I·C·T

COMPUTER EQUIPMENT GROUP

I.C.T. 1900 SERIES

1905/9 FLOATING POINT UNIT

PROVISIONAL DESCRIPTION

SEPTEMBER 1966.

I.M. 47.
ISSUE 1.

INTERNATIONAL COMPUTERS AND TABULATORS LIMITED

Thomas Street West Gorton Manchester 12

I.C.T. 1900 SERIES

1905/9 FLOATING POINT UNIT

PROVISIONAL DESCRIPTION

SEPTEMBER 1966. I.M. 47.
ISSUE 1.

CONTENTS

CHAPTER 1

1905/9 FLOATING POINT UNIT

1.1 Introduction

1905 and 1909 type processors incorporate a hardware unit which is capable of performing floating point arithmetic operations autonomously, thus leaving the central machine microprogram free to proceed simultaneously with non-floating-point operations. The unit contains its own clock system, control bistables, microprogram and separate argument and exponent arithmetic units. When the floating point unit is inoperative, its clock system runs in synchronism with the central machine clock. On the occurrence of a floating point operation in a program, a pair of operands are transferred between the central machine store and the F.P. unit, which on receipt of the second operand, adopts a 'busy' status. The F.P. unit clock then assumes its operational frequency of 1Mc/s and the unit performs the specified calculation, independently of central machine clock operation, at a digit time of 1μsec.

At the completion of a calculation, which includes optional rounding and/or normalising, the unit gives a 'not busy' indication and waits for the next program instruction which may be an order to store a resultant operand or may be a further arithmetic operation.

Following activation of the F.P. unit, the central machine will thus either proceed to carry out the next program instruction (if non floating point), or, on the occurrence of a further floating point instruction, will be held in a waiting loop until the F.P. unit indicates 'not busy'.

1.2 Floating Point number representation.

1905/9 floating point number representation is as shown below in Fig. 1.

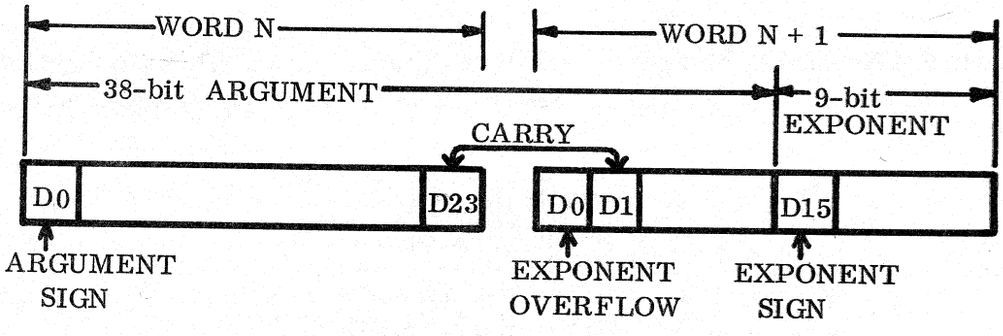


Fig. 1. Number Representation

Each operand is held in double length form and consists of a 38-bit signed argument and a 9-bit exponent. Within the F.P. unit, the exponent is represented as a signed integer which can vary between the limits of +255 and -256. The latter value, together with a zero argument represents floating point zero which, if transferred to the central machine store in this form, would result in a '1' being placed in D8 of word N + 1. This would result in incompatibility between floating point zero and fixed point zero (all noughts), and would thus inhibit the use of the existing 'test for zero' functions in the case of floating point operands.

It is convenient, however, to transfer the exponent to store in the form e + 256 (where e = true exponent). this is a positive number in the range 0 to 511. This is effected by inversion of the exponent sign digit on exchanges between the F.P. unit and the store. By example the exponent 111110100 (-12), on transfer from the F.P. unit to the store, appears as 011110100 (+244) i.e. -12 + 256. Similarly a zero exponent (100000000) is transferred as a true zero.

Bit 23 of word N + 1 is set to indicate the occurrence of exponent overflow i.e., the condition under which the exponent becomes greater than +255.

1.3 Programming control

The following Group 13 functions are applicable to 1905/9 floating point operation.

130	130	0	M/N	n,n* → a.

Convert the mid-point number whose integral part is in N(M) and fractional part in N + 1 (M) to a normalised floating point number and load it to the F.P.A. N(M) and N + 1(M) are unchanged. X has no significance.

131	131	0	M/N	a → n, n*.

Convert and store the contents of the F.P.A. in N(M) and N + 1 (M) in mid-point form. Bit 23 of word N + 1 and central machine overflow will be set if exponent overflow has occurred or if the result exceeded capacity. The contents of the F.P.A. remain unchanged; X has no significance.

Functions 130 and 131 are performed by software. The arithmetic functions 132-135 are performed by F.P. unit hardware. X is used for function extension as indicated below.

X0	=	0	Result rounded.
	=	1	Result unrounded.
X1	=	0	Result normalised.
	=	1	Result not normalised.

X2 = 0 Normal function, e.g. a -n → a
= 1 Inverted function, e.g. n -a → a

132 132 X M/N a + n → a
Add the floating point number in N(M) and N + 1 (M) to F.P.A.

133 133 X M/N a - n → a
Subtract the floating point number in N(M) and N + 1 (M) from the F.P.A.

134 134 X M/N a. n → a
Multiply F.P.A. by the floating point number in N(M) and N + 1 (M)

135 135 X M/N $\frac{a}{n}$ → a
Divide F.P.A. by the floating point number in N(M) and N + 1 (M)

Additive values of X are permissible. It will be noted also that the value X2 = 1 has significance only in functions 133 and 135.

The remaining group 13 functions, 136 and 137, are performed by the central machine microprogram. Their X significance is as indicated below.

136 136 X M/N n: → a

X = 0 Load the normalised F.P. number in N(M) and N + 1 (M) to the F.P.A. The contents of N(M) and N + 1 (M) will be unchanged and exponent overflow will be set only if BO of N + 1 (M) is set.

X = 1 Clear the F.P.A. including exponent overflow. The contents of N(M), N + 1 (M) will be unchanged.

137 137 X M/N a → N:

X = 0 Store the contents of the F.P.A. in N(M), N + 1(M), leaving the F.P.A. unchanged. If exponent overflow has occurred BO of N + 1 (M) and hence V, will be set.

X = 1 As defined above, but the F.P.A. is cleared after the operation.

1.4 Floating point unit hardware.

The floating point unit consists of 11 hardware boxes which are normally housed in doors 3 and 4 of the central machine microprogram compartment. As in the case of the central machine, monitoring facilities for the F.P. unit registers, highways and microprogram are provided on the front panel.

For the purpose of logic circuit description and illustration the unit is considered as consisting of the following main sub-divisions:-

1. Arithmetic units (argument and exponent)
2. Clock system
3. Control circuitry
4. Microprogram
5. Interface with central machine microprogram.

The appropriate I.C.T. 1900 logic circuit drawing numbers are as follows:

07/0 - 07/37	Argument arithmetic unit slices 0-37
07/38	Argument arithmetic unit carry system
07/40 - 07/48	Exponent arithmetic unit slices 40-48
07/49	Exponent arithmetic unit final slice - overflow detection and carry system.
07/50	Clock generation; control circuits; F and X registers.
07/51	Add /Subtract microprogram
07/52	Multiply microprogram
07/53	Normalise microprogram
07/54	Division microprogram
07/55	Register/highway gating (argument arith. unit).
07/56	Register/highway gating (exponent arith. unit). CYBA0, CYBE0, FOVR, FPOVR.
L	Interface, F.P. unit/central processor microprogram.

1.5 Floating point arithmetic units

Diagram 1 shows a simplified representation of the 38-bit argument arithmetic unit and the 9-bit exponent arithmetic unit. Each unit has its own mill, registers and highways, and, apart from the omission of facilities for performing logical operations, is generally similar in circuit design to the central machine arithmetic unit. The following generalised description of digit flow within the F.P.U. will be made more complete by reference to the 'slice' logic drawings 07/0 - 07/49.

1.6 Loading of operands

The double length floating point operands are fed from the store to the F.P.U. via the SOUT lines in two parts. Thus on an initial load order (Fn 130 or Fn 136), the store word N, containing the sign and m.s. 23 bits of the operand argument, is fed via SOUT 0-23 to the FM highway lines (mill output) 14-37 and thence strobed directly to the corresponding FQ highway lines, FQ14-FQ37. The second part of the operand (store word N + 1) is subsequently transferred via SOUT 0-23 as follows.

The l.s. argument bits 0-13 are strobed via the argument mill output highway lines FM0-FM13 to FQ0-FQ13. Bits 0-8, of word N + 1, holding the 9-bit exponent are fed via the exponent mill output highway lines FM40-48 to the corresponding FP highway lines FP40-48.

From the FQ and FP highways the argument and exponent are then simultaneously strobed to the appropriate FPU registers, the registers FA, FE and FB, FD being used as argument/exponent pairs. The arrangement of the F.P.U. hardware is such that regardless of which of the four basic arithmetic operations is specified, the operation is performed by the process of adding the operand in registers FA, FE, or its negated form, to the operand in registers FB, FD, i.e. FA, FE is invariably either added to or subtracted from FB, FD. Since the initial load order (Fn 130 or Fn 136) does not specify which type of arithmetical operation is to be performed, and thus does not indicate whether an inverted function will follow, it loads the first operand into all the F.P.U. registers. The placing of the second operand is thus determined by the nature of the next order.

1.7 Mill operations

For mill operations, the FB register is gated directly via the K highway to the argument mill. The FA register may be gated to the mill via the L highway in any of the forms A, $\bar{A}$ or 2A, dependent on the type of operation being performed. Thus, for an addition the form A → L is used; for subtraction the gating is $\bar{A}$ → L together with the unity input CYBA0, the mill thus performing the operation B + (-A). The F.P.U. performs multiplication operations in a manner which is generally similar to the process used in the central machine arithmetic unit with the exception that the ultimate value of each partial product is determined by inspection of two bits of the multiplier instead of one. The second and third magnitudes of the multiplicand i.e. 2A and 3A, must therefore be made available for addition to the mill. The value 2A is obtained directly by gating A to L with a single left shift, i.e. $A_0 \rightarrow L_1$, $A_1 \rightarrow L_2$, etc. The method by which 3A is obtained will be detailed in the subsequent description of the multiplication microprogram.

