DECUS

PROGRAM LIBRARY

DECUS NO.	12-7
TITLE	DBLFLT - DOUBLE FLOAT MATHEMATICAL ROUTINES
AUTHOR	Donald A. Overton, Ph.D.
COMPANY	Eastern Pennsylvania Psychiatric Institute Philadelphia, Pennsylvania
DATE	Submitted: July 10, 1970
SOURCE LANGUAGE	LAP6-DIAL

DBLFLT - DOUBLE FLOAT MATHEMATICAL ROUTINES

DECUS Program Library Write-up DECUS No. 12-7

Introduction

The DBLFLT Mathematical Routines tape contains various LINC mode mathematical subroutines which operate on a floating point number composed of a 24-bit mantissa and a 12-bit exponent. All these routines use or are derived from the program DBLFLT which was written for the classic LINC computer by Michael McDonald.

The source entitled DBLFLT is identical to the DBLFLT used with LAP4 and LAP6 in the classic LINC and LINC-8 computers. It has not been moved or altered in any way except that the number-letter symbolic tags used by LAP4 and LAP6 have been reversed to form letter-number symbolic addresses which are acceptable to LAP6-DIAL. The reader is referred to the document entitled Double Precision Point System (DEC-L8-SFAA-D) for a discussion of the use of this program.

The source entitled DBLFLT1 is a slightly altered version of DBLFLT in which the utilization of certain subroutines has been made slightly simpler than in the original DBLFLT. A document describing the use of DBLFLT1 is attached. Since DBLFLT1 uses mnemonic symbols for its entry points, users will probably find it easier to use than the original DBLFLT when developing new programs. DBLFLT1S is identical to DBLFLT1 except that comments have been deleted from this program source to make it shorter and more manageable.

DBLFLT3 is a package of mathematical subroutines built around DBLFLT1 allowing a more extended set of mathematical operations. A document describing the use of this program is attached. This package of subroutines was originally put together by Paul Sullivan and w previously released for the LINC-8 as DECUS No. L-68. DBLFLT3S is identical to DBLFLT except that comments have been deleted to shorten the source. DBL3GO, DBL3GOX, DBL3GOY and DBL3GOZ are programs used in conjunction with DBLFLT3 and are described in the DBLFLT3 document.

Obviously all credit for these programs belongs to the original authors. However, as the current author has somewhat revised both DBLFLT1 and DBLFLT3 for the PDP-12, questions concerning the operation of these programs should be initially directed to the current author: Donald A. Overton, Eastern Pennsylvania Psychiatric Institute, 3300 Henry Avenue, Philadelphia, Pennsylvania 19129.

Source Title: DBLFLT1[1]

Source Language: LAP6-DIAL

Computer: PDP-12A[2]

Author: Michael McDonald[3]
Biomedical Computer Laboratory
Washington University
St. Louis, Missouri

ABSTRACT

DBLFLT1 is a double precision floating point package which operates in the LINC mode and uses two quarters of the current instruction field. All its operations are performed on a standard format number which has a double precision mantissa and a 12-bit exponent. DBLFLT1 provides the basic mathematical operations (add, subtract, multiply, divide, absolute value, complement) as well as routines to convert between fixed point and floating point format and to move floating point numbers from place to place in core. DBLFLT1 is the central mathematical program used in DBLFLT3 and this document will reference sections of the DBLFLT3 document wherever possible to avoid duplication.

Utilization

DBLFLT1 must be placed in the current instruction field where it occupies 776 octal locations and uses index registers 1-5. Access to its routines is obtained via JMPS to the appropriate entry points. Most entry jumps are followed by a list of 1-3 arguments which specify the locations of the numbers to be operated on. The numbers so referenced may be located in either the current instruction or data fields.

Notation

Same as in DBLFLT3 document.

Floating Point Format

Same as in DBLFLT3 document.

---

1. DBLFLT1 is virtually identical to the earlier program DBLFLT which was circulated in the LAP4 and LAP6 languages.

2. DBLFLT1 will operate in the PDP-12C. However, at present the source and binary are only available on LINC tape.

3. This document was prepared by D. A. Overton, Eastern Pennsylvania Psychiatric Institute, 3300 Henry Avenue, Philadelphia, Pennsylvania 19129, who is also responsible for the revisions of DBLFLT contained in DBLFLT1.

Precision

Same as in DBLFLT3 document.

Argumented Operations

The format for argumented DBLFLT1 operations is identical to that for argumented DBLFLT3 operations except that the JMP is made directly to the desired DBLFLT1 entry point instead of to the program DBL3GO.

The utilization of index registers is sufficiently different in DBLFLT1 and DBLFLT3 so that it will be described separately here for DBLFLT1. An argument may be indirectly addressed by placing the address of its exponent in an index register in the current instruction field and using the index register as an argument in the calling sequence. Allowable index registers are 6-17. The contents of the index register will be indexed by three immediately after its use by DBLFLT1.

Consider the following instructions, where A is at LOC1 and B is at LOC1+3.

```
SET I 10
LOC1
JMP SUBT          /Enter DBLFLT1
10
-10
returns here
```

B will be subtracted from A and the result stored in the FAC. When control is returned to the user program, index register 10 will contain LOC1+6.

As another example, the following sequence will add 100 DBLFLT numbers starting at addre 1000 in segment 0 and leave the result in the FAC.

```
LDA I             /Set FAC = Ø
4000
STC FAC
STC FAC+1
STC FAC+2
SET I 7           /Points to data
2\1000
SET I 10          /100₁₀ data values
-144
LDF Ø
JMP ADDT
-7
XSK I 1Ø          /Done?
JMP.-3
continues here
```

Note that when an index register is used to reference a DBLFLT number, it is set equal to the address of the exponent of the DBLFLT number, not one address ahead of the exponent.

Halts

There are two halts indigenous to DBLFLT1:

a) Location DIVIDE+7 - A halt in this location means that an attempt was made to divide by zero.

b) Location DF6-1 - A halt in this location indicates an overflow condition, i.e., that the exponent of the result of some operation exceeds +3777.

If a halt occurs at one of these DBLFLT1 locations, inspection of location DT7 will show from where the call originated.

Storage Locations

The DBLFLT1 routine includes locations for storage of three DBLFLT format numbers. The DBLFLT locations designated ARG1 and ARG2 need not generally be referenced by user programs. The third DBLFLT storage location is called the Floating Point Accumulator (FAC). It is implicitly referenced by most DBLFLT operations which are followed by only one or two arguments, and the user will frequently wish to reference this location. Additional storage locations may be assigned by the user in either the current data or instruction fields.

Constants

Several constants in 12-bit fixed point format are located within the DBLFLT1 subroutine, and hence within the current instruction field. These may be referenced by user programs using the direct class instruction ADD.

+0	DF6-1
-0	COM1+1
+1	DK7
-1	DL7
3	DS7-1
5	QAC12
11_8	ADDT
16_8	DD6-1
17_8	ADDT+4
2000_8	ADDT+1
6000_8	DC6-1

Assembly

DBLFLT1 fills two quarters of the current instruction field. The calling program must be placed in the remaining part of the current instruction field and assembled along with DBLFLT1 as a single source. The source DBLFLT1S is identical to DBLFLT1 except that comments have been deleted to shorten the source.

Modifying DBLFLT1

The entire source for DBLFLT1 may be relocated within the instruction field if desired, as long as it does not overlay the index registers which it uses. However, it is wise not to modify the relative positions of the different DBLFLT1 subroutines, i.e., do not insert user subroutines or storage locations between DBLFLT1 subroutines.

Modifying DBLFLT Programs to Use DBLFLT1

Programs written to operate with DBLFLT can be used with DBLFLT1 providing the following modifications are made:

1. All jumps to DBLFLT entry points must be changed to reflect the entry point symbolic addresses of DBLFLT1. All DBLFLT tags have been changed in DBLFLT1. The entry points are provided with mnemonic symbolic addresses. The remaining symbolic addresses (which are generally not of interest to user programs) have been prefixed with the letter D. Note that the absolute addresses of DBLFLT1 entry points are not identical to those of DBLFLT entry points.

2. All non-standard entries to DBLFLT and all addresses calculated relative to DBLFLT tags must be checked to make sure they still reference the appropriate instructions in DBLFLT1.

3. The routines for absolute value, complement, and MOVE12 have been modified. It is now unnecessary to set IR.2 before entering COM1 or ABS1. All three of these routines increment IR.1 by 3 before returning to the user program, whereas in DBLFLT they increment this register only twice. Similarly, IR.1 and IR.2 are incremented by 3 each time MOVE12 is used.

The table below lists the DBLFLT tags deleted and their equivalent symbols in DBLFLT1.

