Univac®

LARC

COMPUTING UNIT

CONTROL SYSTEM

Remington Rand Univac®
DIVISION OF SPERRY RAND CORPORATION
UNIVAC ENGINEERING CENTER • PHILADELPHIA

Univac®

LARC COMPUTING UNIT

INSTRUCTION SEQUENCE CONTROL

—

PART 1

Remington Rand Univac®
DIVISION OF SPERRY RAND CORPORATION
UNIVAC ENGINEERING CENTER • PHILADELPHIA

RESTRICTED DISTRIBUTION

Univac®

LARC COMPUTING UNIT

INSTRUCTION SEQUENCE CONTROL

—

PART 1

AUGUST 1961

Remington Rand Univac®
DIVISION OF SPERRY RAND CORPORATION
UNIVAC ENGINEERING CENTER • PHILADELPHIA

CONTENTS

ILLUSTRATIONS

TABLES

SECTION 1
INTRODUCTION

1-1. BASIC CONCEPT OF INSTRUCTION CONTROL IN THE COMPUTING UNIT

The control unit is the coordination center for the computing unit. The principal function of the control unit is to control the acquisition and sequencing of several instructions in parallel. Parallel operation is achieved through the technique of overlapped instruction execution wherein several instructions are in various distinct phases of execution at any one time. The major logic elements within the instruction control circuits are used on a time-shared bases which imposes rigid timing considerations in the performance of control unit functions. These major logic elements include two instruction registers, two control counters, two memory address-decoders, two fast register selector registers, a fast register decoder, a program counter, an instruction decoding-encoding network, and several control flip-flops.

The term "instruction cycle" denotes the series of functions performed by the control unit to address, obtain, and decode instructions from the memory; to acquire operands from core memory and fast registers; to perform arithmetic and logical operations on the operands; and to return the results to storage. Performance of the complete series constitutes a "normal" instruction cycle and is executed automatically in the control unit for several instructions in a program sequence at the same time. The control unit will alter the cycle in accordance with the execution of certain instructions such as control transfers. Whenever a transfer of control takes place, the overlapped instruction sequencing is temporarily interrupted but is continued upon completion of transfer.

Instruction overlapping can result in various conflicts from time to time that cause delays in the sequencing. For example, an addressed memory unit may be already engaged with a previous operation, or the result of one instruction may not be available in time if it is required for use in the following instruction. Conflicts such as these, which are resolved automatically in the control unit, call for complicated logic because of the need to introduce delays or repeat certain operations.

Because instruction control operations utilize the principle of overlapped instruction execution, the emphasis throughout this book is on timing.

1-2. SCOPE OF THIS BOOK

Instruction sequence control is divided into two parts of which this book constitutes part I. A more or less general approach to the subject is taken in part I in which the overall aspects of instruction sequencing are described for the variety of conditions under which the control unit must operate. Part II is a continuation of part I and deals comprehensively with special instruction sequencing features related to store and control transfer instructions and other control operations.

The primary purpose of this book is to provide detailed information on the timing of instruction sequence control operations in the control unit. A familiarity with the essential features of the computing unit is assumed. The following three books provide additional related material on the control unit:

(1) Computing Unit—General Description

(2) Computing Unit—Control Logic

(3) Computing Unit—Instruction and Function - Signal Analysis

This book is organized to take the reader progressively through the timing of operations in the control unit from the basic instruction cycle to the condition of full overlapped sequencing. The various control features associated with the logic, the timing, and the handling of special cases in the control unit are described separately throughout the book as outlined below.

Section 2 describes the basic instruction cycle for a single instruction and then explains the overlap technique for executing a series of instructions during normal continuous operation.

Section 3 deals comprehensively with most conditions of instruction sequencing in which different aspects of control and timing are covered for the majority of instructions in the computing unit repertory.

Section 4 summarizes the ending pulse functions for all types of operation.

Section 5 presents an introductory discussion of instruction status control which is covered more fully in part II.

Section 6 classifies all the instructions in the computing unit repertory according to the control features described in sections 2 and 3. The principal inclusion in this section is a table which affords quick reference to general control features associated with any particular instruction.

Section 7 describes the logic and timing associated with starting and stopping the computing unit and with the noncontinuous and no-overlap modes of operations.

SECTION 2
CONTROL OF THE BASIC INSTRUCTION CYCLE

2-1. BASIC INSTRUCTION CYCLE

The instruction cycle of the computing unit consists of five basic operations of the control unit which are carried out by the instruction control logic. The names of the basic operations performed in all instruction cycles are:

(1) Instruction Call — addressing the memory and receiving the next instruction in a program sequence in the instruction storage register (IR1).

(2) B-Modification — performing index operation for current instruction; that is, adding the contents of the designated B register to the M address contained in the instruction to obtain the next (operand) storage reference address.

(3) Operand Select — addressing the memory to select the M operand for the current instruction. Also addressing the fast registers to obtain the A operand and supply instruction information and operands to the arithmetic unit (AU). The latter operations are conditional on the former; that is, the fast register reference and the initiation of operations in the AU depend on the successful call to the memory (memory-not-busy signal received) for the M operand. If no operand memory reference is required for the current instruction, the effect of a successful memory call is simulated.

(4) Execution — depending on the instruction, the execution phase is handled by the control unit or the AU. The unit not in use just marks time.

(5) Store Result — store the result for the current instruction.

The speed with which these basic operations are carried out for any one instruction is limited by the time required to address and read out of the memory and regenerate the information. Thus the 4-microsecond read-regenerate cycle of the main memory defines the minimum time needed to perform any of the basic operations. This basic 4-microsecond period, usually

considered as extending from time t0 through t7, is used as the principal time division in the timing diagrams in this manual and is hereinafter referred to simply as the cycle.

Each of the basic operations, except execution, normally takes one cycle to complete. The execution phase of an instruction takes one or more cycles to complete depending on the type of instruction. The total time required to process a single instruction, then, is at least 20 microseconds. However, the normal sequence of operations performed by the control unit is overlapped in time so that the five basic operations may be carried out for five different instructions during any one cycle. In this way a sequence of instructions which takes no more than one cycle of execution time in the AU can be performed at the rate of one instruction every 4 microseconds.

In order to explain the timing considerations involved in the overall operation of the control unit, a single instruction will be traced through all of the steps required to complete its execution. For this discussion, assume that the instruction is one of the common single-precision memory reference types; for example an add instruction which requires the minimum execution time of 4 microseconds. A simplified block diagram of the control unit showing only the most basic timing and function control signals is given in figure 2-1. A timing chart covering five cycles corresponding to the five basic operations of the instruction cycle is presented in figure 2-2. The chart also lists all the normal steps that are followed to complete the execution of an instruction.

2-2. INSTRUCTION CALL

To call the next instruction from memory, its address must first be established: the contents of control counter 1 (C1) are sent to the B-adder as one input, and depending on the sequencing situation, a 0 or +1 as the other input. The two are added together to form the new address. For discussion, assume that C1 already contains the address of the new instruction so that the constant to be added to C1 is a 0.

The contents of C1 enter input 1 of the B-adder at time t0 with function control signal 401 (FS401). The 0 enters input 2 with FS411 also at t0. The two values, C1 + 0, are added in the B-adder during t1 and the sum (next address) is read out to the memory address decoder (MAD) with FS363 at t2. The MAD partially decodes the new address and sends the information over the address lines to the memory at t3.

If the memory unit that is addressed by the current instruction call is not busy, a memory-not-busy control signal (MNB) will be generated in the memory unit and returned to the control unit at t4. If the MNB signal is not received at t4, delays will arise in the control unit and the instruction call will have to be repeated. Assuming no delays, the instruction will be read out from the memory during the 4-microsecond basic memory cycle and appear on the read bus from the memory during t2 of the next cycle.

At t2 of the second cycle, the instruction word is transferred to IR1 with FS320, and at t3 the word is stored in flip-flops in IR1. The divisions of IR1 (I, A, B, and M) shown in figure 2-1 accept the various parts of the instruction word. While the instruction is being transferred into

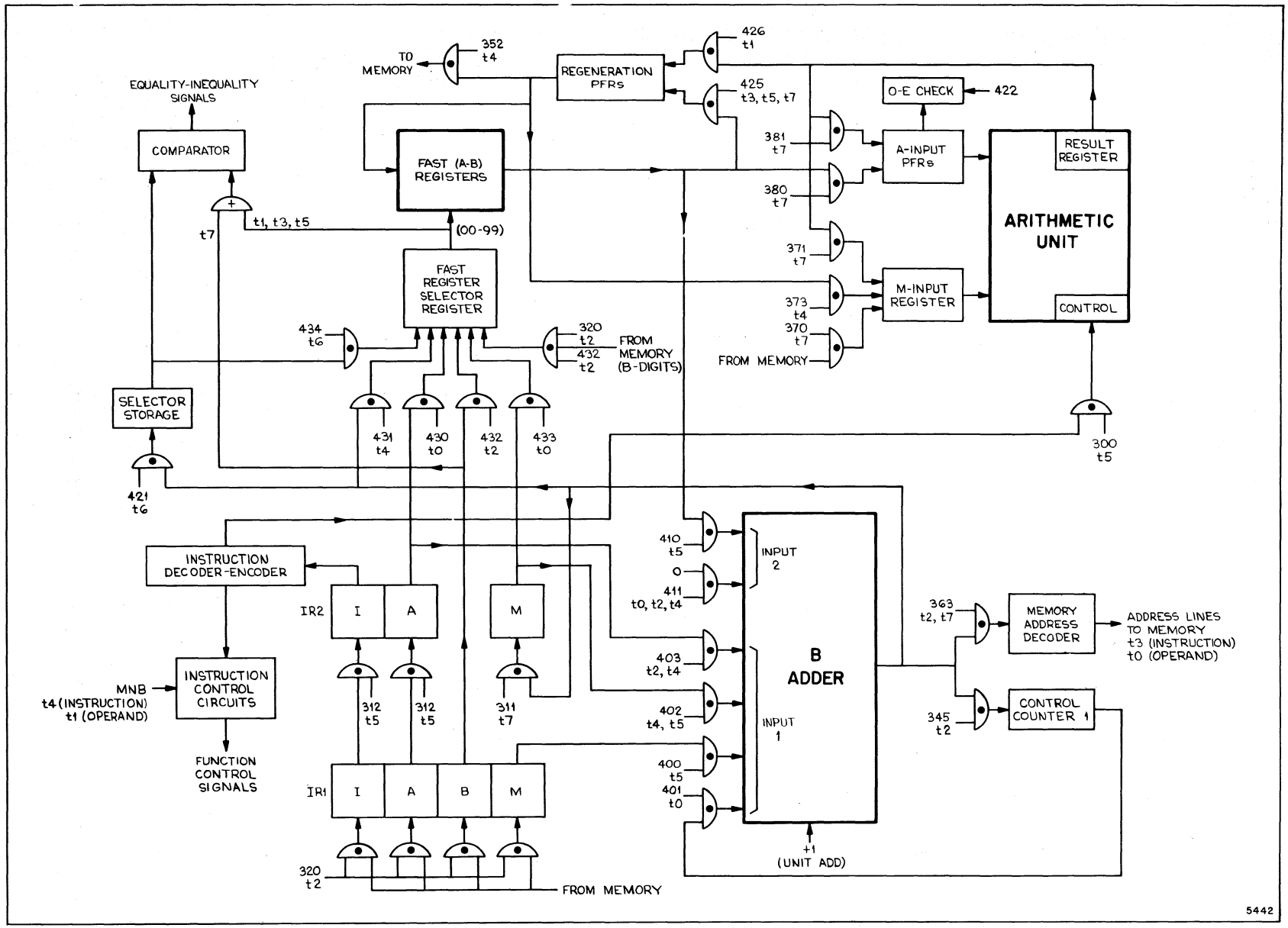


Figure 2-1. Simplified Control Block Diagram

IR1, the B part of the instruction word is transferred directly into the fast register selector register with FS432 and FS320 to start the B-modification (B-mod) portion of the instruction cycle.

2-3. B-MODIFICATION

The B register selected by the two B digits of the instruction word contains the information that will be used to index the current instruction. The designated B register is selected during t3 and t4 and the five least significant digits of the contents are sent with FS410 to input 2 of the B-adder at t5. The information in the B register is regenerated (FS425) and also sent to the A-input pulseformers of the AU for odd-even check (FS380, 422).

Input 1 to the B-adder is supplied by the M part of IR1 which is sent to the B-adder with FS400 at t5. The two inputs are added together during t6 and the sum is read out with FS363 to the MAD at t7. This new address is the modified M address which is sent across the address lines to the memory at t0 of the third cycle to select the M operand. The M address can specify a fast register in which case a memory call is not needed (reference heading 3-2).

During the B-mod operation, the I and A parts of IR1 are transferred to IR2 with FS312 at t5. At t7 the modified M address from the B-adder is also transferred to the M part of IR2 with FS311. IR2 now holds the I and A digits of the original instruction word and the modified M digits of the operand address. IR2 does not store the B digits of the instruction.

2-4. OPERAND SELECT

The decoding of the I digits is static during the time the instruction is in IR2. As shown in figure 2-2, this time extends from t6 of the second cycle through t5 of the third cycle. During this interval all necessary control signals peculiar to the instruction are generated so that specific functions may be carried out in the control unit. As mentioned under heading 2-3, the call to the memory for the M operand is made at t0 of the third cycle. If the memory unit is not busy, an MNB signal will be returned to the control unit at t1 to indicate that the selected M operand will be available on the read bus at the next t7. The control information for the AU is then transferred from a separate function encoding network at t5 with FS300.

The A operand is selected by sending the A digits of IR2 to input 1 of the B-adder at t2 with FS403. Input 2 to the B-adder at this time is a 0 (FS411) so that the sum, A + 0, is read out to the fast register selector register at t4 with FS431. The reason for this apparently devious path in transferring the A part of IR2 to the fast register selector register is that in some instructions the A digits require a modification to their original value.

The designated A register is selected during t5 and t6 and the contents are transferred into the A-input pulseformers of the arithmetic unit at t7 with FS380 and also regenerated with FS425. The M operand arrives simultaneously from the memory read bus and enters the M-input register of

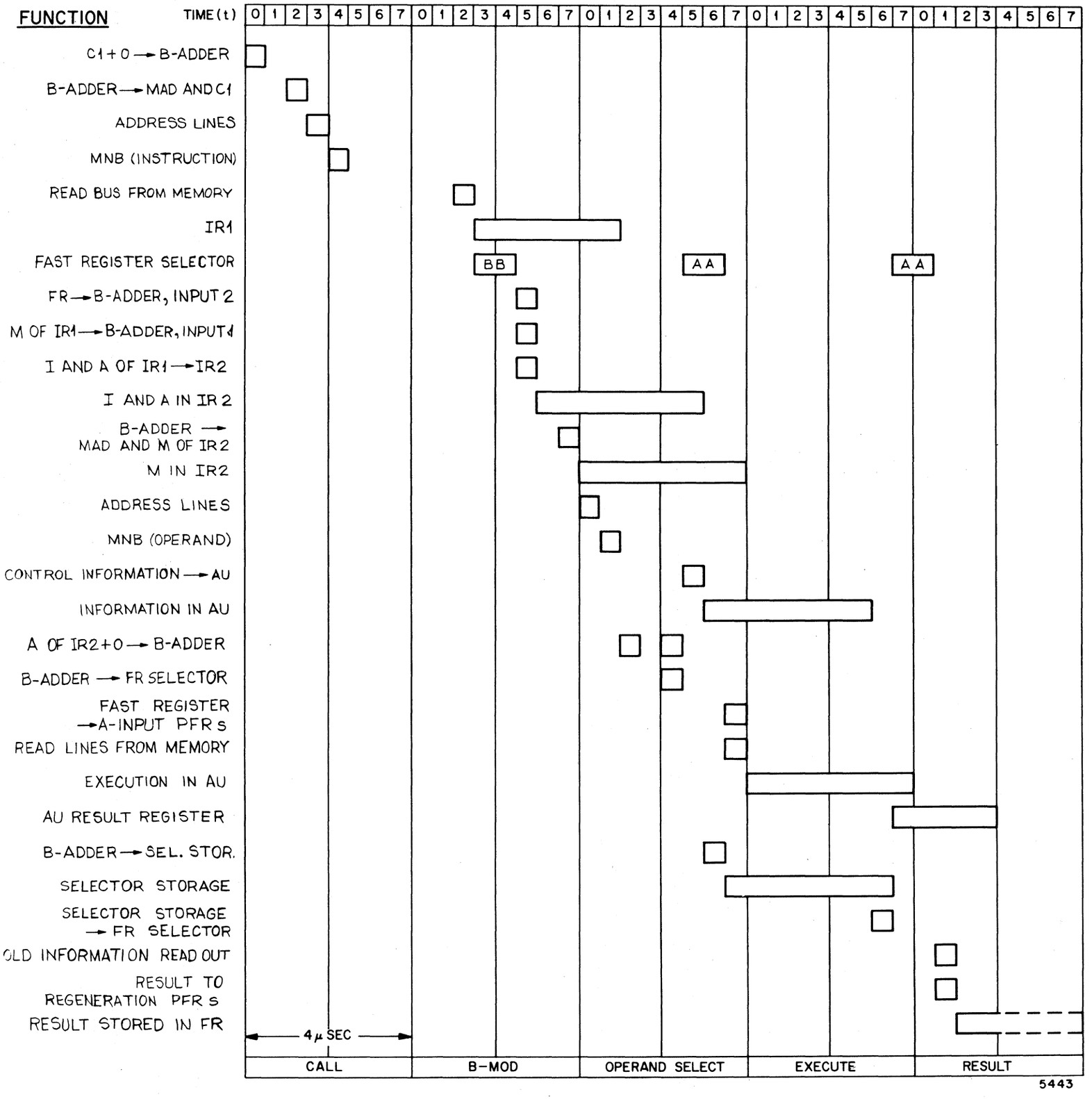


Figure 2-2. Timing Diagram - Basic Instruction Cycle

the AU with FS370. At the following t0 (fourth cycle), the beginning of the instruction execution time (in this case 4 microseconds) in the AU is started.

2-5. EXECUTION

The execution of the instruction in the AU is carried out autonomously once the AU has received operands and instruction information from the control unit. However, since IR2 is cleared before the execution takes place, it is necessary to preserve the result address for the instruction during the execution phase.

To select the correct A register to store the result, the A part of IR2, along with a 0, are again sent to the B-adder at t4 of the third cycle. The sum A + 0 is formed and then transferred to the fast register selector storage at t6 with FS421. The selector storage holds the result address for the next 4 microseconds (t7 through t6 of the fourth cycle) while the instruction is being executed in the AU. At t6 the contents of the selector storage are transferred to the fast register selector register with FS434 so as to select the A register (t7, t0) to receive the result from the AU.

2-6. STORE RESULT

The result portion of the instruction cycle takes place during the fifth cycle. The result of the operation is first stored in the result register of the AU from t7 through t3. The old information in the selected A register is read out at t1 while the contents of the result register are transferred to the fast register regeneration pulseformers with FS426. The result is then written into the selected A register at t2. This operation constitutes a clear-write cycle for the fast register which is the same as the read-regenerate cycle carried out for operand selection except that FS426 replaces FS425.

2-7. NORMAL CONTINUOUS OPERATION

The normal instruction sequence control in the computing unit operates in a continuous mode so that succeeding instructions in a sequence are overlapped in time throughout the various phases of the instruction cycle. Figure 2-3 illustrates the overlap principle by showing a series of instructions passing through the five stages of the instruction cycle. The instructions are staggered by a period of 4 microseconds. When instruction n + 4 is being called from the memory, the previous instruction n + 3 is having its M address modified, the operands for n + 2 are being selected, the arithmetic unit is executing n + 1, and the result of n is being stored. In the next cycle, n + 5 is called from memory, n + 4 is having its M address modified and so on.

As mentioned under heading 2-1, the overlapping technique results in a considerable saving in the effective instruction execution time. However, because of overlap, various conflicts in the timing of operations within the control unit can arise which will from time to time result in some

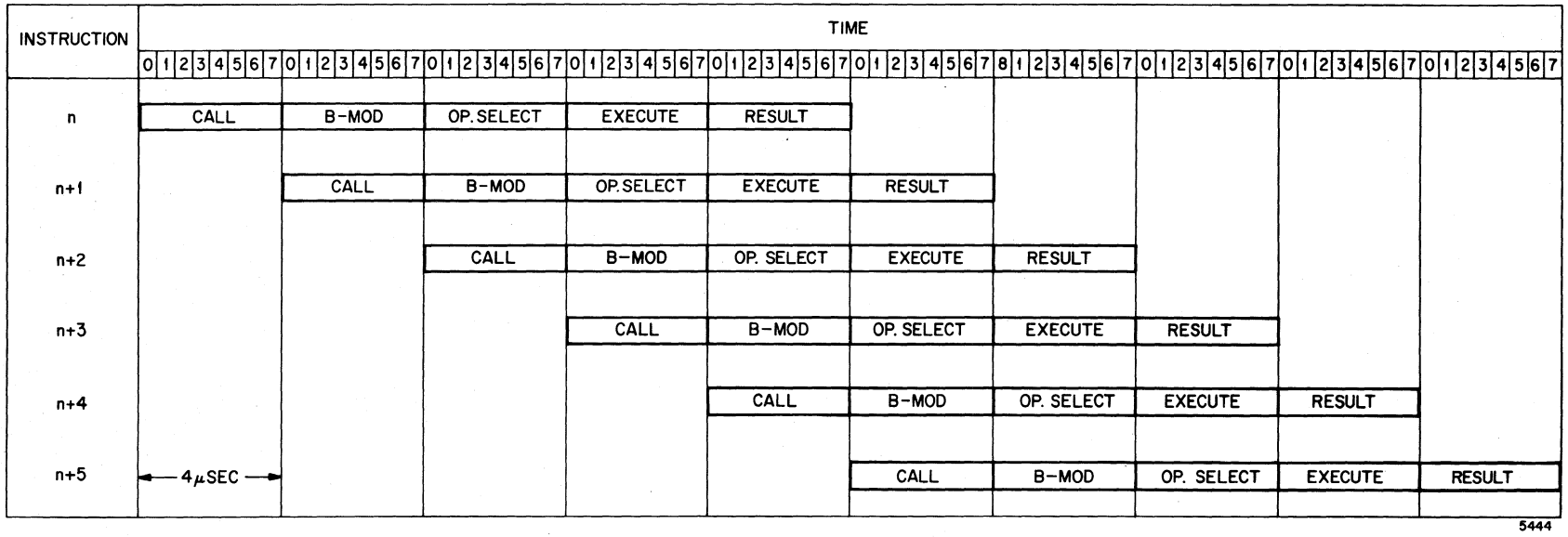
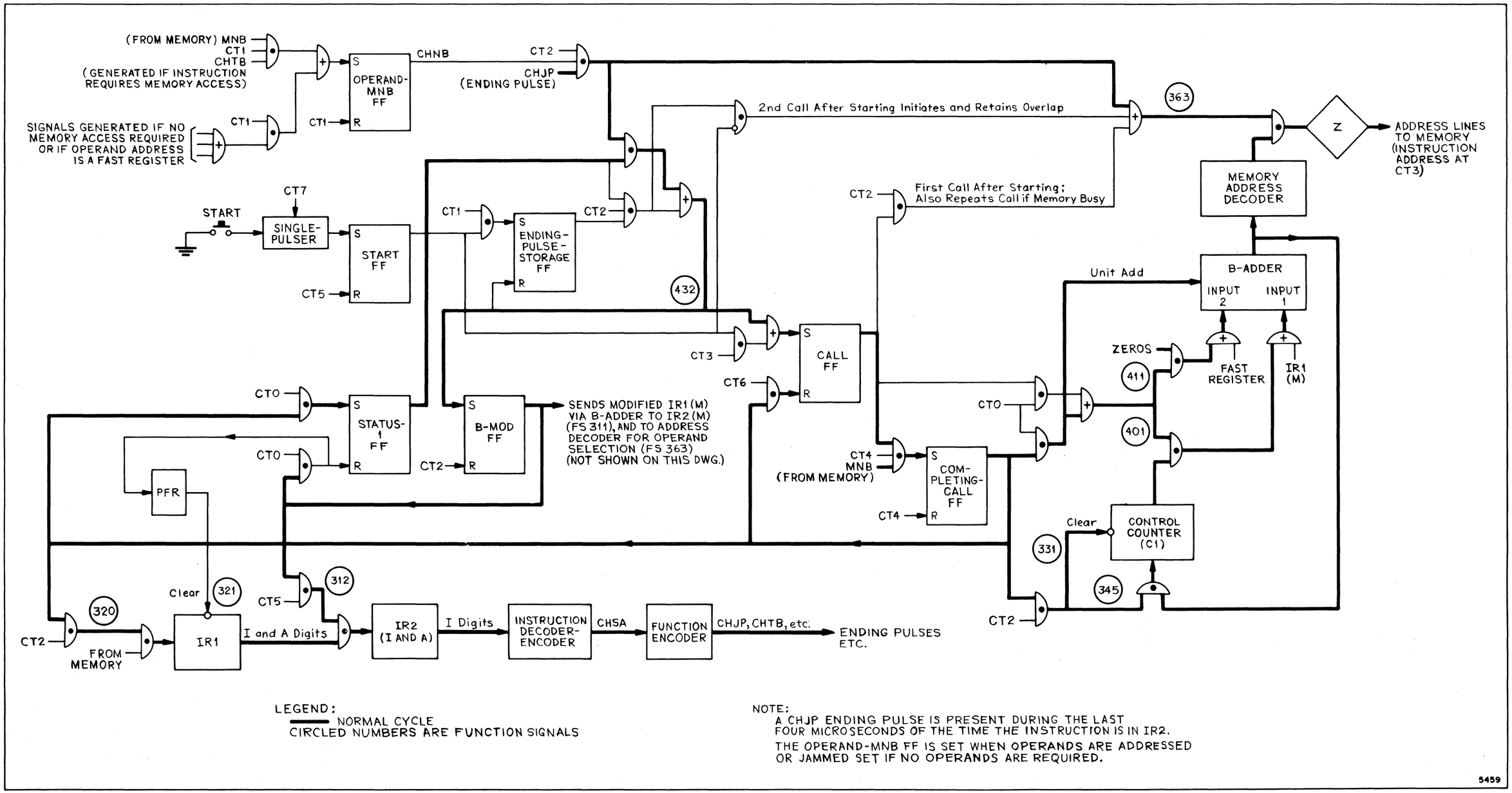


Figure 2-3. Timing Diagram - Instruction Overlapping



LEGEND:
NORMAL CYCLE
CIRCLED NUMBERS ARE FUNCTION SIGNALS

NOTE:
A CHJP ENDING PULSE IS PRESENT DURING THE LAST FOUR MICROSECONDS OF THE TIME THE INSTRUCTION IS IN IR2.
THE OPERAND-MNB FF IS SET WHEN OPERANDS ARE ADDRESSED OR JAMMED SET IF NO OPERANDS ARE REQUIRED.

