Doublelength Floating-Point Arithmetic on the TMS320C30

APPLICATION REPORT: SPRA114

Al Lovrich
Digital Signal Processor Products
Semiconductor Group
Texas Instruments

Digital Signal Processing Solutions



IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain application using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

Copyright © 1997, Texas Instruments Incorporated

TRADEMARKS

TI is a trademark of Texas Instruments Incorporated.

Other brands and names are the property of their respective owners.

CONTACT INFORMATION

US TMS320 HOTLINE (281) 274-2320

US TMS320 FAX (281) 274-2324

US TMS320 BBS (281) 274-2323

US TMS320 email dsph@ti.com

Doublelength Floating-Point Arithmetic on the TMS320C30

Abstract

This chapter, reprinted from *IEEE Micro Magazine*, describes the third generation of the TMS320 family of digital signal processors, the TMS320C30. It describes the origin and development of the 32-bit floating-point device. The topics covered include:

- An overview of the characteristics of the TMS320C30 processor
 A description of the architecture of the 320C30
 A description of the software features of the programmable DSP
 A description of development tools and support
 The types of demanding applications for which the 320C30 is most suitable
 Support graphics include:
 Architecture block diagrams
 Diagrams showing on-chip memory, cache and buses, the 320C30 central processing unit, and peripheral bus and peripherals
- □ A pipeline of 320C30 instructions
- □ Sample code implementations

The chapter closes with an endnote about the likely direction of this technology, a list of references and some biographical information about the authors.



Product Support

World Wide Web

Our World Wide Web site at www.ti.com contains the most up to date product information, revisions, and additions. Users registering with TI&ME can build custom information pages and receive new product updates automatically via email.

Email

For technical issues or clarification on switching products, please send a detailed email to (dsph@ti.com). Questions receive prompt attention and are usually answered within one business day.

In the past, extended-precision arithmetic has been implemented only on fixed-point processors. The introduction of the TMS320C30 Digital Signal Processor (DSP), a floating-point 33-MFLOP device, enables us to represent multilength floating-point math in terms of singlelength floating-point math. Extended-precision arithmetic allows designers to have more accuracy in their applications. Some of these applications include digital filtering, FFTs, image processing, control, etc.

This application report describes how to extend the available precision of floating-point arithmetic on the TMS320C30. Our emphasis is on implementing an efficient extension of the available precision while minimizing both the execution time and the memory usage.

The structure of this report is as follows: The first section describes the TMS320C30 DSP floating-point number representation. The second section discusses doublelength arithmetic and some basic definitions. The third section discusses the algorithms used along with the TMS320C30 implementation. An analysis of the error introduced by the algorithm is presented in the fourth section. The last section provides an insight into generating C-callable functions from assembly language routines. Finally, the appendix provides the source listings for the extended-precision arithmetic.

Floating Point Format

The TMS320C30 supports three floating-point formats [1].

- Short floating-point format, used to represent immediate operands, consisting of a 4-bit exponent and a 12-bit mantissa.
- Single-precision format, used for regular floating-point value representation, consisting of an 8-bit exponent and a 24-bit mantissa.
- The extended-precision format, used with the extended-precision registers, consisting of an 8-bit exponent and a 32-bit mantissa.

For the extended-precision algorithms to work properly on the DSP, it is important to start from the highest-precision floating-point format available in the system that is used for basic floating-point operations. The single-precision format is of particular interest in developing the TMS320C30 code for extended-precision floating-point operations. Therefore, a working knowledge of the properties of this format is essential for the concepts presented in this application report.

In the single-precision format, the floating-point number is represented by an 8-bit exponent field (e) in two's complement notation, and a two's complement 24-bit mantissa field (f) with an implied most-significant nonsign bit. Bit 23 of the mantissa indicates the sign (s), as shown in Figure 1.

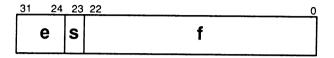


Figure 1. Single-Precision Floating-Point Format of the TMS320C30

Operations are performed with an implied binary point between bits 23 and 22. When the implied most-significant nonsign bit is made explicit, it is located to the immediate left of the binary point after the sign bit. We show the implied bit explicitly throughout this application report for clarity. The floating-point number x is expressed as follows:

```
x = 01.f \times 2^{e} if s = 0;

10.f \times 2^{e} if s = 1;

0 if e = -128, s = 0, and f = 0
```

The range and precision available with the TMS320C30 single-precision floating-point format are illustrated by the following values:

```
Most Positive: x = +3.4028234 \times 10^{+38}

Least Positive: x = +5.8774717 \times 10^{-39}

Least Negative: x = -5.8774724 \times 10^{-39}

Most Negative: x = -3.4028236 \times 10^{+38}
```

Doublelength Floating-Point - The Basics

The techniques used to develop doublelength results in this application report require a singlelength floating-point system and arithmetic that satisfy certain conditions. The TMS320C30 implementation takes the singlelength system as the highest floating-point precision system available. The algorithms presented do not require a doublelength accumulator with respect to the singlelength system used. The extended-precision formats available are used to control the truncation or rounding of the single-precision results.

The doublelength arithmetic presented here increases precision of a given floating-point operation without the need for a doublelength accumulator. Using this method, the result of the floating-point operations on two single-precision numbers can be determined exactly. If x and y are two such numbers and the desired operation is addition, the result can be represented as a pair of floating-point numbers z and zz. The z value represents

the most significant portion of the floating-point operation, while zz represents the least significant portion of the floating-point operation.

As an example, consider the result of the exact addition of two floating-point numbers x and y that are expressed in the single-precision format of the TMS320C30:

```
x = 217FFFFFh (decimal: 1.71798682 × 10<sup>10</sup>)

y = 0C7FFFFFh (decimal: 8.19199951 × 10<sup>3</sup>)
```

The values are represented in the TMS320C30 binary equivalent as follows:

```
x = 2^{33} \times 01.111 \ 1111 \ 1111 \ 1111 \ 1111 \ 1111 \ 1111
```

Addition of two floating-point numbers requires aligning the two variables x and y [1]:

```
x = 2^{33} \times 01.111\ 1111\ 1111\ 1111\ 1111\ 1111b

y = 2^{33} \times 00.000\ 0000\ 0000\ 0000\ 0000\ 0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 11000b
```

As can be seen in this example, most of the precision available for y will not be available to carry out the addition. Maintaining full precision for floating-point addition requires extra mantissa bits beyond the 24 bits available on the DSP. Since the need for such precision is rare, software methods are used to represent the result of the operation as a floating-point number pair (z,zz). In our example, the exact result is represented as follows:

```
z = 2^{34} \times 01.000\ 0000\ 0000\ 0000\ 0000\ 0011b

zz = 2^{9} \times 01.111\ 1111\ 1111\ 1111\ 1111\ 1000b
```

The corresponding hexadecimal representation of (z,zz) is shown below:

```
z = 22000003h (decimal: 1.71798753 × 10<sup>10</sup>)

zz = 097FFFF8h (decimal: 1.0239995 × 10<sup>3</sup>)
```

Some definitions are basic to the development of concepts in this report. First is the definition of the floating-point operations over a system R. The system contains all the possible floating-point numbers that the single-precision format of the TMS320C30 can represent. All the floating-point arithmetic is carried out in base 2. Therefore, R can be represented as follows on the TMS320C30:

$$R = \{x | x = m(x)2e(x), |m(x)| < 2^{24}, -128 < e(x) < 127\}$$

A floating-point operation is *faithful* if the result of the operation fl(x * y) equals either:

The largest element of R that is smaller than or equal to (x * y) or

The smallest element of R that is larger than or equal to (x * y)

where * represents one of the following floating-point operations: $+, -, \times, \div$. In other words, faithful refers to truncating the floating-point operation result. The floating-point

multiplier on the TMS320C30 saves the upper 40 bits of the mantissa in one of the extended-precision registers [1] and drops the least significant byte of the result. By this definition, the floating-point multiplication on the TMS320C30 is *faithful*. Since the algorithms require the floating-point result to be in single-precision format, the floating-point multiplication on the DSP must therefore be followed by a second truncation step. Saving the contents of the extended-precision register to a memory location or masking off the low 8 bits results in truncation.

A floating-point operation is *optimal* if for all x and y, the result of fl(x * y) is an element of R nearest to (x * y). In other words, the round-off error should not exceed one-half of the last remaining bit position. This is commonly referred to as *rounding*.

The results of floating-point operations on the TMS320C30 are stored in the extended-precision registers [1]. The extended-precision register adds 8 bits of precision to the floating-point arithmetic result. Execution of the RND (round) instruction forces the result of the floating-point arithmetic to be *optimal*. When you round the result of the addition or subtraction operations on the TMS320C30, these floating-point operations become *optimal*.

Implementing Doublelength Floating-Point Arithmetic

This section presents the algorithms used in implementing doublelength arithmetic in pseudo-code for a number of fundamental floating-point operations. The basic idea of doublelength arithmetic can be extended to multiplelength precision, given that the start of the implementation is based on the highest precision available on the system. Therefore, to achieve quadruplelength results, the same algorithm can be applied to doublelength values, and so on. The implementation is based on the theoretical results presented in Reference [2].

Exact Singlelength Addition

In this discussion of the algorithm used to carry out *exact* addition and its implementation on the TMS320C30 DSP, the term *exact* refers to performing an operation on two floating-point numbers, x and y, and obtaining a doublelength floating-point number pair (z,zz) to represent the result. In this implementation, we have not accounted for floating-point exponent overflow or underflow. For this algorithm to produce a correct result, the floating-point addition and subtraction must be *optimal*.

