LaFaw Stuart

BULLETIN 3028

Preliminary Edition

DATATRON

ELECTRONIC DATA PROCESSING SYSTEMS

HANDBOOK

FLOATING POINT CONTROL UNIT

Model 360

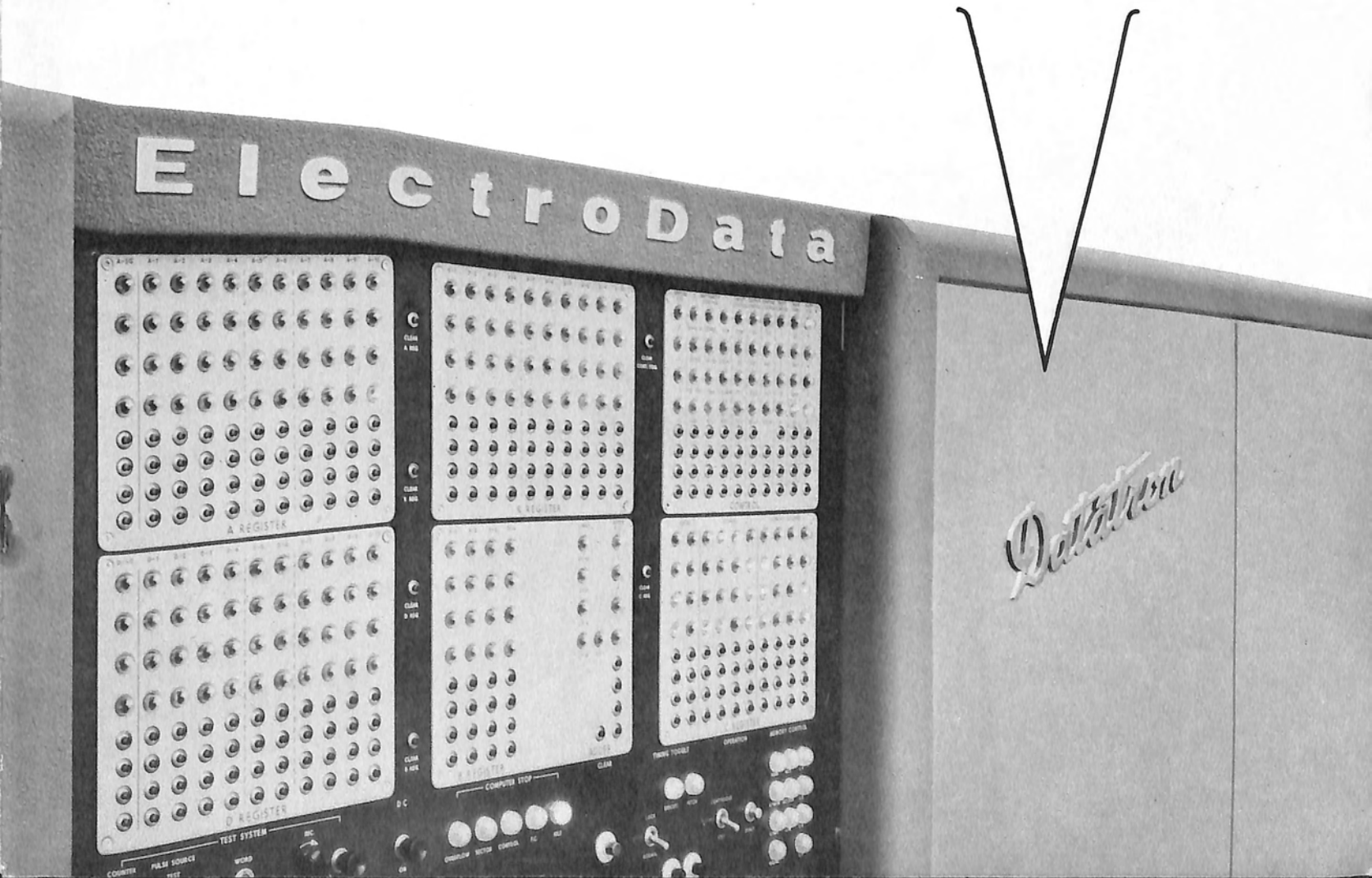
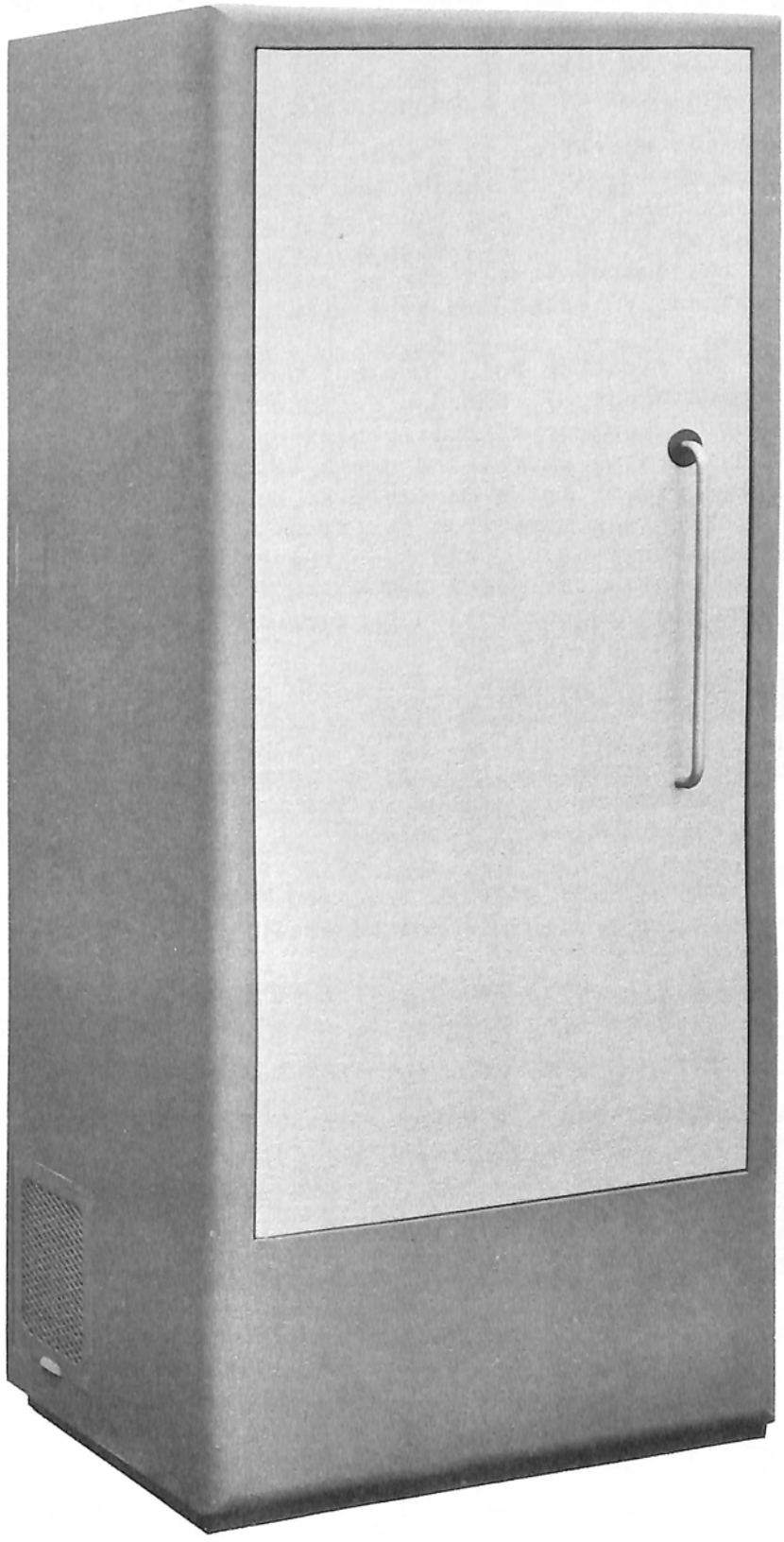