5459

Figure 2-4. Simplified Logic Diagram - Instruction Request Cycle

delays. These delays will be discussed in section 3. The remainder of this section describes how overlapped instruction sequencing is controlled starting with the initial instruction request.

2-8. INITIAL INSTRUCTION REQUEST

A simplified logic diagram of the instruction request cycle is shown in figure 2-4. This figure shows the logic which controls the instruction request cycle by relating various control signals and flip-flops to the timing of overlapped instruction cycles; certain recurrent details are assumed to repeat in each instruction. An accompanying timing diagram for these operations is given in figure 2-5. Although figure 2-5 shows timing relationships for continuous operation, it is simplified in that it does not show all the operational steps of the basic instruction cycle as outlined previously in figure 2-2.

This description assumes that the computing unit is stopped and both instruction registers (IR1 and IR2) are empty. Operation is initiated by pressing the START button on one of the control consoles. The first t7 timing signal then produces an output from the single-pulser circuit to set the start FF. The output from the start FF is gated with CT1 to set the ending-pulse-storage FF, and with CT3 to set the call FF. The ending-pulse-storage FF is used to provide for certain control functions that are normally supplied by an instruction ending pulse during the last 4 microseconds of the time an instruction is in IR2. Essentially it provides a substitute ending pulse for the one usually produced by the previous instruction (reference heading 4-8). The start FF is then reset at t5 since it is no longer relevant to the control cycle.

The output from the call FF is first used to produce function control signals 401 and 411 at t0 which send the contents of the control counter (C1) and zeros to the B-adder at t0. C1 contains the address of the first instruction (n). The sum C1 + 0 is then gated from the B-adder to the memory address decoder by FS363 which is generated by gating the call FF output with CT2. Thus the first instruction call is on the memory address lines at t3. The MNB signal is generated during the next pulse time (t4) to set the completing-call FF which signifies a successful instruction call.

2-9. OVERLAPPED INSTRUCTION SEQUENCING

The setting of the completing-call FF initiates the overlapped instruction sequencing. The functions performed by the output signal from the completing-call FF are as follows:

(1) The call FF is reset for n at t6.

(2) C1 is advanced by gating C1 + 1 to the B-adder at t0 (FS401, 411, and unit add), and B-adder to C1 at t2 with FS345. (Also C1 is cleared at t2 with FS331.)

(3) The status-1 FF is set with CT0 which signifies that an instruction is (or will be) in IR1.

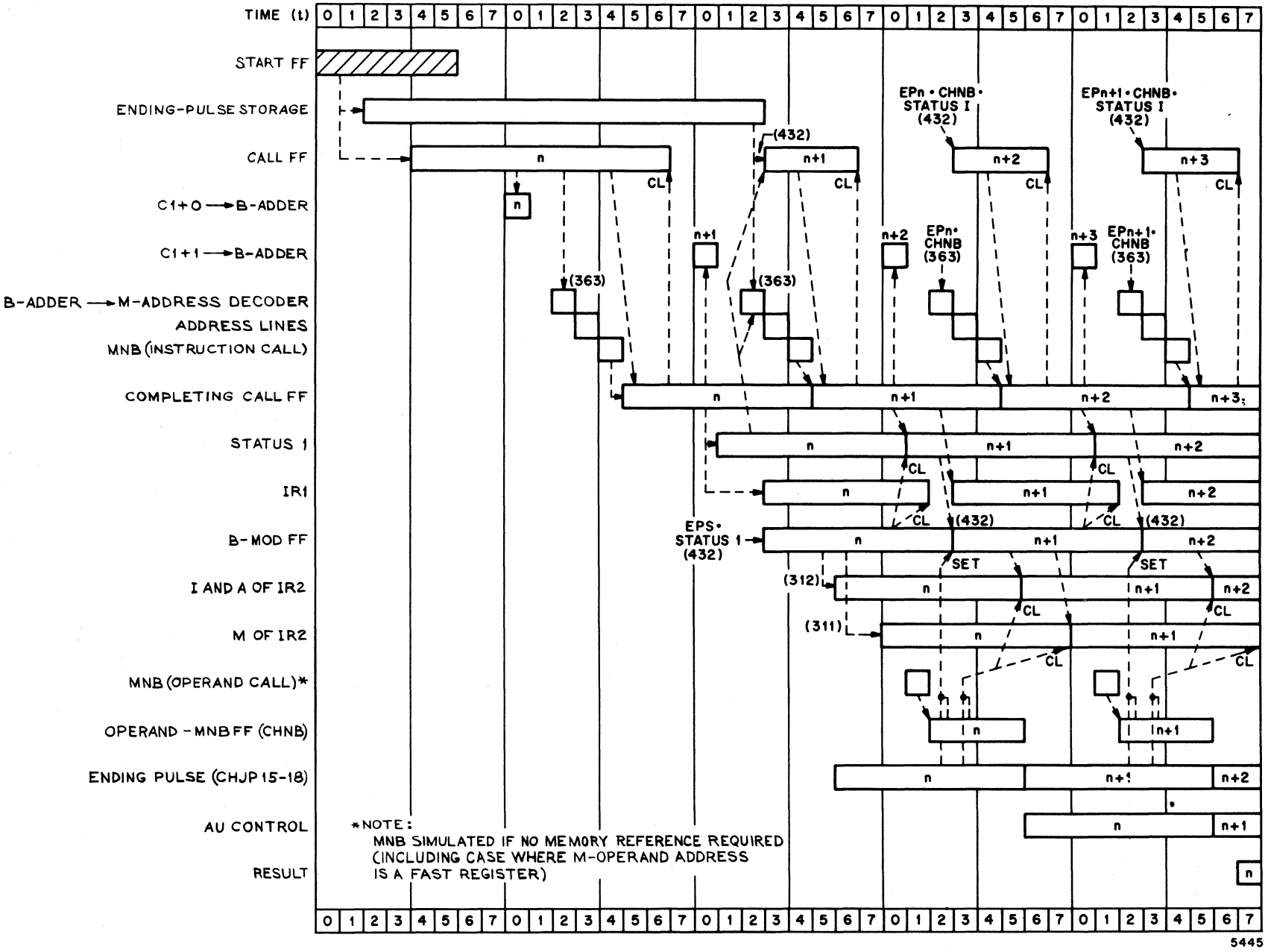


Figure 2-5. Timing Diagram - Normal Continuous Operation

(4) FS320 is produced with CT2 which gates instruction n from the memory to IR1.

Performing function (2) produces the address for instruction n + 1 which is gated to the address lines at t2 of the third cycle by FS363. For this second call, FS363 is generated by gating the outputs of the ending-pulse-storage FF and the status-1 FF. This same line which produces FS363 also generates FS432 which, in turn, resets the ending-pulse-storage FF and sets the call FF for n + 1. FS432 also sets the B-mod FF to coincide with the arrival of n in IR1 (step 4). As noted on figure 2-4, the output of the B-mod FF is used to control the B-modification process for operand selection while also controlling the resetting of the status-1 FF for n at t0 and the clearing of IR1 at t1 (FS321). It also gates, with CT5, the I and A digits from IR1 to IR2 (FS312), and, with CT7, the modified M digits from the B-adder to IR2 (FS311).

The setting of the call FF by FS432 again results in setting the completing-call FF if the MNB signal for n + 1 is received at t4. Thus the same sequence of events produced by the setting of the completing-call FF is repeated for n + 1. During the time n + 1 is being called and set up in IR1, the ending pulse (CHJP 15, 16, 17, or 18) for n is generated. As noted in figure 2-4, the ending pulse is active during the last 4 microseconds of the time that the instruction is being decoded from IR2; that is, during the 4-microsecond period (t6-t5) immediately preceding the last 4 microseconds of instruction execution time in the arithmetic unit. Since we are considering only 4-microsecond instructions at this time, the ending pulse for n is generated during the same interval as I and A of IR2 (figure 2-5).

If the operand call to the memory resulting from the B-mod operation is successful, the MNB signal will be present at t1 to set the operand-memory-not-busy FF. (Actually there are two operand-MNB FF's which are set at this time; reference heading 3-4.) If no operand memory reference is required, either because of the type of instruction or where the M-operand address specifies a fast register, the MNB signal is simulated to set the operand-MNB FF. The operand-MNB FF stores the operand-call MNB signal for 4 pulsetimes to enable various functions required by the instruction in IR2. Some of these functions are (1) referencing a fast register, (2) stepping the program counter, (3) transmission of instruction information to the AU (4) various functions of the instruction ending pulse related to the sequence control of following instructions. The operand-MNB signal together with the ending pulse indicate that the instruction can proceed to the execution phase and therefore the following instructions can advance.

By gating the output of the operand-MNB FF (CHNB) with the instruction ending pulse at t2, the following operations are performed:

(1) Output of B-adder is gated to memory address decoder to make call for instruction n + 2 (FS363).

(2) Call FF is set for n + 2 if status-1 FF is set, indicating n + 1 is in IR1 (FS432).

(3) B-mod FF is set for n + 1 if status-1 FF is set (FS432).

(4) I and A of IR2 is cleared of n at t5 (FS313).*

(5) M of IR2 is cleared of n at t7 (FS314).*

The timing for the execution and result phases of the overall cycle is shown in figure 2-5 simply with two entries: AU control and result.

* FS313 and FS314, not shown in figure 2-4, are derived indirectly from the ending pulse by way of the setting of the program-counter-clear FF (reference heading 3-5).

SECTION 3
INSTRUCTION SEQUENCE CONTROL

This section contains detailed descriptions of most conditions of instruction sequencing. Figure 3-1, which is referred to throughout this section, is a simplified logic diagram of the normal instruction cycle control circuits. In addition to including everything described in general terms under headings 2-8 and 2-9, figure 3-1 shows how various control signals are generated to take care of different situations which can occur during the normal sequence of operations. Each of the simplified logic diagrams in this manual includes references (D312, D844, etc.) to the related logic drawing numbers on which can be found the corresponding detailed logic.

In this section the principal general areas of control, those associated with fast register and memory operand references and the program counter, are described first (headings 3-1 through 3-5); special instruction control cases are then described (headings 3-6 through 3-13).

3-1. FAST REGISTER CHAIN CONTROL

The fast register chain control, figure 3-2, contains the logic for controlling the fast register operand (A register) references. These references include those required to obtain A operands, M operands which specify a fast register, and result storage registers. The control designations and purposes of these three references are:

(1) A-input chain — selects A operand from normal A operand address (AA) and supplies it to the A-input pulseformers of the AU.

(2) M-input chain — selects M operand when M address specifies a fast register address (999AA) and supplies it to the M input register of the AU.

(3) Result chain — selects A register to store result of instruction and controls transfer of result information from AU to fast register.

There are some variations in the use of the fast register chain controls. For example, some instructions use the M-input chain to supply the

normal A operand to the AU. Data transfer instructions which write information into the memory use the M-input chain; they also use the result chain when the M address specifies a fast register address. These operations are described under the next two headings but the classification of the instructions involved is described in section 6.

3-2. FAST REGISTER INPUT CHAIN CONTROL

A timing diagram of the A- and M-input chain control operations is given in figure 3-3. This diagram illustrates the repetitive operations for a series of four instructions of minimum length (4 microseconds). The normal A-operand address for instruction n is set up in the fast register selector register by first sending the A part of IR2 to the B-adder at t2 of the first cycle. This action is produced by FS403 which is generated by instruction control signal CHJP 30, 31, or 32. At the same time, for most instructions, control signal CHJP 33 or 34 causes zeros to enter the other input to the B-adder (FS411) so that the sum (A + 0) is equal to the original A-register address (AA). For some instructions different control signals are generated at t2 so that the A-address is modified by adding +1, +2, or +3 to AA. These cases are needed in double-precision operations for which an example is given under heading 3-10. Since the extra control signals are not shown in figure 3-2, they are listed below and additional information concerning them is in the Computing Unit Instruction and Function-Signal Analysis manual:

CHJP 35 - +1 to B-adder

CHJP 36 - +2 to B-adder

CHJP 37 - +3 to B-adder

CHJP 77 - +1 or 0 to B-adder

CHJP 78 - +1 or 0 to B-adder

All instructions that require the A-input chain generate control signal CHJP 27, 28, or 29. Either of these signals sets the FR-to-A-input FF (signal CHBA) if the operand-MNB FF is set (CHNB). Signal CHNB indicates here that there will be no delay in obtaining the M operand. Signal CHBA is also produced by gating CHJP 77 or 78 with the absence of the operand-MNB-FF signal ($\overline{\text{CHNB}}$). This is a special case associated with double-precision operations (reference heading 3-10).

At t4 the contents of the B-adder (A + 0) are read out and transferred to the fast register selector register by FS431 which is generated by gating CHBA with CT4. Instruction information for the AU is transferred from the group encoder to the AU at t5 with FS300 which is generated by gating CHJP 38, 39, 40, 41, or 75 with CHNB and CT5.

Signal CHBA is applied to a pair of gates that control the setting of either the equality FF or the inequality FF depending on the results of an A-register comparison test. The details of the comparison test are described under headings 3-11 and 3-12 dealing with A-register access conflicts. For the present discussion assume that the inequality FF is set at t6 of the first cycle to produce signal CHBC.

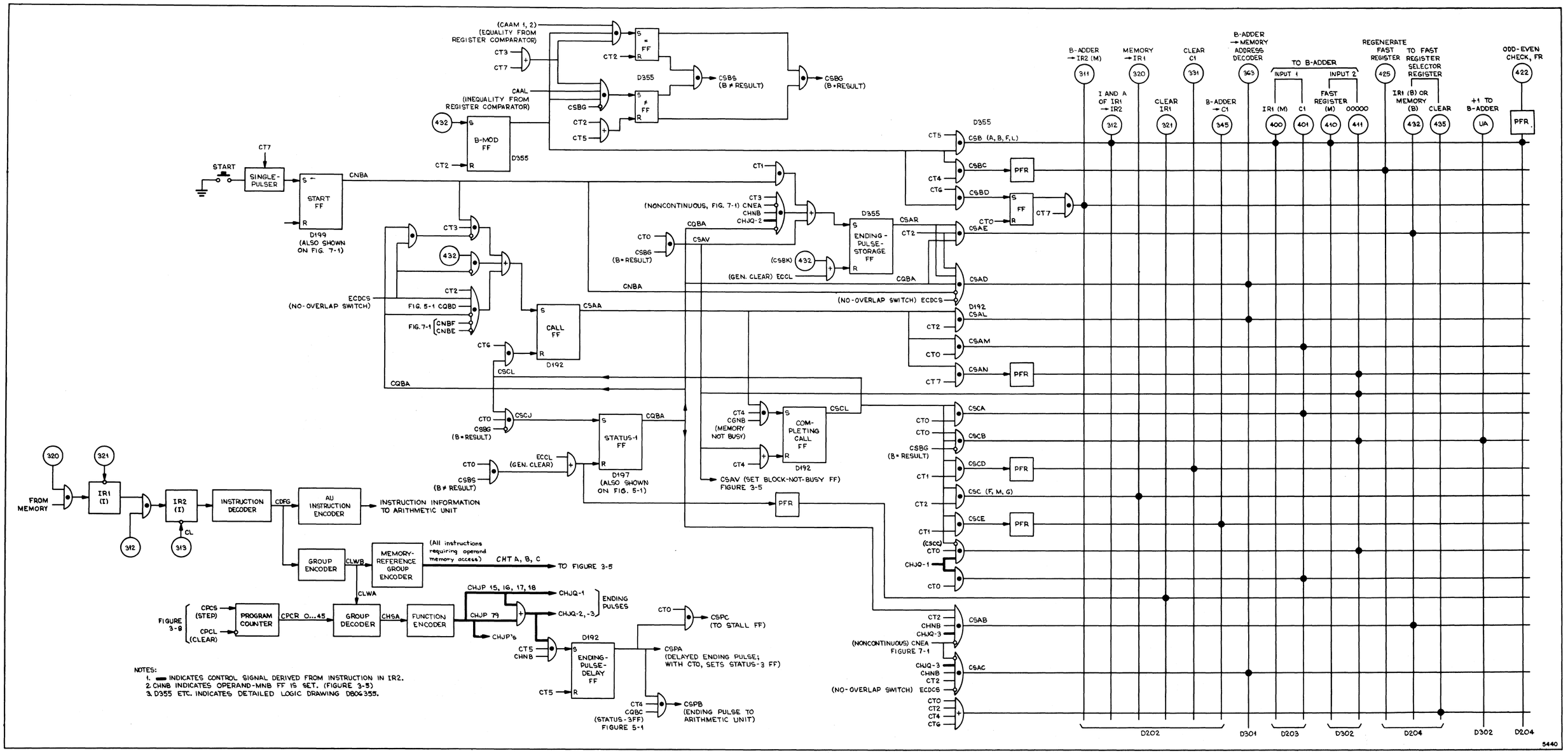
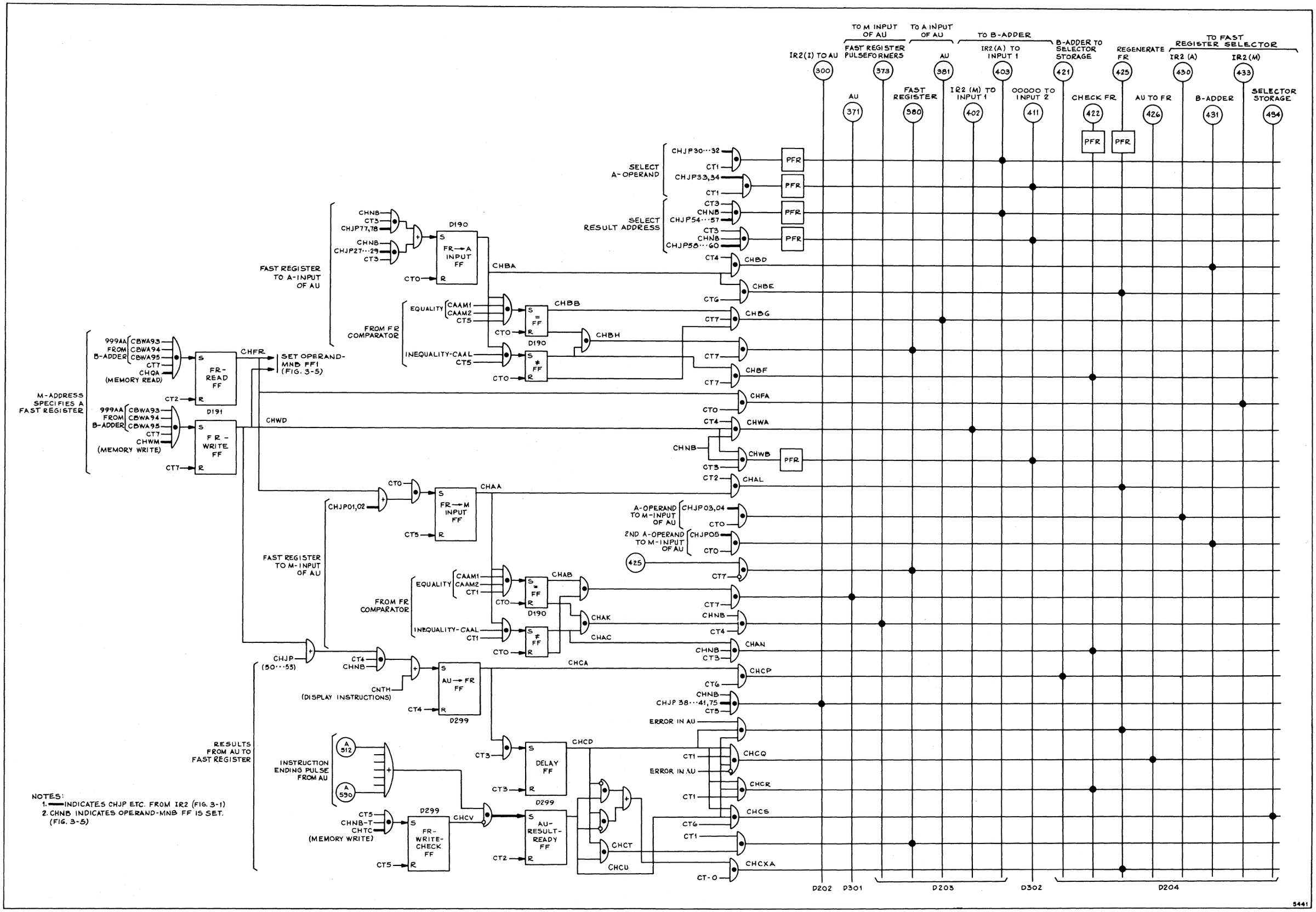


Figure 3-1. Simplified Logic Diagram - Normal Instruction Cycle



NOTES:
1. ——INDICATES CHJP ETC. FROM IR2 (FIG. 3-1)
2. CHNB INDICATES OPERAND-MNB FF IS SET. (FIG. 3-5)

Figure 3-2. Simplified Logic Diagram - Fast Register Chain Control

Signal CHBC is gated with the reset output from the equality FF to produce signal CHBH which, in turn, is gated with CT7 to give FS380. At the same time FS425 is generated by gating CHBA with CT6. (Signal CHBA and CT6 give CHBE which is pulseformed to give FS425 at t7.) FS380 and FS425 then cause the contents of the selected A register to enter the A-input pulseformers of the AU while also being regenerated in the fast register. (The signal path is shown in the block diagram in figure 2-1.)

By gating CHBC with CT7, signal CHBF is generated to produce FS422 at t0. FS422 activates the odd-even check circuits associated with the A-input pulseformers and thereby provides a check on the contents of the fast register. The reason FS422 is dependent on the inequality FF signal is explained under heading 3-12 dealing with A-register access conflicts.

