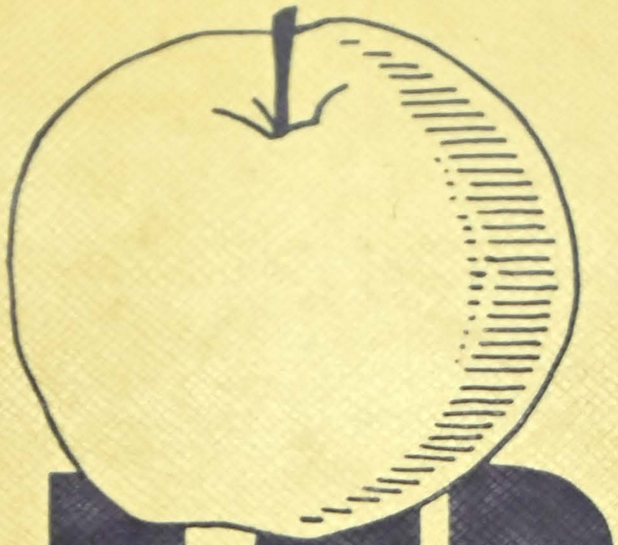


DIGIAC®

DIGITAL SYSTEMS · ELECTROLAB · LAB-SERVO · ETS ELECTRONICS

DIGIAC
CORPORATION

M 68

BI–TRAN SIX

TECHNICAL and OPERATION MANUAL

MODELS 5027/5029

Tm

DIGIAC®
CORPORATION

1261 S. Boyle Avenue ● Los Angeles, California 90023 ● (213) 263-9681
Ames Court ● Plainview, New York 11803 ● (516) 433-7800

A DIGIAC CORPORATION PUBLICATION

Printed in U.S.A.

01-74

TABLE OF CONTENTS

BI-TRAN SIX Computer Education System

Section I

INTRODUCTION AND DESCRIPTION

1-1. PURPOSE OF THIS MANUAL

This manual describes the BI-TRAN SIX digital computer, its theory of operation, and its operating and maintenance procedures. A parts list and instructions for ordering parts are also provided.

1-2. PURPOSE OF THE BI-TRAN SIX

The BI-TRAN SIX is designed specifically for educational purposes. It is the nucleus of a training system that gives the student of computer technology a "see-feel-touch-and-learn" approach to: (1) computer programming; (2) machine organization; (3) logic design; (4) computer circuit design; and (5) electronic troubleshooting and maintenance.

To simplify the understanding of circuit and logic concepts, the BI-TRAN SIX offers a number of unique training features:

(1) Each circuit board is laid out in an easy-to-follow logic diagram configuration.

(2) The layout of each logic diagram exactly matches the layout of the circuit board.

(3) Test equipment leads may be connected to any test point in any circuit of the computer. This allows the student to easily perform signal tracing.

(4) Maximum voltage exposed to the student with power "on," and logic boards raised and operating, is ±10 volts, considerably lower than the recognized safe-voltage level.

(5) The logic-circuit boards extend for use while the computer is operating.

1-3. OPERATING CHARACTERISTICS

The BI-TRAN SIX is an internally-stored-program, single-address computer with a parallel additive arithmetic network. It uses a binary, fractional, magnitude and sign arithmetic system, and has a random-access ferrite-core memory, which stores 128 six-bit words. Data transfer within the computer is accomplished by parallel methods.

The BI-TRAN SIX can carry out 30 instructions (24 basic instructions plus 6 variants), which can be combined in a stored program for problem solving.

A data word consists of six bits. If a data word is used as a numeric, five of these bits represent magnitude and one bit represents sign. Thus, the range of the numeric data is ± 31 decimal.

An instruction consists of two consecutive 6-bit words. The first word, located at an even address in memory, defines the type of instruction to be executed; the second word, located at the consecutive odd address, defines a storage address, a jump address, a shift count, or an input/output load count.

The BI-TRAN SIX operates in any one of four basic modes:

(1) *Program Mode*—Continuous operation until the entire program is carried out.

(2) *Instruction Mode* — One instruction acquired and executed.

(3) *Acquisition/Execution Mode*—Either the acquisition or execution phase is carried out.

(4) *Distributor Mode*—Only one basic timing pulse is generated.

The BI-TRAN SIX can be operated manually either at the registers on the control panel or from an accessory octal-to-binary input keyboard. In addition, it can transfer input-output data to and from peripheral equipment.

1-4. PHYSICAL DESCRIPTION

1-4.1. OVERALL DESCRIPTION

The BI-TRAN SIX (Figure 1-1) is contained in a blue-colored aluminum cabinet with an anodized aluminum front panel. The cabinet is approximately 31 inches wide, 23 inches high, and 17 inches deep. The assembly weighs 98 pounds. A removable cover on the top of the cabinet allows easy access to the logic-circuit boards.

The rear of the cabinet contains five receptacles: EXTERNAL INPUT (J1), EXTERNAL OUTPUT (J2), BI-OCTAL INPUT (J3), AUXILIARY CONTROL PANEL (J5), and AC POWER. The main power fuse for the 110-volt AC line, and —10-volt DC and +10-volt DC fuses are also mounted on the rear of the cabinet. In addition, a removable cover on the rear of the cabinet permits access to the BI-TRAN SIX power supply.

1-4.2. FRONT PANEL

All of the operating controls and indicators for the computer are mounted on the front panel (Figure 1-1). The front panel is divided into two sections. The upper section contains the logic switch-lights for the registers. Each switch-light consists of a lamp and a pushbutton switch. When the pushbutton is pressed, it sets or clears its associated flip-flop (FF), depending upon the particular register. If a switch-light's associated FF is in a binary "1" state, its lamp will be on. The name of the register with which each group of switch-lights is associated is marked immediately above the group. A pushbutton switch, located to the right of each group of register switch-lights, can be used to clear or set the associated register. The significance of all front panel markings is described in Section III.

The lower section of the front panel contains all of the operating controls to start the computer, to set it to a particular mode of operation, to clear all of the registers at once, and so on. In addition, the lower section of the panel contains various error switch-lights that operate when a particular type of error occurs while the computer is in operation. Other switch-lights on the lower section indicate the various operating states of the computer.

The lamps associated with the switch-lights are driven by their own driver amplifiers or are directly coupled through a switch to the power supply. The driver amplifier associated with a lamp is located on the rear of the switch-light on the front panel.

1-4.3. LOGIC CIRCUIT BOARDS

1-4.3.1. General

The BI-TRAN SIX contains seven logic circuit boards and a logic-memory circuit board (Figure 1-1). Each board consists of two separate sections united in a rigid metal frame (Figure 1-2). A riveted metal support down the center of the whole board holds the two sections firmly together. The entire board is approximately 22¼ inches wide, 17¼ inches high, and ¼ inch deep.

Each logic-circuit board has a slide support at one end and a channel guide at the other end, which permit raising the board. Each board can be raised to an upright easy-to-view position while the computer is operating, or can be lifted out of the equipment for access to parts on the board when the computer is not operating. When a board is raised to a viewing position, it is held firmly in its raised position by a detent in the slide support.

Male connectors mounted at the bottom of the board connect electrically to other circuits in the computer. Each of the two sections which make up a logic circuit board has its own connector.

1-4.3.2. Description of Boards

The eight major logic-circuit boards are organized by function. Generally speaking, each performs either one major function or several. Note that the switch-lights associated with the flip-flop registers can be viewed on the console.

Board 1, the instruction register board, holds the instruction register (I_5, I_4, I_3, I_2, I_1, I_0) and the instruction decoding network. All command levels are generated on this board.

Board 2 contains the distributor register (D_3, D_2, D_1, D_0), its associated decoding network, and distributor pulse amplifiers DP_0–DP_{17}. The program address register (P_6, P_5, P_4, P_3, P_2, P_1) is also contained on this board. Note that indicator P_0 on the console is a "dummy" switch-light. It has no corresponding flip-flop on Board 2.

Board 3 is the logic control board, and contains the logic circuits that generate most of the subcommands in the computer. Included are the control circuits of the exchange (X) register, program address (P) register, quotient (Q) register, count down (C) register, memory address (M) register, distributor (D) register, and instruction (I) register. These are marked on the board. A few additional control circuits, mounted on the lower right portion of this board, are associated with storage, external, and stop control.

Board 4, the arithmetic control board, contains the control functions for the accumulator register and the adder network. Included on this board are the adder control register (AC_1, AC_0), the logic zero (LZ) register, the sign (SI) register, the add overflow (AV) register, the divide overflow (DV) register, and the A and Q register control circuits. These various registers and control circuits are primarily associated with arithmetic control. The two other flip-flops on this board are the acquisition/execution flip-flop (AE) and the instruction error (IE) flip-flop.

Board 5 is the accumulator register and adder board. This board contains the accumulator register (A_0, A_1, A_2,

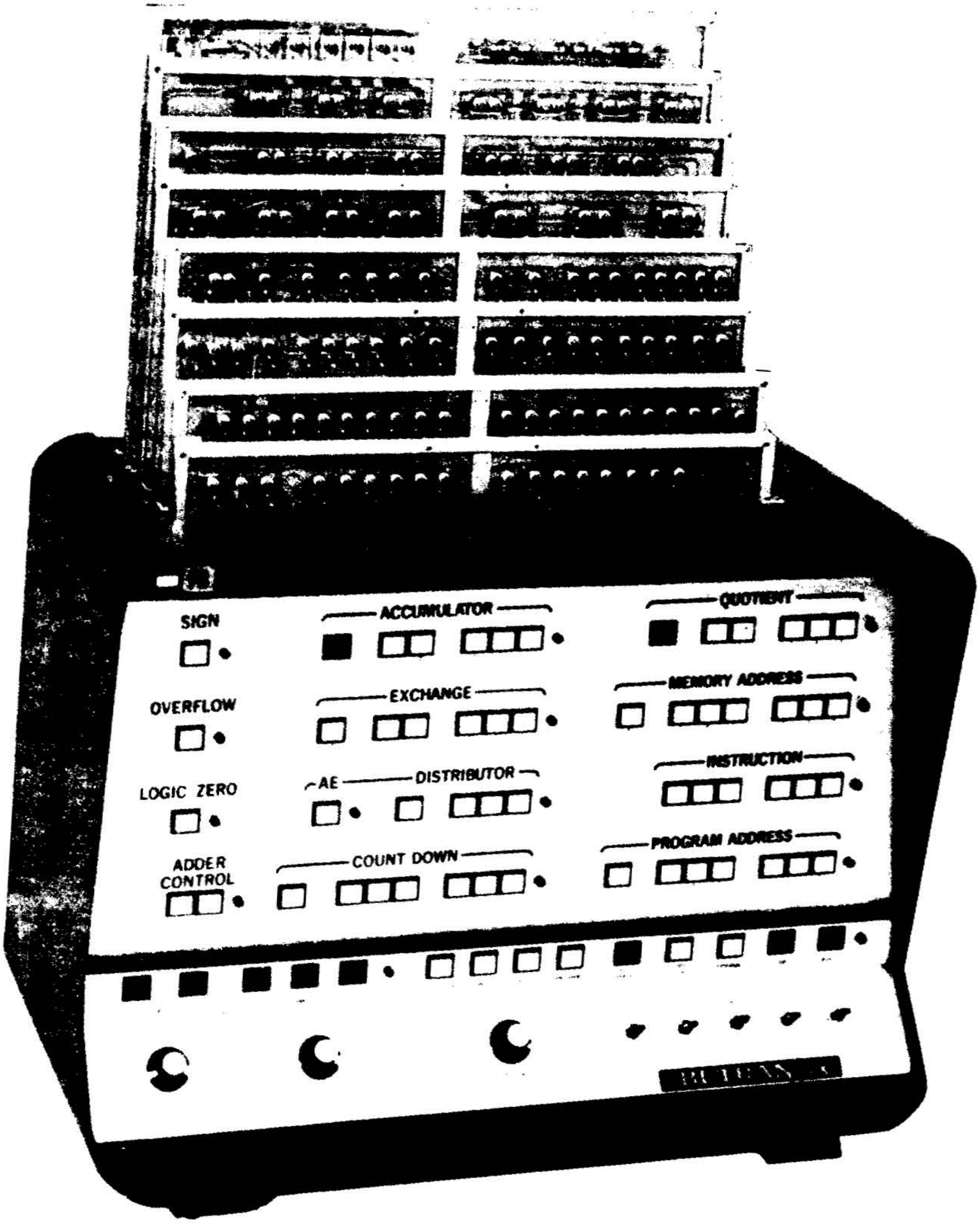


Figure 1-1. Bi-Tran Six

A_3, A_4 A_5), the adder network, and the overflow flip-flop (OV).

Board 6, the X, C, Q register board, contains three registers. These are the exchange (X) register (X_0, X_1, X_2, X_3, X_4, X_5), the count down (C) register (C_6, C_5, C_4, C_3, C_2, C_1, C_0), and the quotient (Q) register (Q_0, Q_1, Q_2, Q_3, Q_4, Q_5).

Board 7 is the main control and M register board. It contains the memory address (M) register (M_6, M_5, M_4, M_3, M_2, M_1, M_0), RUN 1, RUN 2, external flip-flops, and the clock circuits. This board is concerned with start-stop control and the addressing system for the core memory.

Board 8, the memory board, contains the core memory and the associated memory timing circuits. Included on this board are the sense amplifiers, the inhibit drivers, the X-Y transformer matrix, the X and Y current limiters, the X and Y selectors, the core matrix, and the delay line. In addition, the following flip-flops are contained on this board: write FF, read FF, read/write FF's, inhibit FF, and strobe FF. Each of these is carefully identified by appropriate markings.

1-4.3.3. Physical Location Code and Layout Data

Logic boards and prints are coded so that logic circuits and circuit parts can be identified. Military symbols for logic elements are used on the prints. Board No. 1 (Figure 1-2) will be used to explain the code.

All circuits are arranged in functional groups as logic elements (AND/OR) inverters), pulse shapers, delays, isolation diodes, clock, etc. The circuits are boxed within rectangles. Each board is divided into three designation levels, called levels 1, 2, and 3. The levels are designated on the right side of each board. Each level is subdivided into three rows of circuits. The lower two rows consist of diode AND/OR logic circuits or isolation diodes; the third and topmost row consists of amplifier circuits, which may be used either as inverters or delays, or combined to form flip-flops.

The diode AND/OR circuits (or isolation diodes) in the first row on each level are assigned location numbers 1 through 50, with 1 on the right and 50 on the left. Though all are marked on the prints, lack of space prevents marking all circuits on the board. For example, on Board 1, the three-input positive OR circuit on the far right in level 1 is marked with a 1. The circuit next to it, a two-input negative AND, is not marked, but should be referred to as 2.

The diode AND/OR circuits (or isolation diodes) in the second row on each level are assigned numbers 51 through 100, with 51 on the right. For example, on level 1, row 2, the isolation diode for clearing flip-flop I_0 has location number 52.

The third row of each level consists of amplifier circuits. These are designated by their position from right to left with location numbers 1 through 30. As noted previously, the amplifier circuit may represent an inverter, an element of a flip-flop, a delay (single-shot), etc. The position of each transistor and its associated circuit on the board is designated. Thus on level 1, row 3, the inverter to the right of the flip-flop I_1 is designated as being in location 3; i.e., the third transistor from the right. On level 2 the inverter marked STQ has location number 7, and the one on level 3 marked ARG has location number 4.

Location numbers are interspersed on the board to assist in determining the position of the appropriate circuit elements. On the third level, the inverter $\overline{\text{AE}}$ and the inverter NA have been so marked; all other inverters on this level can be found readily from this basic information. Thus, all elements on the board can be easily located by use of the following convention: abbreviation of circuit—board number—level number—location number. For example, on board 1, the three-input positive OR circuit in level 1, row 1, is designated as positive OR-1-1-1 or PO-1-1-1. Inverter STA on level 2 is designated as inverter-1-2-6 or I-1-2-6.

The specific diode in diode AND/OR circuits can be located by adding a subdesignation to the location number. For the three-input positive OR-1-1-1, the three diodes can be described, reading from right to left, as 1-1-1.1, 1-1-1.2 and 1-1-1.3.

All flip-flops are assigned names relating to their functions. The flip-flop names generally correspond to their relationship to a register. For example, I_5, I_4, I_3, I_2, I_1, I_0 are the names of the six flip-flops of the I register. The flip-flops are also annotated further. Each flip-flop has set (S), clear (C), complement inputs, and "1" and "0" output sides designated in the associated rectangle. Certain flip-flops use associated inverter circuits as slaves to permit appropriate fan-out capabilities. Inverter 1-1-3 is the "1" output slave of flip-flop I_1, and inverter 1-1-6 is its "0" output slave. Note that the two inverters of flip-flop I_1 are also counted in the coding scheme. To designate a flip-flop, it is sufficient to describe its board number and name. I_4-1 is flip-flop I_4 on board 1, and A_3-5 is flip-flop A_3 on board 5.

Many inverters and other circuit elements also have logic names to assist in their recognition. A glossary of these names is found in Section VII. On the I board, inverter 1-2-4 is designated MPY (multiply), inverter 1-2-6 is designated STA (store accumulator), and inverter 1-3-13 is designated EXO (external output).

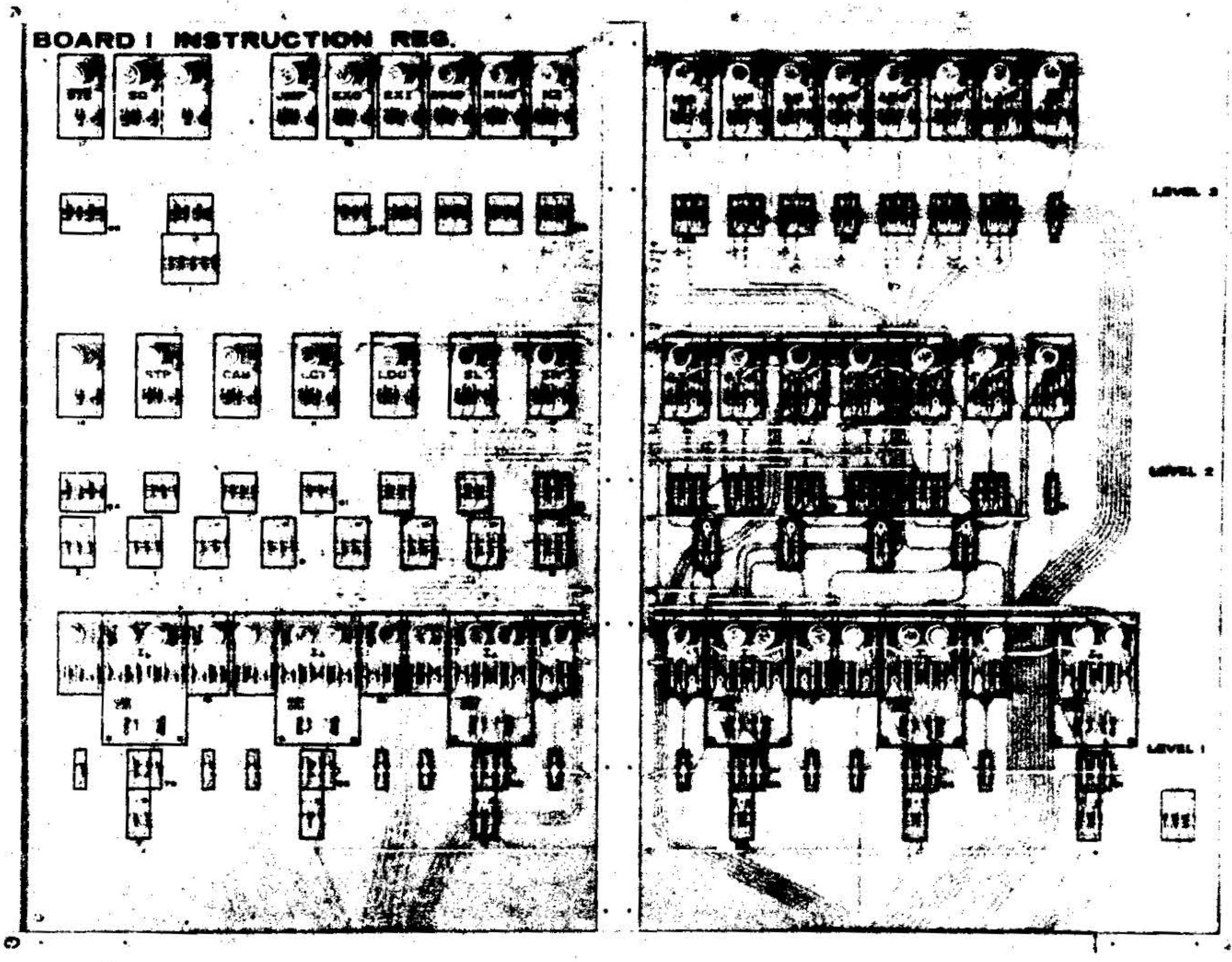


Figure 1-2. Circuit Board Construction

Besides the circuit coding scheme a number of other conventions were established in laying out the boards: (1) All register-to-register transfer subcommands are found on the right-most diode of all AND logic elements; (2) All distributor phase pulses DP_k ($k = 0, 1, 2, \ldots, 17$) are placed on the right-most diode of all AND logic elements. (See board No. 2.) Wherever the DP_k's are OR'd together, they are assembled in order of digit size, the lowest number to the right; (3) The timing pulses, T_k, (board No. 4), where used in AND/OR circuits, have the same priority as the DP_k's; (4) The logic functions AE or $\overline{AE}$ also have the same priority as the distributor pulses where used in AND/OR circuits.

As an example of these conventions, the subcommand which transfers data from the X register to the I register is seen as inputs to the right-most diode of the negative AND circuits 1-1-k, where k can take on the values $k = 2, 3, 4, 5$, and 6. On levels 2 and 3 of board 1, the outputs of inverters $\overline{AE}$ 1-2-1 and 1-3-1 are found as the right-most input of the appropriate negative AND circuits on these levels.

1-5. AUXILIARY EQUIPMENT

1-5.1. DESCRIPTION

A number of optional peripheral devices are available from Fabri-Tek, Incorporated for use with the BI-TRAN SIX. The following devices permit the computer to operate as an integrated data-processing system:

(1) *Paper Tape Punch.* Used to punch binary data into paper tape for tape input to the computer. A six-level binary code is used, with a seventh-level "0" enable and an eighth-level delete.

(2) *Paper Tape Reader.* Converts the binary data on punched tape to electrical pulses and transmits the pulses to the computer.

(3) *Paper Tape Strip Printer.* Converts binary data received from the trainer into alpha numeric-coded information and prints on paper tape.

(4) *Octal-to-Binary Input Keyboard.* Converts manual input data in the form of a 2-digit octal number into a 6-bit binary number and transmits this converted data to the computer.

