV  *R13,R0          GET STATUS
619  295C 02CF        STST R15
620  295E 0380        RTWP
621            *
               *
               *       COPY    'C:\Users\Lee\Downloads\TI-99-4A\TurboForth\MDOS\L10\CSNN_TF.a99'
               *
  1            *---------------------------------------------------------------------------
  2            *
  3            *      WRITTEN: 05/06/1987
  4            *
  5            *      FILE:    WDS1.NEW.CSN
  6            *
  7            *      NAME:    CONVERT STRING TO NUMBER
  8            *                STRING IN CPU RAM
  9            *---------------------------------------------------------------------------
 10
 11
 12            *===========================================================================
 13            *
 14            *    NAME:  CSN
 15            *
 16            *    WHAT:  CONVERT STRING TO NUMBER
```

```
17            *
18            *    REG:   DESCRIPTION
19            *     0     POINTER               CURRENT PGM CHAR
20            *     1     EXPONENT SIGN
21            *     2     POINTER TO CHACTER AFTER REQ POSITION OF FIRST CHARACTER
22            *     3     ADDRESS OF GETCH
23            *     4     ACCUMULATOR FOR
24            *     5     ACCUMULATOR FOR
25            *     6     TEXT POINTER
26            *     7     RELATIVE POSITION
27            *     8     CURRENT PGM CHAR
28            *     9     PROGRAM STACK
29            *     10    LENGTH OF STRING
30            *     11
31            *     12    POINTER TO FIRST NON-ZERO CHARACTER
32            *
33            *    ENTRY: BL   @CSN
34            *
35            *    INPUT: R0 = 17
36            *           R1 = address of FP NUMBER result  <---**les**
37            *                (actually passed in FAC from this routine unless = 0)
38            *           R2 = ADDRESS OF STRING (WORD)<---**les** 1st byte = length
39            *
40            *   OUTPUT: R0 = CONTAINS 0 IF NO ERROR, ELSE CONTAINS ERROR
41            *           R1 = address of FP NUMBER result  <---**les**
42            *                (actually passed in FAC from this routine unless = 0)
43            *
44            *==============================================================================
45  2960 C80B CSN    MOV  R11,@SAVCSN
45  2962 20DA
46  2964 04E0        CLR  @FAC+10             **les** presume no error
46  2966 8354
47  2968 C0ED        MOV  @4(R13),R3        GET string LENGTH pointer  <---for TF   **les**
47  296A 0004
48  296C C183        MOV  R3,R6     **les** INITIALIZE TEXT POINTER
49  296E 0586        INC  R6                (point to 2nd byte)                      **les**
50  2970 D0D3        MOVB *R3,R3            get string length                        **les**
51  2972 0983        SRL  R3,8              get into right byte                      **les**
52  2974 0283        CI   R3,0                                                       **les**
52  2976 0000
53  2978 1356        JEQ  CSNZER            ZERO LENGTH STRING CONVERTS TO 0
54  297A 0707        SETO R7                START OF RELATIVE EXPR
55  297C 04C8        CLR  R8                ZERO OUT LSB OF R8
56  297E 04C4        CLR  R4                ZERO OUT EXP
57  2980 04E0        CLR  @SIGN             ASSUME POSITIVE NUMBER
57  2982 836E
58  2984 C306        MOV  R6,R12            ASSUME NUMBER STARTS HERE
59  2986 D236        MOVB *R6+,R8           GET CHARACTER
60  2988 C086        MOV  R6,R2
61  298A 0288        CI   R8,XPLUS          IS CHARACTER A PLUS
61  298C 2B00
62  298E 1305        JEQ  CSN02             YES IGNORE
63  2990 0288        CI   R8,XMINUS         IS CHARACTER A MINUS
63  2992 2D00
64  2994 1606        JNE  CSN04             YES IGNORE
65  2996 0720        SETO @SIGN             NUMBER IS NEGATIVE
65  2998 836E
66  299A 0582 CSN02  INC  R2
67  299C 0603 CSN03  DEC  R3                ONE LESS TO GET
68  299E 1343        JEQ  CSNZER            ALL ZERO'S
69  29A0 D236        MOVB *R6+,R8           MSB OF R8
70  29A2 0288 CSN04  CI   R8,X3000          NOW EAT UP ALL LEADING 0'S
70  29A4 3000
71  29A6 13FA        JEQ  CSN03
72
73  29A8 0288        CI   R8,XDOT           DO WE START WITH A "." ?
73  29AA 2E00
74  29AC 1304        JEQ  CSN09             IT IS THE START OF A DECIMAL
75  29AE C306        MOV  R6,R12
76  29B0 060C        DEC  R12               NOW R12 POINTS TO START
77  29B2 1017        JMP  CSN06
78
79
80  29B4 0607 CSN10  DEC  R7                EXP IS ONE LESS
```

```
 81  29B6 0603 CSN09  DEC  R3
 82  29B8 1336        JEQ  CSNZER            IT IS ALL ZERO'S
 83  29BA C306        MOV  R6,R12            START RIGHT AFTER THE "."
 84  29BC D236        MOVB *R6+,R8
 85  29BE 0288        CI   R8,X3000          NOW GET THE RELATIVE EXP OF THE NUMBER
 85  29C0 3000
 86  29C2 13F8        JEQ  CSN10             SO ELIMINATE LEADING 0'S AFTER DPT
 87  29C4 1A35        JL   CSNG16            WE MUST BE DONE, SO DIDN'T CONVERT ENTIRE #
 88  29C6 0288 CSN11  CI   R8,X3900
 88  29C8 3900
 89  29CA 1B1D        JH   CSNG              TEST FOR E PWR
 90  29CC 0603        DEC  R3
 91  29CE 1331        JEQ  CSNG1
 92  29D0 D236        MOVB *R6+,R8
 93  29D2 0288        CI   R8,X3000
 93  29D4 3000
 94  29D6 14F7        JHE  CSN11
 95  29D8 102B        JMP  CSNG16            IMPROPER TYPE NUMBER HERE
 96
 97
 98  29DA 0587 CSN05  INC  R7
 99  29DC 0603        DEC  R3
100  29DE 1329        JEQ  CSNG1
101  29E0 D236        MOVB *R6+,R8
102  29E2 0288 CSN06  CI   R8,X3000          LESS THAN ZERO
102  29E4 3000
103  29E6 1A03        JL   CSN07             YES
104  29E8 0288        CI   R8,X3900          LESS THAN NINE
104  29EA 3900
105  29EC 12F6        JLE  CSN05             YES
106  29EE 0288 CSN07  CI   R8,XDOT           END OF INTEGER WHOLE NUMBER PART
106  29F0 2E00
107  29F2 1609        JNE  CSNG
108            *
109            *    CONVERT A FLOATING POINT NUMBER
110            *    THE NUMBER HAS A DECIMAL POINT
111            *
112  29F4 0603 CSNF04 DEC  R3                GET NEXT CHAR IF ANY
113  29F6 131D        JEQ  CSNG1
114  29F8 D236        MOVB *R6+,R8
115  29FA 0288        CI   R8,X3000
115  29FC 3000
116  29FE 1A03        JL   CSNG              TOO SMALL FOR DIGIT
117  2A00 0288        CI   R8,X3900
117  2A02 3900
118  2A04 12F7        JLE  CSNF04            IN RANGE KEEP LOOKING
119            *
120            *    LOOK FOR EXPONENT OR END OF NUMBER
121            *
122  2A06 C086 CSNG   MOV  R6,R2             SAVE IN R2 ONE PAST THE END
123  2A08 0602        DEC  R2
124  2A0A 0288        CI   R8,>4500          IS CHARACTER AN 'E' ?
124  2A0C 4500
125  2A0E 1613        JNE  CSNHM6            NO EXPONENT DEFAULT = ZERO
126  2A10 06A0        BL   @CSINT            GET INTEGER VALUE (USES R0,R1,R3-R6,R8)
126  2A12 2AEA
127            *
128            *   NOW ON RETURN TEST IF MANTISSA IS 0
129            *
130  2A14 808C        C    R12,R2
131  2A16 1307        JEQ  CSNZER
132  2A18 C000        MOV  R0,R0             OVERFLOW?
133  2A1A 130E        JEQ  CSNH              NO
134  2A1C C804 CSNH11 MOV  R4,@EXP           MAKE THE EXP THE CORRECT SIGN
134  2A1E 836C
135  2A20 06A0        BL   @OVEXP1           NUMBER IS SIMPLY TOO LARGE
135  2A22 2354
136  2A24 103A        JMP  CSNH07            ALL DONE
137
138  2A26 04E0 CSNZER CLR  @FAC
138  2A28 834A
139  2A2A 04E0        CLR  @FAC+2
139  2A2C 834C
140  2A2E 1035        JMP  CSNH07
```

```
141
142
143
144  2A30 0606 CSNG16 DEC  R6                COMPENSATE FOR THE ERRORED CHARACTER
145  2A32 C086 CSNG1  MOV  R6,R2
146  2A34 1001        JMP  CSNH
147
148            *
149            *    PACK MANTISSA INTO FAC
150            *
151  2A36 0606 CSNHM6 DEC  R6                ADJUST R6
152  2A38 808C CSNH   C    R12,R2
153  2A3A 13F5        JEQ  CSNZER
154  2A3C 61AD CSNG2  S    @4(R13),R6        GET THE LENGTH USED
154  2A3E 0004
155  2A40 CB46        MOV  R6,@4(R13)        RETURN IT
155  2A42 0004
156  2A44 C244        MOV  R4,R9             SAVE SIGN FOR LATER USE
157  2A46 0229        AI   R9,128
157  2A48 0080
158  2A4A 1901        JNO  CSNH10            IF NO CARRY, CONTINUE
159  2A4C 10E7        JMP  CSNH11            OVERFLOW
160
161  2A4E 04C1 CSNH10 CLR  R1
162  2A50 A247        A    R7,R9
163  2A52 1901        JNO  CSNH09
164  2A54 10E3        JMP  CSNH11
165
166  2A56 C1C9 CSNH09 MOV  R9,R7
167  2A58 0819        SRA  R9,1              BASE 100
168  2A5A C809        MOV  R9,@EXP
168  2A5C 836C
169  2A5E 0B17        SRC  R7,1
170  2A60 0205        LI   R5,8              INITIALIZE LOOP
170  2A62 0008
171  2A64 0200        LI   R0,FAC+1
171  2A66 834B
172  2A68 C18C        MOV  R12,R6
173  2A6A 8086 CSNH01 C    R6,R2            END OF FRACTION
174  2A6C 1319        JEQ  CSNH03
175  2A6E D236        MOVB *R6+,R8           GET THE NEXT CHARACTER
176  2A70 0288        CI   R8,XDOT
176  2A72 2E00
177  2A74 13FA        JEQ  CSNH01
178  2A76 0988        SRL  R8,8
179  2A78 0228        AI   R8,-'0'           ASCII TO BINARY
179  2A7A FFD0
180  2A7C 0547        INV  R7
181  2A7E 1105        JLT  CSNH02
182  2A80 3A20        MPY  @LW10,R8          ZEROS OUT R8
182  2A82 24F0
183  2A84 06C9        SWPB R9
184  2A86 D049        MOVB R9,R1
185  2A88 10F0        JMP  CSNH01
186
187  2A8A 06C8 CSNH02 SWPB R8                ZEROS OUT LSB OF R8
188  2A8C B048        AB   R8,R1             ADD ONES AND TENS DIGIT
189  2A8E DC01        MOVB R1,*R0+
190  2A90 04C1        CLR  R1                IN CASE NUMBER ENDS
191  2A92 0605        DEC  R5                MORE ?
192  2A94 16EA        JNE  CSNH01            YES
193  2A96 06A0 CSNH04 BL   @ROUN1            JUST ROUND IT FROM HERE
193  2A98 2304
194  2A9A C2E0 CSNH07 MOV  @SAVCSN,R11
194  2A9C 20DA
195  2A9E 045B        RT
196            *
197  2AA0 DC01 CSNH03 MOVB R1,*R0+
198  2AA2 0605        DEC  R5
199  2AA4 13F8        JEQ  CSNH04
200  2AA6 04C1        CLR  R1
201  2AA8 DC01 CSNH05 MOVB R1,*R0+
202  2AAA 0605        DEC  R5
203  2AAC 16FD        JNE  CSNH05
```