The case where the M address specifies a fast register (999AA) is handled by the fast register M-input chain control logic of figure 3-2. The memory addresses 99901 to 99999 are reserved for the fast registers. Thus, any M address in which the three most significant digits are 999 is detected as a fast register reference. As indicated in figure 3-3, detection is accomplished following the B-modification of the M address at t7 at the output of the B-adder (signals CBWA 93, 94, and 95).

For a fast register M-read operation, the FR-Read FF is set (signal CHFR) by the B-adder signals, a general M-read signal (CHQA, reference heading 3-4), and CT7. Signal CHFR is then gated with CT0 to generate FS433 which transfers the two least significant digits (AA) of the modified M address from the M part of IR2 to the fast register selector register. Since no true memory reference is made in this case, signal CHFR produces a simulated operand-MNB signal (CHNB) by setting the operand-MNB FF.

The FR-to-M-input FF is set (signal CHAA) to start the M-input chain control by gating CHFR with CT0. As shown in figure 3-2, this FF may also be set by instruction control signal CHJP 01 or 02 for certain instructions that use the M-input chain for the normal A-operand input to the AU. Also in connection with this feature, control signal CHJP 03 or 04 is gated with CT0 to send the A part of IR2 directly to the fast register selector register (FS430). Another special control signal is CHJP 05 which, with CT0, sends the contents of the B-adder to the fast register selector register (FS431). For the particular instructions associated with these operations, refer to the classification of instructions in section 6; also see the Computing Unit Instruction and Function-Signal Analysis manual, heading 2-12 (page 31).

Signal CHAA is applied to the setting gates of a second pair of equality-inequality FF's that also depend on the results of an A-register comparison test (headings 3-11 and 3-12). As with the A-input chain assume that the inequality FF is set so that signal CHAC is produced from t2 of the first cycle through t0 of the second cycle. At t2 signal CHAA is gated with CT2 to produce CHAL which, in turn, is pulseformed to give FS425 at t3. FS425 is then gated with the absence of a CT7 timing signal to give FS380. These two function signals then cause the contents of the selected A register to enter the A-input pulseformers of the AU while also being regenerated in the fast registers.

The output from the inequality FF is gated with the reset output from the equality FF to produce signal CHAK. Signal CHAK is then gated with

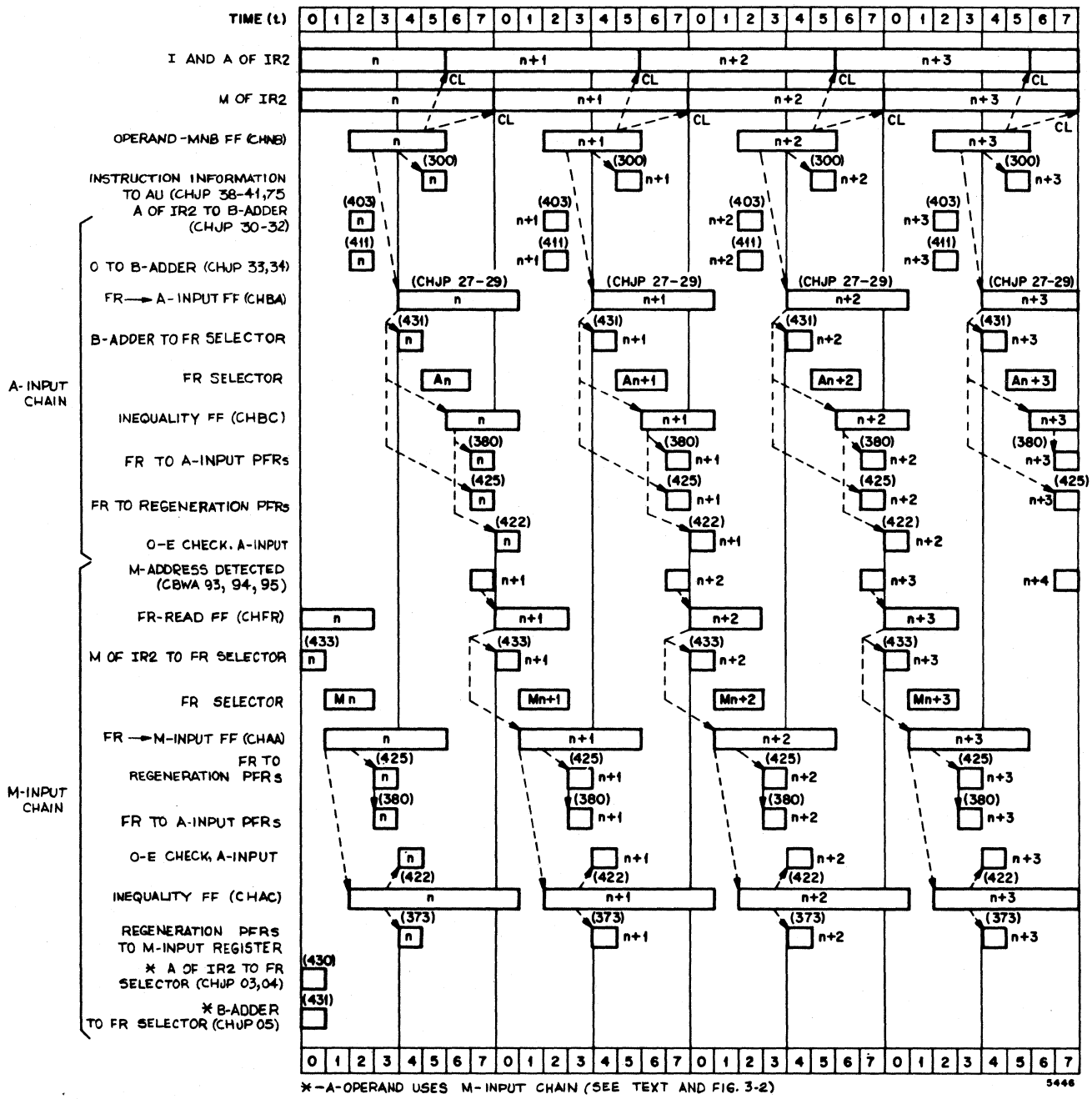


Figure 3-3. Timing Diagram - Fast Register Input Chain Control

CHNB and CT4 to produce FS373 which transfers the contents of the selected A register from the output of the regeneration pulseformers to the M-input register of the AU. The inequality FF signal (CHAC) is also gated directly with CHNB and CT3 to give FS422 at t4. Thus, the M-input chain control transfers information to both the A and M inputs to the AU. The A input is used only to provide the odd-even check on the fast register information. The M-input register then holds the M operand until the normal A operand arrives at the A input at t7.

3-3. FAST REGISTER RESULT CHAIN CONTROL

A timing diagram showing several minimum-length instructions passing through the result chain control logic is given in figure 3-4. The following discussion, however, is confined to instruction n which can be assumed to be the same 4-microsecond instruction set up in IR2 during the first cycle shown in figure 3-3 for the A-input chain. Following the setting of the operand-MNB FF, signal CHNB is gated with control signal CHJP 54, 55, 56, or 57 and CT3, and also with control signal CHJP 58, 59, or 60 and CT3, to produce FS403 and FS411 at t4. These function signals send the A part of IR2 and zeros to the B-adder in order to form the result address. For double-precision operations the second result address is formed by sending A + 1 to the B-adder at t4 in which the +1 (unit add) is generated by control signal CHJP 61 or 62 (reference heading 3-10).

As shown in figure 3-2, the result chain control is activated by setting the AU-to-FR FF (signal CHCA). This flip-flop is set by gating CHNB with CT4 and result-chain-control-signal CHJP 50, 51, 52, or 53. Signal CHCA is then gated with CT6 to produce FS421 which transfers the result address from the B-adder to the selector storage. The selector storage holds the address for 4 microseconds (t7 - t6) until it can be transferred to the fast register selector register. Signal CHCA is gated with CT3 to set the delay FF (CHCD) at t4. The delay FF is needed to store control information until the result of instruction n can be transferred from the AU to the fast register.

Signal CHCD is gated with timing signals and the output of the AU-result-ready FF (CHCU — set by an instruction ending pulse from AU control) to produce the following control signals to complete the operation:

FS434 (t6) — transfers contents of selector storage (result address) to fast register selector register.

FS380 (t1) — reads out old information from selected fast register to A-input pulseformers of AU for check purposes.

FS426 (t1) — transfers contents of AU result register to fast register, if no error in AU.

FS422 (t2) — checks A input of AU to ensure that (1) a fast register was selected and (2) that only one was selected.

The generation of FS426 is conditional on no errors being detected in the AU for instruction n. If an error is detected, the gate that produces FS426 is inhibited and instead a gate is opened to allow the generation of FS425 at t1. This action regenerates the information in the selected fast register.

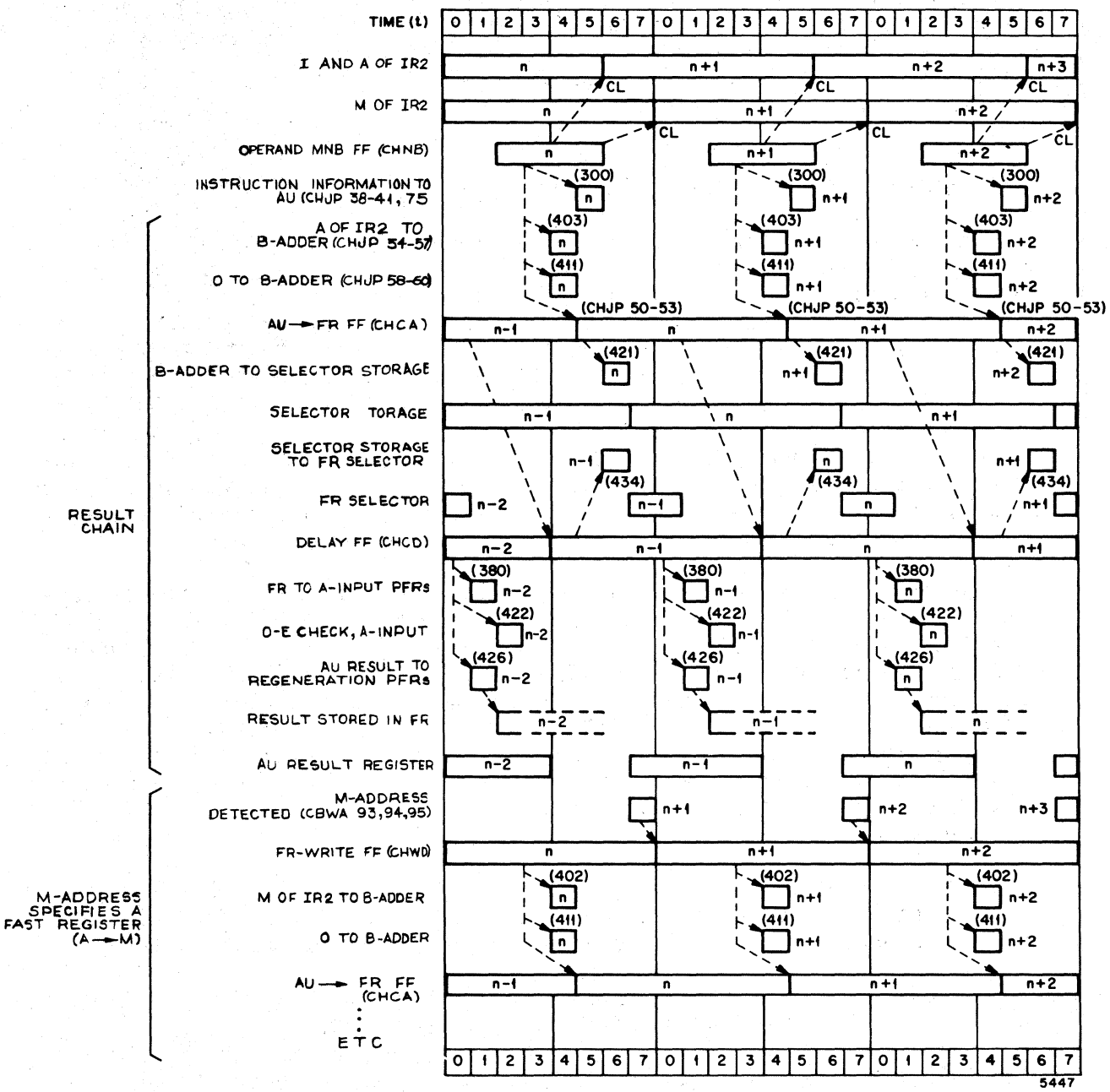


Figure 3-4. Timing Diagram - Fast Register Result Chain Control

The result chain is also activated in the case where the M-address specifies a fast register on memory-write (store) instructions (reference heading 3-4 and section 6). Figures 3-2 and 3-4 include the logic and timing for handling this situation. Detection of the M address is accomplished in the same way as described under the preceding heading for the M-input chain control. In this case the FR-write FF is set (signal CHWD) at t0 and then gated with CHNB (produced by CHWD) and CT4 to generate FS402 and FS411. These function signals transfer the two least significant digits of the modified M address from the M part of IR2 and zeros to the B-adder to form the result address. Signal CHWD also sets the AU-to-FR FF to start the result chain. The rest of the operations are then carried out as before.

Also shown in figure 3-2 is the FR-write-check FF (signal CHCV) which is used to inhibit the setting of the AU-result-ready FF by the AU ending pulse. This action prevents the operations of the result chain in the case of a memory-write instruction which does not refer to a fast register (indicated by signals CHTC and CHNB-T which set the FR-write-check FF; reference heading 3-4). The reason the inhibit is necessary is because all instructions which merely transfer information to or from the memory are handled in the same way in the AU in order to provide for the case where the M address does specify a fast register. Therefore, all such instructions produce the AU ending pulse that sets that AU-result-ready FF.

3-4. OPERAND MEMORY REFERENCE CONTROL

The simplified logic diagram in figure 3-5 shows the basic logic concerned with operand memory reference control for all instructions requiring an operand memory reference. Associated timing diagrams are given in figures 3-6 and 3-7. For a memory read operation, the appropriate memory reference group encoder signals (CLWB'S — figure 3-1) are buffed together to give memory reference control signals CHTA, CHTB-1, and CHTB-2. The group encoder signals are also gated with the relevant program counter stage signal (CPCR-0 for one memory reference or CPCR-1 for second memory reference in double-precision operations) to give memory-read signal CHQA. Signal CHRM (and $\overline{\text{CHQA}}$) is then generated to produce FS363 and FS365 by gating CHQA with CT7. These function signals transfer the modified M address from the B-adder (formed by B-mod operation) to the address lines to make the operand call at t0. At the same time $\overline{\text{CHQA}}$ generates read signal YBD for memory control purposes.

The operand-MNB signal from the memory (CGNB) is received for instruction n at t1 of the second cycle (figures 3-6) to set the operand-MNB FF's. As shown in figure 3-5, there are two operand-MNB FF's which are set by gating CGNB with CHTB-2 and CT1. These flip-flops are designated operand-MNB FF1 (signals CHNB 1, 2) and operand-MNB FF2 (signal CHNB-T). Basically, the operand-MNB FF's are used to remember the receipt of an operand-call MNB signal. When it is set. operand-MNB FF1 enables various functions required by the instruction in IR2 as outlined previously under heading 2-9. Any or all of these functions may be required in those cases where no actual memory reference is involved. These cases arise either because of the nature of the instruction or where the M-operand address specifies a fast register. Another case is that of a long instruction in which the ending pulse occurs during a cycle in which no memory reference is

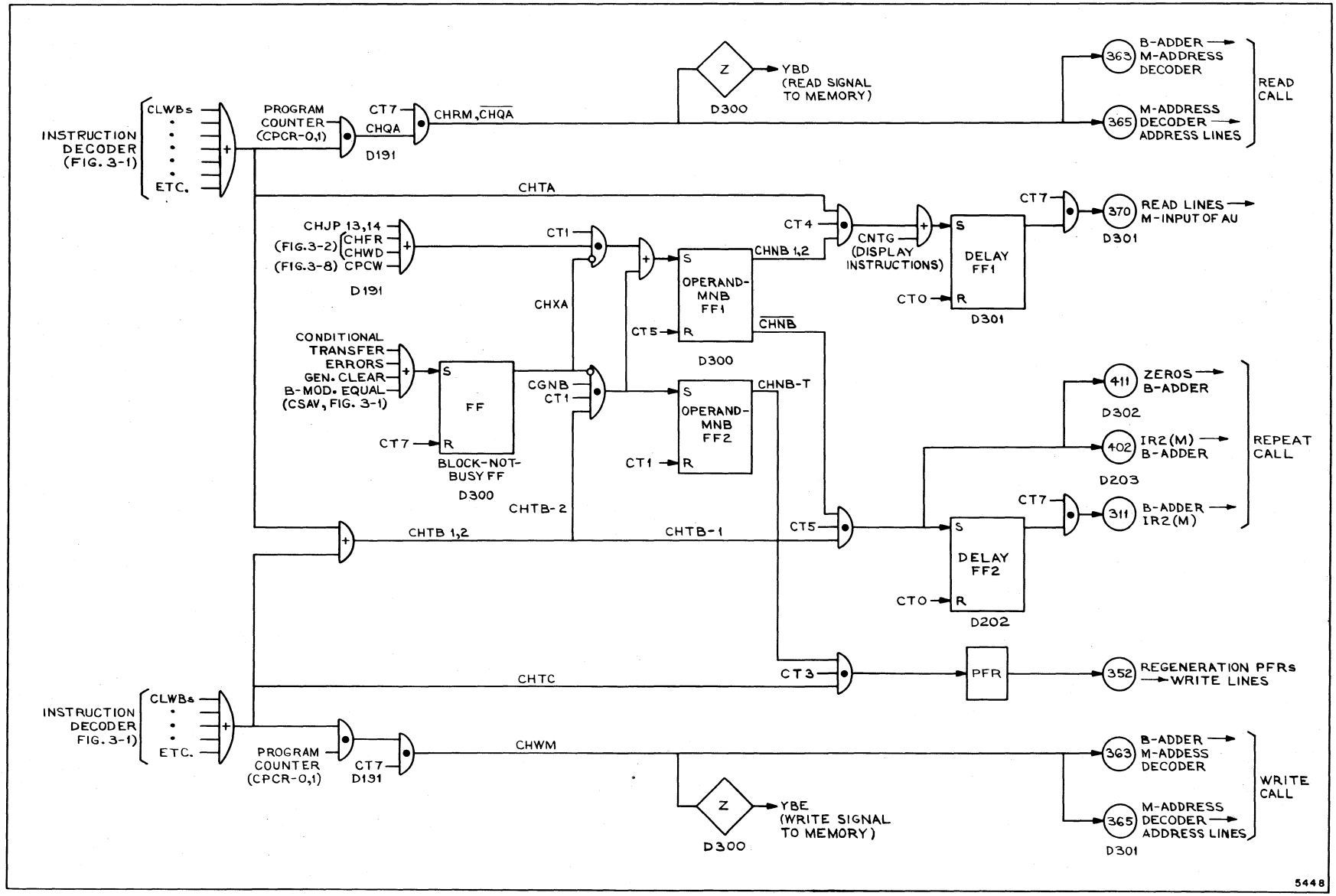


Figure 3-5. Simplified Logic Diagram - Operand Memory Reference Control

made. Consequently, operand-MNB FF1 may also be set by a simulated MNB signal which can be derived in several ways:

(1) By control signals generated for instructions requiring no memory reference (signals CHJP 13, 14)

(2) By signals which indicate that the M-operand address specifies fast register for either memory-read or memory-write instructions (signals CHFR, CHWD — reference headings 3-2 and 3-3)

(3) By program-counter-repeat FF signal which steps program counter during long instructions (signal CPCW — reference heading 3-5).

Operand-MNB FF2 (CHNB-T) is set by a "true" MNB signal only and controls several functions of which only two are considered here (reference heading 3-3 and end of this heading). The other functions controlled by CHNB-T are concerned mainly with the details of memory-write instructions which are described in Instruction Sequence Control, Part II.

Referring to the memory-read operation (figure 3-6), signal CHNB-1 is gated with the memory-read control signal (CHTA) and CT4 to set delay FF1 at t5. The output from the delay FF is then gated with CT7 to produce FS370 which gates the M operand from the memory read lines into the M-input register of the AU.

If the memory is busy on an operand call, certain other control functions must take place in order to repeat the call. The timing of these operations is shown in figure 3-6 where it is assumed that the memory is busy on the first operand call for instruction n + 1. The call is made at t0 of the third cycle (FS363, 365) but the MNB signal is not received at t1 and operand-MNB FF1 is not set. Consequently, the reset output of operand-MNB FF1 ($\overline{\text{CHNB}}$) is gated with control signal CHTB-1 and CT5 to generate FS402 and FS411 which send the M part of IR2 and zeros to the B-adder. The output of the same gate also sets delay FF2 which, with CT7, generates FS311 to gate the B-adder output back to the M part of IR2. Since instruction n + 1 is retained in IR2, signal CHRM is generated at the same t7 (from CHQA and CT7) which again gives FS363 and FS365. These actions gate the address from the B-adder to the M-address decoder and address lines to repeat the call for instruction n + 1. (The details of the overall instruction cycle control for this case is given under heading 3-7.)

Also shown in figure 3-6 is the case where the M address specifies a fast register address (instruction n + 2). As described previously for the M-input chain control under heading 3-2, the M address is detected at the first t7 after the instruction is set up in IR2 to set the FR-read FF (signal CHFR, figure 3-2). This signal is then used to set operand-MNB FF1 and to start the M-input chain control operations which transfer the contents of fast register 999AA to the M-input register of the AU.