The purpose of exact addition is to find a term, zz, that satisfies Equation (2).

$$z + zz = x + y \tag{2}$$

Equation (2) can be rewritten as

$$zz = y - (z - x) \tag{3}$$

Equation (3) can be expanded into Equation (4).

$$w = z - x$$

$$zz = y - w$$
(4)

In particular, |x| > |y| must be valid for Equation (4) to be valid. Implementation of Equation (4) on the TMS320C30 always generates the exact correction term zz if the result of floating-point addition operation is made *optimal*. This requirement guarantees that the result of single-precision floating-point add and subtract belongs to system R. By swapping the x and y values when |x| < |y|, the condition for obtaining an *exact* result is met.

The algorithm requires that x and y be normalized. Normalization guarantees that the floating-point number has only one sign bit, and that sign bit is followed by nonsign bits [1]. Floating-point addition on the TMS320C30 assumes that the operands are normalized.

The TMS320C30 assembly code for obtaining the doublelength sum of two singlelength floating-point numbers x and y is shown in Appendix A. First, the values for x and y are interchanged when |x| < |y|. When you add x and y values, the number with the smaller exponent, y, is shifted repeatedly until the exponents of x and y are equal and their mantissas are aligned. We have now calculated the singlelength number, z, that satisfies Equation (2). Since the floating-point addition on the TMS320C30 is made optimal by rounding, the extra precision is, in effect, dropped. The extra precision value, zz, is obtained by implementing Equation (4). Figure 2 is a graphical representation of the implemented algorithm. The figure also shows the relationship between doublelength number pair (z,zz) and singlelength floating-point numbers and their representation on the TMS320C30.

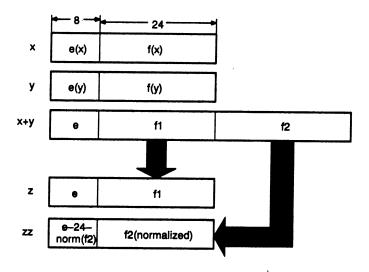


Figure 2. Exact Singlelength Addition

The same algorithm can be used to implement exact floating-point subtraction on the DSP. This is accomplished by negating the second operand and performing an exact addition.

Doublelength Addition

A natural extension of exact singlelength addition and subtraction is its application to doublelength arithmetic. Figure 3 shows an algorithm for implementing doublelength addition on the DSP. Using this algorithm, you can add two doublelength numbers (x,xx) and (y,yy) and represent the result as a doublelength number (z,zz).

The algorithm requires forming a doublelength number (r,rr) that represents an exact addition of x and y. Generating a second number, s = ((rr + yy) + xx), results in a number pair (r,s) that approximates the addition of (x,xx) and (y,yy). Finally, an exact addition of r and s generates a doublelength number (z,zz) that has the same value as (x,xx) + (y,yy).

To obtain exact results for addition and subtraction, subtraction and addition must be optimal; this is guaranteed by following each subtraction or addition instruction on the DSP with a round instruction.

```
; Calculate the doublelength sum of (x,xx) and (y,yy), ; the result being (z,zz);

r = x + y;
if (abs(x)>abs(y))
s = x - r + y + yy + xx;
else
s = y - r + x + xx + yy;
z = r + s;
zz = r - z + s;
```

Figure 3. Doublelength Addition

Exact Singlelength Multiplication

The exact singlelength multiplication is shown in Figure 4. The algorithm requires breaking the x and y mantissas into half-length numbers, referred to as head (hx,hy) and tail (tx,ty) sections [2]. This algorithm requires addition and subtraction to be optimal and multiplication faithful. The TMS320C30 DSP multiplication result is faithful if the contents of the extended-precision register are truncated.

To split x and y into two half-length numbers, a constant value is needed that is dependent on the number of available digits. The TMS320C30 device has t=24 bits of mantissa in the single-precision format. Equation (5) shows that head section hx is chosen to be as near to the value of x as possible.

$$hx = round(m(x)2^{-t1})2^{e(x)+t1}$$
 (5)

Also, t1 is chosen to be approximately one-half of the available precision, or 12, on the processor. This effectively breaks the mantissa into half-length values. Equation (5) shows that hx is obtained by rounding and is defined to be an element of $R\{t1\}$. The tail section tx is easily obtained by subtracting hx from x. Since floating-point subtraction can be made optimal on the TMS320C30, it follows that tx is an element of $R\{t1-1\}$. Setting the constant equal to 2^{12} does not always satisfy Equation (5) when t is even. When the constant is set to $2^{12}+1$, the definition of Equation (5) is satisfied. The proof for the above is given in Reference [2].

```
; Calculate the exact product of x and y, the result being; a doublelength number (z,zz). This algorithm uses the; following syntax when called from a user program as shown; mult12 (x,y,z,zz);

p = x \times constant;
hx = x - p + p;
tx = x - hx;
p = y \times constant;
hy = y - p + p;
ty = y - hy;
p = hx \times hy;
q = hx \times ty + tx \times hy;
z = p + q;
zz = p - z + q + tx \times ty;
```

Figure 4. Exact Singlelength Product

Doublelength Multiplication

The doublelength multiplication algorithm, shown in Figure 5, relies on the singlelength algorithm discussed earlier. The algorithm generates a nearly doublelength approximation of the output result (c,cc). Note that the exact singlelength multiplication routine is used for this approximation. Exact addition is used to generate a doublelength floating-point number that is the closest approximation to the actual result.

The doublelength product program implementation uses the TMS320C30 stack capabilities to save some intermediate variables. These programs are written to be used as callable functions or macros in your program. In either case, the stack pointer must be set to a valid memory segment for proper code execution.

```
; Calculate the doublelength product of (x,xx) and (y,yy); the result being a nearly doublelength number (z,zz).; Program uses exact singlelength multiplication, mult12 (.).;

mult12 (x, y, c, cc);

cc = x × yy + xx × y + cc;

z = c + cc;

zz = c - z + cc;
```

Figure 5. Exact Doublelength Product

Doublelength Quotient and Square Root

Figures 6 and 7 show the algorithm used in calculating the doublelength quotient and doublelength square root routines. Singlelength multiplication is used to generate a doublelength approximation of the quotient or square root values. As with doublelength multiplication, exact addition is used to generate a doublelength floating-point result.

```
; Calculates the doublelength quotient of (x,xx) and (y,yy); the result being (z,zz);

c = x / y;

mult12(c, y, u, uu);

cc = (x - u - uu + xx - c × yy) / y;

z = c + cc;

zz = c - z + cc;
```

Figure 6. Doublelength Quotient

Figure 7. Doublelength Square Root

Error Analysis

This section discusses and determines an upper bound for the error generated in forming a doublelength result. The value of the doublelength number (z,zz) is equal to z + zz. Singlelength addition, subtraction, and multiplication results are always exact. In doublelength addition, any error introduced in the end result is generated by calculating the zz term. An upper bound error magnitude has been calculated in Reference [2] and is shown in Equation (6) as follows:

$$|E^+| \le \{|x + xx| + |y + yy|\} \times 2^{2-2t} = |Z| \times 2^{2-2t}$$
 (6)

where t = 24 for this system. This gives an upper bound of $|Z| \times 2^{-46}$, or approximately $|Z| \times 1.42 \times 10^{-14}$. This translates to a theorical accuracy greater than 13 decimal places. Table 1 shows an example of doublelength addition using the exact addition algorithm previously described. The numbers in the left column represent TMS320C30 hexadecimal notation for the floating-point results, and (z,zz) is the decimal equivalent of the doublelength output result. Appendix B shows a listing of C programs (exact) that convert from TMS320C30 hexadecimal notation to decimal notation.

Table 1. Exact Singlelength Arithmetic Examples

		Singlelength Addition
×	= 217FFFFFh	
у	= 0C7FFFFh	
z	= 22000003h	(z,zz) = 17179876351.9995117 (Exact)
zz	= 097FFFF8h	17179876351.9995117 (DSP)
x y	= FC7C8923h = OA29A7E5h	
z	= 0A29ABD8h	(z,zz) = 1357.37010409682989 (Exact)
zz	= EFA46000h	1357.37010409682989 (DSP)
		Singlelength Multiplication
×	= OF7FFFFh	
у	= 21FFFFFFh	
z	= 30800000h	(z,zz) = -562949986975740 (Exact)
zz	= 18800002h	- 562949986975740 (DSP)
	= FC7CB923h	
X		
y _	= 0A29A7E5h	4) 407 404000004740047
Z	= 07277BF7h	(z,zz) = 167.484236862815123 (Exact)
ZZ	= EBA714F0h	167.484236862815123 (DSP)

The doublelength product, quotient, and square-root algorithms all have a small relative error. The upperbound error magnitude for each is given in Equations (7) through (9).

$$|E^{\times}| = (|x + xx| \times |y + yy|) \times 11 \times 2^{-48}$$
 (7)

$$|E^+| = (|x + xx| + |y \times yy|) \times 21.1 \times 2^{-48}$$
 (8)

$$|E^{\sqrt{}}| = \operatorname{sqrt}(|x + xx|) \times 12.7 \times 2^{-48}$$
 (9)

Equation (7) establishes an upperbound of $|Z| \times 3.9 \times 10^{-14}$, or approximately 13 decimal digits of accuracy for doublelength multiplication. Similarly, an upperbound of $|Z| \times 7.5 \times 10^{-14}$, or greater than 13 decimal digits for the doublelength squareroot algorithm, is established. Table 2 shows examples for each algorithm discussed, along with the algorithm output and expected theorical output.

