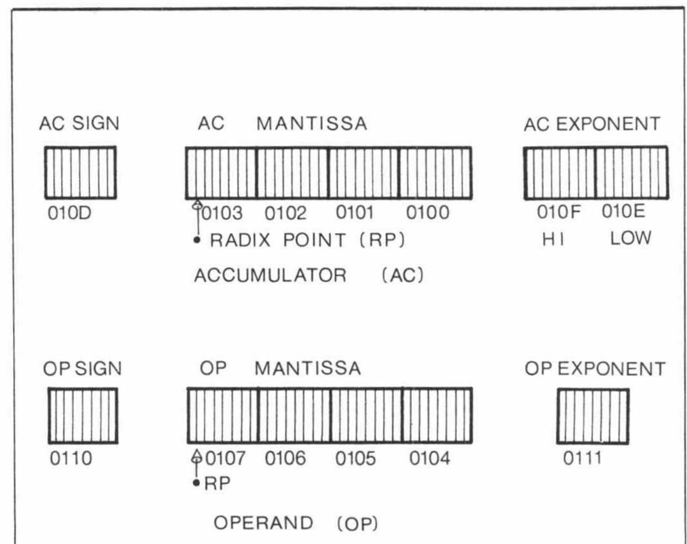
A Floating Point Subroutine Package for the 1802

BY PAUL WASSERMAN
918 Ray Avenue
Union, New Jersey 07083

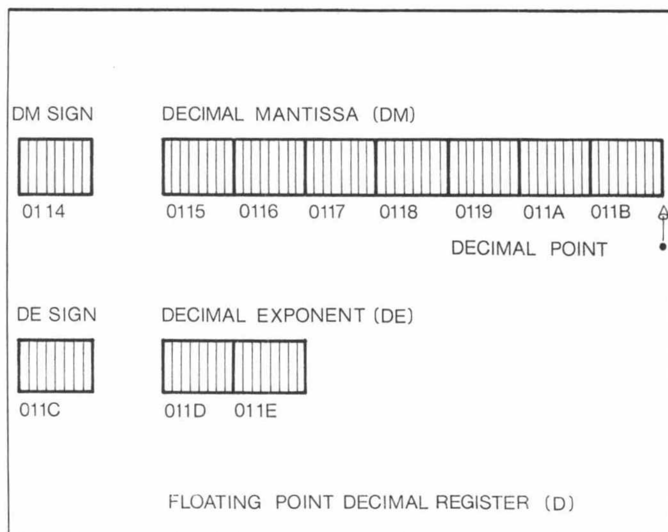
After purchasing my 1802 based microcomputer system, I was surprised to find how little software there is available for it. In particular, I was disappointed by the absence of a readily-available floating point arithmetic subroutine package. To help wipe out this void, I set out to write such a subroutine package. The results of my work appear in the rest of this article.

When this project was started, I decided that the floating point arithmetic routines would implement a set of existing algorithms rather than writing a new set. Suitable algorithms were found in a book written by Neill Graham entitled *Microprocessor Programming for Computer Hobbyists*, published by Tab Books. I strongly recommend the purchase of this book to anyone who wishes to understand or modify the internal workings of this set of floating point arithmetic routines. This article will not attempt to describe the routines' inner workings. It will, however, provide the information needed to use these routines.