The argument mill contains the extra slices FRMS38 and FRMS39 (see Diag. 7/49, for logic details) which normally contain the extension of the sign digit thus providing 'true sign' indication on the occurrence of overflow on a mill operation. The provision of two extra bits is necessary in this case since the 2A → L facility shifts the sign digit A37 into M38 leaving the possibility of a further left shift occurring due to overflow. FRMS39 thus always indicates the true sign of an operation and is set into the extra FB register stage FB38 whenever an FQ to FB strobe occurs. The significance of this feature will be shown in the division microprogram description.

The argument mill output may be gated to the FQ highway directly or shifted as indicated in Diagram 1.

1.8 FC register

The FC register consists of triple entry flip-flops with facilities for single left shift or double right shift. These enable the register to be utilised for specialised functions in the multiplication and division microprogram sequences. On unloading operations (Fn 131 or Fn 137) the operand argument is fed from the FC register via the BIN lines to the central processor, the transfer being effected in two parts as in the loading process.

1.9 Exponent mill

The exponent registers FE and FD may be fed directly or in negated form via their respective highways FO and FN to the exponent mill. In addition to their normal exponent evaluation functions these registers carry out the processing of operational counts during arithmetical calculations in a manner which is analogous to N-register operation in the central processor arithmetic unit. The mill output is fed directly to the FP highway and may thence be gated directly to registers FE and/or FD.

Both the FD register and the mill contain an extra slice for overflow detection, the true sign of any operation being indicated by FRMS 49 which is copied into FD49 whenever an FP → FD strobe occurs. On unload operations the exponent part of the operand is transferred to store via the FD register, the BIN lines (simultaneously with the l.s. argument bits from the FC register), the central processor B register and the SOUT highway.

1.10 F.P.U. mill carry system

Both the argument and exponent arithmetic units incorporate a block carry system similar in nature to that used in the central processor arithmetic unit.

The argument mill carry system consists of seven blocks each containing five slices and one block for the remaining three (m.s.) slices. The method of operation may be seen by first looking at the action within a typical block, e.g. mill slices 0-4 shown in drawings 07/0 - 07/04.

A carry into slice 0 will occur if -FCYBA0 is active at element A30/4 (Diag. 07/0). The carry into each subsequent slice depends on the state of -FCYBA0 and the sum and carry results produced by the preceding slices of the block. The result produced in each slice is indicated to following slices by means of the signals -FRMG (carry generate) and -FRMT (carry transmit). Thus, considering drawing 07/1, the carry input to slice 1 consists of the OR function formed by NOR elements A24/3 and A24/4. The gate will give a positive output signal on the activation of -FRMG0 (carry generated by slice 0) and/or -FCYBA0 and -FRMT0 (carry fed into and transmitted by slice 0).

The signal is fed via element A26/3 as a priming signal to element A26/5. The other input to this element is switched by A24/2 thus setting the mill output highway line FRMS1 to 1 when the K and L inputs (FRK1 and FRL1) are either 00 or 11. For K and L highway inputs of 01 and 10, element A24/2 inhibits element A26/5.

The element A26/1 is inhibited by the positive output of A24/3 and/or A24/4 when a carry input is active. When no carry input is active, element A26/5 is inhibited but A26/1 is primed and will give an active +FRMS1 output for either of the K, L highway input configurations 01 or 10.

The carry is similarly propagated through the remaining slices of the block, the carry input circuitry of each successive slice being extended to consider the results produced by all the preceding slices. For example, considering drawing 07/4, the carry input to slice 4 consists of five NOR elements forming an OR function which is activated under any of the following conditions:-

1) -FCYBA0 is active and slices 0-3 all transmit a carry.

2) Slice 0 generates a carry and slices 1-3 transmit a carry.

3) Slice 1 generates a carry and slices 2 and 3 transmit a carry.

4) Slice 2 generates and slice 3 transmits a carry.

5) Slice 3 generates a carry.

Inter-block carry is effected by means of the decoder circuits shown in drawing 07/38. Each of the first seven carry blocks has an associated decoder circuit which on sensing a carry transmit out of the block, produces a carry input to the next block. The coupled NOR elements J20/3 and J20/4 thus feed the active carry input -FCYBA1 to the first slice of the second block (drawing 07/5) whenever the first block transmits a carry. Since the active output signal is negative, the condition for producing it is that each of the decoder elements has at least one positive input. This condition is satisfied by the decoder input configuration.

a_v (p & +FCYBA0) = 1

The signal a is derived from the OR function formed by the coupled NOR elements H21/3 - H21/2, which senses a carry generated in and propagated through the block. Hence the output is positive if any slice generates a carry and all the subsequent slices transmit carry. Signal p, derived from element H19/4, is positive when all the slices of the first block transmit a carry and will thus activate -FCYBA1 if +FCYBA0 is also active at gate J20/3, i.e. if there was a carry into slice 0.

This action is continued from block to block through the mill by subsequent decoder circuits which produce the signals -FCYBA2 to -FCYBA7.

1.11 Exponent mill carry system.

The exponent mill carry system consists of three blocks each of three slices and uses the same operational principles as the argument mill carry system. The signal -FCYBE0 is the unity input to the first exponent mill slice (drawing 07/40). The inter-block carry decoder circuitry is shown in drawing 07/49, signals -FCYBE1 and -FCYBE2 providing the carry input to the second and third blocks respectively.

1.12 Operation counting

The exponent mill performs the dual functions of exponent evaluation and operation counting, the former function being carried out at an early stage in any arithmetic operation, the mill then being available for counting purposes. Thus, for example, floating point addition/subtraction is effected by addition or subtraction of the operand arguments in terms of the larger exponent. Before carrying out the argument arithmetic it is therefore necessary to extract the exponent difference and store it as a count of the number of compensatory shifts that will be required to adjust the argument associated with the smaller exponent. The following example will clarify the method of mill operation.

Example

The operands held in registers FB/FD and FA/FE, as shown below, are to be added.

	Arg.	Exp.
FB/FD	001101	0111
FA/FE	011101	0011

The exponent difference is initially extracted in all cases by subtracting FE from FD thus,

$$FD \rightarrow FN$$
$$\overline{FE} \rightarrow FO$$
$$FCYBE0 \rightarrow \text{Mill.}$$

The resultant mill sum, 0100, is returned via the FP highway to register FE which is then used as a shift counter during the subsequent argument adjusting process. Since FE is always subtracted from FD, the sign of the exponent difference may be positive or negative, dependent on the relative magnitudes of the two exponents. Thus when FD > FE, the exponent difference is positive and the argument in register FA is adjusted. For FE > FD, the exponent difference is negative and the argument in register FB is adjusted.

The required number of right shifts are then applied by counting register FE to zero. In the case where FD > FE, this is effected by the mill addition of FD and $\overline{FD}$ (effectively -1) to FE. Where FE > FD, i.e., exponent difference is negative, addition of FE and FCYBE0 (+1) is effected. After adjustment the arguments are added, the resultant, together with the larger exponent, being the answer.

1.13 Special counts

For multiplication, division and normalising operations, special counts of +16, +36 and -37 respectively, are required. These are obtained by the usual method of forming an "all ones" pattern in the mill, i.e. by gating FE and $\overline{FE}$ to FO or FD and $\overline{FD}$ to FN but with certain bits suppressed.

The constant +16 is thus obtained by gating FE and $\overline{FE}$ to FO (see drawing 07/40-07/48) with the waveforms +FCSP0 and FCSP1 active. These waveforms inhibit all the FE register stages except E44 (drawing 07/44) which thus sets a 1 into the corresponding mill position. The count is then set into registers FE and FD via the FP highway. The signals +FCSP0 and +FCSP1 are produced on the setting of the FSP bistable (drawing 07/52).

The constant +36 is obtained by gating FD and $\overline{FD}$ to FN with all bits except D42 and D45 suppressed.

The normalising constant -37 is obtained by gating FE and $\overline{FE}$ to FO, digits E42 and E45 being inhibited by waveform +FCCK.

CHAPTER 2

F.P.U. CLOCK SYSTEM

2.1 Operation

During periods of activity, the F.P.U. is driven by its own clock system at a digit time of 1μs.

At the end of each floating point operation the free running action of the clock is inhibited and the F.P.U. is driven by central machine clock pulses, the unit remaining in a waiting loop for a further floating point instruction.

The transitory action of the F.P.U. clock is controlled by a set of indicators within the unit, (END, BUSY and WAIT) and also depends on certain central machine conditions as detailed below.

Thus for free running F.P.U. clock operation,

(1) The BUSY indicator must be set,

(2) The central machine clock must be operating in fixed mode and must be set to either RUN or ONE INSTRUCTION.

The BUSY indicator becomes set on the loading into the F.P.U. of the second of a pair of floating point operands, i.e. on the occurrence of a function in the range 132 - 135. Assuming that central machine clock settings are as stated above, the setting of BUSY initiates F.P.U. clock operation. At the completion of the specified operation the END indicator will be set, and BUSY will be reset thus terminating F.P.U. clock operation. The WAIT indicator will also be set and the F.P.U. will maintain a waiting status under central machine clock control.

2.2 Logic description

The following logic circuit description is based on drawings $07/50_0$, $07/50_1$ and $07/50_2$ which cover the F.P.U. clock control, derived clock signal circuits and F, X registers.

2.3 Clock control circuit (Drawing $07/50_0$)

The F.P.U. clock circuit consists of the loop of elements F12/1, F7/4, F11/2 and F7/3. F12/1 is the control element, its inputs registering the various conditions on which free-running oscillation depends. When these conditions are satisfied all inputs to F12/1 are negative and the free-running clock circuit action is as follows.

The positive output voltage occurring at F12/1 is inverted by F7/4 and applied as a triggering pulse to the "one-shot" element F11/2. The resultant negative pulse from F11/2 is inverted by F7/3 and applied to F12/1 thereby terminating the initial positive output at F12/1. The cycle is repetitive, a further positive pulse being produced at the output of F12/1 at the end of each pulse from the "one-shot". The pulse output from F12/1 is fed to elements G2/1, G2/2, and G2/3 which form the initial stage of the clock pulse generator network.

Elements F7/1 and F7/6 provide an inhibiting input to F12/1 preventing pulsing under any of the following conditions:-

(1) Central machine switched to ONE SHOT. (+O.S. active)

(2) Central machine switched to DOUBLE SHOT (+D.S. active)

(3) Central machine not on RUN (-RUN inactive)

(4) Central machine not on ONE INSTRUCTION (+1 INSTR inactive)

(5) The BUSY indicator is not set (-FBUSY inactive)

F12/1 will be similarly inhibited if the central machine clock is switched to VARIABLE mode, i.e. +VAR is active.