```
204  2AAE 06A0 CSNH06 BL   @PACKUP           SET SIGN, ETC
204  2AB0 2332
205  2AB2 10F3        JMP  CSNH07
206                   PAGE
207             *==============================================================================
208             *
209             *    NAME:  CSINT
210             *
211             *    WHAT:  CONVERT STRING INTO INTEGER
212             *
213             *    REG:   DESCRIPTION       REG:   DESCRIPTION
214             *     0     SIGN               8     USED TO STORE DATA GOTTEN
215             *     1     CHARS USED         9
216             *     2                       10
217             *     3     TEXT LENGTH       11     RETURN
218             *     4     OUTPUT INTEGER    12
219             *     5     USED              13
220             *     6     POINTER TO TEXT   14
221             *     7                       15
222             *
223             *    ENTRY: BL   @CSINT
224             *    INPUT: R0 = 16
225             *           R1 = INTEGER result <---**les**
226             *           R2 = STRING ADDRESS <---**les**  1st byte = length
227             *
228             *   OUTPUT: R0 = ERROR CODE IF NON ZERO  (i.e., OVERFLOW)
229             *           R1 = 16-BIT SIGNED INTEGER
230             *           R2 = Number of digits used
231             *==============================================================================
232             *
233  2AB4 C0ED CSINT$ MOV  @4(R13),R3        GET string LENGTH pointer  <---for TF   **les**
233  2AB6 0004
234  2AB8 C183        MOV  R3,R6     **les** INITIALIZE TEXT POINTER (changed to caller's R1)
235  2ABA 0586        INC  R6                (point to 2nd byte)                      **les**
236  2ABC D0D3        MOVB *R3,R3            get string length                        **les**
237  2ABE 0983        SRL  R3,8              get into right byte                      **les**
238  2AC0 0283        CI   R3,0                                                       **les**
238  2AC2 0000
239  2AC4 130D        JEQ  CSI$00
240  2AC6 06A0        BL   @CSINT
240  2AC8 2AEA
241  2ACA CB44        MOV  R4,@2(R13)        RETURN THE NUMBER
241  2ACC 0002
242  2ACE CB41        MOV  R1,@4(R13)        RETURN THE # OF DIGITS USED
242  2AD0 0004
243  2AD2 C000        MOV  R0,R0             ERROR?
244  2AD4 1303        JEQ  CSGDRT            NO
245  2AD6 DB60        MOVB @ERROVF,@1(R13)   YES, SO MOVE ERROR OVERFLOW CODE
245  2AD8 235C
245  2ADA 0001
246  2ADC 02CF CSGDRT STST R15
247  2ADE 0380        RTWP                   ALL DONE
248             *
249  2AE0 04ED CSI$00 CLR  @2(R13)           IS ZERO
249  2AE2 0002
250  2AE4 04ED        CLR  @4(R13)           ZERO LENGTH USED
250  2AE6 0004
251  2AE8 10F9        JMP  CSGDRT            RETURN
252             *
253  2AEA 04C4 CSINT  CLR  R4                clear out MULTIPLIER
254  2AEC 04C1        CLR  R1                CHARACTER COUNTER FOR EXPONENT
255  2AEE 04C0        CLR  R0                SIGN IS POSITIVE
256  2AF0 04C8        CLR  R8                ZERO OUT R8 LSB
257  2AF2 D236        MOVB *R6+,R8           GET NEXT CHAR
258  2AF4 0288        CI   R8,XPLUS          IS IT POSITIVE?
258  2AF6 2B00
259  2AF8 130D        JEQ  CSI02             YES
260  2AFA 0288        CI   R8,XMINUS         IS IT MINUS?
260  2AFC 2D00
261  2AFE 160E        JNE  CSI08             NO, MUST BE A NUMBER
262  2B00 0700        SETO R0                SIGN IS NEGATIVE
263  2B02 1008        JMP  CSI02
264
265  2B04 3920 CSI01  MPY  @LW10,R4
```

```
265  2B06 24F0
266  2B08 C104        MOV  R4,R4             TEST FOR OVERFLOW
267  2B0A 161B        JNE  CSI05
268  2B0C 0988        SRL  R8,8
269  2B0E A148        A    R8,R5
270  2B10 C105        MOV  R5,R4             IS INTEGER > 32767
271  2B12 110F        JLT  CSI05A            YES - TOO BIG
272  2B14 0581 CSI02  INC  R1                COUNT THE CHARACTER
273  2B16 0603        DEC  R3                GET NEXT CHARACTER IF ANY EXISTS
274  2B18 1307        JEQ  CSI07             ALL TAKEN
275  2B1A D236        MOVB *R6+,R8
276  2B1C 0228 CSI08  AI   R8,>D000          ASCII TO BINARY
276  2B1E D000
277  2B20 0288        CI   R8,>0A00          COMPARE TO TEN
277  2B22 0A00
278  2B24 1AEF        JL   CSI01             OK GOOD NUMBER 0-9
279
280  2B26 0606        DEC  R6                THIS DIGIT NOT USED
281
282           *  TO BE HERE, NOT AN ASCII NUMBER, SO ALL DONE
283
284  2B28 C000 CSI07  MOV  R0,R0             IS THE RESULT NEGATIVE?
285  2B2A 1302        JEQ  CSINTR            NO
286  2B2C 0504        NEG  R4                YES
287  2B2E 04C0        CLR  R0                NO ERROR
288  2B30 045B CSINTR RT                     RETURN TO CALLER
289
290           CSI05A
291           * IS > 32767 SO TEST IF 8000 NEGATIVE
292  2B32 C000        MOV  R0,R0             SO SIGN NEGATIVE?
293  2B34 1306        JEQ  CSI05             NO, SO OVERFLOW ERROR
294  2B36 0284        CI   R4,>8000          IS IT -32768?
294  2B38 8000
295  2B3A 13EC        JEQ  CSI02             YES, SO CONTINUE
296  2B3C 0204        LI   R4,>8000
296  2B3E 8000
297  2B40 045B        RT                     R0 IS >FFFF ALREADY
298
299  2B42 0204 CSI05  LI   R4,>7FFF          GET THE ERRORR ADDRESS IN A REG
299  2B44 7FFF
300  2B46 0700        SETO R0                OVERFLOW ERROR
301  2B48 045B        RT
               *
               *       COPY    'C:\Users\Lee\Downloads\TI-99-4A\TurboForth\MDOS\L10\CFI.a99'
               *
  1            *--------------------------------------------------------------------------
  2            *
  3            *      WRITTEN: 12/27/84
  4            *
  5            *      FILE:    WDS1.136.CFI
  6            *
  7            *      NAME:    CONVERT FLOATING TO INTEGER
  8            *               (originally) DOES NOT DESTROY INCOMING NUMBER  **les**
  9            *               (now) DESTROYS INCOMING NUMBER                 **les**
 10            *               USES R0-R9
 11            *
 12            *     INPUT WORKSPACE (this function)
 13            *            R0 = integer result
 14            *            R2 = pointer to FLOATING POINT NUMBER
 15            *
 16            *     OUTPUT WORKSPACE (caller's)
 17            *            R0 = ERROR CODE
 18            *            R1 = address of INTEGER    **les**
 19            *
 20            *
 21            *       CCCCC    FFFFFFF     III
 22            *      C     C   F            I
 23            *      C         F            I
 24            *      C         FFFFF        I
 25            *      C         F            I
 26            *      C     C   F            I
 27            *       CCCCC    F          III
 28            *
 29            *
```

```
30           *
31           *--------------------------------------------------------------------------
32           *
33  2B4A 06A0 CFI$$  BL   @CFI
33  2B4C 2B5C
34  2B4E C06D        MOV  @2(R13),R1        get address to store result   **les**
34  2B50 0002
35  2B52 C440        MOV  R0,*R1            store integer back to caller  **les**
36  2B54 0987        SRL  R7,8
37  2B56 C747        MOV  R7,*R13           move error code back to caller
38  2B58 02CF        STST R15               return error status as a flag
39  2B5A 0380        RTWP
40  2B5C 0202 CFI    LI   R2,FAC            GET POINTER TO NUMBER
40  2B5E 834A
41  2B60 04C7        CLR  R7                ASSUME NO ERROR
42  2B62 C152        MOV  *R2,R5            IS NUMBER ZERO
43  2B64 C005        MOV  R5,R0
44  2B66 1342        JEQ  CFI11             ALL DONE
45  2B68 04C0        CLR  R0                ZERO RESULT IN CASE FAC=0
46  2B6A C202        MOV  R2,R8
47  2B6C 0228        AI   R8,8              STOPPING CRITERION
47  2B6E 0008
48  2B70 04C3        CLR  R3                CLEAR FRACTION
49  2B72 0752        ABS  *R2               MAKE SURE FIRST DIGIT IS POSITIVE
50  2B74 04C4        CLR  R4                CLEAR OUT UNUSED PART OF REGISTER
51  2B76 D132        MOVB *R2+,R4           GET EXPONENT
52  2B78 02A6        STWP R6                USE R6 FOR INDEXING INTO WORKSPACE
53  2B7A 0209        LI   R9,100            CONSTANT
53  2B7C 0064
54  2B7E 0284        CI   R4,>3F00          IS NUMBER LESS THAN ONE
54  2B80 3F00
55  2B82 1134        JLT  CFI11             YES NUMBER < .01      - RESULT = 0
56  2B84 1316        JEQ  CFI03             NUMBER > .01 AND < 1  - RESULT = 1
57  2B86 0284        CI   R4,>4100          IS NUMBER LESS THAN 100,000
57  2B88 4100
58  2B8A 1110        JLT  CFI02             IT IS BETWEEN 1 AND 100
59  2B8C 1307        JEQ  CFI01             IT IS BETWEEN 100 AND 10,000
60  2B8E 0284        CI   R4,>4200          IS NUMBER TOO BIG TO CONVERT
60  2B90 4200
61  2B92 1B21        JH   CFI08             YES IT IS
62  2B94 D9B2        MOVB *R2+,@1(R6)       GET DIGIT
62  2B96 0001
63  2B98 3809        MPY  R9,R0             RADIX TO BINARY
64  2B9A C001        MOV  R1,R0             GET RESULT FOR NEXT DIGIT
65  2B9C D9B2 CFI01  MOVB *R2+,@7(R6)       GET NEXT DIGIT
65  2B9E 0007
66  2BA0 A003        A    R3,R0             ADD TO PREVIOUS RESULT
67  2BA2 3809        MPY  R9,R0             RADIX TO BINARY
68  2BA4 C000        MOV  R0,R0             TEST FOR OVERFLOW
69  2BA6 1617        JNE  CFI08             YES ERROR
70  2BA8 C001        MOV  R1,R0             GET RESULT FOR LAST DIGIT
71  2BAA 1115        JLT  CFI08             OVERFLOW
72  2BAC D0F2 CFI02  MOVB *R2+,R3           GOT LAST DIGIT TO LEFT OF DECIMAL
73  2BAE 0983        SRL  R3,8
74  2BB0 A003        A    R3,R0             ADD TO RESULT
75  2BB2 9832 CFI03  CB   *R2+,@CBD50       IS ROUNDING NECESSARY
75  2BB4 20D4
76  2BB6 1109        JLT  CFI06             NO PUT ON PROPER SIGN
77  2BB8 1507        JGT  CFI05             YES ADD A 1 TO IT
78  2BBA C145        MOV  R5,R5
79  2BBC 1505        JGT  CFI05             NUMBER IS POSITIVE ROUND UP
80  2BBE D0F2 CFI04  MOVB *R2+,R3
81  2BC0 1603        JNE  CFI05             NONZERO ROUND UP
82  2BC2 8202        C    R2,R8             LOOK AT REST OF DIGITS
83  2BC4 1AFC        JL   CFI04             NO LOOK AT NEXT ONE
84  2BC6 1001        JMP  CFI06             ROUND DOWN
85           *
86           *
87           *
88  2BC8 0580 CFI05  INC  R0                ROUND UP
89  2BCA 0280 CFI06  CI   R0,SGNBIT         IS RESULT 32768
89  2BCC 8000
90  2BCE 1A0B        JL   CFI09
91  2BD0 1B02        JH   CFI08
```

```
 92  2BD2 C145        MOV  R5,R5             IS NUMBER NEGATIVE 32768?
 93  2BD4 110B        JLT  CFI11             YES
 94  2BD6 0207 CFI08  LI   R7,ERROV*256      ERROR CODE
 94  2BD8 0300
 95  2BDA 0200        LI   R0,>7FFF          NOW JUST FINISH UP
 95  2BDC 7FFF
 96  2BDE 0545        INV  R5                IS NUMBER NEGATIVE
 97  2BE0 1105        JLT  CFI11
 98  2BE2 0580        INC  R0
 99  2BE4 1003        JMP  CFI11
100              *
101              *
102              *
103  2BE6 0545 CFI09  INV  R5                IS NUMBER NEGATIVE
104  2BE8 1101        JLT  CFI11             NO RETURN POSITIVE
105  2BEA 0500 CFI10  NEG  R0
106  2BEC C800 CFI11  MOV  R0,@FAC           RETURN NUMBER IN CALLER'S R1
106  2BEE 834A
107  2BF0 C807 CFI12  MOV  R7,@FAC+10        RETURN ERROR CODE IN CALLER'S R0
107  2BF2 8354
108  2BF4 045B        RT
                 *
                 *       COPY    'C:\Users\Lee\Downloads\TI-99-4A\TurboForth\MDOS\L10\CIF.a99'
                 *
  1              *--------------------------------------------------------------------------
  2              *
  3              *   FILE:   WDS1.NEW.CIF
  4              *
  5              *   NAME:   CONVERT INTEGER TO FLOATING
  6              *           POINT NUMBER
  7              *
  8              *   VERSION: 5/25/87           BASE LINE
  9              *
 10              *      INPUT WORKSPACE   (this function)
 11              *            R0 = INTEGER (from caller's R1)
 12              *            R4 = POINTER TO FLOATING-POINT RESULT
 13              *
 14              *      OUTPUT WORKSPACE  (caller's)
 15              *            R1 = POINTER TO FLOATING-POINT NUMBER
 16              *--------------------------------------------------------------------------
 17              *
 18  2BF6 C12D CIF    MOV  @2(R13),R4     get address of number (INT in; FP out)  **les**
 18  2BF8 0002
 19  2BFA C184        MOV  R4,R6          for clearing (see below)                **les**
 20  2BFC C014        MOV  *R4,R0         get the integer before we clobber it    **les**
 21              *
 22  2BFE 0207        LI   R7,100            FOR SPEED
 22  2C00 0064
 23  2C02 04F6        CLR  *R6+              AND CLEAR IT OUT
 24  2C04 04F6        CLR  *R6+
 25  2C06 04F6        CLR  *R6+
 26  2C08 04D6        CLR  *R6
 27  2C0A 0280        CI   R0,0                                          **les**
 27  2C0C 0000
 28  2C0E 131F        JEQ  CIF03             IF ZERO THEN DONE
 29              *
 30  2C10 C140        MOV  R0,R5             save sign                   **les**
 31  2C12 0740        ABS  R0                ENSURE ITS POSITIVE
 32  2C14 0203        LI   R3,>40            GET THE EXPONIENT
 32  2C16 0040
 33  2C18 81C0        C    R0,R7
 34  2C1A 1A11        JL   CIF02
 35  2C1C 0280        CI   R0,10000
 35  2C1E 2710
 36  2C20 1A07        JL   CIF01
 37  2C22 0583        INC  R3
 38  2C24 C040        MOV  R0,R1
 39  2C26 04C0        CLR  R0
 40  2C28 3C07        DIV  R7,R0
 41  2C2A 06C1        SWPB R1
 42  2C2C D901        MOVB R1,@3(R4)
 42  2C2E 0003
 43  2C30 0583 CIF01  INC  R3
 44  2C32 C040        MOV  R0,R1
```