Table 2. Exact Doublelength Arithmetic Examples

		Doublelength Multiplication
<u> </u>		- canceright maniphorner
×	= 2200000h	
xx	= 097FFFFEh	
У	= 21000001h	
уу	= 097FFFFEh	
z	= 43000002h	$(z,zz) = 1.47573996570139475 \times 10^{20}$ (Exact)
zz	= 2A7FFFFCh	1.47573996570139427 × 10 ²⁰ (DSP)
×	= 22000003h	
xx	= 097FFFF8h	
y	= 0A29ABD8h	
yy	= EFA46000h	
Z	= 2C29ABDDh	(z,zz) = 23319450552284.2434 (Exact)
zz	= 13907DC2h	23319450552284.1250 (DSP)
		Doublelength Quotient
×	= 43000002h	
ХX	= 2A7FFFFCh	
У	= 2C29ABDDh	
уу	= 13907DC2h	
z	= 1641205Ah	(z,zz) = 6328365.08044074177 (Exact)
zz	= FC24BE20h	6328365.08044075966 (DSP)
×	= 2200000h	
ХX	= 097FFFFEh	
У	= 21000001h	
уу	= 097FFFEh	
z	= 007FFFDh	(z,zz) = 1.99999964237223082 (Exact)
zz	= D3400000h	1.99999964237217398 (DSP)
		Doublelength Square Root
×	= 2C2BDD00h	
xx		
z	= 61451A4h	(z,zz) = 4860114.04539400958 (Exact)
zz	= FB39EF11h	4860114.04539400712 (DSP)
×	= 21000001h	
xx	= 097FFFFEh	
z	= 103504F5h	(z,zz) = 92681.9110722252960 (Exact)
zz	= F7BC0784h	92681.9110722253099 (DSP)
l		

Note that the results were obtained using the programs shown in Appendix B. The C programs were created and compiled on a 80386-based microcomputer running under MS-DOS 3.3.

How to Generate C-Callable Functions

The source listings for the extended-precision arithmetic presented in Appendix A are optimized for execution speed and code size. These routines are designed to be used as macros in a user program environment or, with a few adjustments, as a C function.

This section provides an overview of TMS320C30 C compiler calling conventions necessary to create functions that can be added to the C compiler library. You need a working knowledge of C language to understand the terminology in this section [4, 5, 6].

The C compiler uses the processor stack to pass arguments to functions, store local variables, and save temporary values. The C compiler uses two registers of the TMS320C30 to manage the stack pointer (SP) and the frame pointer (AR3).

When a C program calls a function, it must

- 1. Push the arguments onto the stack,
- 2. Call the function, and
- 3. Pop the arguments off the stack,

in that order.

On the other hand, the called C function must perform the following tasks:

- 1. Set up a local frame by saving the old frame pointer on the stack.
- 2. Assign the new frame pointer to the current value of stack pointer.
- 3. Allocate the frame.
- 4. Save any dedicated registers that the function modifies.
- 5. Execute function code.
- 6. Store a scalar value in R0.
- 7. Deallocate the frame.
- 8. Lastly, restore the old frame pointer [4].

The following code segment shows the singlelength addition routine modified to be in C-callable form. Note that registers R4 through R7 and AR4 through AR7 are dedicated registers used by the compiler. These registers must be saved as floating-point values.

single	.set	UFF
fp	.set	ar3
X	.set	r0
у	.set	r1
Z	.set	r2
ZZ	.set	r3

```
r4
W
          .set
                   r2
x1
          .set
٧1
          .set
                   r3
                   __add12:
          .global
          .width
                   96
          .text
  add12:
                                 ; Save old fp
          push
                  fp
          pushf
                   r4
          push
                   r4
          ldi
                  sp,fp
                                 ; Point to top of stack
          ldi
                   * - fp(2),r0
                                 ; Load x into rO
                   * - fp(3),r1
                                 ; Load y into r1
          ldi
          absf
                  x,x1
          absf
                  v.v1
          cmpf
                  v1,x1
                                 : |x| > |y|
          Idflt
                  x.x1
          Idflt
                  V.X
          dflt.
                  x1,y
          addf3
                  x,y,z
                                 ; z = x + y
          rnd
                  Z
          subf3
                  X,Z,W
                                 ; Form w = z - x
          rnd
                  W
                                 zz = y - (y - w)
          subf3
                  W,y,ZZ
          rnd
                  ZZ
                  r4
          pop
          popf
                  r4
                                 : Restore fp
          pop
                  fp
          retsu
          .end
```

Conclusion

This report presented an implementation of extended-precision arithmetic routines for the TMS320C30 DSP. The programs presented include singlelength floating-point addition, subtraction, and multiplication, which produce exact doublelength results. Doublelength floating-point addition, subtraction, multiplication, division, and square root were also presented. The doublelength floating-point routines all had a small relative error that appeared in the correction term zz. However, it has been shown that the accuracy of the doublelength floating-point result is at least 13 decimal digits. Table 3 is a summary of information about the routines contained in Appendices A and B. Execution times shown

in the table are given only for the routines in Appendix A. These times do not include the call and return if the routine is implemented as a called function. They also do not include any context saves and restores that may be required.

Table 3. Summary Information

Routine	Mnemonic	Appendix	Code Size (Words)	Execution (Cycles)
Singlelength Add	add12	A1	12	12
Doublelength Add	dbladd	A2	25	25
Singlelength Multiply	mult12	А3	35	35
Doublelength Multiply	mult2	A4	51	51
Doublelength Divide	div2	A5	115	115
Doublelength Square Root	_sqrt2	A6	163	163
Change Two Single-Precision TMS320C30 Numbers to One Double-Precision Result	C30DBL	B1		
Change Two Double-Precision TMS320C30 Numbers to a	C3ODBL	В		
Double-Precision Result	C30DBL2	В2		

References

- [1.] Third-Generation TMS320 User's Guide (literature number SPRU031), Texas Instruments, Inc., 1988.
- [2.] Dekker, T.J., "A Floating-Point Technique for Extending the Available Precision", Numer. Math. 18, 1971, pp 224-242.
- [3.] Linnainmaa, S., "Software for Doubled-Precision Floating-Point Computations", ACM Transactions on Mathematical Software, Vol. 7, No. 3, Sept. 1981, pp 272-283.
- [4.] TMS320C30 C Compiler (literature number SPRU034), Texas Instruments, Inc., 1988.
- [5.] Kernigan, B.W. and Ritchie, D.M., *The C Programming Language*, 2nd Revision, Prentice-Hall, Englewood Cliffs, New Jersey, 1978.
- [6.] Kochan, S.G., *Programming in C*, Second Edition, Howard K. Sams, Indianapolis, Indiana, 1988.

Appendix A

Appendix A1. Single Length Add

+ FINCTION	FINCTION DE 1. add12	12	************
		ł	٠
* AUTHOR:		2/21/89	
+ Texas	Instruments,	, Ik.	
• Estry Co	Entry Conditions:		
+ Upon en	~	centains (x,y)	
• Exit Cer	Exit Conditions		
* Upon exit	(12,13)	contains (z,zz).	
# Registers	's Affected:		
* 10, 11,	r0, r1, r2, r3, r4		
•			
* Revision	Origin		
• Execution		12 cycles	

single	ž	=	
	-globel	_add12	
×	ž.	2	
`	ž	z	
~	ž	2	
22	.se	2	
	.se	ŧ	
¥	ž.	2	
7.	.set	5	
	.text		
_add12:			
	absf	x,x	
	s pa	y,yl	
	Ť	y1,x1	ت-
	1461t	x,xi	; if not, exchange x & y
	1461	y,x	
	1961	x1,y	
•	9		
	5194	x, y, z	1 2 = X + 3
	Į	~	
	seb f3	x, 2, W	formusz-x
	Ž	•	
subf3	W, Y, 22	; 22 = y = 1	
Pe	22		
retsu			
end.			

Appendix A2. Double Length Add

```
**************
* FUNCTION DEF : dbladd
* AUTHOR: Al Lovrich 2/21/89
       lexas Instruments, Inc.
* Entry Conditions:
      Upon entry (r0,r1) contains (x,xx) and
      (r2,r3) contain (y,yy).
* Exit Conditions:
     Upon exit (r4,r5) contains (z,zz).
* Registers Affected:
     r0, r1, r2, r3, r4, r5, r6, r7
* Revision: Original
* Execution time: 25 cycles
**************
      .global dbladd
            ro
      .set
           r1
      .set
хx
           r 2
      .set
y
уу
      .set
            r3
Z
      .set
            r4
27
      .set
           r5
x1
      .set r6
y1
      .set r7
     .set r6
r
      .set r7
dbladd:
      abs f
             x,x1
      abs f
             y, y1
      comp f
             y1,x1
                                ; check for lx! > !y!
      ldflt
                                 ; if not, exchange (x,xx)
            x,x1
      idfit
             xx,y1
                                 ; and (y,yy)
      ldflt
             y,x
      Idflt
            vy.xx
      ldflt
            x1,y
      ldflt
           y1,yy
      addf3
            x,y,r
                                 ; r = x + y
      rnd
      subf3
            r,x,s
                                 ; s = x - r
      rnd
             S
      addf3
            y,s,s
                                 ; s = x - r + y
      rnd
      addf
             yy,s
                                 ; s = x - r + y + yy
      rnd
      addf
            XX,S
                                 ; s = x - r + y + yy + xx
      rnd
      addf3 s,r,z
                                 ; z = r + s
      subf3
             z,r,zz
                                ; zz ≕ r -- z
      rnd
             7.7
      addf3
             5,72,77
                                zz = r - z + s
      rnd
             ZZ
      retsu
      .end
```