Whenever the F.P.U. clock is inhibited, the output of F12/1 is held negative and the clock pulse generator network is driven by the central machine clock pulse, +MC, via elements F7/5 and F12/3. During F.P.U. clock operation, element F7/2 holds the output of F7/5 positive thereby inhibiting the entry of central machine clock pulses.

2.4 F.P.U. indicators

During normal program operation the action of the F.P.U. clock is controlled by the BUSY and END indicators as follows. On entry to a floating point operation, END will already be set and -FBSY1 will be inactive at elements F12/1 and F7/2 thereby holding the F.P.U. clock inoperative and allowing central machine clock pulses to drive the pulse generator network. These conditions are not altered by the initial load order (Fn. 136), but by the subsequent order (in the range Fn 132 - Fn 135) which specifies the arithmetic operation to be carried out.

During the implementation of the 136 order, +FCF6 remains active at F13/1 thus inhibiting +FSB which is the priming signal for the BUSY indicator. +FCF6 becomes inactive on the completion of the 136 order; the subsequent setting of BUSY thus depends on the activation of -FLOVR which occurs when a second operand is loaded by a function in the range Fn 132 - Fn 135.

In order to obtain a clean noise-free changeover from central machine clock to F.P.U. clock, the switching operation is delayed 0.5μs on the setting of BUSY. This is effected by applying the output of the BUSY indicator via the inverting element F13/5 to produce the delayed negative signal -FBUSY1 from F11/1. This signal is then applied to F12/1 and F7/2 to effect the switch.

The BUSY set signal also initiates, via F12/5, the microprogram entry signal -FST which, together with function bit decoding, selects the appropriate arithmetic microprogram sequence (drawing 07/51 - 07/54). Having performed this task, -FST is immediately reset by the WAIT indicator which is already primed by BUSY and is switched by the first F.P.U. clock pulse.

On completion of a floating point operation, -FEO becomes active, setting the END indicator via element F9/2. -FEND then becomes active thus inhibiting the F.P.U. clock via element F9/6. The END indicator also primes the resetting of BUSY which occurs on the first central machine clock pulse. The WAIT indicator is thus primed by BUSY and is switched to its initial state on the next central machine clock pulse. The F.P.U. then awaits further activation by program.

2.5 Derived signal circuits

Signals FX0, FX1 and FX2 occur at all clock pulses (i.e. both F.P.U. and central machine clock), and provide the input drive to the pulse gate network which produces a variety of timing pulses as designated below.

FXA1 - FXA3

These signals occur on all clock pulses and are applied as timing signals to the various special-function staticisers of the F.P.U. most of which are shown in drawings 07/52 - 07/56.

2.6 Highway to register strobe signals

FQ highway to FA register gating is effected by applying the timing signal FX0 to pulse gates B12/1 - B8/1 in conjunction with any active signal input to either of the NOR elements F4/3 or F4/2. The resultant strobe signals FXQAU1 - FXQAL2 are distributed to the FA register stages as follows:-

FXQAU1	-	bits 14 - 22
FXQAU2	-	bits 24 - 33
FXQAU3	-	bits 23, 34-37
FXQAL1	-	bits 0, 1, 8 - 13
FXQAL2	-	bits 2 - 7.

FQ highway to FB register gating is similarly effected by signals FXQBU1 - FXQBL2 generated at pulse gates B12/1 - B8/1 when FX0 is applied in conjunction with an active signal input from either of the NOR elements H09/6 or F4/6. The strobe signals are distributed to the FB register stages as follows:-

FXQBU1	-	bits 14 - 22
FXQBU2	-	bits 24 - 33
FXQBU3	-	bits 23, 34 - 38
FXQBL1	-	bits 0, 1, 8 - 13
FXQBL2	-	bits 2 - 7.

The FQ highway may be strobed directly or with a single left shift or a double right shift to the FC register. Signals FXQC1 - FXQC4 effect direct FQ - FC strobing as follows:-

FXQC1	-	bits 0 - 2, 8, 9, 12, 13, 19 - 21
FXQC2	-	bits 3 - 7, 14 - 18
FXQC3	-	bits 10, 11, 22 - 24, 33 - 37
FXQC4	-	bits 25 - 32.

Signals FXCR21 - FXCR24 strobe FQ to FC with a double right shift as follows:-

FXCR21	-	bits 3 - 7, 14 - 18
FXCR22	-	bits 1, 2, 8, 9, 12, 13, 19 - 21
FXCR23	-	bits 0, 10, 11, 22, 23, 34 - 37
FXCR24	-	bits 24, 33.

Signals FXCL 11 - FXCL 14 strobe FQ to FC with a single left shift as follows:-

FXCL11	-	bits 3 - 7, 14 - 18
FXCL12	-	bits 1, 2, 8, 9, 12, 13, 19 - 21
FXCL13	-	bits 0, 10, 11, 22, 23, 34 - 37
FXCL14	-	bits 24 - 33

Signals FXPD and FXPE perform the strobing operations FP to FD and FP to FE respectively. Signal FXFX is the F.P.U. buffer register strobe.

2.7 F.P.U. buffer register

When the F.P.U. accepts a Group 13 order, its F and X registers are set in accordance with those of the central machine (drawing $07/50_2$). The 3-bit F register is set by signals -RF0, - RF1 and -RF2, and thus holds the number of the function being performed. The X bits X0 and X1 are similarly set into the register by signals -FRX and -RX1. The value of the X2 bit is not of significance to the F.P.U. since it specifies the method of loading operands which is carried out by the central machine microprogram. The setting of the buffer register is strobed by FXFX which is derived as follows:-

+MRN9 $\longrightarrow$ -FLXF (drawing L_0)

-FLXF and FX2 $\longrightarrow$ F5/1 $\longrightarrow$ FXFX. (drawing $07/50_1$). The clearing of the register after the completion of an order is effected by FCFX as follows. When a 136 order is encountered (value 110 stored in F register), +FCF6 becomes active at element E2/6 thereby inhibiting the setting of BUSY at F13/1 during the loading of the first operand. Element E8/2 is also primed by the F register, and on the occurrence of -FQBL, (at the completion of the load operation), it produces the active output +FCFX. This signal is applied to the reset inputs of the F and X registers and also to the strobe generator element F5/1 (drawing $07/50_1$) to produce the timing signal FXFX.

When a function in the range Fn 132 - Fn 135 is stored in the register, the appropriate output signal will be decoded (-FCF2 to -FCF5) and used to select the required arithmetic microprogram. At the completion of any function in this range, -FE0 becomes active thus producing FCFX via F12/2.

The X register similarly produces output signals which control rounding and normalising operations in the arithmetic microprogram sequences and also control the conditions of load/unload operations as previously stated.

CHAPTER 3

LOAD AND UNLOAD OPERATIONS

3.1 Load operations

3.1.1 Fn 136, X = 0

This order will load the double length operand in N and N + 1 into all F.P.U. registers, i.e. the argument into FA, FB, FC, and the exponent into FD, FE, Since the F.P.U. hardware is designed to perform all arithmetical operations by adding or subtracting FA, FE to or from FB, FD, this arrangement allows for the case in which the next instruction specifies an inverted function. The load operation is carried out by central machine microprogram sequences which are shown in detailed logic form in drawing L, and in flow diagram form in drawing $07/60_{01}$. The logic circuit action is as follows.

When an order in the range Fn 132 - Fn 137 is loaded into the central machine F register, -CG28 is activated at element 2E12/5. On the subsequent activation of MEP1, a test is carried out to determine whether the F.P.U. is busy. Thus, if the unit is already in an active state, all inputs to 2E12/6 will be negative and +MNIL6 will be activated. The sequence then enters a waiting loop, the NIL bistable being set on the first clock pulse of the F.P.U. preparation. The occurrence of a fast hesitation request during this loop time will activate +MHES0, at 2E13/1, thus causing an entry to the hesitation sequence. On completion of the hesitation the waiting loop will be re-entered when +MEPI becomes active at 1C37/6.

The test loop will be maintained until, when the F.P.U. is no longer busy, +MRN9 becomes active at 2E7/6 thus producing -FLXF at 2E7/3. The next clock pulse performs the following actions:-

(1) Loads the F.P.U. F and X registers.

(2) Reads the contents of N (m.s. part of operand) to bits 37 - 14 of FPU registers FA, FB and FC. It should be noted that the l.s. bits of these registers and also registers FD and FE are simultaneously cleared.

(3) Gates the contents of the FB register to FK highway.

(4) Reads n + 1 (l.s. part of operand) into central processor register B.

(5) Gates the central processor register P (order number) on to K highway and CYB0 to the mill thereby preparing to address the next program instruction.

During the second digit time, +FLOVR becomes active at 2E13/2 again producing -FQAU and -FQBU, and also -FCSL and -FQBL. -FCSL gates the computer B register (l. s. part of operand) via the SOUT highway on to FQ highways 14 - 0 and to the FP highway. The next (third) clock pulse performs the following actions:-

(1) Strobes FQ (l. s. part of argument) to registers FA, FB, FC, and FP (exponent) to registers FD, FE.

(2) Strobes the status of SOUT 23 to FOVR bistable.

(3) Since the function register holds the binary value 110, (Fn 136), element E8/2 is already primed to produce the active output +FCFX on the activation of -FQBL. +FCFX clears the F and X registers in the previously described manner, in readiness for the loading of the next order.

(4) Gates FB to FK. (See drawing $07/55_1$). +FQBU is applied to G6/1 thus priming the B →K bistable via G6/3. The gating is effected by -FCBK0 and -FCBK1 when the bistable is switched.

(5) +MEO is activated at 2E7/2 causing re-entry to the central computer sequence.

It should be noted that the 136 order cannot set the BUSY indicator since, once the order is stored in the FF register, +FCF6 remains active and inhibits the generation of +FSB at element F13/1 throughout the implementation of the order. When the FF register is subsequently cleared, +FCF6 goes negative but F13/1 remains inhibited until -FLOVR is activated by a further order in the range Fn 132 - Fn 135, i. e. until the l. s. part of the second operand is loaded.

3. 1. 2. Fn 136 X = 1

This order stores a zero operand (argument = 0, exponent = -256) in the F. P. U. registers. The logic sequence is similar to that described for Fn 136 X = 0 except for the following variations.