```
45  2C34 04C0        CLR  R0
46  2C36 3C07        DIV  R7,R0
47  2C38 06C1        SWPB R1
48  2C3A D901        MOVB R1,@2(R4)
48  2C3C 0002
49  2C3E 06C0 CIF02  SWPB R0
50  2C40 D900        MOVB R0,@1(R4)
50  2C42 0001
51  2C44 06C3        SWPB R3
52  2C46 D503        MOVB R3,*R4
53  2C48 0545        INV  R5
54  2C4A 1101        JLT  CIF03
55  2C4C 0514        NEG  *R4
56  2C4E 0380 CIF03  RTWP
              *
              *       COPY    'C:\Users\Lee\Downloads\TI-99-4A\TurboForth\MDOS\L10\TRINSIC1.a99'
              *
 1            *---------------------------------------------------------------------------
 2            *
 3            *      WRITTEN: 07/13/1985
 4            *
 5            *      FILE:    WDS1.133.TRINSIC1
 6            *
 7            *      NAME:    TRINSIC PART ONE FUNCTIONS
 8            *
 9            *
10            *---------------------------------------------------------------------------
11            *===========================================================================
12            *
13            *    NAME:  PWR$$   INVOLUTION
14            *
15            *    WHAT:  POWER ROUTINE
16            *
17            *    INPUT
18            *           R0 = 5
19            *           R1 = EXPONENT (already in FAC)
20            *           R2 = BASE (already in ARG)
21            *    OUTPUT
22            *           R0 = ERROR
23            *           R1 = BASE^EXPONENT (will be copied from FAC upon return)
24            *
25            *===========================================================================
26  2C50 C020 PWR$$  MOV  @FAC,R0      IS EXPONENT 0?
26  2C52 834A
27  2C54 136A        JEQ  PWRG01       YES THEN RESULT=1
28  2C56 C020        MOV  @ARG,R0      IS BASE 0?
28  2C58 835C
29  2C5A 1363        JEQ  PWRG02       THEN RETURN 0 OR WARNING
30  2C5C 0201        LI   R1,ARG       PUT ARG TO STK #3 (BASE)
30  2C5E 835C
31  2C60 0202        LI   R2,STK3
31  2C62 20FC
32  2C64 06A0        BL   @R1$2        move 4 words from *R1 to *R2               **les**
32  2C66 2092
33  2C68 0201        LI   R1,FAC       PUT FAC TO STACK #4 (EXP)
33  2C6A 834A
34  2C6C 0202        LI   R2,STK4
34  2C6E 2104
35  2C70 06A0        BL   @R1$2        move 4 words from *R1 to *R2               **les**
35  2C72 2092
36  2C74 06A0        BL   @GRINT       NOW CHECK TO SEE IF EXPONENT IS INTEGER
36  2C76 3422
37  2C78 D820        MOVB @CW08,@SIGN  ASSUME SIGN IS POSITIVE
37  2C7A 20C4
37  2C7C 836E
38            *
39            * NOW COMPARE THE EXP TO THE INT(EXP)
40            *
41  2C7E 0201        LI   R1,FAC
41  2C80 834A
42  2C82 0202        LI   R2,STK4
42  2C84 2104
43  2C86 8CB1        C    *R1+,*R2+
44  2C88 1670        JNE  PWR$$3
```

```
45  2C8A 8CB1        C    *R1+,*R2+
46  2C8C 166E        JNE  PWR$$3
47  2C8E 8CB1        C    *R1+,*R2+
48  2C90 166C        JNE  PWR$$3
49  2C92 8491        C    *R1,*R2
50  2C94 166A        JNE  PWR$$3
51                *
52                *    COMPUTE INTEGER POWER B^E
53                *       WE KNOW THAT E IS AN INTEGER AN IN FAC
54                *
55                *      BL   @PUSH
56  2C96 D820        MOVB @CW08,@FAC+10 ASSUME NO ERROR IN CFI CONVERSION
56  2C98 20C4
56  2C9A 8354
57  2C9C 06A0        BL   @CFI          NOW EXP IS AN INTEGER IN FAC
57  2C9E 2B5C
58  2CA0 C320        MOV  @FAC,R12      MOVE IT TO R12 AND MAKE IT POSITIVE
58  2CA2 834A
59  2CA4 D80C        MOVB R12,@WSM6     SAVE SIGN OF EXPON FOR LATER USE
59  2CA6 20E0
60  2CA8 074C        ABS  R12
61  2CAA D020        MOVB @FAC+10,R0
61  2CAC 8354
62  2CAE 1664        JNE  PWR$$1
63                *
64                * NOW MOVE ARG (BASE) TO FAC
65                *
66  2CB0 0201        LI   R1,STK3
66  2CB2 20FC
67  2CB4 0202        LI   R2,FAC
67  2CB6 834A
68  2CB8 06A0        BL   @R1$2         move 4 words from *R1 to *R2               **les**
68  2CBA 2092
69  2CBC 060C        DEC  R12
70  2CBE 133D        JEQ  PWRJ40
71  2CC0 091C PWRJ30 SRL  R12,1
72  2CC2 1708        JNC  PWRJ10
73  2CC4 0201        LI   R1,STK3
73  2CC6 20FC
74  2CC8 0202        LI   R2,ARG
74  2CCA 835C
75  2CCC 06A0        BL   @R1$2         move 4 words from *R1 to *R2               **les**
75  2CCE 2092
76  2CD0 06A0        BL   @FMULT
76  2CD2 2244
77  2CD4 C30C PWRJ10 MOV  R12,R12
78  2CD6 1331        JEQ  PWRJ40
79                * PUT FAC IN STK4
80  2CD8 0201        LI   R1,FAC
80  2CDA 834A
81  2CDC 0202        LI   R2,STK4
81  2CDE 2104
82  2CE0 06A0        BL   @R1$2         move 4 words from *R1 to *R2               **les**
82  2CE2 2092
83  2CE4 0203        LI   R3,STK3
83  2CE6 20FC
84  2CE8 0201        LI   R1,FAC            MOVE FAC TO ARG
84  2CEA 834A
85  2CEC 0202        LI   R2,ARG
85  2CEE 835C
86  2CF0 CC53        MOV  *R3,*R1+
87  2CF2 CCB3        MOV  *R3+,*R2+
88  2CF4 CC53        MOV  *R3,*R1+
89  2CF6 CCB3        MOV  *R3+,*R2+
90  2CF8 CC53        MOV  *R3,*R1+
91  2CFA CCB3        MOV  *R3+,*R2+
92  2CFC C453        MOV  *R3,*R1
93  2CFE C493        MOV  *R3,*R2
94  2D00 06A0        BL   @FMULT
94  2D02 2244
95  2D04 0201        LI   R1,STK4
95  2D06 2104
96  2D08 0202        LI   R2,STK3
96  2D0A 20FC
```

```
 97  2D0C 0203        LI   R3,FAC
 97  2D0E 834A
 98  2D10 CC93        MOV  *R3,*R2+
 99  2D12 CCF1        MOV  *R1+,*R3+
100  2D14 CC93        MOV  *R3,*R2+
101  2D16 CCF1        MOV  *R1+,*R3+
102  2D18 CC93        MOV  *R3,*R2+
103  2D1A CCF1        MOV  *R1+,*R3+
104  2D1C C493        MOV  *R3,*R2
105  2D1E C4D1        MOV  *R1,*R3
106  2D20 10CF        JMP  PWRJ30
107            *
108            *
109            *
110  2D22 D020 PWRG02 MOVB @FAC,R0
110  2D24 834A
111  2D26 1155        JLT  PWRG05
112  2D28 101A        JMP  PWRJ45
113            *
114            *    NEED A FLOATING POINT 1 IN FAC
115            *
116  2D2A 0201 PWRG01 LI   R1,FAC            GET A FLOATING ONE
116  2D2C 834A
117  2D2E CC60        MOV  @FLTONE,*R1+
117  2D30 3662
118  2D32 04F1        CLR  *R1+
119  2D34 04F1        CLR  *R1+
120  2D36 04D1        CLR  *R1
121  2D38 1003        JMP  PWRRTN
122
123  2D3A D020 PWRJ40 MOVB @WSM6,R0     TEST EXPONENT SIGN NOW
123  2D3C 20E0
124  2D3E 1102        JLT  PWRJ41
125  2D40 0460 PWRRTN B    @TRINRT
125  2D42 2072
126            *
127            *
128            *
129  2D44 D020 PWRJ41 MOVB @FAC+10,R0
129  2D46 8354
130  2D48 160A        JNE  PWRJ45
131
132  2D4A 0202        LI   R2,ARG            PUT A FLOATING POINT ONE IN ARG
132  2D4C 835C
133  2D4E CCA0        MOV  @FLTONE,*R2+
133  2D50 3662
134  2D52 04F2        CLR  *R2+
135  2D54 04F2        CLR  *R2+
136  2D56 04D2        CLR  *R2
137  2D58 06A0        BL   @FDIV
137  2D5A 2380
138  2D5C 10F1        JMP  PWRRTN
139            *
140            *
141            *
142  2D5E 04E0 PWRJ45 CLR  @FAC
142  2D60 834A
143  2D62 D820        MOVB @FAC,@FAC+10
143  2D64 834A
143  2D66 8354
144  2D68 10EB        JMP  PWRRTN
145            *
146            *   TO BE HERE, EXP IS NOT AN INTEGER, CHECK IF Y>0 OF Y^X
147            *
148  2D6A D060 PWR$$3 MOVB @STK3,R1
148  2D6C 20FC
149  2D6E 1513        JGT  PWR$$2
150  2D70 D820        MOVB @ERRNIP,@FAC+10
150  2D72 20CC
150  2D74 8354
151  2D76 10E4        JMP  PWRRTN
152            *
153            *    INTEGER EXPONENT OUT OF INTEGER RANGE
154            *
```

```
155  2D78 04C1 PWR$$1 CLR  R1
156  2D7A D060        MOVB @STK4,R1
156  2D7C 2104
157  2D7E 150B        JGT  PWR$$2
158            *
159            *    NEGATIVE BASE SO SEE IF EXPONENT IS EVEN OR ODD
160            *    TO SET THE SIGN OF THE RESULT
161            *
162  2D80 0741 PWR$$4 ABS  R1
163  2D82 0281        CI   R1,>4600
163  2D84 4600
164  2D86 1507        JGT  PWR$$2
165  2D88 06C1        SWPB R1
166  2D8A 0221        AI   R1,FAC->003F
166  2D8C 830B
167  2D8E D051        MOVB *R1,R1
168  2D90 0A71        SLA  R1,7
169  2D92 D801        MOVB R1,@SIGN
169  2D94 836E
170  2D96 D820 PWR$$2 MOVB @SIGN,@WSM6
170  2D98 836E
170  2D9A 20E0
171  2D9C 0201        LI   R1,STK3
171  2D9E 20FC
172  2DA0 0202        LI   R2,FAC
172  2DA2 834A
173  2DA4 06A0        BL   @R1$2         move 4 words from *R1 to *R2              **les**
173  2DA6 2092
174  2DA8 0760        ABS  @FAC
174  2DAA 834A
175  2DAC 06A0        BL   @LOG$$$
175  2DAE 2F7C
176  2DB0 0201        LI   R1,STK4
176  2DB2 2104
177  2DB4 0202        LI   R2,ARG
177  2DB6 835C
178  2DB8 06A0        BL   @R1$2         move 4 words from *R1 to *R2              **les**
178  2DBA 2092
179  2DBC 06A0        BL   @FMULT
179  2DBE 2244
180  2DC0 06A0        BL   @EXP$$$
180  2DC2 2E12
181  2DC4 D060        MOVB @WSM6,R1
181  2DC6 20E0
182  2DC8 1101        JLT  PWR$$5
183  2DCA 10BA        JMP  PWRRTN
184
185  2DCC 0520 PWR$$5 NEG  @FAC
185  2DCE 834A
186  2DD0 10B7        JMP  PWRRTN
187
188  2DD2 06A0 PWRG05 BL   @OVEXP
188  2DD4 2354
189  2DD6 10B4        JMP  PWRRTN
190                   PAGE
191  2DD8 0202 EXPONE LI   R2,FAC
191  2DDA 834A
192  2DDC CCA0        MOV  @FLTONE,*R2+
192  2DDE 3662
193  2DE0 04F2        CLR  *R2+
194  2DE2 04F2        CLR  *R2+
195  2DE4 04D2        CLR  *R2
196  2DE6 100E        JMP  EXPRTN
197
198  2DE8 0207 EXP05  LI   R7,NXC127
198  2DEA 34CC
199  2DEC 06A0        BL   @FCOMP7
199  2DEE 2120
200  2DF0 112D        JLT  EXP03
201  2DF2 132C        JEQ  EXP03
202            *
203            *
204            *
205  2DF4 C820 EXP01  MOV  @FAC,@EXP
```