TABLE OF CONTENTS



Model 360 Floating Point Control Unit. Figure 1

MODEL 360 FLOATING POINT CONTROL UNIT

GENERAL

Floating point operation is a method of performing arithmetic computation with numbers which vary in magnitude over a relatively wide range. The magnitude of the number is recorded as an exponent of the base 10 which is carried with the significant digits of the number itself during calculation. Thus, results are automatically scaled and kept within the range of the computer.

The Model 360 Floating Point Control Unit provides the DATATRON with a separate set of commands for the floating point operations of addition, subtraction, multiplication, and division. The cabinet is the same height and depth as the central computer and is 32 inches wide. It is designed to be attached to the left end of the DATATRON, as seen from the front. The DATATRON motor-generator, power control, and line regulator have enough surplus capacity to handle the Model 360 which requires 1.5 kva. Operation of the unit is controlled by computer commands.

ADVANTAGES OF FLOATING POINT OPERATION

Significant programming, coding, and economic advantages are available to many applications by the use of the Floating Point Control for the following reasons:

(1) The area of drum storage required for programs will be less since coding for scaling is not necessary.

(2) The time to code a program will be reduced because the most difficult portions of a program to code are those requiring scaling.

(3) The computer time to debug a program will be reduced because of the shorter program and because of the frequency of coding errors in manual scaling.

(4) The analytical skill needed for scaling a calculation will be largely eliminated.

(5) The machine running time will be reduced, since the portions of a program that require scaling are time consuming.

CHARACTERISTICS OF A FLOATING POINT NUMBER

The structure of a number to be used by the Floating Point Control Unit is as follows:

The mantissa, or numerical value, is stored in the last eight digit positions of the computer word. Non-zero numbers range from .10000000 to .99999999.

The exponent of the base, which indicates the magnitude of the number, is stored in the first two digit positions of the computer word. It may range from 00 to 99 which is interpreted as -50 and 49.

The sign of the mantissa occupies the sign position of the word.

The exponent index of a number which falls within the range from .10000000 to .99999999 is zero, which is written as 50. Numbers greater or less than this range can be written in the same number of digits by using a suitable power of 10. For example, 12.345678×10^{0} can be written as $.12345678 \times 10^{2}$. The exponent index of any number is obtained by adding the appropriate power of ten to 50. Figure 2 contains a table of examples of numbers in the floating point notation.

Any number which falls within the range 10^{-51} to 10^{49} may be represented in the floating point structure. Zero is represented as a zero exponent and mantissa. Its sign may be plus or minus, following the same rules as in fixed point operation.

Fixed Point Number	Power of Ten	Floating Point Structure
+.12345678	0	+5012345678
-.12345678	0	-5012345678
+.00012345678	-3	+4712345678
-.123.45678	3	-5312345678
+ 1234567800000	13	+6312345678

Examples of Floating Point Structure. Figure 2

SIGNIFICANCE OF RESULTS OF FLOATING POINT OPERATIONS

The results of floating point operations may appear to have more significance than actually exists. In a fixed point operation, the scaling involved is known. In floating point operation, the scaling is performed automatically and is not readily available for inspection. Unless a scaling analysis is performed, it is almost impossible to state how significant a given result is. Consider the following example of multiplication following floating point subtraction:

	Exponent	Mantissa
	60	31704162
(-)	60	31704168
(=)	-53	60000000
(x)	60	70134061
(=)	-63	42080436

If the final result is the only factor available for inspection, it will be interpreted as -420904360000. However, the subtraction produced a result which has, at most, one significant digit; hence, the product has, at most, one significant digit. The result should be interpreted as -4000000000000, where the 4 is rather doubtful.