(1) Drawing L_1. Signals - CF6 and -FRX are active at elements 2D7/2 and 2D7/1, both elements thereby producing positive outputs which inhibit -FCSU and -FCSL. Zeros are therefore strobed from FQ to the FPU argument and exponent registers.

(2) +FD1 becomes active at 2E13/6 (drawing L) and is applied to elements G33/3 (drawing $07/48_0$) and G28/5 (drawing $07/49_0$), thus setting a 1 into bits 8 and 9 of the exponent registers.

(3) The overflow bistable FOVR, (drawing $07/56_1$) is cleared by -SOUT 23 and -FCSL at element F31/2.

3.2 Unload operations

3.2.1 Fn 137_5X = 0

Store the contents of the F.P.A. in N and N + 1.

The unload sequence is entered at MEP1 as in the case of Fn 136, and an initial circuit of the +MNIL6 loop is made in order to allow clearing of the central computer mill. The subsequent action depends on the busy status of the F.P.U. So long as the unit is busy the loop is maintained; when the unit becomes free, the sequence continues. The logic circuit action (drawing L_0) is as follows.

-CF 137 becomes active at the common output of 1C36/3 and 1C36/4 when Fn 137 is encountered. -CF137 and -MEP1 produce +MNIL6 at 2E12/6 thereby causing entry to the waiting loop. In the next digit time the busy status of the FPU is tested (element 1C36/1). With all conditions satisfied, +MWN3 becomes active, simultaneously generating -FCB at 2D7/6. -FCB gates the m.s. part of the operand (argument bits 37 - 14) from the FC register to the BIN highway whence it is gated to the computer Q highway.

The second clock pulse strobes the m.s. part of the operand to the computer B register and writes it to store address N. Simultaneously the contents of register N are gated to the K highway, and 1 is added to the mill, resulting in the sum N + 1 appearing on the Q highway. At this stage it is necessary to delay the gating of the l.s. part of the operand to the BIN highway in order to prevent a mix with the N + 1 data which is already on the computer Q highway. It is also essential to allow a delay of one digit time between the strobing of N + 1 from the Q highway to the N register and the subsequent WRITE operation, in order to allow the machine to deal with a datum/limit failure. The required delay is obtained by generating +MNK4 (element 1C36/6) during the second digit time.

On the third clock pulse the sum of N + 1 is strobed into N, and X is gated to J. The latter gating serves the purpose of generating -CXDJ20 which enables the generation of +MWN4 at element 1C37/2. +MWN4 activates -FCDB (2D7/4) which effects the gating of the l.s. part of the operand from registers FC (bits 13 - 0) and FD to the BIN highway, and also puts the status of FOVR to BIN 23 (drawing $07/56_1$).

The fourth clock pulse will strobe the l. s. part of the operand into the computer B register and write it into store address N + 1. The contents of register P will be gated to the K highway with the addition of 1 to the mill, resulting in the address of the next instruction appearing on the Q highway. +MEO becomes active, generating +FLOW (2E13/3) which gates the status of B23 to Q23 thus setting V if exponent overflow has resulted from a floating point operation.

3.2.2. Fn 137, X = 1

Store the contents of the F. P. A. in N and N + 1 and clear the F. P. A.

The sequence is identical to the X = 0 case except that +FCA (1C37/1) and +FD1 (2E13/6) become active at the same time as +MEO. As shown in drawing 07/50, +FCA primes the strobing of FQ (zero at this time) to registers FA, FB, FC and similarly the strobing of FP to registers FD, FE. +FD1 sets a zero exponent (-256) into the FD and FE registers.

CHAPTER 4

ARITHMETIC OPERATIONS (Fn 132 - Fn 135)

4.1 General

Any function in this range causes a second operand to be loaded to the F.P.U. by a modified form of the 136 sequence in which the operand is loaded to only one pair of registers, i.e., to registers FA, FE, for a "straight" function, to FB, FD, for an inverted function. On the completion of the load operation the BUSY indicator is set and the appropriate arithmetic microprogram is entered due to the activation of -FST and the decoding of the F.P.U. function register.

4.1.1 Fn 132 Addition

Add the operand in N and N + 1 to the F.P.A. x has the following significance:-

$\overline{X0}$	result	to	be	rounded
X0	result	to	be	un-rounded
$\overline{X1}$	result	to	be	normalised
X1	result	to	be	un-normalised

4.1.2 Summary of operation

Floating point addition is effected by adding the arguments FB and FA in terms of the larger of the exponents FD and FE. The general method of operation may be seen from the flow diagram (drawing 07/60, sheets 2 and 3).

On entry to the microprogram (FST and Fn 2 active) the subtraction FD - FE is performed in order to evaluate the exponent difference. The signal +FDN0 thus initiates the gatings FD $\longrightarrow$ FN, $\overline{FE}$ $\longrightarrow$ $\overline{FO}$, and FCYBE0, the exponent mill consequently performing the operation FD + ($\overline{FE}$ + 1). The subsequent logic sequence, which may take any one of several alternative paths, is determined by analysis of the sign and magnitude of the exponent difference. In order to allow sufficient time for decoding of the mill output, the subtraction operation is therefore repeated by the activation of +FDN8, the MODE bistable also being set to indicate the completion of the second subtraction.

Analysis of the exponent difference will indicate one of the following conditions:

(1) FD exceeds FE by more than +255 resulting in exponent overflow. The positive result indicates that FE is negative, and since overflow has occurred, the operand in FA, FE is insignificant in comparison to that in FB, FD. This case satisfies the conditions OVR and $\overline{D8}$ thus activating +FBK0 which initiates the register gatings as shown. FB and FD are gated to their respective mills as the final result, the gatings FA $\longrightarrow$ FL, $\overline{FA}$ $\longrightarrow$ FL and FCYBA0 effectively adding zero to the mill sum.

(2) FD exceeds FE by an amount less than +255 but greater than +37. This case satisfies the conditions P $\neq$ 0 (mill sum non - zero) and D >> E, again resulting in the +FBK0 sequence. The operand in FA, FE is again regarded as insignificant since the number of adjusting shifts required would result in a zero argument.

(3) FD exceeds FE by an amount not greater than 37. This case satisfies the conditions P $\neq$ 0, $\overline{OVR}$, D > E, and $\overline{D >> E}$, thus resulting in entry to the +FEO1 sequence in which the argument FA is aligned. On each traverse through the sequence, FA is gated to the mill, right shifted on to FQ and gated back into FA. Simultaneously FE (the exponent difference) is gated to the mill together with FD and $\overline{FD}$ (effectively - 1) and gated back via FP to FE. This loop is repeated until the test FE = 1 is satisfied, indicating that the number of right shifts performed on the argument in FA equals the initial exponent difference.

+FAL0 then becomes active gating FA and FB to the mill to form the final argument whilst +FDN5 gates FD to the mill to give the final exponent.

(4) FD = FE. In the case of equal exponents, conditions $\overline{OVR}$ and P $\neq$ 0 are satisfied, and the argument addition is performed directly by +FAL0, the final exponent being gated from FD to the mill by +FDN5.

The remaining three cases correspond to the conditions (1), (2) and (3) with FD and FE interchanged. In each case the exponent difference is negative, indicating that the greater operand is held in FA/FE and the FB/FD operand is therefore to be adjusted.

(5) FD - FE is less than -256. This case satisfies the conditions OVR and $\overline{E8}$ thus giving entry to the +FAL1 sequence in which FA and FE are gated to their respective mills as the final result, the FB/FD operand being disregarded.

(6) FD - FE is greater than -256 but less than -37. Conditions P $\neq$ 0, $\overline{OVR}$ and E >> D are satisfied and the +FAL1 sequence is entered as in case (5).

(7) FD - FE is not less than -37. Conditions $P \neq 0$, $\overline{OVR}$, E > D and $\overline{E >> D}$ are satisfied and the +FDN2 sequence is entered. This is an argument adjusting loop in which FB is right shifted until the exponent difference (in this case in FD) has been counted to zero. Since the exponent difference is negative, the count is effected by gating FD to the mill with FCYBE0 (+ 1).

4.1.3 Rounding

In order to compensate for the loss of a "1" bit on the final argument-adjusting shift, rounding will normally be specified $\overline{(X0)}$. This is carried out as follows: Signal +FCYBA0 is generated and applied to bit 0 of the argument mill prior to the final adjusting shift. If the l.s. bit of the argument is also a "1", a carry will thus be propagated into mill bit 1, ensuring that a "1" appears in bit 0 of the FQ highway after shifting. If the l.s. bit of the argument is zero, there is no carry into mill bit 1, the final zero being shifted out.

4.2 Detailed logic sequences

4.2.1 Loading sequence (drawing L)

This is similar to Fn 136 with the following exceptions:

(1) +MRN 10 activates -FCSU and -FQAU thus gating the m.s. part of the operand via the FQ highway to the FA register. -FQBU is inhibited by a positive output signal from 2D12/1 since its two inactive input signals indicate that the function is not 134 or 136 and is not inverted.

(2) +FLOVR activates -FCSL. and -FQAU, thus loading the l.s. part of the operand to FA. -FLOVR (drawing $07/50_2$) and +FCF6 (now inactive) generate +FSB thus priming the setting of the BUSY indicator (drawing $07/50_0$). -FST is activated by the setting of BUSY on the next clock pulse.

4.2.2 Addition sequence (Drawing 07/51)

With +FCF2 and -FST active, +FDN0 is activated at E15/5. FDN0 is applied as a priming signal to the control bistables D $\longrightarrow$ N, E $\longrightarrow$ O and FCYBE0 (drawing 07/56). At the next clock pulse the bistables are set and the following actions occur:-

-FCDN	gates	FD to FN
-FCIE	gates	$\overline{FE}$ to FO
-FCYBE0	adds	1 to the mill.

The exponent difference thus appears on the FP highway.

+FDN8 is now activated at E15/1 by -FCIE, -FCYBE0, and the inactive +FCMOD (since the MODE bistable is unset at this time). +FDN8 repeats the actions of +FDN0 and also primes the MODE bistable (drawing $07/52_1$). By this time the decoder chain which determines the signal and magnitude of the exponent difference will have settled, and the signal -FCMOD is applied to certain elements to time the next step, i.e. the selection of the appropriate microprogram path.