```
205  2DF6 834A
205  2DF8 836C
206  2DFA D820        MOVB @CW08,@SIGN       RESULT IS POSITIVE
206  2DFC 20C4
206  2DFE 836E
207  2E00 06A0        BL   @OVEXP
207  2E02 2354
208  2E04 C2E0 EXPRTN MOV  @EXTRTN,R11       USE EXTENDED RETURN
208  2E06 20D6
209  2E08 045B        RT
210            *
211            *
212            *   FAC=EXP(FAC)
213            *
214            *
215  2E0A 06A0 EXP$$  BL   @EXP$$$
215  2E0C 2E12
216  2E0E 0460        B    @PWRRTN
216  2E10 2D40
217
218  2E12 C80B EXP$$$ MOV  R11,@EXTRTN
218  2E14 20D6
219  2E16 C020        MOV  @FAC,R0      IS IT 0 E^0=1
219  2E18 834A
220  2E1A 13DE        JEQ  EXPONE       YES
221  2E1C 0201        LI   R1,LOG10E
221  2E1E 34E4
222  2E20 0202        LI   R2,ARG            SOURCE
222  2E22 835C
223  2E24 06A0        BL   @R1$2         move 4 words from *R1 to *R2              **les**
223  2E26 2092
224  2E28 06A0        BL   @FMULT
224  2E2A 2244
225  2E2C 0201        LI   R1,FAC
225  2E2E 834A
226  2E30 0202        LI   R2,STK1
226  2E32 20EC
227  2E34 06A0        BL   @R1$2         move 4 words from *R1 to *R2              **les**
227  2E36 2092
228  2E38 06A0        BL   @GRINT
228  2E3A 3422
229  2E3C C060        MOV  @FAC,R1         IS IT POSITIVE OR NEGATIVE?
229  2E3E 834A
230  2E40 11D3        JLT  EXP05
231  2E42 0207        LI   R7,EXC127
231  2E44 34C4
232  2E46 06A0        BL   @FCOMP7
232  2E48 2120
233  2E4A 11D4        JLT  EXP01
234  2E4C 0201 EXP03  LI   R1,FAC       GRINT(FAC) TO STK2
234  2E4E 834A
235  2E50 0202        LI   R2,STK2
235  2E52 20F4
236  2E54 06A0        BL   @R1$2         move 4 words from *R1 to *R2              **les**
236  2E56 2092
237  2E58 06A0        BL   @CFI
237  2E5A 2B5C
238  2E5C C320        MOV  @FAC,R12
238  2E5E 834A
239  2E60 0A1C        SLA  R12,1
240  2E62 0201        LI   R1,STK2
240  2E64 20F4
241  2E66 0202        LI   R2,FAC
241  2E68 834A
242  2E6A 06A0        BL   @R1$2         move 4 words from *R1 to *R2              **les**
242  2E6C 2092
243  2E6E 0201        LI   R1,STK1
243  2E70 20EC
244  2E72 0202        LI   R2,ARG
244  2E74 835C
245  2E76 06A0        BL   @R1$2         move 4 words from *R1 to *R2              **les**
245  2E78 2092
246  2E7A 06A0        BL   @FSUB
246  2E7C 2140
```

```
247  2E7E 0207        LI   R7,FHALF
247  2E80 34D4
248  2E82 06A0        BL   @FCOMP7
248  2E84 2120
249  2E86 150B        JGT  EXP04
250  2E88 0202        LI   R2,ARG
250  2E8A 835C
251  2E8C C4A0        MOV  @FHALF,*R2
251  2E8E 34D4
252  2E90 0532        NEG  *R2+
253  2E92 04F2        CLR  *R2+
254  2E94 04F2        CLR  *R2+
255  2E96 04D2        CLR  *R2
256  2E98 06A0        BL   @FADD
256  2E9A 2144
257  2E9C 058C        INC  R12
258  2E9E 0201 EXP04  LI   R1,FAC
258  2EA0 834A
259  2EA2 0202        LI   R2,STK1
259  2EA4 20EC
260  2EA6 06A0        BL   @R1$2        move 4 words from *R1 to *R2              **les**
260  2EA8 2092
261  2EAA 06A0        BL   @POLYX
261  2EAC 30B8
262  2EAE 3548        DATA EXPP
263  2EB0 0201        LI   R1,STK1
263  2EB2 20EC
264  2EB4 0202        LI   R2,ARG
264  2EB6 835C
265  2EB8 06A0        BL   @R1$2        move 4 words from *R1 to *R2              **les**
265  2EBA 2092
266  2EBC 06A0        BL   @FMULT
266  2EBE 2244
267  2EC0 0201        LI   R1,FAC       NOW S*P(S^2) TO STK2
267  2EC2 834A
268  2EC4 0202        LI   R2,STK2      AND S (in STK1) TO FAC
268  2EC6 20F4
269  2EC8 0203        LI   R3,STK1
269  2ECA 20EC
270  2ECC CC91        MOV  *R1,*R2+
271  2ECE CC73        MOV  *R3+,*R1+
272  2ED0 CC91        MOV  *R1,*R2+
273  2ED2 CC73        MOV  *R3+,*R1+
274  2ED4 CC91        MOV  *R1,*R2+
275  2ED6 CC73        MOV  *R3+,*R1+
276  2ED8 C491        MOV  *R1,*R2
277  2EDA C453        MOV  *R3,*R1
278  2EDC 06A0        BL   @POLYX       NOW Q(S^2) IN FAC
278  2EDE 30B8
279  2EE0 3562        DATA EXPQ
280  2EE2 0201        LI   R1,STK2      NOW S*P(S^2) IN ARG
280  2EE4 20F4
281  2EE6 0202        LI   R2,ARG
281  2EE8 835C
282  2EEA 06A0        BL   @R1$2        move 4 words from *R1 to *R2              **les**
282  2EEC 2092
283  2EEE 0201        LI   R1,FAC       PUT Q(S^2) IN STK1
283  2EF0 834A
284  2EF2 0202        LI   R2,STK1
284  2EF4 20EC
285  2EF6 06A0        BL   @R1$2        move 4 words from *R1 to *R2              **les**
285  2EF8 2092
286  2EFA 06A0        BL   @FADD        NOW S*P(S^2)+Q(S^2) IN FAC
286  2EFC 2144
287            *
288            *  NOW S*P(S^2) TO FAC AND Q(S^2) TO ARG
289            *
290  2EFE 0201        LI   R1,STK1      STK1 TO ARG
290  2F00 20EC
291  2F02 0202        LI   R2,ARG       FAC TO STK1
291  2F04 835C
292  2F06 0203        LI   R3,STK2      STK2 TO FAC
292  2F08 20F4
293  2F0A 0204        LI   R4,FAC
```

```
293  2F0C 834A
294  2F0E CC91        MOV  *R1,*R2+     STK1 TO ARG
295  2F10 CC54        MOV  *R4,*R1+     FAC TO STK1
296  2F12 CD33        MOV  *R3+,*R4+    STK2 TO FAC
297  2F14 CC91        MOV  *R1,*R2+     STK1 TO ARG
298  2F16 CC54        MOV  *R4,*R1+     FAC TO STK1
299  2F18 CD33        MOV  *R3+,*R4+    STK2 TO FAC
300  2F1A CC91        MOV  *R1,*R2+     STK1 TO ARG
301  2F1C CC54        MOV  *R4,*R1+     FAC TO STK1
302  2F1E CD33        MOV  *R3+,*R4+    STK2 TO FAC
303  2F20 C491        MOV  *R1,*R2      STK1 TO ARG
304  2F22 C454        MOV  *R4,*R1      FAC TO STK1
305  2F24 C513        MOV  *R3,*R4      STK2 TO FAC
306  2F26 06A0        BL   @FSUB
306  2F28 2140
307  2F2A 0201        LI   R1,STK1
307  2F2C 20EC
308  2F2E 0202        LI   R2,ARG
308  2F30 835C
309  2F32 06A0        BL   @R1$2           move 4 words from *R1 to *R2              **les**
309  2F34 2092
310  2F36 06A0        BL   @FDIV
310  2F38 2380
311  2F3A 081C EXPSQT SRA  R12,1
312  2F3C 1708        JNC  EXPSQ5
313  2F3E 0201        LI   R1,SQRTEN
313  2F40 34DC
314  2F42 0202        LI   R2,ARG
314  2F44 835C
315  2F46 06A0        BL   @R1$2           move 4 words from *R1 to *R2              **les**
315  2F48 2092
316  2F4A 06A0        BL   @FMULT
316  2F4C 2244
317  2F4E 0202 EXPSQ5 LI   R2,ARG           GET A FLOATING ONE
317  2F50 835C
318  2F52 CCA0        MOV  @FLTONE,*R2+
318  2F54 3662
319  2F56 04F2        CLR  *R2+
320  2F58 04F2        CLR  *R2+
321  2F5A 04D2        CLR  *R2
322  2F5C 081C        SRA  R12,1
323  2F5E 1703        JNC  EXPSQ8
324  2F60 D820        MOVB @CBHA,@ARG+1
324  2F62 20D2
324  2F64 835D
325  2F66 06CC EXPSQ8 SWPB R12
326  2F68 B80C        AB   R12,@ARG
326  2F6A 835C
327  2F6C 06A0        BL   @FMULT
327  2F6E 2244
328  2F70 0460        B    @EXPRTN
328  2F72 2E04
329                   PAGE
330            *
331            *
332            *
333  2F74 06A0 LOG$$  BL   @LOG$$$
333  2F76 2F7C
334  2F78 0460        B    @PWRRTN
334  2F7A 2D40
335
336  2F7C C80B LOG$$$ MOV  R11,@EXTRTN
336  2F7E 20D6
337  2F80 C020        MOV  @FAC,R0
337  2F82 834A
338  2F84 1506        JGT  LOG$$3
339  2F86 D820        MOVB @ERRLOG,@FAC+10
339  2F88 20CE
339  2F8A 8354
340  2F8C C2E0 LOGRTN MOV  @EXTRTN,R11
340  2F8E 20D6
341  2F90 045B        RT
342
343  2F92 06A0 LOG$$3 BL   @CNSTIN
```

```
343  2F94 349E
344  2F96 160F        JNE  LOG$$5
345  2F98 0202        LI   R2,ARG
345  2F9A 835C
346  2F9C CCA0        MOV  @FLTONE,*R2+
346  2F9E 3662
347  2FA0 04F2        CLR  *R2+
348  2FA2 04F2        CLR  *R2+
349  2FA4 04D2        CLR  *R2
350  2FA6 D820        MOVB @CBHA,@ARG+1
350  2FA8 20D2
350  2FAA 835D
351  2FAC 06A0        BL   @FMULT
351  2FAE 2244
352  2FB0 06A0        BL   @CNSTIN
352  2FB2 349E
353  2FB4 1002        JMP  LOG$5A
354  2FB6 05A0 LOG$$5 INC  @EXP
354  2FB8 836C
355               *
356               *
357               *
358  2FBA D820 LOG$5A MOVB @CBH3F,@FAC
358  2FBC 350C
358  2FBE 834A
359  2FC0 C320        MOV  @EXP,R12
359  2FC2 836C
360  2FC4 0201        LI   R1,SQRTEN
360  2FC6 34DC
361  2FC8 0202        LI   R2,ARG
361  2FCA 835C
362  2FCC 06A0        BL   @R1$2           move 4 words from *R1 to *R2              **les**
362  2FCE 2092
363  2FD0 06A0        BL   @FMULT
363  2FD2 2244
364  2FD4 06A0        BL   @FORMA
364  2FD6 3124
365  2FD8 0201        LI   R1,FAC
365  2FDA 834A
366  2FDC 0202        LI   R2,STK1
366  2FDE 20EC
367  2FE0 06A0        BL   @R1$2           move 4 words from *R1 to *R2              **les**
367  2FE2 2092
368  2FE4 06A0        BL   @POLYX
368  2FE6 30B8
369  2FE8 3584        DATA LOGP
370  2FEA 0201        LI   R1,STK1
370  2FEC 20EC
371  2FEE 0202        LI   R2,ARG
371  2FF0 835C
372  2FF2 06A0        BL   @R1$2           move 4 words from *R1 to *R2              **les**
372  2FF4 2092
373  2FF6 06A0        BL   @FMULT
373  2FF8 2244
374  2FFA 0201        LI   R1,STK1
374  2FFC 20EC
375  2FFE 0202        LI   R2,FAC
375  3000 834A
376  3002 C0D1        MOV  *R1,R3
377  3004 CC52        MOV  *R2,*R1+
378  3006 CC83        MOV  R3,*R2+
379  3008 C0D1        MOV  *R1,R3
380  300A CC52        MOV  *R2,*R1+
381  300C CC83        MOV  R3,*R2+
382  300E C0D1        MOV  *R1,R3
383  3010 CC52        MOV  *R2,*R1+
384  3012 CC83        MOV  R3,*R2+
385  3014 C0D1        MOV  *R1,R3
386  3016 C452        MOV  *R2,*R1
387  3018 C483        MOV  R3,*R2
388  301A 06A0        BL   @POLYX
388  301C 30B8
389  301E 35AE        DATA LOGQ
390  3020 0201        LI   R1,STK1
```

