

## SECTION II: THEORY OF OPERATION

### 1.0 BLOCK DIAGRAM ANALYSIS

Before attempting to understand the operation of the calculator in detail, it is important to become acquainted with the basic characteristics of the major functional sections of the unit.

#### 1.1 Timing (See Figure 2-1)

Timing for all calculator operations is derived from a 600 kc clock pulse generator. The output of this generator is the  $CL_F$  clock pulse rate.  $CL_F$  pulses are counted down to the 75 kc CL clock rate by the V counter. This is the calculator bit rate.

The status of the V counter is continuously decoded by the V count decoder which provides 8 sub-bit period signals on separate lines. The carry output from the V counter (CL) is applied to other sections of the calculator and is also applied to the bit counter.

The CL pulses are counted down by the bit counter to the 6.25 kc digit rate. The status of the bit counter is continuously decoded by the bit count decoder to obtain twelve outputs on separate lines representing twelve equal sub-divisions of each digit period. The carry output of the bit counter (the 6.25 kc digit rate) is divided by 30 in the digit counter to obtain the  $208 - 1/3$  cps word rate. The binary-coded-decimal data from the digit counter are applied to the  $K_P$  logic, the  $K_{DC}$  logic, and the digit count decoder.

In the  $K_P$  logic, the position of the decimal point switch is compared with the status of the digit counter and a pulse is generated during the first digit period following the decimal point (that is, the units' digit period). This is defined as  $K_P$  time.

In the  $K_{DC}$  logic, the status of the digit counter is compared with the status of a counter designated as the C counter. Upon coincidence of the status of these two counters, a  $K_{DC}$  pulse is generated (along with its complement  $\overline{K_{DC}}$ ).

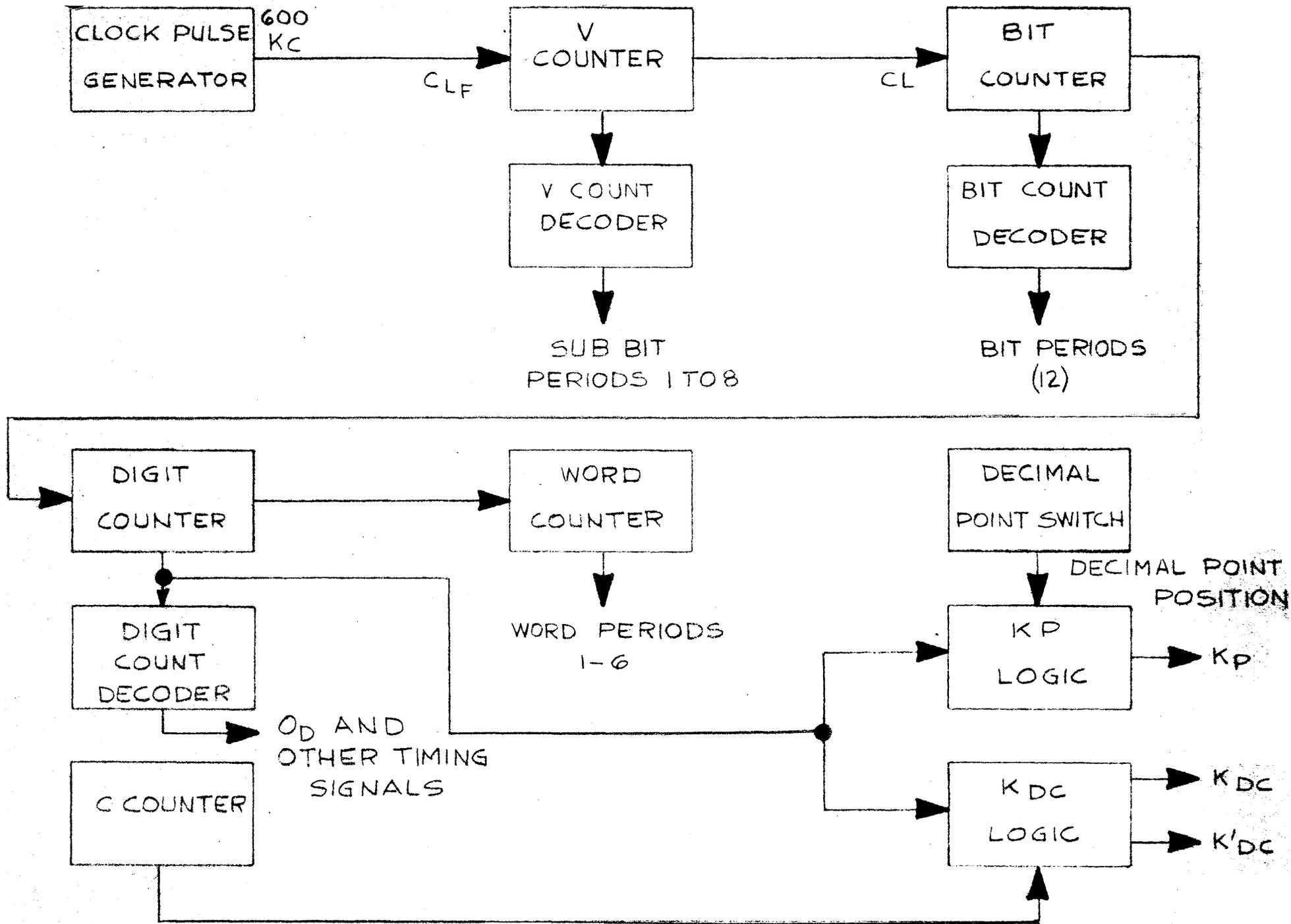


FIG. 2-1: TIMING SECTION, BLOCK DIAGRAM

## SECTION II: THEORY OF OPERATION

### 1.1 Timing (Continued)

During the digit period following the  $K_{DC}$  period, the  $\overline{K'_{DC}}$  and  $K'_{DC}$  pulses are generated. These signals are used to select a single digit of a word for entry into a storage register or for participation in some operation. The digit count decoder develops timing signals representing various significant times during the word cycle. These are used for various control purposes.

Prominent among these is the  $\overline{\Theta_D}$  signal. This is generated during the first bit period of digit time D26 which is after the end of the numerical portion of the word cycle. This signal is used to synchronize the implementation of keyboard or other external inputs with the internally-generated word cycle as well as for a number of other control functions. The carry output from the digit counter (word rate) is applied to the word counter. The word counter divides the word rate by six. The six binary-coded-decimal output states of the word counter represent words 1 through 6.

### 1.2 Memory

The calculator memory is a delay line with three output taps; RD1, RD2, and RD3. (See Figure 2-2) Data passes down the delay line in serial format, least significant digit first. One-word time elapses while data passes from the delay line input to the RD2 tap. Thus, data can be recirculated indefinitely by feeding it back from the RD2 tap to the delay line.

One-word period plus one-digit period elapses while data passes from the delay line input to the RD3 tap. The additional delay of one-digit period increases the significance of each digit by one order. Thus, by feeding the output from the RD3 tap back to the memory delay line input, a shift left (multiplication by 10) is accomplished. One-word period minus one-digit period elapses while data passes from the delay line input to the RD1 tap. Thus, by feeding the output from the RD1 tap back to the memory delay line input, a shift right (division by 10) is accomplished. The output from the RD1 tap is also applied to the arithmetic and display sections where one-digit period delays are encountered.

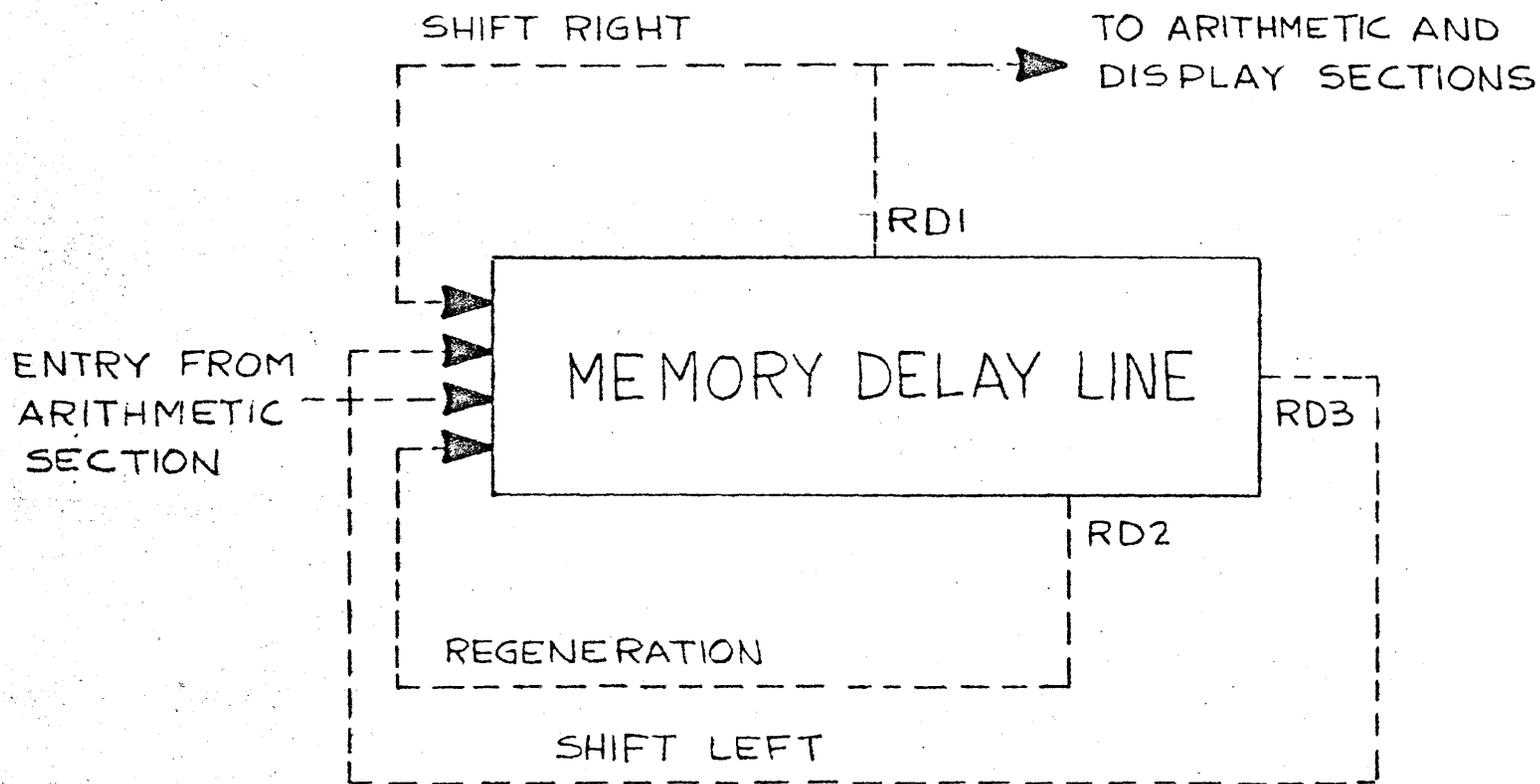


FIG. 2-2 MEMORY SECTION, DATA PATHS

## SECTION II: THEORY OF OPERATION

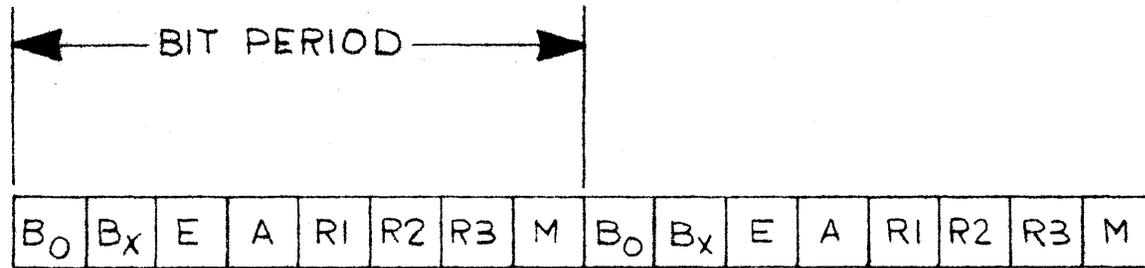
### 1.2 Memory (Continued)

Data entry to the delay line is from the arithmetic section. Six words are simultaneously stored in the delay line; the multiplier-quotient word, the entry word, the accumulator word, word R1, word R2, and word R3.

Each digit is represented by a train of  $n$  pulses (bits) where  $n$  is the value of the digit. For example, the digit 7 is represented by a train of 7 pulses (bits) occurring in a single digit period. The bits of the six stored words are interleaved as shown in Figure 2-3. This form of multiplexing allows all six words to circulate through the delay line during each word period. By sampling a delay line output tap at the appropriate sub-bit times, the bits of a particular word can be separated from the multiplexed data.

As an example, one word may be fed back from the RD1 tap to accomplish a shift right while the other five words are fed back from the RD2 tap so as to recirculate through the memory without change. Multiplexing is accomplished by sampling selected input lines at appropriate times during each bit period.

The six memory locations (defined in terms of time within bit periods) may be thought of as six storage registers. This is a useful convention, because the gating between the memory locations and the arithmetic section is not the same for all locations. As the names suggest, the multiplier-quotient, entry, and accumulator registers are used to store numbers involved in arithmetic operations, while the R1, R2, and R3 registers provide additional storage for data.



E = ENTRY BIT  
 A = ACCUMULATOR BIT  
 R<sub>1</sub> = R<sub>1</sub> BIT  
 R<sub>2</sub> = R<sub>2</sub> BIT  
 R<sub>3</sub> = R<sub>3</sub> BIT  
 M = MULTIPLIER-QUOTIENT BIT  
 B<sub>0</sub> & B<sub>x</sub> ARE UNUSED REGISTERS

FIG.2-3: DATA STORAGE FORMAT

## SECTION II: THEORY OF OPERATION

### 1.3

#### Arithmetic Section

The arithmetic section performs addition, subtraction, multiplication, division, and extraction of square roots. The arithmetic section is also used in the entry of data from the numeral keyboard to a storage register and in the transfer of data from one storage register to another.

All arithmetic operations are accomplished by various combinations of three basic operations; add, shift left (multiply by 10), and shift right (divide by 10). Addition is performed by a counter which operates on one pair of digits at a time and adds them by counting the bits of both digits. Subtraction is performed by complementing the subtrahend and adding it to the minuend.

Figure 2-4 illustrates data flow in the arithmetic section. Notice that data can be entered in parallel to the A counter from the numeral keyboard or entered in serial format via the AC and AC' inputs. Since the AC and AC' inputs are sampled at different times, simultaneous entry from two sources can be accomplished.

Either the true value or complement of any of the register outputs can be selected for connection to the AC input of the A counter. Since the accumulator register output (A) can be connected to the AC' input, this means that the contents of any other register can be added to or subtracted from the contents of the accumulator register. Connection of the multiplier-quotient register output to both the AC and AC' inputs is provided. This allows the contents of the multiplier-quotient register to be doubled, which is required during the extraction of a square root.

Words are handled a digit at a time by the A counter. When the count accumulation is completed at the end of a digit period, the accumulated count can be read out of the A counter and into the R counter through the A transfer gates. The A counter is then reset and is ready to process the next digit or pair of digits (depending upon whether inputs are applied to both the AC and AC' input lines). If a carry occurs during summation of the bits of a pair of digits, the carry flip-flop is set. The output of the carry flip-flop is then read onto the AC input at the start of the next digit time, after which the carry flip-flop is reset.

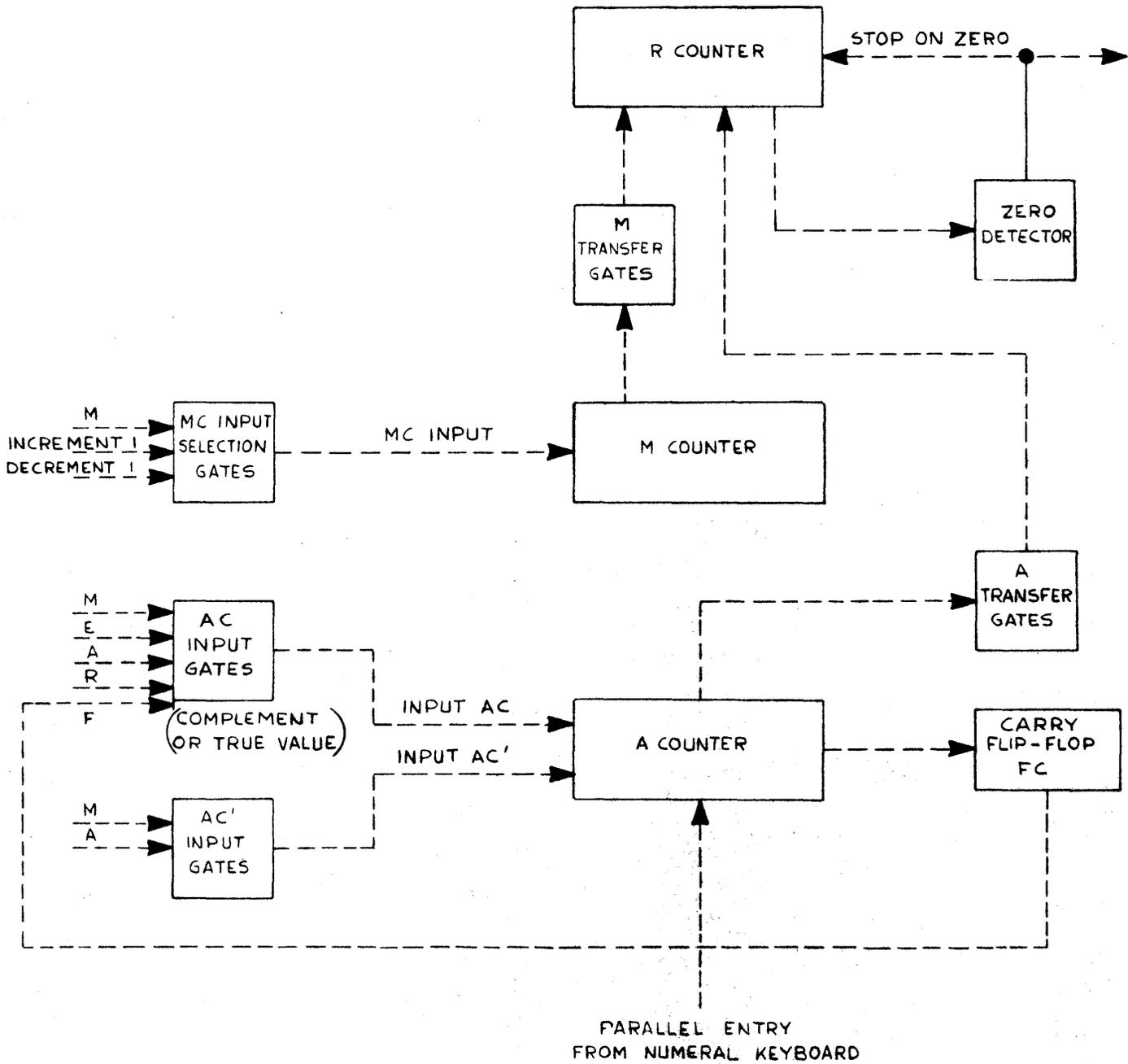


FIG. 2-4: ARITHMETIC SECTION, DATA FLOW, BLOCK DIAGRAM

## SECTION II: THEORY OF OPERATION

### 1.3 Arithmetic Section (Continued)

The discussion of the arithmetic section up to this point has been in terms of signal flow without reference to the control logic required to implement this flow. The basic control elements of the arithmetic section are illustrated in Figure 2-5.