4.2.3 Exponent difference decoder

The action of the decoder circuit can be readily understood by consideration of the various cases of exponent difference which may arise. These are as stated below:

(1) Exponent overflow (OVR and FD >> FE or FE >> FD)

This condition indicates an exponent difference greater than +255 or less than -256 and implies that the lesser of the two operands is non-significant The condition is detected by elements E21/5 and E20/4 which register the values of the exponent mill sign and extended sign bits 8 and 9. On the occurrence of either positive overflow (bit 9 = 0, bit 8 = 1) or negative underflow (bit 9 = 1, bit 8 = 0), -FTOVR becomes active.

Where overflow or underflow occurs as a result of exponent subtraction, the two exponents must be opposite in sign as shown by the following cases:-

FD =	+255	FD =	-256
FE =	- 1	FE =	+ 1
	+256		-257

Since FE is always subtracted from FD, the sign of the result indicates which of the exponents was the smaller (negative) and therefore non-significant one.

The sign bits of FD and FE are registered by elements E20/2 and E21/2 respectively. Thus, if on the activation of -FTOVR, +FRD98 is inactive (FD positive), +FBK0 is activated. If +FRE98 is inactive (FE positive), +FAL1 is activated.

+FBK0 performs the following actions:-

(a) Primes the setting of the B - K bistable (07/55) and the D - N bistable (07/56) thereby enabling the FB/FD operand to be gated to the mills as the final result on the next clock pulse.

(b) Initiates the entry to the normalising sequence. This entails the generation of +FNORM at element F29/4, thus requiring one of the signals +FCAL or +FCIA to be active at E18/5. These two signals respectively perform the gatings FA $\longrightarrow$ FL and $\overline{FA}$ $\longrightarrow$ FL (A - L

and $\overline{A}$ - L bistables, drawing 07/55). In order to generate +FNORM without affecting the mill sum (which currently represents the final result) it is thus essential that +FBK0 enables both the A $\longrightarrow$ L and $\overline{A}$ $\longrightarrow$ L bistables, and also the CYBA0 bistable, the resultant addition to the mill being zero.

+FAL1 primes the setting of the A $\longrightarrow$ L and E $\longrightarrow$ O bistables thus enabling the FA/FE operand to be set to the mills as the final result and also initiating entry to the normalising sequence by the activation of +FNORM.

(2) OVR and FD >> FE or FE >> FD

In these two cases the exponent difference does not overflow but is either greater than +37 or less than -37. Since in either case the small operand will be insignificant after argument adjustment, either the +FBK0 or the +FAL1 sequence will be entered. The two conditions are detected by separate decoder chains which sense the mill output and the FP highway. The chain of elements H32/2, E21/6, E15/2 and E16/5 ($07/51_0$) is thus capable of sensing a positive mill sum and indicating whether it is greater or not greater than +37. The decoding action is as follows:

Any of the following mill output configurations will indicate a mill sum greater than +37:-

Bit 7 active

Bit 6 active

Bit 5, 2 and 1 active

Bit 5 and 3 active

Bit 5 and 4 active

Any of these configurations will give a negative voltage at the common output of elements E21/6 and E15/2, and since the sign bit +FRMS48 will be inactive for a positive exponent difference, E16/5 will give a positive output. This is inverted by E15/4 thus activating +FBK0 at E17/1.

Negative exponent differences are similarly decoded by the chain of elements H30/6, E16/1, H33/2, H33/4 and H32/3. A mill output bit configuration which results in a mill sum less than -37 will again result in a negative output from E25/4 but in this case the active sign bit +FRMS48 will inhibit +FBK0 at E17/1 but will activate +FAL1 at E18/2.

(3) FD > FE or FE > FD

These two cases (exponent difference not greater than +37 or less than -37) provide a negative decoder output voltage at the common output of E16/5 and

H32/3. This is applied via E15/4 and E16/3 to enable either +FEO1 at E18/1 or +FDN2 at E17/6 dependent on which of the cases has occurred as indicated by the status of the sign bit + FRMS 48.

In the FD > FE case, +FEO1 initiates the loop sequence in which the FA argument is adjusted and also the count down of the exponent difference in FE. +FEO1 is thus applied as a priming signal as follows:-

To the SHR1 bistable in order to gate the argument mill output to the FQ highway with a single right shift.

To the D $\longrightarrow$ N and $\overline{D}$ $\longrightarrow$ N bistables thus effectively adding -1 to FE.

To the FXPE pulse generator circuit ($07/50_1$) in order to gate the decremented count back into FE.

During the digit time in which the A $\longrightarrow$ L and SHR1 bistable remain set, +FQA0 becomes active at E20/6 ($07/51_1$) and is applied to F4/2 ($07/50_0$) to initiate the gating of the right shifted argument from FQ back into FA. Simultaneously the value of FE is tested at element E19/1 ($07/51_1$). The input signal +FTE1 remains inactive so long as FE is greater than 1 (see element $H32/1_1$ drawing $07/51_2$). +FEO1 thus remains active, ensuring that the argument adjusting loop is repeated the specified number of times. Since shift operations and exponent decrementing operations are performed simultaneously, the count FE = 1 indicates that the final shift is in progress.

At this stage +FTE1 becomes active, inhibiting +FEO1, and -FTE1 becomes active at E21/1 ($07/51_1$) thereby activating +FDN5 and also FAL0 at E22/5. +FAL0 performs the gatings A $\longrightarrow$ L, and B $\longrightarrow$ K; +FDN5 gates the final exponent FD to the exponent mill, the final result thus appearing on the FQ and FP highways.

In the FE > FD case, the +FDN2 sequence performs an adjusting shift on the FB argument, the count being held in the FD register. Since in this case the exponent difference is negative, the count is progressed, by adding +1 to FD on each traverse through the loop. This is effected by adding +FCYBE0 to the exponent mill, the CYBE0 bistables ($07/56_1$) being primed by +FDN2. After each shift the argument is gated from FQ to FB (+FQB0 active at E19/6), the value of FD being tested at E20/1. On exit from the shift loop, +FAL0 and +FE05 become active, setting the argument sum FA +FB and exponent FE to their respective mills.

In both the above cases, rounding of the shifted argument will be performed when necessary by the addition of +FCYBA0 to bit 0 of the argument mill. +FCYBA0 becomes active at H12/6 whenever a final argument shift occurs, i.e., whenever either +FTE1 or +FTD1 is active at E19/2.

(4) FD = FE

The case of equal exponents (P = 0) is sensed by elements H32/4 and H32/5 ($07/52_1$) the active signal -FTP0 being applied at E22/6 ($07/51_1$) to activate +FDN5 and consequently +FAL0 at E22/5. The operation thus involves only direct addition of the arguments and gating of the final result to the mills.

4.3 Subtraction sequence

The subtraction sequence (Fn 133) differs from the addition sequence only in the following respects:

(1) An inverted function may be specified (X2 = 1) in which case the second operand will be loaded into registers FB/FD. It will be seen from drawing L_1 that the active signal +RX2 allows -FQBU to be generated but -RX2 and the inactive +FCF46 at element 1C37/5 inhibit -FQAU.

(2) The argument arithmetic is performed by the FIA0 sequence which puts B $\longrightarrow$ K, $\overline{A}$ $\longrightarrow$ L and FCYBA0 to the mill (drawing $07/51_1$).

(3) In the OVR FE >> FD case, +FAI1 is activated thus putting a negated operand to the mills.

4.4 Multiplication (Fn 134)

The F.P.U. performs the multiplication of two normalised operands by addition of the exponents and multiplication of the arguments. The product of the arguments is obtained by addition of a series of partial products as is the case in central arithmetic unit multiplication operations. The general method of F.P.U. operation is as follows.

The multiplicand is initially loaded into the FA and FE registers, the multiplier being loaded in FC and FD. The sequence of events following the completion of the load operation may be seen from the flow diagram, drawing 07/60 sheets 4 and 5.

On the first clock pulse after entry to the multiplication microprogram, the following actions occur:-

(1) +FEO0 performs the exponent addition FD + FE, effectively placing the final exponent on the FP highway.

(2) The value of the l.s. multiplier bits C1, C0, is decoded in order to determine which magnitude of the multiplicand must be added to the mill to

form the first partial product. The decoding of two multiplier bits, rather than one, halves the number of operations required to compute the result. Dependent on the status of C1, C0, it may be thus necessary to gate the first, second or third magnitude of the multiplicand (FA) to the mill. In the case where C1, C0 = 01 or 10, the respective gatings A $\longrightarrow$ L (+FAL2) or 2A $\longrightarrow$ L (+FTAL) can be made directly. In the case where C1, C0 = 11 the third magnitude of FA must be gated to the mill and this can most conveniently be performed by an effective gating of (4 - 1) A $\longrightarrow$ L. This involves the subtraction of A in the current operation and the setting of a carry which, added to the C1, C0 decoding in the next operation, will add an effective factor of 4A. The carry is registered by the generation of +FSM and the subsequent setting of the MODE bistable.

The following table co-relates decoder results and subsequent mill actions.

C1	C0	CARRY	VALUE	MILL ACTION
0	0	0	0	None.
0	0	1	1	Add multiplicand
0	1	0	1	(+FAL2 generated).
0	1	1	2	Add twice multiplicand.
1	0	0	2	(+FTAL generated)
1	0	1	3	Subtract multiplicand and
1	1	0	3	Set carry (+FIA2)
1	1	1	4	None but set carry.

(3) The +FBK1 sequence performs the following functions:

(a) Strobes the FQ highway to the FB register and thence to the argument mill and then strobes the mill sum with a double right shift back to the FQ highway. On the first operation the content of FQ is zero and the FB register is therefore cleared. The mill sum FA + FB (first partial product) will thus depend solely on the FA gating, i.e. FA, 2FA or -FA. The double right shift which occurs whenever the mill sum is re-circulated to the FQ highway ensures that, on the next operation, the new value of the multiplicand will be added in the correct significance, i.e. with an effective double left shift w.r.t. the previous partial product.

(b) The FC register is right shifted two places, the original multiplier bits C1 and C0 being replaced with a new pair. The l.s. mill bits M1 and M0 are simultaneously stored in bits 37 and 36 of the FC register. On the first operation the mill sum is zero at the time of this gating resulting in an initial pair of zeros being strobed to C37 and C36. The mill sign bits M39 and M38 are also strobed to Q37 and Q36, each operation thus producing an extended sign bit in the shifted partial product.

In order to complete the multiplication, the above cycle is repeated with some variations a total of 19 times, the resultant final argument being a double length product held in registers FB and FC.