DBLFLT1 Symbol	DBLFLT1 Octal	DBLFLT Tag	DBLFLT Octal
ADDT	1000	5A	1000
SUBT	1043	5B	1045
MULT	1052	5C	1054
DIVIDE	1161	5G	1163
MOVE12	1301	6A	1303
QAC12	1313	6B	1313
COM1	1332	6D-2	1332
ABS1	1335	missing	----
DD6	1337	6D	1334

SETUP	1413	7A	1406
EXIT	1627	7Q	1623
ARG1	1765	7X	1723
ARG2	1770	7Y	1726
FAC	1773	7Z	1731
FLOAT	1736	5O	1743
FIX	1750	5P	1755
TRANS	1727	5N	1734

APPENDIX I - DBLFLT1 OPERATIONS

All DBLFLT1 operations are performed by a JMP to the appropriate entry point in DBLFLT1. The C(AC) are ignored by all routines. The accumulator is cleared following a return from the routines ADDT, SUBT, MULT, DIVIDE, TRANS, and FLOAT; and is indeterminate or contains information following return from the other routines.

ABS1 - Compute Absolute Value via IR.1

ABS1 is an unargumented operation which computes the absolute value of the number pointed to by IR.1.

First set IR.1 to the location of the exponent of the number to be operated on. Then enter via: JMP ABS1. Return is to .+1 with C(AC) = Ø and the result left in the same locations where the number was found. IR.1 is incremented by 3. Note that IR.1 must be set immediately before the JMP to ABS1 as it is used by most other DBLFLT1 operations.

ADDT - Add Two DBLFLT Numbers

Enter via JMP ADDT followed by 1, 2 or 3 arguments as described in the section on Argumented Operations. Returns with C(AC) = Ø to .+2, to .+3 or to .+4 as appropriate.

COM1 - Compute Negative via IR.1

COM1 is an unargumented operation which computes the negative of the number pointed to by IR.1. First set IR.1 to the location of the exponent of the number. Then enter via JMP COM1. Returns to .+1 with C(AC) = Ø and the result left in the same location where the number was found. IR.1 is incremented by 3. IR.1 must be set immediately before JMP COM1 as most other DBLFLT1 operations will alter its contents.

FIX - Fix a DBLFLT Number

This routine will convert a DBLFLT format number into a signed 23_{10} bit fixed point number.

Enter via a JMP FIX followed by a single (negative) argument. The argument specifies the location of the DBLFLT number which is to be fixed and the resulting fixed number is stored in the mantissa words of the same location. Return is to .+2 with the accumulator cleared unless the number was too big to fix in 23_{10} bits, in which case the accumulator is non-zero (and shows how many bits too big the exponent was). The (negative) number at location FIX+7 determines the number of bits to the right of the most significant sign bit that the binary point is located. It is presently equal to -27 (=23_{10}). The FAC is unaffected unless it is explicitly referenced. Express ones complement data field arguments as -2000-L(A).

The following example will fix a DBLFLT number in LOC1 placing sign plus 11 bits to the left of the decimal point and a 12 bit fractional part to the right of the decimal point.

```
LDA I
-13
STC FIX+7       /Modify FIX routine for 11 bits
JMP FIX
-LOC1
AZE
HLT             /Too big to fix
LDA I           /Restore FIX routine
-27
STC FIX+7
```

The signed integer portion of the result is left in LOC1+1; the 12-bit fractional portion in LOC1+2.

<u>FLOAT</u> - Float a 24-bit Fixed Point Number

This argumented operation will convert a fixed point integer of up to 23 bits plus sign into the equivalent DBLFLT number. The float routine expects to find the fixed point number already stored in the mantissa words (.+1 and .+2) of the argument location. The contents of the exponent location are ignored. The sign bit must be placed in the most significant bit of the most significant mantissa word. Enter via JMP FLOAT followed by a single (negative) argument. Return is to .+2 with C(AC) = Ø and the DBLFLT format floated number stored in the 3 core locations referenced by the argument. The Float subroutine assumes that the decimal point follows the least significant bit. If not, the contents of location FLOAT+7 must be changed to 27_8-b where b is the number of bits which follow the decimal point. The FAC is unaffected unless explicitly referenced. Express ones complement data field locations as -2000-L(A).

Consider the following examples:

Float a single precision signed number into locations ABC, ABC+1, and ABC+2 in the calling program instruction field.

```
LDA I           /Load the number
XXX
STA             /Put it in ABC+2
ABC+2
SCR 13          /Get rid of number but keep the sign by
                  scaling right 11₁₀ places
STC ABC+1       /Put sign in Bit Ø of ABC+1
JMP FLOAT       /Float number
-ABC
returns here with DBLFLT number in L(ABC)
```

Float a signed double precision number in which the decimal point is to the left of the least significant 12 bits. Get from 451-452 and leave the result in LOC1.

```
LDA I
13
STC FLOAT+7    /Modify FLOAT+7 for 11 bit fixing
ADD 451        /Get sign and most significant 11 bits
STC LOC1+1     /Save in LOC1+1
ADD 452        /Get least significant 12 bits
STC LOC1+2     /Save in LOC1+2
JMP FLOAT
-LOC1
LDA I          /Restore float routine
27             /to its normal condition
STC FLOAT+7
continue here
```

MOVE12 - Move a DBLFLT Number via IR.1 and IR.2

MOVE12 is an unargumented operation which transfers the contents of three consecutive core locations from the address in IR.1 to the address in IR.2. First set IR.1 and IR.2. Then enter via JMP MOVE12. Return is to .+1 with C(AC) = Ø and with IR.1 and IR.2 each incremented by 3. MOVE12 essentially duplicates the function of TRANS but is quicker and in some cases requires a shorter calling sequence. Note that IR.1 and IR.2 must be set immediately before the JMP to MOVE12 as they are used by most other DBLFLT1 operations. Both the routines below will move a list of 7 DBLFLT numbers from LOC1 to LOC2.

```
SET I 1         SET I 6
LOC1            LOC1
SET I 2         SET I 7
LOC2            LOC2
SET I 3         SET I 10
-7              -7
JMP MOVE12      JMP TRANS
XSK I 3         6
JMP.-2          -7
continue here   XSK I 10
                JMP .-4
                continue
```

MULT - Multiply Two DBLFLT Numbers

Enter via JMP MULT followed by 1, 2 or 3 arguments as described in the section on Argumented Operations. Returns to .+2, to .+3 or to .+4 as appropriate with C(AC) = Ø.

QAC12 - Move MQ Register to AC, All 12 Bits

Enter via JMP QAC12. Return is to .+1 with the accumulator containing 12-bits previously in the MQ. The original C(AC) are lost.

SUBT - Subtract One DBLFLT Number From Another

Enter via JMP SUBT followed by 1, 2 or 3 arguments as described in the section on Argumented Operations. Returns with accumulator cleared to .+2, to .+3 or to .+4 as appropriate.

TRANS - Move a DBLFLT Number

Enter with JMP TRANS followed by one or two arguments. The last argument must be negative. Return is to .+2 or .+3 with C(AC) = Ø.

```
JMP TRANS      JMP TRANS
2\LOC1         -2000-LOC1
-DS2           return here
return here
```

In the first example, C(LOC1) are moved into L(DS2). FAC and LOC1 are unchanged. In the second example, C(FAC) are moved into L(LOC1). The FAC is unchanged by TRANS unless it is listed as the second argument. Express ones complement data field addresses as -2000-L(A).

APPENDIX II

DBLFLT1 OPERATIONS

SUBROUTINE FUNCTION	ENTRY POINT	NUMBER OF ARGUMENTS	C(AC) USED?	OTHER SETUP NEEDED	RETURNS to .+	C(AC)=0 AFTER RETURN?	TYPICAL TIME (msec)
Addition	ADDT	1,2,3	no	**	2,3,4	yes	1.2-2.4
Subtraction	SUBT	1,2,3	no	**	2,3,4	yes	1.2-2.5
Multiplication	MULT	1,2,3	no	**	2,3,4	yes	1.6
Division	DIVIDE	1,2,3	no	**	2,3,4	yes	9.3
Complement	COM1	Ø	no	IR.1	1	yes	.05
Absolute Value	ABS1	Ø	no	IR.1	1	yes	.05
FIX into 24 bits	FIX	1	no	**	2	no=error	0.5-1.8
Float 24 bit no.	FLOAT	1	no	**	2	yes	0.6
Transfer DBLFLT NO.	TRANS	1,2	no	**	2,3	yes	0.4
Move 3 words	MOVE12	Ø	no	IR.1-2	1	yes	.5

**Any index registers listed as arguments must be set before entry

DBLFLT3

Source Titles: DBLFLT3 , DBL3GO[1]

Source Language: LAP6-DIAL[2]

Computer: PDP-12[3]

Authors: Paul F. Sullivan[4,5,6]
Rayna B. Cole
NASA Electronics Research Center
Cambridge, Massachusetts

ABSTRACT

DBLFLT3 contains LINC mode double precision floating point mathematical routines. It can perform many common mathematical functions (add, subtract, sin X e^X, etc.) as well as teletype input and output. All operations use a double precision mantissa and a 12-bit exponent. The program occupies 6 1/2 quarters of memory located outside the current instruction field. Programs designed to use the LINC-8 programs DBLFLT or DBLFLT 2 can easily be modified to operate with DBLFLT3.