Appendix A3. Single Length Multiply

subf3 by,y,ty ; $ty = y - by$ rnd ty	ampyf3 hx,by,p ; p = hx + hy andn single,p ; fl(s) is faithful	hx, ty, temp	anon single, remp ; rile) is rainrul mayf3 tx, hy, q ; q = tx = hy	single, q	addf3 q,temp,q ; q = hx + tx + hy		addf3 p,q,z ; z m p + q rnd z	•	54bf3 2,p,22 ; 22 = p - 2		2	mpyf3 tx,ty,temp ; temp = tx + ty	zz. tem. zz	=	-	· de ta	in the state of th		201						
		, (x,y)	1 (2. 22).		£4.														; p = x = constant ; fl(e) is faithful	4 × × ×	4 + d - x = x4 + d - x4 + d	t x = x = 1x	<pre>i p = y * constant fi(e) is faithful</pre>	d - K = kl s	; hy = y - p + p
+ FUNCTION DEF :walt12 +	ich 2/21/89 ments, Inc.	Conditions: Upon entry (r0,r1) contains (x,y)	Conditions: Then exit (r0.r1) contains (z.zz).	tedi	10, 11, 12, 13, 14, 15, 16, 17	.	* Execution Time: 35 Cycles		# 6	2 7	7	ნ.	2 %	. v	29	2 2			temp,x,p	ă,x, xă	w,e,w	м ,х, ф	tem, y, p single, p	P. y. hy	y, p, h
FUNCTION DEF : _mit12	AUTHOR: Al Levrich 2/21/89 Texas Instruments, Inc.	Entry Conditions: Upon entry	Exit Conditions: Unon exit	Registers Affected	5, 5,	Revisions Original	Execution Times 35 Cycles	Ladolg.	single .set	ž ž	ŧ.	ž.	ž	ž	ž i	įį	te t	_mult12:	£ \$	Sabf3	addf3	subf3	4	£ 1	eddf3

Appendix A4. Double Length Multiply

	***************************************	***************************************	37.		***************************************			;			
+ FUNCTION DEF :emit2			2		tons X a			Ž.	188 ,0	1 cc = x + 33 + 34 + 3 + cc	
•			· \$					Ē	e e		
# AUTHOR: Al Lewrich 2/21/89	ich 2/21/89		•		:						
+ Texas 1	Texas Instruments, Inc.		13	cub63	***	4 1 2		5			
# Entry Conditions:			7				•	577	:		
. Uben entr	Uben estry (r0.r1) cestains (x.v).	\$ (x.v).	1		3	4 4 4 5 5 6 5 6 6 6 6 6 6 6 6 6 6 6 6 6			2,5,2	20 + 2 = 2 1	
and (r2,1	and (r2,r3) contains (xx,yy).	٠,٠	Ę			4 . 4		2			
+ Exit Conditions	==		•								
tixa nodi)	Upon exit (r0,r1) contains (z,zz).	\$ (2,22).	125	subf3 hx	hx,x,tx	tx = x - hx		;			
* Registers Affecteds	ted		Ē					5411	:		
• 	10, 11, 12, 13, 14, 15, 16, 17							2	27'7'7	2 - 3 = 22 1	
•			â	mpyf3 te	temp.y.p	: P = y = constant		24463	:		
4 Algorithm used:			5						1,1,1	33 . 7 . 7 . 7 . 7	
+ mult12(x,	mit12(x, y, c, cc);		•		:			E	3		
+ × = 55	cc = x + yy + xx + y + cc1		98		w.v.hv						
100 + 0 = Z + CC	50		Ž	2				74 SE			
4 22 = C - 2 + CC	2 + CC;		3		2	0 + 0 - > = M		is S			
•			2	2			constant				
4 Revision Original			•					fort	4097	; constant = $2^{-(24-24/2)+1}$	
 Execution Time: 51 cycles 	51 cycles		1	Library Company	,	2 1 2 1		·end			
***************************************	****************	**************	ĺ			t () =) = m)					
islobel.	_malt2		Ĕ •								
single . set	_		•								
	ę		?	_		1 0 = tix + tiy					
	: 7		500		sıngle,p						
	: 2		•								
3	. 2		ì		hx, ty, temp	; temp = hx + ty					
	? 7		44	-	single, temp						
	. 2		à	_		1 4 = tx + hy					
	. 42				single, q						
	2 2		7	_		iq = hx + ty + tx + hy					
	2		P.	-							
	7		•								
, set	2		Î	_		; temp = tx + ty					
	! 2		- Fre								
	2		9	м	P,4,c	D+d= > 1					
¥.	: 2		Ę	-							
	Z		•								
tes.	7		3	_	ະ	1 cc # p - c					
. text			2								
_mult2:			- sedd			1 cc = b = c + d					
may f3	x, yy, temp0	t temp0 = x=yy									
anda			t ppe		ع ع	1 cc = p - c + d + tx + ty					
meyf3	y, xx, temp	temp = yfox	Ž								
-	_		•								
J		: temp = xeyy + yexx	 restore variables 	e les							
Par.			•								
hashf	į	i (xeyy + yexx)	break: pepf		ţ.	i xeyy + yexx					
	7		•								
* ************************************	8		* cc = x + yy + xx + y + cc	+ xx + y +	ម						
•			•								