The second operation is varied from the first in the following respects. The +FEO sequence, having performed its allotted task of placing the final exponent on the FP highway is not required in further operations. The exponent mill can now be used as an operation counter in the argument multiplication sequence. +FIEO thus becomes active, setting a count of 16 into the exponent mill and also storing the final exponent in FD. The count is formed by the usual method of forming an all "ones" pattern by means of the E $\longrightarrow$ O and $\overline{E} \longrightarrow$ O gatings and suppressing unwanted bits by means of the SP bistable. The count of 16 may be explained by the fact that servicing of the count commences only on the third operation, the zero count thus indicating the final operation.

The FBK1 sequence and the C1, C0 decoding operation are repeated as in the first operation.

On the third operation +FDN1 becomes active with +FBK1, strobing the constant 16 from FP to FE and thence to the exponent mill and simultaneously gating D $\longrightarrow$ N and $\overline{D} \longrightarrow$ N (effectively -1) to the mill. The count is thus reduced by 1 on each subsequent traverse through the +FBK1 loop.

The FDN1 and +FBK1 sequences occur in all subsequent operations except the last one, in which the decoding of the final pair of multiplier bits must be treated differently due to the presence of the multiplier sign bit in C1. On this operation the values of the C1, C0 decoding and subsequent mill action are altered as shown in the following table.

C1	C0	CARRY	VALUE	MILL ACTION
0	0	0	0	None
0	0	1	1	Add multiplicand
0	1	0	1	(+FAL2 generated)
0	1	1	+2	Add twice multiplicand
1	0	0	-2	None but set and carry
1	0	1	-2 + 1 = -1	Subtract multiplicand
1	1	0	-2 + 1 = -1	
1	1	1	-2 + 2 = 0	None

Entry is made to this final operation (+FBK2 sequence) when the operational count reaches the value of 1. It will be noted that, in order to preserve the correct significance of the final partial product, the mill sum is strobed to FQ with a single right shift. This ensures that only one sign bit is propagated and a correctly scaled final partial product is thus subsequently strobed to FB.

The final sequence +FBK3 puts the m.s. half of the final argument to FQ and the final exponent to FP, also performing the final gating of the l.s. mill bits to FC37 and FC36. Due to the final shifting process, the final value of M1 is duplicated in FB0 and FC37. The m.s. bit of FC is therefore disregarded.

In the case where the final decoding of C1, C0 is equal to -2, 2A must be subtracted from the final mill sum. This is most conveniently performed by setting MODE during the +FBK2 sequence and subtracting A (+FIA2) in the subsequent +FBK3 operation.

During 07/52 shows the derivation of the main control waveforms of the multiplication microprogram.

4.5 Division (Fn 135)

The F.P.U division method involves the following operations:-

(1) Subtraction of the divisor exponent from the dividend exponent, the resultant being the quotient exponent.

(2) Division of the dividend argument by the divisor argument.

(3) Correction of the quotient argument.

The argument division is performed by the non-restoring method which is described in detail in Chapter 6 of Part 3. In general, the F.P.U division method follows central arithmetic unit practice apart from some differences in the scaling of operands. Since the arguments of floating point operands are in fractional form it is essential that the quotient resulting from any F.P.U. division operation shall also be in this form. Whilst normalised operands satisfy these conditions, care must be taken in the use of un-normalised operands which in certain cases could produce an integral quotient exceeding the storage capacity of the F.P.U argument registers.

Example $1/4 \div 1/16$, i.e. $0.0100 \div 0.0001 = 4$

Un-normalised operands are therefore legal only with a positive divisor which is greater than a positive dividend, or in the case of a negative dividend, the divisor must be of equal or greater numeric value and may be of either sign.

When normalised operands are used, a single bit provisional overflow (FPOVR) will arise when the dividend argument exceeds the divisor argument, i.e., when the first trial addition or subtraction fails to change the sign of the residue.

An attempt to divide by zero will set overflow (FOVR) and the operation will be abandoned. Provisional overflow is corrected during normalising and thus does not stop the division operation.

4.5.1 General method of operation

The division sequence may be followed by means of the flow diagram, drawing D7/60 sheets 6 and 7. Detailed logic sequences are shown in the microprogram drawing 07/54 sheets 0 and 1.

The general method of operation is as follows: The initially loaded operand is set into all F.P.U. register. As in the case of subtraction, either a "straight" or an inverted function may be specified by X2. In either case the dividend is stored in FB/FD, and the divisor in FA/FE, the argument of the quotient being progressively formed in FC.

On entry to the division microprogram, +FAL2 becomes active, gating FA (the divisor) to the mill for a zero/non-zero test. A zero divisor results in the setting of FOVR (element F27/3, drawing $07/56_1$) and also the activation of +FEO (element F15/1, drawing $07/54_0$). +FEO causes entry to the End of Order sequence thus clearing the F.P.U. registers and abandoning the operation

In the event of a non-zero divisor +FBK7 is activated (element F14/1, drawing $07/54_0$), thus implementing the following actions:-

(1) the exponent subtraction FD - FE, and

(2) the gating of the dividend, FB, to the argument mill.

A comparison of the divisor and dividend signs is then made in order to determine whether the first trial operation shall be addition or subtraction of the arguments (elements F16/6, F17/1, F17/2, drawing 07/54). Unlike signs result in addition of the arguments, +FAL2 being activated to gate FA to the mill to form the residue FB + FA. Like signs result in the activation of +FIA2 and thus the subtraction FB - FA. Dependent of which of the above-stated actions is taken, and also on the result, the first quotient bit is formed and stored in FC0.

The rules which determine the value of the quotient bit are as follows:

(1) If the operands were of like sign (i.e. the action was subtract) the quotient bit is 1 when the residue has the same sign as the initial dividend.

(2) If the operands were of unlike sign (i.e. the action was add), the quotient bit is 0 when the residue has the same sign as the initial dividend.

The method of generating the quotient bit is indicated in the division microprogram drawing $07/54_0$. The appropriate value is set into the FRQU0 bistable by decoding of the sign conditions of the dividend, FRB38, and the residue, FRMS 37, which also takes into account the initial sign comparison of dividend and divisor. Hence, considering element F18/4, -FCAL1 will be active when the initial operands are of like sign and will therefore enable the decoding of the inputs to elements F18/5 and F18/3. Any configuration of like signs (00 or 11) provides a second negative input to F18/4 thus resulting in the priming of FRQU0. The subsequent setting of FRQU0 will set a 1 into FC0 via element A33/3 (drawing $07/0_0$).

If the above operation has resulted in a residue which differed in sign to the dividend, either F18/3 or F18/5 would have given a positive output thereby inhibiting the setting of FRQU0 and consequently storing a 0 in FC0.

Since operands of like sign were considered in the above examples, a quotient of positive sign would be expected in all cases. Those cases in which a 1 appears as the first quotient bit thus represent overflow. Where normalised operands are in use this can result in only one-bit overflow (FPOVR set) which is corrected by a single right shift and adjustment of the exponent in the normalising sequence. During the first trial addition/subtraction, the microprogram also carries out a test to determine when the divisor is un-normalised and smaller than the dividend (A37 = A36, M37 = B38). This condition results in the setting of FOVR in addition to FPOVR and, although the operation is allowed to continue to its conclusion, OVR is set thus indicating an incorrect result.

When operands are of unlike sign, the first trial operation is addition and -FCAL0 becomes active, enabling the decoding of the inputs to elements F17/3 and F17/4. In this case, since a negated quotient will result, the rule which determines the value of the quotient bit is inverted, i.e. if the residue is similar in sign to the dividend, the quotient bit is 0 and provisional overflow has occurred.

The following table summarises the method of quotient bit determination for normalised operands of varying sign and magnitude.

Operands of like sign Trial operation - subtraction	
Both operands positive	Both operands negative
Dividend > divisor Residue sign = dividend sign. Quotient = 1. FPOVR set.	Dividend < divisor Residue sign = dividend sign Quotient = 1. FPOVR set.
Divisor > dividend Residue sign ≠ dividend sign Quotient = 0	Divisor < dividend Residue sign ≠ dividend sign Quotient = 0
Operands of unlike sign Trial operation - addition	
+ve dividend -ve divisor	-ve dividend +ve divisor
/dividend/>/divisor/ Residue sign = dividend sign Quotient = 0. FPOVR set	/dividend/>/Divisor/ Residue sign = dividend sign Quotient = 0. FPOVR set
/divisor/>/Dividend/ Residue sign ≠ dividend sign Quotient = 1	/divisor/>/dividend/ Residue sign ≠ dividend sign Quotient = 1

Due to the activation of +FCYBE0 during the FBK7 sequence, +FBK8 and +FDN4 become active. +FBK8 completes the first trial operation by putting FB to the mill for addition to the appropriate FA gating and shifting the mill sum one place left on to the FQ highway. The shifted residue is then strobed from the FQ back into FB, thus on the next trial operation the effective value of the divisor will be reduced by a power of two.

The +FDN4 sequence sets an operational count of 36 into register FE, this providing control of the remaining trial operations. The count is set initially into the exponent mill by the method of gating FD and $\overline{FD}$ with the MODE and SP bistable set as explained in the multiplication sequence description. On the second and subsequent trial operations, the count is serviced by the +FDN1 sequence. During this sequence the exponent difference is held in FD. The setting of the D $\rightarrow$ N bistable activates +FCCL0, which provides a single left shift of register FC and also strobes the status of FRQU0 to FC0. It will be noted that the tests for FOVR and FPOVR require an active gating signal from the MODE bistable and therefore can occur only during the +FDN4 sequence, i.e. on the first trial operation. The +FBK8 loop is repeated until the count in FE reaches zero.

On the final addition/subtraction +FIE1 becomes active gating the mill sum directly to FQ and thence to FB. Since the final mill sum is the remainder, shifting is not required on this operation. The gatings $\overline{D} \rightarrow$ N and $\overline{E} \rightarrow$ O merely provide gating signals for the following quotient correcting sequence. +FCCL0 becomes active, placing the l.s. quotient bit into FC0.

4.5.2 Correction of quotient and remainder

The following examples illustrate the argument division method and indicate also the end correction procedure.