COMMANDS

GENERAL

The floating point commands for addition, subtraction, multiplication, and division all have the following features in common:

- All will cause the overflow in the DATATRON to be set if the range is exceeded. The sign will be set positive on overflow.
- All will cause the A and R Registers to be cleared if the result is less than 10^{-51}. This is referred to as underflow.
- Any floating point command will destroy the contents of the special counter.
- All floating point operations leave their mantissas in a normalized form; that is, within the range .10000000 to .99999999.

FLOATING POINT ADD (FAD)

The execution of this command adds the floating point number in the cell specified by the operand address to the floating point number in the A Register. The command takes the form:

s 000p 80 xxxx

s is the sign position. (See B Modification in the Central Computer Handbook.)

000 is of no significance to this command.

p is the breakpoint position. (See Central Computer Handbook.)

80 is the numerical operation code for Floating Point Add.

xxxx is the address of the floating point number to be added to the floating point number in the A Register.

Location	Operation	Operand Address	Remarks
7001	CAD	2000	Clear A and set to floating point number
7002	FAD	1000	Floating Point Add second number.
7003	ST	3000	Store result.

A Register after Execution of Instruction at 7001	Floating Point Number in Cell 1000	Result in A Register	
0 80 10000000	0 51 10000000	0 80 10000000	
0 80 90000000	0 51 20000000	0 80 90000000	
1 51 20000000	1 51 90000000	1 52 11000000	
1 51 20000000	0 49 20000000	1 51 19800000	
0 31 20000000	1 31 20000000	1 00 00000000	
1 31 20000000	0 31 20000000	0 00 00000000	
0 99 90000000	0 99 10000000	0 01 00000000	(overflow)
1 99 90000000	1 99 10000000	0 01 00000000	(overflow)

Examples of Floating Point Add. Figure 3

The R Register is left undisturbed during the execution of the Floating Point Add command unless the result is less than 10^{-51}, in which case the A Register and the R Register are cleared to zeros.

A Floating Point Clear and Add or Clear and Add Absolute value can be executed with the existing fixed point commands.

FLOATING POINT SUBTRACT (FSU)

The execution of this command subtracts the floating point number in the cell specified by the operand address from the floating point number in the A Register. The command takes the form:

s 000p 81 xxxx

s is the sign position. (See B Modification in the Central Computer Handbook.)

000 is of no significance to this command.

p is the breakpoint position. (See Central Computer Handbook.)

81 is the numerical operation code for Floating Point Subtract.

xxxx is the address of the floating point number to be subtracted from the floating point number in the A Register.

Location	Operation	Operand Address	Remarks
7001	CAD	2000	Clear the A Register and set to a floating point number.
7002	FSU	1000	Floating Point Subtract the number in cell 1000.
7003	ST	3000	Store the result in cell 3000.

A Register after Execution of Instruction at cell 7001	Floating Point Number in Cell 1000	R Result Stored	
1 60 20000000	1 60 10000000	1 60 10000000	
1 60 20000000	0 60 10000000	1 60 30000000	
1 60 20000000	1 60 90000000	0 60 70000000	
1 01 20000000	1 01 90000000	0 01 70000000	
0 49 30000000	0 52 40000000	1 52 39970000	
1 30 20000000	1 20 10000000	1 30 20000000	
1 30 20000000	1 30 20000000	0 00 00000000	
1 99 90000000	0 99 40000000	0 01 30000000	(overflow)
0 51 12345678	0 50 20000000	0 51 10345678	

Examples of Floating Point Subtract. Figure 4

The R Register is left undisturbed during the execution of the Floating Point Subtract command unless the result is less than 10^{-51}, in which case the A Register and the R Register are cleared to zeros.

A Floating Point, Clear and Subtract, or Clear and Subtract Absolute value can be executed with the existing fixed point commands.