(5) *Logic System.* This logic demonstration board which, when interfaced with the computer, permits designing and demonstrating many different types of logic circuits, including timing circuits, and advanced system techniques.

1-5.2. INTERFACE REQUIREMENTS

If the auxiliary equipment available from Fabri-Tek, Incorporated is used with the BI-TRAN SIX, interface requirements need not be considered. However, if other output equipment is used, the following interface voltages and currents are available from the computer.

Output of Flip-Flop

"0" state: —10 volts; 25 microamperes

"1" state: 0 to —0.3 volt; 2 milliamperes

Output of Inverter

"0" state: —10 volts; 1 milliampere; 8.2k ohms (max.)

"1" state: 0 to —0.3 volt; 2 milliamperes

If other input equipment is used, the following voltages and currents must be supplied to actuate the BI-TRAN SIX circuits:

Input to Inverter and Flip-Flop

"0" state: Open circuit

"1" state: 0 to —0.3 volt; 1 milliampere

Current is available from the computer for external equipment. Up to 1 ampere can be drawn from both the +10-volt line and the —10-volt line of the EXTERNAL INPUT (J1) and EXTERNAL OUTPUT (J2) jacks on the rear of the computer. Amphenol connector 17-20150 (Fabri-Tek Part No. 024-420) matches connectors J1, J2, and J3 on the computer.

Section II

PREPARATION FOR USE

2-1. UNPACKING

Each BI-TRAN SIX is inspected, tested, and packed carefully when it leaves the factory. However, the purchaser should perform the following inspection upon unpacking in the event it has been damaged during shipment. If any damage is observed. it should be reported immediately to the shipper. Also notify

Fabri-Tek Incorporated
Educational Products Field Service
705 Keller Ave.
Amery, Wisconsin 54001
Telephone (715)-268-7155

2-1.1. PHYSICAL INSPECTION

(1) Check the exterior for dents, gouges, etc.

(2) Check the switches on the lower part of the front panel to make sure they are not loose or bent.

(3) Remove the top cover, and raise and examine each logic-circuit board in turn. Each board should be pulled up slowly until it locks in its raised position. Check to see that the board has not shaken loose from its mounting and that its frame has not been bent.

(4) Inspect each board for cracks, broken component leads, transistors jarred loose from their sockets, or other signs of damage.

(5) Check to see that the connectors at the bottom of each board are firmly mated.

2-1.2. PRELIMINARY OPERATING INSPECTION

(1) Set the following switches to the positions indicated: POWER—OFF; ERROR STOP—NO BY-PASS; MODE—PROGRAM; MODE REPEAT—OFF; JUMP STOP—OFF.

(2) Rotate the POWER switch clockwise until you sense one detent. The DC and the PROGRAM indicators should light. Disregard other lights.

(3) Rotate the POWER switch clockwise until another detent is sensed. The DC and PROGRAM indicators should remain lighted; the STOP switch-light should come on; all other lights on the console should go out.

(4) Rotate the POWER switch clockwise until the final detent is sensed. The MEMORY indicator should light.

(5) Check all the switch-lights above the blue dividing bar. With two exceptions (described below), each switch-light should light when pressed, stay lighted, and go out only when the pushbutton to the group's right is pressed. Check the DIVIDE OVERFLOW, ADD OVERFLOW, and INSTRUCTION ERROR switch-lights in the same manner.

Exception (1): The COUNT DOWN switch-lights are all lighted by the pushbutton; you must press each individual switch-light to turn each switch-light off.

Exception (2): The rightmost light of the PROGRAM ADDRESS register remains lighted only while it is being pressed. This does not affect the operation of the computer.

(6) With all the lights described in (5) turned on, raise the CLEAR switch. All the switch-lights should go out.

(7) Rotate the MODE switch to the INSTRUCTION position, then the AE position, and then the DISTRIBUTOR position, checking to see that each indicator lights in turn.

(8) Depress the MODE REPEAT and JUMP STOP switches. Their indicators should light.

(9) Press the EXTERNAL switch-light. It should light and remain lit.

(10) With the MODE switch in the DISTRIBUTOR position and the MODE REPEAT switch on, press and hold the RUN 2 switch-light. It should light and the STOP switch-light should go out.

2-2. SETTING UP THE COMPUTER

The BI-TRAN SIX is ready for use when it is received. There are no special setup procedures. But keep in mind these considerations:

(1) The computer should not be exposed to excessive heat, moisture, dust, or corrosive air.

(2) Make sure that the table or bench on which the computer is placed can support its 98 pounds.

(3) Allow a few inches of clearance between the computer and a wall so that air can circulate freely through the vents on the rear of the cabinet.

2-3. POWER SOURCE

The BI-TRAN SIX can be operated from any standard 115-volt, 60-cycle AC power source. It draws about one ampere, and consumes approximately 120 watts. The power plug is designed for a three-prong receptacle. An adapter should not be used.

2-4. PACKING FOR RESHIPMENT

Should it become necessary to return the computer for service, it should be addressed to Fabri-Tek, Inc., 705 Keller Ave., Amery, Wisconsin 54001, Attn: Educational Products Field Service. The best way to pack it is to use the crate in which it was received. If this has been discarded, pack the computer in a crate that is at least two or three inches larger than the computer in all dimensions. Fill all empty space in the crate with shock absorbing material so that the computer cannot move during shipment. Be sure to pack shock absorbing material under and over the computer as well. After the crate has been sealed securely, mark it to indicate that it should be handled with care. Insure it to protect yourself in the event of damage.

NOTICE

DO NOT RETURN EQUIPMENT TO FABRI-TEK WITHOUT PRIOR AUTHORIZATION FROM FABRI-TEK INCORPORATED, EDUCATIONAL PRODUCTS FIELD SERVICE.

Section III

OPERATING INFORMATION

3-1. GENERAL

This section describes the functions of all controls, switch-lights, and registers, and gives operating information and examples of program writing, the use of peripheral equipment and manual operation. This section also lists the repertoire of instructions for programming the BI-TRAN SIX. For those unfamiliar with decimal-to-binary conversion, charts are given to permit easy conversion of data from one number system to the other.

3-2. OPERATING CONTROLS

All of the operating controls and their associated switch-lights for the BI-TRAN SIX are located on the lower section of the front panel (Figure 3-1). The functions of these controls and switch-lights are described in Table 3-1.

3-3. REGISTERS

Table 3-2 lists all of the registers on the BI-TRAN SIX. These registers consist of either 1, 2, 4, 6, or 7 flip-flops. The switch-lights associated with these registers are located on the upper section of the front panel (Figure.3-1). Five additional flip-flops involved with control have their switch-lights located on the lower section of the front panel. The functions of these switch-lights are described in Table 3-1.

3-4. INSTRUCTION REPERTOIRE

3-4.1. INSTRUCTION FORMAT

An instruction consists of two consecutive 6-bit data words. The first word, which must always be located at an even address, represents the operation code. This code tells the computer what operation to perform. The format of the first word is as follows:

XXX XX	X
Code Bits	Additional Address or Operation Code Designator

The second word of an instruction, located at the consecutive odd address, is the operand address of the instruction. Depending on the nature of the instruction, it can specify a storage address of an operand, a jump address, a shift count, or a load (or unload) count. The format of the second word is as follows:

XXX XXX

3-4.2. TYPES OF INSTRUCTIONS

Thirty instructions (24 basic instructions plus 6 variants) make up the repertoire of the BI-TRAN SIX. They include elementary arithmetic operations, instructions permitting decisions, storage transfer-type instructions, housekeeping instruction for loop programs, and input/output instructions. The latter instructions permit manual loading and unloading of data, or the use of on-line input/output equipment, such as a paper-tape reader, paper-tape-strip printer, etc., for automatic loading and unloading.

The basic instruction to be performed is determined by the most significant 5 bits of the operation code. In most cases, the least significant bit of the first word modifies either the instruction to be executed or extends the count or storage address of the operand address. Thus, six of the 24 basic instructions (four jump and two shift instructions) have odd and even variants. The odd and even variants of STP and CAS perform the same functions. The odd LCT code sets the most significant bit of the C register; the other six bits are set by the operand address.

The remaining 15 instructions use the least significant bit as an address extension bit. If this bit is a "0", the operand address of the instruction specifies a location in the lower half of the memory (addresses 0—77_8). When this bit is in a "1" state, the operand address specifies a location in the upper half of the memory (addresses 100_8—177_8). Thus, instruction 44—55 will cause the data stored at address 55 to be added to the accumulator. The instruction 45—55 will cause the data stored at address 155 to be added to the accumulator.

3-4.3. LIST OF INSTRUCTIONS

The repertoire of instructions for the BI-TRAN SIX is given in Table 3-3. The first column of the table lists each instruction by general name and symbolic code. The letters that are underscored in each word of an instruction are the letters used to form the instruction's symbolic code. A description of the instruction and its function are given in the "Description" column of the table. Instructions are normally assigned octal number designations for simplified reference purposes; therefore, the octal number for each instruction is listed in the "Octal Code" column. Although each instruction is referred to by an octal number designation, binary numbers are used in the computer; therefore, the binary code for each instruction is listed in the "Binary Code" column of the table.

3-5. SPECIAL PROGRAM CODING CONVENTIONS

3-5.1. INSTRUCTION FORMAT

A BI-TRAN SIX instruction consists of two consecutive words. The first word, (the operation code, sometimes called "command") must always be stored at an even address. The second word, the operand address, may be a memory address, a jump address, a shift count, or an input/output data load/unload count, and must be stored at the next odd address. For example in the ADD instruction, coded as 44, 35, the operation code 44 must be stored at an even memory address like 12, and the operand address, 35, must be stored at the next odd address, in this case 13.

3-5.2. USE OF ODD-EVEN CODES FOR CERTAIN INSTRUCTIONS

To address the full 7-bit range of memory with the 6-bit operand word, codes have been provided for certain instructions. These instructions use the least significant bit of the operation code as the most significant bit of the operand address. Use the even-numbered instruction code for addressing storage locations 0 through 77_8, and the odd-numbered code for addressing locations 100_8 through 177_8. The data address will be coded the same in both cases; it is only the operation code that differentiates the data address.

3-5.3. CODING JUMP INSTRUCTIONS

The operand of a jump instruction must be coded as one-half the address to which it will jump when executed. Thus the instruction "unconditional jump to address 30" (coded symbolically as UNE 30) must be machine-coded as 12, 14, and not as 12, 30. Note that 14_8 (12_{10}) is one-half of 30_8 (24_{10}). The odd jump codes do not affect the jump address; they are used to stop the computer if the JUMP STOP switch is ON.

3-5.4. THE SUBROUTINE JUMP

The instruction SBE or SBO causes an unconditional jump to the jump address specified. At the same time, it stores the contents of P (always two more than that of the address at which SBE is stored) at address 1 in memory. If there is an unconditional jump operation code at memory address 00, the machine later can be made to return to the interrupted program sequence.

3-6. NUMBER REPRESENTATION

The BI-TRAN SIX uses magnitude-and-sign representation for numbers. The most significant bit of a word determines the sign of the numbers. It is "0" if the word is positive, "1" if it is negative. For example, the positive number 21_{10} has binary representation of 010 101 or 25_8, whereas -21_{10} has a binary representation 110 101 or 65_8. Thus, the binary point is located between the sign bit and the most significant bit of the number. The built-in algorithm of the arithmetic process follows the rules of fractional arithmetic, that is, the sum, difference, product, or quotient of two fractional numbers is a fractional number.

The format of an unscaled numeric word is $n_0 . n_1 n_2 n_3 n_4 n_5$ where n_0 is a sign bit (0 for a positive number, 1 for a negative number) and the remaining five bits denote magnitude. The numerical designations of the A, Q, and X flip-flops and switch-lights have been designed to agree with this fractional representation. The subscripts on the flip-flops and the associated numbers on the console correspond to the negative powers of 2, that is 2^{-1}, 2^{-2}, 2^{-3}, 2^{-4}, 2^{-5}. By appropriate scaling, data can be treated as integers.

Table 3-1. Major Operating Controls and Their Switch-Lights

Control or Switch-Light	Control Positions	Function
POWER switch	OFF/ON	Applies power to the computer when in the ON position. Switching from OFF to ON, the POWER switch has 3 positions to apply power sequentially to various sections of the BI-TRAN SIX. In the first position, power is applied to energize all circuits, except memory; in the second position, the switch clears all flip-flops; in the third position (ON), power is applied to the memory READ/WRITE (R/W). Applying power in this manner prevents spurious memory cycles that could modify data already stored in memory. If the AC power plug is removed inadvertently, the POWER switch should be turned OFF and ON again.
DC lamp		Always lit except when the POWER switch is in the OFF position. It indicates that DC power has been applied to all sections except the memory R/W drivers.
MEMORY lamp		Lights when the POWER switch is rotated to its ON position to indicate that power has been applied to the memory R/W drivers.
Error Switch-Lights:		
ADD OVERFLOW switch-light		Lights if the addition of a number to the accumulator has caused the sum to exceed 5-bit data word range, i.e., ±31 decimal. The computer will then stop if the ERROR STOP switch is in the NO BYPASS mode. This switch-light is controlled internally by the add overflow (AV) flip-flop.
DIVIDE OVERFLOW switch-light		Lights if the division of two numbers will result in a quotient greater than the data word range of the quotient register (±31 decimal). The computer will then stop unless the ERROR STOP switch is in the BYPASS mode. This switch-light is controlled internally by the divide overflow (DV) flip-flop.
INSTRUCTION ERROR switch-light		Lights whenever an illegal instruction is detected in the instruction register. (See paragraph 3-10.) The computer will then stop unless the ERROR STOP switch is in the BYPASS mode. This switch-light is controlled internally by the instruction error (IE) flip-flop.
ERROR STOP switch	NO BYPASS	Stops the computer if any of the errors listed above occur.
	AV BYPASS	Permits the computer to continue operation even though an ADD OVERFLOW error has occurred. The AV switch-light will still light, but the computer will stop only when an INSTRUCTION ERROR or a DIVIDE OVERFLOW error occurs.
	BYPASS	Permits the computer to continue to operate regardless of the type of error. However, the error will be indicated by the appropriate error switch-light.
MODE switch	PROGRAM	Causes the computer, after it is started, to acquire and execute instructions by internal control until either an instruction stop, manual stop, or error stop is executed.
	INSTRUCTION	Causes the computer to stop after each instruction has been acquired and executed. One instruction is acquired and executed each time the start switch is pressed. See MODE REPEAT switch.

Table 3-1 Continued

Control or Switch-Light	Control Positions	Function
MODE switch (cont'd.)	AE	Causes the computer to stop after completion of either the acquisition or execution phase of an instruction. One instruction is either acquired *or* executed each time the START switch is operated, depending upon the state of the AE flip-flop. See MODE REPEAT switch.
	DISTRIBUTOR	Causes the computer to stop after each clock period. This includes both the distributor phases and arithmetic phases. One distributor or adder control pulse is generated each time the START switch is operated. See MODE REPEAT switch.
MODE REPEAT switch	ON	Causes each of the operations—except program—selected by the MODE switch to be repeated until an internal or manual stop is executed. In instruction mode repeat, a given instruction will be continually acquired or executed. In AE mode repeat a given instruction will be either continually acquired or executed. In distributor mode repeat a given distributor or adder control pulse will be continually generated.
MODE REPEAT lamp		Indicates that the MODE REPEAT switch is ON.
JUMP STOP switch	ON	Causes a stop to occur before the execution of a jump when odd jump instructions are being executed. The program is resumed by pressing the START switch.
	off	Permits odd jump instruction to be executed without a stop.
JUMP STOP lamp		Indicates that the JUMP STOP switch is in the ON position.
INPUT/OUTPUT switch	INPUT	Sets up the computer to accept input data manually.
	OUTPUT	Sets up the computer to unload output data manually.
EXTERNAL switch-light		Lit while input or output data are being transferred between the computer and peripheral equipment.
STOP switch		Stops the computer after the manual stop signal is synchronized by the trainer's clock. This is the correct method of manually stopping the computer while it is operating.
STOP switch-light		Lights when the computer has ceased operation. If pressed while the computer is operating, it causes a non-synchronized stop and possible modification of memory data.
START/CLEAR switch	START	When the switch is depressed, the manual start signal starts the computer after being synchronized by the computer's clock. The computer will operate as directed by the MODE and MODE REPEAT switches until a stop is executed.
	CLEAR	When the switch is raised, all flip-flops and registers are cleared. The CLEAR switch should not be operated while the computer is running (RUN 2), as memory data may be modified.
RUN 2 switch-light		Lit while the computer is operating. If pressed, it causes a non-synchronized start with possible modification of memory data.

Table 3-2. BI-TRAN SIX Registers

Register and Abbreviation	Description and Function
ACCUMULATOR A	A 6-bit register; five bits represent magnitude and one bit represents a sign. The five bits representing the magnitude of A are designated as the AM register. The A register flip-flops are numbered according to their binary fractional value, i.e., A_5 is the 2^{-5} (1/32) position in the A register. A_0 is the sign. The A register is used to hold: (1) the *augend* and then the *sum* in addition. (2) the *minuend* and then the *difference* in subtraction. (3) the *most significant bits of a product* in multiplication. (See AQ register.) (4) the *most significant bits of a dividend* prior to a division. (See AQ register.) (5) the *remainder* in division.
QUOTIENT Q	A 6-bit register; five bits represent magnitude and one bit represents sign. The five bits representing magnitude of Q are designated as the QM register. The Q register flip-flops are numbered according to their binary fractional value. The Q register is used to hold: (1) the *multiplier* and then the *least significant bits of a product* in multiplication. (See AQ register.) When QM contains a product or a dividend, the positional values are 2^{-6}, 2^{-7}, 2^{-8}, 2^{-9}, 2^{-10}. (2) the *least significant bits of a dividend* and then the *quotient* in division. In the case of the quotient, the numbers associated with this register correspond to their binary fractional values, with Q_0 as the sign bit. (See AQ register.)
AQ	An 11-bit register; ten bits of AM and QM represent magnitude, and A_0 represents sign. Q_0 duplicates the sign of A_0 in multiplication. AM holds the most significant bits and QM the least significant bits.
COUNT DOWN C	A 7-bit decreasing counter. It holds a count of the number of process steps in multiplication and division; the number of shifts in the shift operation; and a count which determines the number of words that can be loaded sequentially into memory from input, or unloaded sequentially from memory into output. The C register flip-flops are designated according to their binary integer value, i.e., C_0 is the 2^0 position.
MEMORY ADDRESS M	A 7-bit increasing counter used to locate any one of the 128 words in the random-access, magnetic-core memory. The M flip-flops are numbered according to their binary integer value, i.e., M_6 is the 2^6 position.
INSTRUCTION I	A 6-bit register that holds the operation code of the instruction being performed. (Paragraph 3-4 lists the repertoire of instructions.) The flip-flops are numbered according to their binary integer value, i.e., I_5 is in the 2^5 position.

Table 3-2 Continued

Register and Abbreviation	Description and Function
DISTRIBUTOR D	A 4-bit increasing counter that establishes the sequence of distributor pulses, which, with the AE flip-flop and the decoded contents of the I register, controls the logic functions to be performed on initiation of each T_0 timing pulse. D register flip-flops are designated according to their binary integer value, i.e., D_3 is the 2^3 position.
PROGRAM ADDRESS P	A 6-bit increasing counter that determines the location of the instruction to be executed. It is increased during the acquisition phase unless the MODE REPEAT switch is in ON. Since an instruction consists of two consecutive words, with the first word being located at an even address, the P register is restricted to addressing the even-ordered words in memory, i.e., 0, 2, 4, . . . , 126. The dummy switch-light P_0 on the console has no function other than to aid the operator in reading the P register. The P register flip-flops are designated according to their binary integer value, i.e., P_2 is the 2^2 position.
EXCHANGE X	A 6-bit register used as a buffer to communicate with every section of the computer. It performs the following functions: (1) holds the addend, the subtrahend, the multiplicand, and the divisor in arithmetic processes. (2) used in control sequencing to transfer data to the M, I, P, Q, and C registers. (3) acts as the data register for memory. (4) communicates with peripheral equipment for input and output word transfers.
ADDER CONTROL AC	A 2-bit increasing counter that functions only when addition is being performed. When the proper distributor phase is reached in an instruction where addition must be performed, the AC counter takes control away from the D register and issues adder control pulses. When the addition is completed, control of the computer is returned to the D register.
SIGN SI	A 1-bit register used as a control element for the arithmetic instructions to assist in sign determination and instruction algorithmic control.
OVERFLOW OV	A 1-bit register used as a control element to capture end carries for addition overflow recognition and instruction algorithmic control.
LOGIC ZERO LZ	A 1-bit register used as a control element. It is set during arithmetic control if the AM register is equal to 37_8. In conjunction with the sign flip-flop, it is used to recognize that the sum in the AM register may be zero rather than 37_8.
ACQUISITION/EXECUTION AE	A 1-bit register used as a control element. It is in the clear state during the acquisition phase of an instruction, and it is in the set state during the execution phase of an instruction.