The cycle counter and cycle count decoder are used in the implementation of all routines but are particularly significant in multi-cycle routines. Signals from the cycle count decoder are used to gate operations which are performed during only a single word cycle or series of word cycles rather than during every word cycle of a function routine. In the reset condition, an inhibit signal is supplied to clock pulse and transfer gates. When a function is activated, the  $\overline{OD}$  signal is provided to the cycle counter input logic and the cycle counter steps out of the reset condition, removing the inhibit signal. Thus, the start of function implementation is synchronized with the word cycle timing.

The cycle counter has seven active states. The order in which it passes through these states is not entirely fixed by the counting logic but depends upon external inputs to the cycle counter input logic from the M counter or other intermediate results, as represented by the status of the LO flip-flop.

The LO flip-flop is used to establish the time relationship between two signals or the occurrence of some particular event. For example, in several routines it is necessary to establish the time relationship between the  $\overline{K_{DC}}$  and  $\overline{K_P}$  signals. One signal is applied to the set input of the flip-flop and the other to the reset input. The flip-flop is initially reset. Thus, the status of the flip-flop at the end of the word cycle establishes which of the two signals precedes the other. This information is then used in determining whether or not to advance the cycle counter or modify the actions being performed in some other manner.

## SECTION II: THEORY OF OPERATION

### 1.3 Arithmetic Section (Continued)

Data readout from the arithmetic section is in decimal, serial format from the R counter. The R counter, which receives data from the A counter (or, in some cases, the M counter) at the end of each digit cycle, counts down to zero during the succeeding digit cycle. A zero detector detects whether the R counter is at zero or not. The time required for the R counter to reach zero depends upon the value of the data read into the R counter. The stop-on-zero output of the zero detector is used to gate pulses (bits) into a selected register. Since the rate of these pulses is equal to the counting rate of the R counter, the number of pulses counted into the selected register is equal to the count initially set into the R counter. This arrangement allows each digit resulting from an arithmetic operation to be read into the selected register at the same time that the succeeding input digits are being processed.

There is only one data input that can be applied to the M counter. This is the output of the multiplier-quotient (M) register. During multiplication, the multiplier is stored in the multiplier-quotient register. Multiplier digits are successively loaded into the M counter where they are used to control the number of additions of the multiplicand to the partial products in the shift and add routine by which multiplication is performed.

After each addition, a decrement 1 pulse train is applied to the M counter causing the count to decrease by one. The M counter is a scale-of-10 counter which counts forward. In order to decrease the count by 1, it is supplied with 9 pulses. (For example, when 9 is added to an initial count of 5, a count of 14 is obtained. However, the 1 in the ten's order is lost, leaving 4.)

During division and root extraction, routines which make use of shifting and subtraction, the number of subtractions is accumulated in the M counter. During such times, the M counter receives the increment 1 input once during each subtraction cycle.

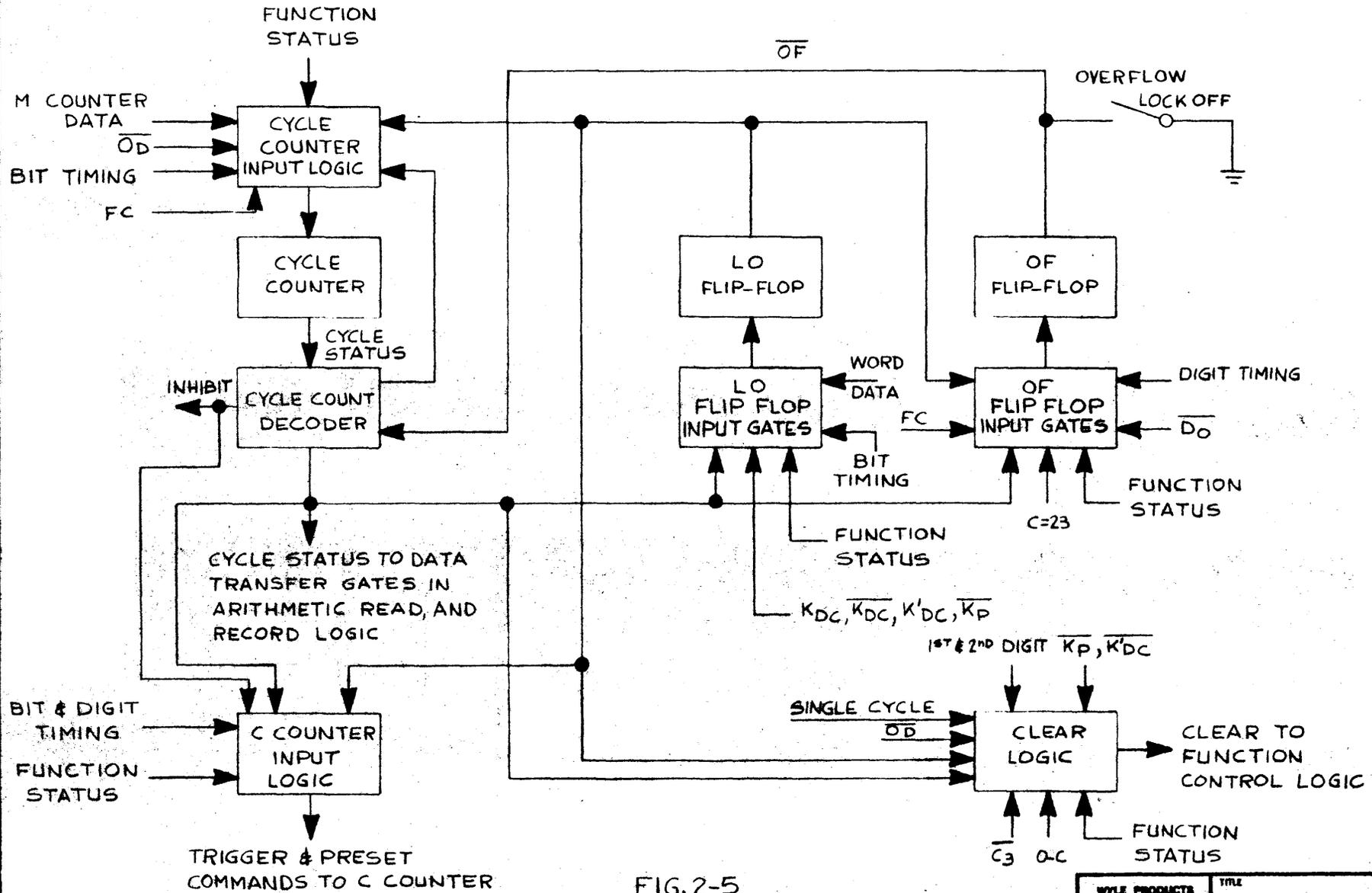


FIG.2-5  
 ARITHMETIC SECTION,  
 CONTROL ELEMENTS, BLOCK DIAGRAM

WYLE PRODUCTS DIVISION OF WYLE LABORATORIES 8100000, CALIFORNIA	TITLE
	S.K.

## SECTION II: THEORY OF OPERATION

### 1.3 Arithmetic Section (Continued)

The overflow (OF) flip-flop is used to indicate when the capacity of the calculator is exceeded during the course of some operation. Examples of such a condition are the generation of a carry (FC) from the highest order during an addition or multiplication routine or the shifting of numerical data out of the highest numerical order during a shift left operation of an arithmetic function. The OF flip-flop input gates receive data signals such as the carry signal (FC) and the left-shifted data ( $\overline{D}_0$ ); digit timing signals; and control status signals such as LO flip-flop status, cycle counter status, and function status. Function status is significant because in inverse operations a carry-out of the highest order represents a normal condition, whereas in direct operations it represents an overflow condition.

The clear logic generates a clear signal in response to inputs which indicate that the particular function routine being performed has been completed. In the case of operations requiring a single word cycle, the function control logic supplies a single-cycle signal to the clear logic. In response to this input and the  $O_D$  signal, the clear pulse is generated at  $O_D$  time.

During multi-cycle arithmetic routines, the C count is decreased as one digit after another is processed. For example, in multiplication, the multiplier digits are processed from left to right one at a time. When the C count reaches zero (O-C), the final digit is being processed. In response to this signal, the clear logic prepares to terminate the routine. On the next cycle when the C count has changed from zero to 23, the routine is terminated.

In the case of the alignment routine during which numerals entered before the decimal point (via the keyboard) are aligned with respect to the decimal point, the status of the LO flip-flop during the initial stage of the routine indicates when no alignment is necessary, or the coincidence of the  $\overline{K}_{DC}$  and  $\overline{K}_P$  signals indicates that alignment has been completed. Thus, these signals are applied to the clear logic for use in generating a clear pulse.

The C counter appears in Figure 2-1 as an element of the timing section. However, the status of the C counter is controlled by the function control and arithmetic sections. Thus, the C counter input logic is shown in Figure 2-5. Inputs to this logic which determine the value to which the C counter is initially set and the

## SECTION II: THEORY OF OPERATION

### 1.3 Arithmetic Section (Continued)

circumstances under which the C count is later increased or decreased are supplied from the function control logic, the cycle count decoder, and the LO flip-flop. Bit and digit timing signals are used in obtaining the required number of triggers to count forward 1 or 2 or to count down 1 (by counting forward the number of counts which is 1 less than the scale of the counter).

## SECTION II: THEORY OF OPERATION

### 1.4 Function Storage and Function Control (See Figure 2-6)

When a function or numeral key on the calculator keyboard is actuated, a corresponding function storage flip-flop is set and remains set until the function routine is completed. Actuation of the function storage flip-flops is controlled by the supervisory logic to prevent selection of one function while another function routine is in progress and to prevent double cycles of a function if the key remains actuated when the first cycle is completed.

The function status, in the form of the output or outputs from the function storage flip-flops, is supplied to various gates in arithmetic and memory reading and recording logic as well as to other control elements of the arithmetic logic such as the cycle counter, the LO flip-flop input gates, and the C counter input gates.

The function status signals are also applied to the function control logic where they are used to provide certain signals associated with groups of functions. A single-cycle signal is supplied to the clear logic in the arithmetic section when a routine requiring a single-word cycle is being performed. The START signal which clears the word counter, A counter, and C counter is generated at the start of most, but not all, function routines. When the START signal is supplied at the start of a routine, then the preset 23 and edit signals are generated at the termination of the routine. The preset 23 signal presets the C counter to 23, facilitating the entry of numerical data. The edit signal sets the edit flip-flop, initiating an edit routine which results in the brightening of all significant digits of the displayed data.

Clear CA and/or clear CE signals are generated at the end of certain arithmetic operations. These set the CA and/or CE function storage flip-flops, causing the accumulator and/or entry registers to be cleared. A keep-remainders input from the KEEP REMAINDER switch inhibits this function.

The function storage enabled signal supplied to the cycle counter and digit count decoder resets the cycle counter and inhibits the generation of an  $\overline{O_D}$  signal when all function storage flip-flops are reset.

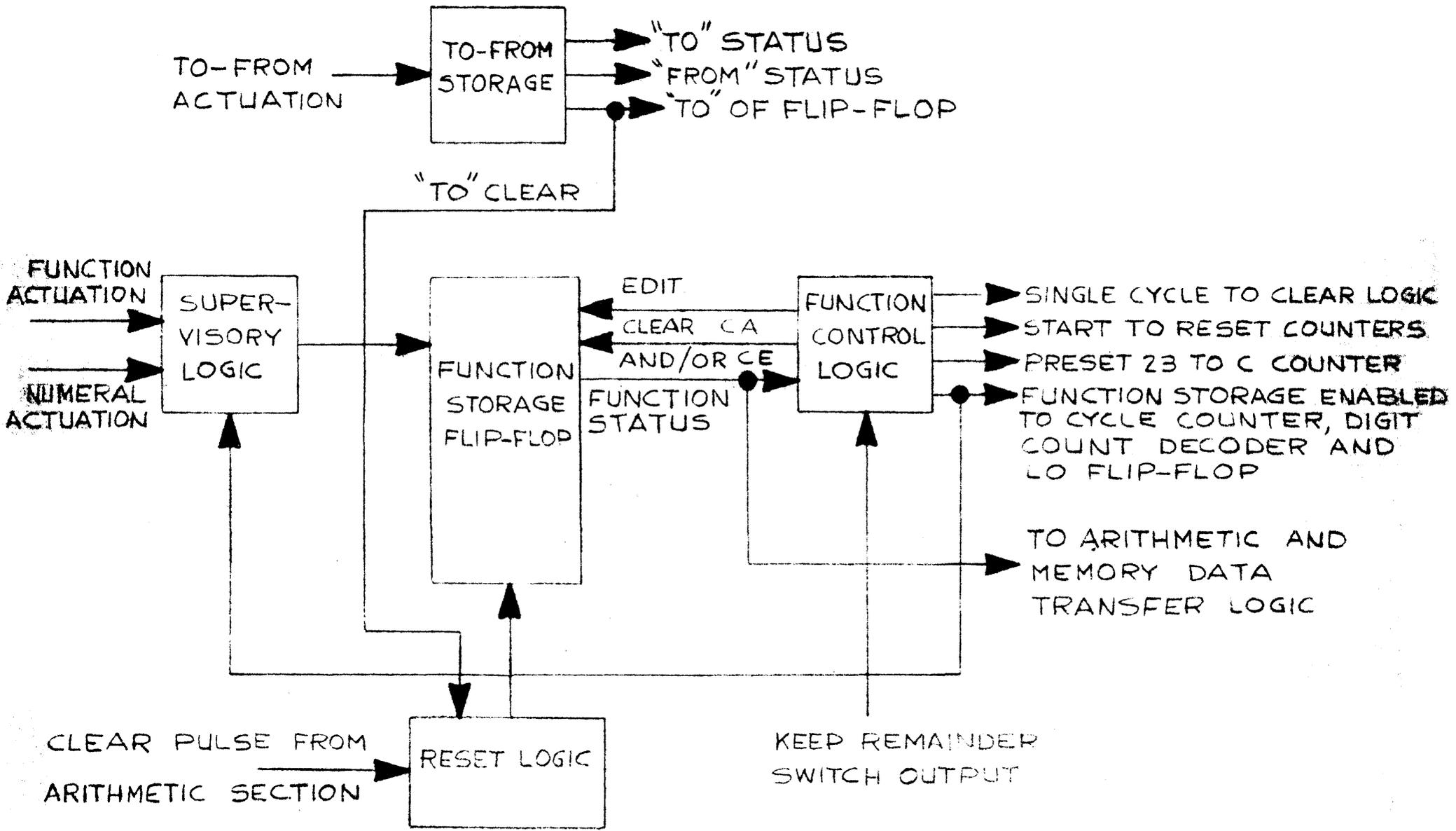


FIG. 2-6  
 FUNCTION STORAGE AND FUNCTION  
 CONTROL, BLOCK DIAGRAM

## SECTION II: THEORY OF OPERATION

### 1.5 Display Section (See Figure 2-7)

The data stored in the six registers are continuously displayed on a cathode ray tube. Reading from top to bottom, the displayed words are the M-Q word, the entry word, the accumulator word, the register 1 word, the register 2 word, and the register 3 word. Significant digits of each word are brightened. Additional brightening is provided for the  $K_{DC}$  digit of the selected TO word. A numeral (0) to the left of the selected FROM word provides visual tagging of the FROM selection. The decimal point also appears on the display at the appropriate position in each word.

Demultiplexed data are gated by the N counter input gates according to the status of the word counter. Gated data are entered serially into the N counter during the numerical portion of each digit cycle. At  $P_{11}$  time, the digit data is transferred to the Q register where it is held until  $P_{10}$  time of the succeeding digit cycle. At  $P_0$  time of each digit cycle the N counter is reset in preparation for receipt of new digit data. BCD character data from the Q register is decoded into decimal data on ten separate lines. These data are supplied to the optical coder which generates the X and Y deflection voltages required to trace digit patterns on the CRT. The optical coder also supplies an unblank signal which unblanks the CRT during the portion of each digit time when the character is being generated.

The X deflection data from the optical coder are summed with analog digit time data in the X deflection summing amplifier to produce a horizontal deflection voltage such that each digit trace appears in the appropriate digit position. The Y deflection voltage from the optical coder is summed with analog word time data in the Y deflection summing amplifier to produce a vertical deflection voltage, such that each digit trace appears in the appropriate word position.

The brightness of individual digits is determined by the status of the dim gate. When all function storage flip-flops are reset (function storage enabled line at low level), the LO flip-flop monitors the NC IN line during  $P_{10}$  time of each digit cycle. If the digit is significant, the LO flip-flop is placed in the set status providing a low level on the LO line to the dim gate. This causes the display to be brightened during the next digit period, when this digit is being held in the Q register and is being optically coded. The I input to the dim gate causes a brightening of the display during function routines. The  $P_D$  input to the dim gate causes a brightening during non-numerical portions of each digit cycle for decimal point display. The FROM input causes a brightening of the display at the time the FROM mark O is being traced on the CRT.