FLOATING POINT MULTIPLY (FM)

The execution of this command uses the floating point number in the A Register as the multiplier and the floating point number in the cell specified by the operand address as the multiplicand. The eighteen digit floating point product is inserted in the A and R Registers, the most significant digits being in the A Register. The command takes the form:

s 000p 82 xxxx

s is the sign position. (See B Modification in the Central Computer Handbook.)

000 is of no significance to this command.

p is the breakpoint position. (See Central Computer Handbook.)

82 is the numerical operation code for Floating Point Multiply.

xxxx is the address of the floating point multiplicand.

The R Register is automatically cleared before the operation.

Location	Operation	Operand Address	Remarks
7001	CAD	2000	Clear A and set to floating point multiplier.
7002	FM	1000	Multiply using floating point number at cell 1000 as multiplicand.
7003	ST	3000	Store result.

A Register after Execution of Instruction at 7001	Floating Point Multiplicand	Operation result in A and R Registers	
0 55 20000000	0 55 20000000	0 59 40000000	0000000000
0 55 20000000	1 55 20000000	1 59 40000000	0000000000
0 40 20000000	0 60 20000000	0 49 40000000	0000000000
1 40 20000000	1 60 20000000	0 49 40000000	0000000000
0 80 20000000	0 80 20000000	0 00 20000000	0000000000 (overflow)
0 51 20000000	0 51 12345678	0 51 24691356	0000000000
0 51 22222222	0 51 11111111	0 51 24691357	5308642000

Examples of Floating Point Multiply. Figure 5

With certain combinations, a Floating Point Multiply may set the overflow in the DATATRON despite the fact that the answer would be within the range of the computer. The spurious overflow is due to the fact that the machine approximates the answer exponent by first performing the indicated operation, next going on with the mantissa, and then modifying the answer exponent. For example, an overflow will be indicated if the sum of the exponents in their coded form is equal to or greater than 150; therefore, a spurious overflow will be indicated when the mantissa product is less than .10000000 and the exponent sum is 150. This is shown in the following examples:

Machine Representation		True Result	Machine Result
(8020000000)	(7040500000)	10^{49} X .81	Overflow
(7990000000)	(7090000000)	10^{49} X .81	9981000000

FLOATING POINT DIVIDE (FDIV)

The execution of this command divides the eighteen digit floating point number in the A and R Registers by the floating point number in the cell specified by the operand address. The command takes the form:

s 000p 83 xxxx

s is the sign position. (See B Modification in the Central Computer Handbook.)

000 has no significance for this command.

p is the breakpoint position. (See Central Computer Handbook.)

83 is the operation code for Floating Point Divide.

xxxx is the address of the floating point divisor.

After execution of the command, the quotient will be in the A Register. The R Register will contain the remainder and the last one or two digits of the quotient. The content of the R Register is determined according to the following rules:

(1) If the absolute value of the mantissa in the dividend is less than the absolute of the mantissa in the divisor, then the digit in the most significant position of the R Register is the least significant digit (ninth) of the quotient. It is followed by two zeros and the seven most significant digits of the remainder. The eighth and last digit of the remainder is lost.

(2) If the absolute value of the mantissa in the dividend is equal to or greater than the absolute value of the mantissa in the divisor, then the digits in the two most significant digits of the R Register are the two least significant digits of the quotient (ninth and tenth). They are followed by two zeros and the six most significant digits of the remainder. The last two digits of the remainder are lost.

Location	Operation	Operand Address	Remarks
7000	CR	0000	Clear the R Register.
7001	CAD	2000	Clear the A Register and set to the floating point number.
7002	FDIV	1000	Divide by the floating point divisor.
7003	ST	3000	Store quotient.