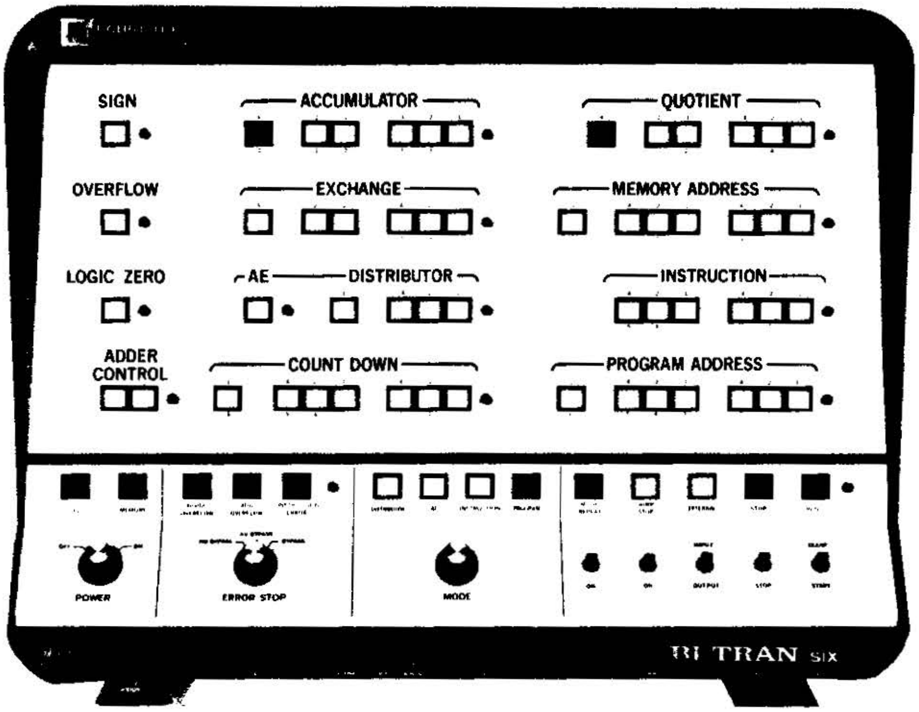


Figure 3-1. BI-TRAN SIX

Table 3-3. Repertoire of Instructions

Instruction and Symbolic Code	Description	Octal Code	Binary Code
Subroutine Jump Even SBE m	Store the contents of the P register at address 1 and then jump to address m for the next instruction. (See Note 1.)	10	001 000
Subroutine Jump Odd SBO m	If the jump switch is set, the computer will stop. Upon restart, the SBE instruction will be executed. If the jump switch is not set, the SBE instruction will be executed. (See Note 1.)	11	001 001
Unconditional Jump Even UNE m	Jump to address m. (See Note 1.)	12	001 010
Unconditional Jump Odd UNO m	If the jump switch is set, the computer will stop. Upon restart, the UNE instruction will be executed. If the jump switch is not set, the UNE instruction will be executed. (See Note 1.)	13	001 011
Negative Accumulator Jump Even NAE m	If the contents of the accumulator are negative, jump to address m; otherwise continue the present sequence of instructions. (See Notes 1 and 2.)	14	001 100
Negative Accumulator Jump Odd NAO m	If the jump switch is set, the computer will stop. Upon restart, the NAE instruction will be executed. If the jump switch is not set, the NAE instruction will be executed. (See Notes 1 and 2.)	15	001 101
Non-Zero Jump Even NZE m	If the accumulator is not equal to zero, jump to address m; otherwise continue the present sequence of instructions. (See Notes 1 and 2.)	16	001 110
Non-Zero Jump Odd NZO m	If the jump switch is set, the computer will stop. Upon restart, the NZE instruction will be executed. If the jump switch is not set, the NZE instruction will be executed. (See Notes 1 and 2.)	17	001 111
Manual Input MNI m	Store the manual input data inserted consecutively in the X register at consecutive memory addresses starting with address m. Continue until the C register is zero. The number of words stored in memory will be one more than the initial count in the C register. (See Note 3.)	20 or 21	010 000 or 010 001

Table 3-3 Continued

Instruction and Symbolic Code	Description	Octal Code	Binary Code
Manual Output MNO m	Transfer the contents of consecutive memory addresses, starting with address m, to the X register for visual output. The number of words displayed in one more than the initial count of the C register. Continue until the C register is zero. (See Note 3.)	22 or 23	010 010 or 010 011
External Input EXI m	Store the input data taken from external equipment at consecutive memory addresses starting with address m. Continue until the C register is zero. The number of words stored in memory will be one more than the initial content of the C register. (See Note 3.)	24 or 25	010 100 or 010 101
External Output EXO m	Transfer the contents of consecutive memory addresses, starting with address m, to external equipment. Continue until the C register is zero. The number of words unloaded from memory will be one more than the initial content of the C register. (See Note 3.)	26 or 27	010 110 or 010 111
Load A LDA m	Clear the accumulator and add the contents of memory address m to the accumulator.	40 or 41	100 000 or 100 001
Load A Negative LDN m	Clear the accumulator and subtract the contents of memory address m from the accumulator.	42 or 43	100 010 or 100 011
Add ADD m	Add the contents of memory address m to the contents of the accumulator. (See Note 4.)	44 or 45	100 100 or 100 101
Subtract SUB m	Subtract the contents of memory address m from the accumulator. (See Note 5.)	46 or 47	100 110 or 100 111
Replace Absolute Unity RAU m	Add 1 to the contents of memory address m. (See Note 6.)	50 or 51	101 000 or 101 001
Replace Subtract Unity RSU m	Subtract 1 from the contents of memory address m.	52 or 53	101 010 or 101 011

Table 3-3. Repertoire of Instructions (cont'd.)

Instruction and Symbolic Code	Description	Octal Code	Binary Code
Multiply MPY m	Multiply the contents of the accumulator by the contents of memory address m, leaving a double-length product in the AQ register. (See Note 7.)	54 or 55	101 100 or 101 101
Divide DIV m	Divide the double-length number in the AQ register by the contents of memory address m. The quotient will be in the Q register and the remainder will be in the accumulator. (See Note 8.)	56 or 57	101 110 or 101 111
Store Accumulator STA m	Store the contents of the accumulator at memory address m.	60 or 61	110 000 or 110 001
Store the Q Register STQ m	Store the contents of the Q register at memory address m.	62 or 63	110 010 or 110 011
Shift AQ Right (Even) SRE k	Shift the contents of the AQM register right k places.	64	110 100
Shift AQ Right (Odd) SRO k	Clear the Q register, then shift the contents of the AQM register right k places.	65	110 101
Shift AQ Left (Even) SLE k	Shift the contents of the AQM register left k places.	66	110 110
Shift AQ Left (Odd) SLO k	Shift the contents of the AQM register left k places and transfer the sign bit of the Q register to the sign bit of the accumulator.	67	110 111
Load Consecutive LDC m	Transfer the contents of memory address m to memory address m + 1.	70 or 71	111 000 or 111 001
Load the Count Down Register LCT k	Set the C register to a count of k. (See Note 9.)	72 or 73	111 010 or 111 011
Change Accumulator Sign CAS	Complement the sign bit of the accumulator. (See Note 10.)	74 or 75	111 100 or 111 101
Stop STP m	Stop the computer. Upon restart, jump to m for the next instruction. (See Note 1.)	76 or 77	111 110 or 111 111

Table 3-3 Continued

NOTES

Note 1. The operand address of a jump instruction should be coded as one-half of the address to which the program should jump. For example, to jump to address 34_8, the operand address m should be coded as 16_8.

Note 2. For the conditional jump instructions, negative zero (40_8) is treated for decision purposes as if it were a positive zero.

Note 3. (a) Using the input/output instructions in a program requires that C be preset by the LCT (72 or 73) instruction if more than one word is to be loaded into or unloaded from the computer. (b) The computer will stop before the input request for instruction MNI. Load the input data into X. Depress the START switch for each word loaded. The data will then be stored at the address designated in M. (c) For instruction MNO the computer stops after the data has been transferred to X. To see the next data word, press the START switch.

Note 4. If the contents of memory address m is 40_8 (negative zero), then the sign of the sum formed in A will have the sign of the original number in A.

Note 5. If the contents of memory address m is 00_8, then the difference resulting from subtracting it from A will have the sign of the original number in A.

Note 6. The RAU instruction treats the 6-bit contents of memory address m as a 6-bit absolute number. The addition of "1" to this number is an absolute addition. For example, if the contents of address m is 37_8, the new contents of this address will be 40_8. If the contents were originally 66_8, the final contents will be 67_8. If the contents of address m (an odd number) is 77_8, the contents of address m will become 00 and the contents of address m — 1 (an even number) will be modified so that the least significant bit of this address is set to a "1" state. For example, assume the RAU instruction is applied to address 61_8 and that the initial contents of addresses 60_8 and 61_8 are respectively:

60: 44_8
61: 77_8

The final contents of these addresses upon application of the RAU instruction will be:

60: 45_8
61: 00

In the case where address m is an even number and its contents is 77_8, then the effect of RAU m will be to modify the contents of m to 01_8 and leave m — 1 (odd number) unaffected.

Note 7. In multiplying two numbers, the sign of the Q_0 bit is set to the same state as the A_0 bit. When numbers are oppositely signed and one number has a magnitude of "0", then the product in the AQ register will be a negative "0", i.e., the states of the A_0 and Q_0 bits will be "1".

Note 8. If the dividend and divisor are oppositely signed and the dividend has a "0" magnitude, then the quotient will be a negative "0" (40_8) and the "0" remainder will carry the initial sign of the dividend.

Note 9. An even LCT operation code is used to set the C register to counts 0 through 77_8. The odd code is for counts ranging from 100 through 177_8. The parameter k determines bits C_0 through C_5. The least significant bit of the operation code therefore determines the state of C_6.

Note 10. The operand address of the instruction CAS is not used.

3-7. DECIMAL-OCTAL-BINARY CONVERSION

The range of data (treated as integer information) permitted by a 6-bit signed numeric word in the BI-TRAN SIX is $\pm 31_{10}$. Numeric data (treated as integer information) in the AQ register has a range of $\pm 1023_{10}$, corresponding to an 11-bit signed word. Using Tables 3-4 and 3-5, numbers can be converted quickly from one system to another. The magnitude of the decimal number is first converted to its octal equivalent. Then each octal digit is replaced by its binary equivalent. For numbers in the range $\pm 31_{10}$ and numbers in the range $\pm 1023_{10}$, the most significant bit of the 6-bit or 11-bit binary equivalent is changed to a "1" if the number is negative.

Table 3-5, the Octal-Decimal Conversion Table, is used as follows: If the decimal numeral whose octal equivalent is to be determined is not on the table, find the closest decimal numeral less than it. Add the difference of the two decimal numerals to the octal equivalent of the smaller decimal numeral.

Example 1: To convert 29_{10} to octal and then to binary, find the closest number less than 29_{10} on Table 3-5. This is 24_{10}. Add the difference between 29_{10} and 24_{10}, that is, 5, to the octal equivalent of 24_{10} (30_8) to get the equivalent of $29_{10} = 35_8$ to binary, replace the two digits 3 and 5 with their binary equivalents from Table 3-4, the Octal-Binary Conversion Table: $35_8 = 011\ 101_2$; -35_8 corresponds to $111\ 101_2$.

Example 2: To convert 1014_{10} to octal, use the Table 3-5, and write the equations:

$$\begin{aligned} 1014_{10} &= 1008_{10} + 6 \\ 1008_{10} &= 1760_8 \\ \text{Thus, } 1014_{10} &= 1760_8 + 6 \\ 1014_{10} &= 1766_8 \end{aligned}$$

Using Table 3-4, replace the octal digits with their binary equivalents: $1766_8 = 01\ 111\ 110\ 110$; $-1766_8 = 11\ 111\ 110\ 110$.

3-8. GENERAL OPERATING INFORMATION

3-8.1. STARTING THE BI-TRAN SIX

To start the BI-TRAN SIX, perform the following steps:

(1) Rotate the POWER switch clockwise to ON. The DC and MEMORY indicators in the lower left corner of the console will light.

(2) Set appropriate switches for desired manner of operation. (See Table 3-1.)

Table 3-4. Octal-Binary Conversion

Octal	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

(3) If a program is to be loaded into the computer, refer to Table 3-6 for manual loading instructions or to paragraph 3-8.3 for external loading instructions. If a program is already stored in the computer, set P to the first program address and press the START switch. The program will then be executed in a manner that depends upon the settings of the various switches.

3-8.2. MANUAL LOADING AND UNLOADING

The procedures for manually loading and manually unloading the computer are almost identical. These procedures are given in Table 3-6.

3-8.3. EXTERNAL LOADING AND UNLOADING

The computer can be loaded from an external paper tape reader and unloaded onto the paper tape strip printer. Both the reader and the printer are available from Fabri-Tek, Incorporated. Procedures for operating the reader and printer are sent with each of these units when they are shipped.

If a reader or printer other than the one available from Fabri-Tek is used, provisions must be made to apply a —10-volt resume signal to the trainer for automatic loading and unloading. Otherwise, the trainer START switch will have to be pressed to load or unload each word.

No matter what type of reader and printer is used, the reader must be connected to the EXTERNAL INPUT (J1) receptacle, and the printer must be connected to the EXTERNAL OUTPUT (J2) receptacle. Both of these receptacles are on the rear of the computer.

To load the computer from the paper tape reader or unload it onto the paper tape strip printer, follow the procedures given in Table 3-7.

Table 3-5. Octal-Decimal Conversion

Octal	Decimal	Octal	Decimal	Octal	Decimal	Octal	Decimal
0	0	400	256	1000	512	1400	768
10	8	410	264	1010	520	1410	776
20	16	420	272	1020	528	1420	784
30	24	430	280	1030	536	1430	792
40	32	440	288	1040	544	1440	800
50	40	450	296	1050	552	1450	808
60	48	460	304	1060	560	1460	816
70	56	470	312	1070	568	1470	824
100	64	500	320	1100	576	1500	832
110	72	510	328	1110	584	1510	840
120	80	520	336	1120	592	1520	848
130	88	530	344	1130	600	1530	856
140	96	540	352	1140	608	1540	864
150	104	550	360	1150	616	1550	872
160	112	560	368	1160	624	1560	880
170	120	570	376	1170	632	1570	888
200	128	600	384	1200	640	1600	896
210	136	610	392	1210	648	1610	904
220	144	620	400	1220	656	1620	912
230	152	630	408	1230	664	1630	920
240	160	640	416	1240	672	1640	928
250	168	650	424	1250	680	1650	936
260	176	660	432	1260	688	1660	944
270	184	670	440	1270	696	1670	952
300	192	700	448	1300	704	1700	960
310	200	710	456	1310	712	1710	968
320	208	720	464	1320	720	1720	976
330	216	730	472	1330	728	1730	984
340	224	740	480	1340	736	1740	992
350	232	750	488	1350	744	1750	1000
360	240	760	496	1360	752	1760	1008
370	248	770	504	1370	760	1770	1016

3-8.4. CLEARING MEMORY ADDRESSES

Whenever a word is loaded into storage at a particular address, any word that is already stored at that address is automatically cleared out. Therefore, any address can be cleared by writing a zero into the address. To clear a single memory address, or a group of nonconsecutive addresses, write zero into the registers by using the manual loading procedure given in Table 3-6. The entire memory storage can be cleared out more quickly by performing the following recommended procedure:

(1) Clear all registers by lifting the CLEAR switch on the lower section of the trainer front panel.

(2) Set the load consecutive instruction (70_8) into I.

(3) Set the AE flip-flop.

(4) Set the MODE switch to AE.

Table 3-6. Manual Loading and Unloading

<table>
<tr><th>MANUAL LOADING</th><th>MANUAL UNLOADING</th></tr>
<tr><td>1. Raise CLEAR switch.</td><td>1. Raise CLEAR switch.</td></tr>
<tr><td>2. Lift INPUT switch.</td><td>2. Depress OUTPUT switch.</td></tr>
<tr><td>3. Set M to the address at which the first word is to be loaded.</td><td>3. Set M to the address from which the first word is to be unloaded.</td></tr>
<tr><td>4. Press START switch. The computer will wait with D = 5.</td><td>4. Press START switch, and the first word will be transferred to the X register. Press START switch and the next word will replace it. (To unload words from nonconsecutive addresses, repeat the procedure for each address.)</td></tr>
<tr><td>5. Load first word into X and press START switch. Load the second word and press the START switch; continue this until all words are loaded. (If words are to be loaded into nonconsecutive addresses, M must be set to the desired address before the START switch is pressed.)</td><td></td></tr>
<tr><td colspan="2">RUNNING THE PROGRAM
To execute the program, raise the CLEAR switch, set MODE switch to PROGRAM, and set P to the starting address. The program will be executed when the START switch is depressed.</td></tr>
</table>

Table 3-7. External Loading and Unloading

<table>
<tr><th>EXTERNAL LOADING</th><th>EXTERNAL UNLOADING</th></tr>
<tr><td>1. Raise CLEAR switch.</td><td>1. Raise CLEAR switch.</td></tr>
<tr><td>2. Raise INPUT switch.</td><td>2. Press OUTPUT switch.</td></tr>
<tr><td>3. Change the instruction in I from 20 to 24.</td><td>3. Change the instruction in I from 22 to 26.</td></tr>
<tr><td>4. Set C to n — 1, where n is the number of words to be loaded.</td><td>4. Set C to n — 1, where n is the number of words to be unloaded.</td></tr>
<tr><td>5. Set M to the address at which the first word should be loaded.</td><td>5. Set M to the address from which the first word is to be unloaded.</td></tr>
<tr><td colspan="2">6(a). To make the computer stop after loading or unloading the last word, set MODE switch to INSTRUCTION.
(b). To make the computer run the program after loading or unloading the last word, set MODE switch to PROGRAM, and set first program address into P.
7. Press START switch.</td></tr>
</table>

(5) Set the MODE REPEAT switch to ON.

(6) Press START switch. Note that the LDC instruction is now being executed continually.

(7) Clear X. All storage locations will be cleared out almost instantly.

(8) Press the STOP switch.

This procedure can also be used to load any 6-bit word into all storage locations. In step (7), set X to the desired value.

3-9. SAMPLE PROGRAM ROUTINES

Typical programs illustrating addition, subtraction, multiplication, division, and basic algebra are given in Sample Programs 1 through 5. The computer can be operated in any mode (PROGRAM, INSTRUCTION, AE, or DISTRIBUTOR) to carry out the programs. In certain training situations, it might be desired to carry out parts of a program in one mode and other parts of the program in other modes. Likewise, the settings of the ERROR STOP, MODE REPEAT, and JUMP STOP switches might be changed during different parts of a program to illustrate different computer concepts. Therefore, a firm understanding of the functions of the various operating controls (Table 3-1) is essential before carrying out the following routines.

For simplicity, it will be assumed that all of the programs are carried out in the PROGRAM mode, with the ERROR STOP switch set to NO BYPASS and the MODE REPEAT switch off. Also, it will be assumed that the JUMP STOP switch is set to ON, so that the computer will stop before it carries out an unconditional odd jump instruction. While a program can start at any address, all of the programs in this section will start at address 00.

3-10. ERROR DETECTION

There are three error lamps on the lower section of the front panel. These are: ADD OVERFLOW lamp; DIVIDE OVERFLOW lamp; and INSTRUCTION ERROR lamp. Should one of these errors occur, the appropriate lamp will light and the computer will stop, provided that the ERROR STOP switch is set to NO BYPASS. If the ERROR STOP switch is set to BYPASS, the computer will not stop no matter what type of error occurs; however, the appropriate error lamp will still light.

If the ADD OVERFLOW lamp lights while a program is being carried out, it indicates that the sum of two numbers exceeds $\pm 31_{10}$ which is the modulus of the computer's arithmetic system. When the ADD OVERFLOW lamp comes on, the computer will stop when the ERROR STOP switch is set to NO BYPASS. In certain types of problems, it is desirable that the computer not stop if an addition overflow occurs. If it is desired that a program be carried out to completion even though an add overflow error might occur, the ERROR STOP switch can be set to the AV BYPASS position. An elementary problem illustrating this point is the determination of the larger of two numbers where the difference may exceed the data number range. The computer, though, will still stop if a divide overflow or an instruction error occurs.

The DIVIDE OVERFLOW lamp will light if the division of two numbers results in a quotient whose magnitude exceeds $\pm 31_{10}$. When the lamp comes on, the computer will stop, unless the ERROR STOP switch is set to the BYPASS position.

There are a number of illegal instruction codes that cannot be used with the BI-TRAN SIX. These are:

00 through 07

30 through 37_8

If one of these illegal codes is accidentally written into a program, the computer will stop when the code is reached, and the INSTRUCTION ERROR lamp comes on, if the ERROR STOP switch is set at NO BYPASS.

3-10.1. EXAMPLE OF ADD OVERFLOW

The algebra program given in Sample Program 5 can be used to illustrate an add overflow error. First, make sure that the ERROR STOP switch is set to NO BYPASS. Then let n equal -31_{10}. The result of subtracting 1 from -31 in the denominator should be -32_{10}, but the sum in the accumulator will be -1. The ADD OVERFLOW switch-light will come on, and the computer will stop.

3-10.2. EXAMPLE OF DIVIDE OVERFLOW

The algebra program given in Sample Program 5 can also be used to illustrate a divide overflow error. With the ERROR STOP switch in the NO BYPASS position, set n equal to 1. Subtracting 1 from 1 in the denominator results in zero; therefore, the DIVIDE OVERFLOW switch-light will come on, and the computer will stop.

3-10.3. EXAMPLE OF INSTRUCTION ERROR

The computer will stop whenever an illegal instruction code (00 to 07, 30 to 37_8) is reached. The subtraction program given in Sample Program 2 can be used to illustrate an instruction error. Instead of loading binary 100 110 (46_8) into address 02, load binary 000 110 (06), which is an illegal instruction code. When the program reaches this instruction, the computer will stop, and the INSTRUCTION ERROR switch-light will come on.

Sample Program 1. Addition: 2 + 3 and 4 + 8_{10}

SYMBOLIC PROGRAM			MACHINE-CODED PROGRAM		
Program Address	Operation Code	Operand Address	Program Address	Operation Code	Operand Address
a_0	LDA	x_0	00	40	14
a_2	ADD	x_1	02	44	15
a_4	UNO	a_6*	04	13	03*
a_6	LDA	x_2	06	40	16
a_{10}	ADD	x_3	10	44	17
a_{12}	STP	00	12	76	00

DATA		MACHINE-CODED DATA	
Data Address	Data Word	Data Address	Data Word
x_0	2	14	02
x_1	3	15	03
x_2	4	16	04
x_8	8_{10}	17	10

Program Operation: The sum of 2 + 3 will be formed in A when the program reaches the unconditional jump (UNO) instruction at address 04, and the computer will stop. The contents of A will be 0 00 101.

The START switch must then be pressed to carry out the next addition (4 + 8). Again the sum of the two numbers will be formed in A when the program reaches the stop (STP) instruction. Now the contents of A will be 0 01 100.

*Because of the transfer relationships between X and P, jump addresses are coded as one half of the address to which the program should jump, i.e., ½ (06) = 03. (See paragraph 3-5.3.)

Sample Program 2. Subtraction: 6 — 2, 3 — 5, and —6 — (8_{10})

SYMBOLIC PROGRAM			MACHINE-CODED PROGRAM		
Program Address	Operation Code	Operand Address	Program Address	Operation Code	Operand Address
a_0	LDA	x_0	00	40	22
a_2	SUB	x_1	02	46	23
a_4	UNO	a_6	04	13	03
a_6	LDA	x_2	06	40	24
a_{10}	SUB	x_3	10	46	25
a_{12}	UNO	a_{14}	12	13	06
a_{14}	LDA	x_4	14	40	26
a_{16}	SUB	x_5	16	46	27
a_{20}	STP	00	20	76	00

DATA		MACHINE-CODED DATA	
Data Address	Data Word	Data Address	Data Word
x_0	6	22	06
x_1	2	23	02
x_2	3	24	03
x_3	5	25	05
x_4	—6	26	46
x_5	8_{10}	27	10

Program Operation: The difference between 6 and 2 will be displayed in A when the program reaches the unconditional jump instruction (UNO) at address 04, and the computer will stop. The contents of A will be 0 00 100.