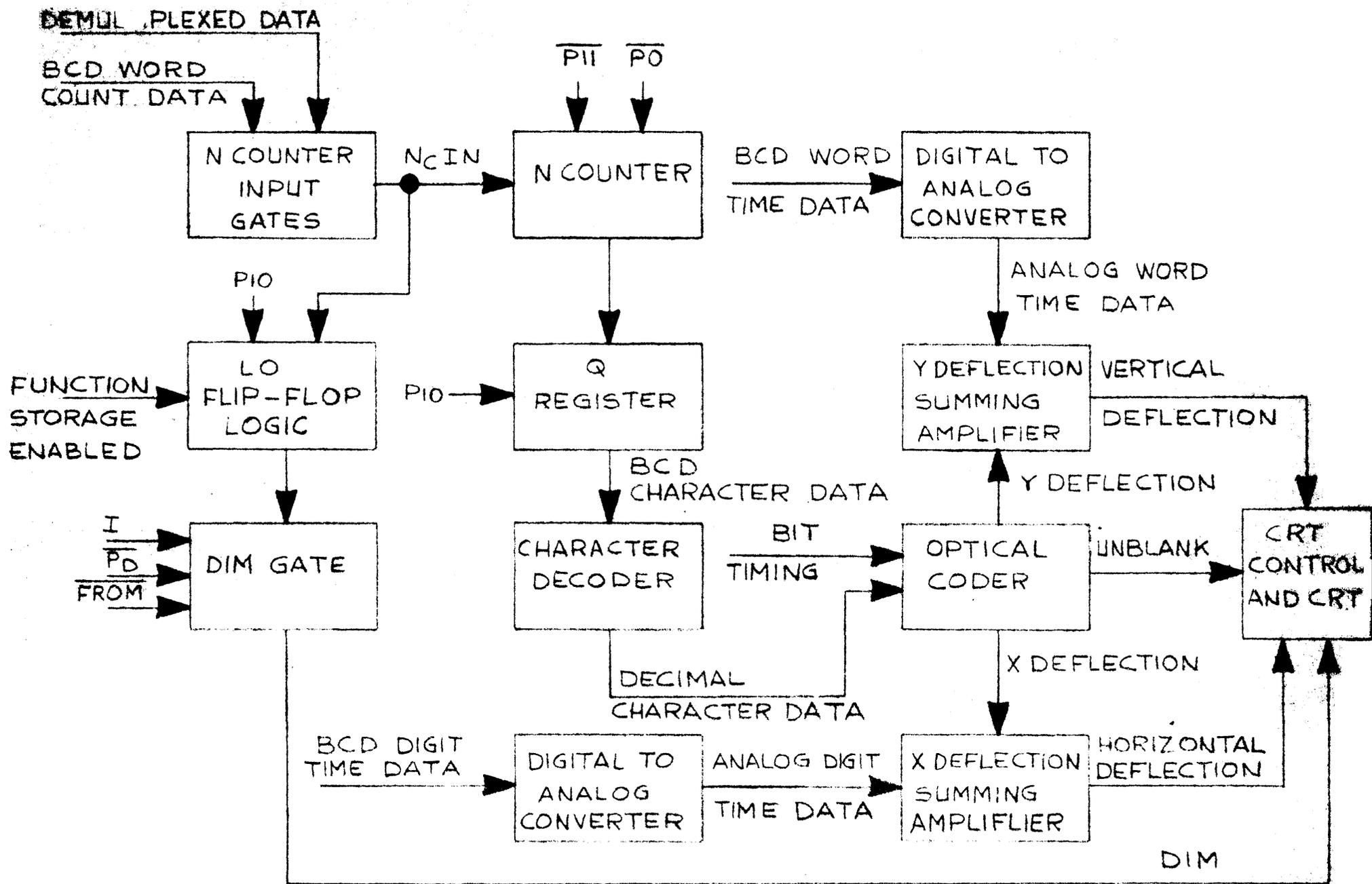


FIG. 2-7  
 DISPLAY SECTION BLOCK DIAGRAM

## SECTION II: THEORY OF OPERATION

### 2.0 SUMMARIES OF FUNCTION ROUTINES

In the paragraphs that follow each of the routines performed by the calculator is summarized. Refer to the summaries to gain a general understanding of the steps performed during each routine and the theoretical considerations underlying each step. Refer to the detailed descriptions for data concerning specific logic elements involved in implementing each step of a routine.

Routines may be classified as single-cycle or multi-single. Single-cycle routines require but one word cycle. In each single-cycle routine, the cycle counter is advanced to the IA status and is returned to the reset status at the completion of the routine. Single-cycle routines include the following:

- a. Numeral entry (also called "Key")
- b. Back space
- c. Forward space
- d. Transfer
- e. Shift left
- f. Shift right
- g. Addition
- h. Subtraction
- i. Clear entry register
- j. Clear multiplier-quotient register
- k. Clear accumulator register

Multi-cycle routines include the following:

- a. Clear and Multiply
- b. Multiply and Add
- c. Multiply and Subtract
- d. Division
- e. Square Root Extraction
- f. Alignment (Decimal Point)
- g. Edit

## SECTION II: THEORY OF OPERATION

### 2.0 SUMMARIES OF FUNCTION ROUTINES (Continued)

During multi-cycle operations, the cycle counter passes through a number of different phases. The changes in cycle counter status are controlled in accordance with the function selection and in some cases are also dependent upon the intermediate results obtained. Thus, the number of word cycles required to complete multi-cycle operation varies as a function of the specific data that is processed.

The majority of function routines begin with the generation of a START signal and are followed by an edit routine. The START signal resets the word counter, the C counter, the A counter, and the LO flip-flop. The generation of the START signal and of the following edit routine is selected by a common OR gate. Those functions which are not begun with a START signal and followed by an edit routine are numeral entry, alignment, back space, forward space, and shift right. Each numeral entry decreases the C count by 1, allowing the next numeral to be entered one digit position to the right. In order to obtain this relationship, the C counter must not be reset. In the same way, the status of the C counter after alignment is such that the next numeral entered is entered just to the right of the decimal point.

The forward space and back space functions are used to obtain a required C counter status. During numeral entry each digit entered is brightened by the addition of a tag bit ( $P_{10}$  time). The other routines not followed by an edit routine do not affect the previous edit of data.

When any routine is initiated, a corresponding function storage flip-flop is set. This results in the advancement of the cycle counter to the IA status, in the generation of a START signal for some functions, and in the priming of a number of AND gates which provide data or control signals during the accomplishment of the function.

Some operations involve the processing of a complete word or two complete words during a word cycle. In other operations, only one or two digits are processed. In these latter cases, the status of the C counter becomes significant. This establishes the C counter-digit counter coincidence digit time,  $K_{DC}$ . The  $K_{DC}$  digit appears at the RD1 tap at  $K_{DC}$  time and appears at the RD2 tap at  $K'_{DC}$  time. Thus, the  $K_{DC}$  signal is used to gate a particular digit to the arithmetic logic A counter from the RD1 tap via the demultiplexer logic. The  $K'_{DC}$  signal is used to gate a particular digit into a storage register. (Since the RD2 tap receives data one word time after it is applied to the recording tap, a digit must be recorded at  $K'_{DC}$  time in order to appear at the RD2 tap at  $K'_{DC}$  time.)

## SECTION II: THEORY OF OPERATION

### 2.1 Numeral Entry Routine

When one of the numeral keys is actuated or when a numeral is entered via the card reader, the KY function storage flip-flop is set, the numeral is entered in binary-coded decimal format (1,2,4,5) into the A counter, and the cycle counter is advanced to the IA status.

With the KY flip-flop set, a data transfer path from the R counter to the selected TO register is set up for  $K'_{DC}$  digit time only. The clearing of the A counter at the end of the digit period is inhibited. During each digit period, the number held in the A counter is transferred to the R counter and during the succeeding digit period is read out of the R counter in serial format.

During digit time  $K'_{DC}$ , the data read out of the R counter is gated to the selected TO register. After the numerical portion of the word cycle is completed, the C count is decreased by 1. Thus, if a second numeral entry follows the first, the second numeral is entered to the right of the first. Or, in more general terms, each successive numeral is entered to the right of the preceding numeral.

Figure 2-8 illustrates data flow and control relationships during numeral entry. In advancing to the IA status, the cycle counter removes the inhibit signal from the transfer gates preparing for the transfer of data from the R counter to the storage register. The IA status signal is used in developing the clear logic signal which resets the KY flip-flop and terminates the numeral entry routine.

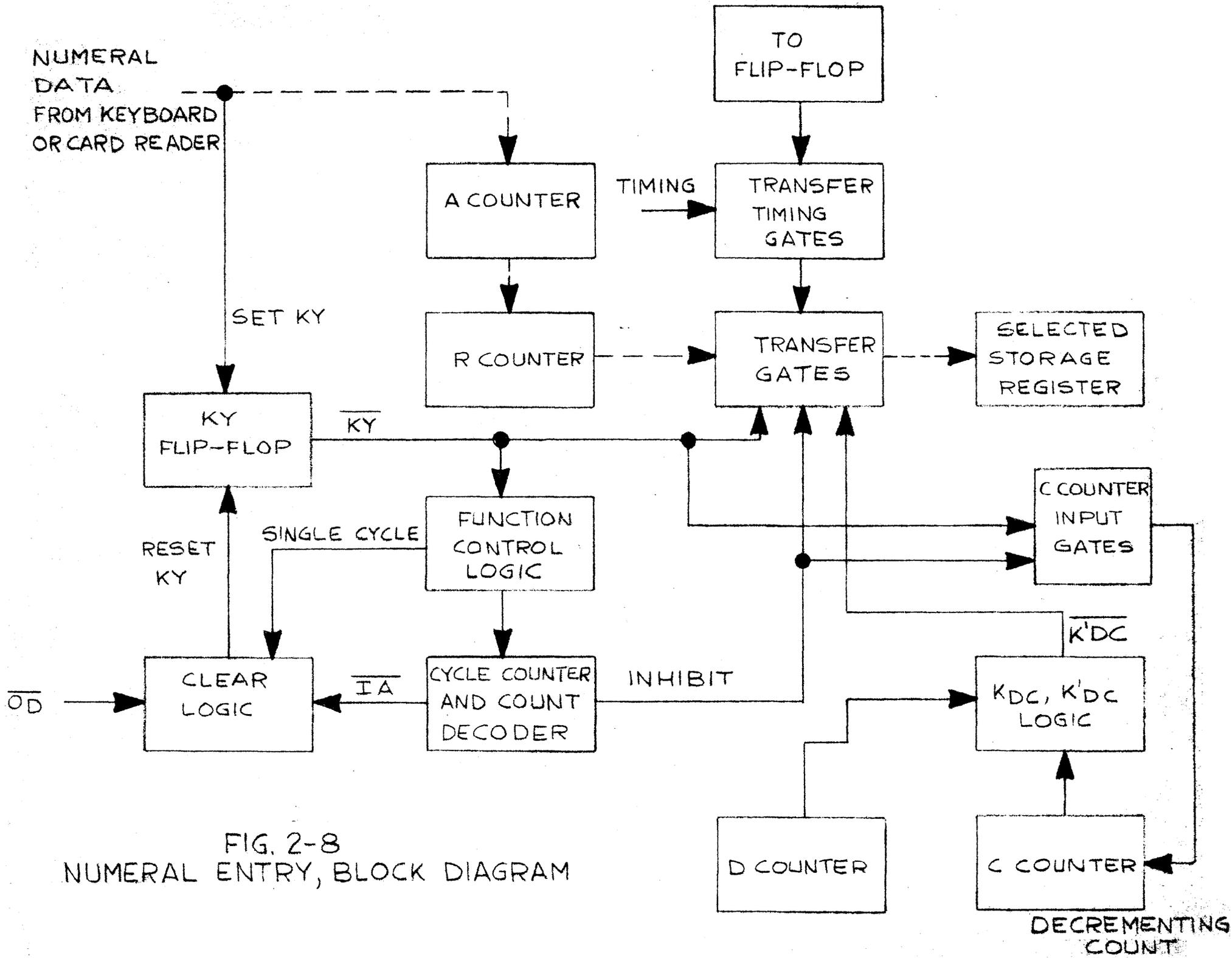


FIG. 2-8  
 NUMERAL ENTRY, BLOCK DIAGRAM

## SECTION II: THEORY OF OPERATION

### 2.2

#### Alignment Routine

The alignment routine is initiated in response to decimal point entry via the keyboard or card reader. It shifts data entered prior to the decimal point as required to position the last digit entered just to the left of the decimal point (that is, in the  $K_P$  digit position). It is convenient to assume some specific position of the decimal point switch in discussing the alignment routine. Assume that the decimal point switch is set to 12. There are 12 available numerical digit positions to the left of the decimal point (D23 to D12) and 12 to the right of the decimal point (D11 to D0). Three cases must be considered:

a. Exactly 12 digits are entered prior to the decimal point entry

In this case, the last digit entered is entered into the  $K_P$  digit position and no shifting of the data is required. In this case, the alignment routine is terminated without shifting the data and without changing the C count. If another numeral is now entered, it is entered into the digit position just to the right of the decimal point (D11).

b. More than 12 digits are entered prior to the decimal point entry

This represents an overflow condition. If the integral portion of data to be entered has more than 12 digits, then the decimal point must be positioned farther to the right. If this condition is encountered during the alignment routine, the overflow (OF) flip-flop is set to indicate that an adjustment is required.

c. Few than 12 digits are entered prior to decimal point entry

In this case, the last digit entered is in some digit position to the left of the D12 position and at least one shift of data to the right is required.

If at least one shift to the right is required, a sub-routine is entered during which the data is shifted one place to the right during each word cycle and a test is made to see whether the shifted data is aligned so that the last digit entered is in the  $K_P$  digit position.

## SECTION II: THEORY OF OPERATION

### 2.2 Alignment Routine (Continued)

The sub-routine continues for as many word cycles as are required to obtain this condition. In determining the position of the last digit entered with respect to the  $K_P$  digit, use is made of the known relationship between the position of the last digit entered and the  $K'_{DC}$  digit position.

During each numeral entry routine, the data is entered in the current  $K_{DC}$  digit position and, at the end of the routine, the C count is decreased by 1. Thus, at the end of each numeral entry routine, the numeral just entered is in the  $K'_{DC}$  digit position. From this it follows that the three cases just considered can be defined in terms of the relationship of the  $K'_{DC}$  and  $K_P$  digit times. After exactly 12 digits have been entered,  $K'_{DC}$  time and  $K_P$  time are in coincidence. If fewer than 12 digits have been entered, then the  $K'_{DC}$  digit position is to the left of the  $K_P$  digit position: that is,  $K_P$  time precedes  $K'_{DC}$  time. If more than 12 digits have been entered, the  $K_P$  time follows  $K'_{DC}$  time.

The first step in the routine is then to establish the time relationship between  $K_P$  and  $K'_{DC}$  signals. This is accomplished using the LO flip-flop. A set pulse is applied to the LO flip-flop at bit time  $P_1$  of  $K'_{DC}$  time and a reset pulse is applied to the LO flip-flop at bit time  $P_0$  of  $K_P$  time. If  $K_P$  time occurs before  $K'_{DC}$  time, then the LO flip-flop remains set at the end of the numerical portion of the word cycle. In this case, the C counter is decremented and a shift right routine is entered. If  $K'_{DC}$  occurs before  $K_P$ , then the LO flip-flop (set at  $K'_{DC}$ ), remains set at bit time  $P_1$  of  $K_P$ .

Under this condition, the overflow (OF) flip-flop is set by a gate that senses that the LO flip-flop is set, that it is  $K_P$  time, and that it is not simultaneously  $K'_{DC}$  time. If  $K_P$  time and  $K'_{DC}$  coincide, then the alignment routine is terminated by a gate that senses that coincidence is occurring on the first cycle of the routine (characterized by the fact that the cycle counter is at Status IA).

In the event that one or more shifts to the right of data are required, each shift is preceded by a decrease of 1 in the C count. In this way, a fixed relationship is maintained between  $K'_{DC}$  and the last digit entered into the calculator before the alignment routine.

## SECTION II: THEORY OF OPERATION

### 2.2

#### Alignment Routine (Continued)

For example, if the digit is originally the D20 digit, then  $K'_{DC}$  time initially coincides with D20 time. After it has been established that  $K_P$  time precedes  $K'_{DC}$ , the C count is decreased by 1. This places  $K'_{DC}$  in coincidence with D19 time. The data is then shifted one place to the right, which places the last digit entered in the D19 time. The data is then shifted one place to the right which places the last digit entered in the D19 position. On each cycle, the test to determine the time relationship between the  $K_P$  and  $K'_{DC}$  signals is repeated. Since the data is shifted one place and the C count is decreased by 1 during each cycle, and since  $K_P$  initially preceded  $K'_{DC}$ , there are now only two cases which must be considered:

- (a)  $K_P$  still precedes  $K'_{DC}$
- (b)  $K_P$  and  $K'_{DC}$  are in coincidence

In Case (a), the LO flip-flop is set at the end of the numerical portion of each word cycle. In Case (b), it is reset (the set and reset pulses being applied under the same conditions as before). If the LO flip-flop is set, the C count is decreased and another word cycle is entered. If the LO flip-flop is reset, the routine is terminated at the end of the numerical portion of the word cycle.

The shift right function is obtained by setting the shift right function storage flip-flop. If, at  $O_D$  time of the first word cycle of the alignment routine, the routine has not been terminated, then the cycle counter advances to the IIA status. The shift right (SR) flip-flop is then set through a gate that senses the coincidence of the alignment routine and Phase IIA. When the shift right routine is selected as a separate function, it is a single-cycle function. However, with the cycle counter at IIA, the routine is repeated until the required coincidence between  $K_P$  and  $K'_{DC}$  is sensed.

At the time that the routine terminates one shift right has been accomplished for each decrease of 1 in the C count. Thus, the last digit entered is in coincidence with  $K'_{DC}$  as it was initially. It follows that the  $K_{DC}$  digit defines the digit position just to the right of the decimal point. Thus, the calculator automatically aligns the next digit entered in the correct position with respect to the decimal point and the previously entered data.

## SECTION II: THEORY OF OPERATION

### 2.3

#### Transfer Routine

The transfer routine is initiated in response to actuation of the TRANSFER/FROM  TO key. The selected FROM register data are transferred into the selected TO register. The routine requires a single word cycle. FROM register data are connected from the demultiplexer output to the A counter input. The A counter is initially reset.

During digit time D0, the D0 digit of the selected FROM register word is applied to the A counter. At the end of the numerical portion of this digit time, the D0 digit data are transferred from the A counter to the R counter. The A counter is then reset. During digit time D1, D1 digit data are applied to the A counter, and the D0 digit data are read from the R counter into the selected TO register.

Figure 2-9 illustrates the transfer routine. The actuation of the TRANSFER/FROM  TO key causes the transfer flip-flop to set. This causes the cycle counter to advance to Phase IA status, and primes transfer gates to pass selected FROM data from the demultiplexer to the A counter and from the R counter to the storage register. The selected TO flip-flop primes the transfer timing gate associated with the selected TO register, so that data are recorded only in that one register.

The function control logic generates a single-cycle signal in response to the  $\overline{XF}$  signal from the transfer flip-flop. The advance of the cycle counter to IA status removes the inhibit signal from the transfer gates, and places a  $\overline{IA}$  signal on the line to the clear logic. This causes the clear logic to generate a clear signal at the end of the numerical portion of the word cycle. This resets the transfer flip-flop and terminates the routine. When the transfer flip-flop resets, an edit command is generated by the function control logic. This sets the edit flip-flop, initiating an edit routine.