The timing for a memory-write operation is given in figure 3-7. Many of the functions produced here are similar to those carried out for a memory-read operation with memory-write control signals CHTC, CHWM, and YBE replacing CHTA, CHRM, and YBD, respectively. The memory is addressed for instruction n at t0 of the second cycle and the MNB signal is returned at t1 to set the operand-MNB FF's. In this case the output of operand-MNB

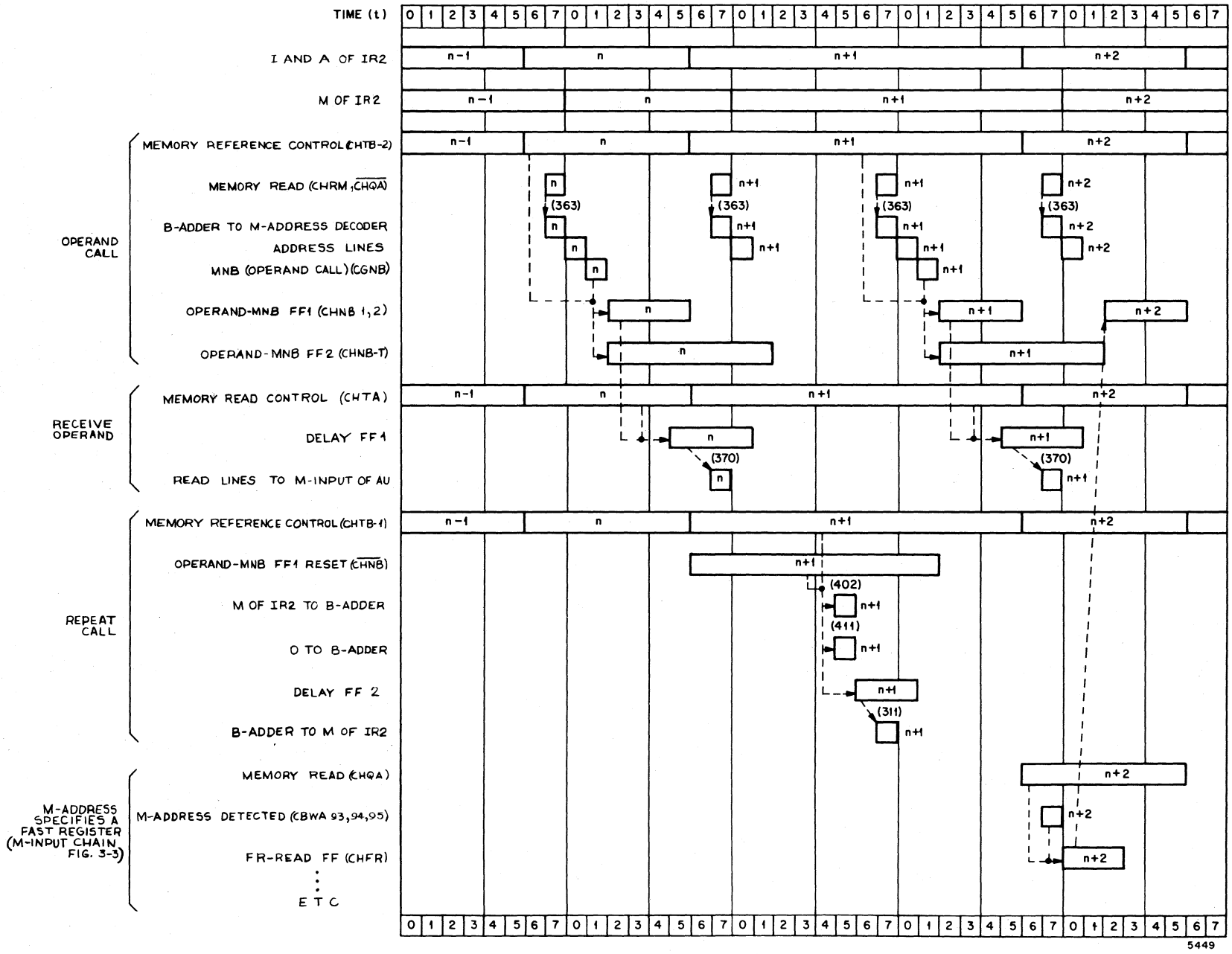


Figure 3-6. Timing Diagram - Memory Reference Control, Read

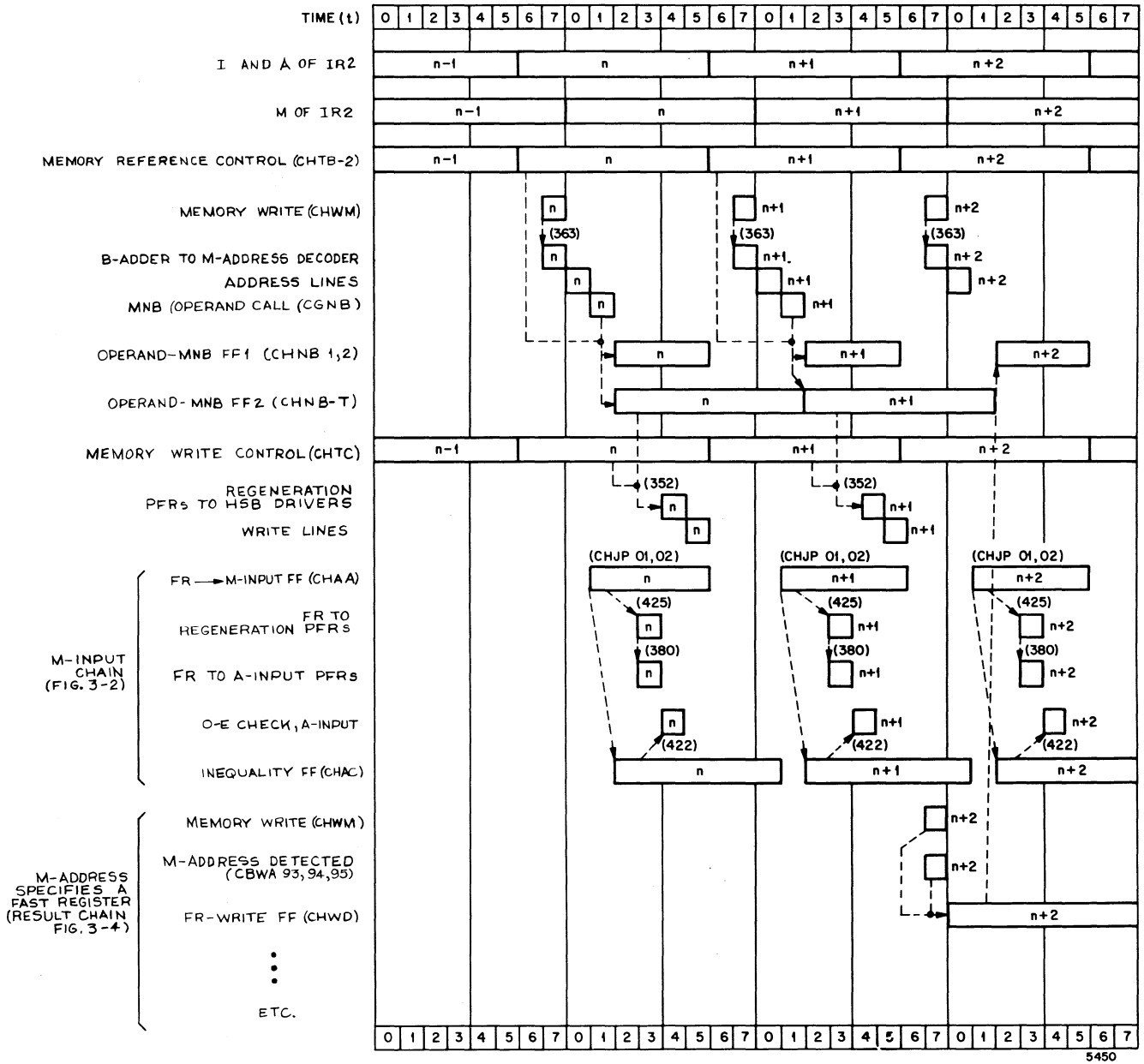


Figure 3-7. Timing Diagram - Memory Reference Control, Write

FF2 (CHNB-T) is gated with CHTC and CT3 to produce FS352 at t4 which transfers the contents of a fast register to the memory write lines.

In order to select and read out the contents of the fast register to be transferred to the memory, the M-input chain control logic is activiated by control signal CHJP 01 or 02 at t1 of the second cycle. The information is then gated to the write lines by way of the fast register regeneration pulseformers (see figure 2-1) while also being gated to the A-input pulseformers of the AU for check purposes (reference heading 3-2). Many other control functions are involved in the case of a memory-write operation in which the memory is busy, or there is a fast register access conflict, and are covered in Instruction Sequence Control, Part II.

The case where the M address specifies a fast register for a memory-write instruction is shown in figure 3-7 for instruction n + 2. By gating the detected M-address signals with memory-write signal CHWM at t7, the FR-write FF is set (CHWD) to activate the result chain control logic while also setting operand-MNB FF1 (reference heading 3-3). Thus, the contents of the selected fast register are transferred to the AU by way of the M-input chain and then written into another fast register by way of the result chain. For this operation operand-MNB FF2 is not set and FS352 is not generated.

Also shown in figure 3-5 is the block-not-busy FF which is set under certain conditions to block many of the functions generated by an instruction in IR2. Most of the conditions which set the block-not-busy FF are concerned with special cases related to errors and conditional transfers and are covered in Instruction Sequence Control, Part II. Its effect here is concerned only with the blocking of the setting of the operand-MNB FF's in the case of a B-register access conflict which is described under heading 3-13.

3-5. PROGRAM COUNTER CONTROL

The program counter is used to distinguish successive 4-microsecond timing intervals, for control purposes. It is always cleared to read zero at the time a new instruction enters IR2 for execution and it can only be stepped upon the reception of an operand-MNB FF1 signal (CHNB-1 or CHNB-2). A simplified diagram of the control circuits associated with the program counter is given in figure 3-8.

The PC-clear FF (signal CPKD) is set from t4 through t1 of the last cycle an instruction is in IR2. This is produced by gating the ending pulse for the current instruction (CHJQ-2, heading 4-3) with CHNB-1 and CT3. Signal CPKD is then gated with timing signals to generate the signals that control the clearing of the program counter and IR2 as shown in figure 3-8. The PC-clear FF is also set by the block-not-busy signal (CHXA) in order to clear IR2 in the case of a B-register conflict (headings 3-4 and 3-13).

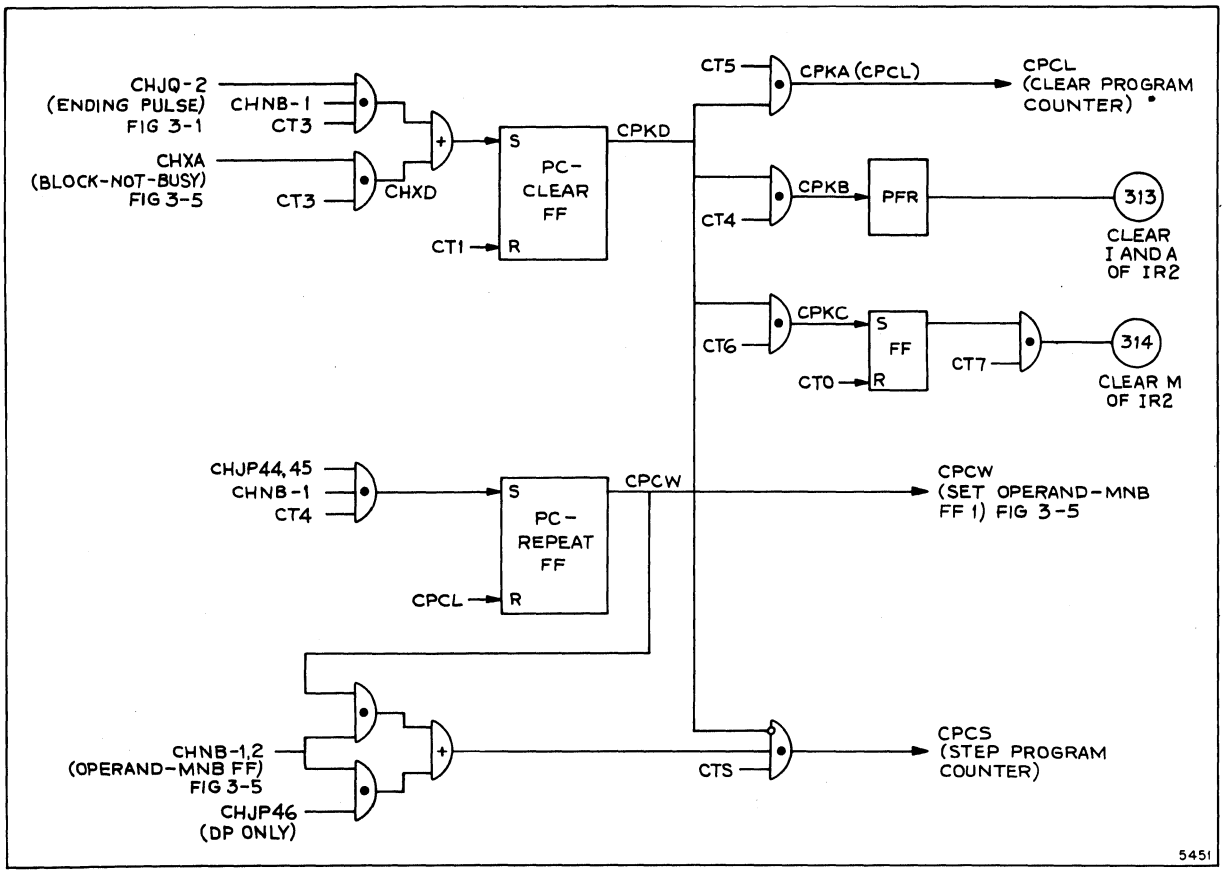


Figure 3-8. Simplified Logic Diagram - Program-Counter Control Circuits

The PC-repeat FF controls the stepping of the program counter during long instructions; that is, any instruction requiring more than 4 microseconds of execution time. The PC-repeat FF (signal CPCW) is set by gating control signal CHJP 44 or 45 with CHNB-1 and CT4. In a memory reference instruction in which the MNB signal is not received, CPCW is not generated and the program counter remains at zero so that the same control signals are generated by the instruction decoder-encoder.

Signal CPCW controls all program-counter steps above zero (CPCR-0) required in single-precision memory reference instructions and in instructions with no memory reference. In double-precision memory reference instruction, CPCW is used for steps higher than one (CPCR-1) only because of the second operand memory reference, and the step from CPCR-0 to CPCR-1 is controlled by a separate step-program-counter signal (CHJP 46 • CHNB-1). The separate control signal is required because CPCW also jams operand-MNB FF1 and this would interfere with the true operand memory reference. An example of the use of signal CPCW to control the timing of program-counter steps in a long instruction is given under heading 3-9.

3-6. SPECIAL INSTRUCTION SEQUENCE CONTROL CASES

The overlapping technique of instruction execution can result from time to time in various conflicts in the timing of operations within the control unit. In addition, sequencing delays arise when an addressed memory unit is already engaged with another operation. Other special sequence control cases are those associated with long instructions (greater than 4 microseconds) and double-precision instructions. Beginning with heading 3-7 and continuing through heading 3-13, all of the special cases related to the normal instruction cycle (figure 3-1) are described. References are continually made to the descriptions given under headings 3-1 through 3-5.

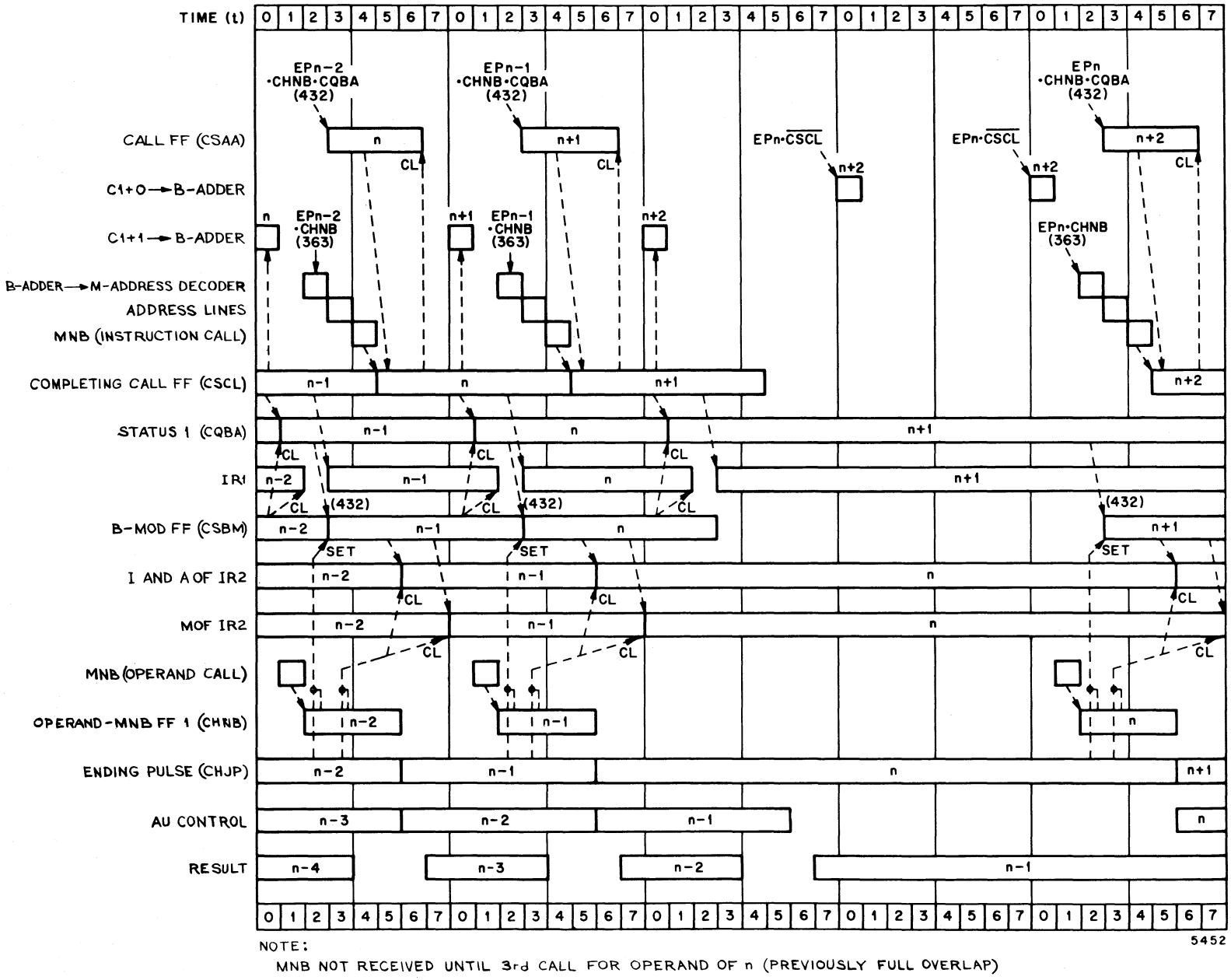
3-7. MEMORY BUSY ON OPERAND CALL

The first case — memory busy on operand call — is shown in the timing diagram of figure 3-9 which assumes that previous full overlap conditions exist. The four instructions preceding instruction n are in various phases of execution during the first time cycle. Thus, the explanation begins with the control counter being stepped to instruction n as shown by C1 + 1 going to the B-adder at t0. Referring to figure 3-1, this action is produced by FS401 and the unit-add signal, UA. These two signals are produced by signals CSCA and CSCB which are generated by gating the output of the completing-call FF (CSCL) for instruction n - 1 with CT0. Also notice that CSCB depends on the absence of signal CSBG (B = result) which indicates no B-register conflict in instruction n - 2 (reference heading 3-13).

The new address (n) is gated from the B-adder to the memory address decoder at t2 by FS363. Signal CSAC produces FS363 by gating the ending pulse (CHJQ-3) and the output of operand-MNB FF1 (CHNB) for instruction n - 2 with CT2. The same signals, gated with the status-1 FF signal (CQBA) for n - 1, produce CSAB to generate FS432. The call FF is then set for n and the B-mod FF for n - 1. The call for n is on the address lines at t3 and the MNB signal is received at t4. By gating the MNB signal (CGNB) with the call FF signal (CSAA), the completing-call FF is set for n. Notice that signals CSAB and CSAC are inhibited by either or both the no-overlap signal (ECDCS) and the noncontinuous-operation signal (CNEA) so that further sequencing is stopped whenever the computing unit is operating in one of interrupted modes of operation (reference section 7).

Meanwhile, instruction n - 2 in IR2 is decoded and an operand call is made to the memory at t0. As with an instruction call, an operand call is made by generating FS363 which gates the output of the B-adder to the MAD (reference heading 3-4). In addition to producing FS363 and FS432 to call instruction n, the ending pulse and CHNB for n - 2 also produce FS313 and FS314 by way of the program-counter-clear FF to clear IR2 (reference heading 3-5).

While n - 2 is sent to the arithmetic unit for execution, n - 1 is decoded, n is transferred from the memory to IR1, and n + 1 is called. As shown during the third cycle on figure 3-9, the control counter is stepped to n + 2 by the completing-call FF (CSCL) which also sets the status-1 FF (CQBA) and brings n + 1 into IR1. Instruction n has its M address modified, is then set up in IR2, and an operand call is made at t0 of the third cycle. However, the MNB signal is not received at t1 and operand-MNB FF1 is not



NOTE:
MNB NOT RECEIVED UNTIL 3rd CALL FOR OPERAND OF n (PREVIOUSLY FULL OVERLAP)

Figure 3-9. Timing Diagram - Memory Busy on Operand Call

set to produce CHNB at t2. This lack of action prevents the generation of FS363 and FS432 by the ending pulse for instruction n so that the call for n + 2 is not sent over the address lines and the M address of n + 1 is not modified. In addition, IR2 is not cleared during the third-cycle. The B-mod and completing-call FF's are reset, however, by timing signals CT2 and CT4, respectively. Thus n is retained in IR2 through the fourth cycle. The absence of the MNB signal then causes a repeat operand call to be made at t0 of the fourth cycle (reference heading 3-4).

The fact that the completing-call FF is reset prevents C1 from being stepped in the fourth cycle. Instead, signal CSCC (figure 3-1; equivalent to $\overline{\text{CSCL}}$) is gated with CT0 and the ending pulse (retained from instruction n) to send C1 + 0 to the B-adder and thereby permit n + 2 to be called if the MNB signal is received on repeat call for operand of n. The failure to set the B-mod FF prevents the clearing of IR1 and the resetting of the status-1 FF for n + 1. As shown in figure 3-9, all of the foregoing conditions prevail again during the fourth cycle because of the absence of the operand-MNB signal at t1.

Following the third call for the n operand at t0 of the fifth cycle, the MNB signal is present to set operand-MNB FF1 and restore the normal sequencing (FS363, 432, and so on). Thus it is evident that the two consecutive missing MNB signals result in an increase of 8 microseconds in the normal execution time of instruction n. During this period, the execution of instructions n - 2 and n - 1 are completed in the arithmetic unit and the results are stored in the fast registers (reference heading 3-3). However, the result of n - 1 is retained in the AU result register beyond the fifth cycle and is not cleared out until t3 of the sixth cycle (sixth cycle not shown in figure 3-9). Normally each instruction in the AU clears the result register of result of last instruction at t3 of the first cycle of arithmetic operations. This is true for most instructions although for a few instructions the clearing takes place during the second cycle of AU operations. Therefore, the delay in clearing n - 1 from the AU result register is due to the delay in getting instruction n into the AU which, in turn, is due to the delay in receiving the operand-MNB signal for n.

3-8. MEMORY BUSY ON INSTRUCTION CALL

The delay which occurs in the instruction sequencing when the memory is busy on an instruction call is illustrated in figure 3-10 with previously full overlap conditions again present. The call for n is made at t3 of the first cycle but the MNB signal is not received at t4. The missing MNB prevents the setting of the completing-call FF at t5 which, in turn, prevents the resetting of the call FF at t6 and the setting of the status-1 FF at t1. The remaining parts of the instruction cycle for n - 1 are carried out in normal fashion.

The control counter is prevented from being stepped during the second cycle by the absence of the completing-call FF signal (CSCL). Instead, the call FF output (CSAA) gates C1 + 0 to the B-adder. Also notice that the ending pulse for n - 1 and signal $\overline{\text{CSCL}}$ (completing-call FF reset) produce the same effect. However, this effect is incidental and would not apply if the MNB signal was again missing on a second instruction call. The call for n (FS363) is then repeated by gating the output of the call

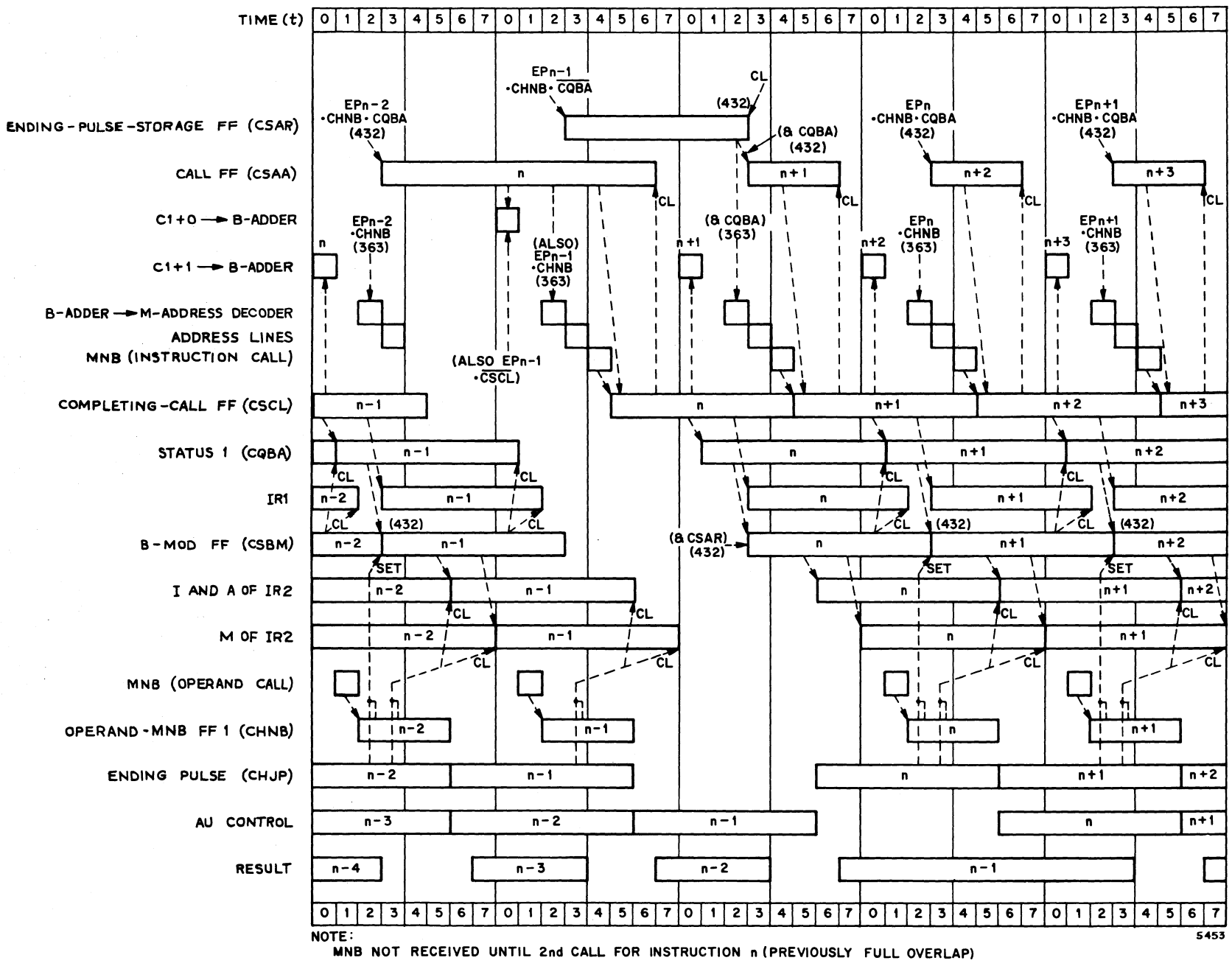


Figure 3-10. Timing Diagram - Memory Busy on Instruction Call

FF with CT2. The MNB signal is received at t4 to set the completing-call FF for n and thereby allow the resumption of normal sequencing of following instructions.