---

1. DBLFLT3 is a revision of DBLFLT 2 (DECUS NO. L-68) for the LINC-8.

2. Many DBLFLT3 routines have previously been circulated in LAP4 or LAP6 language. The documents describing these previous versions are DEC-L8-FLAA-D, DEC-L8-SFAA-D and the DECUS NO. L-68 document.

3. DBLFLT3 will operate in the PDP-12C. However, at present the source and binary are only available on LINC tape.

4. Current address is Cornell Aeronautical Labs, Buffalo, New York 14221

5. The central mathematical program (DBLFLT) as well as some of the subroutines were originally written by Michael McDonald, Biomedical Computer Laboratory, Washington University, St. Louis, Missouri. The remaining routines as well as the control programs which allow DBLFLT3 to be located outside the current instruction field were written by Paul Sullivan and Rayna Cole.

6. This document was prepared by D. A. Overton, Eastern Pennsylvania Psychiatric Institute, 3300 Henry Avenue, Philadelphia, Pennsylvania 19129.

NOTATION

Throughout this report the following notational conventions will be employed:

1. Numbers written with a decimal point are decimal numbers. All other numbers are octal unless subscripted with a 10 or a 2 to indicate decimal or binary.

2. Single capital letters A, B, C, . . . denote the value of double precision floating point numbers, each of which numbers occupies three consecutive registers in core.

3. Multiple capital letters and numbers (FAC, TEM1, ADDT...) represent LAP6-DIAL symbolic addresses. When referencing DBLFLT format numbers, it is customary to specify the symbolic addresses of the first word (exponent) of the number.

4. L( ) designates 1 or 3 consecutive core locations as indicated by the context. If L(A) is the address of the first word (exponent) of the floating point number A, then A occupies the three registers in core symbolized by L(A), L(A)+1, L(A)+2.

5. C( ) specifies the contents of a register or of 1 or 3 memory locations, e.g. C(FAC) indicates the contents of the FAC.

6. $\longrightarrow$ means "is (are, be) placed into", e.g., A + B $\longrightarrow$ L(C) is read "the sum of the floating point numbers A and B is placed into the three consecutive registers beginning with L(C)."

ORGANIZATION

DBLFLT3 consists of three sections: DBLFLTA, DBLFLTB, and DBL3GO. DBLFLTA and DBLFLTB can be located in any two memory segments. A copy of DBL3GO must reside in each segment from which the user program makes a call to DBLFLT3 subroutine.

DBLFLTA contains the basic arithmetic operations (the original DBLFLT), the teletype inp and output routines, the subroutine COMPARE, some useful constants in DBLFLT format, temporary storage areas, and a control program to keep track of the subroutine called and the calling location.

DBLFLTB contains subroutines for calculating commonly used mathematical functions such as sine x, square root x, ln x, e^x, etc. It also contains a control program and a modified version of DBL3GO.

DBL3GO contains the instructions necessary to accomplish the jump to the proper DBLFLT3 subroutine. Only DBL3GO must reside in the same memory segment as the user program. All DBLFLT3 routines are entered by a JMP to DBL3GO. DBL3GO also defines those DBLFLT3 symbols which the user will want to reference.

MEMORY FIELD CONTROL

DBLFLTA fills segment SEGA. DBLFLTB occupies locations 0-1112 of segment SEGB. The unargumented DBLFLTB operations refer only to numbers stored in segments SEGA or SEGB and field control is automatic. Most DBLFLTA instructions (ADDT, SUBT, MULT, DIVIDE, FLOAT, FIX, TRANS, COMPAR, ABS1, COM1) may refer to DBLFLT numbers located in any data field and it is the user program's responsibility to have the data field set pointing to the referenced numbers before entering DBLFLT3. Memory field control is different for DBLFLTA routines than for DBLFLTB routines and each will be described separately.

When executing a DBLFLTA operation (e.g., SUBT) the user program jumps to the SUBT entry point in the copy of DBL3GO located in the current instruction field. DBL3GO transfers control to the control program in segment SEGA. The control program determines what instruction and data fields were established prior to the jump to DBL3GO. It then sets the data field to the calling program memory segment in order to obtain the return jump location and the arguments which follow JMP SUBT in the user program. When this is accomplished, the data field is restored to that set by the user program prior to the jump to DBL3GO. Now a jump is made to the actual DBLFLT1 subtract routine in segment SEGA. Note that during subtraction, the data field is that set by the user program and data field addresses listed as arguments in the user program will be correctly interpreted. When the mathematical operation is complete, control returns to the DBLFLTA control program which supervises the return jump to the calling program. Upon returning to the user program, both data and instruction fields will be set as they were when the initial jump was made to DBL3GO. If a DBLFLTA argument specifies an index register (see below), the appropriate data field address (bit 1 = 1) must be placed in the referenced index register of segment SEGA before entrance to DBL3GO. Note that argumented DBLFLTA instructions may operate on numbers stored in any memory segment either directly or via index registers 6-11 in segment SEGA. Obviously all data field addresses referenced by a single argumented DBLFLTA operation (directly or via index registers) must be in the same data field.

Entrance to DBLFLTB routines is achieved by a jump to the desired entry point in DBL3GO. DBL3GO effects a jump to the DBLFLTB control program which recovers and saves the instruction and data fields set prior to entry and the return jump. It then sets the data field to SEGA. Most DBLFLTB routines operate on specified locations (e.g., FAC) in DBLFLTA and these are accessed as data field addresses. When the DBLFLTB operation is complete, the DBLFLTB control program re-establishes the data and instruction fields set prior to entry to DBL3GO, and executes a return jump to the user program.

If a DBLFLTB routine uses a DBLFLTA operation, control is transferred through the modified version of DBL3GO located at the end of DBLFLTB. In this case the DBLFLTA control program is altered (and later automatically restored) so that the DBLFLTA operation is performed with the data field pointing to DBLFLTB. When the DBLFLTA operation is completed, control is returned to the calling DBLFLTB routine and ultimately to the user program. Note that the data field set by the user program has no effect on DBLFLTB operations. The user may essentially ignore field control when using DBLFLTB routines except in the case of the power series routine. The Table of Constants for this routine must be placed in segment SEGB in order to be accessible to the power series routine.

Floating Point Format

A number X is said to be expressed in binary floating point representation if it consists of two parts: an integer exponent or characteristic $\mathcal{E}$, and a mantissa or fractional part f such that

$$X = 2^{\mathcal{E}} \cdot \underline{f} \qquad \text{where } 0 \leq |\underline{f}| < 1.$$

The number is normalized if the inequality of f is

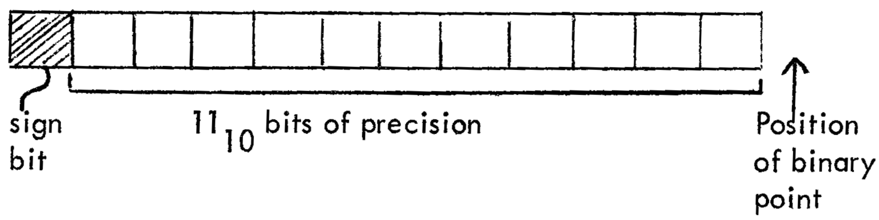
$$1/2 \leq |\underline{f}| < 1.$$

Any non-zero number can be written in what is termed normalized binary floating point representation.