All floating point numbers contain two parts: a signed mantissa and a signed exponent. $+1.16 \times 10^{+2}$ is an example of a decimal floating point number. +1.16 is the signed mantissa and +2 is the signed exponent. Similarly, $+1.1101 \times 2^{+110}$ is the binary floating point equivalent of 116. Here, +1.1101 is the signed mantissa and +110 is the signed exponent. The binary mantissa used by the floating point subroutine is 32 bits long (4 bytes). The radix point lies between the left-most bit (LMB) and the bit to the immediate right of the LMB. The sign of the mantissa is represented by a single byte which can take on two values, OO (hex) or FF (hex). If this byte is OO (hex), the mantissa is positive. Similarly, if this byte is FF (hex), the mantissa is negative. The signed binary exponent is represented by a single byte. In order to have both positive and negative integers in the single byte exponent, a convenient numbering scheme was adopted. This scheme defined 80 (hex) as zero. Anything larger than 80 (hex) is positive and anything smaller than 80 (hex) is negative. For example, 83 (hex) represents a binary exponent of +11 = +3 (decimal) and similarly 7F (hex) represents a binary exponent of −1. The exponent can vary between OO (hex) and FF (hex). These numbers represent decimal exponents between the range of 10^{-39} and 10^{38}. Figure 1 illustrates the representation of floating point numbers in the accumulator (AC) and operand (OP) registers. Notice that the accumulator exponent is 2 bytes wide. The hi byte is OO (hex) unless an over or underflow has occurred. The hexidecimal numbers placed below the diagrams in Figure 1 are the memory locations used to store (in RAM) the bytes that make up the accumulator and operand.



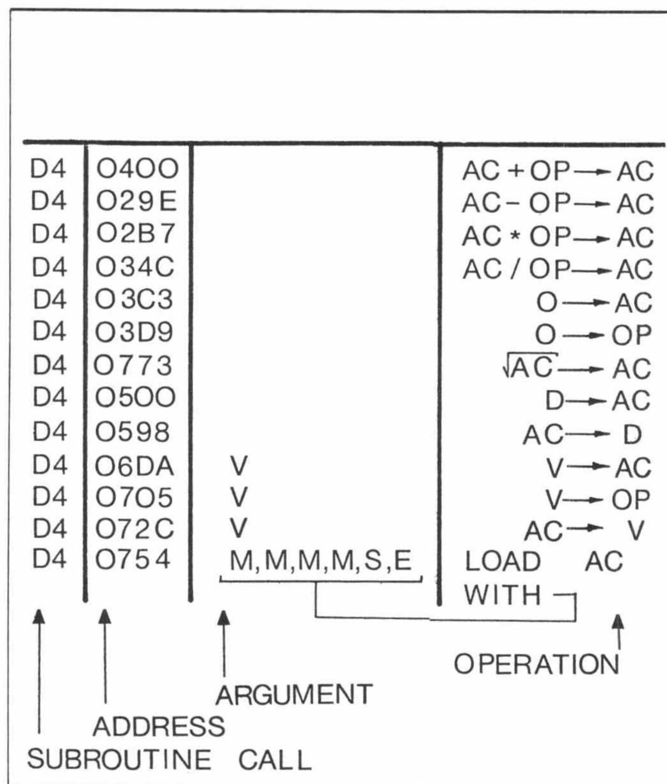
Most people can handle floating point decimal numbers with relative ease. The computer, unlike most people, is more efficient at handling binary numbers. To accommodate this difference, two routines are included in the floating point arithmetic package which allow for the conversion between binary and decimal numbers. Both of these routines use the accumulator register to hold binary numbers. An additional register is used by both routines for decimal numbers. This decimal register (D) is shown in Figure 2. The decimal register is composed of four parts: the sign of the mantissa (DM Sign), the 7-digit decimal mantissa (DM), the sign of the exponent (DE Sign) and the 2-digit exponent (DE). Both signs are represented by their standard ASCII codes; that is, 2D (hex) for negative and 2B (hex) for positive. The decimal digits (0-9) of both the exponent for the mantissa are represented by their corresponding 4-bit binary codes. These codes are placed in the four least significant bits of each byte. The four most significant bits are ignored by the conversion routines. This allows the use of ASCII codes on all entries in decimal register. For example, the number $+1.234567 \times 10^{+2} = +1234567. \times 10^{-4}$ could be represented in the decimal register in the following ways: 2B 01 02 03 04 05 06 07 2D 00 04 or 2B 31 32 33 34 35 36 37 2D 30 34 (ASCII).