The two calls for instruction n result in a loss of 4 microseconds between the execution of n - 1 and the resumption of execution of n. During this time the ending pulse for n - 1 is no longer generated due to the clearing of IR2. To provide for the ending-pulse function that enables the call for n + 1 and the B-modification of n, the ending-pulse-storage FF is set at t4 of the second cycle by gating the ending pulse for n - 1 with CHNB and the absence of the status-1 FF signal ($\overline{CQBA}$). The output of the ending-pulse-storage FF (CSAR) is then gated with CQBA for n and CT2 in the third cycle to produce FS363 and FS432. These actions complete the call for n + 1, permit the B-modification of n, and thereby restore full overlap. The ending-pulse-storage FF is immediately reset by FS432.

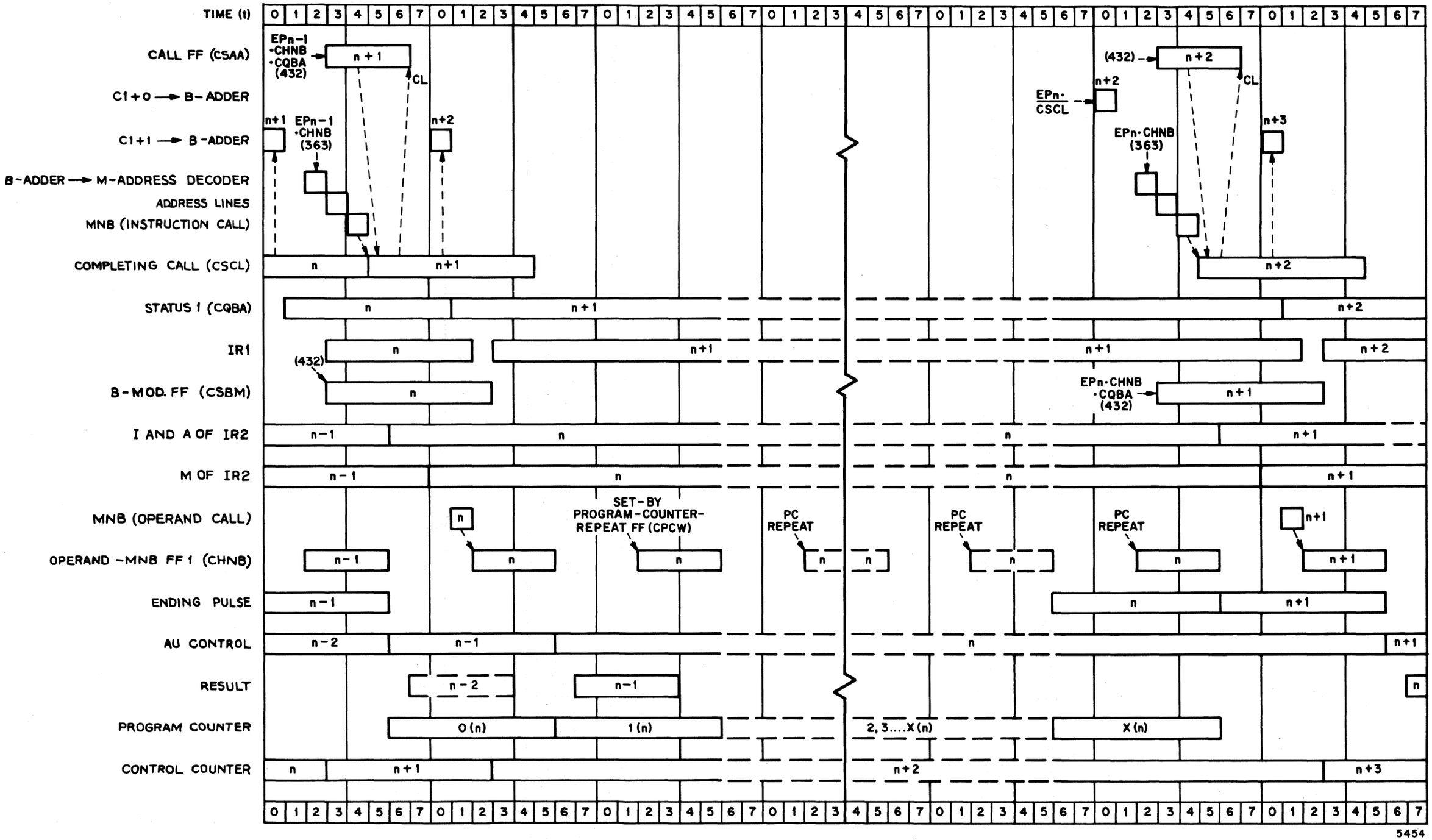
3-9. LONG INSTRUCTIONS

A long instruction is any instruction longer than 4 microseconds. The previous descriptions in this manual were limited to single-precision 4-microsecond instructions. However, all double-precision and many single-precision instructions take more than 4 microseconds to execute. Under this heading certain features common to all long instructions are described whereas those features peculiar to double-precision operations are covered separately under the next heading (3-10).

The sequencing of instructions immediately before and after a long instruction is the same as in the normal continuous operations discussed previously. The total time that the long instruction remains in IR2 is equal in duration to the execution time required in the AU, but it begins before the start of execution and ends before end of execution in the AU. The main factor to be remembered with regard to a long instruction is that the instruction ending pulse is not generated until the last 4 microseconds of the time that the instruction is in IR2. Since the program counter distinguishes successive 4-microsecond intervals, it controls the generation of the ending pulse until the appropriate time. For example, a 30 instruction (divide order requiring 32 microseconds of execution time) extends for 8 program-counter stages (PC0 - PC7) and the ending pulse is not generated until PC7.

The timing for a generalized single-precision long instruction (n) is given in figure 3-11. Instruction n + 1 is called during the first cycle while n is being set up in IR2. (The program counter is cleared to zero by the ending pulse and CHNB of n - 1; reference heading 3-5.) During t0 of the second cycle, C1 is stepped to n + 2 and the operand call is made for n. At t2, operand-MNB FF1 is set and at t3 instruction n + 1 is in IR1. Due to the absence of the ending pulse for n, nothing else happens in the control unit as far as instruction sequencing is concerned.

In order to step the program counter for the number of stages required to complete the instruction, the program-counter-repeat FF (signal CPCW — reference heading 3-5) is set during the first t5 after the setting of operand-MNB FF1 for instruction n. If the memory is busy on the operand call, then the program counter remains at zero and the same control signals are produced by the instruction in IR2. Signal CPCW remains on until



5454

Figure 3-11. Timing Diagram - Long Instruction

after the ending pulse for n is generated. While CPCW is being generated, it jams operand-MNB FF1 to give CHNB during each t2 - t5 interval which, in turn, gates with CPCW to produce the program-counter-step signal (CPCS) at each t5.

During the last program counter stage (Xn in figure 3-11), the ending pulse for n is generated to set the program-counter-clear FF which, in turn, clears IR2 and resets the program-counter-repeat FF. Normal sequencing (C1 + 0 ⟶ B-adder, FS363, 432 and so on) is then resumed as described in previous sections.

3-10. DOUBLE-PRECISION INSTRUCTIONS

Double-precision instructions always require more than 4 microseconds of execution time and thus everything described under the preceding heading for single-precision long instructions applies equally well here. A timing diagram of a double-precision add operation in which no delays are encountered is given in figure 3-12. This instruction extends over 12 microseconds (program-counter stages PC0-PC2) with two operand references in the first and second program-counter stages and two uses of the result chain in the second and third stages. The most significant half (MSH) of the operation is executed first and the A and M operands for the least significant half (LSH) are then obtained simply by adding 1 to the original A and M addresses.

Starting with the double-precision instruction (n) in IR2, the M-operand call for the MSH (Mn) is made in the normal fashion at t0 of the second cycle and operand-MNB FF1 is set at t2 (reference heading 3-4). Signal CHNB is then gated with double-precision control signals from the instruction group encoder (CHJP 09, 10, 12, and 74) to set up the M-operand call for the LSH (M + 1n). Signals CHJP 09 and 10 cause the M part of IR2 + 1 to enter the B-adder at t5 and CHJP 74 clears Mn from IR2 at t7. CHJP 12 then reads the contents of the B-adder to the M part of IR2 at t7 to store M + 1n in order to allow repeat operand call if the memory is busy. The call for M + 1n is made at t0 (4 microseconds after the call for Mn) and operand-MNB FF1 is set again at t2.

Meanwhile, the A address for the MSH (An) is sent to the fast register selector register in the normal way at t4 of the second cycle (as shown in figure 3-12). The two operands (An) and (Mn) then enter the AU for execution at t0 of the third cycle. The second A operand (A + 1n) is obtained by sending A or IR2 + 1 to the B-adder at t2 of the third cycle. The +1 is generated by control signal CHJP 77. At t5, A + 1n is set up in the fast register selector register and at t7 the A and M operands for the LSH arrive at the AU inputs for the second add operation.

In order to generate the +1 for the second A-operand reference, signal CHJP 77 depends on the MNB signal for the second operand call, M + 1n. If the memory is busy, then the first operand (Mn) is retained in the M-input register (as controlled by AU) and the first A operand must be supplied again to the AU so that the AU can repeat its first cycle. Therefore, if the memory is busy (indicated by $\overline{\text{CHNB}}$), CHJP 77 supplies a 0 instead of +1 to the B-adder at t2 and starts the A-input chain control (reference heading 3-2).

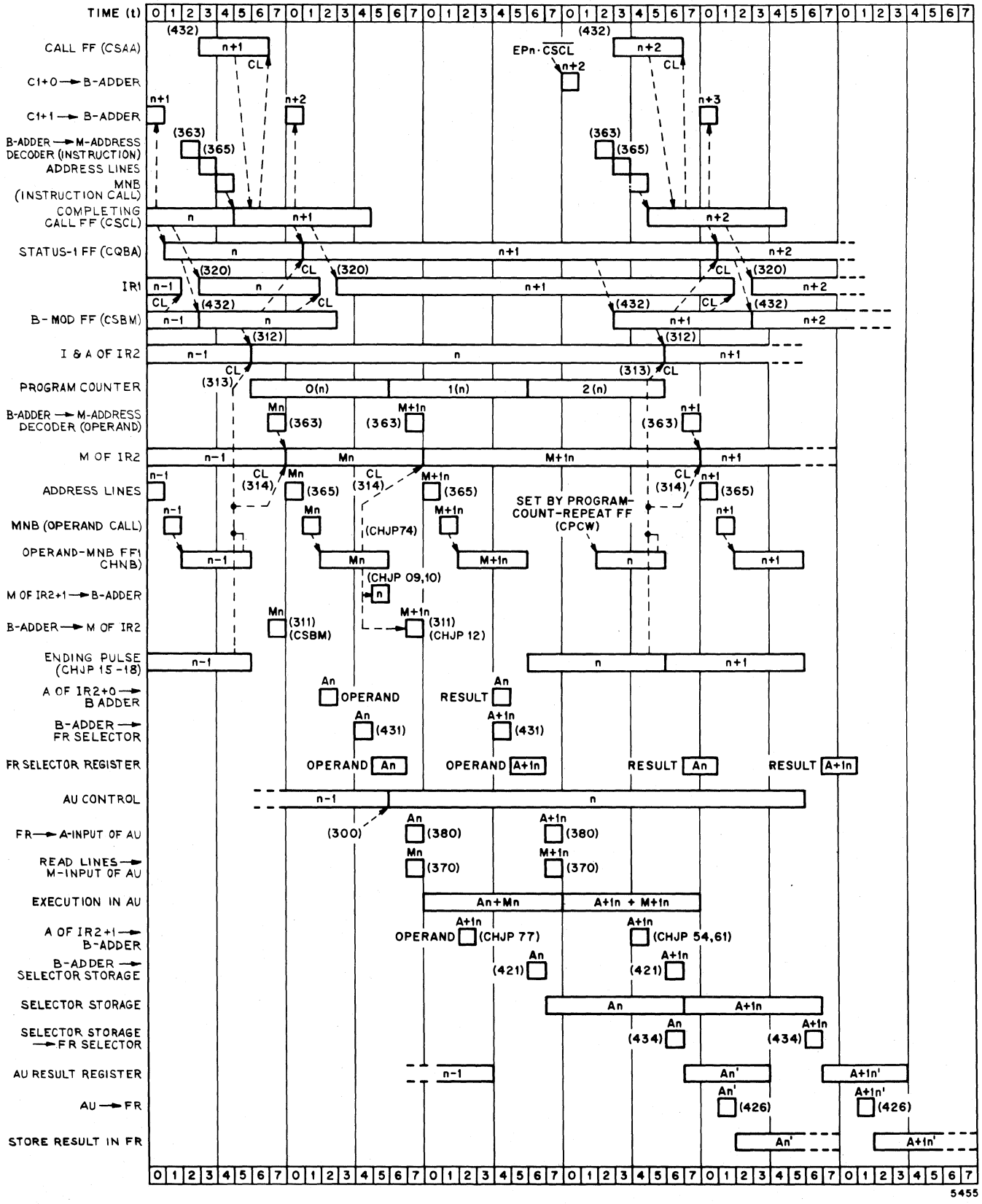


Figure 3-12. Timing Diagram - Double-Precision Instruction

The result storage for the MSH is set up in the third cycle (program-counter stage 1n) by first sending An + 0 to the B-adder at t4 and then to selector storage with FS421 at t6 (reference heading 3-3). For a double-precision addition, the result for the MSH is not complete until the LSH has been executed. (The result of the first add operation is held in the AU until the LSH is finished.) Hence, the AU result register is not filled with the result for the MSH until t7 of the fourth cycle.

The result of the MSH operation is transferred from the AU result register to the selected fast register with FS426 at t2 of the fifth cycle. Similarly, the result address for the LSH is set up at t4 of the fourth cycle (CHJP 54 and 61) and sent to selector storage. At t6 of the fifth cycle, the contents of the selector storage (A + 1n) are sent to the fast register selector register with FS434. The LSH result (A + 1n') is then stored in A + 1n at t2 of the sixth cycle.

There is one variation in the sequence control regarding double-precision multiply and divide instructions. Because of the way in which the arithmetic unit operates on these instructions, the LSH part of the operation must be started first and then the MSH. To take care of the M-operand call for the LSH (M + 1n), a + 1 (unit add) is generated and sent to the B-adder during the normal B-modification operation so that the modified M address is increased by 1. In order to provide for this special condition to the B-modification process, a separate set of decoding gates is attached to IR1 so that the double-precision multiply and divide orders are detected in adequate time.

After the successful call for M + 1n, signal CHNB is gated with instruction control signals CHJP 09, 11 and 74 to set up the M-operand call for the MSH (Mn). Signals CHJP 09 and 11 cause the M part of the IR2 minus 1 to enter the B-adder at t5 and CHJP 74 clears M + 1n from IR2 at t7. Thus, the M-operand select process is just the reverse of that shown in figure 3-12 for a double-precision add operation.

The same reversal of operand selection is also provided for in obtaining the two A operands for a double-precision multiply or divide operation. The A operand for the LSH (A + 1n) is obtained by sending A of IR2 + 1 to the B-adder at t2 of the second cycle, and then to the fast register selector register at t5. In this case the +1 is generated by control signal CHJP 35. The second A operand (An) is obtained simply by sending A + 0 to the B-adder in the normal way at t2 of the third cycle.

The result chain control is not affected by multiply and divide orders since the result is not stored until both halves of the operation are complete. Thus, the result for the MSH is stored first and is followed by the LSH result as shown in figure 3-12.

3-11. FAST REGISTER ACCESS CONFLICTS

The use of the fast registers to store A operands, results, and index information raises the possibility of conflicts in the timing of control operations during normal (overlapped) instruction sequencing. The conflicts arise when access to a fast register for certain information is made prematurely. This situation can occur simply because of the nature of the program sequence. For example, a certain instruction, n, may require as an

A operand the result of the preceding instruction, n - 1. In this case the A-register address for n would be the same as the result address of n - 1. Because of overlap, the result of n - 1 will not yet be available from the AU at the time the A operand for n is selected and a conflict occurs. Another example is where the M-operand address of an instruction is to be modified by the result of a previous instruction.

The control unit includes additional logic circuits which detect these conflicts and automatically correct for them when they occur. The instruction execution time may or may not be increased depending on the type of conflict. Detection is accomplished by means of the fast register selection comparator which continually compares the result address in selector storage with the addresses of fast registers selected by any following instruction during the time that the address is in selector storage. See control block diagram, figure 2-1.

The 1-microsecond access time of the fast registers permits them to be addressed four times during any one 4-microsecond period. On the other hand, a single instruction may require up to four different fast register references over the period of the complete instruction cycle. The timing for these references are:

(1) B-register address (t3)

(2) A-operand address (t5)

(3) Result address (t7)

(4) M-operand address when M address specifies a fast register (t1).

Therefore, during a single 4-microsecond period of overlapped operation, the fast register selector register may hold up to four different fast register addresses for three different instructions as follows:

(1) B-register address for instruction n

(2) A-operand address for instruction n - 1

(3) Result address for instruction n - 2

(4) M-operand address for instruction n when M specifies a fast register.

The selector storage holds the result address for an instruction for 4 microseconds while that instruction is being executed in the AU. So long as the result address stored in selector storage is not equal to any of the fast register addresses of following instructions, no conflicts can occur and the comparator will produce an inequality signal to permit normal control operations to proceed. If the comparison process discloses two addresses that are alike (that is, in conflict), an equality signal will be generated to initiate variations in the sequence control.

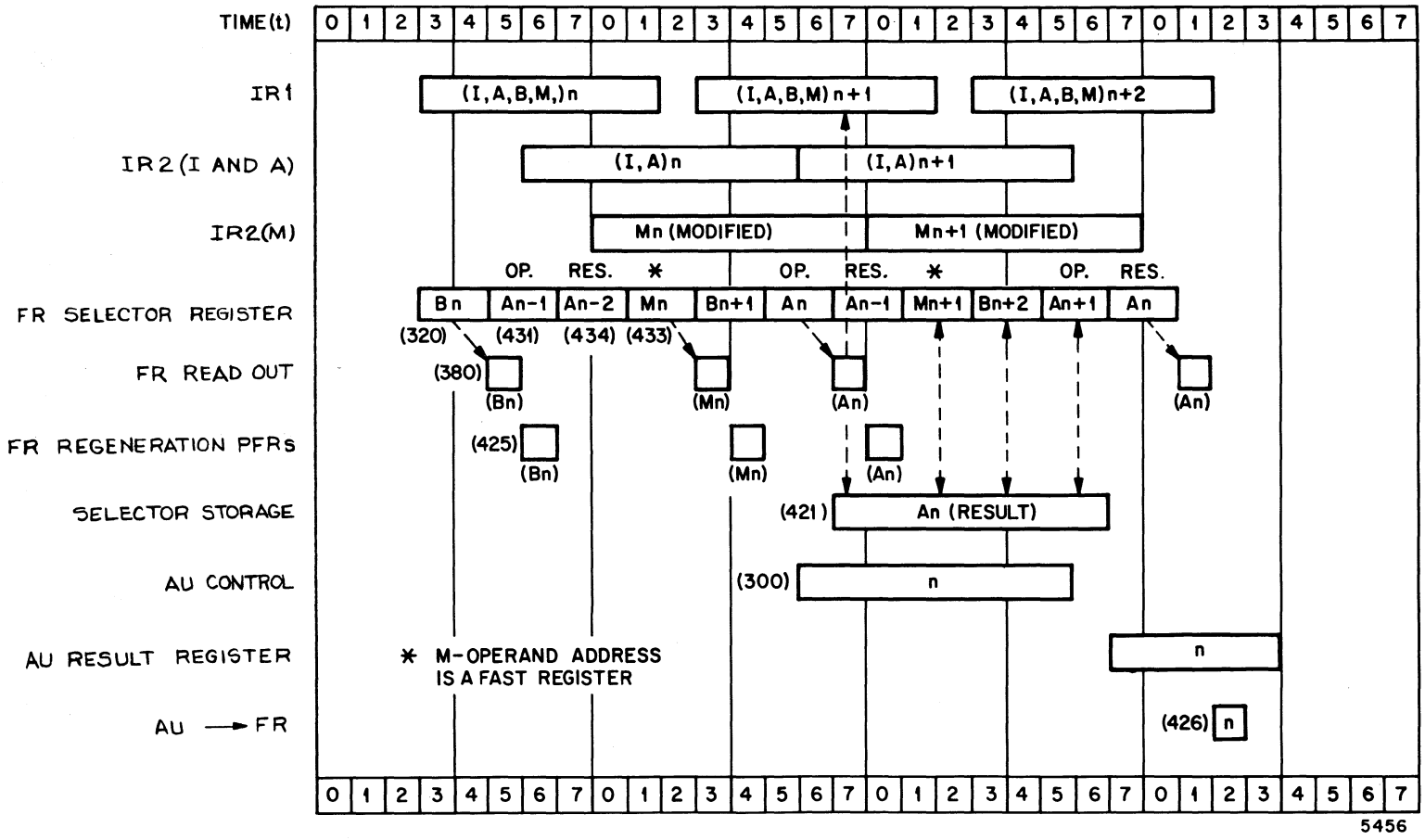


Figure 3-13. Timing Diagram - Fast Register References

The actual conflicts in timing that can occur with the result address for instruction n (An) stored in the selector storage are as follows:

Time	Conflict
t7	B n + 1 = A n
t1	M n + 1 = A n (M is a fast register address)
t3	B n + 2 = A n
t5	A n + 1 = A n

The case which occurs at t7 does not involve the fast register selector register and is discussed further in section 3-13. An overall timing diagram of all fast register references which shows these conflict possibilities more clearly is given in figure 3-13. The next two headings cover the details of how the various conflicts are resolved.

3-12. OPERAND REFERENCE (A REGISTER). Fast register operand references include the normal A-operand address (AA) and the case where the M-operand address specifies a fast register (999AA). Conflicts between AA and the result address of the preceding instruction can occur at t5; those between the result address and 999AA at t1. The logic for handling these conflicts is shown in the fast register chain control for the A- and M-inputs to the AU; figure 3-2.

The A-input chain selects the A operand and supplies it to the A-input pulseformers of the AU (reference heading 3-2). Normally, FS380 gates the contents of the selected fast register to the AU at t7 if the inequality FF is set by signal CAAL from the comparator. However, if the comparator detects equal addresses, the equality FF is set by signals CAAM 1, 2 to produce signals CHBB and CHBG. This action indicates that the result of the preceding instruction, now in the result register of the AU, is to become the A-operand input for the current instruction. Consequently at t7 FS381 is generated instead of FS380 to gate the result from the AU back to the A-input pulseformers and no time is lost. See control block diagram, figure 2-1.

The odd-even check on the A-input pulseformers is not instituted in this case since it is intended as a fast register check only. Therefore FS422 is not generated whenever the equality FF is set instead of the inequality FF (reference heading 3-2).

The case where the M address specifies a fast register is handled in a similar manner by the pair of equality/inequality FF's associated with M-input chain control logic; figure 3-2. If an inequality condition exists, the contents of the selected fast register are transferred to the M-input register of the AU with FS373 and the information is checked with FS422 (reference heading 3-2).