Any number X that is manipulated (or generated) by DBLFLT must be (or is) in a normalized binary double precision floating point format consisting of three consecutive words, a one word signed exponent $\mathcal{E}$, and two signed mantissa words as follows:

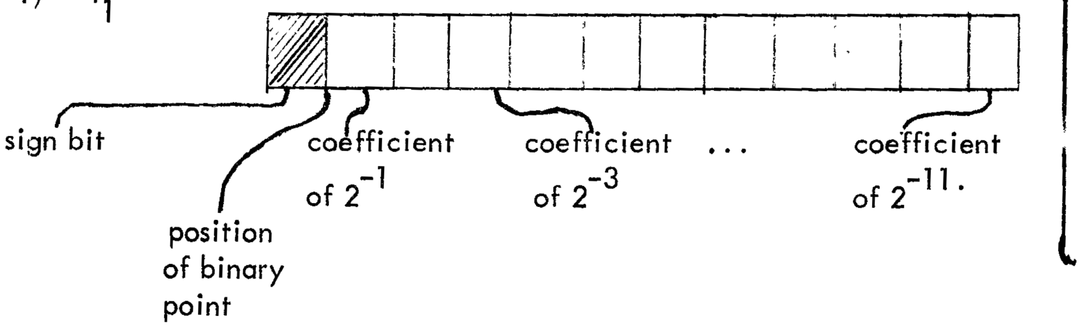
Exponent

$C(L(X)) = \mathcal{E} =$



Most significant portion of mantissa

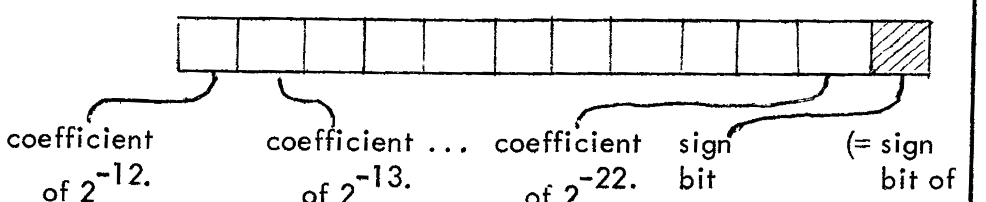
$C(L(X) + 1) = f_1 =$



total of 22_{10} bits of precision

Least significant portion of mantissa

$C(L(X)+2) = f_2 =$



For $f \geq 1/2$ (i.e. for the number to be normalized) bit 1 of f_1 must be different from the sign of the mantissa (bit 0 of f_1 and 11. of f_2). A normalized zero is defined to be $\mathcal{E} = 4000 = -3777$ (most negative exponent) and $f_1 = f_2 = 0000$. Negative exponents and mantissas are ones complement numbers. The two sign bits in the mantissa must, of course, be equal.

For clarification, let us look at a few examples of numbers in DBLFLT format.

1. $X = +1.0$

0001
2000
0000

$E = 1$ and $f = 2000\ 0000 = 0_{\wedge}\ 10\ 000\ 000 \ldots 000_2$

binary
point

$X = 2^E \cdot f = 2^1(1 \cdot 2^{-1} + 0 \cdot 2^{-2} + \ldots) = 2(1/2) = 1.0$

2. $X = -100.0$

0007
4677
7777

$E = 7$ and $f = 4677\ 7777$

$f = -\bar{f} = -(3100\ 0000) = -(0_{\wedge}\ 11\ 001\ \ 000\ \ldots\ 000_2)$

binary
point

$X = 2^E \cdot f = -2^7(1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} + 0 \cdot 2^{-4} + 1 \cdot 2^{-5} + 0 \cdot 2^{-6} + \ldots)$

$= -(2^6 + 2^5 + 2^2) = -(64 + 32 + 4) = -100.0$

3. $X = +.125$

7775
2000
0000

$E = 7775 = -(0002)$ and $f = 2000\ 0000$

$= 0_{\wedge}\ 10\ \ 000\ \ 000\ \ldots\ 000_2$

binary
point

$X = 2^E \cdot f = 2^{-2}(1 \cdot 2^{-1} + 0 \cdot 2^{-2} + \ldots) = 1/4(1/2) = 1/8 = +.125$

Precision

Two extra bits of precision, beyond the 22_{10} bits in the double precision fraction, are maintained by the DBLFLT1 subroutines during their operation in order to insure there always being an extra bit on which to round. After the mantissa of the result is calculated, its absolute value is obtained; this value is normalized and then rounded to 22_{10} bits by the addition of a 1 to the 23rd bit followed by truncation after the 22nd bit. Finally the correct sign is restored.

The user is advised to note that many numbers which terminate in the decimal system, e.g. 0.1, are periodically infinite when expressed in their binary representation. Thus truncation error can become a problem where none is normally expected; such error cannot be totally eliminated, but can be minimized by careful programming, e.g., to calculate 0.1˙X˙Y, instead of multiplying 0.1 by X and that result by Y, divide the product X˙Y by 10.0.

For a thorough discussion of the errors associated with roundoff in both fixed and floating point computations, it is recommended that the reader see J. H. Wilkinson's book, Rounding Errors in Algebraic Processes, Prentice-Hall, 1963.

ARGUMENTED OPERATIONS

Execution of the DBLFLT3 operations ADDT, SUBT, MULT and DIVIDE is accomplished by a jump to the DBL3GO entry point for the specific routine followed in consecutive registers by a list of arguments. This argument list may consist of 1, 2, or 3 items. The method of argumenting these instructions is described below. The DBLFLT3 operations FLOAT, FIX, TRANS and COMPAR are also argumented but only with one or two arguments. The procedure for argumenting these latter operations is described in Appendix I for each operation individually. All other DBLFLT3 operations are accomplished by unargumented jumps.

The three argument calling sequence is as follows:

General Form	Specific Examples		
JMP EP	JMP SUBT	JMP DIVIDE	JMP MULT
L(A)	TEM1	DS1	2\ LOC1
L(B)	DS1	FAC	2\ LOC2
±L(C)	DS2	-TEM2	FAC
return here			

EP is the DBL3GO entry point. The arguments are the locations of the exponents of 3-word DBLFLT numbers (A, B, and C). Letting R stand for one of the operations (+, -, X, ÷), the action taken by the subroutine will be:

$$A\ R\ B \longrightarrow L(C)$$

i.e., the operation R will be performed on the floating point numbers A and B, in the order indicated, and the floating point result will be stored in L(C), L(C)+1, and L(C)+2. Control is returned to the main program at location .+4; A and B are unchanged. The minus sign is optional in 3-argument entries and need not be present. (It may only be placed in front of the third argument.) In the first example the DBLFLT format number in DS1 will be subtracted from that in TEM1 and the result left in DS2. The FAC is not effected by this operation, nor are TEM1 or DS1. In the second example the C(DS1) are divided by the C(FAC) and the 3-word result is left in TEM2. DS1 and FAC are unchanged. In the third example, the number in LOC1 is multiplied by that in LOC2 and the result is placed in the FAC. Both 2\LOC1 and 2\LOC2 are data field locations and the data field must have been correctly set prior to the JMP MULT operation. LOC1 and LOC2 must be in the same memory segment.

The same four DBLFLT operations may be followed by only two arguments. In this case the three consecutive addresses in DBLFLT called the floating point accumulator (FAC) are assumed to be the third argument and the result is left in these locations.

General Form	Specific Examples	
JMP EP	JMP SUBT	JMP DIVIDE
L(A)	TEM1	2\LOC1
-L(B)	-2000-LOC1	-TEM2
returns here		

These operations will perform:

$$A\ R\ B \longrightarrow L(FAC)$$

Note that the second item <u>must</u> be -L(B), the ones complement of L(B). Control is returned to the main program at location .+3; A and B are unchanged. In the first example C(LOC1) in the data field are subtracted from C(TEM1) and the result is left in the FAC. The second example divides C(LOC1) by C(TEM2).

The one argument calling sequence is as follows:

General Form	Specific Examples	
JMP EP	JMP SUBT	JMP ADDT
-L(A)	-TEM1	-TEM1
returns here		

The stated argument is taken to be the second argument and the first and third arguments are assumed to be the FAC. The action taken is:

$$C(FAC)\ R\ A \longrightarrow L(FAC)$$

i.e., the first argument is the current contents of the FAC and the result is stored into the FAC. Note that the argument must be -L(A), a ones complement number. Control is returned to the main program at location .+2; A is unchanged. The first example subtracts C(TEM1) from C(FAC) leaving the result in the FAC. The second adds C(TEM1) to C(FAC) leaving the result in the FAC.

The FAC is always altered if these operations (+, -, X, ÷) have only one or two arguments. However, the FAC is not altered by operations with three arguments unless it is specifically referenced. Many unargumented DBLFLT3 operations alter the FAC (see Appendix II).

An argument may be indirectly addressed by placing the address of its exponent in an index register of segment SEGA and using the index register as an argument in the calling sequence. Allowable index registers are 6-11. Registers 13-16 may also be used if the routines TTYIN and TTYOUT are not called. The location of the argument will be indexed by 3 immediately after its use by DBLFLT1. Consider the following instructions, where A is at LOC1 and B is at LOC1+3.

```
LDF SEGA              /Data field to DBLFLT A
LDA I                 /Load IR.10 in SEGA
2\LOC1
STA
2\ 10
LDF SEGC              /Data field to LOC1
JMP EP
  10
 -10
returns here
```

The operation R will be performed on the numbers A and B, in that order, and the result stored in the FAC. When control is returned to the main program, index register 10 in SEGA will contain LOC1+6. Note that when an index register is used to reference a DBLFLT number, it is set equal to the address of the exponent of the DBLFLT number, not one address ahead of the exponent.