Appendix A5. Double Length Divide

The flatting-point number v is stored in Rt. After the computation is completed, 1/v; a less stored in Rt. **Register used as inspect in Rt.		•		
e The Fleating-point number v is standing capitate and as input RI Register containing result RR, RR, RR, RR, RR, RR, RR, RR, RR, RR	AUTHUR: A) Lowrich 2/21/89	* c * x / y;		
Register used as input RI Register containing results RI Register containing register RI Register Co Register RI Register Co Register RI Register Right RI Register RI Regis	Texas Instruments, Inc.	+ + The floating-point	number v is s	tered in RI. After the computation is
Register used as input RI Register containing result RI Register containing result RI Register containing result RI Register containing result RI R		+ completed, 1/v is a	ilso stored in	7. F.
* Registers modified No., Ni., No., No., No., No., No., No., No., No	Upon entry (r0,r1) contains (x,y), and (r2,r3) contains (xx.yv).	Register used as in	8	1
Extract the exponent of V. pushf r1 pushf r1 pushf r1 push r2 ash -24,r0 ash -24,r0 the R1 = -127 - 1 = -128. Thus x1 the R2 = -127 - 1 = -128. Thus x1 the R3 = -127 - 1 = -128. Thus		* Negisters modified.	ž ž	2
Extract the exponent of V. pushf r1 pushf r1 pushf r1 push r2 and -24,00 and -24,00 and -24,00 the RI = -127 - 1 = -128. Thus xC reasonable, for a result, boundary reasonable fashion. x(0) formation given the exponent x(0) formation given the exponent way 73 r0,r1,r2 and single, 72 and single, 72 and single, 72 and r2, r2 and single, 72 and r2, r2 and r2, r2 and r2, r2 and r3, r3, r2 and r3, r3, r3 and	Upon exit (FU,FI) Contains (2,22). Jenisters Affertedi	•		•
pushf r1 peo r0 ash -24,r0 ash -24,r0 A few comments on boundary condit e fellowing xXXXX calculation xivel e vereflow and saturate since xXXX the xXXXX calculation xivel analises e reasonable feathion xXXXX for fermation given the exponent e xXXXX fermation given the exponent e xXXX fermation given the exponent e xXX fermation given the exponent e xXX fermation given the exponent e xXX fermation given the exponent e	10, 11, 12, 13, 14, 15, 16, 17		ت ت 3	; V is saved for later. ; The algorithm uses V = !v!.
pushf ri posh ro and -24,00 and -24,00 and -24,00 the Ri = 127 - 1 = -128. Thus xi the Ri = 127		***	;	
peep r0 ash -24,0 6 A few comments on beaundary condit 6 (1) to piel a struct's since the markins 1		# EXICACI ING EXPONEN		
ash -24,r0 ash -24,r0 A few comments on beaundary condit the fellowing xCO calculation yields the response of the result, beaundary a reasonable, he a result, beaundary a reasonable fashion. the fill -127 - 1 = -128. Thus xCO fermation given the exponent the fill -27 - 1 = -128. Thus the fill -2 - 12 - 12 - 128. Thus the fill -2 - 12 - 12 - 128. Thus the fill -2 - 12 - 128. Thu	•	jushq	7	
ash -24,r0 the delivating XCO calculation yields to wherefow and saturate since XCO to the RI = -122 - 1 = -128. Thes XI to yield second since the mantises to reasonable. As a result, beautises a reasonable fashion. the XCO fermation given the exponent ash 24,r0 subject to subject the subject to subject to subject the subject to subject t	cc = (x - n - nn + xx - c + λλ) / λ	£	የ	
# A few comments on beaumdary condit # fellowing xXIO calculation yields # everlow and saturate since xXIO # the RR = -127 - 1 = -128. Thus XX # the yield zero. Since the mantises # reasonable fashion. # XXIO fermation given the exponent # XXIO fermation given the exponent # XXIO fermation given the exponent # pash		43	- 74 ,70	; The 8 LSBs of R1 contain the exponent
# A few comments on beauchary condition yields # offilowing XCD calculation yields # the RR = -LZT - 1 = -LZD. Thes XC # to yield zero. Since the markissa # reasonable, fe a result, boundary # x reasonable feation # XCD fermation gives the exponent # XCD fermation gives the		•		· • • ·
# Fellowing XCIO calcutation yields # PRI = 127 - 1 = 128 Thus XCIO # THE RI = 127 - 1 = 128 Thus XCIO # Presconable feation. # XCIO formation given the exponent # XCIO formation given the exponent # PRI = 127 - 1 = 128 # PRI = 128 # PRI = 127 - 1 = 128 # PRI = 127 - 128 # PRI = 127 # PRI		* * ********	headery condi	itions 16 am -128 then v m 0. The
e veerflow and saturate since x(0) to the R = 1.27 - 1.28 / hies x to yield sere. Since the manifies a reasonable fashion. x(0) formation gives the exponent x(0) formation gives the exponent sabi 1,0 sabi 0,0 sabi 0,0 sabi 0,0 sabi 0,0 sabi 0,0 sabi 0,0 sabi 0,0,0 sabi 2,0 sabi 2,0,0 sabi		+ fellowing x[0] calc	culation yield	ds R1 = -128 - 1 = 127 and the algoritm
e to yield zero. Since the mantissa reasonable features Since the mantissa a reasonable featine the mantissa a reasonable featine to yield zero. Since the mantissa a reasonable featine e x(0) featatine gives the exponent set of the component set of the com	***************************************	+ everflow and satura	ate since x[0]	l is large. This seems reasonable. If 127,
* reasonable feathins * a reasonable feathins * x(0) feathins * nest in the exponent * x(0) feathins in the exponent * and in the exponent * and in the exponent * t		+ the RI = -127 - 1	= -128. Thus	x[0] = 0 and this will cause the algorithm
* x(0) fermation given the exponent * x(0) fermation for the exponent * x(0)		e to yield zero. Sind	Ce the mantis	sa or V is dimays between 1 (His is disc
* X(0) formation given the exponent ** **Morting** **Morting** ***Morting** **Morting** ***Morting** ***Mort		+ reasonable, HS & re	PSUIT, DOUBLE	ry tonestrons are managed accommissing in
* x(0) ferantine given the exponent * negi		*	i	
1.00 1.00		+ x[0] fermation give	en the exponer	nt of v.
nest 1, 0 sub 1, 1, 0 each 24, 0 push 10 push 10 push 10 each 24, 0 may 62, 12 may 73 10, 11, 12 may 73 10, 11, 12 may 73 10, 11, 12 may 74 12, 10 may 75 10, 11, 12 may 75 12, 10		•		
subi 1,10 subi 1,10 push 10 pu		negi	ያ	:
pash ro, pash ro pash ro pash ro and ro pash ro pas		ida.	٠ <u>.</u>	; New use have -e-1, the exponent of x[0].
pass 10 pass 1		653	2	
* New the iterations begin. * may/3 r0,r1,r2 and single,r2 sub/f r2,r0 may/f r2,r0 and single,r2 sub/f r2,r0 may/f r0,r1,r2 and single,r2 and r2,r2 rd r2 rd red r2 r2,r2 red r2 red r2 rd		100	2 9	. Mar 81 = x[0] = 1.0 + 244(-4-1).
* New the iterations begin. * mayer of the single, real			2	
## ## ## ## ## ## ## ## ## ## ## ## ##		* Now the iterations	begin.	
## ## ## ## ## ## ## ## ## ## ## ## ##		•		;
annel 21,0912, r. 20,172 annel 12,0,172 annel 13,1918, r. 20,172 annel 13,1918, r. 20,172 annel 13,1918, r. 20,172 annel 12,0,172 annel 12,0,172		mpy63	0,1,12	t R2 = v + x(0)
### 12,00 ##################################		2002 2003	Single, FZ	
#97 72,10 and 13mg1e,70 and 13mg1e,72 subf. 70,11,72 subf. 20,72 rd 72 mpyf 72,10			7,0,7 7,	1 N2 - 2.0 - 4 - XIO
and single, ro may f ro, ri, r2 and r2 rad r2 may f 2,00		\$^ **	72.0	1 R1 = x[1] = x[0] + (2.0 - v + x[0])
mays rd,rl,r2 and single,r2 subset 2.0,r2 red r2 mays r2,r0 mays r2,r0			single, ro	
angy f 10,11,12 and single,12 sabrif 2,0,12 rad r2 rad r2 ngy f 2,10		•	•	
and single, r2 subtr 2.0, r2 rad r2 rad r2 ray r2, r3		♣yf	10,11,12	; R2 = v = x[1]
2.0,72 rad 72 rady 72,70		spura .	single, r2	
2. July July July July July July July July	3 3 3 4		2.0,12	; K2 = 2.0 - v = x[]]
0.17.	× × ×		2 9	. 64 = 5(2) = 5(1) 4 (2.0 = 5.4 5(1))
	> enes :	a de la composición dela composición de la composición de la composición dela composición de la composición de la composición dela composición dela composición de la composic	2,7	' KI = XIZI = XIII = 1710 - A - XIII

R = x(3) = x(2) = 0 - v = x(2) R = x(3) = x(2) = (2.0 - v = x(2)) R = x(4) = x(3) = (2.0 - v = x(3)) R = x(4) = x(3) = (2.0 - v = x(3)) R = x(4) = x(3) = (2.0 - v = x(3)) R = x(4) = x(3) = (2.0 - v = x(3)) R = x(4) = x(3) = (2.0 - v = x(3)) R = x(4) = x(3) = (2.0 - v = x(3)) R = x(4) = x(3) = (2.0 - v = x(3)) R = x(4) = x(3) = (2.0 - v = x(3)) R = x(4) = x(3) = (2.0 - v = x(3)) R = x(4) = x(3) = (2.0 - v = x(3)) R = x(4) = (1.0 - v = x(4)) R = x(4) = (1.0 - v = x(4)) R = x(4) = (1.0 - v = x(4)) R = x(4) = (1.0 - v = x(4)) R = x(4) = x(4) = (1.0 - v = x(4)) R = x(4) = x(4) = (1.0 - v = x(4)) R = x(4) = x(4) = (1.0 - v = x(4)) R = x(4) = x(4) = (1.0 - v = x(4)) R = x(4) = x(4) = (1.0 - v = x(4)) R = x(4) = x(4) = x(4) = x(4) R = x(4) = x(4)	single, r2		<u>*</u>	Constant, tono	•
; R1 = x(3) = x(2) + (2.0 - v + x(2)) ; R2 = v + x(3) ; R2 = 2.0 - v + x(3) ; R1 = x(4) = x(3) = (2.0 - v + x(3)) ; This minimizes errer in the LSBs. • fermulation:)) + x(4) ; R2 = 1.0 - v*x(4) = 0.0.01 => 0 ; R2 = 1.0 - v*x(4) = 0.0.01 => 0 ; R2 = x(4) = (1.0 - v + x(4)) ; R2 = x(5) = (x(4)+(1.0 - v + x(4))) ; R2 = x(5) = (x(4)+(1.0 - v + x(4))) ; R2 = x(5) = (x(4)+(1.0 - v + x(4))) ; R2 = x(5) = (x(4)+(1.0 - v + x(4))) ; R2 = x(5) = (x(4)+(1.0 - v + x(4))) ; R2 = x(5) = (x(4)+(1.0 - v + x(4))) ; R2 = x(5) = (x(4)+(1.0 - v + x(4))) ; restore y ; restore y ; restore x ; save 1/y ; restore x ; save x ; save x		2.0 - v * x[2]	mpyf3	tos,x,p	1 p = x e constant
			ep es	single, p	
R2 = v = x(3)	5	x[3] = x[2] + (2.0 - v + x[2])			
RZ = v + x(3)	, mare, 10		Selbf3	p,x,hx	1 bx = x - p
		v + x[3]	E :	ă .	
RZ = 2.0 - v = x(3)				ж. •	thx = x - b + b
		2.0 - v + x(3]	2	ž	
			C3 Yan	2	: : : : : : : : : : : : : : : : : : : :
This minimizes ceree in the LSBs. + xtd + xtd + xtd + xtd		k[4] = x[3] + (2.0 - v + x[3])	Ž	ţ,ţ	¥ · × = × : •
		ainimizes error in the LSBs.	mpyf3	temp.y.p	p m y & constant
+ x(4)		•		single, p	
RZ = v * K[4] = 1.001 => 1 RZ = x.[4] = 1.001 => 0 RZ = X[4] = (1.0 - v * x(4)) RZ = x[5] = (x[4]+(1.0 - v * x(4)) Round since this is fellow by a MPYF. This sets condition flags. If v < 0, then Rl = -Rl save 1/y restore x save x save x save x	we use the formula! (v + x(4]))) + x(4]	ijent	Subf3	P. y. hy	; hy = y - p
			Padd63	į	:
RZ = 1.0 - vex(4) = 0.0.01 => 0 RZ = x(4) + (1.0 - v + x(4)) RZ = x(5) = (x(4)e(1.0 - (vex(4)))*x(4)] Phis sets condition flags. If v < 0, then Ri = -Ri save 1/y restore y restore y restore x save x save x		/ # x[4] = 1.001 => 1	2	1	1 my = y = p + p
RZ = x(4) = (1.0 - v + x(4)) RZ = x(5) = (x(4) = (1.0 - v + x(4)) Round since this is fellow by a MPPF. Round since this is fellow by a MPPF. I his sets condition flags. I for (0, then RI = -RI -RI save 1/y restore y restore y restore x the sex the sex		+ 0 \- 10 0 0 = [#] \		,	
R2 = x(4) + (1,0 - v + x(4)) R2 = x(5) = (x(4)+(1,0 - (vx(4)))+x(4) Round since this is fellow by a MPYF. If v < 0, then R1 = -R1 save 1/y restore y restore x save x save x save c			54bf3	hy, y, ty	; ty = y - hy
ROUND since this is fellow by a MPPF. Round since this is fellow by a MPPF. This sets condition flags. F v C 0, then R! = -R! save 1/y restore y restore x save x c = x + (1/y)		(4) # (1.0 - v + x[4])	ğ	ţ	•
# R2 = x(3) = (x(4)e(1,0-(vm(4)))+x(4) # Round since this is fellow by a MPYF. # This sets condition flags. # If v < 0, then R1 = -R1 # setw 1/y # restore y # restore y # restore x # setw x # c = x * (1/y)		•	5		
: Round since this is fellow by a MPYF. This sets condition flags. If V C 0, then RI = -RI save 1/y restore y restore y restore y restore y restore x save x save x		(5) = (x(4)+(1.0-(v+x(4))))+x(4)	ander s	nx, hy, p single, p	: P = hx + hy
This sets condition flags. If $h < 0$, then $Ri = -Ri$ I save $1/y$ I restore y I restore x I and $x = x + (1/y)$ I save $x = x + (1/y)$		since this is fellow by a MPYF.	;	: , , ,	
This sets condition flags. If v (0, then R! = -R! I save 1/y I restore y I restore x I deve x I c = x * (1/y)			andn	in, ty, temp	; temp = hx + ty
1,72 1,18 1,19 1,19 1,19 1,19 1,19 1,19 1,19	13 indicine.		may 63	tx, hy, a	. o = tx + hv
1,73	2		- Fug	single, q	
			addf3	g, temp, g	hx + tv + tx + hv
1 save 1/y 1 save 1/y 1 save 1/y 2 save 1/y 3 save x 3 save x 3 save x 3 save x 3 save c 3		0, then RI = -RI	Ē	-	
			Derform tx + tv o	meration and ct.	a the secult is then
payed prestore y			optimize use of r	egisters on the	device.
		•	Single	4	
			5	single tem	transfer to a th
	101611	.	addf3	9.0.4	
y, icmxe(1/y) subf3 u,p,uu ngle,x icmxe(1/y) au addf q,uu nhd uu inhd	Save x	×	Pu		
x 1 C = x + (1/y)		•	enh63	:	:
and from the state of the state		· (1/y)	2	5,	9 1 d = 20
The en	χ',		- Add	5	*********
			Pac	3	
	. Save C				