A and R Registers after Execution of the Command at 7001	Floating Point Divisor at Cell 1000	A and R Registers after Execution of the Command at 7002	
0 54 80000000 0000000000	0 52 20000000	0 53 40000000 0000000000	
1 08 40000000 0000000000	0 04 20000000	1 55 20000000 0000000000	
0 10 40000000 0000000000	0 50 20000000	0 11 20000000 0000000000	
0 50 40000000 0000000000	1 50 30000000	1 51 13333333 3300100000	
0 50 30000000 0000000000	0 50 40000000	0 50 75000000 0000000000	
0 50 10000000 0000000000	0 50 30000000	0 50 33333333 3001000000	
0 80 50000000 0000000000	0 20 50000000	0 00 50000000 0000000000	(overflow)

Examples of Floating Point Divide. Figure 6

With certain combinations, a Floating Point Divide may clear the A Register and the R Register despite the fact that the answer would be within the number range of the machine. The spurious underflow is due to the fact that the machine approximates the answer exponent by first performing the indicated operation on the exponent and then the mantissa and finally modifying the answer exponent. The A and R Register will be cleared if the exponents differ by 51, with the exponent in the divisor being greater and with the quotient mantissa equal to or greater than .10000000. This is shown in the following example.

Dividend	Divisor	Machine Result	True Result
0 09 20000000	0 60 10000000	0 00 00000000	10^{-50} X .2
0 09 16000000	0 59 80000000	0 00 20000000	10^{-50} X .2

The Floating Point Divide will produce no spurious overflow.

The R Register should be cleared before a Floating Point Divide unless an 18 digit dividend is to be used. The examples in Figure 7 illustrate the use of a cleared R Register and also of an 18 digit dividend.

Contents of A and R Registers before Floating Point Divide	Floating Point Divisor	Contents of A and R Registers after Execution of FDIV
0 50 33333333 3333333333	0 50 60000000	0 50 55555555 5003333333
0 50 33333333 0000000000	0 50 60000000	0 50 55555555 0000000000
0 52 20000000 8000000000	1 52 40000000	1 50 50000002 0000000000
0 52 20000000 0000000000	1 52 40000000	1 50 50000000 0000000000
1 52 88888888 8888888888	0 56 40000000	1 47 22222222 2200088888
0 52 88888888 0000000000	0 56 40000000	0 47 22222222 0000000000

Examples of Floating Point Divide. Figure 7

TIMING

In general, the time required to perform an arithmetic floating point operation will be equal to or less than its fixed point counterpart. For instance, the Floating Point Multiply and Floating Point Divide are faster than their fixed point counterparts for two reasons. First, since the operands are only eight decimal digits, the average number of partial additions during multiply and partial subtractions during divide is reduced an average of nine "add-times" or 1.5 milliseconds. Second, a further reduction is made if either floating point operand is zero during multiply, or if the dividend is zero during divide. In these cases, the operation is completed within seven microseconds after the order is started. This speed-up becomes significant in a large class of problems where a number of factors are zero.

The time needed to perform floating point add or subtract is increased by one word-time (85 microseconds) as compared with fixed point. The average time required to execute floating point commands, including access time for both command and operand, is given in Figure 8. Use of the high speed loops of the DATATRON is assumed.

Floating Point Operation	Average Execution Time in Milliseconds
Add or Subtract	2.5
Multiply	10.1
Divide	13.5

Average time for Floating Point Operations. Figure 8

EXAMPLE OF FLOATING POINT USE

Solve for x in the formula

$$x = \frac{ab}{c} + d - r$$

where		Machine Coded Floating Point Number	Storage Location
a =	222.2222200	0 53 22222222	6000
b =	8.8800000	0 51 88800000	6001
c =	.0000700	0 46 70000000	6002
d =	314.36210	0 53 31436210	6003
r =	-4123.00000	1 54 41230000	6004

The R Register is cleared to zeros.