FLOATING POINT DECIMAL REGISTER (D)

These conversion routines appear in Figure 3 as D → AC (decimal to binary conversion) and AC → D (binary to decimal conversion).

All the routines in the floating point arithmetic package use the Standard Call and Return Technique (SCRT). This technique is explained in RCA's User Manual for the CDP 1802 on pgs. 61-64. The microprocessor registers R(2)–R(5) must be set up to handle the SCRT before any subroutines can be used. The user must also provide the SCRT code before using any subroutines. A subroutine is called by a D4 (hex) instruction, followed by the address of the subroutine. In some cases, additional numbers (labeled arguments in Figure 3) follow the address. Figure 3 lists all the user subroutines in the floating point arithmetic package.



D4	0400		AC + OP → AC
D4	029E		AC - OP → AC
D4	02B7		AC * OP → AC
D4	034C		AC / OP → AC
D4	03C3		0 → AC
D4	03D9		0 → OP
D4	0773		√AC → AC
D4	0500		D → AC
D4	0598		AC → D
D4	06DA	V	V → AC
D4	0705	V	V → OP
D4	072C	V	AC → V
D4	0754	M,M,M,M,S,E	LOAD AC WITH
SUBROUTINE CALL	ADDRESS	ARGUMENT	OPERATION

The top four routines shown in Figure 3 perform floating point addition, subtraction, multiplication and division, respectively. The next two routines clear (set to zero) the contents of the accumulator or the operand. A square root routine follows next on the list. This routine uses the Newton-Raphson iterative method to find the square root of the contents of the accumulator. The next two routines perform conversions between binary and decimal numbers as previously described. The next three routines allow the user to store and recall numbers from memory. There are forty-two variable storage registers (referred to as variables) numbered 00(hex)-29 (hex). Any one of these variables can be loaded into either the accumulator or operand registers. Similarly, the contents of the accumulator can be stored in any one of the variables. For example, if we wanted to load the accumulator with the contents of variable 08, the following code would be used D4 06DA 08 (08 = V). The last routine shown in Figure 3 loads the accumulator with the binary floating point number that immediately follows the instruction. The first four bytes following the calling address (0754) make up the binary mantissa. These bytes should be in reverse order. For example, if the mantissa equals 42 6C 3D 0E, this subroutine requires 0E, 3D, 6C, 42 be placed after the calling address. As shown in Figure 3, the mantissa sign (S) and exponent (E) should be placed immediately after the mantissa.

The example shown in Figure 4 illustrates how some of these routines may be used to find the area of a circle, given its radius.

D4	0500		D ⟶ AC	Convert radius (R) to binary
D4	072C	01	AC ⟶ V1	Store R in variable1 (V1)
D4	0754	08,EE,87,64,00,82	π ⟶ AC	Load AC with π
D4	0705	01	V1 ⟶ OP	Load OP with V1=R
D4	02B7		AC*OP ⟶ AC	AC = πR
D4	02B7		AC*OP ⟶ AC	AC = πR^2
D4	0598		AC ⟶ D	Convert back to decimal πR^2 is in D

Figure 5 is a listing of the hex code for the floating point arithmetic subroutine package. These subroutines start at the beginning of memory page 01 and end at the last byte of page 08. Page 08 is used for the variable storage. If you want to change the location in memory of these subroutines you must always start them at the beginning of a page. Also, certain modifications to the routines must be made. For example, if you want to move the beginning of the subroutine package two pages, from page 01 to page 03 in memory, you must add 02 to *all* of the underlined bytes. You would also have to add 02 to the high order byte of all the routine calling addresses. This subroutine package uses the following 1802 registers: R(A), R(B), and R(C). In addition, the square root routine uses register R(D) and variables 26 (hex) and 27 (hex). Please note that the SCRT (which you must supply) uses R(2), R(3), R(4) and R(5).