If the result address for the preceding instruction is equal to the fast register M-operand address, the contents of the AU result register must be gated back into the AU as with the case of equality with the A input. In this case the equality FF is set (signal CHAB) and at t7 FS371 is

generated instead of FS373 to gate the AU result directly to the M-input register and no time is lost. In addition FS422 is not generated for the same reason given for the A-input equality case.

3-13. ADDRESS MODIFICATION (B REGISTER). The B-modification operation takes place as soon as a new instruction is set up in IR1. The B digits are transferred into IR1 along with the rest of the instruction word (FS320 at t2), and also directly into the fast register selector register (FS320 and FS432 at t2). As shown in the simplified timing diagram of figure 3-13, the B-register references for both n + 1 and n + 2 occur before the result is stored for instruction n. Therefore, a conflict can occur for either of these references. The B digits for n + 2 are in the fast register selector register during t3 and t4 of the time that An is in selector storage so that a comparison between the contents of the selector register and selector storage at t3 will detect any conflict.

The selection of the B register for n + 1 is made before An is in selector storage. However, the B digits for n + 1 are still in IR1 when An enters selector storage so that a comparison is also made at t7 between selector storage and the B part of IR1 (figure 2-1). Thus, two comparisons are made for every B-register selection; that is, with the result addresses of the two preceding instructions.

As shown in figure 3-1, the output signal from the B-Mod FF (CSBM) is applied with timing signals CT3 and CT7 to the setting gates of the B-modification comparison FF's. Depending on the comparator signals (CAAM or CAAL), either the equality FF (CSBG) or the inequality FF (CSBS) will be set to control the sequencing. Unlike A-register conflicts, a B-register conflict always causes some loss in time. The timing diagrams given in figures 3-14 and 3-15 illustrate the delays caused by these conflicts.

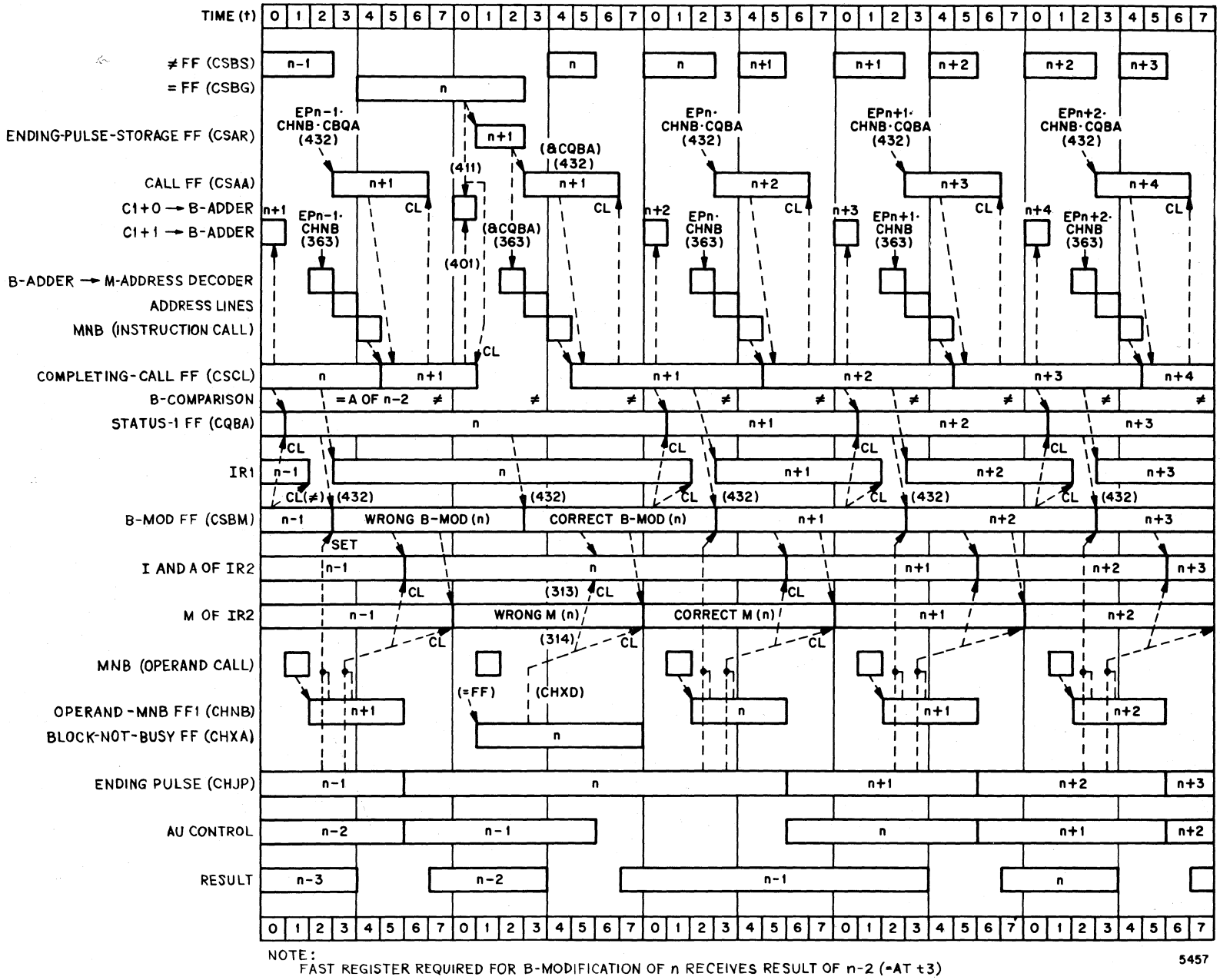
Figure 3-14 shows the case when equality is detected at t3 between the B-register address for instruction n and the result address of instruction n - 2; that is, the M address of n is to be modified by the result of n - 2. During the first cycle of the illustration, the call for n + 1 is made and the completing-call FF is set at t5. Instruction n is set up in IR1 and the B-Mod FF is set at t3 so that the normal B-modification process takes place. However, the equality signal from the comparator (CAAM) then sets the equality FF (CSBG) at t4 to indicate that the wrong B-modification took place and therefore must be repeated. Signal CSBG is gated with CT0 to produce CSAV which performs the following functions:

(1) With completing-call FF, gates C1 + 0 to B-adder to allow repeat call for n + 1. Notice that CSBG inhibits the +1 from going to the B-adder as is usual with the setting of the completing-call FF.

(2) Resets completing-call FF to prevent n + 1 from entering IR1.

(3) Sets ending-pulse-storage FF to repeat call for n + 1 and repeat B-mod for n.

(4) Sets block-not-busy FF to prevent CHNB and also to clear wrong modified M-address from IR2.



NOTE:
FAST REGISTER REQUIRED FOR B-MODIFICATION OF n RECEIVES RESULT OF n-2 (=AT t3)

5457

Figure 3-14. Timing Diagram - B-Register Conflict at t3

The block-not-busy FF (signal CHXA—shown in figure 3-5) blocks the setting of either operand-MNB FF by a true or simulated MNB signal for instruction n at t1 (reference heading 3-4). This blocking is necessary since the wrong B-modification is completed with the old B-register content, the wrong memory operand is called, and the instruction goes to IR2 and is decoded while it is at the same time retained in IR1 for the repeat B-mod. The clearing of the wrong M address from IR2 is produced by setting the program-counter-clear FF with signal CHXD as shown in figure 3-8 (reference heading 3-5).

Instruction n is retained in IR1 because of the absence of the inequality FF signal (CSBS) at t1 of the second cycle. The B-mod FF is set again by FS432 (from ending-pulse storage FF) to repeat the B-mod operation for n. Since CSBG remains on through t2 of the second cycle, the B-register comparison at t7 is ignored by blocking the setting of the inequality FF at this time.

The second B-modification operation will use the correct modifier since by this time the result of instruction n - 2 is stored in the fast register. The correct operand call for n is made at t0 of the third cycle and the MNB signal generates CHNB for n to restore full overlapped operation. As indicated in the timing diagram of figure 3-14, this conflict causes an increase of 4 microseconds in the effective execution time of instruction n.

The last case of fast register conflict occurs when the M address of an instruction is to be modified by the result of the preceding instruction. The timing for this case is shown in figure 3-15 where the result address of n - 1 is equal to the B address of n. The B-modification of n is started in the usual way at t3 of the first cycle. At t7 the result address of n - 1 is in selector storage and the equality FF is set to produce CSBG at t0 of the second cycle. The variations in control operations initiated by CSBG are similar to those described for the preceding case in figure 3-14 and an initial delay of 4 microseconds results.

The next B-register comparison at t3 will also result in a case of equality since this is the same comparison as the one that produced equality at t7. This situation occurs because FS432 comes on at t2 of the second cycle to gate the B digits of n from IR1 to the fast register selector register. Since the selector storage still contains An - 1, the comparator will detect the same equality condition. Thus, signal CSBG will again be generated to institute another 4-microsecond delay in the sequencing process so that the total time lost is 8 microseconds. The temporary delay situations and the resumption of normal sequencing is controlled as described previously for the equality-at-t3 case.

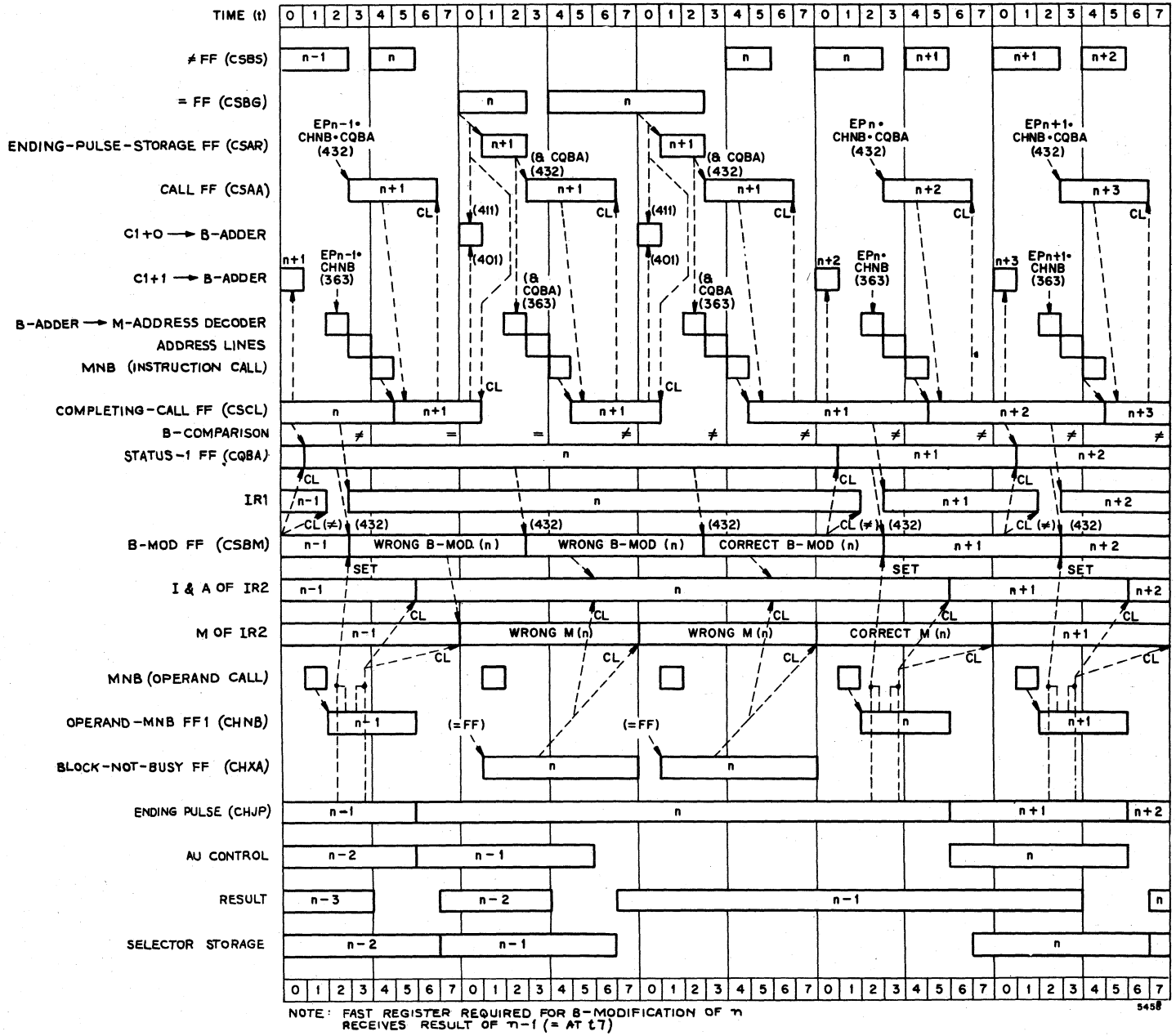


Figure 3-15. Timing Diagram - B-Register Conflict at t7

SECTION 4
SUMMARY OF ENDING PULSE FUNCTIONS

4-1. INSTRUCTION ENDING PULSE

All instructions generate an ending pulse (CHJP 15, 16, 17, 18, or 79) during the last program-counter step used by the instruction; that is, during the 4-microsecond period (t6 through t5) immediately preceding the last 4 microseconds that the instruction is in the arithmetic unit (AU). Signal CHJP 79 is the ending pulse for unconditional transfer of control operations (instructions 90, 91, and 92). It is also used for index instructions 80 and 81 which are sequenced as unconditional control transfers until a signal is received from the AU indicating that control must be returned to the main program; that is, the sequence control for instructions 80 and 81 assumes that the transfer of control will normally occur. (The details of unconditional transfer of control operations are covered in Instruction Sequence Control, Part II. Additional references to operations covered in part II are made simply to "Part II.")

Ending pulses CHJP 15 through 18 are identical in function and one or another of these is generated for any instruction other than the five discussed in the preceding paragraph. The function of the ending pulse is to control the overlapped sequencing of following instructions; that is, the ending pulse indicates that control unit operations are completed for a particular instruction (apart from the result chain control operations which are carried out automatically once they are started) so that the following instructions in sequence can be advanced to the next phase of the instruction cycle. Most of the ending pulse functions are conditional since they depend on the results of other events in the control sequence.

As shown in figure 3-1, signals CHJP 15 through 18 are buffed together to produce three main control lines (CHJQ-1, -2, -3) which are, therefore, all active for the duration of any normal ending pulse. The unconditional transfer ending pulse (CHJP 79) activates CHJQ-2 and CHJQ-3 only.

The functions performed by the ending pulse and the conditions under which they occur are listed under headings 4-2 through 4-8. Headings 4-2 through 4-5 assume normal continuous operations and ignore the results of operation in noncontinuous or no-overlap modes. Headings 4-6 and 4-7 discuss those functions affected by noncontinuous and no-overlap. Heading 4-8 summarizes the functions of the ending-pulse-storage FF. The functions

are grouped according to the relevant CHJQ line, enabling the difference between the functions of the normal and unconditional transfer ending pulses to be ascertained.

4-2. ENDING PULSE FUNCTIONS FOR NORMAL CONTINUOUS OPERATION

The ending-pulse summary here is based primarily on normal full overlap of instruction execution where the instruction in which the ending pulse occurs is designated as n. In this and the following headings refer to the simplified logic diagram of figure 3-1, which shows the generation of most of the function control signals associated with the ending pulse.

4-3. ENDING PULSE CHJQ-2

Signal CHJQ-2 performs the following functions for instruction n if operand—MNB FF1 is set (signal CHNB, real or simulated) during the same program counter stage:

(1) Clears IR2 (FS313, 314)

(2) Clears program counter to zero (CPKA)

(3) Sets ending-pulse-storage FF if status-1 FF ($\overline{\text{CQBA}}$) is not set (reference heading 4-8).

If the ending pulse occurs during a cycle in which an operand memory reference is made and the MNB is not received, then IR2 and the program counter are not cleared and a delay in the sequencing results (reference heading 3-7).

The functions of CHJQ-2 are blocked under certain conditions. These conditions result in setting the block-not-busy FF (CHXA) which, in turn, blocks CHNB and thereby blocks the normal functions of CHJQ-2. These conditions are:

(1) B-register conflict for n detected by equality at t3 or t7 (reference heading 3-13).

(2) Control transfer if n - 1 was a conditional transfer (part II).

(3) Control transfer if error or contingency in n - 1.

4-4. ENDING PULSE CHJQ-3

Signal CHJQ-3 performs the following functions if operand-MNB FF1 is set:

(1) Calls n + 2 if MNB on call for n + 1 (FS363, 365). If the memory is busy on call for n + 1, ending pulse for n still generates FS363, 365 but this is superfluous since repeat call for n + 1 is made by call FF. Also, calls M (M is transfer-of-control address) where n is an unconditional control transfer instruction (part II).

(2) Generates FS432 if status-1 FF is set (CQBA), indicating MNB on call for n + 1. FS432 performs the following functions:

(a) Sets call FF which, in turn, sets completing-call FF (CSCL) if MNB on call for n + 2

(b) Sets B-mod FF (CSBM) which controls B-modification for n + 1 and puts modified M address and I and A parts of IR1 in IR2 (FS311, 312, 400, 410, 422, 425). Signal CSBM also alerts the equality/inequality FF's (CSBG, CSBS) for B-register comparison operations for n + 1

(3) Sets ending-pulse-delay FF (CSPA) for n. Signal CSPA performs the following functions:

(a) Sets status-3 FF (CQBC) to indicate n is in last cycle of execution time (reference heading 5-4)

(b) Sends ending pulse to AU (CSPB) to clear AU instruction decoder, program counter, and static control FF's and for synchronization check with AU ending pulse

(c) Provides pulse (CSPC) to stall FF circuits (RDF) to indicate computing unit is not stalled.

4-5. ENDING PULSE CHJQ-1

If the MNB signal is not received on the operand call for instruction n, the instruction is retained in IR2 and the ending pulse remains on. (This condition is applicable only if CHJQ-1 occurs during a cycle in which a true memory reference is made; that is, in 4-microsecond single-precision and 8-microsecond double-precision memory reference instructions). If, then, the instruction MNB signal is received on the call for n + 1 two or more cycles before the MNB signal is received on a repeat operand call for n, the retained ending pulse is gated with the reset output of the completing-call FF (signal CSCC) to give C1 + 0 to B-adder (FS401, 411, and no unit add). This action permits the call for n + 2 to be made when the operand-MNB signal for n is received. See timing diagram in figure 3-9.

These functions of CHJQ-1 are required in two other cases:

(1) Where n is a long instruction and CHJQ-1 occurs two or more cycles after the MNB signal is received on the call for n + 1 (figure 3-11).

(2) Where the MNB signal is not received on the call for n + 1; but this is superfluous since, in this case, the call FF (normally reset by the completing-call FF) performs the same function (figure 3-10).

Signal CHJQ-1 is not generated during unconditional control transfer instructions for reasons which are given in part II.

4-6. ENDING PULSE FUNCTIONS FOR NONCONTINUOUS OPERATION

In noncontinuous operation the computing unit stops when an ending pulse occurs. The stop function is produced if the stop FF is set while in continuous operation, or if the computing unit is being operated in either the one-instruction or multivibrate mode. (The details of the logic and timing of noncontinuous operation are covered in section 7.) Two noncontinuous-operation signals, CNEA and CNEB, (generated by gating the stop FF signal (CNBB) and the multivibrate FF signal (CNBF); shown in figure 7-1) are used to prevent further sequencing of instructions by blocking three of the normal functions of the ending pulse as shown in figure 3-1. The normal functions that are blocked are:

(1) Call for instruction n + 2 (no FS363, 365)

(2) B-modification of n + 1, and control counter step to n + 3 (no FS432)

(3) Setting of ending-pulse-storage FF since, if memory is busy on call for n + 1, this call is repeated by call FF. Thus, if the ending-pulse-storage FF were set it would call n + 2 and again set the call FF (when status-1 FF is set for n + 1).

The other normal functions of the ending pulse (clearing of IR2 and program counter, and setting of ending-pulse-delay FF) are still carried out in noncontinuous operation.

4-7. ENDING PULSE FUNCTIONS FOR NO-OVERLAP MODE

The no-overlap mode is combined with either a continuous or noncontinuous operation in which a new instruction is called only at 20-microsecond intervals, assuming minimum-length instructions and no memory reference conflicts. (The details of the logic and timing of no-overlap operation are described under headings 7-6 to 7-8.) The only ending pulse function directly inhibited by the no-overlap signal (ECDCS - from switch on engineer's console) is the call for a following instruction; that is, the generation of FS363 by the ending pulse is blocked (figure 3-1). However, ECDCS also blocks any instruction call by the ending-pulse-storage FF, and blocks the setting of the call FF by FS432.

The effective functions of the ending pulse in no-overlap operation, therefore, are:

(1) Clearing IR2

(2) Clearing program counter

(3) Setting ending-pulse-storage FF (since status-1 FF is not set because next instruction was not called) in order to set the B-mod FF for next instruction, when status-1 FF is again set.

(4) Gating C1 + 0 to B-adder (since completing-call FF is not set), but the B-adder output is not used and this is superfluous.

It should be noted that since in no-overlap operation all the normal means of making the call for a new instruction are blocked, special provisions must be made. Thus, signal ECDCS is gated with the absence of the status-1 FF signal (CQBA) and the status-4 FF output (CQBD-reference heading 5-4) to set the call FF which gives C1 + 0 to the B-adder and B-adder to the address lines (figure 3-1). This gate is inhibited, however, whenever the computing unit is operation in the multivibrate or one-instruction mode (reference heading 7-8).

4-8. ENDING PULSE STORAGE FLIP-FLOP

The ending-pulse-storage FF (signal CSAR) is used to provide a substitute ending pulse when conditions are such that the normal ending pulse (CHJQ-3) is either not present or is prevented from maintaining sequencing. Under conditions of normal overlap the ending pulse for an instruction n controls the B-modification for n + 1 and the call for n + 2. If n + 1 is subject to a delay because of failure to receive the MNB signal (or because n is an unconditional control transfer; part II), then the ending pulse for n is unable to initiate the B-modification of n + 1. Consequently, the ending pulse is stored so as to initiate the B-modification of n + 1 and the call for n + 2 when n + 1 is finally received.

Referring to figure 3-1, if the MNB signal (CGNB) is not received on call for n + 1, the completing-call FF will not be set which, in turn, prevents the status-1 FF from being set. Thus, signal CQBA is not generated and its inhibiting effect is removed from the setting gate of the ending-pulse-storage FF. The call FF then makes repeat call(s) for n + 1 until the MNB signal is received. The status-1 FF is then set to give CQBA which gates with the output of the ending-pulse-storage FF at t2 (signals CSAD and CSAE) to control call for n + 2 (FS363) and B-modification for n + 1 (FS432). The ending-pulse-storage FF is reset by FS432 (via signal CSBK); also, it may be prevented from being set by the noncontinuous operation signal (CNEA) as described under heading 4-6. (The reference timing diagram is in figure 3-10).

If n + 1 is received without delay but is subject to a B-register conflict, then an incorrect B-modification is initiated by the ending pulse of n but n + 1 is retained in IR1 and the B-modification must be repeated. Therefore a substitute ending pulse is provided by setting the ending-pulse-storage FF to control this repeat B-modification.

In this case signal CSAV (figure 3-1) is generated to set the ending-pulse-storage FF. It also sets the block-not-busy FF which, in turn, blocks the setting of operand-MNB FF1 so that signal CHNB is not generated. Hence, signal CQBA is gated with CSAR instead of with the normal ending pulse and CHNB to control the repeat call for n + 2 and the repeat B-modification of n + 1. (Reference timing diagram figure 3-14.)

The conditional transfer instructions, when they transfer control, require the ending pulse substitute to enable the B-modification of the first instruction obtained from the transfer-of-control (M) address, and also to enable the call for the next instruction, M + 1. The details of these operations are given in Instruction Sequence Control, Part II.

The substitute ending pulse is required to control the B-modification of the first instruction executed when starting, since there is no previous instruction in IR2 to provide an ending pulse. If starting with IR1 empty, the ending-pulse storage also controls the call for the next instruction. Thus the start signal (CNBA) gives CSAR to control the B-modification of instruction n and the call for n + 1 (reference figure 2-5).