### 2.4

#### Forward Space Routine

When the FWD SPACE key is actuated, the C count is decreased by 1. This moves the  $K_{DC}$  digit one place to the right. The routine is illustrated in Figure 2-10. Actuation of the key sets the FS flip-flop. This places a low level on the  $\overline{FS}$  line priming C counter input gates, which pass a 23-bit count priming level to the C counter. This causes the C counter to advance 23 counts. Since the C counter is a scale-of-24 counter, an advance of 23 counts is equivalent to a decrease of one count.

With the low level on the  $\overline{FS}$  line, the function control logic advances the cycle counter to Phase IA and provides a single-cycle signal to the clear logic. With cycle counter at IA, a low level is placed on the  $\overline{IA}$  line.

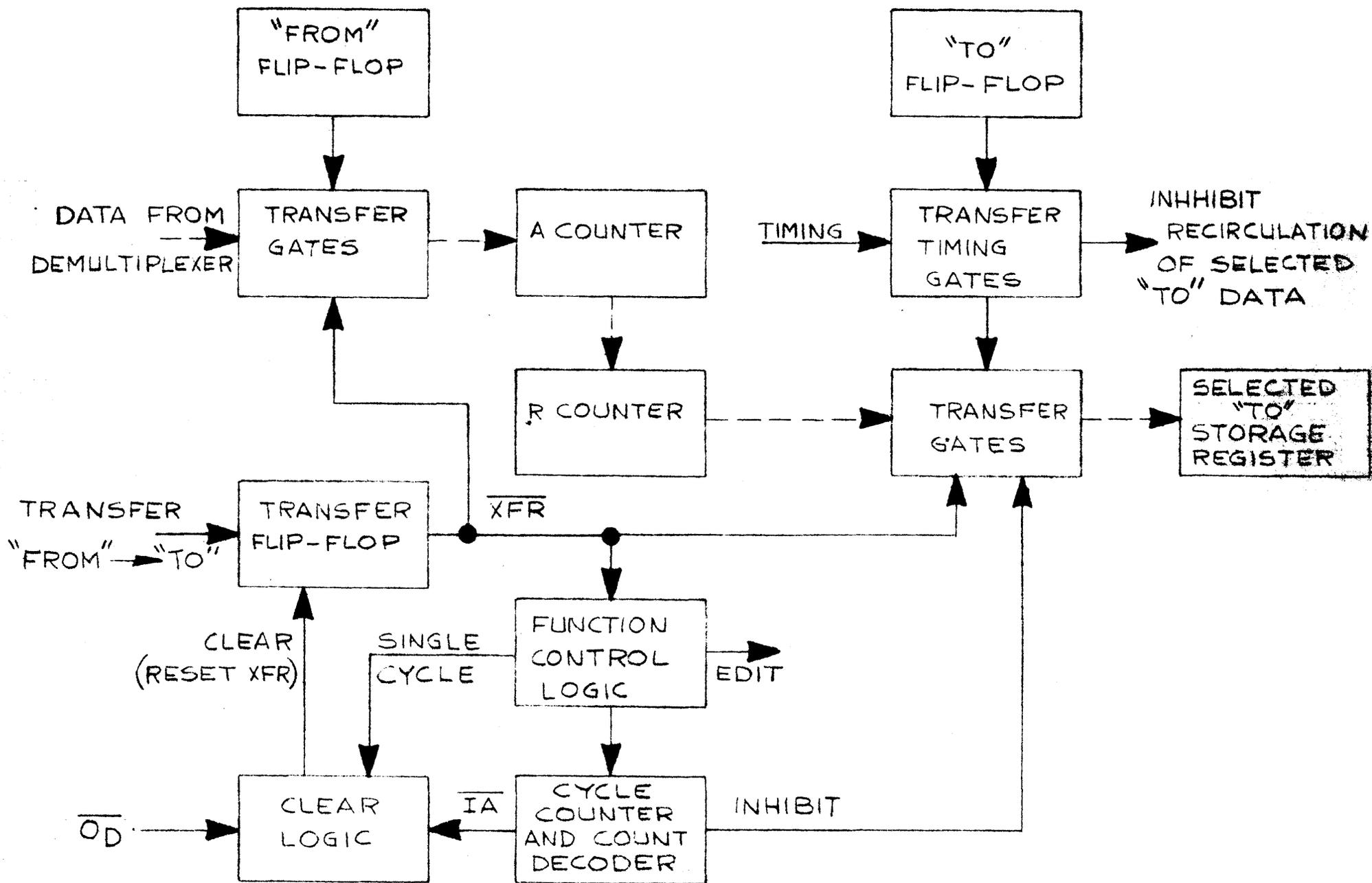


FIG. 2-9: TRANSFER ROUTINE, BLOCK DIAGRAM

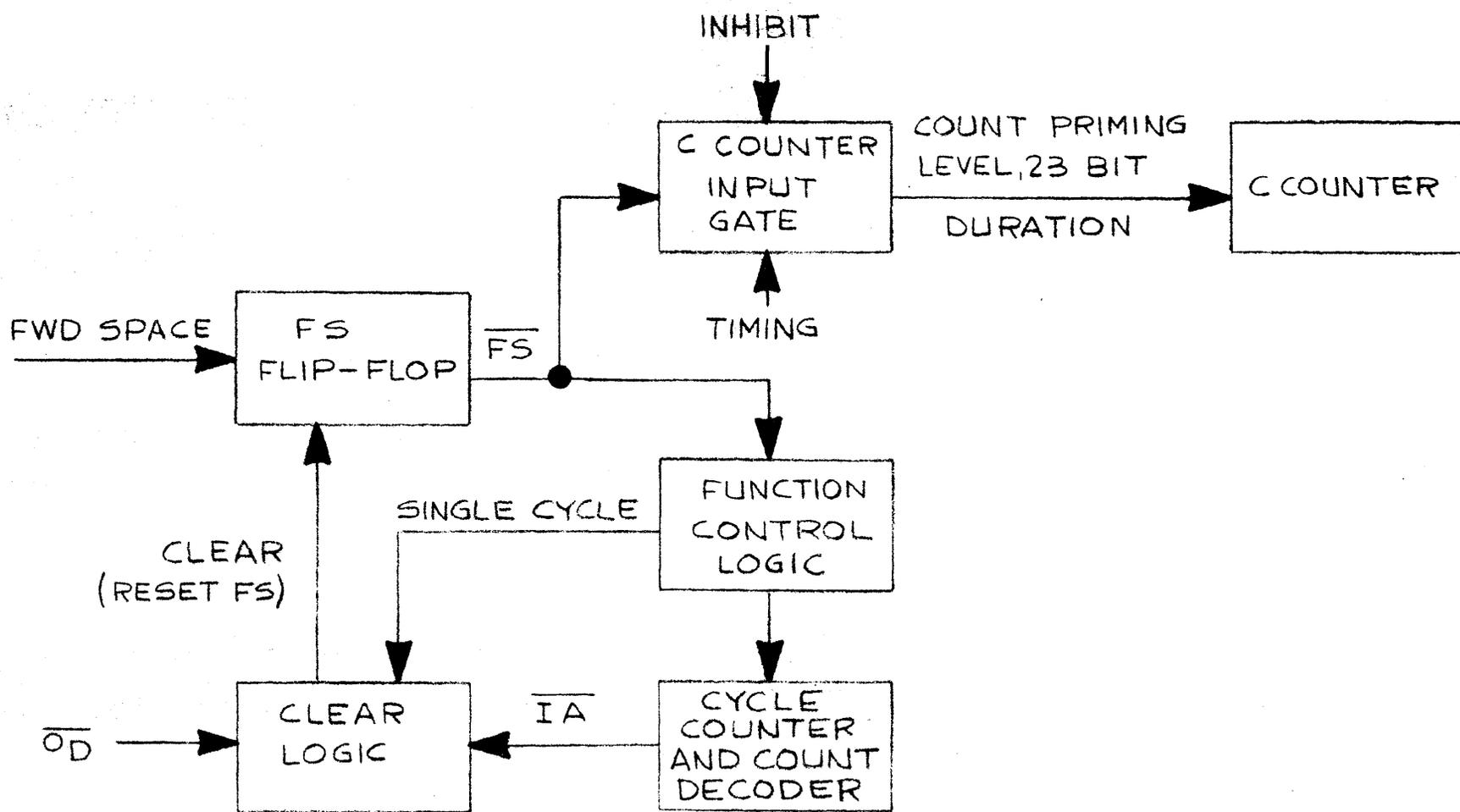


FIG. 2-10  
FORWARD SPACE ROUTINE, BLOCK DIAGRAM

## SECTION II: THEORY OF OPERATION

### 2.4 Forward Space Routine (Continued)

In response to the single-cycle and  $\overline{IA}$  inputs, the clear logic generates a clear pulse at the end of the numerical portion of the word cycle. This resets the FS flip-flop and terminates the routine.

### 2.5 Back Space Routine

When the BACK SPACE key is actuated, the C count is increased by 1. This moves the  $K_{DC}$  digit one place to the left. The routine, which is illustrated in Figure 2-11, is identical to the FWD SPACE routine, except that the count priming level has a duration of only one bit period so that the C count is advanced by one.

### 2.6 Clear Multiplier-Quotient Register Routine

When the CLEAR MQ key is actuated, the content of the multiplier-quotient register is erased. This is accomplished by inhibiting the recirculation of multiplier-quotient data. As a part of the routine the M-Q register is selected as the TO register. The clear M-Q register routine is illustrated in Figure 2-12. Actuation of the CLEAR MQ key sets the CM flip-flop, placing a low level on the CM line. The low level on the CM line primes transfer timing gates to inhibit the recirculation of data during bM sub-bit periods.

In response to the low level on the  $\overline{CM}$  line, the function control logic supplies a set pulse to the T1 flip-flop, causing the multiplier-quotient register to be selected as the TO register. Also, the function control logic advances the cycle counter to Phase IA and supplies a single-cycle signal to the clear logic. With the cycle counter at IA, the inhibit and  $\overline{IA}$  lines are placed at the low level. The low level on the inhibit line permits the recirculation gate to be inhibited. The clear logic responds to the single-cycle and  $\overline{IA}$  inputs by generating a clear pulse after the end of the numerical portion of the word cycle. This resets the CM function storage flip-flop and terminates the routine.

### 2.7 Clear Entry Register Routine

When the CLEAR ENTRY key is actuated (and at the termination of any of the multiplication, division, and the square root extraction routines), the contents of the entry register are erased. As a part of the routine (with CLEAR ENTRY key actuated) the entry register is selected as the TO register.

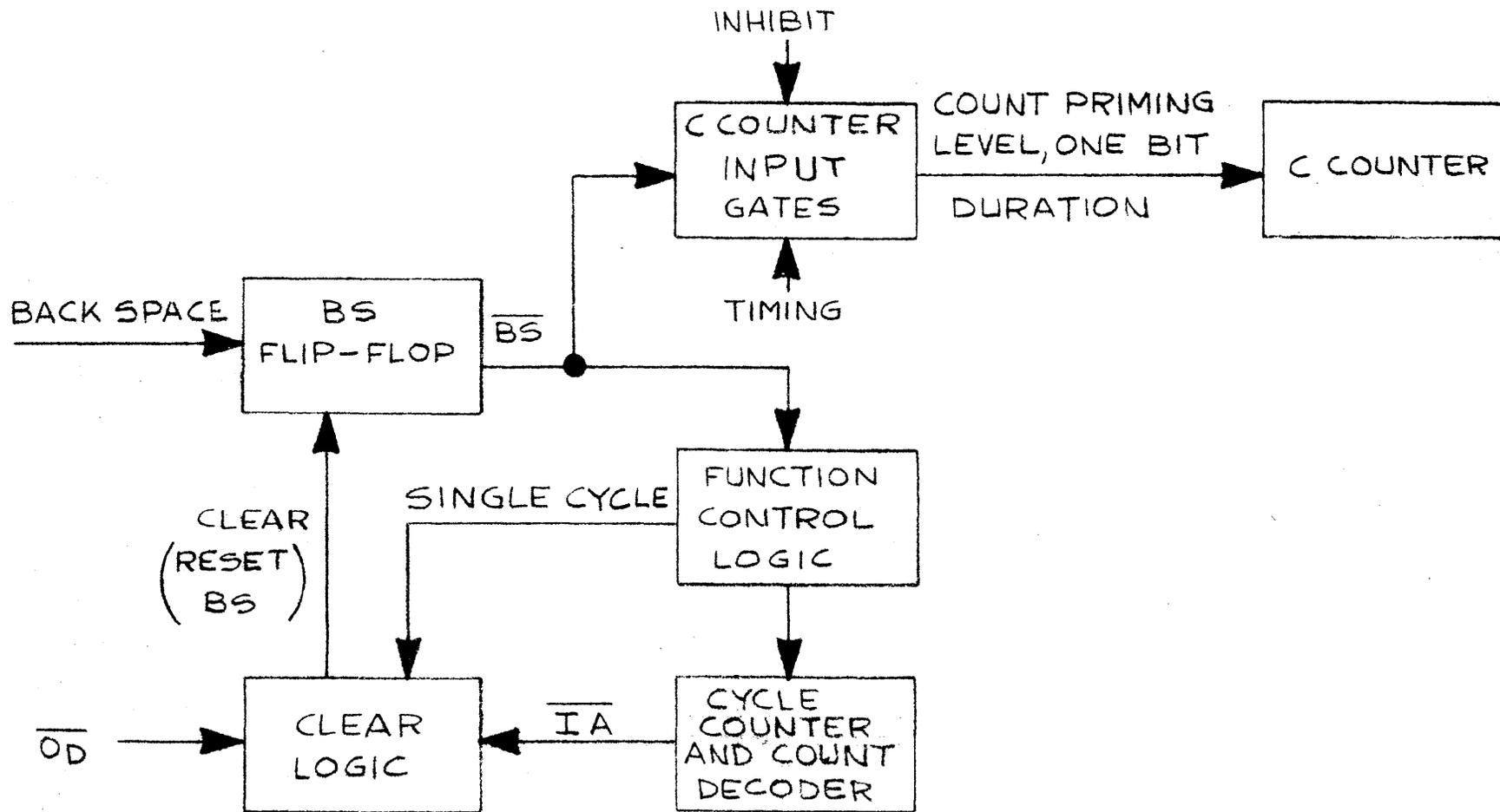


FIG. 2-11  
BACK SPACE ROUTINE BLOCK DIAGRAM

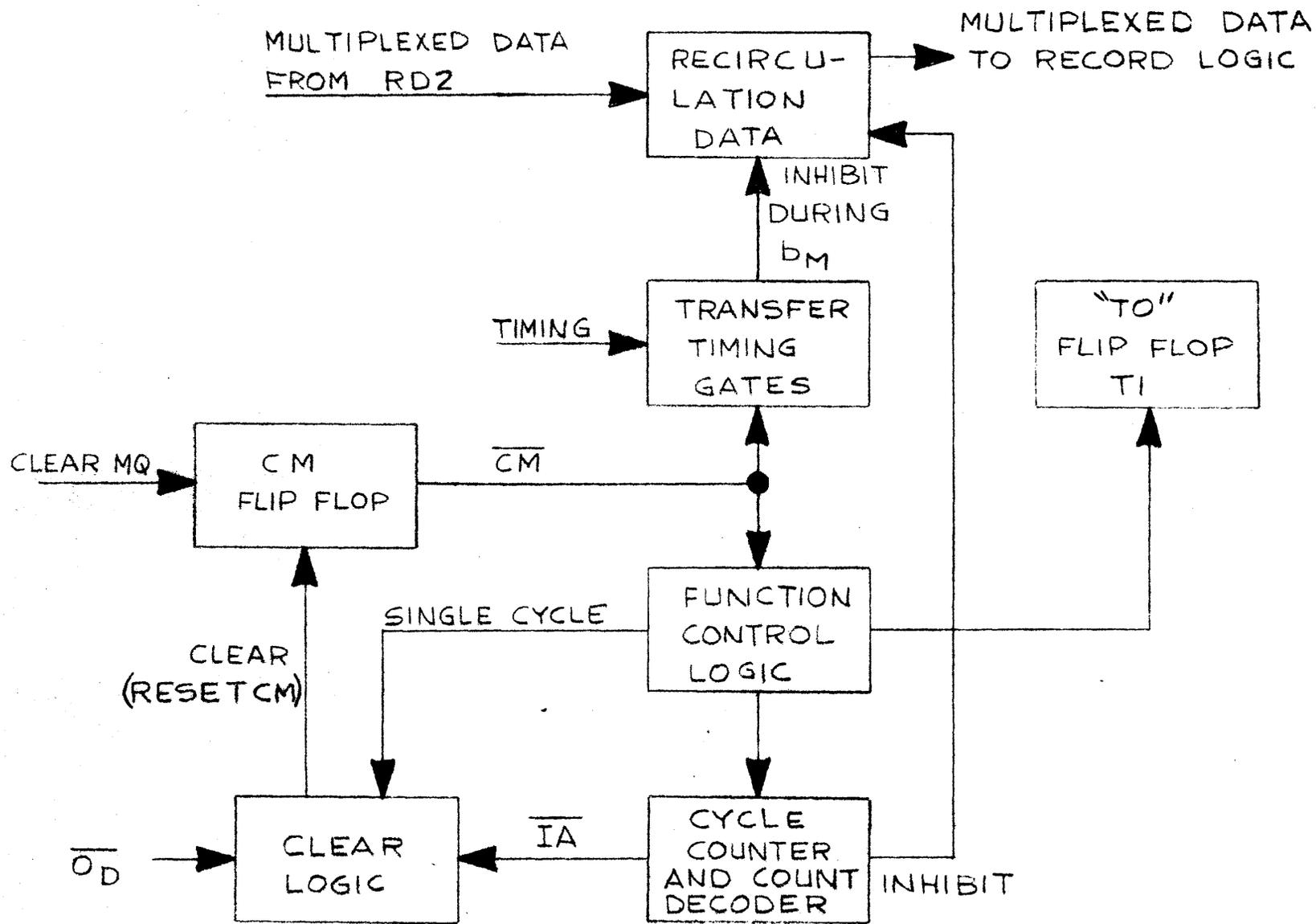


FIG. 2-12: CLEAR MULTIPLIER-QUOTIENT REGISTER ROUTINE, BLOCK DIAGRAM

## SECTION II: THEORY OF OPERATION

### 2.7 Clear Entry Register Routine (Continued)

The routine is identical to the CLEAR M-Q register routine except (a) the CE function storage flip-flop is set rather than the CM flip-flop, (b) the recirculation of data is inhibited during bE sub-bit periods (rather than during bM sub-bit periods), and (c) the T2 flip-flop is set rather than the T1 flip-flop.