Instruction Location	Operation	Operand Address	A and R Registers after Execution of the Instruction
7000	CAD	6000	0 53 22222222 0000000000
7001	FM	6001	0 54 19733333 1360000000
7002	FDIV	6002	0 58 28190475 9000600000
7003	FAD	6003	0 58 28190789 9000600000
7004	FSU	6004	0 58 28194912 9000600000
7005	ST	6005	0 58 28194912 9000600000
7006	STOP	0000	0 58 28194912 9000600000

ROUNDOFF AFTER FLOATING POINT MULTIPLY OR DIVIDE

A roundoff operation after a Floating Point Multiply or Divide is usually irrelevant. The amount of significance gained by a roundoff is negligible when compared with the amount of significance lost by a single addition or subtraction operation. For example, the maximum amount of significance lost by not rounding during one thousand successive multiplication operations is three digits, whereas one subtraction or addition can cause a loss of all significance in one operation.

If a roundoff is desirable, however, it can be accomplished in the following way.

Assuming that the A and R Registers contain the product or quotient, the following commands accomplish the roundoff.

```
RO 0000
SL 0002
CNZ ──┐
UA 0000 │
SL 0018 ◄┘
```

This short routine must be used rather than a single RO command because errors can arise in cases like the following.

Assume that the product in the A and R Registers is

+ 52 99999999 7500000000

After an RO command, the result would appear as

+ 53 00000000 0000000000

This is incorrect. The result should be

+ 53 10000000 0000000000

The given routine allows for this possibility.

EXTRACTING INTEGRAL VALUES

In cases where only the integral value of the result is desired, a + 58 00000000 should be added. This will cause the number in the A Register to be shifted right until only its integral part remains in the A Register. This integral value will then be normalized to produce the final result. Figure 9 illustrates this operation.

A Register Before Operation	A Register After an FAD Using + 58 00000000 as Operand
+53 12345678	+53 12300000
-49 12345678	+00 00000000
-56 12345678	-56 12345600

Examples of Extraction of Integral Values. Figure 9

TOLERANCE CHECKING IN USING FLOATING POINT OPERATION

Assume that the difference between two numbers, a and b, is to be less than a given amount, c. The usual fixed point method of checking tolerance is to subtract b from a and test against c.

$$|a - b| - c$$

However, this method is not feasible when numbers are in the Floating Point structure because the exponent of $|a - b|$ may differ from that of c to such a degree that they are not comparable. One method which may be used with Floating Point numbers is as follows:

$$\left|\frac{a - b}{a}\right| - c$$

This automatically scales the exponent down by dividing. The coding for this operation is given below.

Location	Command	Remarks
7000	CAD 6000	Bring a to A Register.
7001	FSU 6001	Subtract b.
7002	FDIV 6000	Divide by a.
7003	EX 7018	Remove sign.
7004	FSU 6002	Subtract c.
7005	OSGD 700X	Test sign.
7006	CC	Branch.
7018	0 1111 11 1111	

CONVERSION OF FLOATING POINT NUMBERS TO FIXED POINT NOTATION

In some problems, it is desirable to convert Floating Point numbers to fixed point numbers before print-out or some other operation. A routine for this conversion is given below. The original Floating Point number is assumed to be in location 6000. The integral part of the result is stored in 6001, the fractional part in 6002. The range of Floating Point numbers which this routine is capable of handling is therefore $10^{-10} \leq N < 10^{10}$.

Location		Command	Remarks
7000		CADA 6000	Bring Floating Point number to A Register.
7001		AD 6003	Add 3900000000. (Test oversize)
7002		CC 7019	To STOP if oversize.
7003		AD 6004	Add 2100000000 (Test undersize)
7004		CC 7006	Change control if within range.
7005		CU 7018	Exit if undersize.
7006		SR 0008	Shift mantissa to R Register.
7007		STC 6006	Temporarily store shift count.
7008		CAD 6005	Bring shift command to A Register.
7009		SU 6006	Subtract shift count.
7010		STC 7014	Store shift command.
7011		CAD 6000	Set A Register to correct sign.
7012		SL 0010	Shift mantissa into A Register.
7013		CR 0000	Clear R.
7014		(SR 00XX)	Stored shift command.
7015		ST 6001	Store integral result.
7016		SL 0010	Shift fractional result to A Register.
7017		ST 6002	Store fractional result.
7018	0001	CU XXXX	EXIT (with breakpoint)
7019		STOP 0000	STOP if oversize.