mult12(c, y, u, uu)

```
; cc = ( x - u - uu + xx - c + yy ) / y
                                                                                         ; cc = x - u - uu + xx - c + yy
                                                                                                                                                                                                                                 ; constant = 2^{\circ}(24-24/2)+1
                                                      XX + M - M - X = 33 1
                                   10 - n - x = 33 t
                                                                                                                                                                                         32 + C - 2 + CC
; restore 1/y
; restore c
; restore x
; cc = x - u
                                                                    ; restore yy
; c * yy
                                                ; restore xx
                                                                                                                                         3 2 = C + CC
                                                                                                                                                                            2 - 3 = 22 :
                                                                           c, temp
single, temp
                                                                                        temp,cc
cc
y1,cc
single,cc
                    u, temp, cc
cc
                                                temp
temp,cc
                                                                                                                                         c,cc,2
2
                                                                                                                                                                            2,C,22
22
CC,22
                                                                                                                                                                                                                                 4097
addf3
                                                                                                                                                                                                                                 float.
                                                                                                                                                              DD + Z - D = ZZ +
                                                                                                                                                                                                             retsu
data
                                                                                                                                                                           22 + 2 = 2 + 66
                                                                                                                                                                                                                           constant:
```

Appendix A6. Double Length Square Root

and 120,10 and 120,10 post 11 and 120,11 and 120,11 and 120,11 and 120,11 and 120,11 bearate v/2. and 120,11			save x
and single, roll formation given the exponent may in 1 and 12 and			; add a rounding bit in the exponent
ash 25,r1 ash 28,r1 ash 28,r1 ash 28,r1 ash 28,r1 ash 11 beet 11,r1,r2 and single,r2 and single,r2 and single,r1 and single,r2 and single,r1 and single,r2			
* X(0) femation given the exponent * (1)			
megi ni ni ash 24,11 ash 2			; The 8 LSBs of R1 contain 1/2 the
nest n. 1 and 24,r1 posts n. 1 e. Generate v/2. andre single, r0 andre single, r0 andre single, r2 andre sin		* x(0) fermation gives the expen	2 to 10 to 1
0.00 0.00 0.00		•	
passh ri pas			
# Generate v/2. # Generate v/2. # How the iterations begin. # How the iteration is ingle, r2 and iteration. # # # How the iteration is ingle, r2 and iteration in iteration is ingle, r2 and iteration in i			
Benerate v/2. By f 0.25,r0 By f 0.25,r0 By f 1,r1,r2 By f 1,r2 By			(C)) and (O) (A) (A) (C)
# Generate v/2. # New the iterations begin. # New the iterations begin. # White iterations begin.			17/4 \ COT - FON - T. AM. I
why 0.25,r0 when the iterations begin. why isingle,r2 and single,r2 and single,r1 and single,r2		* Generate v/2.	
and single, r2 and r2 and r3 and r3			
and single, ro where the iterations begin. why ri, ri, r2 and single, r2 and r2 and r2 and r3 why ri, r1, r2 and r3 and single, r2 and r3 a			
where iterations begin. why ri,ri,r2 and single,r2 and r2 why ri,ri,r2 and ringle,r1 why ri,ri,r2 and single,r2 and single,r1 why ri,ri,r2 and ringle,r1 why ri,ri,r2 and single,r1 why ri,ri,r2 and single,r1 and single,r2 and single,r1 why ri,ri,r2 and single,r1 and single,r2 and single,r1 why ri,ri,r2 and single,r1 and single,r2 and single,r1 and single,r2			
### 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,		* New the iterations begin.	
anden single, r2 anden single, r2 ander single, r2 rud r2, rud r2, ander single, r1 ander single, r1 ander single, r1 and r2, ander r1, r1, r2 ander single, r1 and r2, r1 and r3, r3, r2, r1 and r3, r3, r1 and r3, r3, r2, r1 and r3, r1, r1, r2 and r3, r3, r1 and r3, r3, r1 and r3, r3, r1 and r3, r1 and r3, r1, r1, r2 and r3, r3, r1 and r3, r3, r2 and r3,			; r2 = x[0] + x[0]
## ## ## ## ## ## ## ## ## ## ## ## ##	**************************************		
subject 15,72 may f 2,71 may f 2,71 may f 1,71,72 may f 1,71,7			
samer 1.5,12 rad 72 may 7			
## ## ## ## ## ## ## ## ## ## ## ## ##			1. F = 1.5 = (2/V) = XIU] + XIU]
and single, r.l. and single, r.l. and r.l. subtrible, r.l. and single, r.l. and r)-4(2/~) - 5) + [() = [1] - 1 .
why fritting and single, 72 and single, 72 and single, 72 and for 1.5, 72 and for 1.5, 72 and for 1.5, 72 and single, 71 and single, 72 and single, 71 and single, 72 and s			
and single, r2 and single, r2 and single, r2 and single, r1 and single, r1 and single, r1 and single, r2 and single, r1 and r2 and single, r2			:
and single, r2 subset 15, r2 subset 15, r2 subset 12, r2 s			; r2 = x(1] + x(1]
anden single, r2 rad r2 rad r2 mayy r2, r1 mayy r2, r1 mayy r2, r1 mayy r1, r1, r2 mayy r1, r1, r2 mayy r1, r2, r2 mayh r1, r1, r2 mayh r2, r1 mayh r2, r2 mayh r2, r2 mayh r2, r2 mayh r2, r2 mayh r3, r2 mayh r3			. r2 = (v/2) + x[1] + x[1]
subbř 11,5,72 mnd 12,72 mpyř 72,11 mpyř 71,17,172 mpyř 11,71,172 mpyř 11,71,72 mpyř 11,71,72 mpyř 12,17 mpyř 12,17 mpyř 11,71,72 mpyř 11,72 mp			
rnd r2 mpyf r2,r1 andn single,r1 mpyf r1,r1,r2 andn single,r2 andn single,r2 andr single,r2 subrf 1.5,r2 rnd r2,r1 mpyf r1,r1,r2 andn single,r1 andn single,r1 andn single,r2			
mayy r2, r1 and single, r1 mayy r1, r1, r2 and single, r2 mayy r2, r1 mayy r2, r1 mayy r2, r1 mayy r2, r1 mayy r1, r1, r2 mayy r1, r2, r1 mayy r1, r2 mayy r1, r1, r2 mayy			
andn single, rl. mayf rl, rl, r2 andn single, r2 andn single, r2 andn single, r2 andn single, r2 and r2, rl mayf r2, rl andn single, r2			; r1 = x[2] = x[1] + (1.5 - (v/2)#x[1]#x
mpyf ri,ri,r2 andn single,r2 andn single,r2 andn single,r2 rnd r2 rnd r2,ri andn single,r1 andn single,r2 andn single,r2 andn single,r2 andn single,r2			
and single, r2 and r1, r1, r2, r2 and r2, r3, r2 and r2, r1 and r2, r1 and r1, r1, r2 and single, r2			
may from a single, r2 and single, r2 sub-from r2 r1 may from r2, r1 may from r2, r1 and r2 may from r2, r1 and r2 may from r2, r1 and r2 may from r2, r1 and r2, r2 and r3, r3, r2 and single, r2 and single, r2 and single, r2			1 LZ = XLZJ + XLZJ
anden single, r.2 suber 1.5, r.2 rod r.2 mayr r.2, r.1 mayr r.2, r.1 anden single, r.1 mayr r.1, r.1, r.2 mayr r.1, r.1, r.2 mayr r.1, r.1, r.2 mayr r.1, r.2, r.2 mayr r.1, r.2, r.3 mayr r.3, r.3, r.3, r.3, r.3, r.3, r.3, r.3			· r2 = (v/2) + v[2] + v[2]
subrf 11,5,72 rnd 72,71 supyr 72,71 and single,71 and single,72 and single,72 and single,72			
red 72 mays 72,51 andn single,11 * mays 11,51,52 andn single,72 andn single,72			; r2 = 1.5 - (v/2) + x[2] + x[2]
mayyi r2,rl anda single,rl anda single,r2 mayyi r1,rl,r2 mayyi r2,rl,r2 mayyi r3,rl,r2 anda single,r2 anda single,r2			
and Single, FI and Single, FZ and Single, FZ and Single, FZ			; r1 = x[3] = x[2] + (1.5 - (v/2)+
mapy ri,ri,r2 andm single,r2 may? ro,r2 andm single,r2			
anden single, r2 appyf r0, r2 anden sinele, r2			; r2 = x(3) + x(3)
single.r2	; return if number non-positive		
			FST - FST - (7/1) - 7: 1