The START switch must then be pressed for the computer to perform the next subtraction (3 — 5). Again, the difference between the two numbers will appear in A when the program reaches the unconditional jump instruction at address 12, and the computer will stop. Now, however, the A_0 bit, which represents sign, will be lit to indicate that the difference is a negative number. The contents of the A register will be 1 00 010.

To carry out the last subtraction (—6 — 8_{10}), the computer START switch must be pressed again. The difference between the two numbers will appear in A when the stop instruction (STP) is reached. Again, the A_0 bit will be lit to indicate that the difference is a negative number. The contents of A will be 1 01 110.

Sample Program 3. Multiplication: —5 × 4 and 12_{10} × 14_{10}

SYMBOLIC PROGRAM			MACHINE-CODED PROGRAM		
Program Address	Operation Code	Operand Address	Program Address	Operation Code	Operand Address
a_0	LDA	x_0	00	40	14
a_2	MPY	x_1	02	54	15
a_4	UNO	a_6	04	13	03
a_6	LDA	x_2	06	40	16
a_{10}	MPY	x_3	10	54	17
a_{12}	STP	00	12	76	00

DATA		MACHINE-CODED DATA	
Data Address	Data Word	Data Address	Data Word
x_0	—5	14	45
x_1	4	15	04
x_2	12_{10}	16	14
x_3	14_{10}	17	16

Program Operation: The product of —5 × 4 is formed in AQ when the program reaches the unconditional jump instruction (UNO). The contents of AQ will be:

A	Q
1 00 000	1 10 100

The A_0 bit is set to indicate that the product is a negative number. The Q_0 bit, the sign bit for the Q register, will be set also. This bit is ignored. Thus, the magnitude is read as 0 000 010 100 which equals 20_{10}. Since the magnitude of the product is less than the maximum magnitude of Q, one can treat the sign bit of Q as being the sign A. This is the reason for Q_0 duplicating A_0.

To carry out the next multiplication (12_{10} × 14_{10}), the START switch must be pressed again. The new product will be formed in AQ and, since the product will be a positive number, neither the A_0 bit nor the Q_0 bit will be set. The contents of AQ will be 0 00 101 01 000.

Note that Q_0 has been ignored. The magnitude should be read as 0 010 101 000. This equals 250_8 or 168_{10}.

Sample Program 4. Division: 14 ÷ 3 and —14 ÷ 4

SYMBOLIC PROGRAM			MACHINE-CODED PROGRAM		
Program Address	Operation Code	Operand Address	Program Address	Operation Code	Operand Address
a_0	LDA	x_0	00	40	20
a_2	SRE	05	02	64	05
a_4	DIV	x_1	04	56	21
a_6	UNO	a_{10}	06	13	04
a_{10}	LDA	x_2	10	40	22
a_{12}	SRE	05	12	64	05
a_{14}	DIV	x_3	14	56	23
a_{16}	STP	00	16	76	00

DATA		MACHINE-CODED DATA	
Data Address	Data Word	Data Address	Data Word
x_0	14_{10}	20	16
x_1	3	21	03
x_2	-14_{10}	22	56
x_3	4	23	04

Program Operation: The quotient formed by dividing 14_{10} by 3 is in Q; the remainder is formed in A. Then, after division is performed, the results are:

A	Q
0 00 010	0 00 100
(+2)	(+4)

After restarting the program, a new quotient is formed by dividing (-14_{10}) by 4. Initially, the AQ register reads: 1 00 000 01 110. After division is performed, the results are:

A	Q
1 00 010	1 00 011
(—2)	(—3)

Sample Program 5. Algebra: Evaluate $\frac{n^2}{n-1}$ for varying value of n

SYMBOLIC PROGRAM / MACHINE-CODED PROGRAM

Program Address	Operation Code	Operand Address	Program Address	Operation Code	Operand Address
a_0	MNI	x_0	00	20	16
a_2	LDC	x_0	02	70	16
a_4	RSU	x_1	04	52	17
a_6	LDA	x_0	06	40	16
a_{10}	MPY	x_0	10	54	16
a_{12}	DIV	x_1	12	56	17
a_{14}	UNO	a_0	14	13	00

DATA / MACHINE-CODED DATA

Data Address	Data Word	Data Address	Data Word
x_0	n (temp.)	16	n_8
x_1	n — 1 (temp.)	17	$(n-1)_8$

Program Operation: This program will evaluate the ratio $\frac{n^2}{n-1}$ for varying values of n. Upon starting the program, the computer will stop prior to completing the manual input instruction. The number, n, for which the ratio will be evaluated, should then be loaded into the X register. Then, after restarting the computer, it will stop at the unconditional (odd) jump instruction, and the quotient will appear in Q and the remainder in A. The program will return to a new request for input (a new value of n) when the computer START switch is pressed.

Section IV

THEORY OF OPERATION

4-1. GENERAL

This section of the manual describes the general theory by which the BI-TRAN SIX carries out the operating functions explained in Section III. Using the functional block diagram of the computer (Figure 4-1), the interrelationships of the main sections of the computer are explained. Each main section is broken down to its major logical functions to show how the individual computer steps are carried out. Finally, the basic electronic circuits are described in detail. The detailed logic theory for the arithmetic control section is given in this manual because a thorough understanding of arithmetic control is essential to understand how the BI-TRAN SIX operates.

4-2. OVERALL THEORY

4-2.1. GENERAL

The BI-TRAN SIX computer (Figure 4-1) contains six main sections: (1) input, (2) control, (3) memory, (4) arithmetic, (5) output, and (6) power supply.

The input section is the communication link between the computer and the external world. Binary information in the form of program and numeric data must be stored in the computer's memory. This input data must be prepared in a form understandable to the computer. To accomplish this, a paper tape punch, which converts octal data into a six-level binary equivalent on standard one-inch paper tape, a paper tape reader, and an octal-to-binary keyboard, which converts octal data into binary coded electrical pulses and transmits these pulses into the BI-TRAN SIX, are available as peripheral equipment. The octal-to-binary keyboard permits manually loading the BI-TRAN SIX, while the paper tape reader permits automatic input of the coded data. Data can also be loaded into the computer manually through the front panel.

The output section allows the final results from the memory section to be displayed or printed on external equipment. A peripheral paper tape strip printer is available which receives electrical output signals from the trainer, converts the information into alpha numeric code, and prints the results on strip paper tape. Results can also be displayed in the registers on the front panel.

The power supply converts common 115-volt AC line voltage to +10 volts and —10 volts DC to energize the various sections of the computer.

The control, memory, and arithmetic sections of the computer carry out all the logical operating functions of the 30 instructions (24 basic instructions plus 6 variants) from its instruction repertoire.

The program, as well as other data, is stored in the memory section. The arithmetic section performs all the mathematical operations upon receiving subcommands from the control section. These operations are performed sequentially. The arithmetic section receives its data from the memory section and after carrying out the appropriate operations, can send the answers back to the memory section.

The control section performs the most important function within the computer. It actuates the other sections—and controls the data flow between sections that is necessary to solve a problem—by generating commands and subcommands.

4-2.2. ACQUISITION/EXECUTION

The BI-TRAN SIX carries out each of its instructions in two phases: acquisition and execution. Each instruction consists of an operation code (the function or command to be performed) and an operand address (generally speaking, the location of the data word in memory on which the function is to be performed). The acquistion phase is completed when the operation code is in the I register and the operand address is in the M register.

To accomplish acquisition, the content of the P register is transferred to the M register. The operation code in memory at the address indicated by the M register is transferred to the I register through the X register. The M register is then incremented by one (which is the asso-

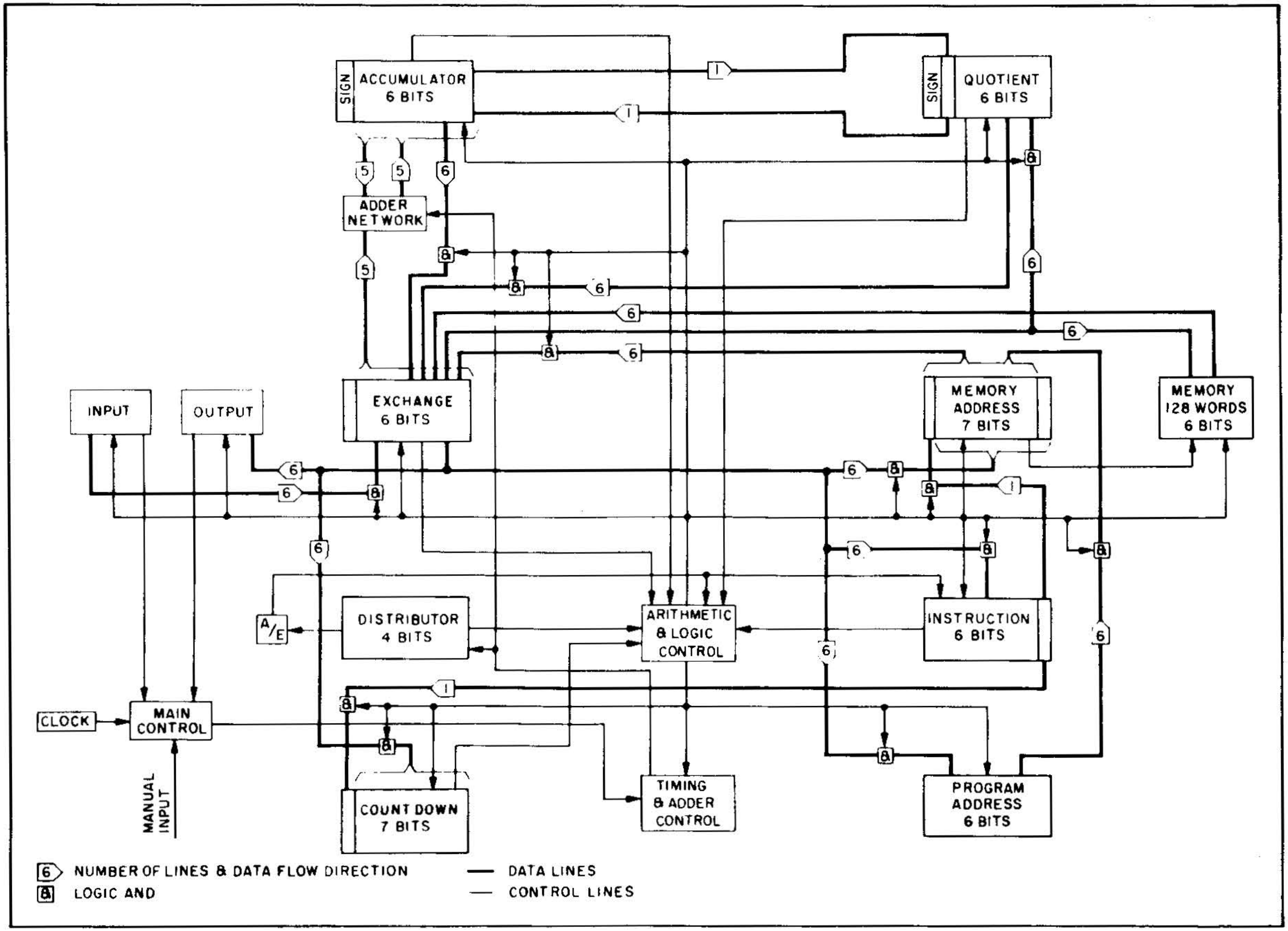


Figure 4-1. BI-TRAN SIX. Functional Block Diagram

ciated odd address of the current instruction) in order to obtain the operand address. The operand address is transferred from memory to the M register through the X register. Concurrently, the P register is incremented to indicate the instruction in the program that would normally be next (if there were no jump instruction).

During the execution phase, the instruction is carried out; i.e., add, subtract, multiply, divide, shift, etc. For example, during the add sequence of the execution phase, the operation code for ADD (in the I register) is decoded, and the data word in storage (indicated by the operand address in the M register) is transferred to the X register, where through the adder network, it is then added to the content of the A register. Similar procedures are followed for all basic instructions.

At the completion of the execution phase, the computer automatically reverts back to the acquisition phase in order to acquire the next instruction. It will be noted that during the previous acquisition phase, the P register was incremented. This incrementation was necessary so that the stored program can be executed sequentially. Acquisition and execution of the instructions continue in this manner until the stored program has been completed.

4-2.3. LOADING

The operational description just described assumes that a program had already been stored within the memory section of the computer. In actual practice, the computer can be loaded either manually or by punched paper tape. Loading manually can be accomplished by setting an appropriate sequence such as described in Table 3-6. It would, of course, be more expeditious to punch the sequence of instructions and data words on a paper tape, and feed this data into the memory of the computer through the input section. In this case, the external load sequence would be followed as indicated in Table 3-7. This would result in subcommands being issued within the computer to read the paper tape into the memory section for storage. Storage addresses for each word would be supplied automatically by the control section. The external load instruction (EXI) would continue loading until the entire program and/or data were stored.

4-2.4. EXCHANGE REGISTER

The X register plays an important role in the arithmetic operations, but it also serves as the means of exchanging binary words between sections. All words from input or to output pass through the X register, as do words being transferred between the control, memory, and arithmetic sections.

4-3. CONTROL SECTION

4-3.1. GENERAL

The control section (Figure 4-1) issues the commands and subcommands in the computer that cause the instructions to be acquired and executed. To do this, the control section includes the:

(1) arithmetic and logic control circuits, which issue subcommands.

(2) acquisition/execution (AE) flip-flop.

(3) instruction (I) register and its decoding network (commands).

(4) distributor (D) register and its decoding network (distributor pulses).

(5) clock, main control, and timing circuits.

(6) adder control (AC) register.

(7) count down (C) register.

(8) program address (P) register.

The AE flip-flop determines whether acquisition or execution subcommands are issued. The decoded outputs of the I register are ANDed with the output of the AE flip-flop to determine the various commands. The decoded outputs of the D register are ANDed with the basic timing pulse T_0, generating the distributor pulses DP_k. The distributor pulses and the commands are logically interconnected within the arithmetic and logic circuits to issue the internal subcommands. The clock source, main control, and start/stop control generate the basic timing pulse T_0 when the AC register is in a "0" state; otherwise, adder control pulses T_1, T_2 and T_3 are issued to generate subcommands to cause the adder network to carry out the addition process. The main control contains the external resume circuits required for input/output. The C register is used during certain instructions to permit the repetition of certain subcommands. The P register determines the memory address of the next instruction to be executed.

4-3.2. ACQUISITION/EXECUTION

The acquisition/execution (AE) flip-flop determines whether the control section will carry out the acquisition or execution phase of a particular instruction. During the acquisition phase, the AE flip-flop is in the zero state. In this state, the I register and associated command decoding network are disabled so that any operation code present there will not affect the logic control circuits. Concurrently, the logic control circuits will allow the distributor pulse signals from the D register and its decoding network to cause the acquisition internal subcommands to be generated.

When the acquisition phase has been completed, the AE flip-flop is set, causing the computer to enter the execution phase. This causes the logic control circuits to disable the acquisition internal subcommand circuits. Concurrently, the AE flip-flop is applied to the I register and decoding network so that the instruction to be executed will send its command to the arithmetic and logic control circuits. At this time, the instruction command allows the individual distributor pulses from the D register and its decoding network to cause the execution of internal subcommands to be sent to the other sections of the computer.

4-3.3. INSTRUCTION REGISTER

The I register contains six flip-flops, and therefore, can hold a 6-bit instruction code. These flip-flops are set to the binary code of the particular instruction that must be excuted. The outputs of the I register flip-flops enable certain logic circuits, which decode the operation code. The logic circuits then send command signals to the logic control circuits.

4-3.4. DISTRIBUTOR REGISTER

The D register is a 4-bit up-counter that is incremented by the timing pulse T_0, which, essentially, is generated by ANDing the clocking circuits and start/stop circuits with the zero decoded output of the adder control (AC) counter so that it can count in binary the equivalents of decimal zero through 15. However, since octal notation is used with the BI-TRAN SIX, the D register is said to count octally from zero through 17. The binary counts of the D register are decoded, in conjunction with the timing pulse, to generate distributor pulses DP_0 through DP_{17}. The D register starts the acquisition phase at DP_0 and then advances one count for each timing pulse T_0 it receives, until DP_{17} is generated. Each distributor pulse is sent to the logic control circuits, where it allows the proper acquisition subcommands to be generated. After DP_{17} is reached, the AE flip-flop is set and the D register is cleared back to zero to start the execution phase. The count is then repeated so that the DP signals, in conjunction with the commands, allow the execution subcommands to be generated. Refer to Figure 4-7 for the specific subcommands that are carried out during each distributor phase of any instruction.

In certain instructions, the D register may be preset to a prior value so that it will repeat certain subcommands. The distributor is incremented by one 15 microseconds after the generation of any distributor pulse. Furthermore, for the instructions in which the subcommands are repeated, the D register is reset. The setting of the D register to its new state occurs 30 microseconds after generation of the distributor pulse initiating the reset of D. The D register is temporarily disabled whenever adder control pulses, T_1, T_2, and T_3 are being generated.

4-3.5. CLOCK AND TIMING CIRCUITS

The clock is basically a free-running multivibrator that produces 15-microsecond square wave pulses every 50 microseconds. Thus, its pulse repetition time is 50 microseconds; pulse width is 15 microseconds; and pulse repetition frequency is 20 kc.

The clock circuits generate timing pulses T_0, T_1, T_2, and T_3. Timing pulse T_0 is generated if the AC register is in a zero state. This pulse permits the logic control circuits to execute all internal subcommands, with the exception of those working with the adder network. Timing pulses, T_1, T_2, T_3 are generated whenever the AC register is in a non-zero state. These pulses generate the subcommands involved with the addition of unsigned data contained in the AM and XM registers.

Timing pulse T_0 is also disabled temporarily whenever an instruction calls for adder subcommands. This is covered in paragraph 4-3.6.

4-3.6. ADDER CONTROL

When an unsigned addition takes place, it is accomplished in a sequence of three steps. The BI-TRAN SIX uses the AC register to take over the subcommand sequence during addition. The AC register is a 2-bit counter that is inhibited at a count of zero. When an add sequence is called for, the logic circuits cause the AC register to start counting and disable timing pulse T_0. The D register then stops counting and keeps the control section in the particular distributor phase it was in. The AC register counts to three, and during each increment, it issues adder timing pulses (T_1, T_2, and T_3) to the arithmetic section. At the end of the third count, the adder sequence is completed, and the AC register is cleared. This enables the T_0 pulse to continue incrementing the D register. If the computer is in the distributor mode, only one of the pulses T_1, T_2, or T_3 will be issued unless the START switch is pressed.

4-3.7. COUNT DOWN REGISTER

The C register is a 7-bit down-counter. It is used as a decision counter for the sequenced instructions—multiply, divide, shift right, and shift left—as well as for the input/output instructions. This register is either preset by the LCT instruction to be used by the input/output instruction or is set to a specified count during the execution phase of the remaining instructions. The most significant bit of C, i.e. C_6, is determined by I_0. By setting this register, certain distributor phases can be repeated.

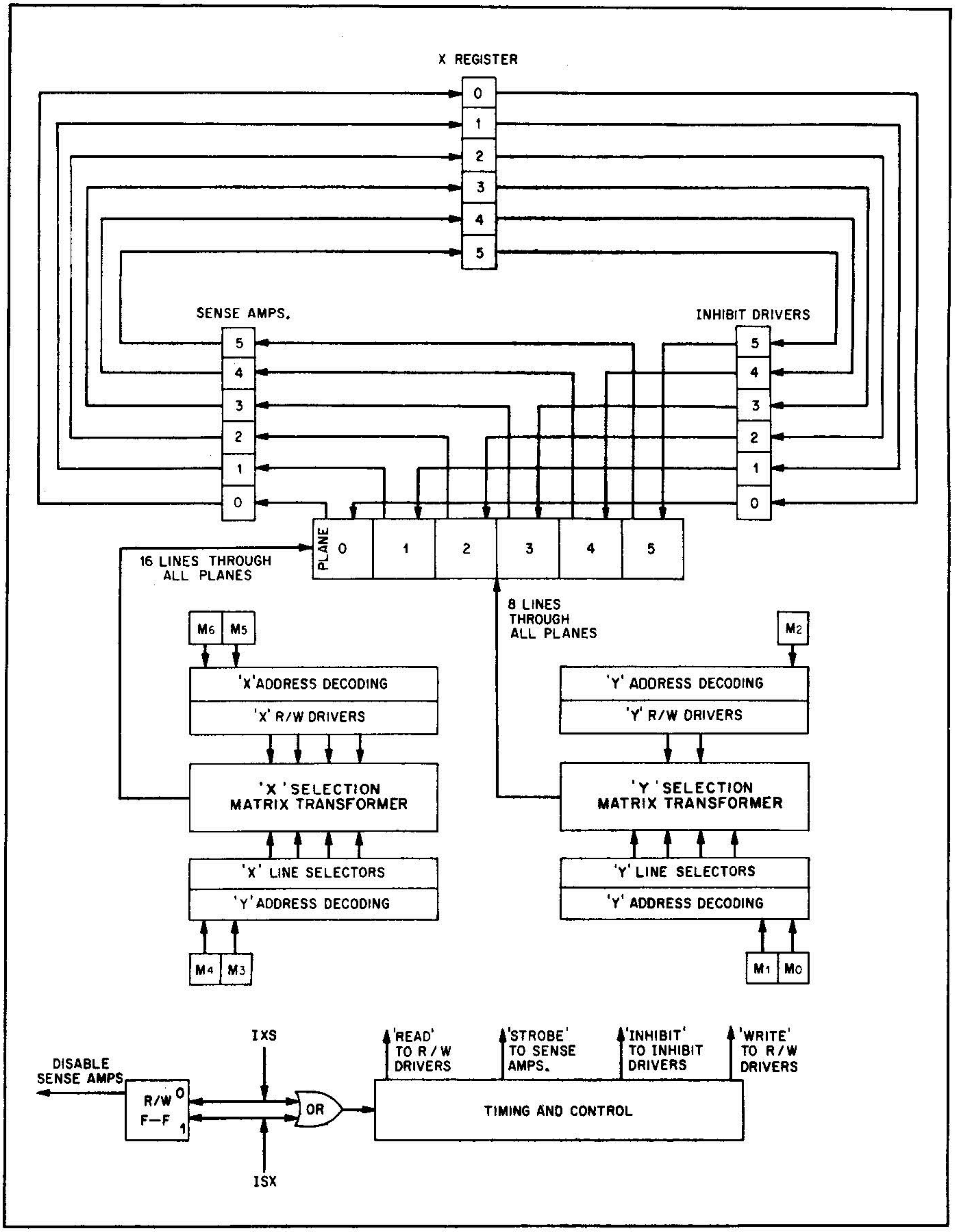


Figure 4-2. Memory System, Block Diagram

After each repetition of a given set of distributor phases, the C register is decreased by one until it reaches the value 0. Upon reaching this state, the execution phase is completed and control is returned to the acquisition phase.