```
390  3022 20EC
391  3024 0202        LI   R2,ARG
391  3026 835C
392  3028 06A0        BL   @R1$2           move 4 words from *R1 to *R2              **les**
392  302A 2092
393  302C 06A0        BL   @FDIV
393  302E 2380
394  3030 0201        LI   R1,FAC
394  3032 834A
395  3034 0202        LI   R2,STK1
395  3036 20EC
396  3038 06A0        BL   @R1$2           move 4 words from *R1 to *R2              **les**
396  303A 2092
397  303C 0200        LI   R0,ARG
397  303E 835C
398  3040 CC0C        MOV  R12,*R0+
399  3042 04F0        CLR  *R0+
400  3044 04F0        CLR  *R0+
401  3046 04D0        CLR  *R0
402            *
403            *    STATUS WAS SET BY THE MOVE ABOVE
404            *
405  3048 130E        JEQ  LOG$$7
406  304A 0760        ABS  @ARG
406  304C 835C
407  304E C020        MOV  @ARG,R0
407  3050 835C
408  3052 0280        CI   R0,99
408  3054 0063
409  3056 1522        JGT  LOG$$9
410  3058 D820        MOVB @FLTONE,@ARG
410  305A 3662
410  305C 835C
411  305E D30C LOG$$6 MOVB R12,R12
412  3060 1302        JEQ  LOG$$7
413  3062 0520        NEG  @ARG
413  3064 835C
414  3066 0202 LOG$$7 LI   R2,FAC
414  3068 834A
415  306A CCA0        MOV  @FHALF,*R2+
415  306C 34D4
416  306E 04F2        CLR  *R2+
417  3070 04F2        CLR  *R2+
418  3072 04D2        CLR  *R2
419  3074 06A0        BL   @FSUB
419  3076 2140
420  3078 0201        LI   R1,LN10
420  307A 34EC
421  307C 0202        LI   R2,ARG
421  307E 835C
422  3080 06A0        BL   @R1$2           move 4 words from *R1 to *R2              **les**
422  3082 2092
423  3084 06A0        BL   @FMULT
423  3086 2244
424  3088 0201        LI   R1,STK1
424  308A 20EC
425  308C 0202        LI   R2,ARG
425  308E 835C
426  3090 06A0        BL   @R1$2           move 4 words from *R1 to *R2              **les**
426  3092 2092
427  3094 06A0        BL   @FADD
427  3096 2144
428  3098 0460        B    @LOGRTN
428  309A 2F8C
429            *
430            *
431            *
432  309C 6820 LOG$$9 S    @CW100,@ARG
432  309E 20CA
432  30A0 835C
433  30A2 D820        MOVB @ARG+1,@ARG+2
433  30A4 835D
433  30A6 835E
434  30A8 D820        MOVB @CBH411,@ARG
```

```
434  30AA 34C4
434  30AC 835C
435  30AE 10D7        JMP  LOG$$6
436                   PAGE
437            *
438            *
439            *  EVALUATE POLYNOMIAL
440            *
441            *
442  30B0 C83B POLY    MOV  *R11+,@P$
442  30B2 20E6
443  30B4 C28B         MOV  R11,R10
444  30B6 100B         JMP  POLY01
445            *
446            *
447            *
448  30B8 C83B POLYX   MOV  *R11+,@P$
448  30BA 20E6
449  30BC C28B POLYX1  MOV  R11,R10
450  30BE 0201         LI   R1,FAC       SQUARE NUMBER IN FAC
450  30C0 834A
451  30C2 0202         LI   R2,ARG
451  30C4 835C
452  30C6 06A0         BL   @R1$2        move 4 words from *R1 to *R2              **les**
452  30C8 2092
453  30CA 06A0         BL   @FMULT
453  30CC 2244
454  30CE 0201 POLY01  LI   R1,FAC
454  30D0 834A
455  30D2 0202         LI   R2,PLYBUF
455  30D4 210C
456  30D6 06A0         BL   @R1$2        move 4 words from *R1 to *R2              **les**
456  30D8 2092
457  30DA C060         MOV  @P$,R1
457  30DC 20E6
458  30DE 0202         LI   R2,FAC
458  30E0 834A
459  30E2 CCB1         MOV  *R1+,*R2+
460  30E4 CCB1         MOV  *R1+,*R2+
461  30E6 CCB1         MOV  *R1+,*R2+
462  30E8 C4B1         MOV  *R1+,*R2
463  30EA C801         MOV  R1,@P$
463  30EC 20E6
464  30EE 1014         JMP  POLY03
465
466  30F0 0201 POLY02  LI   R1,PLYBUF
466  30F2 210C
467  30F4 0202         LI   R2,ARG
467  30F6 835C
468  30F8 06A0         BL   @R1$2        move 4 words from *R1 to *R2              **les**
468  30FA 2092
469  30FC 06A0         BL   @FMULT
469  30FE 2244
470  3100 C060         MOV  @P$,R1
470  3102 20E6
471  3104 0202         LI   R2,ARG
471  3106 835C
472  3108 CCB1         MOV  *R1+,*R2+
473  310A CCB1         MOV  *R1+,*R2+
474  310C CCB1         MOV  *R1+,*R2+
475  310E C4B1         MOV  *R1+,*R2
476  3110 C801         MOV  R1,@P$
476  3112 20E6
477  3114 06A0         BL   @FADD
477  3116 2144
478  3118 C0E0 POLY03  MOV  @P$,R3
478  311A 20E6
479  311C 9813         CB   *R3,@CBH80
479  311E 20D3
480  3120 16E7         JNE  POLY02
481  3122 045A         B    *R10
482                   PAGE
483            *
484            *
```

```
485           *
486  3124 C28B FORMA  MOV  R11,R10      SAVE RETURN
487  3126 0201        LI   R1,FAC
487  3128 834A
488  312A 0202        LI   R2,FORBUF
488  312C 210C
489  312E 06A0        BL   @R1$2        move 4 words from *R1 to *R2              **les**
489  3130 2092
490  3132 0202        LI   R2,ARG
490  3134 835C
491  3136 CCA0        MOV  @FNEG1,*R2+
491  3138 34C2
492  313A 04F2        CLR  *R2+
493  313C 04F2        CLR  *R2+
494  313E 04D2        CLR  *R2
495  3140 06A0        BL   @FADD
495  3142 2144
496  3144 0201        LI   R1,FAC       NOW MOVE FOR BUF TO ARG
496  3146 834A
497  3148 0202        LI   R2,FORBUF    FAC TO FOR BUF
497  314A 210C
498  314C 0203        LI   R3,ARG       AND +1 TO FAC
498  314E 835C
499  3150 CCD2        MOV  *R2,*R3+     FORBUF TO ARG
500  3152 CC91        MOV  *R1,*R2+     FAC TO FORBUF
501  3154 CC60        MOV  @FLTONE,*R1+ +1 TO FAC
501  3156 3662
502  3158 CCD2        MOV  *R2,*R3+     FORBUF TO ARG
503  315A CC91        MOV  *R1,*R2+     FAC TO FORBUF
504  315C 04F1        CLR  *R1+         +1 TO FAC
505  315E CCD2        MOV  *R2,*R3+     FORBUF TO ARG
506  3160 CC91        MOV  *R1,*R2+     FAC TO FORBUF
507  3162 04F1        CLR  *R1+         +1 TO FAC
508  3164 C4D2        MOV  *R2,*R3      FORBUF TO ARG
509  3166 C491        MOV  *R1,*R2      FAC TO FORBUF
510  3168 04D1        CLR  *R1          +1 TO FAC
511  316A 06A0        BL   @FADD
511  316C 2144
512  316E 0201        LI   R1,FORBUF
512  3170 210C
513  3172 0202        LI   R2,ARG
513  3174 835C
514  3176 06A0        BL   @R1$2        move 4 words from *R1 to *R2              **les**
514  3178 2092
515  317A 06A0        BL   @FDIV
515  317C 2380
516  317E 045A        B    *R10
517           *
518           *
519           *
              *
              *       COPY    'C:\Users\Lee\Downloads\TI-99-4A\TurboForth\MDOS\L10\TRINSIC2.a99'
              *
  1           *-----------------------------------------------------------------------
  2           *
  3           *      WRITTEN: 07/13/1985
  4           *
  5           *      FILE:    WDS1.133.TRINSIC2
  6           *
  7           *      NAME:    TRINSIC PART TWO FUNCTIONS
  8           *
  9           *
 10           *-----------------------------------------------------------------------
 11  3180 C80B SQR$$  MOV  R11,@EXTRTN       SAVE RETURN
 11  3182 20D6
 12  3184 C320        MOV  @FAC,R12
 12  3186 834A
 13  3188 1359        JEQ  SQR03
 14  318A 1155        JLT  SQR02
 15  318C D820        MOVB @CBH3F,@FAC
 15  318E 350C
 15  3190 834A
 16  3192 022C        AI   R12,>C100
 16  3194 C100
```

```
17  3196 088C        SRA  R12,8
18  3198 0A1C        SLA  R12,1
19  319A 0201        LI   R1,FAC
19  319C 834A
20  319E 0202        LI   R2,STK1      SAVE A COPY OF FAC
20  31A0 20EC
21  31A2 06A0        BL   @R1$2        move 4 words from *R1 to *R2              **les**
21  31A4 2092
22  31A6 06A0        BL   @POLY
22  31A8 30B0
23  31AA 351C        DATA SQRP
24  31AC 0201        LI   R1,FAC       SAVE P(A) TO STK2
24  31AE 834A
25  31B0 0202        LI   R2,STK2      MOVE STK1 (or A) TO FAC
25  31B2 20F4
26  31B4 0203        LI   R3,STK1
26  31B6 20EC
27  31B8 CC91        MOV  *R1,*R2+
28  31BA CC73        MOV  *R3+,*R1+
29  31BC CC91        MOV  *R1,*R2+
30  31BE CC73        MOV  *R3+,*R1+
31  31C0 CC91        MOV  *R1,*R2+
32  31C2 CC73        MOV  *R3+,*R1+
33  31C4 C491        MOV  *R1,*R2
34  31C6 C453        MOV  *R3,*R1
35  31C8 0201        LI   R1,SQRQ+8
35  31CA 353E
36  31CC 0202        LI   R2,ARG
36  31CE 835C
37  31D0 06A0        BL   @R1$2           move 4 words from *R1 to *R2              **les**
37  31D2 2092
38  31D4 06A0        BL   @FADD
38  31D6 2144
39  31D8 0201        LI   R1,STK2
39  31DA 20F4
40  31DC 0202        LI   R2,ARG
40  31DE 835C
41  31E0 06A0        BL   @R1$2           move 4 words from *R1 to *R2              **les**
41  31E2 2092
42  31E4 06A0        BL   @FDIV
42  31E6 2380
43  31E8 C820        MOV  @CW03,@P$
43  31EA 20C2
43  31EC 20E6
44  31EE 0201 SQR01  LI   R1,STK1
44  31F0 20EC
45  31F2 0202        LI   R2,ARG
45  31F4 835C
46  31F6 06A0        BL   @R1$2           move 4 words from *R1 to *R2              **les**
46  31F8 2092
47  31FA 0201        LI   R1,FAC
47  31FC 834A
48  31FE 0202        LI   R2,STK2
48  3200 20F4
49  3202 06A0        BL   @R1$2           move 4 words from *R1 to *R2              **les**
49  3204 2092
50  3206 06A0        BL   @FDIV
50  3208 2380
51  320A 0201        LI   R1,STK2
51  320C 20F4
52  320E 0202        LI   R2,ARG
52  3210 835C
53  3212 06A0        BL   @R1$2           move 4 words from *R1 to *R2              **les**
53  3214 2092
54  3216 06A0        BL   @FADD
54  3218 2144
55  321A 0202        LI   R2,ARG
55  321C 835C
56  321E CCA0        MOV  @FHALF,*R2+
56  3220 34D4
57  3222 04F2        CLR  *R2+
58  3224 04F2        CLR  *R2+
59  3226 04D2        CLR  *R2
60  3228 06A0        BL   @FMULT
```