	and $\sin \theta_{ij}$		•	a manform to be amounting and store the result in temp.	e This is to optimize use of registers on the device.	•	mpyf3 tx,ty,temp ; temp = tx = ty andn single,temp	_	, pu	n - d = nn t nn d'n Eyens	3	b+n-d = nn · nn · ppe	an pu	1 / 10 O O O O O O O O O O O O O O O O O O		J		page of temptor to the page of		ម	<u>.</u>	THE CC CC X - I - XX - XX - II - XX - XX -		DAMES I DESCRIPTION OF THE PARTY OF THE PART	int number v is st	e completed, I/v is also stored in M.	e Register used as inputi R2	+ Registers modified NO, NJ, NZ, NZ A Basistan combinion menulty NZ		ldf r2,r3 ; v is saved for later. absf r2 ; The algorithm uses v = lvi.	•	
; r2 = 1.5 - (v/2) + x[3] + x[3]	$_1$ F1 = x[4] = x[3] + (1.5 - (v/2)+x[3]*x	; r2 = x(4) + x(4)	:	$\frac{1}{2}$ r2 = (v/2) + x[4] + x[4]	1 r2 = 1.5 - (v/2) + x[4] + x[4]		$\frac{1}{2}$ r! = x[5] = x[4] + (1.5 - (v/2)*x[4]*x			; sqrt(v) from sqrt(v**(-1))			; save c = sqrt(x)	; get ready for multiplication				י א א נפשארפונו		1 px = x = b	:	g + d X X 1		- TX = X = EX	: B = V & Constant		1 hy = y - p	4 4 1 2 3		t ty = y - My		; p = hx + hy
1.5,12	r2,r1 single,r1	11,11,12	single, r2	70, 12	1.5, 1.2	2	12,11	single, ri	5,5	5,6	single, ro		×	x,y	~		Constant, temp	single p		þ,x,hx	¥ .	ž Ž		χ, χ, χ	tee.v.e	single,p	P.y.hy	1	i A	by, y, ty	4	tx, by, e
and bec	e e	•	enda	j.	sebré	Ę	a yé	andn	ž	¥.	sud	+ Cario com Cario	pushf	±	• mult12(c, c, u, wu)		<u>*</u>		į	subf3	2		!	Pages	mvf3	up up	Sub f3	784 84463	Į	subf3	Į	may 63

### 1.00	1			
### 1 The 8 Libbs of 80 contain the exponent ### 17.0	4 6 6 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	subré	1.0,0.	; R1 = 1.0 - v + x[4] = 0.001 => 0
1 or V audit		A P	6 <u>1</u>	; R1 = x(4) + (1.0 - v + x(4))
### ### ##############################	11-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1	ende	single, ro	
1 Now we have -w-1, the exponent of x(0) 1 Now we have -w-1, the exponent of x(0) 1 Now we have -w-1, the exponent of x(0) 1 Now we have -w-1, the exponent of x(0) 1 Now No 1 No		***		: No = x[5] = (x[4]+(1.0-(vex[4])))+x[4]
1 Mea se have -r-1, the exponent of x(0] 1 may 1,2,10 1,2,10 1 may 1,2,10 1 m		Z	11,12	; Round since this is followed by a HPPF
1 Now 80 1 Now 100 1,0 + 2 cet(-4-1)		•		
1 May NO = x(0) = 1.0 e 2ee(-q-1).		+ New the case of v <	0 is handled	
1 Now NO = x(0) = 1.0 + 2***(-4-1), 167 16		•		
14fn 10,12		## 1# 1# 1# 1# 1# 1# 1# 1# 1# 1# 1# 1# 1	5, 5, 5, 5,	; This sets condition flags.
		ldfn	79,12	; If v < 0, then R2 = -R2
	11,72,70 2.0,70 7.0,71 81mgle,70 7.0,71 81mgle,71 7.0,71 81mgle,70 7.0,71 81mgle,70 7.0,71 81mgle,70 7.0,71 81mgle,70 7.1,72,70 81mgle,70 7.0,71 81mgle,70 7.0,71 81mgle,70 7.0,71 81mgle,70 7.0,71 81mgle,70 7.0,71	+ restere cariables		
Single, rO	single, 70 2.0, 70 7.0, 71 8.1919, 71 8.1919, 70 8.1919			
20,r0 ; R1 = 2.0 - v = x(01)	2.0, r0 10, r1 11, r2, r0 11, r2, r0 12, r2, r0 12, r3, r3, r3, r3, r3, r3, r3, r3, r3, r3	jdod	•	i restore c
Part	70,011 single,r1 r1,r2,r0 single,r0 r0,r1 r2,r0 single,r0 r0,r1 r2,r0 r1,r2,r0		. 8	: restore co
of, I. 1 ND = X(1) = X(0) = (2.0 - v = X(0)) andmost single, cr. single, cr. ri, P.2.0 ri, P.2.0 andmost single, cr. andmost single, cr. 2.0, r0 ri, P.2.0 andmost single, cr. andmost single, cr. 2.0, r0 ri, R.1 = 2.0 - v = X(1) andmost single, cr. andmost single, cr. ri, r2, r0 ri, R.1 = 2.0 - v = X(2) andmost single, cr. andmost single, cr. r0, r0 r1, r2, r0 r1, r2, r0 r1, r2, r0 r1, r2, r0 r0, r1 r1, r2, r0 r1, r2, r0 r1, r2, r0 r1, r2, r0 r1, r2, r0 r1, r2, r0 r1, r2, r0 r1, r2, r0 r1, r2, r0 r1, r2, r0 r1, r2, r0 r1, r2, r0 r1, r2, r0 r1, r2, r0 r0, r0 r1, r2, r0 r1, r2, r0 r1, r2, r0 r1, r2, r0 r0, r0 r1, r2, r0 r1, r2, r0 r1, r2, r0 r1, r2, r0 r0, r0 r1, r2, r0 r1, r2, r0 r1, r2, r0 r1, r2, r0	1,0,11 1,7,70 1,1,7,70 2,0,10 1,1,7,70 1,1,70 1,1,70 1,1,70 1,1,70 1,1,70 1,1,70 1,1,70 1,1,70 1,1,70		0.5,cc	1 CC = (X - 88 - 87) + 0*2
	single, rl rl, r2, ro 2,0, ro 70, rl 8,0, rl 8,0, rl 70, rl 70, ro 8,0, rl 8,0, rl 8,0		single, cc	
	11,72,70 20,70 70,71 81991e,71 71,72,70 81991e,71 70,71 81991e,70 71,72,70 81991e,70 81991e,70 81991e,70 81991e,70		12,00	1 CC = (x - n - m + xx) + 0.5 / c
	single, to 20,000 to 100 to 10	wa	single, cc	
2.0,r0 1. R1 = 2.0 - v + x(1) 1. r0 1. r0 1. r0 1. r0 1. r0 1. r0 2.0,r0 2.0,r0 1. R1 = 2.0 - v + x(2) 2.0,r0 1. R1 = 2.0 - v + x(2) 2.0,r0 2.	2.0,00 2.0,00 3.0,00	•		
1	7.0, 1.1 1.1, 2, 70 1.1, 2, 70 2.0, 70 1.1, 72, 70 1.1, 72, 70 1.1, 72, 70 1.1, 72, 70 1.1, 72, 70 1.2, 70 1.0, 70 1.0	33 + 3 = 2 ·		
	10,11 11,72,10 11,72,10 2,0,10 0,11 11,72,10 11,72,10 11,72,10 11,72,10 11,72,10 11,72,10 11,72,10 11,72,10 11,72,10			
single,rl * zz = c - z + cc rl,r2,r0 ; Rl = v = x(2) * subf z; temp; zz single,r0 ; Rl = 2.0 - v = x(2) * subf z; temp; zz r0 ; Rl = 2.0 - v = x(2) * subf z; temp; zz r0,r0 ; Rl = 2.0 - v = x(2) * subf z; temp; zz r0,r1 ; R0 = x(3) = x(2) = (2.0 - v = x(2)) * r1 single,r0 * Rl = v + x(3) * retsus data single,r0 * Rl = 2.0 - v + x(3) * retsus data r0 * Rl = 2.0 - v + x(3) * risus data r0 * Rl = 2.0 - v + x(3) * risus data r0 * r1 * r1	single, r1 2.0, r0 7, r2, r0 2.0, r0 7, r2, r0 8ingle, r1 8ingle, r1 7, r2, r0 8ingle, r0 7, r2, r0 8ingle, r0 7, r2, r0 8ingle, r0 70, r0 80, r	22.00		1 2 = C + CC
	ri,r2,r0 single,r0 20,r0 r0,r1 single,r1 r1,r2,r0 single,r0 r0,r1	•	•	
	single, ro 2.0, ro ro, ro ro, ro, ro single, ro ro, ro, ro ro, ro ro, ro	25 + Z - 3 = ZZ +		
2.0,r0 i Ri = 2.0 - v * x(2) red 2.0, red 3.0, r	2.0,70 2.0,70 0,71 5.10gle,71 7.1,72,70 2.0,70 7.0	•		
Fig. 1 Fig. 1 Fig. 2 Fig. 3 Fig. 3 Fig. 3	7.05 11.05 1		z, temp, zz	1 22 = C - 2
10	20,7 1,12,7 2,0,70 2,0,70 2,0,70	200	22	
single,rl	single, r1 r1, r2, r0 single, r0 2.0, r0 r0 r0 r1	497	cc, 22	1 22 = C - 2 + CC
ri, r2, r0 ; RI = v + x(3) deta simplo, r0 deta 2.0, r0 ; RI = 2.0 - v + x(3) constant: float 4097 r0 i R0 = x(4) = x(3) = (2.0 - v + x(3)) end	11,72,70 single,70 2.0,70 70	2	×	
simple, ro 2.0, ro ro ro ro, ri RO = x(4] = x(3) = (2.0 - v * x(3))	2.0,0 2.0,0 0 10	retsu		
2.0,r0 ; RI = 2.0 - v + x(3) r0 r0,r1 ; R0 = x(4) = x(3) + (2.0 - v + x(3)) .end	5.0.5 5.0.5			
r_0 , $r_1 = r_1 = r_1 = r_2 = r_2 = r_3 = r_3$	2 2		1604	: constant = 2^(24-24/2)+1
		pua.		
	•			