As another example, the following sequence will add 100 DBLFLT numbers starting at address 1000 in segment Ø and leave the result in the FAC.

```
JMP TRANS             /Set FAC = Ø
   DCØ
  -FAC
SET I 7               /Set IR.7 in SEGA
 2\ 1000              /Points to DATA
JMP IRLOAD
SET I 7               /100₁₀ data values
 -144
LDF Ø                 /Points to data
JMP ADDT              /Add one data value to FAC
-7
XSK I .'              /Added 100?
JMP .-3               /No, do another
continue here
```

The coding of ones complement addresses for argumented instructions requires some comment. For locations within DBLFLTA, a simple minus sign will suffice (e.g., -FAC, -DS1). To assemble a ones complement data field address for LOC1, you must type -2000-LOC1. DIAL does not assemble the correct code from the statement -2\ LOC1, nor from -2\ -LOC1 or 2\ -LOC1.

Halts

There are two halts indigenous to DBLFLT1:

a) Location DIVIDA+7 - A halt in this location means that an attempt was made to divide by zero.

b) Location DF6-1 - A halt in this location indicates an overflow condition, i.e., that the exponent of the result of some operation exceeds +3777.

If a halt occurs at one of these DBLFLT1 locations, inspection of V4+4 and V4+5 in the DBLFLTA control program and DT7 in DBLFLT1 will show from where the call originated.

If underflow occurs, i.e., the exponent of the result becomes more negative than -3777, the result is set equal to 0.0 and no halt occurs. N.B. Dividing 0.0 by a number with exponent greater than zero or multiplying 0.0 by a number with exponent less than zero will result in the correct result of 0.0, but this answer will be obtained because of underflow.

There are also halts in other DBLFLT3 routines. The listing indicates the meaning of each halt.

Storage Locations

Space for temporary storage of 6 DBLFLT numbers is provided within DBLFLTA. These 18 memory locations are not used by DBLFLT3, unless listed as an argument. They are as follows:

Symbolic	Octal
DS1	1731
DS2	1734
DS3	1737
DS4	1742
DS5	1745
DS6	1752

In addition user programs will often want to refer to the following four DBLFLTA storage locations:

FAC	1763
TEM1	1766
TEM2	1771
TEM3	1774

Additional storage locations may be placed in any memory segment and referenced as data field addresses. Be sure to correctly set the data field before executing DBLFLTA operations referring to such locations.

Constants

DBLFLTA contains the following useful constants in DBLFLT format:

Number	Symbolic Address of Exponent	Octal Location of Exponent
0.0	DCØ	1660
1.0	DC1	1663
2.0	DC2	1674

Number	Symbolic Address of Exponent	Octal Location of Exponent
3.0	DC3	1645
4.0	DC4	1666
10.0	DC1Ø	1655
90.0	DC9Ø	1642
$\sqrt{2}$	DCRT2	1671
$\log_{10}2$	DCLOG2	1652
ln 2	DCLN2	1647
π	DCPI	1701
$\pi/2$	DCPID2	1704
$\pi/4$	DCPID4	1676

Additional constants which are frequently used may be added to this list at the sacrifice of these constants or of temporary storage locations. Constants which are seldom used should be located elsewhere in memory and accessed by using the appropriate data field address. The constants for the expansions of sin X, arctan X, and arcsin X are stored in the memory bank of DBLFLTB since they are not of general use. Many of the constants in DBLFLTA are used by the routines in DBLFLT3.

ASSEMBLY

The DBLFLT3 source practically fills the working area so that user programs cannot be added to it, and must be assembled separately. Typically DBLFLT3 will be assembled first, followed by the user program.

The source DBL3GO defines the locations in DBLFLT3 which the user may have to access Four versions of DBL3GO have been prepared with non-identical entry point symbols (r ADDT, ADDTX, ADDTY, ADDTZ).

To illustrate the use of these sources, suppose the user program consists of two sources. One fills segments 0 and 2. The second fills segment 5. All three segments require access to DBLFLT3. DBL3GO may be located in segments 2 and 5 and DBL3GOX in segment Ø. Assemble DBLFLT3 first (it need not be removed from file). Then assemble segment 5. Assemble segments 0 and 2 last.

MODIFYING DBLFLT3

The package DBLFLT3 is designed to permit ready modification by user. Some advice in making certain types of modifications is mentioned in this section.

The simplest type of modification is a change to a different memory field configuration. To accomplish this change, it is necessary only to modify the Memory Segment Assignment equalities at the beginning of DBLFLT3 and in DBL3GO. The symbols SEGA and SEGB must be set to the desired memory segments of DBLFLTA and DBLFLTB. Ordinarily, these symbols will have the same values in every program segment in which they occur.

The DBLFLTB control program can easily be used to access additional routines placed in locations 1113-1777 of segment SEGB. Additional entry points must be added to DBL3GO just ahead of location NEGFAC and the jump list in SEGB must be expanded to show the new entry points. New entry points to DBLFLTA can also be added if desired.

DBL3GO may be shortened only by deleting unused entry points starting at the top of each XSK I 17 series (i.e., starting with NEGFAC and KBD). It is not legal to delete entries internal to the XSK I 17 series (i.e., SIGN) unless all entry points above the one to be deleted have also been deleted.

If a wholesale reshuffling of DBL3GO is attempted, it is necessary that the jump lists in SEGA and SEGB match the sequence of entry points in the copies of DBL3GO located in each segment of the user program (and at the end of DBLFLTB). Note also the instructions at locations V6+10 to V6+14 in the DBLFLTA control program.

If a slight increase in speed will help, the DECUS No. L-68 control program (along with the program called DBLFLTGO) takes about half as long as that in DBLFLT3 and may be modified for use with DBLFLT3. However, it uses more locations in the main program instruction field than DBL3GO and, in its present form, does not give the user control over the data field established during DBLFLT1 operations.

The subroutines SQROOT, FIX12, ARCSIN, and LOGS are presently set up to return to location .+1 if the argument is outside the interval normally expected for these routines. This configuration was designed to permit easy error recovery but it necessitates the allocation of an extra memory location to each call to one of these routines. If the error recovery option is not needed and the programmer is strapped for space, these routines can be readily modified by replacing certain XSK and JMP instructions with NOP's and HLT's so that the program halts in DBLFLTB under error conditions and returns to .+1 as the normal exit. The changes necessary are obvious from the program listings.

If you wish to delete substantial sections of DBLFLT3 it may be reassuring to know that all memory references are symbolic except those within individual subroutines and the JMP 20 instruction in DBL3GO. Hence, if assembly causes no error messages, the resulting binary will probably run. DBLFLTB, along with its half of DBL3GO, may be deleted entirely (or overlaid) without affecting the operation of DBLFLTA.

<u>Modifying DBLFLT Program to Use DBLFLT3.</u>

Programs written for use with the original DBLFLT can generally be used with DBLFLT3, provided the following modifications are made:

1. All jumps to locations in DBLFLT or its subroutines must be changed to become jumps to DBL3GO. Some infrequently-used entry points to DBLFLT have not been included in DBL3GO. These entries to DBLFLT are not possible without modifying DBL3GO.

2. Addresses relative to tags in DBLFLT, IFORL8(TTYIN) or OFORL8 (TTYOUT) should be checked to determine whether the modifications in these routines necessitate modification of the address calculation.

3. All references to locations in DBLFLT except those appearing as arguments of DBLFLT instructions must be incremented by 2000_8, i.e., made into data field addresses. (This is most easily accomplished by typing 2\ prior to the address in manuscripts to be assembled by LAP6-DIAL.) Note that this precludes direct addressing of these locations.

4. All references to locations in the main program which appear as arguments of DBLFLT instructions must be incremented by 2000, i.e., made into data field addresses.

5. Data field control must be inserted so that DBLFLT3 references the appropriate arguments.

6. Instructions which set index registers for use as arguments of DBLFLT instructions must be replaced by routines which set the equivalent SEGA register.

7. The manuscript of DBL3GO must be added to the main program source.

APPENDIX I - DBLFLT3 OPERATIONS

All DBLFLT3 operations are performed by a JMP to the appropriate entry point in DBL3GO. The C(AC) are used by a few operations and are ignored by the other routines. Upon return the contents of most registers (Multiplier Quotient, FLO) are indeterminate. The accumulator is cleared in many cases, indeterminate in others and in a few cases contains information (KBDI, FIX12).

ABS1 - Compute Absolute Value via IR.1

ABS1 is an unargumented operation which computes the absolute value of the number pointed to by IR.1 in segment SEGA. First set IR.1 to the location of the exponent of the number to be operated on using IRLOAD. Then enter via JMP ABS1. Return is to .+1 with C(AC) = Ø and the result left in the same locations where the number was found. IR.1 is incremented by 3. Note that IR.1 must be set immediately before the JMP to ABS1 as it is used by most other DBLFLT3 operations.