### 2.8 Clear Accumulator Register Routine

When the CLEAR ACC key is actuated (or at the termination of the divide or square root extraction routines), the contents of the accumulator register are erased. As a part of the routine (when the CLEAR ACC key is actuated) the accumulator register is selected as the TO register. The routine is identical to the clear M-Q register routine, except (a) the CA function storage flip-flop is set rather than the CM function storage flip-flop, (b) the recirculation of data is inhibited during bA sub-bit periods (rather than during bM sub-bit periods), and (c) the T3 flip-flop is set rather than the T1 flip-flop.

### 2.9 Shift Left Routine

When the SHIFT  key is actuated, the contents of the selected TO register are shifted left. The routine is illustrated in Figure 2-13. Actuation of the key sets the SL flip-flop, placing a low level on the  $\overline{SL}$  line. The low level on the  $\overline{SL}$  line primes transfer gates to pass data from the RD3 tap to the storage registers. The selected TO flip-flop primes the timing gate associated with the selected TO register, so that data is recorded from the RD3 line only in that one register. The low level on the  $\overline{SL}$  line causes the function control logic to advance the cycle counter to IA, and to generate a single-cycle signal to the clear logic.

When the cycle counter advances to IA, the inhibit signal is removed from the transfer gates, allowing the required transfer of data. With the single-cycle and IA inputs applied to it, the clear logic generates a clear pulse at the end of the numerical portion of the word cycle. This resets the SL flip-flop, terminating the shift left routine. When the SL flip-flop resets, an edit command is generated by the function control logic. This sets the edit flip-flop, initiating an edit routine.

### 2.10 Shift Right Routine

When the SHIFT  key is actuated, the contents of the selected TO register are shifted to the right. The routine is similar to the shift left routine illustrated in Figure 2-13. The difference is that the SR flip-flop is set rather than the SL flip-flop, with the result that data are transferred into the selected "TO" register from the RD1 read logic rather than from the RD3 read logic. Thus, data for each digit are shifted ahead one digit time, implementing the shift right.

## SECTION II: THEORY OF OPERATION

### 2.11

#### Addition Routine

When the ADD + key is actuated (with the ADD ANY REG switch in the down position), the contents of the entry register are added to the contents of the accumulator and the entry register is cleared. The addition routine is illustrated in Figure 2-14. When the ADD + key is actuated, the  $\overline{AD}$  flip-flop sets, placing a low level on the  $\overline{AD}$  line. With the  $\overline{AD}$  line at the low level, entry register data from the demultiplexer are applied through the AC input gates to the A counter, and accumulator data are applied through the  $AC'$  input gates. During digit time DO, the DO digits of the entry and accumulator register data are summed by counting the bits of each into the counter, which is initially reset to 0. Since the A counter is a scale-of-10 counter, a carry is generated if the sum of the two digits is greater than 9. This carry is used to set the FC flip-flop. At the end of digit time DO, the contents of the A counter (equal to the units digit of the sum of the DO digits from the entry and accumulator registers) are transferred into the R counter and then the A counter is reset.

During digit time D1, the bits of the D1 digits of the entry register and accumulator register data are counted into the A counter. If a carry was generated during DO time, one extra count is entered into the A counter at the start of D1 time, and the FC flip-flop is reset.

During D1 time, the sum developed during DO time is transferred in serial format from the R counter to the accumulator register. The transfer gates required to transfer data from the R counter to the storage delay line and the transfer timing gate, which limits this transfer to  $bA$  sub-bit times so as to address the accumulator register, are primed by the low level on the  $\overline{AD}$  line. The low level on the  $\overline{AD}$  line also primes the transfer timing gate, which inhibits the recirculation of entry register data. Thus, the entry register is cleared as the addition progresses.

At the end of D1 time, the contents of the A counter, which now represent the units digit of the sum of the D1 digits and the carry (if any) from the DO order, are transferred to the R counter and the A counter is again reset. This process continues until all 24 numerical digits have been processed.

Start of the addition routine is synchronized with the word cycle by the cycle counter and function control logic. The low level on the  $\overline{AD}$  line applied to the function control logic causes the cycle counter to be advanced to Phase  $\overline{IA}$  at  $O_D$  time. This removes the inhibit signal, allowing the transfer of data from the R counter to the accumulator register, and the clearing of the entry register. The removal of the inhibit signal is also required to operate the A counter and R counter, which are clocked by pulses gated by the low level on inhibit line.

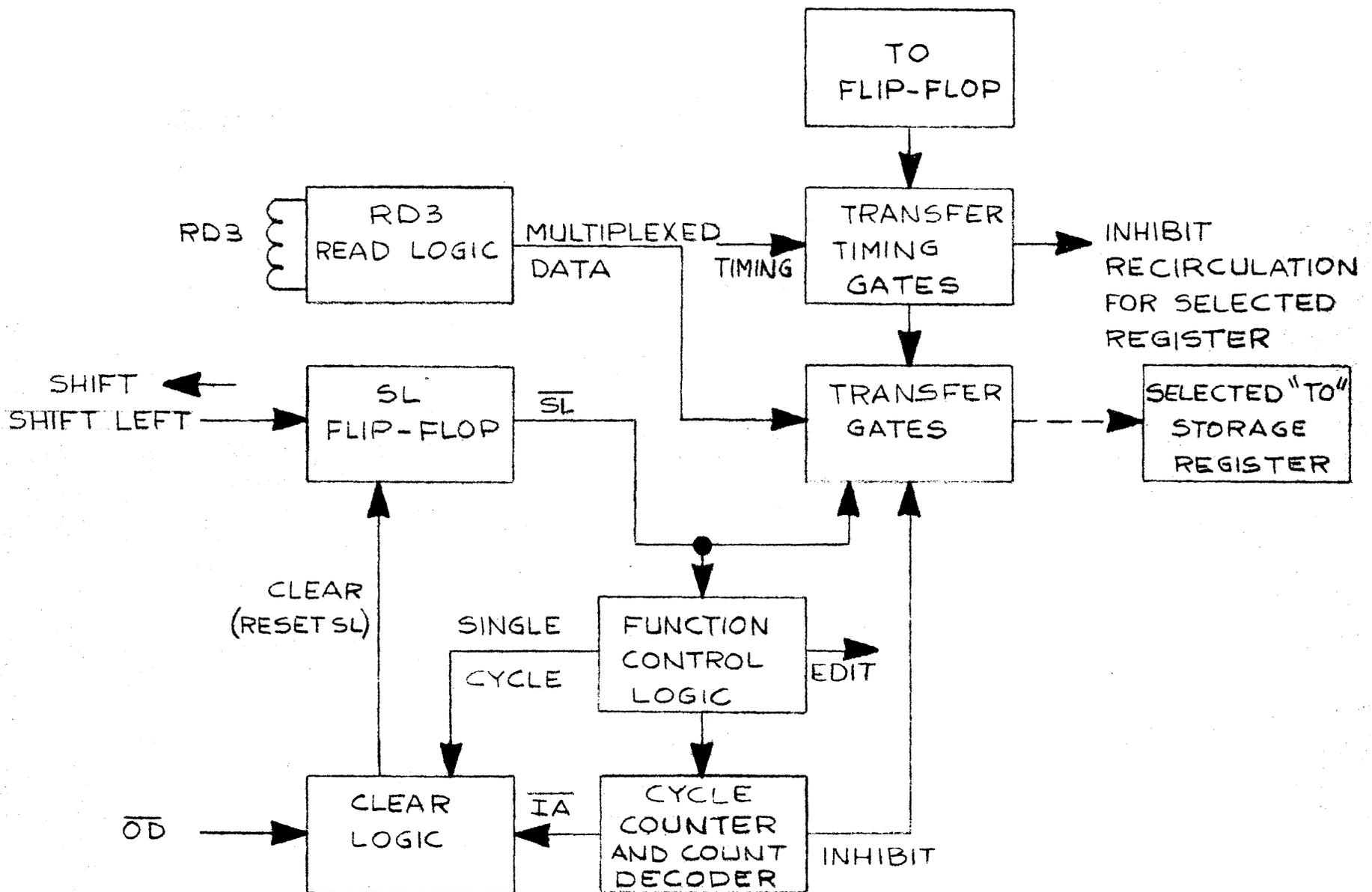


FIG. 2-13: SHIFT LEFT ROUTINE, BLOCK DIAGRAM

## SECTION II: THEORY OF OPERATION

### 2.11 Addition Routine (Continued)

In response to the low level on the  $\overline{AD}$  line, the function control logic supplies a single-cycle signal to the clear logic. In response to this signal and the low level on the  $\overline{IA}$  line from the cycle count decoder, the clear logic generates a clear pulse at  $0_D$  time of the word cycle. This resets the AD function storage flip-flop, and thus terminates the routine. As the AD flip-flop resets, switching the  $\overline{AD}$  line to the high level, the function control logic generates a set edit signal which initiates an edit routine.

### 2.12 Subtraction Routine

When the SUB - key is actuated (and the ADD ANY switch is in the down position), the contents of the entry register are subtracted from the contents of the accumulator, and the difference is stored in the accumulator.

Subtraction is performed by adding 1 plus the nine's complement of the entry register to the contents of the accumulator. As explained below, this produces the true value of a positive difference and the complement of a negative difference. When the difference is negative, no special indication is given by the calculator. However, the negative number in nine's complement value is likely to produce an overflow if it is added to a positive number in some later step. In such event, an overflow indication is obtained.

The subtraction routine is quite similar to the addition routine illustrated in Figure 2-14. Entry register data of opposite polarity from that used in addition are gated through the AC input gates to the A counter in order to obtain the nine's complement. Since a carry is obtained from the D23 order for a positive difference, the overflow indication is not generated in the case of subtraction. The 1 which must be added to the nine's complement is obtained by setting the FC flip-flop at the start of the routine so that a "carry" into the D0 order is obtained. These minor differences are obtained as a result of the low level on the  $\overline{SB}$  line. In other respects, a low level on the  $\overline{SB}$  line produces the same results as the low level existing on the  $\overline{AD}$  line during the addition routine.

The justification of subtraction by addition of nine's complements plus 1 follows from the definitions of these terms and the characteristics of the calculator data format. The complement of any number,  $a$ , with respect to  $S$  is defined as

$$\overline{a} = S - a \quad (3-1)$$

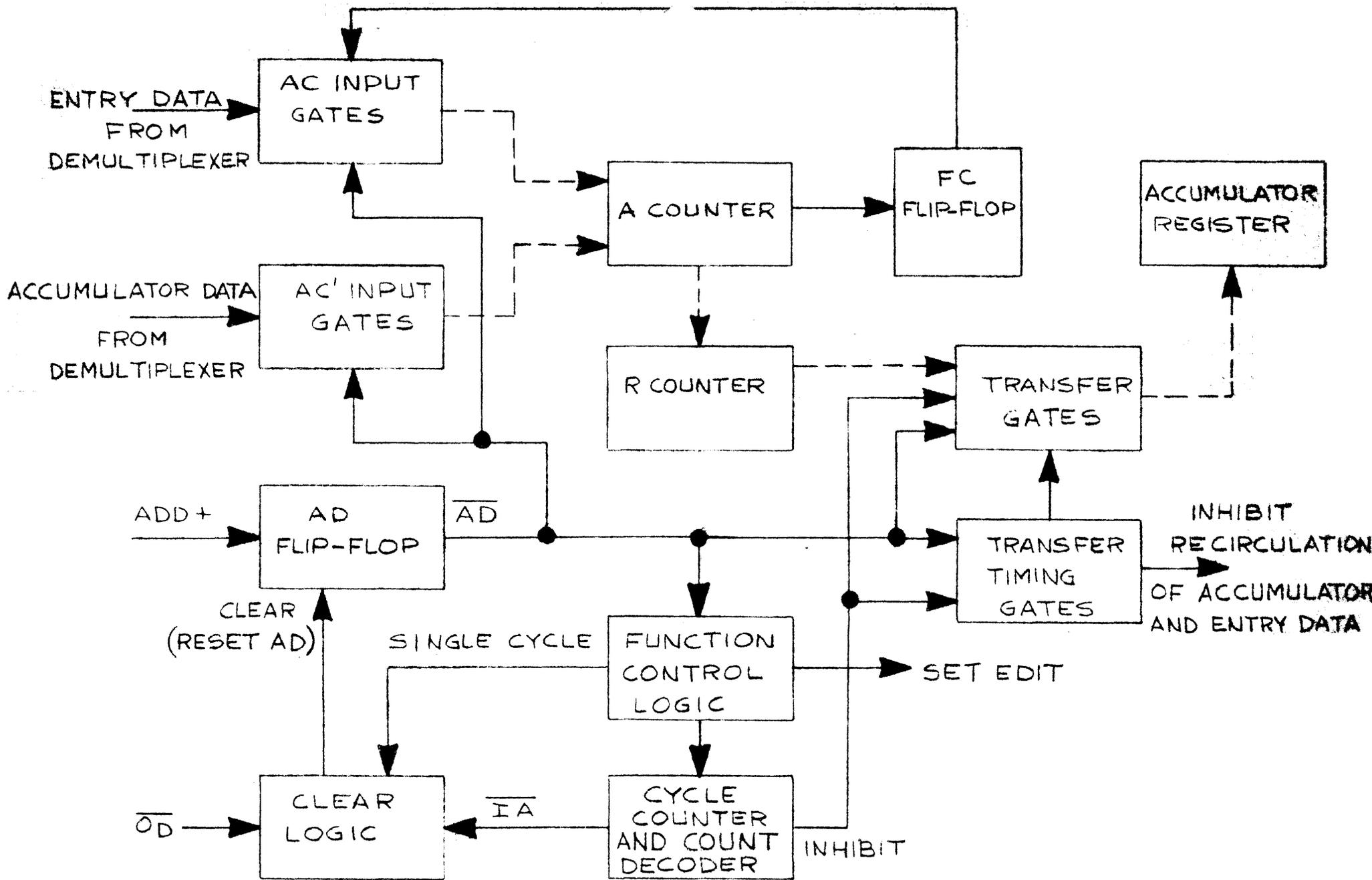


FIG. 2-14 ADDITION ROUTINE, BLOCK DIAGRAM

## SECTION II: THEORY OF OPERATION

2.12

### Subtraction Routine (Continued)

It is required to form the difference

$$D = b - a \quad (3-2)$$

This can be accomplished by adding  $\bar{a}$  to  $b$  if  $S$  is chosen equal to the number of states that can be represented by the device used to implement the calculation. Let the explicit operation performed be,

$$D = b + \bar{a} \quad (3-3)$$

Substituting the right side of equation 3-1 into equation 3-3,

$$D = b + S - a = S + b - a \quad (3-4)$$

Two cases must be considered.

#### Case 1. $b \geq a$

In this case,  $b - a$  is positive and  $D$  is larger than  $S$ , but  $S$  is chosen equal to the scale of the device. Thus, the device can represent states zero through  $S - 1$ . For any result larger than  $S - 1$  (and less than  $2S$ ) an amount  $S$  is lost by overflow. Thus, the portion of  $D$  that is retained,  $D'$  is

$$D' = D - S = b - a \quad (3-5)$$

Consider the simple case of a scale-of-100 counter. Such a counter consists of two decades which can accumulate any count from 00 through 99. To subtract 70 from 85, proceed as follows:

First, obtain the complement of 70 with respect to 100;

$$\bar{70} = 100 - 70 = 30$$

Next, add this complement to 85;

$$D = 85 + 30 = (1) 15$$

Here, the bracket around the 1 in the hundred's order indicates that it is lost in the form of a carry from the 10's order decade to the non-existent 100's order decade.

## SECTION II: THEORY OF OPERATION

### 2.12 Subtraction Routine (Continued)

#### Case 2. $b < a$

In this case,  $b - a$  is negative and  $D$  is less than  $S$ . Thus,

$$D = S + b - a = S - (a - b)$$

is obtained.

This is the complement of the negative difference resulting when  $a$  is subtracted from  $b$ . Consider the subtraction of 88 from 70 in the scale-of-100 counter.

$$\overline{88} = 100 - 88 = 12$$

$$D = 70 + 12 = 82$$

This is the complement of  $88 - 70 = 18$ ; that is,

$$\overline{18} = 82$$

To summarize: if the complement of the subtrahend is added to the true value of the minuend, then a positive difference is obtained in true value form, and a negative difference is obtained in complement form.

The formation of the complement by an explicit implementation of equation 3-1 itself involves a subtraction. Therefore, the indirect method of forming the nine's complement and adding 1 to it is used. The nine's complement is formed on a digit-by-digit basis according to the expression

$$\overline{d} = 9 - d \quad (3-6)$$

This provides an immediate simplification in that there are no carry terms generated in the formation of the nine's complement. Further, the serial decimal format used to represent numbers in the calculator lends itself to the formation of nine's complements by a simple inversion of the polarity of the data in the nine bit positions used to represent the digit values 0 through 9. Since nine data slots (bit periods P1 through P9) are used to represent the digits 0 through 9, it follows that a reversal of the polarity of the data in each slot yields the nine's complement of the data. For example, decimal 0 is represented by 0 bits in all nine data slots while 9 is represented by 1's in all nine slots. From 3-6 the nine's complement of 0 is 9, and the nine's complement of 9 is 0. Thus, to complement either value it is merely necessary to invert the polarity of the bit data. By the same reasoning, it can be shown that any inversion of the bit data produces the nine's complement of the initial data.

## SECTION II: THEORY OF OPERATION

### 2.12 Subtraction Routine (Continued)

For any decimal device, the nine's complement is equal to the complement with respect to the scale of the device minus 1. For example, in the scale-of-100 device, the nine's complement is formed in accordance with equation 3-6 as,

$$\bar{d} = 99 - d \quad (S - 1) - d \quad (3-7)$$

It follows from equation 3-7 that the complement with respect to S can be formed by first forming the nine's complement and then adding a 1 to it.

## SECTION II: THEORY OF OPERATION

2.13

### Multiplication Routine

Multiplication is performed by means of a shift and add routine. The multiplicand, held in the entry register, is shifted left (multiplied by 10) a number of times equal to the ten's order of the most significant non-zero digit of the multiplier. The shifted multiplicand is then added to the contents of the accumulator a number of times equal to the value of the multiplier digit. For example, let the multiplicand be 351 and the multiplier be 422. The most significant digit of the multiplier is then  $4 \times 10^2$ . Accordingly, the multiplicand is shifted two places to the left, yielding 35100. The shifted multiplicand is then added to the contents of the accumulator 4 times, corresponding to the value of the most significant multiplier digit. Assuming for the moment that the accumulator content is initially 0, the following partial products is obtained:  $35100 + 35100 + 35100 + 35100 = 140400$ .