When starting after a stop, with instruction n already in IR1, CSAR controls the B-modification for n but does not call n + 1. (The address of n + 1 is not available at the output of the B-adder at this time and the generation of FS363 by CSAR and CQBA is blocked by CNBA; figure 3-1.) In this case the call FF makes the call for n + 1.

When the computing unit is operating in the no-overlap mode, the ending-pulse-storage FF is needed to give FS432 in order to control the B-modification for the next instruction. This is similar to the case of failure to receive the MNB signal for n + 1; that is, the ending pulse is prevented from performing normal sequence control functions and is therefore stored until the next instruction is received in IR1.

SECTION 5
INTRODUCTION TO STATUS CONTROL

5-1. STATUS CONTROL

The status-1 FF, which has been dealt with extensively throughout this manual, stores information concerning the early phases of the execution of an instruction. This flip-flop is one of a set of four status-control flip-flops which are used to monitor the progress of an instruction through the various stages of the instruction cycle in the control unit. The status-control flip-flops, which are designated as status-1, status-2, status-3, and status-4, are normally set by appropriate signals in a sequential fashion for any one instruction beginning with the B-modification portion of the instruction cycle. The time during which each flip-flop is set corresponds roughly to the instruction stage as follows:

Status-1 — B-modification

Status-2 — Operand select

Status-3 — Execute

Status-4 — Result

Except for status-1, these flip-flops are not used to any great extent in normal sequence control, as covered in this manual. The status flip-flops as a group, however, indicate the overlap status of the instruction sequencing; that is, they indicate how many instructions are in the control unit at any one time and the status of each. For example, whenever the normal sequence of instructions is interrupted either by a programmed transfer of control or by a transfer to the error or contingnecy routines, the point at which the interruption occurs is automatically recorded for future reference by the program. The combination or status flip-flops which are set enables the control unit to derive the point at which the interruption occurred, from the control counter reading. These features of status control are discussed more fully in Instruction Sequence Control, Part II.

The remaining parts of this section deal with each status flip-flop individually and summarize the general function of each in conjunction with the various signals required for correct sequence control. Figure 5-1 is a

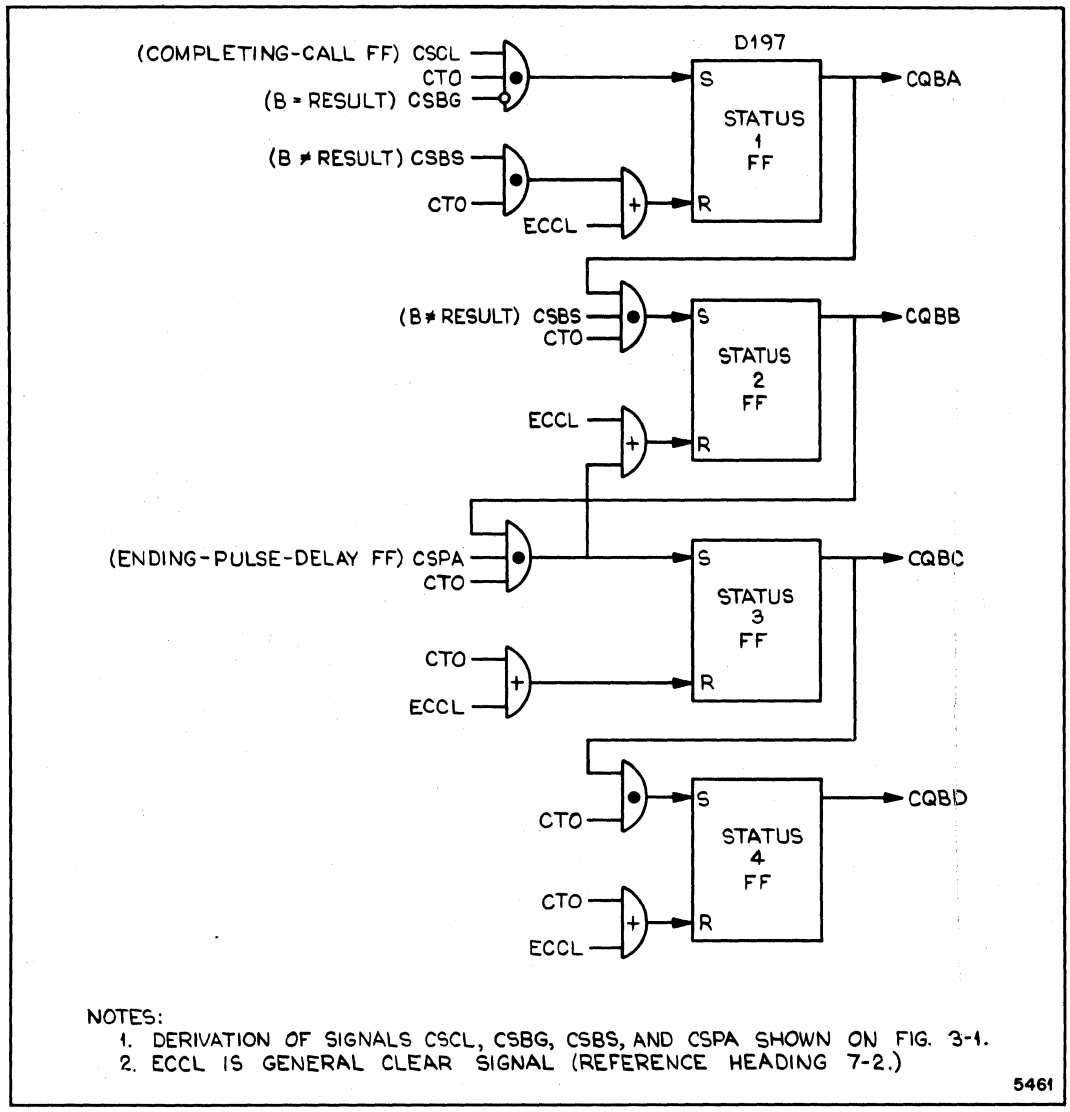


Figure 5-1. Simplified Logic Diagram - Status Control Flip-Flops

simplified logic diagram showing the set-reset conditions for each flip-flop as related only to the signals described in the preceding sections of this manual; a general timing diagram for a 4-microsecond instruction with no delays is given in figure 5-2.

5-2. STATUS-1 FLIP-FLOP

The status-1 FF, when set (signal CQBA), indicates that the call for a new instruction (n) was successful and that the instruction is either already in IR1 or is being gated to IR1 at this time by FS320 (resulting from the setting of the completing-call FF, signal CSCL). Signal CQBA is used to control the B-modification of n (in conjunction with either the ending pulse of the previous instruction (n - 1) or with the ending-pulse-storage FF if it has been necessary to delay the B-modification of n due to missing instruction-MNB signal(s) or a B-register conflict). It is also used to control the call for n + 1 in cases where the B-modification of n is delayed beyond the ending pulse of n - 1.

The status-1 FF is set for a time roughly equivalent to the B-modification phase of the instruction cycle. The B-modification time is not variable from one instruction to another but may be delayed by a missing operand-MNB signal for the preceding instruction or by a B-register conflict for the instruction in IR1. The time that the flip-flop is set, then, is equal to the time that an instruction is in IR1 which is 4 microseconds if there is no operand-MNB delay, plus an additional 4 or 8 microseconds delay if there is a B-register conflict. When a B-register conflict occurs, an incorrect B-modification is performed and the instruction moves up to IR2. However, it is also retained in IR1 and the status-1 FF remains set for the repeat B-modification.

The status-1 FF is reset by gating the inequality FF signal CSBS (indicating no B-register conflicts, or that a conflict has been resolved) with CTO. However, if operating in full overlap with no delays, the reset signal is overridden by the simultaneous set signal resulting from the completing-call FF output for the next instruction.

5-3. STATUS-2 FLIP-FLOP

The status-2 FF is set (CQBB) during the operand-select time for an instruction (corresponding to the time the instruction is in IR2) which lasts for one or more cycles depending on the instruction and on delay conditions. The operand-select time varies with the length of the instruction and includes all of the time that an instruction is in the AU except the last 4 microseconds. The time an instruction is in IR2 is equal to the nominal execution time plus any delay time resulting from missing operand-MNB signals.

The status-2 FF is normally set by gating the output of the status-1 FF (CQBA) with the inequality signal (CSBS) and CTO. Signal CSBS indicates that no B-register conflict exists and that the correct B-modification for the instruction, now in IR2, is complete. When a B-register conflict occurs, the instruction is transferred to IR2 while also being retained in IR1 and the B-modification is repeated to obtain the correct modified M address. However, the status-2 FF is not set during the time that the instruction is in IR2 prior to the correct B-modification.

The status-2 FF is normally reset by gating the output of the ending-pulse-delay FF (signal CSPA) with CTO. As shown in figure 5-2, signal CSPA is active for 4 microseconds (t6 - t5) following the ending pulse (CHJQ-3) for the instruction in IR2. The significance of CSPA is that the control unit operations for the current instruction are complete and that only the execute and result phases of the instruction cycle remain.

5-4. STATUS-3 AND STATUS-4 FLIP-FLOPS

The status-3 FF (CQBC) is set by the same signal (CSPA) that resets the status-2 FF, and always remains set for 4 microseconds corresponding to the last cycle of the execution of the instruction; that is, the execute time is always considered as the last 4 microseconds of the time that the instruction is in the AU regardless of the length of the instruction. Status-4 (CQBD), in turn, is set for 4 microseconds by CQBC with the next CTO corresponding to the result phase of the cycle. (The result phase is

the final result cycle in a double-precision instruction and the time in which the first half of the result is stored is not considered as part of result time.) Thus, whenever the instruction control cycle reaches the execute phase, the result phase automatically follows 4 microseconds later to complete the cycle.

Notice in figure 5-1 that each one of the status-control flip-flops is reset by the general-clear signal ECCL (reference heading 7-2).

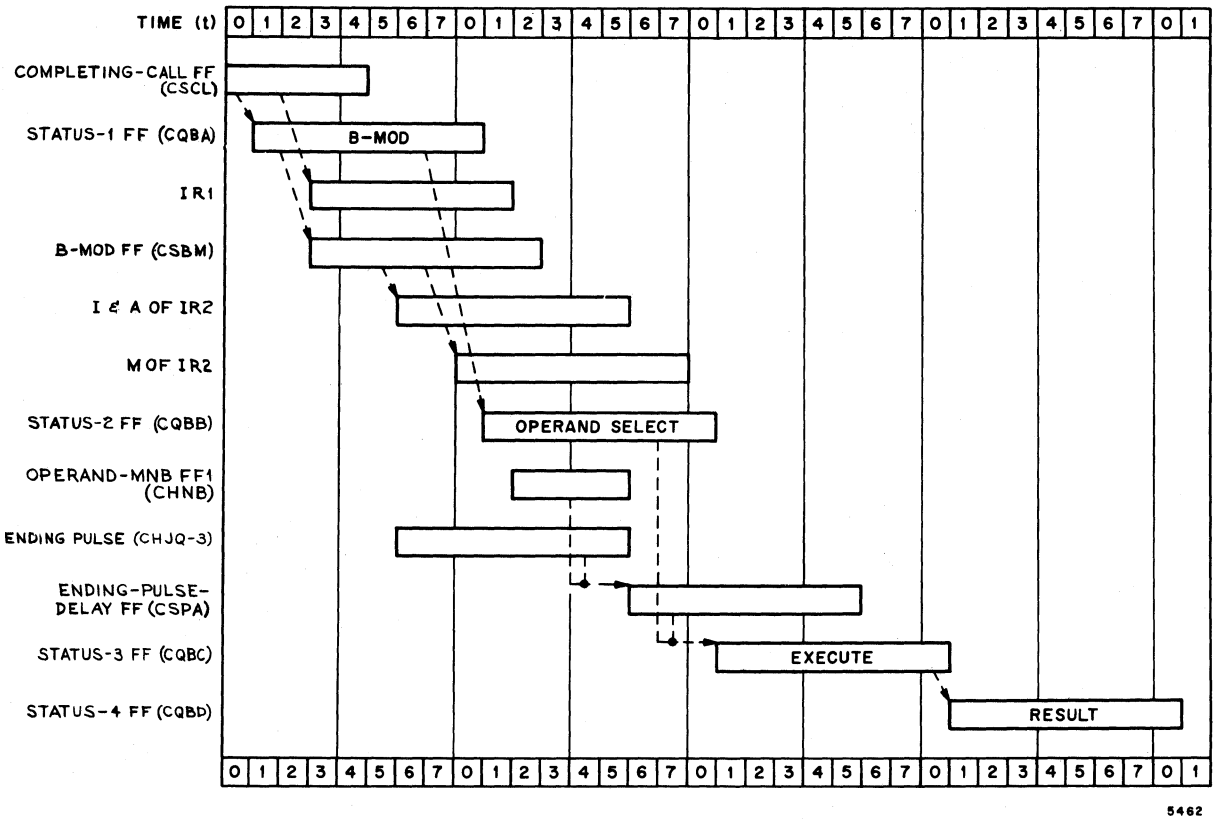


Figure 5-2. Timing Diagram - Status Control

SECTION 6
CLASSIFICATION OF INSTRUCTIONS ACCORDING TO CONTROL FEATURES

This section summarizes the differences between the various instructions in the computing unit repertory insofar as control unit operations are concerned. The previous sections describe the basic sequence control using the add instruction as a representative example. Most of the instructions are similar in sequence control requirements and make use of some combination of the features already discussed. For example, some instructions require both A and M operands for their execution while others require one or the other. Similarly, the use of all three fast register chain controls is required in some instructions while others use only one or two of these.

Instructions which are related in terms of sequence control are not necessarily similar in arithmetic control requirements, and vice versa. The main instruction lines therefore go to two separate group encoders; one is specific to the control unit and the other to the arithmetic unit. Instructions are grouped in each encoder according to identity of basic control requirements in the corresponding unit.

In the control unit group encoder, the 76 instruction lines (one CDFG line for each of the 76 instructions in the computing unit repertory) are arranged in 49 groups in which many of these groups differ only slightly in control unit operations. These groups, in turn, are further encoded into 90 lines (CHSA lines) corresponding to the program-counter stages a associated with each instruction. The CHSA lines go to an encoder where each activates several of the group of 86 chain-start lines (CHJP 01-87) associated with specific functions or sequences of functions. In some cases the CHJP\line is gated directly with a timing signal to generate a specific function signal or signals. In other cases final function encoding is conditional on signals derived from other occurrences in the instruction sequence control.

The complete analysis of CHJP functions is given in the manual "Computing Unit: Instruction and Function-Signal Analysis". It is intended here only to present a very general classification of instructions according to basic control operations; that is, with regards to the CHJP lines that activate the fast register chain control logic (reference heading 3-1), and the need for selecting operands from fast registers or memory (reference heading 3-4) to carry out the execution of an instruction.

Table 6-1 classifies the instructions according to the control features mentioned above. The table indicates the selection of A and/or M operands and the relevant chain control operation for each instruction class. These operations correspond to the basic instruction sequence control described in the preceding sections of this manual. Thus, the table summarizes the general control operations associated with all instructions in the repertory as related to the earlier descriptions.

All arithmetic operations are similar in control requirements to the basic 4-microsecond add order (heading 2-1) and some are identical. Other instructions contain minor variations on this, from a sequence control point of view. For example, a single-precision multiply or divide order requires one A operand and one M operand and therefore uses the A-input chain and the result chain. However, there is a difference in the elapsed time between the input and result during which the control unit effectively "marks time" while operations are carried out in the arithmetic unit.

Double-precision arithmetic instructions require two successive references to the memory and two uses of the A-input chain followed by two uses of the result chain (reference heading 3-10). Again, however, there are variations in the elapsed time from one instruction to another. Also, in the case of double-precision multiply instructions there is a total of four uses of the A-input chain for reasons peculiar to the nature of multiplication operations in the arithmetic unit.

The extract instructions are similar in control requirements to the arithmetic instructions except that the extract-lower and extract-upper instructions (65 and 66) also use the M-input chain to supply the second A-operand input from the fast registers to the AU.

Shift and conversion instructions contain a major variation in that they transfer the two least significant digits from the M part of IR2 directly to the AU (register AH) in order to provide control information concerning the shift and scale factors. This is done with FS470 which is generated by gating control signal CHJP 42 with CT5. In addition, these instructions, as well as the comparison instructions, use the M-input chain for the normal A-operand input to the AU. As indicated in table 6-1, instructions which use the M-input chain in place of the A-input chain transfer the A part of IR2 to the fast register selector register directly instead of by way of the B-adder. This is done with FS430 which is generated by gating control signal CHJP 03 or 04 with CT0.

The fetch instructions (43 and 48) simply transfer the M operand to a fast register using the result chain as in arithmetic operations. Store instructions (40-42 and 45-47), as well as two of the visual display instructions (29 and 39), use the M-input chain to transfer the A-operand to the memory (reference heading 3-4). The use of the M-input chain for handling the A operand is similar to the case for shift, conversion, and comparison instructions.

In the case of instructions which provide for conditional transfer of control, there is no change in the basic pattern of sequence control when the transfer does not take place.

Many of the instructions classified in table 6-1 have not been discussed in detail in this manual. These instructions, which involve special

control requirements in addition to features in common with other instructions, are covered in Instruction Sequence Control, Part II and include the store, control transfer, and visual display instructions. Also, several instructions are not included in the general classification of the table because they do not use operands or chain control logic. These instructions are skip, unconditional control transfers (90 and 91), flip-flop instructions, and stop.

Table 6-1. Classification of Instructions

Instruction Class	Numeric Code	Operands		Chain Control			Remarks
		A	M	A Input	M Input	Result	
Arithmetic	01-06, 11, 12 14-16, 20-27, 30-32, 34-37	X	X	X		X	Includes all arithmetic operations (single-precision, double-precision, fixed-point, floating-point). Uses M-input chain if M operand is a fast register.
Extract	60-64	X	X	X		X	Uses M-input chain if M operand is a fast register.
Extract	65, 66	X	X	X	X	X	Uses M-input chain for second A-operand input to AU.
Shift and Conversion (Single-precision)	50-53	X			X	X	Uses M-input chain for A-operand input to AU.* Two LSD's transferred directly from M of IR2 to AU.
Shift and Conversion (Double-precision)	55-59	X		X	X	X	Uses M-input chain for first A-operand input to AU.* Uses A-input chain for second A-operand input to AU. Two LSD's transferred directly from M of IR2 to AU.
Data Transfer (Fetch, M ⟶ A)	43, 48		X			X	Uses M-input chain if M operand is a fast register.
Data Transfer (Store, A ⟶ M)	40-42, 45-47	X			X		Uses M-input chain for transfer to HSB and to provide for A-register conflict check.* Uses result chain if M is a fast register.

* For these operations the A part of IR2 is transferred to the fast register selector register directly instead of by way of the B-adder.

Table 6-1. Classification of Instructions (cont)

Instruction Class	Numeric Code	Operands		Chain Control			Remarks
		A	M	A Input	M Input	Result	
Store Last Jump (C2 ⟶ M)	93						Uses result chain if M is a fast register; contents of control counter 2 transferred to M input of AU by way of B-adder.
Unconditional Transfer (C1 ⟶ A and M ⟶ C1)	92	X		X		X	Uses A-input chain to transfer A operand to AU to allow contents of control counter 1 to be inserted in 5 LSD. Contents of control counter 1 transferred to M input of AU by way of B-adder.
Conditional Transfers:							These instructions provide for M ⟶ C option—ref. Instruction Sequence Control, Part II.
Comparison	70, 71, 75, 76	X		X	X		Uses M-input chain for A-operand input to AU.* Uses A-input chain for A + 1 operand input to AU.
Comparison	72-74	X			X		Uses M-input chain for A-operand input to AU.*
Index	80-83, 85, 86	X		X		X	Uses A-input chain for B-register modification. Instructions 85 and 86 do not contain M ⟶ C option.
Visual Display	09, 19		X			X	M operand obtained from visual display register (5- or 12-digit) by way of memory location 02650.
Visual Display	29, 39	X			X		A operand transferred to visual display register (5- or 12-digit) by way of memory location 02650. Uses M-input chain for transfer to HSB and to provide for A-register conflict check.*

* For these operations the A part of IR2 is transferred to the fast register selector register directly instead of by way of the B-adder.

The following instructions do not require A or M operands or use fast register chain controls: 00, 90, 91, 95-97, 99.

SECTION 7
NONCONTINUOUS AND NO-OVERLAP OPERATIONS

7-1. NONCONTINUOUS OPERATION

The computing unit is capable of operating in four different modes. The normal mode of operation is the "continuous" mode in which the sequencing of instructions is carried out in an uninterrupted manner and where the execution of succeeding instructions is overlapped in time. Maintenance of the system is facilitated, however, by operating the computer in a noncontinuous manner so that the analysis and isolation of faults can more readily be determined. The noncontinuous modes of operation are the "one-instruction" and "multivibrate" modes.

In the one-instruction mode, the computer executes a single instruction while bringing a second instruction into IR1 and then stops automatically. In the multivibrate mode the computer operates in a similar manner, except that the computer can automatically restart and stop alternately, at a manually preselected repetition rate.

The fourth mode, the "no-overlap" mode, is used in conjunction with any of the other three modes, but is also used for test purposes.

Six pushbutton switches are included on the engineer's console for manually starting, stopping, and selecting the mode of operation of the computing unit, as indicated in figure 7-1. These switches are:

Start	Multivibrate
Stop	One Instruction
Continuous	No Overlap

Start and stop switches are also included on the operator's console. Once the computer is stopped, it is normally started again by generating the start signal. Headings 7-2 through 7-5 discuss the logic and timing associated with starting and stopping the computing unit and the noncontinuous modes of operation. Headings 7-6 through 7-8 cover the various no-overlap modes of operation.

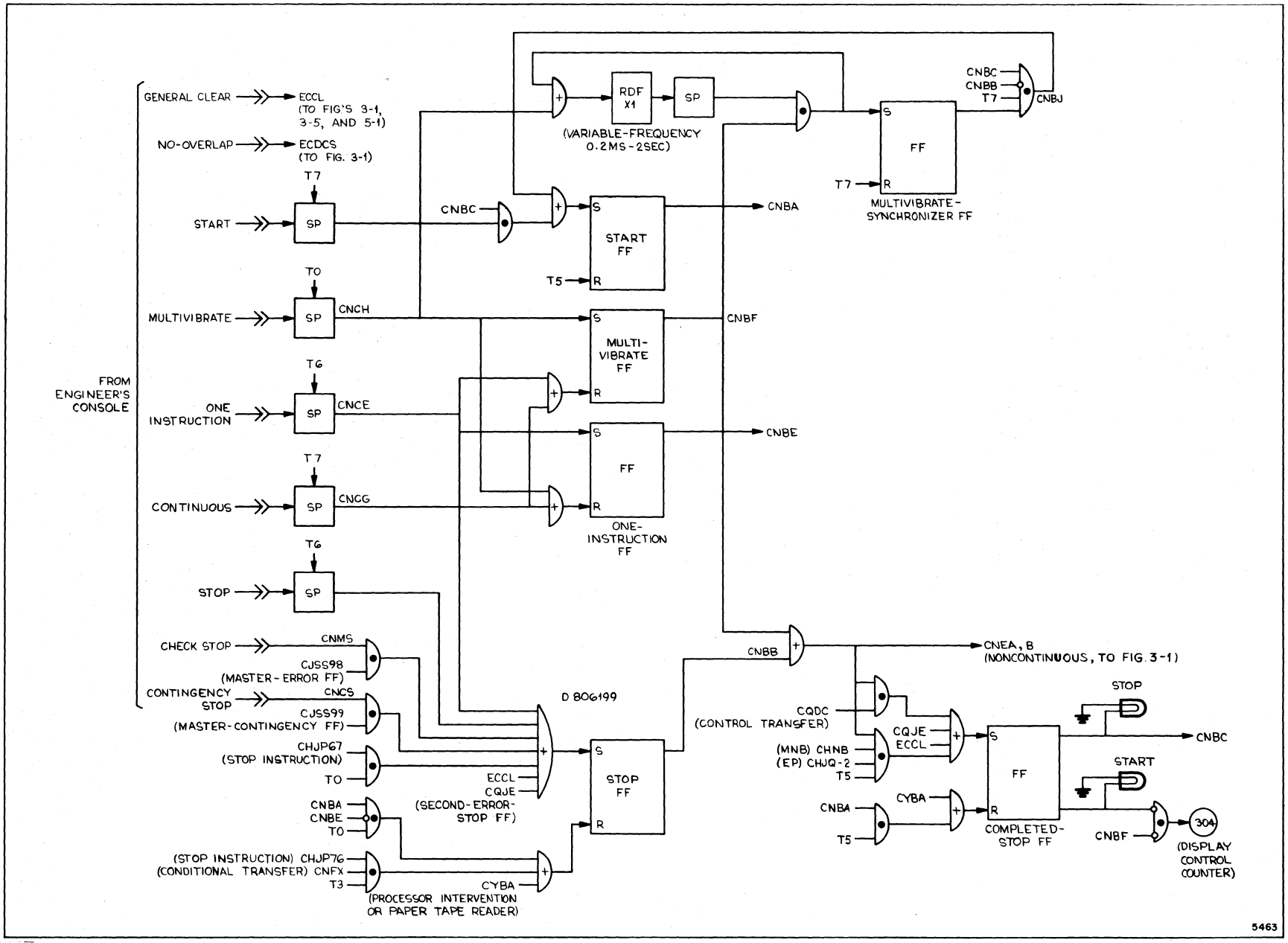


Figure 7-1. Simplified Logic Diagram - Noncontinuous Operation Controls

7-2. START AFTER GENERAL CLEAR

The starting of a new operation in the computing unit is usually preceded by a general clearing of all key control elements in the control and arithmetic units. This action is implemented by the general clear signal (ECCL) which is generated by closing the general clear (GEN CLEAR) push-button switch on the engineer's console. Signal ECCL stops the computing unit and resets several flip-flops to insure that the new operation starts under initial sequencing conditions. The functions performed in the control unit by ECCL are:

(1) Setting stop FF

(2) Setting completed-stop FF

(3) Resetting of all status-control FF's

(4) Resetting ending-pulse-storage FF

(5) Setting block-not-busy FF

(6) Resetting error-delay and error-stop FF's

(7) Clearing IR1 and IR2

Performing the above functions renders the control unit completely devoid of any previous instruction sequencing processes.