ADDT - Add Two DBLFLT Numbers

Enter via JMP ADDT followed by 1, 2 or 3 arguments as described in the section on Argumented Operations. Returns with C(AC)=Ø to .+2, to .+3 or to .+4 as appropriate.

In this routine the exponents of A and B are compared. The larger exponent becomes the exponent of the result; and the fraction of the number with the smaller exponent is shifted right a number of places equal to the difference between the two exponents, i.e., the binary points are aligned. The fractions are then added, the sum becoming the mantissa of the result.

ARCSIN - Compute Arcsin X

ARCSIN calculates ARCSIN X for $-1<X<1$. The answer is given in radians between $-\pi/2$ and $\pi/2$. To use ARCSIN, place X in the FAC and enter via a JMP ARCSIN. The program returns to .+1 if $|X| > 1$. Otherwise, return is to .+2 with arcsin X in the FAC and $\longleftarrow$ C(AC) ≠ Ø.

This subroutine uses the approximation, given by Hastings*:

$$\text{Arcsin } X = \pi/2 - \sqrt{1-X}\ \psi(X).$$

where:

*Hastings, Cecil, Jr., Approximations for Digital Computers, Princeton University Press (1955).

$$\psi(X) = A_0 + A_1X + A_2X^2 + A_3X^3 + A_4X^4 + A_5X^5$$

$A_0 = 1.570795207$

$A_1 = -0.214512362$

$A_2 = 0.087876311$

$A_3 = -0.044958884$

$A_4 = 0.019349939$

$A_5 = -0.004337769$

and $0 \leq X \leq 1$.

For X<0, the subroutine uses the absolute value of X in the calculation and complements the answer automatically.

<u>ARCTAN</u> - Compute Arctangent X

ARCTAN calculates the arctangent of X for any X. The answer is given in radians between $-\pi/2$ and $\pi/2$. To use ARCTAN, place X in the FAC and enter via a JMP ARCTAN. Return is to .+1 with arctan X in the FAC and C(AC) ≠ Ø.

The following approximation from Hastings is used:

$$\text{Arctan } Y = \pi/4 + C_1Z + C_3Z^3 + C_5Z^5 + C_7Z^7 + C_9Z^9$$

where:

$$Z = \frac{Y-1}{Y+1}$$

$C_1 = 0.9998660$

$C_3 = -0.3302995$

$C_5 = 0.1801410$

$C_7 = -0.0851330$

$C_9 = 0.0208351$

and $0 \leq Y < \infty$

If X<0, the calculation is performed with $|X|$ and the result is complemented automatically before exiting ARCTAN.

COM1 - Compute Negative via IR.1

COM1 is an argumented operation which computes the negative of the number pointed to by IR.1 in segment SEGA. First set IR.1 to the location of the exponent of the number using IRLOAD. Then enter via JMP COM1. Returns to .+1 with C(AC) = Ø and the result left in the same location where the number was found. IR.1 in SEGA is incremented by 3. IR.1 must be set immediately before JMP COM1 as most other DBLFLT1 operations will alter its contents.

COMPAR - Compare Two Numbers

Compare is argumented instruction which looks at any two DBLFLT numbers and determines whether A=B, A>B or A<B. The entry jump to COMPAR must be followed by one or two arguments (the last argument is negative).

```
JMP COMPAR          JMP COMPAR
-L(A)               L(A)
.+2                 -L(B)
.+3                 .+3
.+4                 .+4
                    .+5
```

If a single argument is used, the routine compares the FAC with the number A at location L(A). Return is to .+2 if FAC=A, to .+3 if FAC>A, and to .+4 if FAC<A. If two arguments are used, A and B are compared. Return is to .+3 if A=B, to .+4 if A>B, and to .+5 if A<B. C(AC) ≠ Ø upon exit. The numbers referenced by the arguments are not changed by COMPAR. Express ones complement data field arguments as-2000-L(A).

COSINE - See SIN COS

DIVIDE - Divide One DBLFLT Number By Another

Enter via JMP DIVIDE followed by 1, 2 or 3 arguments as described in the section on Argumented Operations. Returns to .+2, to .+3, or to .+4 as appropriate, with C(AC) = Ø.

In this routine the exponent of the result is set equal to C(L(A)) - C(L(B)). The mantissa of A is then divided by the mantissa of B, the quotient becoming the mantissa of the result.

EXPON - Computes e^X

To use EXPONENTIAL, place the DBLFLT number X in the FAC. Then enter via JMP EXPON. Return is to .+1 with e^X left in the FAC and C(AC) = Ø.

FIX - Fix a DBLFLT Number

This routine will convert a DBLFLT Format number into a signed 23_{10} bit fixed point number. Enter via a JMP FIX followed by a single (negative) argument. The argument specifies the location of the DBLFLT number which is to be fixed and the resulting fixed number is stored in the mantissa words of the same location. Return is to .+2 with the accumulator cleared unless the number was too big to fix in 23_{10} bits in which case the accumulator is non-zero (and shows how many bits too big the number was). The (negative) number at location FIXA+7 determines the number of bits to the right of the most significant sign bit that the binary point is located. It is presently equal to -27 (=23_{10}). The FAC is unaffected unless it is explicitly referenced. Express ones complement data field arguments as -2000-L(A).

The following example will fix a DBLFLT number in DS1 placing sign plus 11 bits to the left of the decimal point and a 12 bit fractional part to the right of the decimal point.

```
LDF SEGA          /Data field to DBLFLTA
LDA I
-13
STA               /Modify FIX routine for
2\ FIXA+7         /11 bits before decimal point
JMP FIX
-DS1
AZE
HLT
LDA I             /Restore FIX routine
-27
STA
2\ FIXA+ 7
```

The signed integer portion of the result is left in DS1+1; the 12-bit fractional portion in DS1+2.

FIX12 - Fix DBLFLT Number Into 12 Bits

Enter via JMP FIX12. This routine takes the DBLFLT number in the FAC and attempts to make it into a signed 12 bitinterger. If successful, return is to .+2 with the 12 bit num in the accumulator. If the number is too big to fit in 11 bits plus sign ($|X| >2047$), return is to .+1 with C(AC) equal to the number of bits by which the DBLFLT number exceeded 11 bits. In either case, FAC is altered. The fraction to the right of the decimal point is rounded into the integer to minimize truncation error.

FLOAT - Float a 24-bit Fixed Point Number

This argumented operation will convert a fixed point integer of up to 23 bits plus sign into the equivalent DBLFLT number. The float routine expects to find the fixed point number already stored in the mantissa words (.+1 and .+2) of the argument location. The contents of the exponent location are ignored. The sign bit must be placed in the most significant bit of the most significant mantissa word. Enter via JMP FLOAT followed by a single (negative) argument. Return is to .+2 with C(AC) = Ø and the DBLFLT format floated number stored in the 3 core locations referenced by the argument. The Float subroutine assumes that the decimal point follows the least significant bit. If not, the contents of location FLOATA+7 must be changed to 27_8-b where b is the number of bits which follow the decimal point. The FAC is unaffected unless explicitly referenced. Express ones complement data field locations as -2000-L(A).

Consider the following examples:

Float a single precision signed number into locations ABC, ABC+1, and ABC+2 in the calling program instruction field.

```
LDF SEGC                        /Set data field to calling program
LDA I                           /Load the number
XXX
STA                             /Put it in ABC+2
ABC+2
SCR 13                          /Get rid of number but keep the sign
                                /by scaling right 11_10 places
STC ABC+1                       /Put sign in Bit Ø of ABC+1
JMP FLOAT                       /Float number
-2000-ABC                       /in data field register
returns here with DBLFLT number L(ABC).
```

Float a signed double precision number in which the decimal point is to the left of the least significant 12 bits. Get from 451-452 and leave the result in DS1.

```
LDF SEGA                        /LDF to DBLFLTA
LDA I
 13
STA                             /Modify FLOATA+7 for fixing
2\ FLOATA+7                     /with 11 bits prior to decimal point
SET I 1
2\ DS1
SET I 2
 451
LDA 2                           /Get sign and most significant 11 bits
STA I 1                         /Save in DS1+1
LDA I 2                         /Get least significant 12 bits
STA I 1                         /Save in DS1+2
JMP FLOAT
-DS1
LDA I                           /Restore float routine
 27                             /to its normal condition
STA
2\ FLOATA+7
continue here
```

<u>FLOT12</u> - Float Contents of Accumulator

This subroutine will float a signed single precision (12-bit) fixed point number. Enter via JMP FLOT12 with the number to be floated in the accumulator. Return is to .+1 with the number left in the FAC in DBLFLT format and C(AC) = Ø. This entry is not argumented.