4-3.8. PROGRAM ADDRESS REGISTER

The program address register, a 6-bit up-counter, is concerned only with program control, i.e., determining the next instruction to be executed after completion of the current instruction. It must be manually preset to the starting memory address before a program is run, and will automatically call instructions from sequentially, even-ordered memory cells (unless a jump instruction is executed) until the program stops. The associated odd memory cell is obtained by incrementing the M register during the acquisition phase. Although the P register is only a 6-bit register, it is treated from a control standpont as if it were capable of only representing even numbers. To address memory, the contents of P are transferred to the most significant six-bits of the M register, a 7-bit up-counter. The M register is first cleared, and then the transfer from P to M occurs, leaving the M_0 bit cleared, It should be noted that the contents of M is an even number since M_0 is in a clear state at this point.

4-4. MEMORY SECTION

4-4.1. GENERAL

The memory section (Figure 4-2) stores the program to be carried out by the BI-TRAN SIX. In most cases, the data upon which the program operates is also stored in the memory. Basically, the memory section contains the: (1) core matrix; (2) memory address register; (3) X and Y address circuits; (4) timing and control circuits; (5) inhibit drivers; (6) sense amplifiers; and (7) exchange (X) register.

The core matrix stores binary words at the addresses specified by the M register. The X and Y address circuits decode the M register addresses in order to write words into or read words out of the desired location in the core matrix. The inhibit drivers, under the control of the X register, are either activated to store a "0", or remain inactivated to store a "1". The timing and control circuits enable the other circuits in the proper sequence.

4-4.2. CORE MATRIX

The BI-TRAN SIX core matrix consists of 768 ferromagnetic cores arranged in six 8 × 16 planes of 128 cores each (Figure 4-3). Each core stores one bit, and has a specific address. To store one 6-bit computer word, one core in each of the six planes is selected.

There are 128 X and Y wire combinations in each plane. The location at which the associated X and Y wires cross through a core is the address of that core. The identical position of a core in each of the six core planes comprises the address of the 6-bit word. A core is said to store a "1" when its residual magnetic flux circulates within the core in one direction, and a "0" when its residual magnetic flux circulates in the opposite direction. When a writing current flows in the selected X and Y lines, the core through which the lines cross will tend to be magnetized in the "1" direction. However, each plane also has an inhibit wire that passes through all of the cores in the plane. If a core should not have a "1" written into it, the inhibit wire will carry a current that will produce à magnetic field to cancel the effects of the X and Y line, so that the core will have a "0".

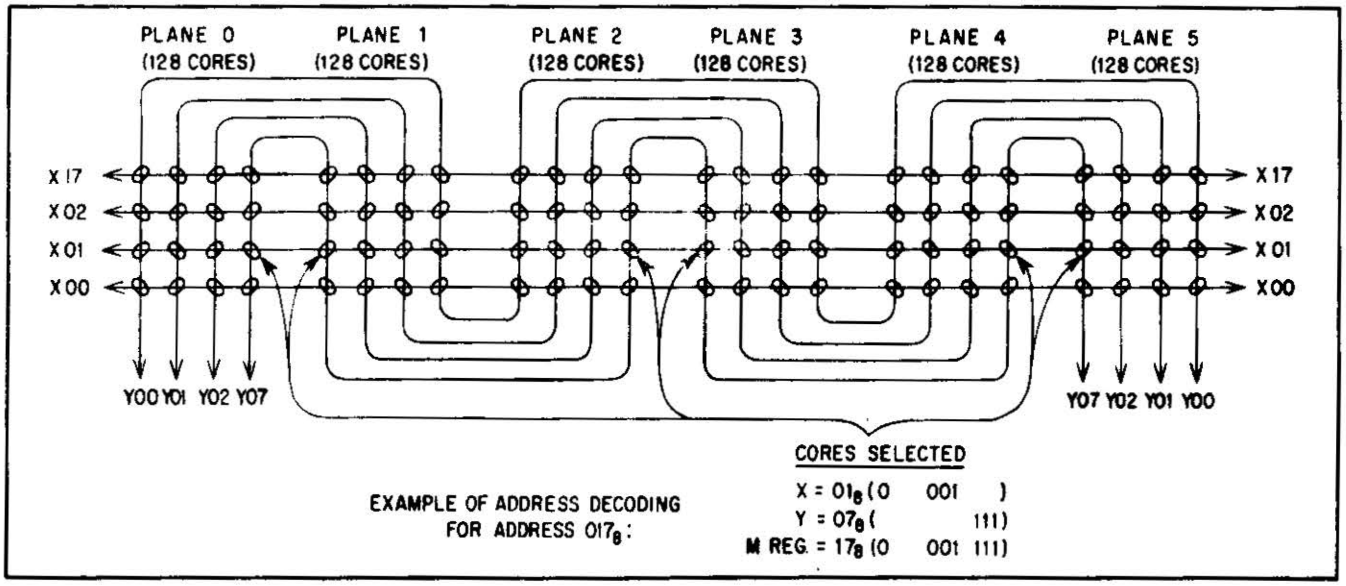


Figure 4-3. Core Matrix, Simplified Diagram (Selection Lines X03-X16, Y03 through Y06, and Sense and Inhibit Lines Not Shown)

Each plane also has a fourth wire, called a sense wire, passing through all the cores. These wires are used when reading the bit that is stored in a particular addressed core. The read-out is accomplished by having the X and Y wires of a core address carry currents that will write "0's" into those cores. Any core at that address that has a "1" in it will have its magnetic field reversed. The reversing flux lines will then induce a pulse in the sense wire to signify that a "1" is being read out. If the core already held a "0", its flux lines would not be reversed, and so no pulse would be induced in the wire, signifying that a "0" was read out.

4-4.2.1. Core Selection

The four wires passing through the center of each core act as transformer windings and are used as follows:

The X and Y wires select cores for read-out or write-in. The X and the Y wires each supply half the current needed to magnetize a core. Only one X and one Y wire can be energized at any given time. Thus only one core can have coincidence of X + Y current. A core receiving only X or only Y current is not selected and will not become magnetized.

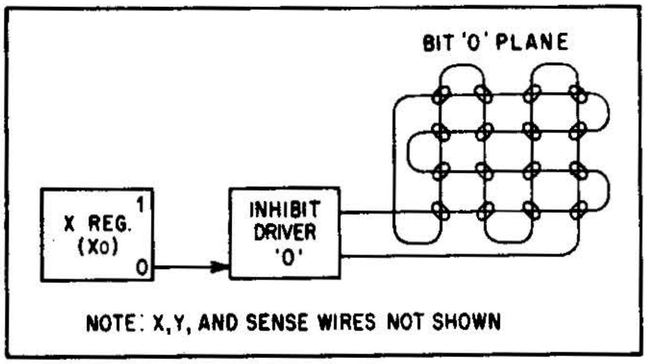
Each core in a plane has an X and a Y address. X addresses are 0 through 17_8, Y addresses are 0 through 7_8. The four upper bits of the M register (M_6-M_3) produce the X address, the three lower bits (M_2-M_0) of M produce the Y address.

The X and Y selection circuits are common to all six planes; i.e., a selected X and Y address energizes one bit in each of the six planes. The six bits energized are, of course, a computer word. Current can flow through the X and Y wires in either direction. In a selected core, the X and Y currents will add to set the core to the "1" or the "0" state depending upon their directions. The direction of X and Y current flow is determined by the read/write timing control. During the read portion of a memory cycle, X and Y currents add to store "0", and in the write portion, to store "1".

All combinations of X and Y current possible in a core are as follows (assuming no inhibit current):

	Current in Write Direction	Current in Read Direction
No X + No Y	Core is unchanged	Unchanged
X + No Y	Core is unchanged	Unchanged
No X + Y	Core is unchanged	Unchanged
X + Y	Core is set to 1	0

Note that if a core is storing a "0" and receives X plus Y currents in the read direction to store a "0", the core will remain unchanged.



NOTE: X,Y, AND SENSE WIRES NOT SHOWN

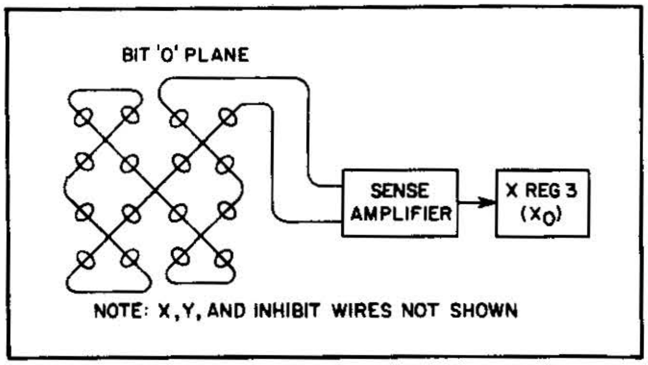
Figure 4-4. Inhibit Wire, Simplified Diagram

Current in the inhibit wire cancels the effect of the write current in the X and Y selection wires when the currents are adding to store a "1". (See Figure 4-4.) The combination of write currents X plus Y plus inhibit current will leave the core in the "0" state. There is one inhibit wire per plane, and when an inhibit driver is energized, it inhibits all cores in its plane, i.e., one bit of the computer word.

The sense wire (Figure 4-5) is the read-out wire for a core. When a core changes state ("1" to "0", or "0" to "1"), current is generated in the sense winding by the changing flux lines. Note that this current is only generated *while* the core is changing its state. If a core is selected but does not change state, there will be no changing flux lines to generate current in the sense line. One sense wire senses all cores in a plane and in turn drives its associated sense amplifier. Only one core can be selected in a plane at any given time so that each sense amplifier will handle only one bit of the selected word.

4-4.2.2. Data Representation in Core Matrix

A given computer word in the core matrix will be stored in one core in each plane. These six cores will



NOTE: X,Y, AND INHIBIT WIRES NOT SHOWN

Figure 4-5. Sense Wire, Simplified Diagram

have a specific address and are referred to as a cell, or core register. Thus, the BI-TRAN SIX core memory has 200_8 cells (or registers) of six bits each.

Data is read from a cell as follows: Read currents from the X and Y selection circuits will coincide at the six cores of one cell. These cores will be cleared by the read current. As the cores change from "1" to "0", they generate current in their associated sense wires. A core that was originally clear (storing "0") does not change state, so it develops no sense wire current. Each sense wire drives an associated sense amplifier. The six sense amplifiers amplify the core outputs and load the X register with the core data, i.e., a core that changes from "1" to "0" develops sense-wire current that is amplified by a sense amplifier that sets its associated X register flip-flop. Reading data from a cell destroys the contents of the cell. Regardless of the original state of the cores in a cell, they are all cleared by the read current.

Data is loaded into a cleared cell as follows: Write currents from the X and Y address selection circuits will coincide at the six cores of one cell. These cores will all be set unless inhibit current is present. Each of the cores has an associated inhibit driver, which is controlled by an associated bit in the X register. If X_k = "1", Inhibit Driver_k will not provide inhibit current. If X_k = "0", Inhibit Driver_k will provide inhibit current. In this manner, the M register, through the address circuits, has selected a cell by trying to write all "1's" into it, and the X register, through the inhibit drivers, determines which cores in the selected cell are set.

4-4.3. MEMORY ADDRESS REGISTER

The M register indicates the address of the cores that are to be written in or read out. The M register has seven flip-flops, and it can be loaded by the P or X register or it can operate as a counter. When the M register is used as a counter, it is incremented by subcommands from the control section. Even-numbered addresses for instructions are transferred into the M register from the P register. When an operand address is transferred to the M register from the X register, the least significant six bits of M are loaded. The most significant bit of M, i.e.. M_6, is determined by the state of I_0.

4-4.4. X AND Y ADDRESS CIRCUITS

The X and Y address circuits decode the binary word in the M register to provide the desired path for the read and write currents. These circuits comprise read/write drivers, select circuits, and transformer matrices for both the X and Y lines. The read/write drivers and the select circuits for the sixteen X lines decode the four most significant bits in the M register; and those circuits for the eight Y lines decode the three least significant bits. Therefore, any given binary code in the M register will select a unique X and a unique Y line when the memory timing circuits actuate the drivers. When the associated R/W drivers and select circuits are actuated, their corresponding X and Y transformers send the read or write currents through their lines.

4-4.5. INHIBIT DRIVERS

The inhibit drivers are used during the write portion of the memory cycle to determine whether the addressed core will have a "0" or a "1". There are six inhibit drivers, one for each core plane. Each inhibit driver corresponds to one bit of the 6-bit word to be written into the addressed cores. The inhibit drivers are controlled by the binary word in the X register. Any inhibit driver whose associated bit is a "1" will *not* send an inhibit current to its plane. Thus, the X and Y wires will write a "1". But any inhibit driver whose associated bit is a "0" *will* send current. This will inhibit the writing of a "1", so that the addressed core will hold a "0". The inhibit drivers function only when they are enabled by the read/write timing circuits.

4-4.6. SENSE AMPLIFIERS

The sense amplifiers are used during the read sequence to sense whether the addressed cores hold a "0" or a "1". There are six sense amplifiers, one for each core plane. Each sense amplifier senses "1" bit of the 6-bit word that is read out. When a sense amplifier is enabled by the read/write timing circuits, it amplifies and transfers its bit to the corresponding flip-flop in the X register.

4-4.7. READ/WRITE TIMING CIRCUITS

The read/write timing circuits send enable signals to the other circuits in the memory section so that they will operate at the proper times. The timing circuits do this after receiving IXS or ISX subcommands from the control section. The timing circuits contain a delay line, and the read, write, strobe, and read/write flip-flops (board No. 8) to develop the memory circuit enabling signals. These signals are shown on the timing diagram (Figure 4-6). Actually, the operation that takes place for an ISX or an IXS subcommand is the same, except for the sense amplifier control.

4-4.7.1. Read Sequence (ISX)

The read sequence (ISX) transfers data from a selected cell to the X register and restores that cell to its original configuration. Before a read memory cycle starts, the address of the cell to be read out is put in the

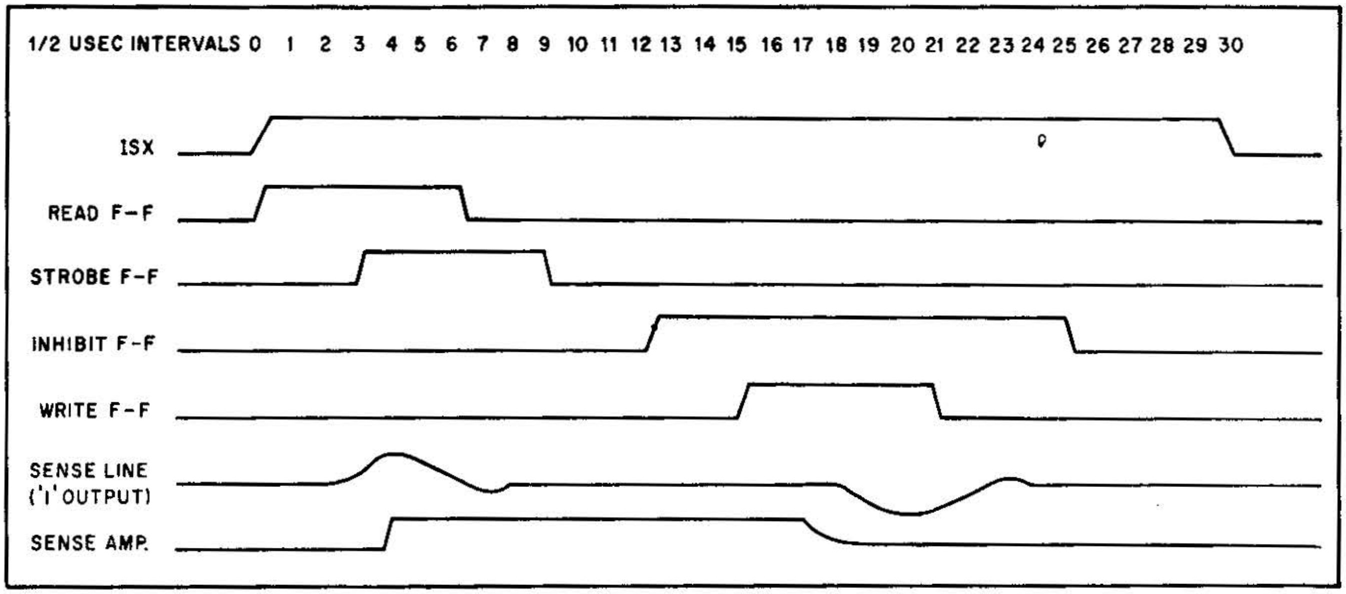


Figure 4-6. Memory Timing Waveforms: (ISX) Read-Restore Cycle

M register, and the X register is cleared. Then the ISX subcommand starts the memory cycle.

First, the read/write flip-flop is set in order to enable the sense amplifiers. Next, a read level is generated that causes the X and Y address circuits to send the read currents to the addressed cores. At the same time, the strobe pulse is applied to the sense amplifiers, and because the amplifiers are enabled, they amplify the sense-line currents. As explained earlier, the read current actually clears all the addressed cores; and any core that has to shift from a "1" to a "0" sends a "1" pulse to its sense amplifier. The binary word that is read out of the cores is transferred to the X register. The word can then be sent out of the X register to any other section.

Because the cores that were read out were all changed to "0's", the cell must be restored with its original data. The read-out word, now in the X register, must be written back in the cell. To do this, the timing and control circuits enable the inhibit drivers and X and Y address circuits to provide write currents to the cores just read (the M register is still holding the same address). The X and Y lines tend to write all "1's" in the addressed cores. But the inhibit drivers sense the word in the X register, and inhibit those core planes that should have a "0" restored. The original word is therefore written back into the cell.

4-4.7.2. Write Sequence (IXS)

The write sequence clears a selected cell and transfers the contents of the X register to that cell. Before a write memory cycle starts, the address of the cell to be written in is put in the M register and the X register is loaded with the data to be stored. Then the IXS subcommand starts the memory cycle. First the read/write flip-flop is cleared, disabling the sense amplifiers. Next the read level is generated and the X and Y address circuits send the read currents to the addressed cores. However, becaused R/W flip-flop is clear, the sense amplifiers are inoperative. Strobe will occur, but the sense amplifiers will not respond to the sense wire currents. Thus, the original word in the cell is lost. The inhibit and write levels now write the new data (from the X register) into the selected cores.

4-4.8. MEMORY CIRCUIT CODING

Control circuits:

Coded in accordance with their function. Thus, the "strobe" flip-flop generates the strobe pulse.

Bit-oriented circuits:

Assigned same number as associated X register flip-flop. Thus, inhibit driver 5 is controlled by X_5.

Address-oriented circuits:

Numbered according to addressing function:

Selection Matrix Transformers. Decode the entire X or Y portion of the address and are so numbered, for example when "M" = 173, XT17 and YT03 will produce output drive currents, because X = 17 and Y = 03.

Line Selectors. Each Y line selector is actuated by a given combination of M_1 and M_0, and each X line selector, by a given combination of M_4 and M_3. The line selectors are numbered to reflect these combinations. Thus, when "M" = 173, XS3 and YS3 are actuated, because $M_{(4,3)} = 3$ and $M_{(1,0)} = 3$.

Read/Write Drivers. Each X R/W driver is actuated by a given combination of M_6 and M_5, and each Y R/W driver, by the state of M_2. The drivers are numbered to reflect these combinations. Thus, when M = 173, R/W drivers X3 and Y1 will supply drive currents, because $M_{(6,5)} = 3$ and $M_2 = 1$.

4-5. ARITHMETIC SECTION

4-5.1. GENERAL

The arithmetic section performs all of the mathematical and data processing functions of the BI-TRAN SIX. To perform these functions, the arithmetic section contains the following:

(1) Exchange register
(2) Accumulator register
(3) Adder network
(4) Quotient register
(5) AQ register
(6) Overflow flip-flop
(7) Sign flip-flop
(8) Logic zero flip-flop
(9) Adder control register

4-5.2. EXCHANGE REGISTER

The X register is the central point for most of the binary words that pass from one section of the computer to another. It communicates with memory, I/O devices, and the A, Q, M, P, I, and C registers. It supplies the data words to the A and Q registers, and also holds the addend in addition, the subtrahend in subtraction, the multiplier in multiplication, and the divisor in division. Data transfers to and from memory are via the X Register.

The X register contains six flip-flops to hold a 6-bit binary word. When the X register is used in arithmetic, the bit at the extreme left is the sign bit. Since the BI-TRAN SIX, like most digital computers, is a fractional computer, the other five bits (XM) of the X register hold the data in fractional form with the extreme rightmost bit the least significant one. When the X register holds other words, such as operation (instruction) codes or addresses, the sign bit is used as the most significant coded bit, which in the case of storage addresses, is treated as an integer.

4-5.3. ACCUMULATOR REGISTER

The A register holds the sum derived from adding the initial contents of A and X and can transfer data to the X register. During certain arithmetic operations, however, the contents of A can be shifted right into the Q register or the contents of the Q register can be shifted left into the A register. The A register holds the sum in addition, the difference in subtraction, the most significant bits of a product in multiplication, and the remainder in division.

The A register contains six flip-flops; for numeric data, A_0 holds a sign bit and the remaining five bits (AM) hold the magnitude of the data word. Like the X register, it treats numeric data in fractional form.

4-5.4. ADDER NETWORK

The adder network is comprised of logic circuits that accomplish the addition of data between the XM and AM registers. The arithmetic functions are described in paragraph 4.6.

4-5.5. QUOTIENT REGISTER

The Q register receives its data from, and transmits it to, the X register. During certain arithmetic operations, however, the contents of Q can be shifted left into the A register, or the contents of A can be shifted right into the Q register. The Q register holds the least significant bits of a dividend in division, the quotient in division, and the least significant bits of a product in multplication.

The Q register contains six flip-flops. In division, the quotient is stored in Q. Q_0 is treated as a sign bit and the remaining five bits (QM) represent the magnitude of the number in fractional form.

4-5.6. AQ REGISTER

During multiplication, the product is formed in the AQ register, which is an 11-bit register. The most significant bits of the product appear in the AM register and the least significant bits in the QM register. The sign bit of the product is stored in A_0. For reason of program usage, the Q_0 bit duplicates the state of A_0.

During division, the dividend is held in the AQ register and the divisor in the X register. The quotient is formed in Q and the remainder in A, with Q_0 and A_0 as the sign bits of Q and A respectively.

The previous discussion illustrates the mechanics by which the adder network determines the sum of two numbers. The logic used in the BI-TRAN SIX to change the state of the AM register to effect the results indicated above is that of complementation. That is, based on the interrelationship of the bits of AM and XM, the bits of AM are complemented or left in their original state.