The next step in the multiplication routine is to shift the multiplicand one place to the right. The net shift of the multiplicand with respect to its initial position is now equal to the ten's order ( $10^1$ ) of the second-most significant digit of the multiplier. The shifted multiplicand is now added to the contents of the accumulator 2 times corresponding to the value of the second-most significant digit of the multiplier. The partial product resulting from this second series of additions is  $140400 + 3510 + 3510 = 147420$ .

The next step in the multiplication routine is to shift the multiplicand one place to the right again. The net shift of the multiplicand with respect to its initial position is now equal to the ten's order ( $10^0$ ) of the third-most significant digit of the multiplier. The shifted multiplicand is now added to the contents of the accumulator 2 times corresponding to the value of the third-most significant digit. The product resulting from this third series of additions is  $147420 + 351 + 351 = 148122$ . This is a complete product in terms of the significant digits given. However, in the calculator the process is continued until the lowest order digit of the multiplier (the DO digit) has been processed.

There are three versions of the multiplication routine. When the CLEAR & MULT key is actuated, the accumulator is initially cleared and then the product is formed as described above. When the MULT + key is actuated, the product is formed as described above without any initial clearing of the accumulator. The resultant held in the accumulator is  $a + xy$  where  $a$  is the initial content of the accumulator, and  $x$  and  $y$  are the multiplier and multiplicand entered respectively into the multiplier-quotient and entry registers.

## SECTION II: THEORY OF OPERATION

2.13

### Multiplication Routine (Continued)

When the MULT - key is actuated, the partial products are subtracted from the initial contents of the accumulator so that the resultant is  $a - xy$ . In this case, subtraction is performed by addition of 1 + the nine's complement of the shifted multiplicand. This process corresponds to that employed in the subtraction routine.

The various steps in the routine are implemented under the control of the function storage flip-flop and the cycle counter. Figure 2-15 illustrates the multiply and add routine in terms of the status of the cycle counter. Phases IA, IIA, and IIIA accomplish the required shift of the multiplicand and set the C counter so that  $K_{DC}$  time coincides with the highest order non-zero digit of the multiplier. During Phase IA, the C counter is set so that  $K_{DC} = K_P$ ; that is, so that the  $K_{DC}$  digit is the units digit. As a byproduct of this sub-routine, the LO flip-flop (LO) is set. This status of the LO flip-flop is made use of during Phase IIIB. The cycle counter is advanced from Phase IA to Phase IIA at the end of a single-word cycle. In Phase IIA, a test is made to determine whether the units ( $K_{DC}$ ) digit of the multiplier is the highest order non-zero digit of the multiplier. If it is, the cycle count is advanced to Phase IIIB. If it is not, the cycle count is advanced to Phase IIIA. In Phase IIIA, the C count is increased by 1 and the multiplicand is shifted to the left. The cycle count is then reduced to IIA. Cycling between IIA and IIIA continues until the  $K_{DC}$  digit of the multiplier is the highest order non-zero digit. Since the multiplicand is shifted left each time that the C count is increased, the total shift of the multiplicand is always equal to the tens order of the  $K_{DC}$  digit.

When the  $K_{DC}$  digit is the highest order non-zero digit of the multiplier, then the cycle count is advanced to IIIB. During Phase IIIB, the  $K_{DC}$  digit of the multiplier is entered in the M counter. On the first cycle through IIIB this is the highest order non-zero digit. The multiplicand is shifted to the right if the LO flip-flop is reset. Since the LO flip-flop is set on the first cycle through IIIB, this function is inhibited on this first cycle. The C count is decreased by 1. The LO flip-flop is reset, preparing for a shift right of the multiplicand on the second and succeeding cycles. If the content of the M counter is zero, another word cycle in Phase IIIB is entered. This speeds up the processing of zero digits of the multiplier. If the C count is zero at the start of Phase IIIB then the least significant (DO) digit of the multiplier is being processed. In this case, the clear flip-flop is set in preparation for the termination of the routine. If the contents of the M counter are non-zero, then the cycle count is advanced to the Phase IVB status.

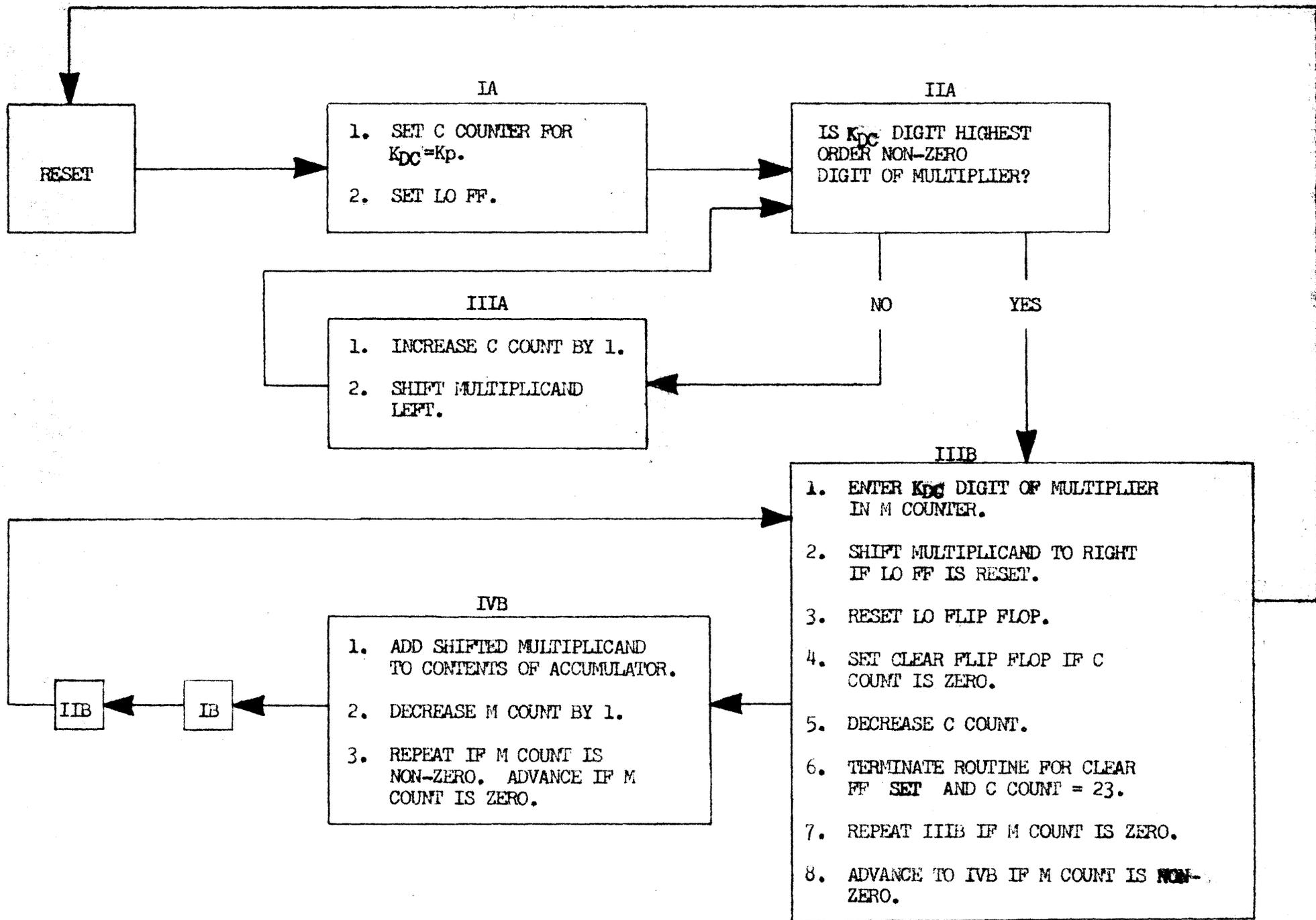


FIG. 2-15: MULTIPLY AND ADD ROUTINE, BLOCK DIAGRAM

## SECTION II: THEORY OF OPERATION

### 2.13 Multiplication Routine (Continued)

In Phase IVB, the shifted multiplicand is added to the contents of the accumulator and the M count is decreased by 1. If the M count remains non-zero, another word cycle in Phase IVB is entered. Thus, the number of additions of the shifted multiplicand equals the value of the multiplier digit entered in the M counter in Phase IIIB at the time that this repetitive subroutine is completed. When the M count has been reduced to 0, then the cycle count is stepped through IB and IIB back to Phase IIIB. Phases IB and IIB are not active steps in the multiplication routine and are stepped through in one-digit time at the end of the word cycle in IVB.

The cycling between IVB and IIIB continues until the C count of zero is sensed and the clear flip-flop is set. The word cycle continues (the DO digit of the multiplier is being processed). After the end of the numerical portion of the word cycle, the C count is again decreased as on every cycle through IIIB. However, in this case, since a scale-of-24 counter is involved, the count goes from zero to 23. The routine cycles through IVB and when it returns to IIIB, the coincidence of the C count of 23 with the set status of the clear flip-flop is used to terminate the routine.

The routine just described is the multiply and add routine performed when the MULT + key is actuated. The CLEAR & MULT routine is identical to this routine except that the accumulator is cleared during Phase IA. The MULT - routine is identical to the MULT + routine, except that each contribution of the partial product is subtracted from the contents of the accumulator rather than added to it. The subtractions are performed by addition of 1 + the nine's complement as in the subtraction routine.

### 2.14 Division Routine

Division is performed by a shift and subtract routine. Initially, the divisor is shifted so that its most significant non-zero digit is at least as significant as the most significant digit of the dividend. The shifted divisor is then subtracted from the dividend as many times as required to obtain a negative remainder. When a negative remainder is sensed, the final subtraction is cancelled by adding the shifted divisor to the remainder once. The number of subtractions performed that leave a positive remainder are counted, and this count provides a digit of the quotient.

## SECTION II: THEORY OF OPERATION

### 2.14 Division Routine (Continued)

Assume that the divisor is shifted left twice, and that it is then subtracted from the dividend three times, and that the remainder becomes negative on the third subtraction. The highest order digit of the quotient is then  $2 \times 10^2$ . In other words, it has been determined that one hundred times the divisor (the divisor shifted left twice) can be subtracted twice from the dividend and still leave a positive remainder.

After cancellation of the last subtraction by an addition of the shifted divisor, the dividend is shifted left and then the subtraction routine is being repeated. In the example where the divisor was initially shifted to the left twice, the net shift of the divisor with respect to the dividend is now 1; that is, on this second subtraction sub-routine, ten times the divisor is subtracted from the remainder during each subtraction. It might seem more natural to shift the divisor to the right and keep the position of the remainder fixed. However, if this were done, significant digits of the divisor would be lost as the routine progressed. (For example, on the final step of the routine, only one significant digit would remain.) No significant digits are lost by shifting the remainder to the left because the value of the remainder is continually reduced by the successive subtractions. The absolute position of the divisor and remainder is not important during the routine; only their relative positions.

The significance of each quotient digit that is developed is determined by the status of the C counter. The C counter is initially set so that  $K_{DC}$  is the units order digit. The C count is then increased by 1 for each shift left of the divisor, during the steps when the most significant non-zero digit of the divisor is being made at least as significant as the most significant non-zero digit of the dividend. Thereafter, the C count is decreased by 1 each time that the remainder is shifted to the left. Thus, at each step in the routine, the  $K_{DC}$  digit coincides with the power of 10 which is the current coefficient of the divisor (based on its position relative to the dividend).

In the example where the divisor is shifted left twice, the C count is advanced by 2 from the initial  $K_{DC} = K_P$  count. Thus, the  $K_{DC}$  digit is initially the hundred's order digit. On the second series of subtractions after the remainder has been shifted to the left once and the C count has been decreased by 1, the  $K_{DC}$  digit is the ten's order digit corresponding to the fact that ten times the divisor is being removed from the remainder on each subtraction.

## SECTION II: THEORY OF OPERATION

2.14

### Division Routine (Continued)

The shift and subtract routine continues until the C count reaches zero; that is, until a DO digit of the quotient has been developed.

The division routine is illustrated in Figure 2-16 in terms of the status of the cycle counter. The "A" phases are used to obtain the required initial alignment of the divisor and the correct status of the C counter. During Phase IA, the C counter is set for  $K_{DC} = K_P$ ; that is,  $K_{DC}$  digit time is set for unit's digit time. Also, the multiplier-quotient (M-Q) register is cleared in preparation for entry of the quotient digits on later steps. During Step IIA, a test is made to determine whether the highest order non-zero digit of the divisor is of at least the same order as the highest-order non-zero digit of the dividend. If not, the cycle counter advances to Phase IIIA.

During Phase IIIA, the C count is increased by 1 and the divisor is shifted left. The cycle count is then reduced by IIA. Cycling continues between IIA and IIIA until the required result is sensed in IIA. At this time, the cycle count advances to IIIB.

During this phase, the shifted divisor is subtracted from the remainder (which is the dividend on the initial cycle), and the M count (which is initially 0) is increased by 1 if the remainder is positive. If the C count is zero, then all of the quotient digits have been developed. In this case, the clear flip-flop is set in preparation for terminating the routine. When the remainder becomes negative, the cycle count is advanced to the IVB status. During IVB, the last subtraction is cancelled by one addition of the shifted divisor to the remainder. The cycle count then advances to IB.

During Phase IB, the quotient digit developed during Phase IIIB is transferred from the M counter to the  $K_{DC}$  digit slot in the multiplier-quotient (M-Q) register. The C count is decreased by 1, the content of the accumulator is shifted left, and the M counter is reset. The cycle count then returns to IIIB via IIB which is not an active step in division. This cycling between IIIB, IVB, and IB continues until a C count of zero is sensed in Phase IIIB. At this time, the clear flip-flop is set.

The word cycle is completed (the DO digit of the quotient is being developed), and the cycle counter advances once more through Phases IVB and IB. This allows the DO digit of the quotient to be stored in the M-Q register. When the scale-of-24 C counter count is decreased by 1 in Step IB, it goes from 0 to 23. The cycle counter then steps through IIB and IIIB. The coincidence of a C count of 23 and a set clear flip-flop is then sensed and is used to terminate the routine.

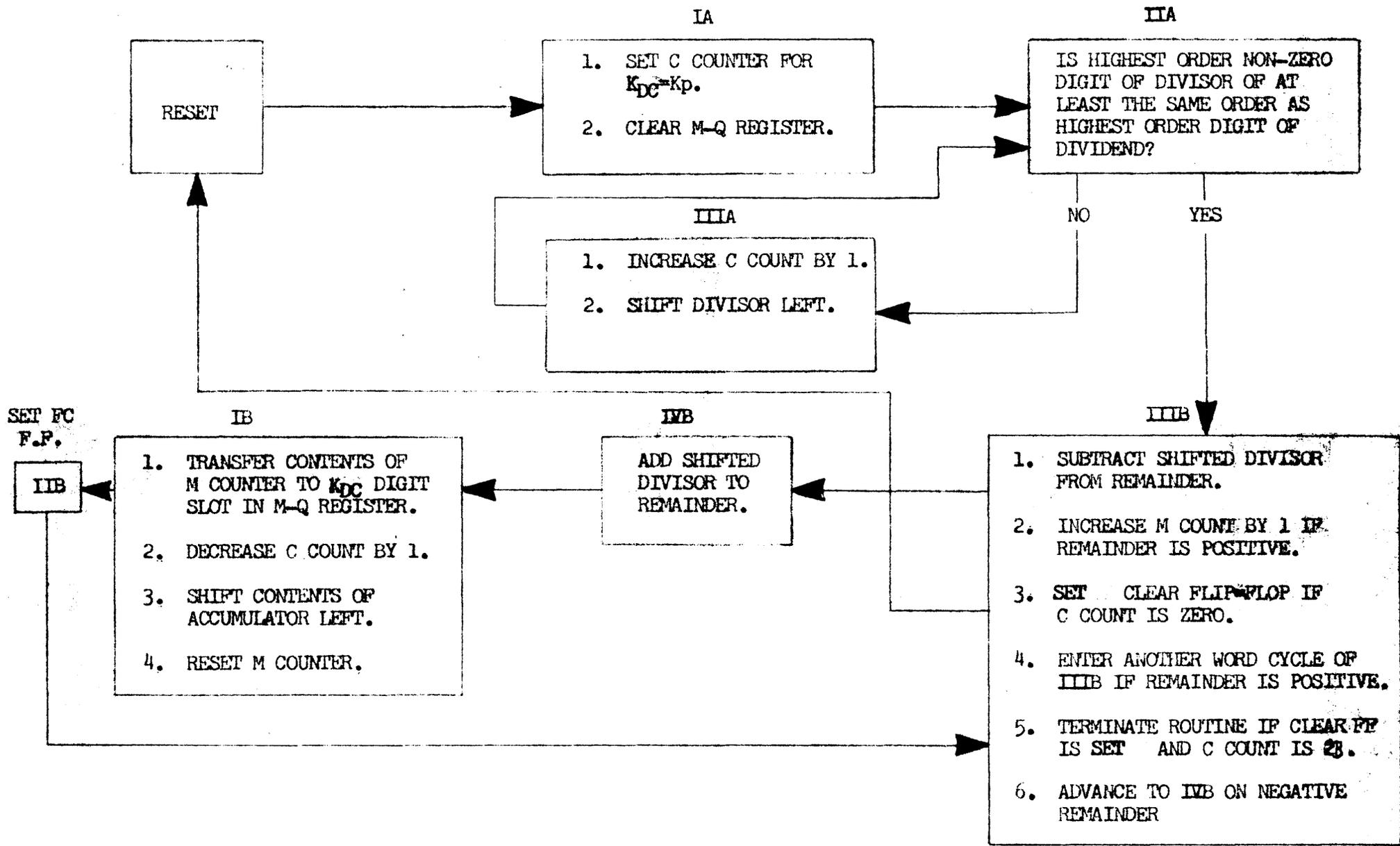


FIG. 2-16: DIVISION ROUTINE, BLOCK DIAGRAM

## SECTION II: THEORY OF OPERATION

### 2.15 Square Root Extraction Routine

As with the familiar manual routine for extraction of square roots, the routine used by the calculator is dependent upon the identity:

$$(a + b)^2 = a^2 + 2ab + b^2 \quad (3-8)$$

The routine develops the digits of the quotient one at a time, starting with the highest order digit and proceeding to the lowest. Each digit is developed by searching for the largest  $2ab + b^2$  term (which is equal to or smaller than  $N^2 - a^2$ ) where  $N^2$  is the number whose root is sought and  $a^2$  is the square of that portion of the root previously developed.