A new operation is normally started by first setting the start FF by means of the start switch. For continuous operation, the continuous switch (CONT.) is closed to reset the one-instruction and multivibrate FF's (reference headings 7-4 and 7-5). As shown in figure 7-1, the start signal from the engineer's (or operator's) console is passed through a single-pulser circuit (SP) to produce a synchronized 0.5-microsecond pulse at t7. The start FF is then set to produce signal CNBA at t0 which initiates the instruction sequence control operations. The functions of CNBA are:

(1) Resetting stop FF (if not in one-instruction mode) and completed-stop FF

(2) Setting ending-pulse-storage FF and call FF

These actions initiate the instruction request cycle as described previously under heading 2-8. Refer to timing diagram in figure 2-5.

The computing unit may also be started when the processor-intervention FF is set (processor interrupting computing unit program), or by a signal derived from the paper tape reader controls. These conditions are indicated by signal CYBA which resets the stop and completed-stop FF's. (Processor intervention is discussed in book on Error and Contingency Control; the tape reader controls in book on Manual and Display Controls.)

7-3. STOP IN CONTINUOUS MODE

The stopping of the computing unit requires the setting of the stop FF which, in turn, is followed automatically by the setting of the completed-stop FF. When the completed-stop FF is set, it lights the STOP lamp on the control console (indicating the computing unit is stopped) and, if not in multivibrate mode, allows the generation of FS304 which transfers the contents of control counter 1 to its display register. In the reset condition, the completed-stop FF lights the START lamp on the control console. Figure 7-1 shows the various ways in which the stop FF may be set. These are:

(1) Manually, by pushing the stop, one instruction, or general clear pushbuttons.

(2) Stop instruction (99 order produces control signal CHJP 67).

(4) Setting master-error FF (CJSS98) or master-contingency FF (CJSS99), if respective stop-option switch has been selected.

(5) Setting second-error-stop FF (CQJE). This flip-flop is set when an error occurs while the computing unit is already on the error routine (master-error FF set).

The output from the stop FF (CNBB) is buffed with the multivibrate FF output (CNBF) to produce the non-continuous signals (CNEA, B). Signal CNEA then sets the completed-stop FF (CNBC) when the ending pulse (CHJQ-2) and operand-MNB signal (CHNB) occur for the current instruction. Signals CNEA, B are used to prevent further sequencing of instructions by blocking three of the normal functions of the ending pulse as described previously under heading 4-6. The next instruction in sequence following the stop is retained in IR1. A timing diagram of these operations is given in figure 7-2 where it is assumed that the stop switch is closed after instruction n enters IR1. Notice that the stop FF is set at t7 of the first cycle by synchronizing the stop signal-pulser circuit with a CT6 timing signal. The completed-stop FF is then set at the following t6. If the memory is busy on the call for the next instruction (n + 1), the call FF repeats the call until the instruction is received.

When the computing unit is stopped in the continuous mode due to the setting of the stop FF, the start signal is usually provided by pushing the start button. The asynchronous start signal is synchronized with CT7 and gated with CNBC to set the start FF (CNBA) at t0. See timing diagram in figure 7-3 which is a continuation of figure 7-2. Signal CNBA then resets the stop and completed-stop FF's while setting the ending-pulse-storage FF (CSAR) to enable the resumption of continuous operation (reference preceding heading). The instruction retained in IR1 (status-1 FF set) along with CSAR produces FS432 to set the call FF for n + 2, the B-mod FF for n + 1, and so on.

As shown in figure 7-1, another means for resetting the stop FF is provided to take care of the case where a stop instruction follows a conditional transfer order. If the transfer of control takes place, signal CNFX is gated with the stop instruction control signal (CHJP 76) to reset the stop FF and thereby cancel the effect of the stop order. (Reference Instruction Sequence Control, Part II.)

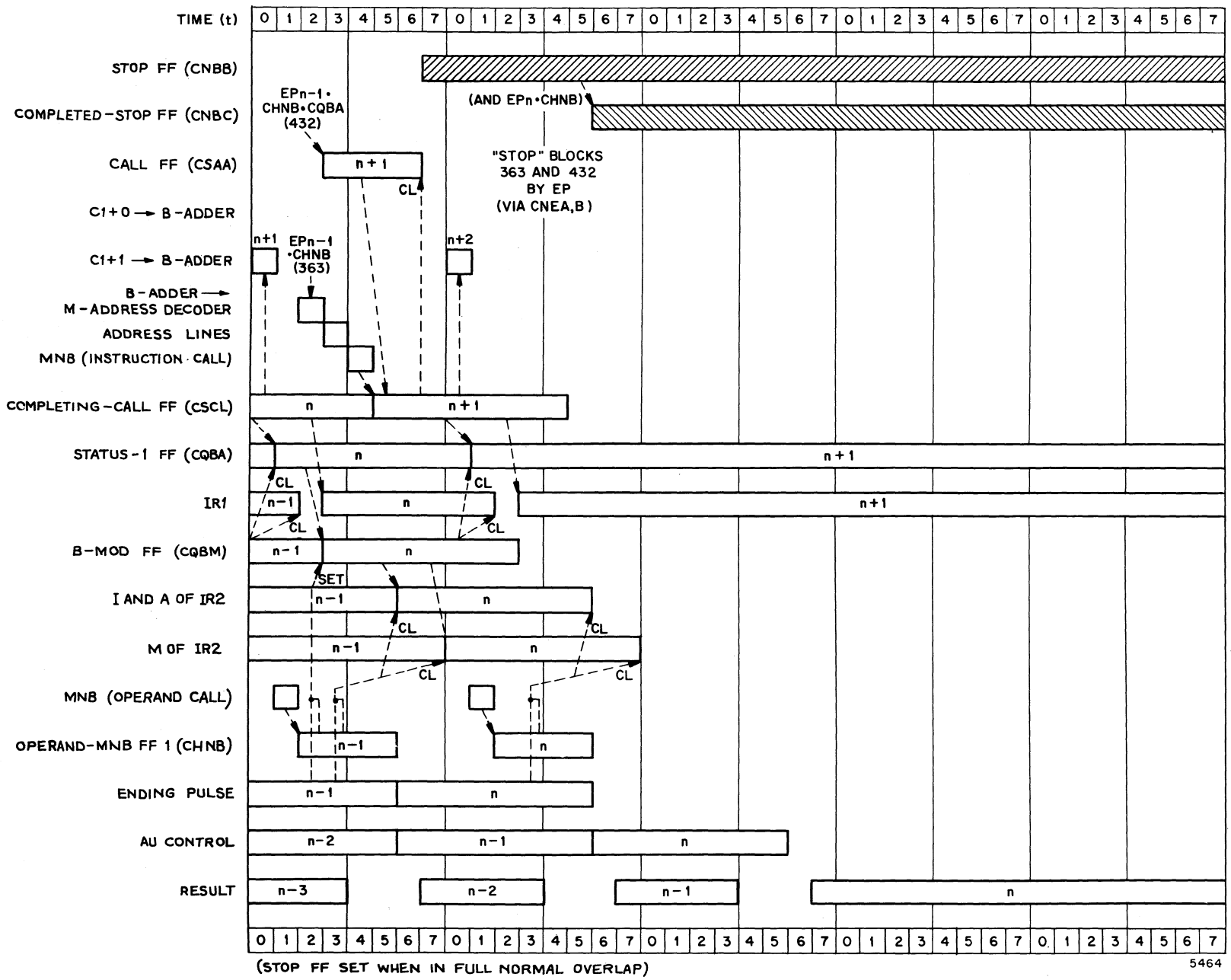
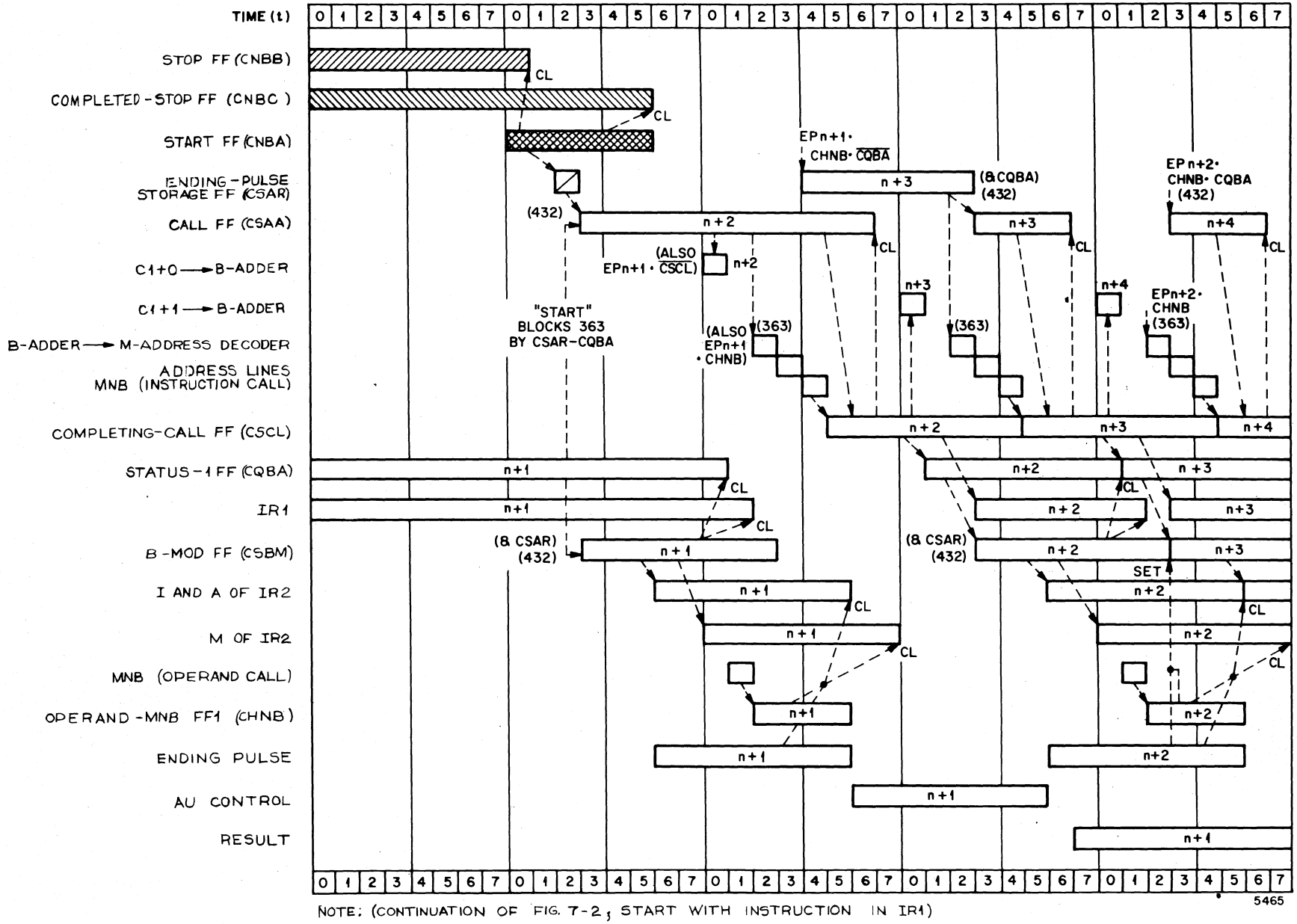


Figure 7-2. Timing Diagram - Stop in Continuous Mode



NOTE: (CONTINUATION OF FIG. 7-2, START WITH INSTRUCTION IN IR1)

Figure 7-3. Timing Diagram - Restart After Stop

7-4. ONE-INSTRUCTION MODE

The one-instruction mode of operation processes a single instruction through to completion (while bringing the next instruction in sequence into IR1) and then stops. The one-instruction mode is entered by closing the one-instruction (O.I.) switch which produces a signal synchronized to t6 (CNCE) that sets the one-instruction FF and the stop FF (figure 7-1). The setting of the stop FF then produces the same interruption to the sequencing (via signals CNEA, B) as described under the preceding heading for a stop operation; that is, if the one-instruction mode is selected while operating in the continuous mode the computing unit stops exactly as if the stop switch had been closed, as shown in figure 7-2.

Only two functions are performed by the output of the one-instruction FF (CNBE). One is to prevent the resetting of the stop FF by the start signal (CNBA) when the computing unit is restarted. This action enables additional one-instruction operations to be carried out consecutively each time the start switch is closed to set the start FF. The other function of CNBE is to block the setting of the call FF by the no-overlap signal (ECDCS, figure 3-1) when operation in the no-overlap mode (reference heading 7-8).

Starting in the one-instruction mode, with an instruction in IR1, corresponds to the situation shown in figure 7-3 except that the stop FF is not reset by the start signal at t0. Therefore, instruction n + 1 (in IR1) is completed and n + 2 is called and brought into IR1 and the computing unit again stops. The B-modification for n + 2 and the call for n + 3 are prevented by blocking the setting of the ending-pulse-storage FF (via CNEA) during the third cycle shown in figure 7-3. To return to continuous operation, the continuous switch is closed to produce signal CNCG which resets the one-instruction FF.

7-5. MULTIVIBRATE MODE

The multivibrate mode operates in the same way as the one-instruction mode except that the restart feature is produced automatically. Thus, the need for pushing the start button is eliminated and the computing unit executes single instructions at a controlled rate. The frequency with which instructions are executed can be varied in five steps over a range from 0.2 millisecond to 2 seconds per instruction by means of the FREQUENCY selector on the engineer's console.

As shown in figure 7-1, the multivibrate mode is entered by closing the multivibrate (M.V.) pushbutton switch on the engineer's console to produce signal CNCH which is synchronized to t0. Signal CNCH sets the multivibrate FF (CNBF), resets the one-instruction FF, and triggers a variable RDF unit (X1). The X1 circuit produces an output pulse which is delayed by an amount corresponding to the selected frequency of operation. Signal CNBF produces the noncontinuous signals (CNEA, B) to control the interruption to the instruction sequencing as described under the two preceding headings; that is, a single instruction is completed while the next instruction is retained in IR1. Signal CNBF also blocks the setting of the call FF during no-overlap operations (reference heading 7-8).

The restart feature is produced by gating CNBF with the output of the RDF-X1 circuit to set the multivibrate-synchronizer FF. The output from

this flip-flop is then gated with CT7 and the set output of the completed-stop FF (CNBC) to give the restart signal CNBJ. Thus, the start FF is set to initiate the execution of another instruction each time the RDF circuit recovers from the preceding input.

Notice that signal CNBJ is blocked if the stop FF is set (signal CNBB). Hence when the multivibrate mode is selected while the computing unit is stopped, the RDF is triggered and begins to cycle but no instructions are processed. Therefore it is necessary to push the start button which resets the stop FF and later the completed-stop FF. Setting the start FF allows an instruction to be processed and the next instruction to be called and brought into IR1, but signals CNEA, B (produced by CNBF) prevent further sequencing as in the one-instruction mode. Although the stop FF is reset, the fact that the completed-stop FF is also reset blocks the generation of CNBJ until the one instruction generates an ending pulse and sets the completed-stop FF. A new start signal is then generated and the cycle repeats. The continuous signal (CNCG) resets the multivibrate FF to enable a return to continuous operation.

7-6. NO-OVERLAP MODE

The no-overlap mode may be used in conjunction with any one of the other three modes—continuous, one-instruction, or multivibrate. In each case it has the effect of allowing only one instruction to be processed at a time, and prevents any part of sequencing of following instructions. Thus, if an instruction requires 4 microseconds of execution time in normal overlapped sequencing, the no-overlap operation for that instruction takes a total of 20 microseconds to complete (reference heading 2-1). The no-overlap mode is especially useful in analyzing faults which appear to be due to the overlapping features of the instruction control logic. The timing diagram in figure 7-4 illustrates the timing for a no-overlap operation.

7-7. NO-OVERLAP CONTINUOUS OPERATION

A no-overlap continuous operation is initiated after the computer is stopped by closing the no-overlap (NO OL) switch and the start switch on the engineer's console. The no-overlap switch generates signal ECDCS which is applied to various points in the instruction control logic to prevent the call for the following instruction in sequence until the current instruction is completed. The primary function of signal ECDCS, then, is to block the generation of FS363 which gates the address for a new instruction call to the memory. This blocking is accomplished by using ECDCS to inhibit the following gates (figure 3-1) whose outputs normally produce FS363:

(1) Ending pulse (CHJQ-3) and operand-MNB FF1 (CHNB) — block signal CSAC

(2) Ending-pulse-storage FF (CSAR) and status-1 FF (CQBA) — block signal CSAD

(3) Call FF (CSAA) when set by FS432 — block signal CSAL

(4) Call FF (CSAA) when set by start FF (CNBA) and status-1 FF (CQBA) — block signal CSAL

As shown in figure 7-4, the initial setting of the call FF is made at t3 (with IR1 empty) by gating the no-overlap signal with the status-4 FF output signal (CQBD). (As was described under heading 4-7, ECDCS is always gated with CQBD to provide for succeeding calls in the no-overlap mode.) For this case CQBD is produced incidentally at t2 by the start FF signal (CNBA). Thus, the call FF is set at t3 for the first instruction call although it would also be set at t4 by CNBA directly as in normal continuous operation (figure 2-5). Signal CNBA then sets the ending-pulse-storage FF (CSAR) so that FS432 will be generated (when the status-1 FF (CQBA) is set) to provide for the B-modification operation.

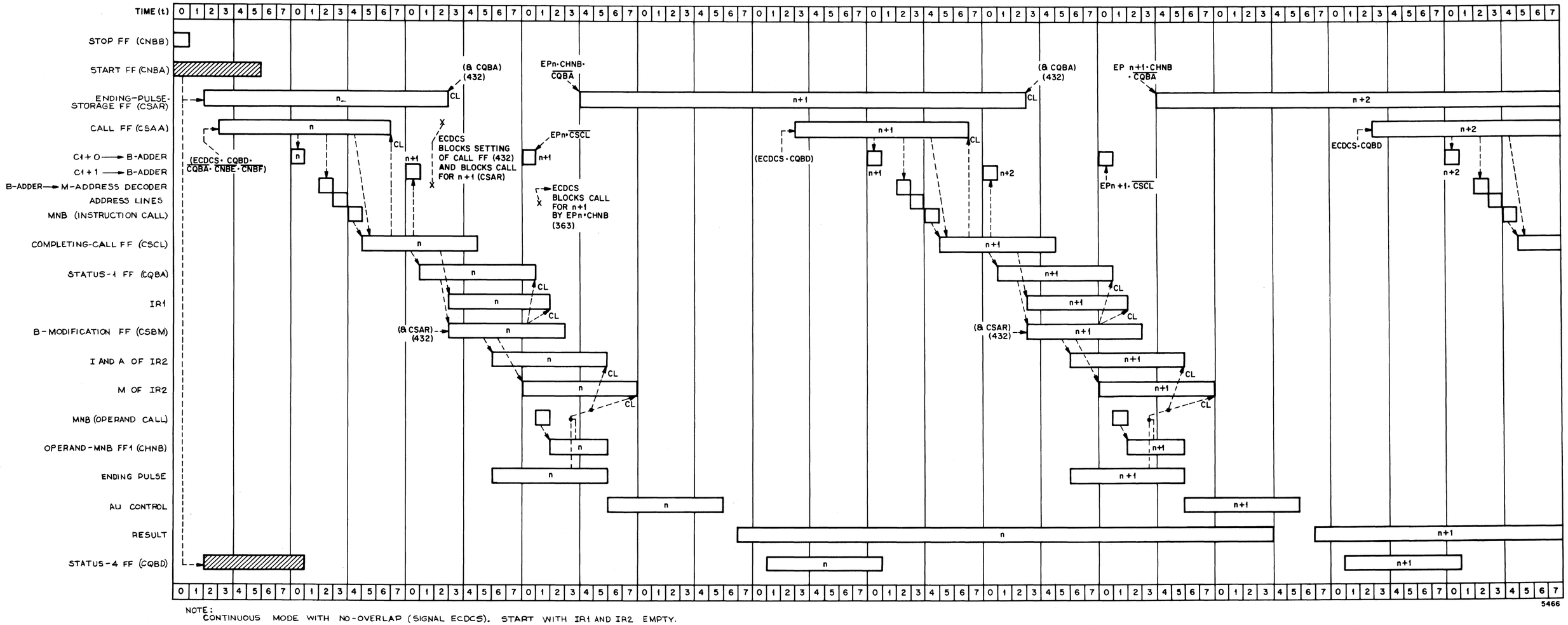
The first instruction in the no-overlap mode (n) is executed through the various stages of the instruction cycle in the usual way. The control counter is advanced by one but the call for the next instruction is blocked until the status-4 FF is set for n. The call FF is then set for n + 1, twenty microseconds after it was set for the first call. Notice that in the absence of status 1, the ending-pulse-storage FF is set by the normal ending pulse and CHNB for instruction n in order to set the B-mod FF for n + 1, when the status-1 FF is again set.

7-8. NO-OVERLAP INTERRUPTED OPERATION

As a further aid in the analysis and isolation of faults, the no-overlap mode can be combined with either the one-instruction mode or the multivibrate mode. In this way a single instruction is executed completely without bringing in the next instruction in sequence and the computing unit then either stops or proceeds at a controlled rate.

A combined one-instruction/no-overlap mode (or multivibrate/no-overlap mode) is carried out essentially as described separately for these modes under the four preceding headings; that is, the functions performed by the no-overlap signal and the setting of the various flip-flops shown in figure 7-1 produce the same control effects as in an uncombined operation, with one exception—the setting of the call FF to call the next instruction in sequence. This is shown in figure 3-1 where the call FF is set by gating the no-overlap signal (ECDCS) with the status-4 signal (CQBD) and no status-1 (CQBA). The output from this gate, however, is inhibited if either the one-instruction FF (CNBE) or multivibrate FF (CNBF) is set. Hence, any succeeding instruction calls are dependent on the setting of the start FF (CNBA) which is the normal case in the one-instruction or multivibrate mode.

When starting a combined operation with an instruction already in IR1 (for example, stop in continuous mode and then select combined one-instruction/no-overlap mode), the setting of the call FF by CNBA is blocked by gating ECDCS with CQBA (status 1). The one-instruction FF has no influence until the completion of the instruction initially in IR1, when it blocks the setting of the call FF by ECDCS and CQBD (status 4) and so on.



NOTE:
CONTINUOUS MODE WITH NO-OVERLAP (SIGNAL ECDCS). START WITH IR1 AND IR2 EMPTY.

Figure 7-4. Timing Diagram - No-Overlap Mode