<u>IRLOAD</u> - Load Index Registers in DBLFLT3

Enter this routine via JMP IRLOAD. Return is to .+1 with C(AC) ≠ Ø. The routine transfers the contents of locations 1-11 and 13-16 from the calling program field into the corresponding locations in segments SEGA and SEGB. This allows the programmer to use SET and STC commands followed by JMP IRLOAD to set index registers in DBLFLT3. Registers 12 and 17 are specifically excluded from transfer as these are used by the control program and must not be disturbed. Not all of the 14 registers transferred are available for use. In DBLFLTA registers 6-11 are free and 13-16 may also be used if TTYIN and TTYOUT are not called. In DBLFLTB, registers 10, 11, and 13-16 are free.

<u>KBD</u> - Read ASR33 Keyboard

Enter with JMP KBD. Returns at .+1 with 8-bit ASCII code in accumulator if a character was waiting. Otherwise returns immediately to .+1 with C(AC) = Ø.

<u>KBDI</u> - Read ASR33 Keyboard

Enter with JMP KBDI. If no character has been struck the routine waits until one is struck. Return is to .+1 with the 8-bit ASCII code in the accumulator.

<u>LFCR</u> - Type Line Feed and Carriage Return

Enter with JMP LFCR. Returns to .+1 with C(AC) = Ø. Typing can be speeded up by changing JMP TYP8B to NOP at location TLFCR+5.

<u>LOGS</u> - Compute LOG X

LOGS calculates $\log_2 X$, $\log_{10} X$, or $\log_e X$ depending on the entry point. To use LOGS, place the DBLFLT number X in the FAC. Then enter via a JMP LOG2 if $\log_2 X$ is desired (mostly for logarithmic scaling of data), JMP LOG10 for $\log_{10} X$, or a JMP LOGN for t' natural logarithm, $\log_e X$. LOGS returns to .+1 if $X \leq 0$ with the FAC unchanged. If X [illegible] return is to .+2 with the appropriate logarithm in the FAC. The accumulator is cleared [illegible] re-turn is to .+2 but not cleared if return is to .+1.

The basic calculation in this subroutine yields $\log_2 X$; the other logarithms are derived from this one by multiplying by the appropriate constants. The algorithm makes use of the special format of DBLFLT numbers:

1) $$X = 2^L (Y_1)$$

where $1/2 \leq |Y_1| < 1$.

Taking the logarithm of eq. 1,

2) $$\log_2 X = L + \log_2 Y_1.$$

and L is the first approximation to $\log_2 X$. If X itself satisfies the inequality $1/2 \leq X < 1$, then L = 0 and we have no significant bits yet for the log calculation. On the other hand, if X lies outside this interval, then we have already obtained some bits (at most 11) of $\log_2 X$.

To get the remaining bits, we must get an approximation for $\log_2 Y_1$. As it stands, Y_1 would have zero in the exponent register, and so the above scheme would not give us a useful approximation to $\log_2 Y_1$. Since Y_1 is less than 1, raising it to some positive power Q will result in a non-zero, negative number in the exponent register, and this number can be used to derive an approximation of $\log_2 Y_1$ from the relation:

3) $$\log Y = Q^{-1} \log Y^Q$$

The higher the power Q, the more significant bits in the approximation, so Q should be as large as possible without producing overflow. The value of Y_1 for which the exponent register increases fastest is the smallest value, namely 1/2. In this case,

4) $$1/2^Q = 2^{-Q} = 2^{-(Q-1)} \; (1/2)$$

where the far right term of the equality represents the DBLFLT format number. In order that the most bits be obtained without the exponent register overflowing, Q should be chosen so that Q-1 is equal to the capacity of the exponent register or $2^{11}-1$. In DBLFLT format, we then have:

5) $$Y_1{}^Q = 2^M (Y_2)$$

and

6) $$\log_2 Y_1 = Q^{-1}(M + \log_2 Y_2).$$

From this we get a better approximation to $\log_2 X$, namely

7) $$\log_2 X = L + 2^{-11} M.$$

If we now treat Y_2 as we did Y_1, we get

8) $$\log_2 X = L + 2^{-11} M + 2^{-22} N,$$

where N is the exponent which results from raising Y_2 to the 2^{11th} power.

Since the original uncertainty in Y_1 was at least equal to the least significant bit or $\pm 2^{-22}$, and the uncertainty of a number raised to a power Q is Q times the uncertainty or the original number, we can extract no more significant bits by continuing this process. The subroutine LOGS, therefore, uses eq. 8 to determine $\log_2 X$.

<u>MAGTST</u> - Magnitude Test

This subroutine determines whether $|X| \leq 1$ or $|X| > 1$. Place X in the FAC. Then enter via JMP MAGTST. Return is to .+1 if $|X| > 1$ or to .+2 if $|X| \leq 1$ with $C(AC) \neq \emptyset$.

MULT - Multiply Two DBLFLT Numbers

Enter via JMP MULT followed by 1, 2 or 3 arguments as described in the section on Argumented Operations. Returns to .+2, to .+3, or to .+4 as appropriate with C(AC) = Ø.

In this routine the exponent of the result is set equal to the sum of the exponents of A and B. The least significant portion of the mantissas of A and B are rotated right one place in order to restore the sign bit to its normal position for use by the MUL command. The result fraction is calculated by forming the proper sums of the most and least significant products of the most and least significant parts of the fractions of A and B.

NEGFAC - Complement FAC

This subroutine computes the negative of the number in the FAC. First place X in the FAC. Then enter via: JMP NEGFAC. Return is to .+1 with -X left in the FAC with C(AC) ≠ Ø.

POWSER - Power Series

For any reasonable number of terms, this program calculates the power series:

$$C_n X^n + C_{n-1} X^{n-1} + \ldots + C_1 X + C_o$$

The table of constants must be placed in segment SEGB (in DBLFLT format) ordered sequentially beginning with C_n in the lowest 3 addresses and ending with C_o. Before entering the routine place X in TEM2 in DBLFLT format and set C(IR.4) = -n in segment SEGB (n = number of terms - 1). Put the starting address of the table of constants in the accumulator (as a data field address). Then enter via JMP POWSER . Return is to .+1 with the result left in the FAC and C(AC) = Ø. A sample calling sequence follows:

to compute $C_3 n^3 + C_2 n^2 + C_1 n + C$

```
LDF SEGB                  /If necessary
LDA I                     /Put -n in IR.4
 1-4
STA
2\ 4
JMP TRANS                 /Put X in TEM2
 L(X)
-TEM2
LDA I
2\L(C3)
JMP POWSER
returns here
```

SIGN - Sign Test

SIGN TEST determines whether X = 0, X>0, or X<0. Place the DBLFLT number X in the

FAC. Then enter via: JMP SIGN. Return is to .+1 if X = 0, to .+2 if X>0, or to .+3 if X<0 with C(AC) ≠ Ø.

SIN COS - Compute Sine or Cosine

To use SIN COS, place X in the FAC. Then enter via one of the following jumps:

To compute sin X:

JMP SINDEG if X is expressed in degrees
JMP SINRAD if X is expressed in radians
JMP SINPI2 if X is expressed in $\pi/2$ radians

To compute cos X:

JMP COSDEG if X is expressed in degrees
JMP COSRAD if X is expressed in radians
JMP COSPI2 if X is expressed in $\pi/2$ radians

Return is to .+1 with the answer left in the FAC and C(AC) ≠ Ø.

SIN COS calculates sin $\pi/2$ X according to the following approximation from Hastings:

$$\sin \pi/2\, X = C_1X + C_3X^3 + C_5X^5 + C_7X^7$$

where

$$C_1 = 1.570794852$$
$$C_3 = -0.645820978$$
$$C_5 = 0.079487663$$
$$C_7 = -0.00436246$$

SIN COS calculates the sine or cosine of any argument. Internal prescaling permits the direct calculation of sin x for either radian or degree arguments. Cosines are calculated by increasing the argument by $\pi/2$ radians and then calculating the sine of the resultant. An internal normalization routine automatically shifts the argument to the interval between $-\pi/2$ and $\pi/2$ radians, thus allowing solutions for any value of the argument. The ($\pi/2$ radians) measure equals 1 for 90° angle, 2 for 180°, etc.

SQROOT - Square Root

This program calculates square root X. Place X in FAC as a DBLFLT number. Then JMP SQROOT. The program returns with square root $|X|$ in the FAC. Return is to .+1 if X<Ø and to .+2 otherwise with C(AC) ≠ Ø.

SUBT - Subtract One DBLFLT Number From Another

Enter via JMP SUBT followed by 1, 2 or 3 arguments as described in the section on Argumented Operations. Returns with accumulator cleared to .+2, to .+3 or to .+4 as appropriate. In this routine the mantissa of the second argument is complemented and control is transferred to the add subroutines.