On the first step,  $a = 0$ ; therefore, the search is for the largest perfect square of a number,  $b$ , having but one non-zero decimal digit. The stipulation that  $b$  have but one non-zero digit is necessary, in order to develop the root on a digit-by-digit basis. Since the routine operates on a decimal basis, there are, of course, only nine possible non-zero digits available for  $b$ . Associated with each of these is a corresponding family of squares, where any of the squares can be described by the expression:

$$b^2 = d^2 \times 10^{2n} \quad (3-9)$$

For example, associated with the digit 5 is a family of squares: 25, 2500, 250,000, 25,000,000, et cetera.

The value of  $d^2$  can be, at most,  $9 \times 9 = 81$  and can thus occupy a maximum of two digit positions. Notice that the term  $10^{2n}$  always yields an even power of 10. Thus, the less significant digit of  $d^2$  must always be lined up with a digit of  $N$ , which is a coefficient of an even power of 10 when searching for the largest  $b^2$  satisfying the required condition.

Consider, as a simple example, the number  $N^2 = 29,242$ . For the purpose of square root extraction, this number consists of the following three components:

$10^5$	$10^4$	$10^3$	$10^2$	$10^1$	$10^0$
0	2	9	2	4	2

## SECTION II: THEORY OF OPERATION

2.15

### Square Root Extraction Routine (Continued)

In testing for the largest perfect square which meets the requirements described above, the less significant digit of  $d^2$  is lined up under the 2 in the  $10^4$  position. Since the digit of  $N^2$  in the  $10^5$  position is 0, it follows that  $d^2$  consists of a single digit. Moreover, 1 is the largest single digit that is a square of another integer and is less than or equal to 2. Thus  $d^2 = 1$  and  $2n = 4$ ; that is,  $b^2 = 10,000$ . It follows that  $d \times 10^n = 100$  is a component of the square root. The highest order digit of the root is then a 1 in the 100's order.

With one component of the square root obtained,  $a = 100$ ,  $a^2 = 10,000$ . Thus, the remainder is:

$$R = N^2 - a^2 = 29242 - 10,000 = 19,242$$

What is now required is to find the largest  $b^2$ , such that the following inequality is satisfied:

$$R \geq 2ab + b^2 \quad (3-10)$$

Or, substituting the numerical values:

$$19,242 \geq 200b + b^2 \quad (3-11)$$

The quantity,  $b$ , is some digit between 0 and 9. The largest  $b$  that satisfies the required condition can be found by successively substituting  $b = 1$ ,  $b = 2$ ,  $b = 3 \dots$  into the inequality until a  $b$  that is too large to satisfy it is found. The required  $b$  is then 1 less than this.

This is essentially what is accomplished in the square root routine used in the calculator. However, instead of substituting values of  $b$  directly in the righthand side of the equation and then making comparisons with a fixed remainder, the remainder is reduced by successive subtractions of a term,  $2a + \Delta b^2$  such that after  $b$  subtractions the total amount removed from  $R$  is always  $2ab + b^2$ . After the first subtraction,  $b=1$ . The value  $\Delta b^2 = 1$  is used in this first subtraction. The total amount subtracted is thus  $2a + 1 = 2ab + b^2$  as required. On the second subtraction  $\Delta b^2 = 3$  is used. Thus,  $2a + 3$  is subtracted. The total amount removed after the second subtraction is

$$(2a + 1) + (2a + 3) = 4a + 4 = 2ab + b^2$$

On the third subtraction,  $\Delta b^2 = 5$ ,  $b = 3$ ,  $b^2 = 9$ . Totaling the amounts subtracted during the first three subtractions,

$$(2a + 1) + (2a + 3) + (2a + 5) = 6a + 9 = 2ab + b^2$$

## SECTION II: THEORY OF OPERATION

### 2.15 Square Root Extraction Routine (Continued)

Since the first subtraction results in a negative remainder, the  $10^{-1}$  digit of the root is zero. The process continues in order to seek the  $10^{-2}$  digit of the root. Since the range of  $b$  for this digit is 0.01 to 0.09,  $2a$  is again shifted to the right one place so that  $0.02a$  is removed from the remainder on each subtraction. The range of  $b^2$  is 0.0001 to 0.0081. Thus, the less significant digit of  $\Delta b^2$  is lined up with the  $10^{-4}$  digit of the remainder. The subtractions required to develop the  $10^{-2}$  digit are as follows:

$$\begin{array}{r}
 1.0000 \\
 \underline{3.4201} \\
 -2.4201 \\
 \underline{3.4201} \\
 1.0000
 \end{array}$$

Again, the first subtraction results in a negative remainder. This establishes that the  $10^{-2}$  digit of the root is zero. The process continues in order to seek the  $10^{-3}$  digit of the root. The range of  $b$  is now 0.001 to 0.009 while the range of  $b^2$  is 0.000001 to 0.00081. Thus,  $2a$  is again shifted to the right one place so that  $0.002a$  is removed from the remainder on each subtraction and the less significant digit of  $\Delta b^2$  is lined up with the  $10^{-6}$  digit of the remainder. The subtractions required to develop the  $10^{-3}$  digit of the root are as follows:

1.000000	Subtraction
<u>.342201</u>	Number (1000b)
.657999	1
<u>342003</u>	
.315996	2
<u>342005</u>	
- 26009	
<u>+342005</u>	
.315996	

Two subtractions were accomplished leaving a positive remainder. Thus, the  $10^{-3}$  digit of the root is 2 and the root has now been determined to be 171.002 with a remainder of 0.315996. The process is continued in the calculator until DO digit of the quotient has been developed.

SECTION II: THEORY OF OPERATION

2.15 Square Root Extraction Routine (Continued)

In the same way, it can be shown that the entire  $\Delta b^2$  series for 1 through 9 subtractions is 1, 3, 5, 7, 9, 11, 13, 15, 17. The special property of this series is that, if the first  $m$  terms are added, the sum is  $m^2$ . Thus, for example, when the first four terms are added:

$$1 + 3 + 5 + 7 = 16 = 4^2$$

In the numerical example of  $N^2 = 29,242$ , the second digit of the square root is a 7 in the 100's order. This means that 71 successive subtractions would be required to obtain a negative remainder, if the term  $2a + \Delta b^2$  were to be lined up with the decimal point according to its true decimal weight. If the quantity  $2a$  is shifted one place to the left, then it becomes  $20a$ . Each subtraction of  $20a$  corresponds to 10 subtractions of  $2a$ . Since the term  $b$  is equal to the number of subtractions of  $2a$ ,  $b = 10$  after one subtraction of  $20a$ . Thus,  $b^2 = 100$ .

Similarly, after 2 subtractions of  $20a$ ,  $b = 20$ ,  $b^2 = 400$ . A shift of one place to the left of the  $2a$  term must then be accompanied by a shift to the left of 2 places of the  $\Delta b^2$  term. More generally, if the  $2a$  term is shifted  $n$  places to the left, the  $\Delta b^2$  must be shifted  $2n$  places to the left. In the numerical example,  $2a = 200$ . Thus, shifting this to the left one place,  $20a = 2,000$ . The  $\Delta b^2$  digit must be shifted left 2 places; that is, it must be placed under the 100's order digit of the remainder. Combining  $20a$  and  $100 \Delta b^2$  into a single numerical term, the successive subtractions would proceed as follows:

	1	92	42.		Subtraction
-		21	00.		Number (b/10)
	1	71	42.		1
-		23	00.		
	1	48	42.		2
-		25	00.		
	1	23	42.		3
-		27	00.		
		96	42.		4
-		29	00.		
		67	42.		5
-		31	00.		
		36	42.		6
-		33	00.		
		3	42.		7
-		35	00.		
-		31	58.		
+		35	00.		
		3	42.		

## SECTION II: THEORY OF OPERATION

### 2.15 Square Root Extraction Routine (Continued)

The foregoing series of subtractions develops the second most significant digit of the square root by establishing that  $b = 70$  is the largest  $b$  which satisfies inequality (3-10). Thus, a second component of the root has been determined and this is added to the first component. The resultant sum can then be substituted for  $a$  in inequality (3-10). The reduced remainder can be substituted for  $R$  in (3-10). The third-most significant digit must then satisfy the inequality:

$$342 \geq 2ab + b^2 = 340b + b^2 \quad (3-12)$$

Since the units digit is now being developed,  $2a + \Delta b^2$  is lined up with the decimal point according to its true decimal weight. The subtractions required to develop the units digit of the root are as follows:

	Subtraction Number (b)
3. 42	
3. 41	
. 01	1
3. 43	
- 3. 42	
3. 43	
. 01	

Thus, the third most significant digit of the square root is 1. The component thus far developed is 171 and there is at this point a remainder of 1.

The process can now be continued to determine the fractional portion of the root. Since the  $10^{-1}$  digit of the root is now being sought, the  $2a$  term is now shifted one place to the right of the position associated with its true decimal weight. In this way, each subtraction is a subtraction of  $0.2a + \Delta b^2$ . Since  $b$  is within the range of 0.1 to 0.9,  $b^2$  must be within the range of 0.01 to 0.81. Thus, the less significant digit of  $\Delta b^2$  is lined up with the  $10^{-2}$  digit of the remainder. The subtractions required to develop the  $10^{-1}$  digit are as follows:

	Subtraction Number (10b)
1. 00	
-34. 21	
-33. 21	
34. 21	
. 01	

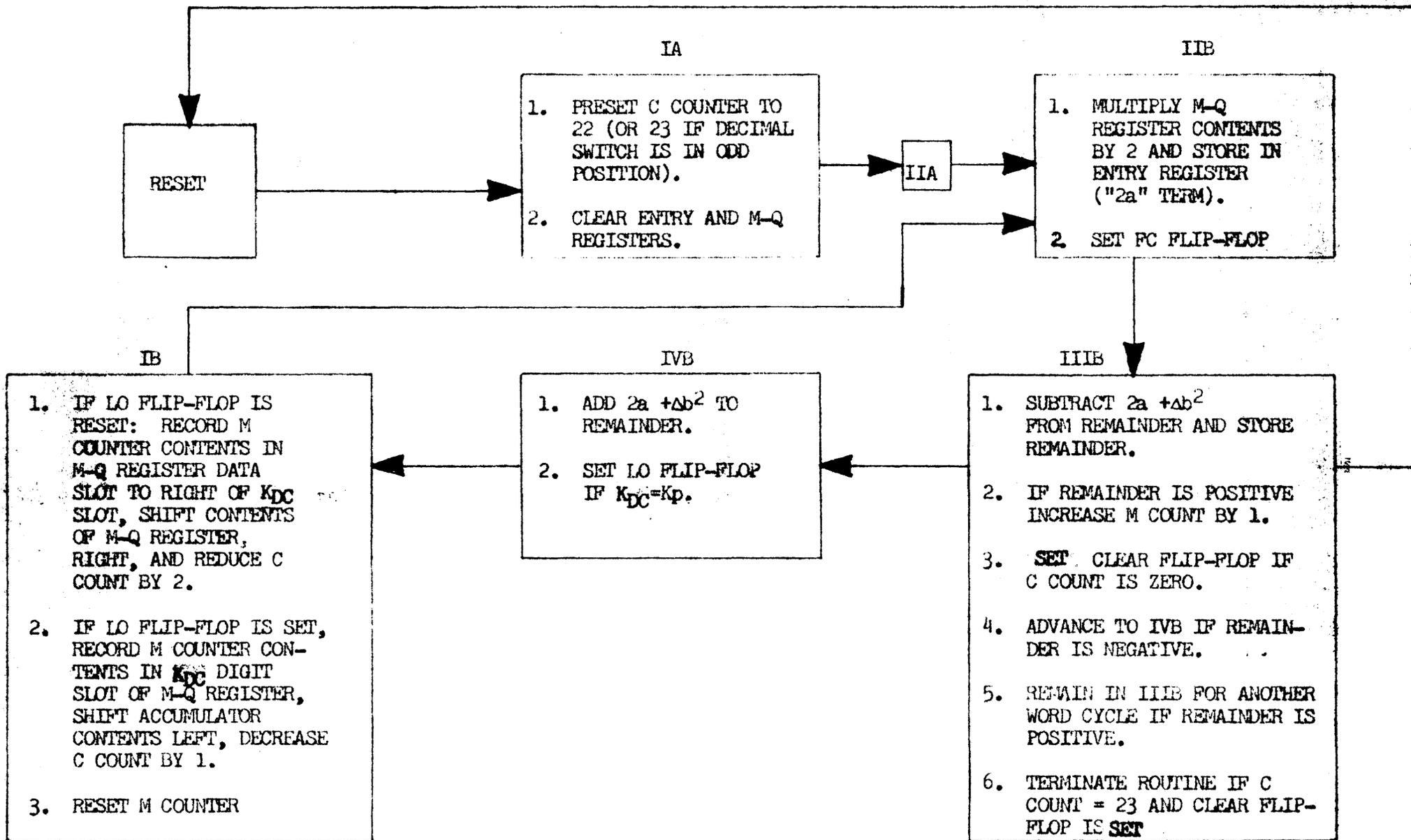


FIG. 2-17: SQUARE ROOT EXTRACTION ROUTINE, BLOCK DIAGRAM

## SECTION II: THEORY OF OPERATION

2.15

### Square Root Extraction Routine (Continued)

While the essentials of the routine performed by the calculator are as described above, one simplification has been made for purposes of discussion. In the actual calculator routine, the remainder is shifted to the left when developing fractional digits of the square root. This results in the same relative position of the remainder and the  $2a$  term as is obtained by shifting the  $2a$  term to the right. However, it prevents the loss of significant digits from the  $2a$  term and simplifies the positioning of the  $\Delta b^2$  term and of the developed quotient digit as explained below.

The square root extraction routine is illustrated in terms of the status of the cycle counter in Figure 2-17. During Phase IA, the C counter is preset to 22, if the unit's digit is an even digit or to 23, if the unit's digit is an odd digit (as determined by the position of the decimal point switch). This prepares the calculator for the development of the largest  $b^2$  satisfying equation 3-9. If the unit's digit is in an even digit position, then all  $10^{2n}$  digit positions are even digit positions. Thus, the most significant  $10^{2n}$  digit position provided by the calculator is the D22 position. If the unit's digit is in an odd position, then all  $10^{2n}$  digit positions are in odd digit positions and the most significant  $10^{2n}$  digit position provided by the calculator is the D23 position. In either case, the C counter is preset so that  $K_{DC}$  coincides with the most significant  $10^{2n}$  digit.

During Phase IA, the entry register which is used to store the "2a" term and the multiplier-quotient register (in which the quotient is stored as it is developed) are cleared.

The cycle count advances after one word cycle in Phases IA to IIB via Phase IIA. Phase IIA is not an active phase in square root extraction. During Phase IIB, the "2a" term is developed by multiplying the partial quotient (stored in the M-Q register) by 2. The result is stored in the entry register. The multiplication by 2 is actually accomplished by adding the contents of the M-Q register to itself. This, in turn, is accomplished by applying the contents of the M-Q register, through both the AC and AC' input gates to the A counter. The first cycle through IIB does not produce any significant result, since the content of the M-Q register is initially zero. After one word cycle in IIB, the cycle count is advanced to the IIIB status. During Phase IIIB, the  $2a + \Delta b^2$  term is subtracted from the remainder (held in the accumulator) and the difference is again stored in the accumulator. On the first cycle through IIIB, the contents of the accumulator are the number  $n^2$  whose root is sought. The  $\Delta b^2$  term consists of either one or two digits.

## SECTION II: THEORY OF OPERATION

2.15

### Square Root Extraction Routine (Continued)

The unit's digit is subtracted from the  $K_{DC}$  digit of the remainder while the ten's digit is subtracted from the  $K'_{DC}$  digit of the remainder. On the first cycle through Phase IIIB (assuming, that the decimal point switch is in an even position), the unit's digit is subtracted from digit D22 of the dividend. If the remainder after the subtraction is positive, then the M count is increased by 1 and another word cycle of Phase IIIB is entered. When a negative remainder is obtained, the cycle count is advanced to IVB. If a C count of zero is sensed during IIIB, the DO digit of the quotient is being developed and the clear flip-flop is set in preparation for the termination of the routine. On the next cycle through IIIB, the C count is 23 and the coincidence of  $C = 23$  with the set status of the clear flip-flop is used to terminate the routine.

During Phase IVB, the  $(2a + \Delta b^2)$  term is added to the remainder, cancelling the last subtraction in IIIB and restoring the remainder to a positive status. Also, the LO flip-flop is set if  $K_{DC}$  time coincides with  $K_P$  time. The status of the LO flip-flop determines the actions performed in Phase IB.

After one word cycle in IVB, the cycle count advances to IB. If the LO flip-flop is reset, the  $K_{DC}$  time coincides with an order associated with a positive, non-zero, even power of 10. In this case, the contents of the M counter are recorded in the digit slot to the right of the  $K_{DC}$  slot in the multiplier-quotient register and the contents of the remaining slots of this register are shifted to the right, and the C count is decreased by 2. If the LO flip-flop is set, then  $K_{DC}$  time coincides with an order associated with a zero or negative power of 10. In this case, the contents of the M counter are recorded in the  $K_{DC}$  slot in the multiplier-quotient register, the contents of the accumulator register (the remainder) are shifted left, and the C count is decreased by 1.

Consider the most significant digit (MSD) of the quotient under the assumption that the decimal point switch is set at 12. In this case, D12 is the unit's order digit and the C counter is preset to 22 during Phase IA. Thus, during the first cycle of Phase IB, the MSD is entered in digit slot D21 and the C count is decreased to 20. On the second cycle through Phase IB, the MSD is shifted to digit slot D20 and the C count is decreased to 18. On the third cycle, the MSD is shifted to the slot D19 and the C count is decreased to 16. On the fourth cycle, the MSD is shifted to slot D18 and the C count is decreased to 14. On the fifth cycle, the MSD is shifted to slot D17 and the C count is reduced to 12. Since the D12 digit is the unit's order digit, the LO flip-flop is set during Phase IVB. Thus, on the sixth and succeeding cycles of IB, there is no further shift of the MSD.

## SECTION II: THEORY OF OPERATION

2.15

### Square Root Extraction Routine (Continued)

Under the assumption that D12 is the unit's order digit, then slot D17 is the  $10^5$  order slot and D22 is the  $10^{10}$  order slot. This verifies that the MSD is correctly positioned in the multiplier-quotient register, since  $10^5$  is the square root of  $10^{10}$ .