To fully comprehend the details of the adder logic, the following discussion on truth tables and Boolean algebra will be helpful:

Consider the case of bits A_5 and X_5. The proper sum and carry of these two can be found from the following truth table:

X_5	A_5	Sum (A_5')	Carry	Remark
0	0	0	0	—
0	1	1	0	—
1	0	1	0	Toggle A_5
1	1	0	1	Toggle A_5

Since complementing logic is used to determine the final state of the accumulator, it can be seen that the proper sum can be formed by leaving A_5 alone if $X_5 = 0$, and complementing A_5 if $X_5 = 1$. Expressed in Boolean form—

$$(1)\quad \text{Complement } A_5 = X_5$$

This states that the A_5 bit is complemented if $X_5 = 1$.

It can further be seen that a carry results if, and only if, $X_5 = A_5 = 1$. Stated in Boolean form—

$$(2)\quad \text{Carry} = X_5A_5$$

Equation (1) gives the appropriate logic to handle the least significant bit of the rightmost group of AM and XM. Similar equations are valid for the sum of A_3 and X_3 and the sum of A_1 and X_1.

Consider again the sum of the following bits of A:

$$\begin{array}{c} A_4\,A_5 \\ \underline{X_4\,X_5} \\ A_4'A_5' \end{array}$$

As has been noted above, a carry will be produced on adding bits A_5 and X_5 if $A_5 = X_5 = 1$. This carry, called an inner group carry, is added to the sum of the bits A_4 and X_4 to form the final sum $A_4'A_5'$ during phase 1 time.

A carry may be generated in forming this 2-bit sum. This carry, G_4, called a group carry, is processed during phase 2 time. For temporary expedience, write $C_5 = X_5A_5$. *(Do not confuse this notation with the C_5 flip-flop of the C-Register.)* The truth table for the sum and carry resulting from adding X_4 and A_4 and the inner group carry C_5 is as follows:

X_4	A_4	C_5	Sum (A_4')	Carry (G_4)	Remark
0	0	0	0	0	—
0	1	0	1	0	—
1	0	0	1	0	Toggle A_4
1	1	0	0	1	Toggle A_4
0	0	1	1	0	Toggle A_4
0	1	1	0	1	Toggle A_4
1	0	1	0	1	—
1	1	1	1	1	—

In the remarks column, the cases where the state of A_4 is to be changed to give proper sum are noted. In all other cases, A_4 is left untouched. The Boolean equation that characterizes this truth table is—

$$(3)\quad \text{Complement } A_4 = X_4\,(\overline{A_5} + \overline{X_5}) + \overline{X_4}\,A_5\,X_5$$

where complement A_4 will cause the state of A to be complemented. The bar over a letter indicates complementation of the value, i.e., if $X_4 = 0$, $\overline{X_4} = 1$ and if $X_4 = 1$, $\overline{X_4} = 0$.

The addition that has just been discussed occurs during phase 1 time. The group carry, G_4, generated from this addition must look at the state of the new bits A_4' A_5' of AM and the bits of XM, i.e., X_4 X_5. It can be seen from the truth tables that C_5 can also be determined from X_5 and A_5, i.e., $C_5 = X_5\,\overline{A_5'}$. Thus, the Boolean expression for G_4 in terms of the state of the AM and XM register at the beginning of phase time is:

$$G_4 = C_5\,(X_4 + \overline{A_4'}) + X_4\,\overline{A_4'}$$

or

$$(4)\quad G_4 = X_5\,\overline{A_5'}\,(X_4 + \overline{A_4'}) + X_4\,\overline{A_4'}$$

where the "primes" indicate the state of A_4 and A_5 at the beginning of phase 2 time.

A set of equations similar to (3) and (4) can be written for the sum of the bits A_2 A_3 and X_2 X_3. A complete set of these equations is given in paragraph 4-6.4.

As was indicated, a carry can be generated from the middle group of bits of AM and XM due either to the group carry, G_2, or a propagated carry. This propagated carry can be generated only if the sum of A_2' A_3' resulting from the addition during phase 1 time

$$\begin{array}{c} A_2\,A_3 \\ \underline{X_2\,X_3} \\ A_2'A_3' \end{array}$$

is equal to 11_2 and if a group carry G_4 exists (Ex. 3).

Let $P_2' = A_2' \cdot A_3'$, the propagated enable, represent the Boolean condition for this to occur. P_2 is determined only during phase 2 time; hence the primes on the A's (Ex. 3). It is easy to show that the condition for a carry to arise as a result of propagation through the middle group of bits is $P_2 \cdot G_4$, i.e., $P_2 = $ "1" and $G_4 = $ "1".

This carry will effect the leftmost bit of AM, i.e., A_1 during phase 2 time.

The addition of two numbers in AM and XM can generate a carry from the leftmost group during phase 1 time (Ex. 2) due to the addition of bits A_1 or X_1, or can be generated due to propagation of carries during phase 2 time (Ex. 3). The exact equations that characterize these facts can be found in paragraph 4-6.4.

As can be seen from Ex. 4, the all "1's" case for the sum of numbers in AM and XM can only occurs if $A_k = \overline{X_k}$ for $k = 0, 1, 2, 3, 4$; that is, the corresponding bits in these registers are complements of one another. Although this sum arises during phase 1 time, it is only sensed during phase 3 time.

4-6.3. ADDER TIMING

The addition that takes place between registers AM and XM makes use of the adder network, which has been discussed, two logic flip-flops called OV (overflow) and LZ (logic zero), and a 2-bit increasing counter, AC (adder control). The function of flip-flop OV is to capture any end carry generated on overflow addition of AM and XM, i.e., if this sum exceeds 37_8. The flip-flop LZ is used to sense the condition that the sum of the registers is 37_8.

The heart of the timing sequence for the adder timing is the AC counter. Since it is a 2-bit counter, it can assume four values—0, 1, 2 or 3. During non-adder sequences it is in a "0" condition. This permits timing pulses, T_0, to be generated. This pulse essentially permits the distributor register, D, to be increased and allows the generation of the various distributor pulses, DP_k. During certain specific distributor phases conditioned on certain instructions (see Figure 4-7), an initiate add timing pulse (INA) is generated that increases the AC counter to a "1", and this inhibits the generation of further DP_k. The D register is also held fixed until the AC register is returned to the "0" state. Thus, control is taken away from the main control and given to arithmetic control. The following clock pulses, CP, will generate timing pulses T_1, T_2 and T_3 using the AC counter as a decoding network to generate these pulses. The generation of timing pulse T_3 will cause the AC register to return to the "0" state, returning control to the D register, whereupon main control can complete the execution of its function.

The pulses T_1 (corresponding to a "1" state of AC), T_2 (corresponding to a "2" state of AC), and T_3 (corresponding to a "3" state of AC) generate the required timing for addition to take place. These timing pulses are related to the phase times described in paragraph 4-6.2. The following chart indicates what occurs during each phase time. A precise set of Boolean equations is then given to represent these results.

Phase Time	T_0	T_1	T_2	T_3
State of AC	0	1	2	3

T_0	Initiate add sequence (INA)	Increase AC to "1"
T_1	a) Add without group carries contents of AM and XM	Increase AC to "2"
	b) If end carry is generated set the overflow flip-flop (SOV)	
T_2	Add group carries to AM to complete addition. If end carry is generated, SOV.	Increase AC to "3"
T_3	If AM = 37_8, set the logic zero flip-flop (SLZ).	Increase AC to "0"

4-6.4. ADDER NETWORK BOOLEAN EQUATIONS

The Boolean equations for the adder network are given in Table 4-1. With the exception of the term ADG (= LDA + LDN + ADD + SUB), all other terms are readily found in the description in the logic timing charts in paragraph 4-7. The notation CA_5 corresponds to the statement—complement bit A_5. All the coefficients multiplying the timing pulses T_1, T_2, T_3 are assumed to be evaluated during the phase time corresponding to that generating the timing pulse.

4-6.5. MULTIPLICATION-DIVISION PROCESSES

The methods of performing multiplication and division on the BI-TRAN SIX are comparatively straightforward. Since multiplication is the easier of the two processes, it will be considered first. The assignment of the appropriate sign to the product is part of the overall multiplication algorithm. Since this computer uses magnitude and sign number representations, only multiplication of two positive numbers need be considered. It is assumed that the multiplier is in the QM register and the multiplicand in the AM register.

Since QM and AM consist of only 5 bits each, only 5 addition steps are required. The product of these two numbers will have at most 10 significant bits. The resultant product is found in the AQ register, where A_0 represents the sign bit of the product. The most significant bits of the product will be in AM and the least significant bits of the product will be in QM. Keep in mind at all times that AM is only a 5-bit register; if adding two 5-bit numbers causes an end carry, it must be captured. An example of the procedure for multiplication is given on top of page 4-16. Use is made of the elementary technique of multiplication. The results are annotated to indicate how it will be handled in the BI-TRAN SIX.

4-5.7. ARITHMETIC CONTROL LOGIC FLIP-FLOP

Three flip-flops are concerned with arithmetic algorithmic control. They are the overflow flip-flops, the sign flip-flop and the logic zero flip-flop.

4-5.7.1. Overflow Flip-Flop

The overflow flip-flop is used as a control element for many of the instructions. It captures the end carry that may occur during adder control. Within the division algorithm, it captures the most significant bit of AM, i.e. A_1, when AQ is shifted left. During the shift right subcommand, the contents of the OV flip-flop is shifted into the most significant bit of AM, i.e. A_1. It is also used for control logic during the division algorithm.

4-5.7.2. Sign Flip-Flop

The sign flip-flop is used as a control element for many of the instructions. It is set whenever $A_0 \neq X_0$. It is used to establish the proper sign of: the product in multiplication, the remainder and quotient in division, and the sum or difference in addition or subtraction. It is also used by the RAU instruction (see the logic timing charts).

4-5.7.3. Logic Zero Flip-Flop

The logic zero flip-flop is used to determine the state of the AM register during the adder control sequence. It is set when AM $= 37_8$. Based on the state of this flip-flop and that of the overflow and sign flip-flop, certain logic functions are performed.

4-5.8. ARITHMETIC OPERATIONS

The BI-TRAN SIX is basically an additive computer. It performs all arithmetic operations by means of addition. Subtraction (reduced to machine addition) is accomplished by complementary arithmetic; multiplication is carried out by a sequence of additions and shifts; and division is carried out by a sequence of subtractions and shifts.

4-5.8.1. Addition

The sum of two numbers is formed in the A register by preloading the augend in the A register (unless previously precomputed), and extracting the addend from memory and loading it into the X register. If the numbers to be added are of opposite sign, the sign flip-flop is set and this result initiates the complementing of the magnitude bits of X, i.e. XM. Through a logic network (the adder network), the contents of AM and XM are added together. Depending on the state of the sign flip-flop, the overflow flip-flop, and the logic zero flip-flop, appropriate corrections are made to the contents of A to yield the proper sum. The details of this process can be understood by studying paragraph 4.6 on arithmetic control, and by studying the appropriate logic timing sequences for the ADD instruction. (Figure 4-7.)

4-5.8.2. Subtraction

The difference of two numbers is formed in the A register by preloading the subtrahend in the A register (unless it has been previously computed), and the minuend is extracted from memory and loaded into the X register. The resultant difference is stored in the A register. The process of subtraction is reduced to the process of addition by forming the negative of the minuend. This is accomplished by complementing the sign bit of the X register, i.e., X_0. This technique is illustrated by the appropriate logic timing sequence for the SUB instruction. (Figure 4-7.)

4-5.8.3. Multiplication

When multiplication is to be performed, the multiplier, extracted from memory, is put into the Q register, and the multiplicand, initially in the A register, is put into the X register. The A register is cleared. The computer then carries out the multiplication by additions and shifting just the way binary numbers are multiplied by hand, eventually forming a signed 10-bit product in the AQ register. The least significant bit of the multiplier in the Q register determines whether or not the AM and XM registers will be added prior to the shifting operation. If that bit is a "1", there will be an addition. If it is a zero, there will not be an addition. In either case a right shift of one place will occur with the least significant bit of Q being dropped. The sign bit of A will be formed according to the appropriate rule of multiplication for magnitude and sign arithmetic (see paragraph 4-6). The sign bit of Q duplicates the sign bit of A.

4-5.8.4. Division

The dividend of a ratio of numbers (preloaded or precomputed) is formed in the AQM register and the divisor (extracted from memory) is loaded into the X register. The resultant quotient and remainder with appropriate signs are found in the Q and A registers, respectively. In manual methods, the procedure of dividing is normally one of subtracting the divisor, on a trial and error basis, successively from the dividend in a proper step-by-step procedure.

In the BI-TRAN SIX, this process of subtracting is accomplished in effect by the addition of the negative of the divisor. An initial subtraction (actually machine addition) is made to determine whether division overflow will occur, that is, to determine if the potential quotient will be greater in magnitude than 37_8 (31_{10}). If this happens, a divide error will occur and the computer will come to a halt. If no divide overflow occurs, the contents of AM are restored so that the contents of AQM are in their original state. Effectively, a sequence of left shifts of the AQM register, followed by a subtraction (machine addition), are carried out. If a proper subtraction (machine addition) occurs, the least significant bit of Q, i.e., Q_5, is set to a "1". If not, the contents of AM are restored to their original state. This sequence is repeated for five steps. The sign bits of Q and A, namely Q_0 and A_0, are set according to the appropriate rules of division for magnitude and sign arithmetic. Exact details of this procedure are illustrated in paragraph 4-6 on arithmetic control, as well as on the logic timing sequence for the divide instruction.

4-6. ARITHMETIC CONTROL

The BI-TRAN SIX uses a fractional parallel additive network to perform all additions and subtractions. It is essential to comprehend the arithmetic processes involved, particularly the algorithms that characterize them. In this system, use is made of magnitude and sign arithmetic as contrasted to the 1's and 2's complement arithmetic system used in many computers. The green A_0 switch light emphasizes the fact that the arithmetic is magnitude and sign. The combination of bits A_1, A_2, A_3, A_4, and A_5 represent the magnitude bits of a number in the accumulator (designated as AM), and A_0 defines the sign bit. If A_0 = "0" (lamp off), the number in the accumulator is positive. If A_0 = "1" (lamp on), the number is negative. Thus,

(1) $+25_{10} = +31_8 = 011\ 001$

whereas

(2) $-25_{10} = -31_8 = 111\ 001$

4-6.1. ADDITION—SUBTRACTION PROCESSES

The rules for addition and subtraction in a magnitude and sign number representation are fairly straightforward. Let the Accumulator hold one of the numbers, called $a = (A)$, to which a second number is to be added or subtracted. Let the exchange register hold the second number, called $x = (X)$. The process of subtraction can be reduced to addition by changing the sign of the subtrahend, i.e., X_0, and adding. Thus the rules need only be promulgated for addition.

Like Sign Case

(1) If a and x are of like signs, the sign of the sum is the sign of a. An end carry can result in this case.

Unlike Sign Case

If the contents of A and X are of opposite sign, the sign FF is set. Let $|a|$ and $|x|$ represent the magnitude bits of A and X, i.e. those of AM and XM respectively. *Complement the bits of XM prior to adding the contents of AM and XM.*

(2) If an end carry results from the addition process, then $|a| - |x| > 0$. The end carry must be added to the previously formed sum. The sign of the resultant sum is the same as the sign of a. The overflow FF is set to note this fact.

(3) If no end carry results from the addition process, and the resultant sum in AM is 11111_2 or 37_8, then $|a| - |x| = 0$. To get the correct sum, clear the accumulator. The logic zero FF is set to note this fact.

(4) If no end carry results from the addition process and the sum in AM is not equal to 37_8, then $|a| - |x| < 0$. The proper sum can be found by complementing AM. To obtain the correct sign for the sum, change the sign of A, i.e., complement A_0.

Overflow Case

If the signs of A and X are alike and an end carry results in addition, the resultant sum will exceed the modulus of the AM register, i.e., 31_{10} (37_8). An overflow error will occur. The effect of this is seen in example 5 below.

The following examples will illustrate the various cases:

Ex. 1.1 Like Sign Addition

	Decimal	Binary		BI-TRAN SIX Process	
A:	+14	$\boxed{0}$	01110	$\boxed{0}$	01110
X:	+ 7	$\boxed{0}$	00111	$\boxed{0}$	00111
	+21			0	10101

Ex. 1.2 Unlike Sign Subtraction

	Decimal	Binary		BI-TRAN SIX Process	
A:	−14	$\boxed{1}$	01110	$\boxed{1}$	01110
X:	+ 7	$\boxed{0}$	00111	$\boxed{1}$	00111
	−21			1	10101

Ex. 2 Unlike Sign Cases: Addition

$|a| - |x| > 0.$

	Decimal	Binary			BI-TRAN SIX Process		
A:	−14	1	01110		1	01110	
X:	+ 7	0	00111		0	11000	Note: XM has been
	− 7			Temporary Ans.	1	00110	complemented.
				End Carry		1	
				Final Sum	1	00111	
				Sign of A			

It can be easily verified that if A contains +14 decimal, and X has −7 decimal, the rule of addition gives the correct result.

Ex. 3 $|a| - |x| = 0.$

	Decimal	Binary			BI-TRAN SIX Process		
A:	−14	1	01110		1	01110	
X:	+14	0	01110		0	10001	
	0			Temporary Ans.	1	11111	Note: All 1's case
				No End Carry			
				Final Sum	0	00000	

Ex. 4 $|a| - |x| < 0.$

	Decimal	Binary			BI-TRAN SIX Process		
A:	+ 7	0	00111		0	00111	
X:	−14	1	01110		1	10001	
	− 7			Temporary Ans.	0	11000	Note: Not All 1's
				No End Carry			
				Final Sum	1	00111	Note: XM has been complemented.

Ex. 5 Overflow Case

$|a| + |x| > 31_{10}$

	Decimal	Binary			BI-TRAN SIX Process		
A:	+16	0	10000		0	10000	
X:	+18	0	10010		0	10010	
	+34					00010	"Overflow Condition"
				End Carry		1	
				Final Sum	0	00011	

4-6.2. THE ADDER NETWORK

It will be seen from the previous examples that, in forming the sum or difference of two numbers, the only process that is used is addition, i.e., addition between the registers AM and XM. The network that permits us to carry out this process in the BI-TRAN SIX is the adder network, which is the major section of Board 5.

The adder network is concerned only with the process of addition of the numbers represented by AM and XM, not with the addition or subtraction of numbers represented by the contents of A and X (which are signed). The entire algorithm for addition or subtraction makes use of the results of the adder network. To fully comprehend the process, consider the registers AM and XM as separated into three groups:

AM:	A_1	A_2	A_3	A_4	A_5
XM:	X_1	X_2	X_3	X_4	X_5

Four examples, given below, illustrate the application of the following general statements. The actual addition between these registers will take place in a three-phase sequence. During the first phase, the numbers represented by each group are added without regard to intergroup carries. The new state of AM would appear as:

AM:	A_1'	A_2'	A_3'	A_4'	A_5'
XM:	X_1	X_2	X_3	X_4	X_5

with XM remaining the same as it was before. The carries that might have been generated from the two rightmost groups are ignored during this first phase but the carry that might be generated as a result of adding A_1 and X_1 is noted. In the BI-TRAN SIX, the OV flip-flop is set to note the occurrence of this carry during phase 1.

During the second phase, the carries that were generated from the first two groups are propagated so that the proper final *adder sum* can be obtained. From the rightmost two-bit group, a carry (called a group carry) can be generated. It is denoted as G_4 (Ex. 2 and 3). The carry generated from the middle 2-bit group can only arise from two situations. During phase 1, a group carry, G_2 (Ex. 2), may have been generated. If so, the carry G_4, which enters this group during phase 2, will produce no further carry from this group. If there is no group carry, G_2, generated during phase 1 (Ex. 3), then a carry (propagated carry) can be generated from the middle group if the sum resulting from phase 1 is equal to 11_2, i.e., $A_2' = A_3' =$ "1", and if there was a group carry G_4 during phase 1.

If a carry emerges from the middle group, either as a result of a group carry, G2, or propagated carry, then it will change the state of the bit A_1. It is also possible during phase 2 to obtain a propagated carry from the leftmost group (Ex. 3). If this occurs, this fact is noted by setting the OV flip-flop.

During phase 3, the AM register is analyzed to determine if it is in an all "1's" condition. This can only occur if the bits of AM and XM are complements of one another (Ex. 4). This fact is noted by setting the logic zero (LZ) flip-flop.

The following examples illustrate the principles stated above:

Ex. 1—No Group or propagated carries generated:

AM	0	1	0	0	1
XM	1	0	1	0	1
Phase 1	1	1	1	1	0
Phase 2	1	1	1	1	0
Phase 3	1	1	1	1	0

Ex. 2—Group carries generated from each group:

AM	1	1	1	0	1
XM	1	1	0	1	1
Phase 1	0	0	1	0	0
	e.c.	G_2		G_4	
Phase 2	1	1	0	0	0
Phase 3	1	1	0	0	0

(Arrow indicates group carry. During phase 1, OV will be set to 1 due to end carry (e.c.)).