Using this floating point arithmetic package, I have been able to write routines that generate exponentials, sines and cosines. I hope that you will find these routines as useful as I have.

```
0100  A0 A1 A2 A3 00 01 02 03 00 00 00 00 00 A5 AE AE
0110  05 0E 00 00 5D D1 D2 D3 D4 D5 D6 D7 5E E1 E2 00
0120  00 00 00 00 00 00 00 00 00 00 00 00 00 00 F8 01
0130  BA F8 00 AA EA F0 BB F8 04 AA 9B F4 BB F8 00 AA
0140  9B 5A F8 05 AA F0 BB F8 01 AA 9B 74 5A F8 06 AA
0150  F0 BB F8 02 AA 9B 74 5A F8 07 AA F0 BB F8 03 AA
0160  9B 74 5A D5 F8 01 BA F8 04 AA EA F0 BB F8 00 AA
0170  9B F5 5A F8 05 AA F0 BB F8 01 AA 9B 75 5A F8 06
0180  AA F0 BB F8 02 AA 9B 75 5A F8 07 AA F0 BB F8 03
0190  AA 9B 75 5A D5 F8 01 BA F8 00 AA EA F8 01 F4 5A
01A0  1A F8 00 74 5A 1A F8 00 74 5A 1A F8 00 74 5A D5
01B0  F8 01 BA F8 00 AA EA F8 00 F7 5A 1A F8 00 77 5A
01C0  1A F8 00 77 5A 1A F8 00 77 5A D5 F8 01 BA F8 00
01D0  AA EA F0 FE 5A 1A F0 7E 5A 1A F0 7E 5A 1A F0 7E
01E0  5A D5 F3 01 BA F8 08 AA EA F0 FE 5A 1A F0 7E 5A
01F0  1A F0 7E 5A 1A F0 7E 5A D5 F8 01 BA F8 03 AA EA
0200  F0 F6 5A 2A F0 76 5A 2A F0 76 5A 2A F0 76 5A D5
0210  F8 01 BA F8 0B AA EA F0 F6 5A 2A F0 76 5A 2A F0
0220  76 5A 2A F0 76 5A D5 F8 01 BA F8 07 AA EA F0 FE
0230  F0 76 5A 2A F0 76 5A 2A F0 76 5A 2A F0 76 5A D5
0240  F8 01 BA F8 00 AA EA F0 BB F8 08 AA 9B 5A F8 01
0250  AA F0 BB F8 09 AA 9B 5A F8 02 AA F0 BB F8 0A AA
0260  9B 5A F8 03 AA F0 BB F8 0B AA 9B 5A F8 00 AA 5A
0270  1A 5A 1A 5A 1A 5A F8 01 AB D4 01 F9 D4 02 10 3B
0280  84 D4 01 2E 8B FB 1F 32 8C 1B 30 79 D5 D4 01 64
0290  F8 01 AB D4 01 E2 D4 01 CB 3B A0 D4 01 2E 30 AE
02A0  D4 01 64 F8 01 BA F8 08 AA EA F0 F9 01 5A 8B FB
02B0  20 32 B6 1B 30 93 D5 F8 01 BA F8 10 AA EA F0 BB
02C0  F8 0D AA 9B F3 5A F8 11 AA F0 BB F8 0E AA 9B F4
02D0  5A 1A F0 7C 00 5A F8 0E AA F0 FF 81 5A 1A F0 7F
02E0  00 5A D4 02 40 D4 03 00 D5 F8 01 BA F8 10 AA EA
02F0  F0 FB FF 5A D4 04 00 D5 00 00 00 00 00 00 00 00
0300  F8 01 BA F8 00 AA EA F0 1A F1 1A F1 1A F1 3A 19
0310  F8 0F AA F8 00 73 73 73 D5 F8 03 AA F0 FA 80 32
0320  31 D4 01 F9 F8 0E AA EA F0 FC 01 5A 1A F0 7C 00
0330  5A F8 03 AA F0 FA 40 3A 4B D4 01 CB F8 0E AA EA
0340  F0 FF 01 5A 1A F0 7F 00 5A 30 31 D5 F8 01 BA F8
0350  10 AA EA F0 BB F8 0D AA 9B F3 5A F8 11 AA F0 BB
0360  F8 0E AA 9B F5 5A 1A F0 7F 00 5A 2A F0 FC 80 5A
0370  1A F0 7C 00 5A D4 02 8D F8 01 BA F8 08 AA EA F0
0380  BB F8 00 AA 9B 5A F8 09 AA F0 BB F8 01 AA 9B 5A