TEN2N - Compute 10^N

This routine will raise 10 to the power N where N is a positive or negative integer. N must be a signed 12 bit fixed point number with $|N|<1000_8$. Enter via JMP TEN2N with N in the accumulator. Return is to .+1 with 10^N left in TEM1 as a DBLFLT number and C(AC) ≠ Ø. This routine alters FAC, TEM1, and TEM2. An overflow halt will occur in DBLFLT1 if $|N| > 777_8$.

TRANS - Move a DBLFLT Number

Enter with JMP TRANS followed by one or two arguments. The last argument must be negative. Return is to .+2 or .+3 with C(AC) = Ø.

```
JMP TRANS            JMP TRANS
2\ LOC1               -2000-LOC1
-DS2                 return here
return here
```

In the first example, C(LOC1) are moved into (DS2). FAC and LOC1 are unchanged. In the second example, C(FAC) are moved into (LOC1). The FAC is unchanged by TRANS unless it is listed as the second argument. Express ones complement data field addresses as -2000-L(A).

TTYIN - Enter Numbers via Teletype. Make a DBLFLT Number.

This subroutine allows the user to enter decimal numbers via the ASR-33 teletype. A number may be entered in any allowable FORTRAN I, F, or E format, e.g., the number 497 may be entered in any of the following ways:

```
497
497.
497.0
49.7000 E+1
.497 E3.0
4970 E-1
```

To use this routine enter via JMP TTYIN with the accumulator either cleared or with the first 8-bit ASCII character in the accumulator. The routine then interrogates the teletype and enters decimal digits and characters until RETURN is struck. Each character except RETURN is echoed on the printer as it is struck. Normal return is to .+2 with the number left in the FAC in DBLFLT format, and C(AC) = Ø. Striking RUBOUT causes an immediate return to .+1 (in this case the contents of FAC are meaningless). The minus sign may be entered at any point in the number. The decimal point (.) is sensed and interpreted. Commas, spaces, and illegal characters are ignored. This routine alters FAC, TEM1, TEM2, and TEM3.

TTYOUT - Type a DBLFLT Number in Exponential Format

This routine will print out a DBLFLT number on an ASR33 teletype. Only the number itself is printed by TTYOUT; any desired formatting (including line feed or carriage return) must be done by the user. The printed number will be in the format:

±x.xxxxxxE±yyy

Enter via JMP TTYOUT with the number to be printed in the FAC. Return is to .+1 with C(AC) = Ø. This routine alters FAC, TEM1, TEM2, and TEM3.

TYP6 - Type 6-bit ASCII Character

Enter via JMP TYP6 with 6-bit ASCII code in bits 6-11 of accumulator and bits 0-5 set to zero. Returns to .+1 with C(AC) = Ø. Typing may be speeded up if the present instructions are replaced with those on the right below. However, the user program must then issue a TLS command before the first use of the teletype routine.

Present	Faster
TTYP8, SET 4	TTYP8, SET 4
Ø	Ø
IOB	IOB
6046/TLS	6041/TSF
IOB	JMP.-2
6041/TSF	IOB
JMP.-2	6046/TLS

TYP8 - Type 8-bit ASCII Character

Enter via JMP TYP8 with 8-bit ASCII code in bits 4-11 of the accumulator. Return is to .+1 with C(AC) = Ø.

APPENDIX II - TABLE OF DBLFLT 3 OPERATIONS

SUBROUTINE FUNCTION	DBL3GO ENTRY POINT	NUMBER OF ARGUMENTS	C(AC) USED?	OTHER SETUP NEEDED?	RETURNS TO .+	C(AC) AFTER RETURN	DBLFLT REGISTERS ALTERED	TYPICAL TIME (msec.)
Absolute value via IR.1	ABS1	Ø	no	IR.1[a]	1	Ø	viaIR.1[a]	0.35
Addition	ADDT	1,2,3	no	IR[a]	2,3,4	Ø	arg.	1.4-2.6
Arc sine of FAC	ARCSIN	Ø	no	F	1[e],2	≠Ø	F,TEM1-3	140.
Arc tangent of FAC	ARCTAN	Ø	no	F	1	≠Ø	F,TEM1-2	32.
Complement (negate) via IR.1	COM1	Ø	no	IR.1[a]	1	Ø	via IR.1[a]	0.35
Arithmetic comparison	COMPAR	1,2	no	IR[a]	2,3,4,5	≠Ø	none	0.35
Cos X (degrees)	COSDEG	Ø	no	F	1	≠Ø	F,TEM1-2	27.
Cos X (radians)	COSRAD	Ø	no	F	1	≠Ø	F,TEM1-2	27.
Cos X (π/2 radians)	COSPI2	Ø	no	F	1	≠Ø	F,TEM1-2	27.
Divide	DIVIDE	1,2,3	no	IR[a]	2,3,4	Ø	arg.	9.5
Compute e^X	EXPON	Ø	no	F	1	Ø	F,TEM1-3	6-23000
Fix into 24 bits	FIX	1	no	IR[a]	2	≠Ø:error	arg.	0.7-2.0
Fix into 12 bits	FIX12	Ø	no	F	1[e],2	number	F	0.8-1.5
Float a 24 bit number	FLOAT	1	no	IR[a]	2	Ø	arg.	0.8
Float a 12 bit number	FLOT12	Ø	yes	no	1	Ø	F	1.1
Load IR.1-11 & 13-16	IRLOAD	Ø	no	no	1	≠Ø	none	0.7
Read 8-bit ASCII (no waiting)	KBD		no	no	1	ASCII	none	0.3
Read 8-bit ASCII (wait)	KBDI	Ø	no	no	1	Ø,ASCII	none	---

APPENDIX II - TABLE OF DBLFLT 3 OPERATIONS (page 2)

SUBROUTINE FUNCTION	DBL3GO ENTRY POINT	NUMBER OF ARGUMENTS	C(AC) USED?	OTHER SETUP NEEDED?	RETURNS TO .+	C(AC) AFTER RETURN	DBLFLT REGISTERS ALTERED	TYPICAL TIME (msec.)
Line feed & carriage return	LFCR	Ø	no	no	1	Ø	none	300.
Log base 2 of X	LOG2	Ø	no	F	1[e],2	≠Ø,Ø	F,TEM1-3	50.
Log base 10 of X	LOG10	Ø	no	F	1[e],2	≠Ø,Ø	F,TEM1-3	50.
Log base e of X	LOGN	Ø	no	F	1[e],2	≠Ø,Ø	F,TEM1-3	50.
Test absolute value	MAGTST	Ø	no	F	1,2	≠Ø	none	0.2
Multiply	MULT	1,2,3	no	IR[a]	2,3,4	Ø	arg.	1.8
Complement (negate) FAC	NEGFAC	Ø	no	F	1	≠Ø	F	0.2
Power series	POWSER	Ø	yes	IR.4[b]	1	Ø	F,TEM2	3.5/term
Test sign of FAC	SIGN	Ø	no	F	1,2,3	≠Ø	none	0.2
Sin X (degrees)	SINDEG	Ø	no	F	1	≠Ø	F,TEM1-2	27.
Sin X (radians)	SINRAD	Ø	no	F	1	≠Ø	F,TEM1-2	27.
Sin X (π/2 radians)	SINPI2	Ø	no	F	1	≠Ø	F,TEM1-2	27.
Square root of FAC	SQROOT	Ø	no	F	1 ,2	≠Ø	F,TEM2-3	110.
Subtraction	SUBT	1,2,3	no	IR[a]	2,3,4	Ø	arg.	1.4-2.7
Raise 10 to the power N	TEN2N	Ø	yes	no	1	≠Ø	F,TEM1-2	4.0-40.
Transfer DBLFLT number	TRANS	1,2	no	IR[a]	2,3	Ø	arg.	0.66

APPENDIX II - TABLE OF DBLFLT 3 OPERATIONS (page 3)

SUBROUTINE FUNCTION	DBL3GO ENTRY POINT	NUMBER OF ARGUMENTS	C(AC) USED?	OTHER SETUP NEEDED?	RETURNS TO .+	C(AC) AFTER RETURN	DBLFLT REGISTERS ALTERED	TYPICAL TIME (msec.)
Type in a number	TTYIN	Ø	ASCII	no	1[e],2	Ø	F,TEM1-3	---
Print out a number	TTYOUT	Ø	no	F	1	Ø	F,TEM1-3	1400
Print 6-bit ASCII	TYP6	Ø	yes	no	1	Ø	none	100.
Print 8-bit ASCII	TYP8	Ø	yes	no	1	Ø	none	100.

arg) These operations alter DBLFLT3 registers as specified by the user program arguments.

a) in segment SEGA

b) in segment SEGB

e) return to this location indicates error, i.e., overloading or illegal number.

F) uses or alters contents of FAC

IR[a]) Any index registers listed as arguments must be set (in segment SEGA) before entry.