Suppose, for example, that  $N^2 = 810,000,000,000 = 81 \times 10^{10}$ . Then  $N = 900,000 = 9 \times 10^5$ . The  $K_{DC}$  digit is initially  $10^{10}$  so that the digit 9, developed by the  $\Delta b^2$  generator, is initially recorded in the  $10^9$  digit slot. It is then shifted to the right four times so that its final position is in the  $10^5$  digit slot, as required. To generalize this relationship, if the C counter is initially preset to a digit corresponding to the 2nth order of 10, then the MSD is initially recorded in the  $(2n-1)$  order and is shifted to the right  $(n-1)$  times so that its final position is in the  $(2n-1) - (n-1) = n$  order.

It has been shown that the routine leads to the correct final position for the MSD. By the same reasoning it can be shown that each less significant integral digit of the root is placed in the correct final position. The  $K_{DC}$  signal is used not only to determine the storage slots into which the quotient digits are recorded but is also used to control the digit positions from which the  $\Delta b^2$  term is subtracted. The unit's order digit of this term is subtracted from the  $K_{DC}$  digit of the remainder while the ten's order digit is subtracted from the  $K'_{DC}$  digit of the remainder. Thus, in the first cycles of IIIB before Phase IB is entered (still assuming the decimal point switch is set at 12), the unit's order of the  $\Delta b^2$  term is subtracted from D22 and the ten's order of the  $\Delta b^2$  term is subtracted from D23 of the remainder. Each time the C count is decreased by 2, the  $\Delta b^2$  term is effectively shifted 2 places to the right, as required.

At the time that the unit's order digit is recorded in the multiplier-quotient register, the LO flip-flop is set. The unit's order digit is thus recorded in the  $K_{DC}$  slot which at this time is the unit's order slot. At this point in the routine, all integral quotients have been developed and all are stored in the correct final positions in the multiplier-quotient register.

## SECTION II: THEORY OF OPERATION

2.15

### Square Root Extraction Routine (Continued)

The C count is now decreased by 1 (rather than 2) and the contents of the accumulator are shifted left. This results in a net shift of the accumulator positions from which the  $\Delta b^2$  term is subtracted of two places to the right as in earlier steps of the routine. At the same time, it places the  $K_{DC}$  digit in coincidence with the  $10^{-1}$  digit as required to enter the next digit of the quotient in its correct final position. The routine now continues on this basis until all fractional digits of the quotient, including the D0 digit, have been developed.

During Phase IIIB when the D0 digit of the quotient is being developed, the C count is zero. This causes the clear flip-flop to be set. During the next cycle through IB, the C count is decreased 1 and goes to 23. The coincidence of  $C = 23$  with the set status of the clear flip-flop causes the routine to be terminated as previously noted.

2.16

### Edit Routine

The edit routine tags each significant digit in each storage register with a binary 1 in the  $P_{10}$  bit position. This binary 1 is used by the display logic to brighten the significant digit portion of the display. The edit routine is initiated automatically after most other routines. Exceptions are the forward space and backspace routines (which modify only the C count), the right shift routine (which preserves the data from the previous edit), and the numeral entry and alignment routines. During the numeral entry, each digit is tagged as it is entered so that no separate edit is required. Alignment merely involves right shifts which do not destroy previous edit data.

The edit routine is illustrated in Figure 2-18 in terms of the status of the cycle counter. As soon as the edit function storage flip-flop is set, the word counter is set to word 1 (W1) status and the triggering of the word counter from the digit counter is disabled. At the same time, the cycle counter is advanced to IA. During Phase IA, the C counter is set so that the  $K_{DC}$  digit is the unit's order digit ( $K_{DC}$  coincides with  $K_P$ ). The cycle count is then advanced to Phase IIA.

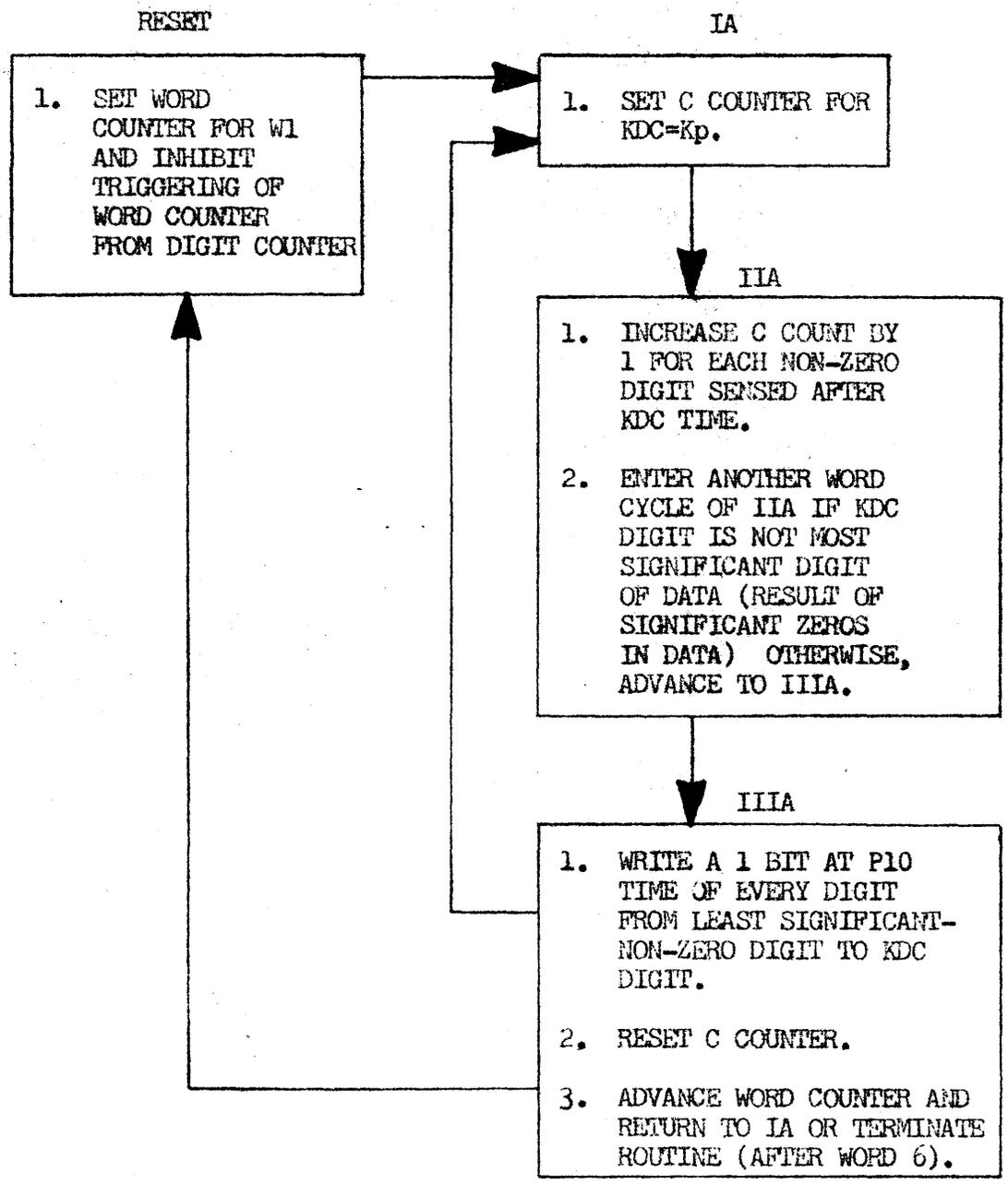


FIG. 2-18: EDIT ROUTINE, BLOCK DIAGRAM

37/2

## SECTION II: THEORY OF OPERATION

2.16

### Edit Routine (Continued)

During Phase IIA, the word under edit is monitored for non-zero digits occurring after the  $K_{DC}$  digit. The C count is advanced by 1 for each non-zero digit that is sensed after the  $K_{DC}$  digit. If at least one non-zero digit is sensed, another cycle of Phase IIA is entered. Thus, the routine continues to cycle through Phase IIA until the  $K_{DC}$  digit coincides with the highest order non-zero integral digit of the word (or with the unit's digit if all integral digits of the word are zero). When no non-zero digits are sensed after  $K_{DC}$  time, the routine advances to Phase IIIA.

During Phase IIIA, a 1 bit is recorded at P10 time of every digit from the least significant to the most significant. Three cases must be considered. If there are non-zero fractional digits, then the least significant of these is the most significant digit and the tagging of digits begins with this non-zero digit. If there are no non-zero fractional digits but there are non-zero integral digits, then the unit's digit is the least significant digit of the number, whether or not it is a zero. If all digits of the number are zeros, then there are no significant digits. The tagging starts, then, either when the first non-zero fraction digit is sensed or at the unit's digit ( $K_P$ ) time if there are no fractional non-zero digits but are integral non-zero digits. Tagging continues for all digits through the  $K_{DC}$  digit. Since, during Phase IIA, the  $K_{DC}$  digit is lined up with the most significant non-zero digit, it follows that all significant digits are tagged as required.

For the case where all digits are 0, the  $K_{DC}$  digit remains in coincidence with unit's digit time (since no non-zero digits are sensed during Phase IIA). Coincidence of  $K_{DC}$  with  $K_P$  is used to prevent the start of tagging at  $K_P$  time for this case.

At the end of the word time in Phase IIIA, the C counter is reset and the word counter is advanced by 1. The cycle count is then returned to Phase IA and the entire cycle is repeated for the next word. After the sixth word has been edited (word count goes from 6 to 1), the routine is terminated.

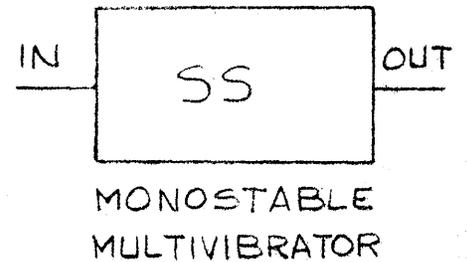
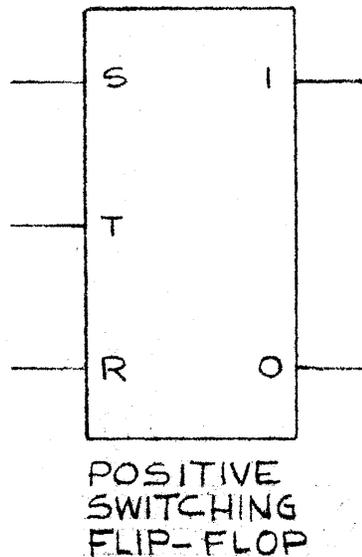
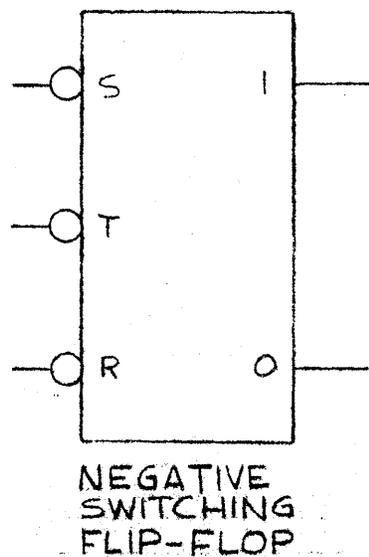
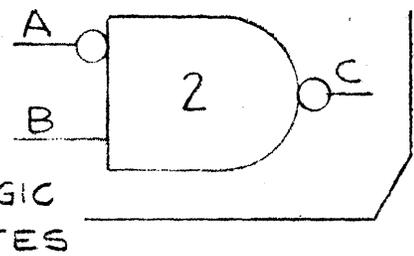
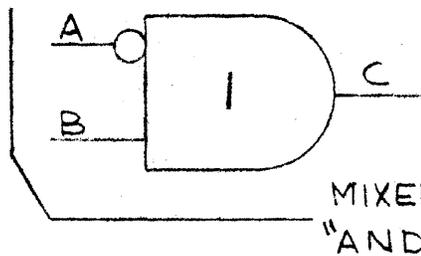
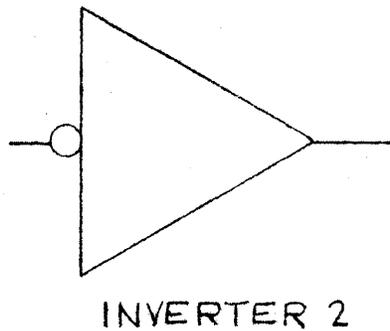
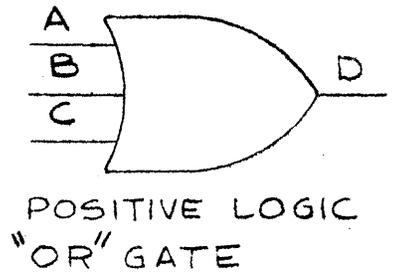
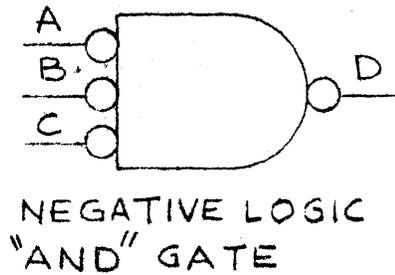
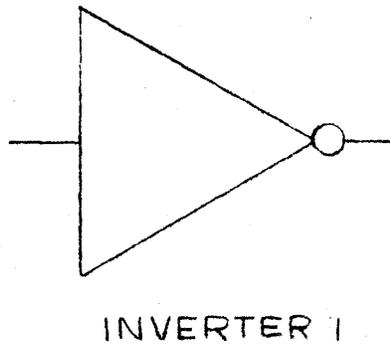
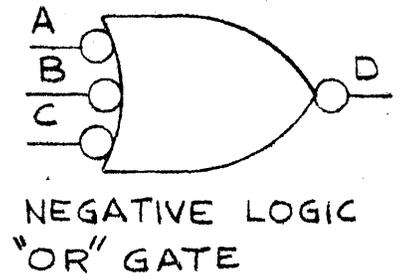
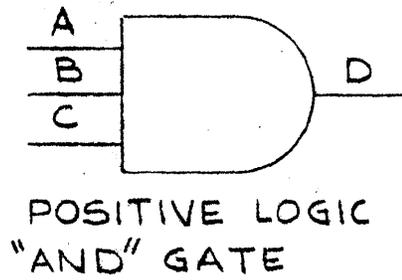
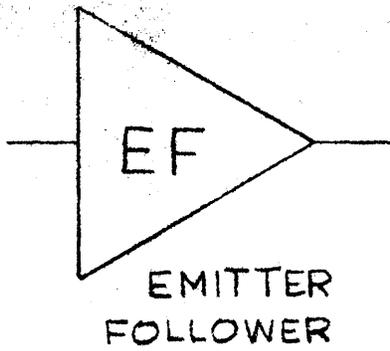


FIG. 2-19: LOGIC SYMBOLS

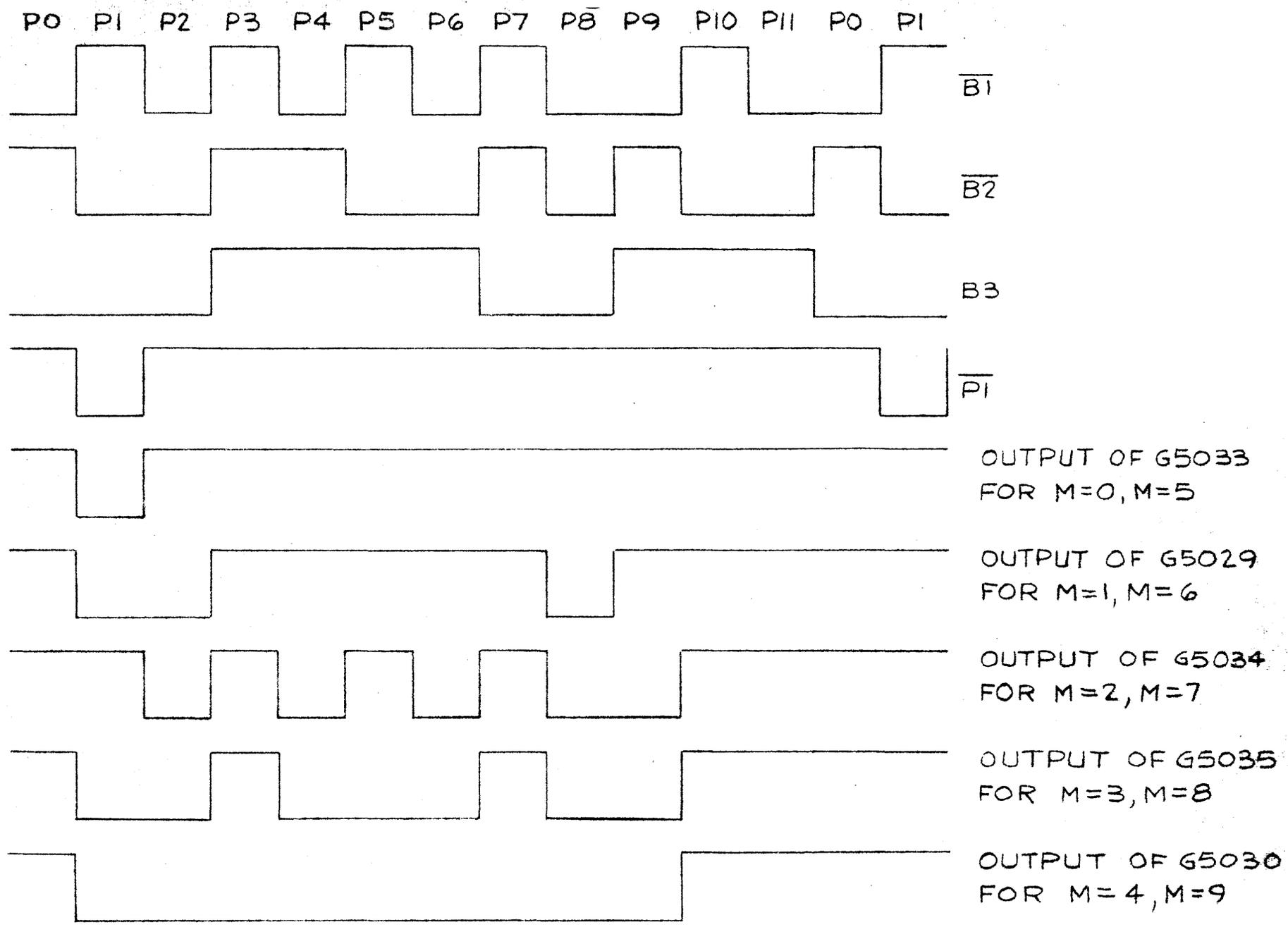


FIG. 2-20:  $\Delta b^2$  GENERATOR WAVEFORMS

CIRCUIT ANALYSIS