Ex. 3—Illustration of propagated carries:

AM	0	0	1	1	0
XM	1	1	0	1	0
Phase 1	1	1	1	0	0
				G_4	
Phase 2	0	0	0	0	0
	e.c.				
Phase 3	0	0	0	0	0

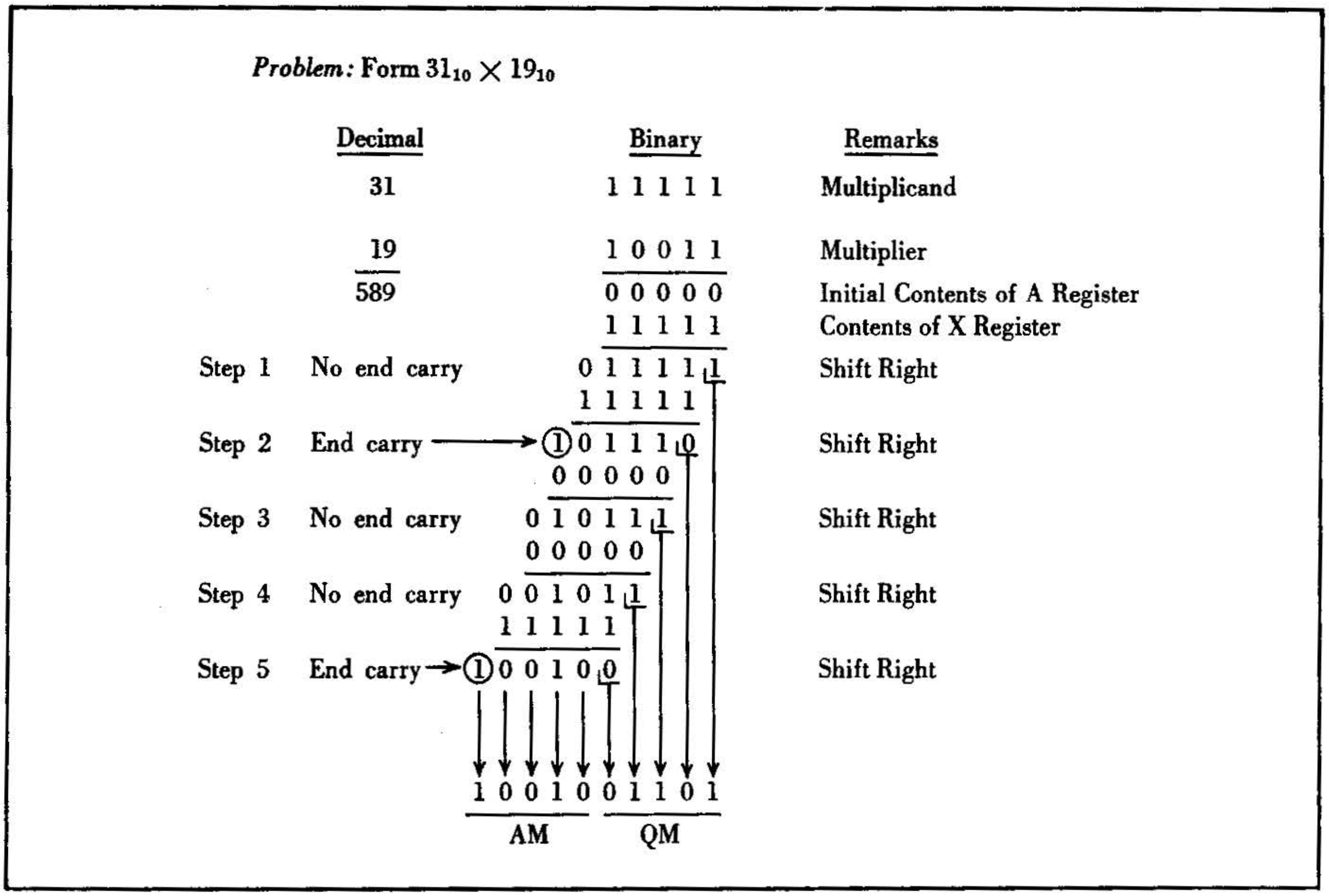
(Group carry, G_4, generates propagated carry as end carry to set OV.)

Ex. 4—Illustration of all "1's" sum:

AM	0	1	0	0	1
XM	1	0	1	1	0
Phase 1	1	1	1	1	1
Phase 2	1	1	1	1	1
Phase 3	1	1	1	1	1

(During this phase assess all "1's" and set LZ.)

Problem: Form $31_{10} \times 19_{10}$



		Decimal	Binary	Remarks
		31	1 1 1 1 1	Multiplicand
		19	1 0 0 1 1	Multiplier
		589	0 0 0 0 0	Initial Contents of A Register
			1 1 1 1 1	Contents of X Register
Step 1	No end carry		0 1 1 1 1 1	Shift Right
			1 1 1 1 1	
Step 2	End carry →		① 0 1 1 1 0	Shift Right
			0 0 0 0 0	
Step 3	No end carry		0 1 0 1 1 1	Shift Right
			0 0 0 0 0	
Step 4	No end carry		0 0 1 0 1 1	Shift Right
			1 1 1 1 1	
Step 5	End carry →		① 0 0 1 0 0	Shift Right

In steps 1, 3 and 4, after addition the sum results only in 5 bits; hence no end carry results. In these cases the OV flip-flop is in a "0" state. In steps 2 and 5, the addition produced a 6-bit sum, and the end carry is captured by OV and later shifted into A_1 for further processing.

The least significant bit of QM, Q_5, is sensed. If it is a "1", addition takes place between AM and XM; if it is "0", no addition takes places. In either case the value of Q_5 is no longer needed. The AQM register is then shifted right one position. The OV flip-flop is also used in this right shift—its contents being shifted to bit A_1.

The states of AQM and OV are shown, in brief, in the following sequence for the previous example prior to a right shift and following a right shift.

		OV	AM	QM	XM
Initial State		0	00000	10011	11111
Step 1	Before Shift	0	11111	10011	
	After Shift	0	01111	11001	
Step 2	Before Shift	1	01110	11001	
	After Shift	1	10111	01100	
Step 3	Before Shift	0	10111	01100	
	After Shift	0	01011	10110	
Step 4	Before Shift	0	01011	10110	
	After Shift	0	00101	11011	
Step 5	Before Shift	1	00100	11011	
	After Shift	1	10010	01101	

In this example, the multiplicand (initially in AM) is placed in XM; the multiplier is placed in QM. The final product is found in AQM, with A_0 representing the sign bit of the product. Thus, the least significant bits of the product are found in QM, the most significant bits in AM. Further, for system reasons the sign bit Q_0 is made to have the same value as A_0.

In order to preserve the position of the binary point of a data word after multiplication of two numbers, it is necessary to think of the arithmetic section as one that operates on fractional numbers. By this the following is meant. Consider the multiplier in Q as having its binary point between Q_0 and Q_1, the multiplicand in X as having its binary point between X_0 and X_1. Then the product of these numbers in A will have the binary point between

Table 4-1. Adder Network Boolean Equations*

$$(1)\quad \mathrm{INA} = (\mathrm{ADG} + \mathrm{RAU} + \mathrm{RSU}) \cdot \mathrm{DP}_5 + \mathrm{Q}_5 \cdot \mathrm{MPY} \cdot \mathrm{DP}_{14} + \mathrm{DIV} \cdot (\mathrm{DP}_3 + \mathrm{DP}_5 + \mathrm{DP}_{10}) + \overline{\mathrm{OV}} \cdot \mathrm{DIV} \cdot \mathrm{DP}_{14}$$

$$(2)\quad \mathrm{CA}_5 = \mathrm{X}_5 \cdot \mathrm{T}_1$$

$$\mathrm{CA}_4 = [\mathrm{X}_4 \cdot (\overline{\mathrm{A}_5} + \overline{\mathrm{X}_5}) + \overline{\mathrm{X}_4} \cdot \mathrm{A}_5 \cdot \mathrm{X}_5] \cdot \mathrm{T}_1$$

$$\mathrm{CA}_3 = \mathrm{X}_3 \cdot \mathrm{T}_1 + \mathrm{G}_4 \cdot \mathrm{T}_2$$

$$\mathrm{CA}_2 = [\mathrm{X}_2 \cdot (\overline{\mathrm{A}_3} + \overline{\mathrm{X}_3}) + \overline{\mathrm{X}_2} \cdot \mathrm{A}_3 \cdot \mathrm{X}_3] \cdot \mathrm{T}_1 + \mathrm{G}_4 \cdot \mathrm{A}_3 \cdot \mathrm{T}_2$$

$$\mathrm{CA}_1 = \mathrm{X}_1 \cdot \mathrm{T}_1 + (\mathrm{P}_2 \cdot \mathrm{G}_4 + \mathrm{G}_2) \cdot \mathrm{T}_2$$

$$(3)\quad \mathrm{SOV} = \mathrm{X}_1 \cdot \mathrm{A}_1 \cdot \mathrm{T}_1 + \mathrm{A}_1 \cdot (\mathrm{P}_2 \cdot \mathrm{G}_4 + \mathrm{G}_2) \cdot \mathrm{T}_2$$

$$(4)\quad \mathrm{SLZ} = \mathrm{A}_1 \cdot \mathrm{A}_2 \cdot \mathrm{A}_3 \cdot \mathrm{A}_4 \cdot \mathrm{A}_5 \cdot \mathrm{T}_8$$

$$(5)\quad \mathrm{G}_4 = \mathrm{X}_5 \cdot \overline{\mathrm{A}_5} \cdot (\mathrm{X}_4 + \overline{\mathrm{A}_4}) + \mathrm{X}_4 \cdot \overline{\mathrm{A}_4} \quad \text{(evaluated at } \mathrm{T}_2 \text{ phase time)}$$

$$(6)\quad \mathrm{G}_2 = \mathrm{X}_3 \cdot \overline{\mathrm{A}_3} \cdot (\mathrm{X}_2 + \overline{\mathrm{A}_2}) + \mathrm{X}_2 \cdot \overline{\mathrm{A}_2} \quad \text{(evaluated at } \mathrm{T}_2 \text{ phase time)}$$

$$(7)\quad \mathrm{P}_2 = \mathrm{A}_2 \cdot \mathrm{A}_3 \quad \text{(evaluated at } \mathrm{T}_2 \text{ phase time)}$$

* These Boolean equations represent only the input terms contributing to the complementation of the bits of the AM register for the adder control. There are other inputs that are not shown here that also contribute to the toggling logic of AM.

A_0 and A_1. By use of scaling procedures in the programming of a given problem, the binary point can be assigned to any position. Similar remarks apply in dividing a number (the dividend), found in AQM, by a number (the divisor), found in X.

The algorithm for division is somewhat more complex than that for multiplication. Since the BI-TRAN SIX is only concerned with magnitude and sign arithmetic, the rules for establishing the signs of the quotient and remainder are fairly straightforward. The instruction sequence will concern itself with these assignments. Division overflow, however, will be considered in the following paragraphs.

The essential feature that characterizes division in the BI-TRAN SIX is the requirement that use be made of successive additions of negative numbers (using, in some degree, the addition algorithms previously developed for this purpose). If the resultant sum (actual difference) is positive or zero, the addition (actual subtraction) process is acceptable, that is, the divisor has been acceptably subtracted once from the most significant bits of the dividend. To note this fact, bit Q_5 of the Q register is set equal to "1". If this sum is negative, it implies that the divisor is larger than the most significant bits of the current dividend. To note this fact, Q_5 is set equal to "0" and the dividend is restored to its previous value. Upon completion of this step, the dividend is shifted left one place, including the information bit stored in Q_5. The process of subtraction is repeated until the appropriate number of division steps has been completed. In the BI-TRAN SIX, this corresponds to five division steps. This algorithm is known as a restore-type division process.

The dividend in the BI-TRAN SIX is held in the AQ register. The divisor is held in X. The sign bits of AQ and X are sensed to determine the proper signs of the quotient (placed in Q_0) and the remainder (placed in A_0). The actual division takes place between a 10-bit dividend in AQM and a 5-bit divisor in XM. The division process will place a 5-bit quotient in QM with the proper sign in Q_0, and a 5-bit remainder in AM with the proper sign in A_0.

Two anomalous cases can arise for which one algorithm can be devised. Clearly division by zero is improper. Furthermore, under certain circumstances it is possible to ask for the quotient of two numbers where the resulting quotient will be greater than the allowable word range. For example, if 124_{10} is divided by 3 (the AQ can hold a number such as 124_{10} as it is 10 bits in length), a quotient of 41 should be obtained. This is greater than 5 bits permit. This fact should be detected early in the procedure of division. These two anomalous cases give rise to a condition known as division overflow.

Both of these cases can be treated as one. An intial subtraction (addition of negaive numbers in the BI-TRAN SIX) is made; if this subtraction yields a non-negative number, then division overflow will occur; if not, the division algorithm can be followed without further concern for overflow. In the examples that follow, subtraction is used to illustrate the point in question.

Case 1. Division by 0

	AM	QM
	10010	10111
XM	00000	
	10010	

Non-negative difference, 10010, in preliminary test indicates overflow will occur.

Case 2. Overflow case where divisor is not equal to 0

Form $124_{10} \div 3$.

	AM	QM
	00011	11100
XM	00011	
	00000	

The preliminary subtraction produces a non-negative difference indicating overflow will occur.

As was stated above, the division process in the BI-TRAN SIX makes use of the addition of negative numbers. This is effectively treated by use of a proper divide algorithm. To achieve the correct results, certain rules must be followed. They are merely stated here. Examples are given as Division Problems 1 and 2 to illustrate the use of these rules. To accomplish the process of addition of negative numbers, the division algorithm makes use of the complement of the contents of XM. Let *am* represent the contents of AM and *xm* represent the contents of XM.

Rules to be followed:

(1) Complement bits of XM. Call the new contents xm′.

(2) Division overflow test: If the addition of am and xm′ causes an end carry or the final state of AM $= 37_8$, overflow will occur. Restore the contents of AM to its original state. (See Rule 5.1 below.)

(3) Repeat Rules 4, 5 and 5.1 or 5.2 for a total of 5 division steps.

(4) Shift the contents of AQM one place to the left. In the BI-TRAN SIX, the left shift is open-ended, i.e., Q_5 will be left in a "0" state regardless of its previous state. If A_1 is in a "1" state prior to shifting, this bit is captured by the OV flip-flop in shifting left. In this case if OV $= 1$, AM is to be increased by 1.

(5) Add am and xm′.

(5.1) If the addition of am and xm′ causes no end carry and AM is not equal to 37_8, then am $-$ xm < 0. Hence the subtraction should not take place. Restore the contents of AM to its original state. This is done as follows:

(a) Complement AM; in shorthand notation, CAM.

(b) Add AM and XM.

(c) CAM.

In step (c) AM is restored to its original state.

(5.2) If addition of am and xm′ causes an end carry or produces AM $= 37_8$, then am $-$ xm $\geqq 0$, and a permissive subtraction has occurred. Set $Q_5 = 1$. In either of these cases, increase AM by 1 and ignore any end carries that may occur at this point. For the case where AM $= 37_8$, after an addition am $=$ xm. For example:

	Original Data	Complemented Data
AM:	00010	00010
XM:	00010	11101
	00000	11111
		1
		00000
	Subtraction	Addition

If an end carry occurs, it is captured in OV. If AM $= 37_8$, then LZ is set to 1. For hardware logic simplicity, the OV flip-flop is set if LZ $= 1$. The state of OV is sensed to determine whether the addition (actual subtraction) procedure is permissive, and hence that AM is to be increased by 1.

Division Problem 1: Form $59_{10} \div 3$

Rule	AM	QM	XM		Remarks
	00 001	11 011	00 011		Initial State of Registers
(1)	00 001	11 011	11 100		Complement XM
(2)	11 100			Add XM	Overflow Test
	11 101			SUM	No End Carry and AM $\neq 37_8$
	00 010			CAM	
	11 100			Add XM	
	11 110			SUM	
	00 001			CAM	Restored AM
(4)	00 011	10 110			Shift AQM
(5)	11 100			Add XM	Step No. 1
(5.2)	11 111			SUM	AM $= 37_8$
	1			Increase AM by 1	
	00 000	10 111		SUM	Set $Q_5 = 1$
(4)	00 001	01 110			Shift AQM
(5)	11 100			Add XM	Step No. 2
(5.1)	11 101			SUM	No End Carry and AM $\neq 37_8$
	00 010			CAM	
	11 100			Add XM	
	11 110			SUM	
	00 001			CAM	Restored AM
(4)	00 010	11 100			Shift AQM
(5)	11 100			Add XM	Step No. 3
(5.1)	11 110			SUM	No End Carry and AM $\neq 37_8$
	00 001			CAM	
	11 100			Add XM	
	11 101			SUM	
	00 010			CAM	Restored AM
(4)	00 101	11 000			Shift AQM
(5)	11 100			Add XM	Step No. 4
(5.2)	00 001			SUM	End Carry
	e.c. 1			Increase AM by 1	
	00 010	11 001		SUM	Set $Q_5 = 1$
(4)	00 101	10 010			Shift AQM
(5)	11 100			Add XM	Step No. 5
(5.2)	00 001			SUM	End Carry
	e.c. 1			Increase AM by 1	
	00 010	10 011		SUM	Set $Q_5 = 1$
	00 010	10 011			Final Results
	AM	QM			

4-7. LOGIC TIMING

The logic timing that permits the BI-TRAN SIX to carry out instructions is controlled by the D register, AE flip-flop, and I register. It consists of two distinct phases: (1) acquisition, and (2) execution (Figure 4-7). Each phase, in turn, consists of sixteen distributor phases (numbered octally 0 through 17) issued as distributor pulses. During acquisition, various subcommands (listed in Section VII) are issued to acquire an instruction; during execution, various subcommands are issued to execute, or carry out the instruction. From Figure 4-7, it can be seen that subcommands are not issued during all acquisition and execution distributor phases for any given instruction.

Command Generation. In the BI-TRAN SIX, commands are the decoded outputs of the I register and the AE flip-flop. A command usually represents one instruction, and will be labeled as such. These are indicated on Board 1. Thus, the command MPY is present ("0" volt) only when the instruction code for "multiply" is in the I register: The following commands are exceptions to this rule:

Command ADG (addition group) is present for instructions: LDA, LDN, ADD and SUB

Command ARG (arithmetic group) is present for instructions: LDA, LDN, ADD, SUB, RAU, MPY, and DIV

Also, for commands to be generated, the AE flip-flop must be set, i.e., in the execution phase.

Subcommand Generation. Subcommands are the control section output pulses that cause the computer to acquire and execute instructions. All operations in the computer are initiated by subcommands. Subcommands are generated two different ways: during the acquisition phase when the AE flip-flop is in a clear state, subcommands are produced by ANDing the AE output with certain distributor pulses.

***Division Problem** 2*: **Form $527_{10} \div 31$**

In problem 1, no illustration arose of the situation where OV was set to 1 by shifting AQ left one place (Rule 4). This problem illustrates this. The case where AM = 37_8 (Rule 5.2) also occurs. The only steps illustrated are those relating to these two rules.

Rule	OV	AM	QM	XM		Remarks
	0	10 000	01 111	11 111		Initial State of Register
	0	10 000	01 111	00 000		Complement XM
	0	10 000	01 111			State of registers after divide overflow test.
(4)	1	00 000	11 110			Shift AQM
(5)		00 000			Add XM	Step No. 1
		00 000			SUM	
(4)		1			Increase AM by 1	OV = 1
		00 001	11 111		SUM	Set $Q_5 = 1$ (note that the OV is cleared by DP_{16})
	0	11 111	10 000			Shift AQM 4 places
(5)		00 000			Add XM	Step No. 5
(5.2)		11 111			SUM	AM = 37_8
		1			Increase AM by 1	
	1	00 000	10 001		SUM	Set $Q_5 = 1$ Note OV= 1 This is of no consequence at this point.
	0	00 000	10 001			Final results
		(0)	(17_{10})			
		AM	QM			

During the execution phase, subcommands are produced by ANDing distributor pulses with command levels and, in some cases, with levels from other registers. Any given subcommand can usually be generated by several instructions. Subcommands are labeled according to the action they cause to occur. Thus, subcommand TQX causes data to be transferred from Q to X.

The logic timing chart (Figure 4-7) illustrates the sequence of subcommands for each instruction.

4-7.1. ACQUISITION PHASE

The acquisition phase is identical for all instructions. No matter what instruction is to be performed, three major functions are carried out during acquisition:

(1) The first word of the instruction, the operation code, is obtained from storage, tested to make sure that it is a legal code, and then transferred to the I register. This function is carried out during distributor counts DP_0 to DP_4.

(2) The computer is prepared for the next instruction (the P register is incremented) at DP_5.

(3) By incrementing the M register so as to make an odd number, the second word, the operand address, is obtained from storage, transferred to X, then to M. This operand address may be a storage address, a jump address, a shift count or an I/O load/unload count. This is carried out during DP_{12} through DP_{17}. During DP_{12} and DP_{13}, the operand address is obtained from storage and transferred to X. At every DP_{16}, certain housekeeping subcommands are executed (the overflow, logic zero, and sign flip-flops are cleared). At DP_{17}, the operand address is transferred from X to M. Also at DP_{17}, the AE flip-flop is complemented, which places the trainer in the execution phase. When this occurs, the AE flip-flop switch-light will turn on.

4-7.2. EXECUTION PHASE

The subcommands carried out during execution vary with the particular instruction being performed. The execution phases of two typical instructions are described below.

4-7.2.1. Add Instruction (44)

The following functions are carried out during the execution phase of the add instruction (presuming that data exists in the accumulator):

(1) The operand, which in this case is the addend, is transferred from storage to the X register. This occurs at DP_1.

(2) At DP_3, the signs of the augend and addend are compared. If they are different the sign flip-flop is set.

(3) Preliminary corrections for addition are made at DP_4.

(4) At DP_5, the AC register inhibits the D register and generates timing pulses T_1 through T_3, during which time the contents of the XM and AM registers are added by the adder network leaving the result in AM. After this addition is completed, the logic timing function is returned to the D register.

(5) End corrections to complete the arithmetic addition algorithm are made at DP_6 and DP_7.

At the completion of the execution phase, the AE flip-flop is complemented to establish the acquisition phase of the next instruction.

To obtain a good understanding of how acquisition and execution are carried out, it is suggested that a simple program be carried out while studying the logic timing chart (Figure 4-7). Load the program shown on page 4-26, which adds 2 and —5, into the computer.

(1) Press the CLEAR switch and set the MODE switch to AE. Now, press the START switch to acquire the LDA instruction. Note that as explained in paragraph 4-7.1., the operation code (40) has been placed in the I register, the operand address (06) has been placed in the M register and the X register; and the P register has been incremented from 00 to 02 for the next instruction acquisition phase. Note also, that the AE flip-flop was set at the end of the acquisition phase to place the computer in the execution phase.

(2) Press the START switch again to carry out the execution phase of the LDA instruction. Note that the data from memory address 06 has been transferred to the A register. Actually, there is not a direct transfer from memory to the A register. First, the data is transferred to the X register (DP_1), and then the contents of the X register is added to the contents of the A register (DP_5). Since there is nothing in the A register (cleared at DP_0), adding the contents of the X and A registers is, effectively, a transfer of data from X to A.

The AE flip-flop was cleared at the end of the execution phase. The computer is now set up to acquire the next instruction (44).

(3) Place the MODE switch in the DISTRIBUTOR position. Now the effect of each distributor pulse during the acquisition and execution phases can be studied.