In the first example the correct quotient 0101 and remainder zero are produced by the fourth trial operation. However since the F.P,U, always carries out the fixed count of trial operations, a number of redundant operations occur in each of which the quotient bit is 0 and the divisor forms an incorrect remainder in FB. On storing the last quotient bit the microprogram thus checks for this condition. If FC0 is zero ($\overline{RQUO}$), the +FAL0 sequence is entered and the error condition is corrected by restoration of the divisor.

Example 1. 35/64 ÷ 56/64. (c.f. fixed point division 35 ÷ 7).

Dividend in FB	0 1 0 0 0 1 1	
Divisor in FA	0 1 1 1 0 0 0	
FB to mill (FM)	0 1 0 0 0 1 1	

FA and FM are of like sign so first trial operation is subtraction

FB	0 1 0 0 0 1 1	
FA	0 1 1 1 0 0 0	
FM	1 1 0 1 0 1 1	

Residue sign changed (FM ≠ FB) so first quotient bit ≠ 0
Residue sign ≠ divisor sign (FA ≠ FM) so next operation is add.

FB (left shifted)	1 0 1 0 1 1 0	
FA	0 1 1 1 0 0 0	
FM	0 0 0 1 1 1 0	Quotient bit = 1

Next operation is subtraction

FB	0 0 1 1 1 0 0	
FA	0 1 1 1 0 0 0	
FM	1 1 0 0 1 0 0	Quotient bit = 0

Next operation is addition

FB	1 0 0 1 0 0 0	
FA	0 1 1 1 0 0 0	
FM	0 0 0 0 0 0 0	Quotient bit = 1

Next operation is subtraction

FB	0 0 0 0 0 0 0	
FA	0 1 1 1 0 0 0	
FM	1 0 0 1 0 0 0	Quotient bit = 0

Next operation is addition

FB	0 0 1 0 0 0 0	
FA	0 1 1 1 0 0 0	
FM	1 0 0 1 0 0 0	Quotient bit = 0

Final operation is addition

FB	0 0 1 0 0 0 0	
FA	0 1 1 1 0 0 0	
FM	1 0 0 1 0 0 0	Quotient bit = 0

It will be noted that in the example chosen the error condition was introduced by the subtraction of the divisor from the zero remainder in the fifth operation. The negated divisor thus appears as the false remainder in FB after the last trial operation. The +FAL0 sequence forms the correct remainder by the addition of FA and FB, the resultant mill sum being returned to FB by +FQB1.

When FC0 is 1, (RQUO), the +FAL0 sequence is by-passed.

The microprogram then checks the sign of the divisor since, if this is negative, correction of the quotient will be required. As shown in the following example, a negative divisor forms a remainder equal to itself thus giving an incorrect quotient as answer.

Example 2. 35/64 ÷ - 56/64

	FB	0 1 0 0 0 1 1	
	FA	1 0 0 1 0 0 0	
ADD	FM	1 1 0 1 0 1 1	Quotient = 1
Shift	FB	1 0 1 0 1 1 0	
	FA	1 0 0 1 0 0 0	
SUBTRACT	FM	0 0 0 1 1 1 0	Quotient = 0
Shift	FB	0 0 1 1 1 0 0	
	FA	1 0 0 1 0 0 0	
ADD	FM	1 1 0 0 1 0 0	Quotient = 1
Shift	FB	1 0 0 1 0 0 0	
	FA	1 0 0 1 0 0 0	
SUBTRACT	FM	0 0 0 0 0 0 0	Quotient = 0
Shift	FB	0 0 0 0 0 0 0	
	FA	1 0 0 1 0 0 0	
ADD	FM	1 0 0 1 0 0 0	Quotient = 1
Shift	FB	0 0 1 0 0 0 0	
	FA	1 0 0 1 0 0 0	
SUBTRACT	FM	1 0 0 1 0 0 0	Quotient = 1
Shift	FB	0 0 1 0 0 0 0	
	FA	1 0 0 1 0 0 0	
SUBTRACT	FM	1 0 0 1 0 0 0	Quotient = 1

In example 2, the error arises in the fourth operation in which the correct quotient bit should be 1 with zero remainder. The following redundant operations thus propagate the negative divisor through to the final result as a false remainder.

In the event of a negative divisor (A37) the +FIA0 sequence becomes active and subtracts the divisor from the final residue. A mill test is then performed to determine whether a true remainder exists. If M = 0 the remainder is zero and the +FCK1 sequence becomes active setting FC (the quotient) + 1 to the mill. The correct quotient argument thus appears on the FQ highway and the exponent (FD) on the FP highway. If a true remainder exists ($M \neq 0$), further tests are carried out to determine whether rounding of the quotient argument will be required.

In the case of a positive divisor, the +FIA0 sequence and the mill test are by-passed, the tests for rounding being entered directly.

The rounding sequence is by-passed if X0 = 1 (unrounded result specified), or if FPOVR is set, in which case a corrective right shift is carried out before rounding in the normalising sequence. In either of these cases +FCK2 becomes active setting the quotient argument and exponent to their respective mills and thence to the FQ and FP highways.

Where rounding is specified (X0 = 0) and FPOVR is not set, a rounding increment is added to the quotient argument if the remainder is equal to or greater then half the divisor. The comparison between remainder and divisor is made in the following manner.

The +FIA3 sequence subtracts FA (the divisor) from FB (the remainder) thus forming R - D in the mill. The +FBK9 sequence strobes FQ (R - D) to FA and then forms the sum FA + FB, i.e. (R - D) + R, effectively twice the remainder minus the divisor. The sign test, M37 = B37, determines whether rounding (+FCYA) is performed, the increment thus being made only when 2R > D, i.e., when the sign of 2R - D is the same as that of R. The +FCK2 sequence places the quotient exponent and argument on the appropriate highways and entry is made to the normalising sequence. The setting of SP in the +FIA3 and +FBK9 sequences is performed solely for sequence gating purposes.

CHAPTER 5

NORMALISING

5.1 General

At the completion of any F.P.U. arithmetic operation, the result is placed in the argument and exponent mills and entry is made to the normalise microprogram which produces a corrected final answer in the form requested. The action taken in any normalise operation depends on the requested form of result (X0, X1), and on the state of the operand on entry to normalising. The microprogram thus has four main sequences which deal with the following operand states:-

1. The operand has a zero argument and is therefore equal to 0.

2. Single-bit overflow has occurred and the operand must be corrected by a single right shift of the argument with compensatory incrementing of the exponent.

3. Normalising is required.

4. Normalising is not required.

The decoding conditions for entry to each of the above sequences are indicated in the flow diagram, drawing 07/60 sheets 8 and 9, and in more detailed form in the normalise microprogram drawing 07/53.

5.2 Zero argument case

The zero argument case is detected by a mill test which, in the case of functions other than Fn 134, results in the activation of +FDN6 (element E37/1, drawing $07/53_0$). The purpose of the +FDN6 sequence is to form a zero operand in the FPU registers, i.e., a zero argument and an exponent of -256. The gatings D $\rightarrow$ N, $\overline{D}$ $\rightarrow$ N and FCYBE0 result in an exponent mill sum of zero, but the gating of -FCNORM and -FC1D at element G30/5, (drawing 07/48), forces the -256 bit on to highway line F048 thereby forming the required exponent. The zero argument is simultaneously placed in FB by the strobing of FQ to FB.

It will be noted that +FDN6 can be activated only if FPOVR is unset. The reason for this is that the setting of FPOVR is used, in division operations only, to register one-bit overflow. Hence, in division operations, a mill sum of zero on entry to normalising does not indicate a zero argument if FPOVR is set, since the subsequent corrective right shift will produce a non-zero argument.

The activation +FDN6 does, however, prime FPOVR but the setting is of significance only in multiplication operations as will be seen from the following paragraphs.

Multiplication is a special case in that it produces a double length argument, both halves of which must be mill-tested. Hence, on multiplication operations, if the initial mill test indicates that the m. s. half of the argument is zero, the+FCK3 sequence is activated. This sequence puts FC (l. s. half of argument) to the mill thereby increasing its significance by an effective left shift of 37 places and also making a compensatory adjustment of -37 to the exponent. The latter operation is performed by putting the exponent, FD, to FN and forming the constant -37 on the F0 highway. The constant is formed by the normal method of gating E → 0 and $\overline{E}$ → 0 to generate an "all ones" pattern and suppressing unwanted bits by the application of a suitable inhibiting signal. In this case the active signal +FCCK (C ⇒ K bistable) is applied to exponent mill slices 42 and 45.

The subsequent action depends on the result of a further mill test which, if the result is again zero, activates +FDN6 (element E27/6, drawing $07/53_0$). If the mill test result is non-zero the next step depends on whether the argument is normalised (FC36 = 1) or un-normalised (FC36 = 0). In the un-normalised case, +FBK4 becomes active at element E33/3 (drawing $07/53_0$) thus giving entry to the normalise sequence which is described in subsequent paragraphs. In the normalised case, tests are carried out to determine whether argument or exponent overflow exists.

The possible overflow conditions may be briefly summarised as follows. One-bit argument overflow may be present resulting in the indication M38 ≠ M37. Exponent overflow exists when the exponent, on entry to the normalising sequence, lies between the possible limits of +256 and +511. This condition gives the indication FP49 = 0, FP48 = 1. Exponent underflow may be present in either of two forms dependent on the magnitude of the underflow. Thus, if underflow exists on entry to the normalising sequence, the exponent value will be in the possible range of -257 to -512 and will be signified by the indication FP49 = 1, FP48 = 0. If subsequent normalising of the underflowed operand is required, the resultant adjustment of the exponent may reduce it to a value less than -512 in which case the false indication FP49 = 0, FP48 = 1 is given.

This false overflow indication is illustrated by the following example. It is assumed that an overflowed operand has an exponent value of -505 on entry to normalising, and a 37 place normalising shift is subsequently made.

Exponent on entry (-505)	1	0	0	0	0	0	0	1	1	1
Normalising action adds -37	1	1	1	1	0	1	1	0	1	1
Resultant mill sum, -542	0	1	1	1	1	0	0	0	1	0

The false overflow condition is detected by means of the FUN bistable which registers the exponent sign on entry to normalising.

The method of carrying out the overflow tests is as follows. If argument overflow exists (M38 $\neq$ M37), +FM 78 will be inactive at elements E36/2, E36/3 (drawing $07/53_0$), thereby enabling the activation of +FBK5 at element E31/2. This sequence applies a corrective right shift to the argument, increments the exponent by 1, and inverts the argument sign. If argument overflow is not present, the FBK5 sequence is not performed, the subsequent step being determined by the state of the exponent.