```
 60  322A 2244
 61  322C 0620        DEC  @P$
 61  322E 20E6
 62  3230 16DE        JNE  SQR01
 63  3232 0460        B    @EXPSQT
 63  3234 2F3A
 64
 65  3236 D820 SQR02  MOVB @ERRSQR,@FAC+10
 65  3238 20D0
 65  323A 8354
 66  323C 045B SQR03  RT
 67                   PAGE
 68           *===============================================================================
 69           *
 70           *
 71  323E 06A0 COS$$  BL   @COS$$$
 71  3240 3246
 72  3242 0460 SIN$$1 B    @TRINRT           ALL DONE, RETURN NUMBER
 72  3244 2072
 73
 74  3246 C80B COS$$$ MOV  R11,@EXTRTN
 74  3248 20D6
 75  324A 0201        LI   R1,PI2
 75  324C 34F4
 76  324E 0202        LI   R2,ARG
 76  3250 835C
 77  3252 06A0        BL   @R1$2           move 4 words from *R1 to *R2              **les**
 77  3254 2092
 78  3256 06A0        BL   @FADD
 78  3258 2144
 79  325A 1005        JMP  SIN$$2
 80           *
 81           *
 82           *
 83  325C 06A0 SIN$$  BL   @SIN$$$
 83  325E 3262
 84  3260 10F0        JMP  SIN$$1
 85
 86  3262 C80B SIN$$$ MOV  R11,@EXTRTN
 86  3264 20D6
 87  3266 04E0 SIN$$2 CLR  @FAC+8
 87  3268 8352
 88  326A 0201        LI   R1,RPI2
 88  326C 34FC
 89  326E 0202        LI   R2,ARG
 89  3270 835C
 90  3272 06A0        BL   @R1$2           move 4 words from *R1 to *R2              **les**
 90  3274 2092
 91  3276 06A0        BL   @FMULT
 91  3278 2244
 92  327A D320        MOVB @FAC,R12
 92  327C 834A
 93  327E 0760        ABS  @FAC
 93  3280 834A
 94  3282 9820        CB   @FAC,@CBH44
 94  3284 834A
 94  3286 35E2
 95  3288 154E        JGT  TRIERR
 96  328A 0201        LI   R1,FAC
 96  328C 834A
 97  328E 0202        LI   R2,STK1
 97  3290 20EC
 98  3292 06A0        BL   @R1$2           move 4 words from *R1 to *R2              **les**
 98  3294 2092
 99  3296 06A0        BL   @GRINT
 99  3298 3422
100  329A 04C1        CLR  R1
101  329C 04C0        CLR  R0
102  329E D020        MOVB @FAC,R0
102  32A0 834A
103  32A2 130C        JEQ  SIN02
104  32A4 0220        AI   R0,>BA00
104  32A6 BA00
105           *
```

```
106           *
107           *
108  32A8 1507        JGT  SIN01
109  32AA 0220        AI   R0,FAC+7-PAD*256  (FAC+7-PAD)*256
109  32AC 5100
110  32AE 0980        SRL  R0,8
111  32B0 0220        AI   R0,PAD
111  32B2 8300
112  32B4 D050        MOVB *R0,R1
113  32B6 06C1        SWPB R1
114  32B8 0241 SIN01  ANDI R1,3
114  32BA 0003
115  32BC C801 SIN02  MOV  R1,@Q$
115  32BE 20EA
116  32C0 0201        LI   R1,STK1
116  32C2 20EC
117  32C4 0202        LI   R2,ARG
117  32C6 835C
118  32C8 06A0        BL   @R1$2           move 4 words from *R1 to *R2              **les**
118  32CA 2092
119  32CC 06A0        BL   @FSUB
119  32CE 2140
120  32D0 C060        MOV  @Q$,R1
120  32D2 20EA
121  32D4 0911        SRL  R1,1
122  32D6 C801        MOV  R1,@Q$
122  32D8 20EA
123  32DA 1709        JNC  SIN03
124  32DC 0202        LI   R2,ARG
124  32DE 835C
125  32E0 CCA0        MOV  @FLTONE,*R2+
125  32E2 3662
126  32E4 04F2        CLR  *R2+
127  32E6 04F2        CLR  *R2+
128  32E8 04D2        CLR  *R2
129  32EA 06A0        BL   @FSUB
129  32EC 2140
130  32EE C060 SIN03  MOV  @Q$,R1
130  32F0 20EA
131  32F2 1301        JEQ  SIN04
132  32F4 054C        INV  R12
133  32F6 0201 SIN04  LI   R1,FAC
133  32F8 834A
134  32FA 0202        LI   R2,PLWBUF
134  32FC 2114
135  32FE 06A0        BL   @R1$2           move 4 words from *R1 to *R2              **les**
135  3300 2092
136  3302 06A0        BL   @POLYX
136  3304 30B8
137  3306 35D8        DATA SINP
138  3308 0201        LI   R1,PLWBUF
138  330A 2114
139  330C 0202        LI   R2,ARG
139  330E 835C
140  3310 06A0        BL   @R1$2           move 4 words from *R1 to *R2              **les**
140  3312 2092
141  3314 06A0        BL   @FMULT
141  3316 2244
142  3318 054C        INV  R12
143  331A 1102        JLT  SINRTN
144  331C 0520        NEG  @FAC
144  331E 834A
145  3320 C2E0 SINRTN MOV  @EXTRTN,R11
145  3322 20D6
146  3324 045B        RT
147
148  3326 D820 TRIERR MOVB @CBH7,@FAC+10
148  3328 34F7
148  332A 8354
149  332C 10F9        JMP  SINRTN
150
151              PAGE
152           *==============================================================================
153           *
```

```
154           *
155           *
156  332E 0201 TAN$$  LI   R1,FAC
156  3330 834A
157  3332 0202        LI   R2,STK2
157  3334 20F4
158  3336 06A0        BL   @R1$2        move 4 words from *R1 to *R2              **les**
158  3338 2092
159  333A 06A0        BL   @SIN$$$
159  333C 3262
160  333E 0201        LI   R1,FAC       EXCHANGE FAC AND STK2
160  3340 834A
161  3342 0202        LI   R2,STK2
161  3344 20F4
162  3346 C0D1        MOV  *R1,R3
163  3348 CC52        MOV  *R2,*R1+
164  334A CC83        MOV  R3,*R2+
165  334C C0D1        MOV  *R1,R3
166  334E CC52        MOV  *R2,*R1+
167  3350 CC83        MOV  R3,*R2+
168  3352 C0D1        MOV  *R1,R3
169  3354 CC52        MOV  *R2,*R1+
170  3356 CC83        MOV  R3,*R2+
171  3358 C0D1        MOV  *R1,R3
172  335A C452        MOV  *R2,*R1
173  335C C483        MOV  R3,*R2
174  335E 06A0        BL   @COS$$$
174  3360 3246
175  3362 0201        LI   R1,STK2
175  3364 20F4
176  3366 0202        LI   R2,ARG
176  3368 835C
177  336A 06A0        BL   @R1$2        move 4 words from *R1 to *R2              **les**
177  336C 2092
178  336E 9820        CB   @FAC+10,@CBH7
178  3370 8354
178  3372 34F7
179  3374 1305        JEQ  TANRTN
180  3376 C020        MOV  @FAC,R0
180  3378 834A
181  337A 1304        JEQ  TAN01
182  337C 06A0        BL   @FDIV
182  337E 2380
183  3380 0460 TANRTN B    @TRINRT
183  3382 2072
184
185  3384 D820 TAN01  MOVB @ARG,@SIGN
185  3386 835C
185  3388 836E
186  338A 06A0        BL   @OVEXP
186  338C 2354
187  338E 10F8        JMP  TANRTN
188              PAGE
189           *==============================================================================
190           *
191           *    NAME:  ARC TANGENT
192           *
193           *==============================================================================
194  3390 D320 ATN$$  MOVB @FAC,R12
194  3392 834A
195  3394 04E0        CLR  @FAC+8
195  3396 8352
196  3398 0760        ABS  @FAC
196  339A 834A
197  339C 04E0        CLR  @Q$
197  339E 20EA
198  33A0 0207        LI   R7,TANPI8
198  33A2 350C
199  33A4 06A0        BL   @FCOMP7
199  33A6 2120
200  33A8 1318        JEQ  ATN02
201  33AA 1517        JGT  ATN02
202  33AC 0207        LI   R7,TAN3P8
202  33AE 3514
```

```
203  33B0 06A0        BL   @FCOMP7
203  33B2 2120
204  33B4 150C        JGT  ATN01
205  33B6 0201        LI   R1,ARG
205  33B8 835C
206  33BA CC60        MOV  @FNEG1,*R1+
206  33BC 34C2
207  33BE 04F1        CLR  *R1+
208  33C0 04F1        CLR  *R1+
209  33C2 04D1        CLR  *R1
210  33C4 06A0        BL   @FDIV
210  33C6 2380
211  33C8 0203        LI   R3,PI2
211  33CA 34F4
212  33CC 1004        JMP  ATN02A
213
214  33CE 06A0 ATN01  BL   @FORMA
214  33D0 3124
215  33D2 0203        LI   R3,PI4
215  33D4 3504
216  33D6 C803 ATN02A MOV  R3,@Q$
216  33D8 20EA
217  33DA 0201 ATN02  LI   R1,FAC
217  33DC 834A
218  33DE 0202        LI   R2,PLWBUF
218  33E0 2114
219  33E2 06A0        BL   @R1$2           move 4 words from *R1 to *R2              **les**
219  33E4 2092
220  33E6 06A0        BL   @POLYX
220  33E8 30B8
221  33EA 361A        DATA ATNP
222  33EC 0201        LI   R1,PLWBUF
222  33EE 2114
223  33F0 0202        LI   R2,ARG
223  33F2 835C
224  33F4 06A0        BL   @R1$2           move 4 words from *R1 to *R2              **les**
224  33F6 2092
225  33F8 06A0        BL   @FMULT
225  33FA 2244
226  33FC C060        MOV  @Q$,R1
226  33FE 20EA
227  3400 1306        JEQ  ATNSGN
228  3402 0202        LI   R2,ARG
228  3404 835C
229  3406 06A0        BL   @R1$2           move 4 words from *R1 to *R2              **les**
229  3408 2092
230  340A 06A0        BL   @FADD
230  340C 2144
231            *
232            *
233            *
234  340E 054C ATNSGN INV  R12
235  3410 1102        JLT  ATNSG3
236  3412 0520        NEG  @FAC
236  3414 834A
237  3416 0460 ATNSG3 B    @TRINRT
237  3418 2072
238                   PAGE
239            *===============================================================================
240            *
241            *    NAME:  GRINT
242            *
243            *    WHAT:  GREATEST INTEGER FUNCTION
244            *
245            *===============================================================================
246  341A 06A0 GRI$$  BL   @GRINT
246  341C 3422
247  341E 0460        B    @TRINRT
247  3420 2072
248
249  3422 C28B GRINT  MOV  R11,R10          SAVE RETURN
250  3424 D820        MOVB @FAC,@SIGN
250  3426 834A
250  3428 836E
```

```
251  342A 0760        ABS  @FAC
251  342C 834A
252  342E D160        MOVB @FAC,R5
252  3430 834A
253  3432 0985        SRL  R5,8
254  3434 C805        MOV  R5,@EXP
254  3436 836C
255  3438 0285        CI   R5,>0040
255  343A 0040
256  343C 1122        JLT  BITINT
257  343E 0285        CI   R5,>0045
257  3440 0045
258  3442 1517        JGT  INT02
259  3444 0225        AI   R5,->0046
259  3446 FFBA
260            *      SWPB R5
261            *      MOVB R5,@FAC10
262            *      SWPB R5
263  3448 C045        MOV  R5,R1          REPLACES ABOVE 3 INSTRUCTIONS
264  344A 04C2        CLR  R2
265  344C 0203        LI   R3,FAC+8
265  344E 8352
266  3450 A0C5        A    R5,R3
267  3452 F093 INT01  SOCB *R3,R2
268  3454 06C2        SWPB R2
269  3456 DCC2        MOVB R2,*R3+
270  3458 06C2        SWPB R2
271  345A 0585        INC  R5
272  345C 16FA        JNE  INT01
273  345E D020        MOVB @SIGN,R0
273  3460 836E
274  3462 150C        JGT  INT03
275  3464 D082        MOVB R2,R2
276  3466 1305        JEQ  INT02
277  3468 0221        AI   R1,7
277  346A 0007
278  346C 06A0        BL   @ROUNUP
278  346E 2310
279  3470 1005        JMP  INT03
280            *
281            *
282            *
283  3472 D020 INT02  MOVB @SIGN,R0
283  3474 836E
284  3476 1502        JGT  INT03
285  3478 0520        NEG  @FAC
285  347A 834A
286  347C 04E0 INT03  CLR  @FAC+10
286  347E 8354
287  3480 045A        B    *R10
288
289  3482 0200 BITINT LI   R0,FAC
289  3484 834A
290  3486 0201        LI   R1,>BFFF          DEFAULT TO A MINUS ONE
290  3488 BFFF
291  348A D0A0        MOVB @SIGN,R2
291  348C 836E
292  348E 1101        JLT  INT04
293  3490 04C1        CLR  R1
294  3492 CC01 INT04  MOV  R1,*R0+
295  3494 04F0        CLR  *R0+
296  3496 04F0        CLR  *R0+
297  3498 04D0        CLR  *R0
298  349A 10F0        JMP  INT03
299  349C 0380        RTWP         <-----NEVER EXECUTED ???? *********les**
300                   PAGE
301  349E 04C0 CNSTIN CLR  R0
302  34A0 D020        MOVB @FAC,R0
302  34A2 834A
303  34A4 0220        AI   R0,>C000
303  34A6 C000
304  34A8 0A10        SLA  R0,1
305  34AA 0880        SRA  R0,8
306  34AC 04C3        CLR  R3
```