Constants

	Floating Point number	
6000		
6001	0000000000	Integral result
6002	0000000000	Fractional result
6003	3900000000	
6004	2100000000	
6005	0000130020	
6006	0000000000	

If the number to be converted is 10^{10} or greater, the routine will stop. (An exit command can be substituted for the STOP command, if desired.) If the number is too small, the normal exit is used with zeros left as the converted value. In every case, the exit is a CU command with a breakpoint 1 for stopping if desired.

CONVERSION OF FIXED POINT NUMBERS TO FLOATING POINT NOTATION

In some problems, it is desirable to change a number in fixed point notation to one in floating point notation so that calculation may be performed in the floating point mode. A code for converting fixed point numbers to floating point is given below. The fixed point number to be converted is assumed to be in location 6000, and a scaling constant is located in 6001. This constant gives the desired position of the decimal point in the A Register. If this figure is positive, it indicates that the decimal point is the given number of positions to the right of its normal position. If this figure is negative, it indicates that the decimal point is the given number of positions to the left of its normal position.

For example:

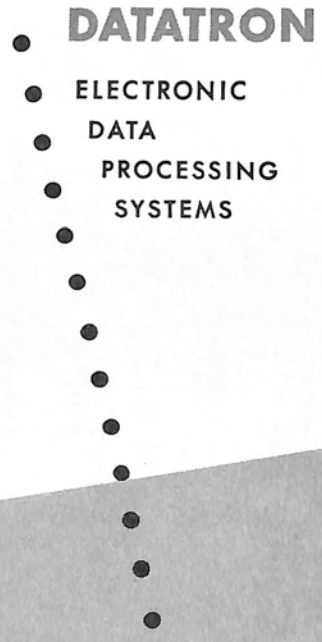
Figure in 6001	Point Position in A Register
+ 0000000000	.xxxxxxxxxx
+ 0000000003	xxx.xxxxxxx
+ 0000000007	xxxxxxx.xxx
+ 0000000010	xxxxxxxxxx.
- 0000000004	.0000xxxxxxxxxx
- 0000000009	.000000000xxxxxxxxx

The floating point result is stored in location 6002.

Location		Command	Remarks
7000		CAD 6000	Bring fixed point number to A Register.
7001		NOR 7001	Normalize.
7002		SR 0010	Shift number into R Register.
7003		CAD 6001	Bring scaling constant to A Register.
7004		AD 6003	Add 50 to scaling constant. (Overflow if too great.)
7005		CC 7016	To STOP if too great.
7006		EX 6004	Extract exponent.
7007		SUSC 0000	Correct for normalizing.
7008		OSGD 6003	Test undersize.
7009		CC 7017	Change control if undersize.
7010		SR 0002	Shift exponent into R Register.
7011		CAD 6000	Bring fixed point number to A Register.
7012		EX 6005	Extract sign.
7013		SL 0010	Shift Floating Point number to A Register.
7014		ST 6002	Store Floating Point number.
7015	0001	CU XXXX	EXIT (with breakpoint)
7016		STOP 0000	STOP if too great.
7017		SR 0010	Clear A Register if undersize.
7018		CU 7014	Return.

Constants

6000	Fixed Point number
6001	Scaling constant
6002	Floating Point result
6003	9999999950
6004	0000000011 Extractor
6005	1 0000000000 Extractor



ElectroData

DIVISION OF BURROUGHS CORPORATION

460 SIERRA MADRE VILLA, PASADENA, CALIFORNIA

Printed in U.S.A.