0390  F8 0A AA F0 BB F8 02 AA 9B 5A F8 0B AA F0 BB F8
03A0  03 AA 9B 5A D4 03 00 D5 F8 01 BA F8 04 AA EA F8
03B0  00 F7 5A 1A F8 00 77 5A 1A F8 00 77 5A 1A F8 00
03C0  77 5A D5 F8 01 BA F8 03 AA EA F8 00 73 73 73 5A
03D0  F8 0F AA F8 00 73 73 73 D5 F8 01 BA F8 07 AA EA
03E0  F8 00 73 73 73 73 F8 11 AA F8 00 73 73 D5 00 00
03F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0400  F8 01 BA F8 11 AA EA F0 BB F8 0E AA 9B F7 3B 76
0410  F8 10 AA F0 BB F8 0D AA F0 AB 9B 5A F8 10 AA 8B
0420  5A F8 11 AA F0 BB F8 0E AA F0 AB 9B 5A F8 11 AA
0430  8B 5A F8 00 AA F0 BB F8 04 AA F0 AB 9B 5A F8 00
0440  AA 8B 5A F8 01 AA F0 BB F8 05 AA F0 AB 9B 5A F8
0450  01 AA 8B 5A F8 02 AA F0 BB F8 06 AA F0 AB 9B 5A
0460  F8 02 AA 8B 5A F8 03 AA F0 BB F8 07 AA F0 AB 9B
0470  5A F8 03 AA 8B 5A F8 0E AA F0 BB F8 11 AA 9B F7
0480  AC FF 1E 3B 86 D5 F8 0D AA F0 BB F8 10 AA 9B F3
0490  32 95 D4 03 A8 8C 32 9E 2C D4 02 27 30 95 D4 01
04A0  2E F8 01 BA F8 0D AA EA F0 BB F8 10 AA 9B F3 32
04B0  CA F8 03 AA F0 FA 80 32 CA D4 01 B0 F8 01 BA F8
04C0  10 AA EA F0 BB F8 0D AA 9B 5A D4 03 00 D5 D4 01
04D0  CB F8 01 BA F8 00 AA EA F0 BB 1A F0 AB F8 04 AA
04E0  9B 5A 1A 8B 5A F8 02 AA F0 BB 1A F0 AB F8 06 AA
04F0  9B 5A 1A 8B 5A D4 01 CB D4 01 CB D4 01 2E D5 00
0500  D4 03 C3 F8 00 AC D4 03 D9 8C FC 15 AA F8 01 BA
0510  EA F0 FA 0F BB F8 04 AA 9B 5A D4 01 2E D4 04 CE
0520  1C 8C FB 07 3A 06 F8 01 BA F8 0E AA EA F8 9F 5A
0530  D4 03 00 F8 01 BA F8 14 AA EA F0 FB 2B 32 45 F8
0540  0D AA F8 FF 5A F8 1D AA F0 FA 0F FE BB F8 13 AA
0550  9B 5A FE FE F4 5A F8 1E AA F0 FA 0F BB F8 13 AA
0560  9B F4 AC D4 03 D9 F8 01 BA F8 11 AA EA F8 84 5A
0570  F8 07 AA F8 50 5A D4 03 4C F8 01 BA F8 1C AA EA
0580  F0 FB 2B 32 8E 8C 32 97 2C D4 03 4C 30 85 8C 32
0590  97 2C D4 02 B7 30 8E D5 D4 03 D9 F8 01 BA F8 11
05A0  AA EA F8 84 5A F8 07 AA F8 50 5A F8 12 AA F8 7A
05B0  5A F8 0E AA F0 FF 85 3B C9 D4 03 4C F8 01 BA F8
05C0  12 AA EA F0 FC 01 5A 30 B1 F8 0E AA F0 FF 81 33
05D0  E1 D4 02 B7 F8 01 BA F8 12 AA EA F0 FF 01 5A 30
05E0  C9 F8 0E AA F0 FB 84 3A FB F8 03 AA F0 FF 51 3B
05F0  FB F8 12 AA F0 FC 01 5A D4 03 4C F8 01 BA F8 0E
0600  AA EA F0 FB 84 3A 0C D4 01 CB 30 21 F0 FF 83 33
0610  21 D4 01 F9 F8 01 BA F8 0E AA EA F0 FC 01 5A 30
0620  0C D4 03 D9 F8 01 BA F8 04 AA EA F8 86 5A D4 01
0630  2E F8 01 BA F8 03 AA EA F0 FF A0 3B 52 F8 12 AA
0640  F0 FC 01 5A F8 03 AA F8 10 5A F8 00 2A 5A 2A 5A