If neither overflow nor underflow is present, +FE0 is activated at F9/1 and the End of Order procedure is carried out, the final answer on FQ and FP being strobed to all F. P. U. registers. If overflow has occurred, the FE0 sequence is performed by the FOVR bistable (drawing $07/56_1$) is set via element H33/3. In the event of underflow of either of the previously stated magnitudes, a zero operand is stored via the +FDN6 sequence. The straight-forward case in which the exponent is not less than -512 is detected by element E31/1, the extreme case which results in FP49 = 0, FP48 = 1 and FUN set, being detected by element G19/1. In either case the FE0 sequence will be inhibited until the completion of the FDN6 sequence.

5.3 Single-bit overflow case

Elements E35/6 and E35/2 respectively detect the cases in which either FPOVR is unset and M38 $\neq$ M37 or FPOVR is set and M38 = M37. The former case is applicable to Fns. 132 and 133, the latter to Fn. 135. In either case the FBK5 sequence is entered, the operand being corrected by a single right shift of the argument, incrementation of the exponent and inversion of the sign bit. In the case of Fn 135, when FPOVR is set, rounding is also carried out, if specified by the activation of FCYA at element E35/3 and the resultant addition of FCYBA0/ to the argument mill.

On completion of the FBK5 sequence, entry is made to the exponent overflow tests as previously described.

5.4 Normalising required case

Entry is made to the normalising sequence, +FBK4, via element E37/6 when the following conditions are satisfied:-

FPOVR is unset
No argument mill overflow exists (M38 = M37 = M36)
The mill sum is non-zero
A normalised answer has been specified

The sequence left-shifts the argument, subtracts 1 from the exponent and checks the new values of the m.s. argument bits B36, B35, in order to determine the next step. If B36 = B35, i.e. the argument is still un-normalised, +FBK4 is re-activated via elements E34/5, E32/3 and E33/2, the normalising loop being re-entered.

In the case of Fn 134 which involves a double length argument, both halves of which may be significant, arrangements must be made to ensure simultaneous shifting of both halves (i.e. registers FB and FC) during the normalising process. As previously stated, the FCK3 sequence deals with cases in which the m.s. half of the product of a multiplication operation is zero. In cases where the m.s. half is non-zero, the +FCCL0/ sequence is activated via element E33/6 after each traverse of the +FBK4 loop. The l.s. half of the argument (in FC) is thus left shifted to the FQ highway simultaneously with each normalising shift which is applied to FB.

From the input gating arrangements of element E33/6 it will be noted that +FCCL0/ can be activated only when FPOVR is unset, since in multiplication operations FPOVR is set solely to indicate that the m.s. half of the argument is zero. Furthermore, +FCCL0/ is inhibited when +FEO is activated (i.e. when the End of Order Sequence is entered) to prevent interference with the FQ → FC gating which occurs at this time.

The final normalising shift is indicated by the condition B36 ≠ B35 (i.e. on completion of the shift the condition M37 ≠ M36 will be satisfied), and, on operations other than Fn 134, the exponent tests will be entered. In the case of Fn 134, the rounding sequence, +FBK6, will be carried out if the m.s. bit of the l.s. half of the argument is 1. Since rounding may result in mill overflow, provision is made for entry to the +FBK5 sequence prior to the subsequent exponent test.

Normalising not required

The +FBK10 sequence covers three particular cases in which no normalising action is necessary. These may be briefly summarised as follows:

a) Un-normalised answer specified (X1 = 1)

b) A normalised answer resulted directly from calculation.

c) (Fn 135 only). The special case of the operation $^{+a}/_{-a}$ which results in double overflow, i.e. single bit argument mill overflow and FPOVR set.

These three cases are detected by elements E36/6, E36/1 and E31/6 respectively. The +FBK10 sequence is a dummy operation in which the operand resulting from an arithmetic operation is submitted directly to the exponent tests. Provision is also made for entry to the +FBK5 sequence, since, in the case of Fn 134 only, rounding will occur (+FCYA sequence) if the m. s. bit of the l. s. half of the argument (C36) is 1. Argument overflow could therefore arise as a result of the rounding operation.

Dependent on the result of the subsequent exponent tests, entry will be made to +FE0 by the appropriate path.

End of Order Sequence (+FE0)

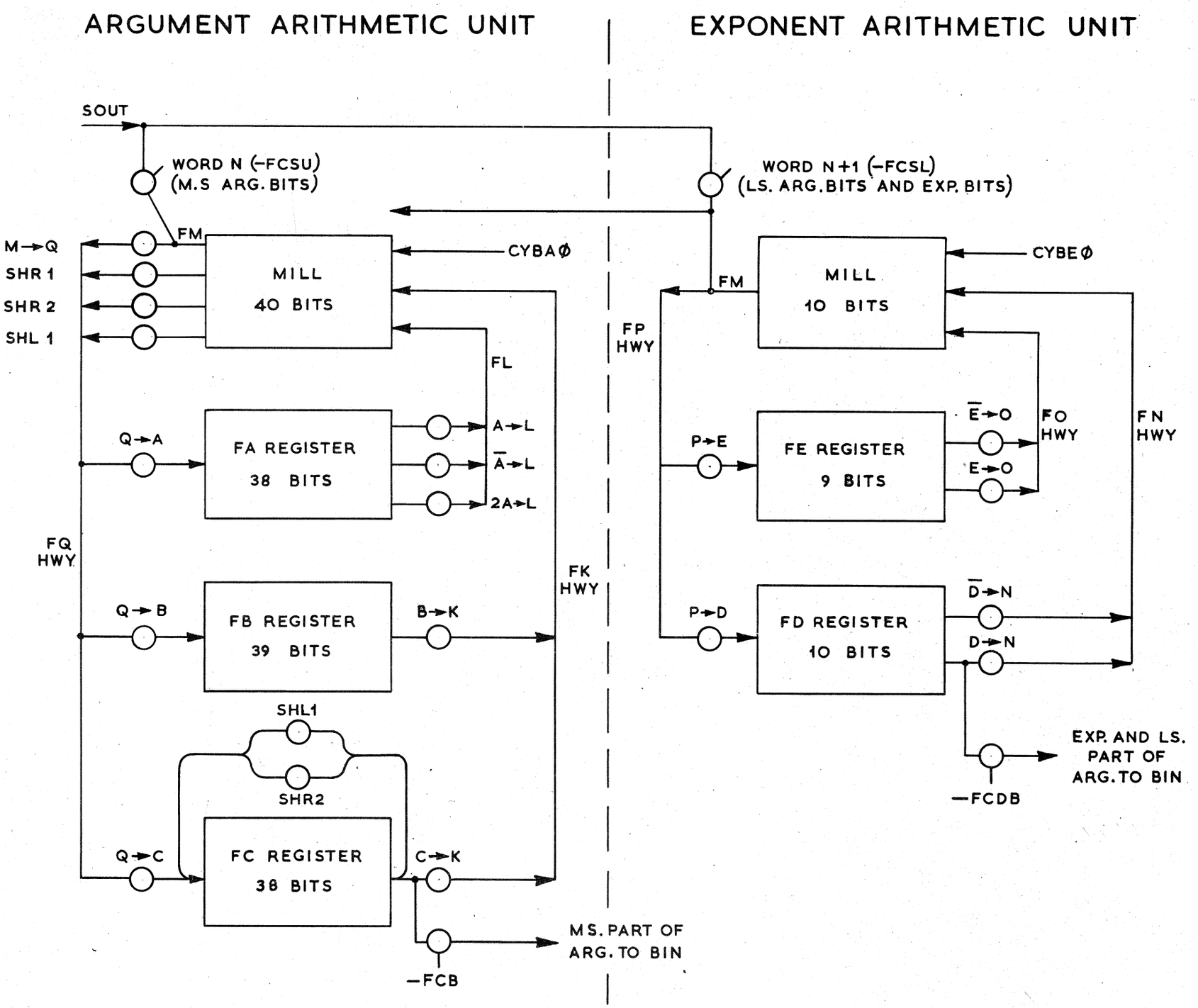
The +FE0 sequence terminates all normalise operations regardless of the path taken through the microprogram. Its functions are as follows:-

(a) Strobing of the final answer from the FQ and FP highways to F. P. U. registers FA, FB, FC, and FD, FE. +FE0 is thus applied to pulse generator circuits to initiate the Q → A, Q → B, and Q → C strobe signals and similarly the P → D and P → E strobes (drawing 07/50).

(b) Clearing of the F. P. U. registers and control bistables. -FE0 is thus applied to element F12/2 (drawing $07/50_2$) to produce +FCFX which primes the clearing of the FF and FX registers and also initiates the register strobe signal FXFX (element F5/1, drawing $07/50_1$). -FE0 also primes the resetting of FPOVR via element F29/5 (drawing $07/56_1$) thereby initiating +FRST which primes the resetting of the NORM bistable ($07/53_1$) and the FUN bistable ($07/49_0$).

(c) +FE0 primes the setting of FOVR via elements F27/5 and H33/3 when exponent overflow has occurred.

(d) Resetting of the F. P. U. BUSY status.
-FE0 primes the setting of the END bistable ($07/50_0$) thereby initiating the resetting of BUSY and the setting of WAIT. -FEND is applied via element F9/6 to inhibit the action of the F. P. U. clock. The resetting of BUSY primes the resetting of END and also de-activates the signal +FBUSY2 via which the status of the F. P. U. is indicated to the central machine microprogram.


ARGUMENT ARITHMETIC UNIT
EXPONENT ARITHMETIC UNIT
SOUT
WORD N (-FCSU)
(M.S ARG. BITS)
WORD N+1 (-FCSL)
(LS. ARG. BITS AND EXP. BITS)
FM
M→Q
SHR 1
SHR 2
SHL 1
MILL
40 BITS
CYBAØ
FL
A→L
Ā→L
2A→L
Q→A
FA REGISTER
38 BITS
FQ HWY
FK HWY
Q→B
FB REGISTER
39 BITS
B→K
SHL1
SHR2
Q→C
FC REGISTER
38 BITS
C→K
MS. PART OF ARG. TO BIN
-FCB
FP HWY
FM
MILL
10 BITS
CYBEØ
P→E
FE REGISTER
9 BITS
Ē→O
E→O
FO HWY
FN HWY
P→D
FD REGISTER
10 BITS
D̄→N
D→N
EXP. AND LS. PART OF ARG. TO BIN
-FCDB