```
307  34AE 9820        CB   @FAC+1,@CBHA
307  34B0 834B
307  34B2 20D2
308  34B4 1102        JLT  CNST10
309  34B6 0580        INC  R0
310  34B8 0583        INC  R3
311  34BA C800 CNST10 MOV  R0,@EXP
311  34BC 836C
312  34BE C0C3        MOV  R3,R3
313  34C0 045B        RT
314                   PAGE
315            *
316            *    MISCELLANEOUS CONSTANTS
317            *
318            *
319  34C2 BFFF FNEG1  DATA >BFFF             FLOATING POINT -1 FIRST WORD
320  0000 34C4 CBH411 EQU  $
321  34C4 4101 EXC127 BYTE >41,1,27,0        127
321  34C6 1B00
322  34C8 0000        BYTE 0,0,0,0
322  34CA 0000
323  34CC BEFF NXC127 DATA ->4101,27*256,0,0
323  34CE 1B00
323  34D0 0000
323  34D2 0000
324  34D4 3F32 FHALF  BYTE >3F,50            .5
325  34D6 0000 ZER3   BYTE 0,0,0,0,0,0
325  34D8 0000
325  34DA 0000
326  34DC 4003 SQRTEN BYTE >40,3,16,22
326  34DE 1016
327  34E0 4D42        BYTE 77,66,01,69
327  34E2 0145
328  34E4 3F2B LOG10E BYTE >3F,43,42,94
328  34E6 2A5E
329  34E8 3013        BYTE 48,19,03,25
329  34EA 0319
330  34EC 4002 LN10   BYTE >40,2,30,25
330  34EE 1E19
331  34F0 5509        BYTE 85,09,29,94
331  34F2 1D5E
332  0000 34F7 CBH7   EQU  $+3
333  34F4 4001 PI2    BYTE >40,1,57,7
333  34F6 3907
334  34F8 6020        BYTE 96,32,67,95
334  34FA 435F
335  34FC 3F3F RPI2   BYTE >3F,63,66,19
335  34FE 4213
336  3500 4D17        BYTE 77,23,67,58
336  3502 433A
337  3504 3F4E PI4    BYTE >3F,78,53,98
337  3506 3562
338  3508 1021        BYTE 16,33,97,45
338  350A 612D
339  0000 350C CBH3F  EQU  $
340  350C 3F29 TANPI8 BYTE >3F,41,42,13
340  350E 2A0D
341  3510 3817        BYTE 56,23,73,10
341  3512 490A
342  3514 4002 TAN3P8 BYTE >40,2,41,42
342  3516 292A
343  3518 0D38        BYTE 13,56,23,73
343  351A 1749
344            *
345            *    SQR POLYNOMINALS
346            *
347  351C 3F3A SQRP   BYTE >3F,58,81,22
347  351E 5116
348  3520 5A00        BYTE 90,00,00,00
348  3522 0000
349  3524 3F34        BYTE >3F,52,67,87
349  3526 4357
350  3528 3200        BYTE 50,00,00,00
350  352A 0000
```

```
351  352C 3E3A        BYTE >3E,58,81,20
351  352E 5114
352  3530 0000        BYTE 00,00,00,00
352  3532 0000
353  3534 8000        DATA SGNBIT
354  3536 4001 SQRQ   BYTE >40,01,00,00
354  3538 0000
355  353A 0000        BYTE 00,00,00,00
355  353C 0000
356  353E 3F09        BYTE >3F,09,99,99
356  3540 6363
357  3542 5000        BYTE  80,00,00,00
357  3544 0000
358  3546 8000        DATA SGNBIT
359  3548 4012 EXPP   BYTE >40,18,31,23
359  354A 1F17
360  354C 3C0F        BYTE 60,15,92,75
360  354E 5C4B
361  3550 4108        BYTE >41,08,31,40
361  3552 1F28
362  3554 4315        BYTE  67,21,29,37
362  3556 1D25
363  3558 4133        BYTE >41,51,78,09
363  355A 4E09
364  355C 135B        BYTE  19,91,51,62
364  355E 333E
365  3560 8000        DATA SGNBIT
366  3562 4001 EXPQ   BYTE >40,01,00,00
366  3564 0000
367  3566 0000        BYTE  00,00,00,00
367  3568 0000
368  356A 4101        BYTE >41,01,59,37
368  356C 3B25
369  356E 2934        BYTE  41,52,36,03
369  3570 2403
370  3572 411B        BYTE >41,27,09,31
370  3574 091F
371  3576 4528        BYTE  69,40,85,16
371  3578 5510
372  357A 412C        BYTE >41,44,97,63
372  357C 613F
373  357E 2339        BYTE  35,57,40,58
373  3580 283A
374  3582 8000        DATA SGNBIT
375  3584 3F23 LOGP   BYTE >3F,35,67,05
375  3586 4305
376  3588 0A1E        BYTE  10,30,88,44
376  358A 582C
377  358C BFF5        BYTE >BF,>F5,98,30
377  358E 621E
378  3590 211F        BYTE  33,31,36,88
378  3592 2458
379  3594 403F        BYTE >40,63,77,54
379  3596 4D36
380  3598 521C        BYTE  82,28,86,17
380  359A 5611
381  359C BEFF        BYTE >BE,>FF,08,83
381  359E 0853
382  35A0 4716        BYTE  71,22,35,58
382  35A2 233A
383  35A4 4039        BYTE >40,57,94,73
383  35A6 5E49
384  35A8 5126        BYTE  81,38,44,44
384  35AA 2C2C
385  35AC 8000        DATA SGNBIT
386  35AE 4001 LOGQ   BYTE >40,01,00,00
386  35B0 0000
387  35B2 0000        BYTE  00,00,00,00
387  35B4 0000
388  35B6 BFF3        BYTE >BF,>F3,13,25
388  35B8 0D19
389  35BA 6148        BYTE  97,72,88,46
389  35BC 582E
390  35BE 402F        BYTE >40,47,45,18
```

```
390  35C0 2D12
391  35C2 1624        BYTE  22,36,02,61
391  35C4 023D
392  35C6 BFC0        BYTE >BF,>C0,07,64
392  35C8 0740
393  35CA 3A07        BYTE  58,07,52,56
393  35CC 3438
394  35CE 401C        BYTE >40,28,97,36
394  35D0 6124
395  35D2 5A45        BYTE  90,69,22,22
395  35D4 1616
396  35D6 8000        DATA SGNBIT
397            *
398            *    SIN POLYNOMIAL
399            *
400            *SINP   BYTE >C4,>FA,44,62    THIS IS THE OLD POLYNOMIAL USED
401            *      BYTE  13,67,49,00
402  35D8 C4FA SINP   BYTE >C4,>FA,44,73
402  35DA 2C49
403  35DC 1000        BYTE  16,00,00,00
403  35DE 0000
404  0000 35E2 CBH44  EQU  $+2
405  35E0 3C05        BYTE >3C,05,68,82
405  35E2 4452
406  35E4 0321        BYTE  03,33,26,88
406  35E6 1A58
407  35E8 C2FD        BYTE >C2,>FD,59,88
407  35EA 3B58
408  35EC 090B        BYTE  09,11,70,31
408  35EE 461F
409  35F0 3E01        BYTE >3E,01,60,44
409  35F2 3C2C
410  35F4 0B44        BYTE  11,68,46,98
410  35F6 2E62
411  35F8 C1D2        BYTE >C1,>D2,81,75
411  35FA 514B
412  35FC 291F        BYTE  41,31,06,02
412  35FE 0602
413  3600 3F07        BYTE >3F,07,96,92
413  3602 605C
414  3604 3E3E        BYTE  62,62,45,62
414  3606 2D3E
415  3608 C0C0        BYTE >C0,>C0,59,64
415  360A 3B40
416  360C 094B        BYTE  09,75,06,22
416  360E 0616
417  3610 4001        BYTE >40,01,57,07
417  3612 3907
418  3614 6020        BYTE  96,32,67,95
418  3616 435F
419  3618 8000        DATA SGNBIT
420            *
421            *    ATN POLYNOMIAL
422            *
423  361A C0FE ATNP   BYTE >C0,>FE,53,57
423  361C 3539
424  361E 124F        BYTE  18,79,88,20
424  3620 5814
425  3622 3F05        BYTE >3F,05,02,79
425  3624 024F
426  3626 0D54        BYTE  13,84,38,85
426  3628 2655
427  362A C0FA        BYTE >C0,>FA,50,69
427  362C 3245
428  362E 635E        BYTE 99,94,01,40
428  3630 0128
429  3632 3F07        BYTE >3F,07,67,37
429  3634 4325
430  3636 0C2B        BYTE  12,43,91,64
430  3638 5B40
431  363A C0F7        BYTE >C0,>F7,08,95
431  363C 085F
432  363E 2F5B        BYTE  47,91,96,72
432  3640 6048
```

```
433  3642 3F0B        BYTE >3F,11,11,10
433  3644 0B0A
434  3646 315C        BYTE  49,92,50,53
434  3648 3235
435  364A C0F2        BYTE >C0,>F2,28,57
435  364C 1C39
436  364E 0C45        BYTE  12,69,75,96
436  3650 4B60
437  3652 3F13        BYTE >3F,19,99,99
437  3654 6363
438  3656 6361        BYTE  99,97,89,96
438  3658 5960
439  365A C0DF        BYTE >C0,>DF,33,33
439  365C 2121
440  365E 2121        BYTE  33,33,32,25
440  3660 2019
441
442  3662 4001 FLTONE BYTE >40,01,00,00      A CONVIENT FLOATING ONE
442  3664 0000
443  3666 0000        BYTE 00,00,00,00
443  3668 0000
444  366A 8000        DATA SGNBIT
445            *
446            *
447            *
               *
               *       COPY    'C:\Users\Lee\Downloads\TI-99-4A\TurboForth\MDOS\L10\MATHS_END.a99'
               *
  1            ******************************
  2            * END directive for Asm994a
  3            ******************************
  4  366C 0000     END
  4


Assembly Complete - Errors: 0,  Warnings: 0


------ Symbol Listing ------

AGTOFC ABS:2150 AGTOFC
ARG    ABS:835C ARG
ATN$$  ABS:3390 ATN$$
ATN01  ABS:33CE ATN01
ATN02  ABS:33DA ATN02
ATN02A ABS:33D6 ATN02A
ATNP   ABS:361A ATNP
ATNSG3 ABS:3416 ATNSG3
ATNSGN ABS:340E ATNSGN
BADM1  ABS:2004 BADM1
BIGF01 ABS:236A BIGF01
BIGFLT ABS:235E BIGFLT
BITINT ABS:3482 BITINT
CBD50  ABS:20D4 CBD50
CBH08  ABS:20C5 CBH08
CBH3F  ABS:350C CBH3F
CBH411 ABS:34C4 CBH411
CBH44  ABS:35E2 CBH44
CBH59  ABS:24F9 CBH59
CBH63  ABS:24FA CBH63
CBH7   ABS:34F7 CBH7
CBH80  ABS:20D3 CBH80
CBHA   ABS:20D2 CBHA
CFI    ABS:2B5C CFI
CFI$$  ABS:2B4A CFI$$
CFI01  ABS:2B9C CFI01
CFI02  ABS:2BAC CFI02
CFI03  ABS:2BB2 CFI03
CFI04  ABS:2BBE CFI04
CFI05  ABS:2BC8 CFI05
CFI06  ABS:2BCA CFI06
CFI08  ABS:2BD6 CFI08
CFI09  ABS:2BE6 CFI09
CFI10  ABS:2BEA CFI10
```

```
CFI11  ABS:2BEC CFI11
CFI12  ABS:2BF0 CFI12
CH0002 ABS:24E2 CH0002
CIF    ABS:2BF6 CIF
CIF01  ABS:2C30 CIF01
CIF02  ABS:2C3E CIF02
CIF03  ABS:2C4E CIF03
CNS    ABS:24FC CNS
CNS01  ABS:253A CNS01
CNSA01 ABS:2924 CNSA01
CNSA02 ABS:292A CNSA02
CNSAST ABS:2908 CNSAST
CNSCHK ABS:28E2 CNSCHK
CNSD01 ABS:27C6 CNSD01
CNSD02 ABS:27DC CNSD02
CNSD03 ABS:27E6 CNSD03
CNSD04 ABS:27F0 CNSD04
CNSD06 ABS:27FA CNSD06
CNSDIG ABS:27B6 CNSDIG
CNSDRT ABS:27B4 CNSDRT
CNSE01 ABS:2814 CNSE01
CNSE02 ABS:2836 CNSE02
CNSE03 ABS:2852 CNSE03
CNSE04 ABS:282C CNSE04
CNSE05 ABS:2846 CNSE05
CNSEX1 ABS:2954 CNSEX1
CNSEXP ABS:27FE CNSEXP
CNSF01 ABS:2568 CNSF01
CNSF02 ABS:2578 CNSF02
CNSF04 ABS:2596 CNSF04
CNSF05 ABS:259E CNSF05
CNSF06 ABS:25BA CNSF06
CNSF07 ABS:25D0 CNSF07
CNSF08 ABS:25D4 CNSF08
CNSF1  ABS:2552 CNSF1
CNSF10 ABS:25E2 CNSF10
CNSF11 ABS:25F4 CNSF11
CNSF12 ABS:25E8 CNSF12
CNSG   ABS:25FA CNSG
CNSG01 ABS:2610 CNSG01
CNSI01 ABS:2860 CNSI01
CNSINT ABS:285C CNSINT
CNSITT ABS:24EA CNSITT
CNSJ00 ABS:266E CNSJ00
CNSJ01 ABS:269A CNSJ01
CNSJ02 ABS:26BC CNSJ02
CNSJ03 ABS:26C8 CNSJ03
CNSJ04 ABS:266A CNSJ04
CNSK   ABS:26EE CNSK
CNSK01 ABS:2720 CNSK01
CNSK1  ABS:26FA CNSK1
CNSL01 ABS:2892 CNSL01
CNSL02 ABS:28BE CNSL02
CNSL03 ABS:28C4 CNSL03
CNSLEA ABS:288C CNSLEA
CNSMLS ABS:2888 CNSMLS
CNSPER ABS:2876 CNSPER
CNSR01 ABS:273C CNSR01
CNSR02 ABS:2772 CNSR02
CNSR03 ABS:278E CNSR03
CNSR04 ABS:2790 CNSR04
CNSR05 ABS:2798 CNSR05
CNSRND ABS:2726 CNSRND
CNSS01 ABS:2936 CNSS01
CNSS02 ABS:293E CNSS02
CNSS03 ABS:295A CNSS03
CNSS04 ABS:2954 CNSS04
CNSSTR ABS:2934 CNSSTR
CNST01 ABS:27B2 CNST01
CNST10 ABS:34BA CNST10
CNSTEN ABS:279A CNSTEN
CNSTIN ABS:349E CNSTIN
CNSU01 ABS:28CA CNSU01
CNSU02 ABS:28DE CNSU02
```