* For the last iteration we use the formulations * x(S) = (x(4) + (1.0 - (v + x(4)))) + x(4) *

Appendix B

Appendix B1. Change Two Single-Precision Numbers to One Double-Precision Result

```
/s expm-128 corresponds to 0. expm-127 is denormalized in isset
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  if (exp > 127) return(0); /# tee large number; return error #/
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              mantissa = (mantissa & OxOO7fffff) ! (sign << 8) ! (exp << 23);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            /* make exponent 127-excess and return ieee number #/
                                                                                                                                                                                                                                                                                                                                                                                      /* add implied bit and sign-extend mantissa */
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   /* convert mantissa to sign-magnitude +/
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     /* adjust mantissa if it was -2.0 */
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           if (sign) mantissa = -mantissa;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                if (mantissa == 0x01000000)(
                                                                                                                                                                                                                                                                                                                              if (exp <= -127) return(0);
ang int c3Otee(long int x)
                                                                                                                                                                                                                                                                                                                                                                                                                                                    mantissa = x & 0x007fffff;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              mantissa != 0xff000000;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      mantissa := 0x00800000;
                                                          long int mantissa, sign;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           mantissa = 0x00800000g
                                                                                                                                                  sign = x & 0x00800000;
                                                                                                                                                                                                                                                                       represent it as 0. #/
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               return(mentissa);
                                                                                                                                                                               xp = x >> 24;
                                                                                       ong int exp;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    exp += 127:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    if (sien)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  /s C3070E -- routine to convert from a c30 floating point number to a
         -- Program to operate on two single-precision numbers
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               number in ieee format. Both input and output in hex. */
                                       in C30 format and produce a double-precision result #/
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           printf("Add(1), Sub(2), May(3), Div(4), Sqrt(5): ");
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      printf("\n\nType 0 to exit, else continue : ");
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       printf("\n\nType in C30 bex resulti\n");
                                                                                                                                                                                                                                                                                                                                                                                                                      printf("Type two C30 hex numbers:\n");
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      ) while (operation() !! operation();
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    x = (long double)(#(float #)(fx(1));
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      x = (long double)(f(float f)(kx1))_1
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 y = (long demble)(#(fleat #)(Ey1));
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 y = (leng double)(*(float *)(ky1))_3
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               if (operation == 5) z = sqrt(x);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 if (operation == 1) z = x + y;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           if (operation = 2) z = x - y_1
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        if (operation == 3) z = x + y_1
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  if (operation == 4) z = x / y_1
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            printf("\az = 2.18Lg", z);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      printf("\nz = 1.18Lg", z);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         scanf("Zd", hoperation);
                                                                                                                                                                                                                                                                                                           long int c30toe(long int);
                                                                                                                                                                                                                   long double x, y, z;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   scanf ("24", &i);
                                                                                                                                                                                                                                                                              int i, operation;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    y1 = c30ter(y1);
                                                                                                 linclude (stdio.h)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    scanf("XX", bx1);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               scanf("XX", by1);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       y1 = c30tee(y1);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  scanf("ZI", bx1);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            printf("zz = ");
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            scanf("ZX", by1);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             x1 = (30tee(x1);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 ) while (i != 0);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               x1 = c30tee(x1);
                                                                    linclude (meth.h)
                                                                                                                                                                                                                                                long int xl, yl;
                                                                                                                                                                                                                                                                                                                                                                                                                                                       printf("x = ");
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  printf("y = ");
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 printf("z = "):
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               2 = x + V:
                                                                                                                                                        ()uin
                                                                                                                                                                                                                                                                                                                                                                  ij
```

Appendix B2. Change Two Double-Precision Numbers to One **Double-Precision Result**

```
/* C30TOE -- routine to convert from a c30 floating point number to a
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   /* exp=-128 corresponds to 0. exp=-127 is denormalized in ieee!
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        if (exp > 127) return(0); /# too large number; return error #/
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              Mantissa = (mantissa & 0x007fffff) | (sign << 8) | (exp << 23);
                                                                                                                                                                                                              Number in ieee format. Both input and output in hex. e/
                                       printf("\n\nType 0 to exit, else centinue : ");
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            /* make exponent 127-excess and return ieee number
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  /* add implied bit and sign-extend mantissa +/
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          /* convert mantissa to sign-magnitude #/
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        /* adjust mantissa if it was -2.0 */
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           if (sign) mantissa = -mantissa;
printf("\nz = 1.18Lg", z);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             if (mantissa == 0x01000000)(
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    if (exp <= -127) return(0);
                                                                                                                                                                                                                                                                               eng int c30toe(long int x)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   mantissa = x & 0x007fffff;
                                                                                                                                                                                                                                                                                                                                            long int mantissa, sign;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    mantissa != 0xff000000;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            mantissa := 0x00800000;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          mantissa = 0x00800000_1
                                                                                                                                                                                                                                                                                                                                                                                                                                                 sign = x & 0x0080000;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   represent it as 0. #/
                                                                           scanf ("24", bi);
                                                                                                        ) while (i != 0);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               return(mantissa);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  exp = x >> 24;
                                                                                                                                                                                                                                                                                                                                                                                    long int exp;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              exp += 127;
```

printf("Add(1), Sub(2), Mpy(3), Div(4), Sqrt(5): ");

> while (operation(1 !! operation)5);

scanf("%d", Moperation);

if (operation == 1) $z = x + y_3$ if (operation == 2) $z = x - y_1$ if (operation == 3) z = x # y; if (operation == 4) $z = x / y_1$

x = (long double)(#(float #)(&x1)) +y = (long double)(#(float #)(Ay1)) +

x1 = c30tee(xx1); yy1 = c30tee(yy1);

y1 = c30toe(y1);

scanf("ZX", byy1);

scanf("XX", 8y1); printf("yy = "); x1 = c30toe(x1); (long double)(*(float *)(&cx1)); (long double)(*(float #)(kyy1)); printf("\n\nType in C30 hex result:\n");

if (operation == 5) z = sqrt(x)

printf("\nz = %.18Lg", z);

x = (long double)(*(float *)(tx1));y = (long double)(#(float #)(Ly1));

y1 = c30toe(y1);

2 = x + y;

printf("zz = ");

scanf("XX", &x1); scanf("XX", by1); x1 = c30tee(x1);

printf("z = ");

```
/* C3008L2 -- Program to operate on two double-precision numbers
                                      in C30 format and produce a double-precision result #/
                                                                                Pinclude (math.h)
```

binclude (stdio.b)

min()

printf("Type two C30 hex numbers:\n");

Ē

scanf("XX",box1);

printf("y = ");

printf("xx = ");

scanf("XX", bx1);

printf("x = ");

long int x1, y1, xx1, yy1; long int c3Otoe(long int);

lang double x, y, z; int i, operation;