0650  2A 5A F8 03 AA F0 BB F8 15 AA 9B F6 F6 F6 F6 5A
0660  F8 06 AC 8C 32 86 F8 03 AA F0 FA 0F 5A D4 04 CE
0670  F8 01 BA F8 03 AA EA F0 BB 8C FD 1C AA 9B F6 F6
0680  F6 F6 5A 2C 30 63 F8 0D AA F0 32 96 F8 2D BB F8
0690  14 AA 9B 5A 30 9E F8 2B BB F8 14 AA 9B 5A F8 00
06A0  AC AB F8 1C AA F8 2B 5A F8 12 AA F0 FF 80 33 BD
06B0  FB FF FC 01 5A F8 1C AA F8 2D 5A 30 BE 5A F8 12
06C0  AA F0 FF 0A 3B CA 5A 1B 30 C1 F8 1D AA 8B 5A F8
06D0  12 AA F0 BB F8 1E AA 9B 5A D5 F8 01 BA F8 12 AA
06E0  EA 46 FE 5A FE F4 AB F8 08 BB F8 00 AA 4B 5A 1A
06F0  4B 5A 1A 4B 5A 1A 4B 5A F8 0D AA 4B 5A 1A 4B 5A
0700  1A F8 00 5A D5 F8 01 BA F8 12 AA EA 46 FE 5A FE
0710  F4 AB F8 08 BB F8 04 AA 4B 5A 1A 4B 5A 1A 4B 5A
0720  1A 4B 5A F8 10 AA 4B 5A 1A 4B 5A D5 F8 01 BA F8
0730  12 AA EA 46 FE 5A FE F4 AB F8 08 BB F8 00 AA 4A
0740  5B 1B 4A 5B 1B 4A 5B 1B 4A 5B 1B F8 0D AA 4A 5B
0750  1B 4A 5B D5 F8 01 BA F8 00 AA EA 46 5A 1A 46 5A
0760  1A 46 5A 1A 46 5A F8 0D AA 46 5A 1A 46 5A 1A F8
0770  00 5A D5 F8 0A AD D4 07 2C 26 F8 01 BA F8 0E AA
0780  0A FF 01 5A D4 07 2C 27 8D 32 AE D4 06 DA 26 D4
0790  07 05 27 D4 03 4C D4 07 05 27 D4 04 00 F8 01 BA
07A0  F8 0E AA 0A FF 01 5A D4 07 2C 27 2D 30 88 D5 00
07B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
07C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
07D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
07E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
07F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0800  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0810  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0820  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0830  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0840  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0850  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0860  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0870  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0880  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0890  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
08A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
08B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
08C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
08D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
08E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
08F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```