```
CNSUTR ABS:28CC CNSUTR
CNSV01 ABS:26E2 CNSV01
CNSVZR ABS:26CC CNSVZR
CNSX   ABS:261C CNSX
CNSX01 ABS:2638 CNSX01
CNSX02 ABS:2644 CNSX02
CNSX03 ABS:2650 CNSX03
CNSZ01 ABS:287C CNSZ01
CNSZER ABS:2880 CNSZER
COS$$  ABS:323E COS$$
COS$$$ ABS:3246 COS$$$
CSGDRT ABS:2ADC CSGDRT
CSI$00 ABS:2AE0 CSI$00
CSI01  ABS:2B04 CSI01
CSI02  ABS:2B14 CSI02
CSI05  ABS:2B42 CSI05
CSI05A ABS:2B32 CSI05A
CSI07  ABS:2B28 CSI07
CSI08  ABS:2B1C CSI08
CSINT  ABS:2AEA CSINT
CSINT$ ABS:2AB4 CSINT$
CSINTR ABS:2B30 CSINTR
CSN    ABS:2960 CSN
CSN02  ABS:299A CSN02
CSN03  ABS:299C CSN03
CSN04  ABS:29A2 CSN04
CSN05  ABS:29DA CSN05
CSN06  ABS:29E2 CSN06
CSN07  ABS:29EE CSN07
CSN09  ABS:29B6 CSN09
CSN10  ABS:29B4 CSN10
CSN11  ABS:29C6 CSN11
CSNF04 ABS:29F4 CSNF04
CSNG   ABS:2A06 CSNG
CSNG1  ABS:2A32 CSNG1
CSNG16 ABS:2A30 CSNG16
CSNG2  ABS:2A3C CSNG2
CSNH   ABS:2A38 CSNH
CSNH01 ABS:2A6A CSNH01
CSNH02 ABS:2A8A CSNH02
CSNH03 ABS:2AA0 CSNH03
CSNH04 ABS:2A96 CSNH04
CSNH05 ABS:2AA8 CSNH05
CSNH06 ABS:2AAE CSNH06
CSNH07 ABS:2A9A CSNH07
CSNH09 ABS:2A56 CSNH09
CSNH10 ABS:2A4E CSNH10
CSNH11 ABS:2A1C CSNH11
CSNHM6 ABS:2A36 CSNHM6
CSNZER ABS:2A26 CSNZER
CW03   ABS:20C2 CW03
CW08   ABS:20C4 CW08
CW100  ABS:20CA CW100
CW128  ABS:20C6 CW128
CW16   ABS:20C8 CW16
DIVZER ABS:234E DIVZER
DOIT   ABS:2070 DOIT
ERRLOG ABS:20CE ERRLOG
ERRNIP ABS:20CC ERRNIP
ERROV  ABS:0003 ERROV
ERROVF ABS:235C ERROVF
ERRSQR ABS:20D0 ERRSQR
ERRXI1 ABS:237A ERRXI1
EXC127 ABS:34C4 EXC127
EXP    ABS:836C EXP
EXP$$  ABS:2E0A EXP$$
EXP$$$ ABS:2E12 EXP$$$
EXP01  ABS:2DF4 EXP01
EXP03  ABS:2E4C EXP03
EXP04  ABS:2E9E EXP04
EXP05  ABS:2DE8 EXP05
EXPONE ABS:2DD8 EXPONE
EXPP   ABS:3548 EXPP
EXPQ   ABS:3562 EXPQ
```

```
EXPRTN ABS:2E04 EXPRTN
EXPSQ5 ABS:2F4E EXPSQ5
EXPSQ8 ABS:2F66 EXPSQ8
EXPSQT ABS:2F3A EXPSQT
EXTRTN ABS:20D6 EXTRTN
FAC    ABS:834A FAC
FADD   ABS:2144 FADD
FADD02 ABS:2160 FADD02
FADD03 ABS:2162 FADD03
FADD05 ABS:219A FADD05
FADD06 ABS:21E0 FADD06
FADD07 ABS:21EC FADD07
FADD08 ABS:21F6 FADD08
FADD09 ABS:21FA FADD09
FADD1  ABS:2144 FADD1
FADD10 ABS:221E FADD10
FADD11 ABS:2220 FADD11
FADD12 ABS:222C FADD12
FADD13 ABS:2236 FADD13
FADD14 ABS:223C FADD14
FADD15 ABS:2242 FADD15
FADD21 ABS:218C FADD21
FADD2B ABS:2188 FADD2B
FADD30 ABS:2214 FADD30
FCOM01 ABS:2134 FCOM01
FCOMP  ABS:211C FCOMP
FCOMP1 ABS:211C FCOMP1
FCOMP7 ABS:2120 FCOMP7
FCOMRT ABS:213E FCOMRT
FDIV   ABS:2380 FDIV
FDIV01 ABS:234E FDIV01
FDIV04 ABS:23EE FDIV04
FDIV05 ABS:23F8 FDIV05
FDIV06 ABS:241A FDIV06
FDIV07 ABS:241E FDIV07
FDIV08 ABS:2444 FDIV08
FDIV09 ABS:2474 FDIV09
FDIV10 ABS:2476 FDIV10
FDIV11 ABS:247A FDIV11
FDIV12 ABS:2486 FDIV12
FDIV13 ABS:24A4 FDIV13
FDIV14 ABS:24B8 FDIV14
FDIV15 ABS:24CC FDIV15
FDIV16 ABS:24D2 FDIV16
FDVLP  ABS:23EA FDVLP
FDVLPA ABS:2416 FDVLPA
FDVSR  ABS:8354 FDVSR
FHALF  ABS:34D4 FHALF
FLDBIG ABS:2614 FLDBIG
FLTONE ABS:3662 FLTONE
FMCLR  ABS:2270 FMCLR
FMEND  ABS:22CA FMEND
FMUL02 ABS:227C FMUL02
FMUL03 ABS:2286 FMUL03
FMUL04 ABS:229C FMUL04
FMUL05 ABS:22A6 FMUL05
FMULT  ABS:2244 FMULT
FMULZR ABS:22DC FMULZR
FNEG1  ABS:34C2 FNEG1
FORBUF ABS:210C FORBUF
FORMA  ABS:3124 FORMA
FPLLNK ABS:2000 FPLLNK
FPMLIB ABS:2008 FPMLIB
FSUB   ABS:2140 FSUB
FXNTYP ABS:20D8 FXNTYP
FZERO  ABS:22DC FZERO
GRI$$  ABS:341A GRI$$
GRINT  ABS:3422 GRINT
INT01  ABS:3452 INT01
INT02  ABS:3472 INT02
INT03  ABS:347C INT03
INT04  ABS:3492 INT04
LB1    ABS:24F3 LB1
LB10   ABS:24F1 LB10
```

```
LB100  ABS:24EF LB100
LBAST  ABS:24F5 LBAST
LBE    ABS:24F7 LBE
LBPER  ABS:24F6 LBPER
LBSPC  ABS:24F4 LBSPC
LBZER  ABS:24F8 LBZER
LN10   ABS:34EC LN10
LOG$$  ABS:2F74 LOG$$
LOG$$$ ABS:2F7C LOG$$$
LOG$$3 ABS:2F92 LOG$$3
LOG$$5 ABS:2FB6 LOG$$5
LOG$$6 ABS:305E LOG$$6
LOG$$7 ABS:3066 LOG$$7
LOG$$9 ABS:309C LOG$$9
LOG$5A ABS:2FBA LOG$5A
LOG10E ABS:34E4 LOG10E
LOGP   ABS:3584 LOGP
LOGQ   ABS:35AE LOGQ
LOGRTN ABS:2F8C LOGRTN
LW10   ABS:24F0 LW10
LW100  ABS:24EE LW100
LWCNE  ABS:24E6 LWCNE
LWCNF  ABS:24E8 LWCNF
LWCNP  ABS:24E4 LWCNP
LWCNS  ABS:24E2 LWCNS
MATH## ABS:209C MATH##
MATH2  ABS:2032 MATH2
NORM01 ABS:22D2 NORM01
NORM02 ABS:22E6 NORM02
NORM03 ABS:22F6 NORM03
NORM04 ABS:22FE NORM04
NORMAL ABS:22CE NORMAL
NXC127 ABS:34CC NXC127
OCOM01 ABS:2024 OCOM01
OCOMP  ABS:200C OCOMP
OCOMRT ABS:202E OCOMRT
OE$    ABS:20E8 OE$
OV     ABS:235A OV
OV1    ABS:235A OV1
OVEXP  ABS:2354 OVEXP
OVEXP1 ABS:2354 OVEXP1
P$     ABS:20E6 P$
PACK01 ABS:234C PACK01
PACKUP ABS:2332 PACKUP
PAD    ABS:8300 PAD
PI2    ABS:34F4 PI2
PI4    ABS:3504 PI4
PLWBUF ABS:2114 PLWBUF
PLYBUF ABS:210C PLYBUF
POLY   ABS:30B0 POLY
POLY01 ABS:30CE POLY01
POLY02 ABS:30F0 POLY02
POLY03 ABS:3118 POLY03
POLYX  ABS:30B8 POLYX
POLYX1 ABS:30BC POLYX1
PWR$$  ABS:2C50 PWR$$
PWR$$1 ABS:2D78 PWR$$1
PWR$$2 ABS:2D96 PWR$$2
PWR$$3 ABS:2D6A PWR$$3
PWR$$4 ABS:2D80 PWR$$4
PWR$$5 ABS:2DCC PWR$$5
PWRG01 ABS:2D2A PWRG01
PWRG02 ABS:2D22 PWRG02
PWRG05 ABS:2DD2 PWRG05
PWRJ10 ABS:2CD4 PWRJ10
PWRJ30 ABS:2CC0 PWRJ30
PWRJ40 ABS:2D3A PWRJ40
PWRJ41 ABS:2D44 PWRJ41
PWRJ45 ABS:2D5E PWRJ45
PWRRTN ABS:2D40 PWRRTN
Q$     ABS:20EA Q$
R0     ABS:0000 R0
R1     ABS:0001 R1
R1$2   ABS:2092 R1$2
```

```
R10    ABS:000A R10
R11    ABS:000B R11
R12    ABS:000C R12
R13    ABS:000D R13
R14    ABS:000E R14
R15    ABS:000F R15
R2     ABS:0002 R2
R3     ABS:0003 R3
R4     ABS:0004 R4
R5     ABS:0005 R5
R6     ABS:0006 R6
R7     ABS:0007 R7
R8     ABS:0008 R8
R9     ABS:0009 R9
ROUN02 ABS:2318 ROUN02
ROUN1  ABS:2304 ROUN1
ROUND  ABS:2304 ROUND
ROUNUP ABS:2310 ROUNUP
RPI2   ABS:34FC RPI2
SAVCSN ABS:20DA SAVCSN
SAVR12 ABS:20DC SAVR12
SAVR13 ABS:20DE SAVR13
SGNBIT ABS:8000 SGNBIT
SIGN   ABS:836E SIGN
SIN$$  ABS:325C SIN$$
SIN$$$ ABS:3262 SIN$$$
SIN$$1 ABS:3242 SIN$$1
SIN$$2 ABS:3266 SIN$$2
SIN01  ABS:32B8 SIN01
SIN02  ABS:32BC SIN02
SIN03  ABS:32EE SIN03
SIN04  ABS:32F6 SIN04
SINP   ABS:35D8 SINP
SINRTN ABS:3320 SINRTN
SQR$$  ABS:3180 SQR$$
SQR01  ABS:31EE SQR01
SQR02  ABS:3236 SQR02
SQR03  ABS:323C SQR03
SQRP   ABS:351C SQRP
SQRQ   ABS:3536 SQRQ
SQRTEN ABS:34DC SQRTEN
STEX   ABS:234C STEX
STEX01 ABS:234C STEX01
STK1   ABS:20EC STK1
STK2   ABS:20F4 STK2
STK3   ABS:20FC STK3
STK4   ABS:2104 STK4
TAN$$  ABS:332E TAN$$
TAN01  ABS:3384 TAN01
TAN3P8 ABS:3514 TAN3P8
TANPI8 ABS:350C TANPI8
TANRTN ABS:3380 TANRTN
TRIERR ABS:3326 TRIERR
TRINR2 ABS:208C TRINR2
TRINRT ABS:2072 TRINRT
WSG    ABS:83A0 WSG
WSM10  ABS:20E4 WSM10
WSM6   ABS:20E0 WSM6
WSM8   ABS:20E2 WSM8
X3000  ABS:3000 X3000
X3900  ABS:3900 X3900
XDOT   ABS:2E00 XDOT
XMINUS ABS:2D00 XMINUS
XPLUS  ABS:2B00 XPLUS
ZER3   ABS:34D6 ZER3**
```

8 Questions, Bug Reports, *etc*.

The author of the Floating-point library, and the author of TurboForth are regular posters in the TI-99/4A Programming Forum at http://www.atariage.com

Additionally, TurboForth has a dedicated website with a newly opened Forum where support requests can be posted:

http://turboforth.net