DIST. PULSE	DP_0	DP_1	DP_2	DP_3	DP_4	DP_5	DP_6	DP_7	DP_{10}	DP_{11}	DP_{12}	DP_{13}	DP_{14}	DP_{15}	DP_{16}	DP_{17}
ACQUISITION EXECUTION	CLM ΔTPM	CLX ΔISX	CLI CLIE	TXI	IF INSTRUCTION ERROR, SIE AND STOP(*1)	INCP (*2)					SM_0	CLX ΔISX			CLOV CLLZ CLSI	ΔCAE(*6) CLEL CLM ΔTXM ΔTI_0M_6
		INCREASE "D" AFTER EACH DIST. PULSE									INCREASE "D" AFTER EACH DIST. PULSE					
SBE (10) SBO (11) SUB-ROUTINE JUMP (00100X)	CLM ΔTPM				IF I_0 = 1 AND JUMP SWITCH IS SET, STOP (*9)	CLP	TXP	CLX	TMX	CLM	SM_0			IXS	CLOV CLLZ CLSI	ΔCAE(*6) CLEL
UNE (12) UNO (13) UNCON-DITIONAL JUMP (00101X)					IF I_0= 1 AND JUMP SWITCH IS SET, STOP (*9)	CLP	TXP								CLOV CLLZ CLSI	ΔCAE(*6) CLEL
NAE (14) NAO (15) SIGN TEST JUMP (00110X)					IF I_0= 1 AND JUMP SWITCH IS SET, STOP (*9)	IF A < 0, CLP IF A ≥ 0, CONTINUE (*3)	IF A<0, TXP IF A ≥ 0, CONTINUE (*3)								CLOV CLLZ CLSI	ΔCAE(*6) CLEL
NZE (16) NZO (17) ZERO TEST JUMP (00111X)					IF I_0= 1 AND JUMP SWITCH IS SET, STOP (*9)	IF A ≠ 0, CLP IF A = 0, CONTINUE (*3)	IF A ≠ 0, TXP IF A = 0, CONTINUE (*3)								CLOV CLLZ CLSI	ΔCAE(*6) CLEL
MNI (20) (21) MANUAL INPUT (01000X)		CLX			STOP (*9)					IXS	IF C=0, ΔΔSD16 IF C≠0, CONTINUE	DEC	ΔINCM	ΔΔ CLD	CLOV CLLZ CLSI	ΔCAE(*6) CLEL
MNO (22) (23) MANUAL OUTPUT (01001X)		CLX ΔISX			STOP (*9)						IF C=0, ΔΔ SD16 IF C≠0, CONTINUE	DEC	ΔINCM	ΔΔCLD	CLOV CLLZ CLSI	ΔCAE(*6) CLEL
EXI (24) (25) EXTERNAL INPUT (01010X)		CLX		INL SEL	STOP (*9)	TEX RESUME (*8)				IXS	IF C=0, ΔΔ SD16 IF C≠0, CONTINUE	DEC	ΔINCM	ΔΔCLD	CLOV CLLZ CLSI	ΔCAE(*6) CLEL
EXO (26) (27) EXTERNAL OUTPUT (01011X)		CLX ΔISX		INU SEL	STOP (*9)	RESUME (*8)					IF C=0, ΔΔSD16 IF C≠0, CONTINUE	DEC	ΔINCM	ΔΔCLD	CLOV CLLZ CLSI	ΔCAE(*6) CLEL

*SEE NOTES ON SHEET 4

Figure 4-7. Logic Timing Chart (Sheet 1 of 4)

DIST.PHASE	DP$_0$	DP$_1$	DP$_2$	DP$_3$	DP$_4$	DP$_5$	DP$_6$	DP$_7$	DP$_{10}$	DP$_{11}$	DP$_{12}$	DP$_{13}$	DP$_{14}$	DP$_{15}$	DP$_{16}$	DP$_{17}$
ACQUISITION EXECUTION	CLM △TPM	CLX △ISX	CLI CLIE	TXI	IF INSTRUCTION ERROR, SIE AND STOP(*1)	INCP (* 2)					SM$_0$	CLX △ I3X			CLOV CLLZ CLSI	ΔCAE(*6) CLEL CLM △ TXM △ TI$_0$ M$_6$
		INCREASE "D" IN EACH DIST. PHASE									INCREASE "D" IN EACH DIST. PHASE					
LDA (40) (41) LOAD ACCUMULATOR (10000X)	CLA	CLX Δ ISX		IF $A_0 \neq X_0$, SSI	IF SI = 1 CXM	INA	IF $SI \cdot \overline{OV} \cdot \overline{LZ} = 1$, CAM AND CA$_0$	IF $\overline{SI} \cdot OV = 1$, SAV AND STOP (*5) IF $SI \cdot LZ = 1$ CLA IF OV = 1, INCA							CLOV CLLZ CLSI	ΔCAE(*6) CLEL
LDN (42) (43) LOAD ACCUMULATOR NEGATIVE (10001X)	CLA	CLX Δ ISX	CX$_0$	IF $A_0 \neq X_0$, SSI	IF SI = 1 CXM	INA	IF $SI \cdot \overline{OV} \cdot \overline{LZ} = 1$ CAM AND CA$_0$	IF $\overline{SI} \cdot OV = 1$, SAV AND STOP (*5) IF $SI \cdot LZ = 1$ CLA IF OV = 1, INCA							CLOV CLLZ CLSI	ΔCAE(*6) CLEL
ADD (44) (45) ADD (10010X)		CLX ΔISX		IF $A_0 \neq X_0$, SSI	IF SI = 1 CXM	INA	IF $SI \cdot \overline{OV} \cdot \overline{LZ} = 1$ CAM AND CA$_0$	IF $\overline{SI} \cdot OV = 1$, SAV AND STOP (*5) IF $SI \cdot LZ = 1$ CLA IF OV = 1, INCA							CLOV CLLZ CLSI	ΔCAE(*6) CLEL
SUB (46) (47) SUBTRACT (10011X)		CLX Δ ISX	CX$_0$	IF $A_0 \neq X_0$, SSI	IF SI = 1 CXM	INA	IF $SI \cdot \overline{OV} \cdot \overline{LZ} = 1$ CAM AND CA$_0$	IF $\overline{SI} \cdot OV = 1$, SAV AND STOP (*5) IF $SI \cdot LZ = 1$ CLA IF OV = 1, INCA							CLOV CLLZ CLSI	Δ CAE(*6) CLEL
RAU (50) (51) REPLACE ADD UNITY (10100X)	CLA	CLX Δ ISX	SA$_5$			INA	IF $X_0 \cdot \overline{OV} + \overline{X_0} \cdot OV = 1$, SA$_0$ IF $X_0 \cdot OV = 1$, SSI	CLX	TAX	IXS	CLM$_0$	CLX Δ ISX	IF SI = 1, SX$_5$	IXS	CLOV CLLZ CLSI	ΔCAE(*6) CLEL
RSU (52) (53) REPLACE SUBTRACT UNITY (10101X)	CLA	CLX ΔISX	SA$_0$ SA$_5$	IF $A_0 \neq X_0$, SSI	IF SI = 1, CXM	INA	IF $SI \cdot \overline{OV} \cdot \overline{LZ} = 1$, CAM AND CA$_0$	CLX IF $\overline{SI} \cdot OV = 1$, SAV AND STOP (* 5) IF $SI \cdot LZ = 1$, CLA IF OV = 1, INCA	TAX					IXS	CLOV CLLZ CLSI	ΔCAE(*6) CLEL
MPY (54) (55) MULTIPLY (10110X)	CLQ SC	CLX ΔISX		IF $A_0 \neq X_0$, SSI			TXQ SC5	CLX	TAX		CLA	DEC IF SI = 1, SA$_0$ AND SQ$_0$	IF $Q_5 = 1$, INA	AQR	CLOV CLLZ CLSI IF $C \neq 0$, △△SD13	ΔCAE(*6) CLEL
DIV (56) (57) DIVIDE (10111X)	CLQ$_0$ SC	CLX Δ ISX	CXM	INA IF $A_0 \neq X_0$, SSI	IF OV + LZ = 1, SDV AND STOP (* 4) IF OV + LZ = 0, CAM IF SI = 1, SQ$_0$	INA	IF OV = 0, CAM SC5 CLLZ	AQL	INA	IF LZ = 1, SOV	IF OV = 0, CAM IF OV = 1, SQ$_5$	DEC	IF OV = 0, INA IF OV = 1, INCA	IF OV = 0, CAM	CLOV CLLZ CLSI IF $C \neq 0$, △△ SD7	ΔCAE(*6) CLEL

* SEE NOTES ON SHEET 4

Figure 4-7. Logic Timing Chart (Sheet 2 of 4)

DIST.PHASE	DP_0	DP_1	DP_2	DP_3	DP_4	DP_5	DP_6	DP_7	DP_{10}	DP_{11}	DP_{12}	DP_{13}	DP_{14}	DP_{15}	DP_{16}	DP_{17}
ACQUISITION EXECUTION	CLM ΔTPM	CLX ΔISX	CLI CLIE	TXI	IF INSTRUCTION ERROR SIE AND STOP (*1)	INCP (*2)					SM_0	CLX Δ ISX			CLOV CLLZ CLSI	ΔCAE(*6) CLEL CLM ΔTXM ΔTI_0M_6
		INCREASE "D" WITH EACH DIST. PHASE								INCREASE "D" WITH EACH DIST. PHASE						
STA (60) (61) STORE A (11000X)		CLX							TAX					IXS	CLOV CLLZ CLSI	ΔCAE(*6) CLEL
STQ (62) (63) STORE Q (11001X)		CLX									TQX			IXS	CLOV CLLZ CLSI	ΔCAE(*6) CLEL
SRE (64) SRO (65) SHIFT AQ RIGHT (11010X)	SC IF $I_0 = 1$, CLQ		TXC CLC_6								IF C = 0, ΔΔSD16 IF C ≠ 0, CONTINUE	DEC		AQR	CLOV CLLZ CLSI IF C ≠ 0, ΔΔSD7	ΔCAE(*6) CLEL
SLE (66) SLO (67) SHIFT AQ LEFT (11011X)	SC		TXC CLC_6 IF $I_0 \overline{Q_0} = 1$, $CL\ A_0$ IF $I_0\ Q_0 = 1$, SA_0								IF C = 0, ΔΔSD16 IF C ≠ 0, CONTINUE	DEC	AQL		CLOV CLLZ CLSI IF C ≠ 0, ΔΔSD7	ΔCAE(*6) CLEL
LDC (70) (71) LOAD CONSECUTIVE (11100X)		CLX ΔISX											Δ INCM	IXS	CLOV CLLZ CLSI	ΔCAE(* 6) CLEL
LCT (72) (73) LOAD COUNTDOWN REGISTER (11101X)	SC		TXC TI_0C_6												CLOV CLLZ CLSI	ΔCAE(*6) CLEL
CAS (74) (75) CHANGE ACCUMULATOR SIGN (11110X)							CA_0								CLOV CLLZ CLSI	ΔCAE(*6) CLEL
STP (76) (77) STOP (11111X)					STOP (*9)	CLP	TXP								CLOV CLLZ CLSI	ΔCAE(*6) CLEL

*SEE NOTES ON SHEET 4

Figure 4-7. Logic Timing Chart (Sheet 3 of 4)

ADDER CONTROL TIMING				
PHASE				
T_0	T_1	T_2	T_3	T_0
INA (SETS AC = 1, WHICH INHIBITS THE D REGISTER UNTIL AC = 0) (*7)	ADD, WITHOUT GROUP CARRIES, CONTENTS OF AM AND XM	ADD GROUP CARRIES TO AM TO COMPLETE ADDITION. IF END CARRY RESULTS, SOV	IF AM = 37_8, SLZ	ENABLE D REGISTER
INCREASE ADDER CONTROL REGISTER WITH EACH PHASE				

NOTES:

*1 UNLESS ERROR SWITCH IS IN BYPASS POSITION

*2 IF RPT SWITCH IS ON, THE TRAINER WILL NOT INCP IF AE OR INST. MODE

*3 TREAT CASE $|A| = 0$ AND $A_0 = 1$ AS $A = 0$

*4 IF STOP SWITCH IS IN BYPASS POSITION, THE TRAINER WILL NOT STOP ON DIVIDE OVERFLOW

*5 FOR ADDITION GROUP, IF STOP SWITCH IS IN AV BYPASS OR BYPASS POSITIONS THE TRAINER WILL NOT STOP ON OVERFLOW

*6 IF RPT SWITCH IS ON, IT WILL NOT CAE IN DIST. OR AE MODE

*7 IF IN DISTRIBUTOR MODE REPEAT, INA WILL NOT SET AC = 1

*8 PERIPHERAL EQUIPMENT GENERATES RESUME TO RESTART COMPUTER

*9 STOPS WITH D = 05

Δ = THE FUNCTION IS : DELAY 15μSEC

ΔΔ = THE FUNCTION IS : DELAY 30μSEC

Figure 4-7. Logic Timing Chart (Sheet 4 of 4)

SYMBOLIC PROGRAM			MACHINE-CODED PROGRAM		
Program Address	Operation Code	Operand Address	Program Address	Operation Code	Operand Address
a_0	LDA	x_0	00	40	06
a_2	ADD	x_1	02	44	07
a_4	STP	00	04	76	00

DATA		MACHINE CODED DATA	
Data Address	Data Word	Data Address	Data Word
x_0	2	06	02
x_2	—5	07	45

(4) Press the START switch and observe what occurs at DP$_0$. Then press the START switch again to observe what happens at DP$_1$; continue doing this through DP$_{17}$. Note that with the execution of each subcommand the DISTRIBUTOR register is incremented by one. The following will be observed at each distributor pulse during acquisition;

DP$_0$—The address in the P register (02) was transferred to the M register.

DP$_1$—The instruction code stored at address 02 (ADD instruction 44) was transferred to the X register.

DP$_2$—The I register was cleared. If the INSTRUCTION ERROR flip-flop had been set, it, too, would have been cleared at this time.

DP$_3$—The ADD instruction was transferred from the X register to the I register.

DP$_4$—The instruction code was checked to determine if it was a legal code.

DP$_5$—The P register was incremented from 02 to 04, i.e., the address of the next sequential instruction.

DP$_6$ through **DP$_{11}$**—No subcommands were generated.

DP$_{12}$—M$_0$ on the M register was set. This sets up the computer to acquire the operand address of the instruction.

DP$_{13}$—The operand address is read from memory and loaded into the X register.

DP$_{14}$ and **DP$_{15}$**—No subcommands were generated.

DP$_{16}$—If the sign, overflow, or logic zero flip-flop were set, they would be cleared at this time.

DP$_{17}$—The operand address is transferred to the M register. Also, the AE flip-flop was complemented to set up the computer to carry out the execution phase. Note that the X register has not been cleared. It still has the operand address.

(5) The computer is now ready to carry out the execution phase of the ADD instruction. Press the START switch to observe what occurs at DP$_0$, and press the switch once for each distributor pulse through DP$_{17}$. The following will be observed at each distributor phase during execution:

DP$_0$—No subcommands were generated.

DP$_1$—The data (—5) or (45) stored at memory address 07 was transferred to the X register. Note that at this point, the number 02 is in the A register and (45) is in the X register; and the ADD instruction (44) is in the I register.

DP$_2$—No subcommands were generated.

DP$_3$—The signs of the addend and augend are examined. Since the signs of the two numbers to be added are different, the sign flip-flop set.

DP$_4$—Preliminary corrections for arithmetic additions were made. In this case, the contents of the XM register were complemented because the sign flip-flop equals "1" (set).

DP$_5$—At this time, logic timing is relinquished by the D register and taken over the adder control register. From T$_1$ to T$_3$ (Figure 4-7, Sheet 4), the adder sequence is carried out. When the adder sequence is completed ($AC = 0$), the D register again assumes control of logic timing.

DP$_6$—End corrections for arithmetic additions were started. In this case, both the sign and the magnitude contents of the A register were complemented because $SI \cdot \overline{OV} \cdot \overline{LZ} = 1$.

DP_7—End corrections for arithmetic additions were completed. Since none of the logic conditions specified for DP_7 exist, with this combination of numbers, no subcommands were carried out during this distributor pulse.

DP_{10} through **DP_{15}**—No subcommands were generated.

DP_{16}—Since the Sign flip-flop was set, it was cleared during this distributor pulse.

DP_{17}—The AE flip-flop was complemented to set up the computer for the next instruction acquisition pulse.

(6) Set the MODE switch to INSTRUCTION or PROGRAM and press the START switch to carry out the last program instruction (stop). Note that the BI-TRAN SIX stops with D = 5, in the execution phase of the stop instruction.

4-7.2.2. Loop Sequence Instructions

Eight of the computer's instructions contain loop sequences: manual input, 20; manual output, 22; external input, 24; external output, 26; multiply, 54; divide, 56; shift AQ right, 64; and shift AQ left, 66. During a loop sequence, a series of distributor pulses and their subcommands are repeated n times, where n equals the number contained in the C register. When the C register has counted down to zero, the loop is broken and the execution phase of the instruction is then carried out through DP_{17} to complete the instruction execution phase.

The shift right instruction (64) will be used to describe a loop sequence. Two primary functions are carried out during the execution phase of the shift right even instruction:

As noted above, the X register still contains the operand address which in this case is the shift count.

(1) The shift count is transferred from the X register to the C register. This is accomplished during DP_1 through DP_3.

(2) A loop sequence is performed to shift the magnitude contents of the AQ register right the number of places specified by the C register.

To see how a loop sequence is carried out, perform the following procedure while studying the logic timing chart (Figure 4-7):

(1) Load the shift right even instruction (64) into address 00.

(2) Load the number 5, which is the shift count, into address 01.

(3) Load the stop instruction (76) into address 02.

(4) Press the CLEAR switch.

(5) Set the MODE switch to AE.

(6) Set the number 02 into the A register.

(7) Press the START switch. The acquisition phase of the shift right instruction will be carried out and the trainer is now set up to carry out the execution phase. The operand address of the instruction (05) has been transferred from memory to the X and M registers. Since in this case it is not a storage address, the M register will not be used. The instruction code has been transferred to the I register.

(8) Set the MODE switch to DISTRIBUTOR.

(9) Press the START switch to observe what happens at DP_0, and continue to press the switch to observe what happens at each distributor pulse through DP_{17}. The following will be observed during each distributor count.

DP_0—The C register was set.

DP_1—No subcommands were generated.

DP_2—The shift count (05) was transferred from the X register to the C register.

DP_3 through **DP_{11}**—No subcommands were generated.

DP_{12}—The C register was checked to see if it equals zero. Since it was not equal to zero, the normal sequence of distributor pulses is continued.

DP_{13}—The C register count was decreased by 1.

DP_{14}—No subcommands were generated.

DP_{15}—The contents of the AQM register were shifted right one place.

DP_{16}—The computer looped back to DP_7 because the C register was not equal to zero. Note that the D register now reads DP_7. As the START switch is continually pressed, the computer will repeat DP_7 through DP_{16} four more times, for a total of five shifts. At the end of the fifth shift, Q_4 will be set. Also, at the end of the fifth shift, the C register will read zero; therefore, the computer will advance from DP_{16} to

DP_{17}—The AE flip-flop is cleared to establish the acquisition phase for the next instruction.

4-8. BASIC CIRCUITS

This section describes the basic circuits used in the BI-TRAN SIX. Most of them are used many times in various sections of the computer. Throughout the circuit descriptions reference will be made to various reference points (RP). These are indicated on the schematic diagrams by circled numbers. The waveshapes normally existing at the reference points are included at the end

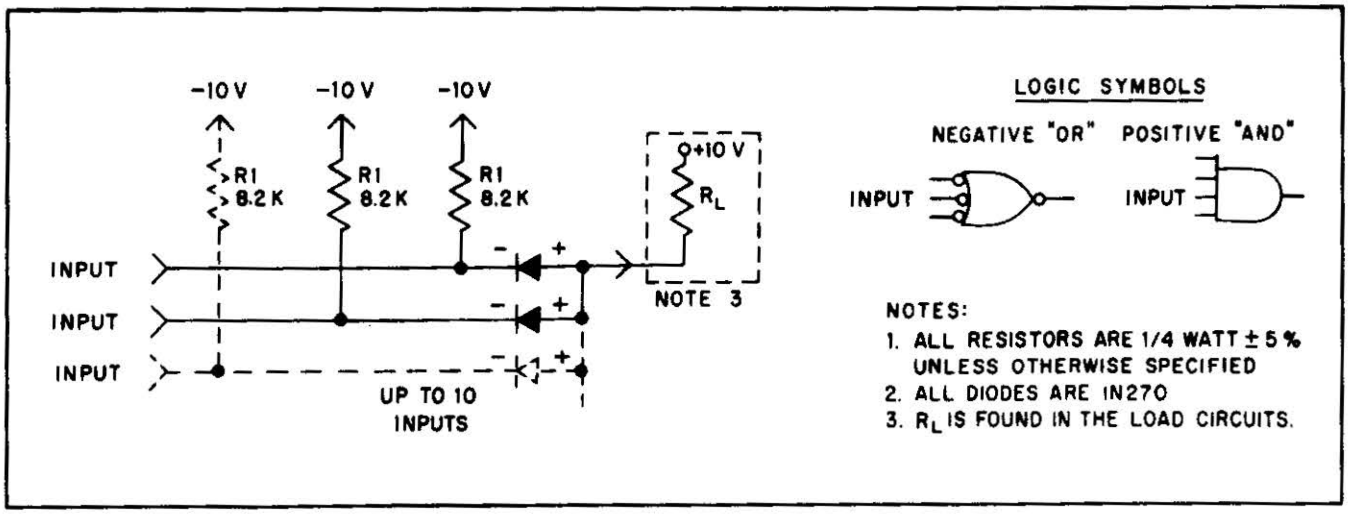


Figure 4-8. Positive AND & Negative OR Circuit

of this section (Figure 4-24). Nominal logic levels are zero volts and —10 volts. Because of circuit loading, however, the negative level may be either —10 volts or approximately —4.5 volts. Therefore, where —10 volts is indicated as the input voltage to cause a certain action, —4.5 volts is also acceptable.

4-8.1. "POSITIVE AND" AND "NEGATIVE OR" CIRCUIT

A positive AND circuit and a negative OR circuit (Figure 4-8) are identical electrically but differ logically. The circuit requires all ground inputs to deliver a ground output. If any one input is a —10 volts, the output of the circuit is —10 volts. This is because a —10-volt input to any of the diodes will cause that diode to conduct. Since all of the diode anodes are connected, all of the diode anodes will be at —10 volts. Only by cutting off all the diodes with ground inputs will the common anode output be at ground. As shown in the truth table (Table 4-2), the circuit acts as an AND gate for positive (0-volt) levels, and as an OR gate for negative (—10-volt) levels.

4-8.2. "NEGATIVE AND" AND "POSITIVE OR" CIRCUIT

A negative AND circuit and a positive OR circuit (Figure 4-9) are identical electrically but differ logically, and operate essentially the opposite of the positive AND and negative OR circuit. If a ground input is applied to the anode of any one of the diodes, that diode will conduct, placing its cathode at ground potential. Since the diode cathodes are connected, they will all be at ground potential. The output of the circuit will therefore be ground. The ouput of the circuit will be —10 volts only when all of the diodes are cut off by having —10 volts applied to their anodes, or by having their inputs open. Table 4-2 summarizes the input-output relationships of the negative AND and positive OR circuits. As shown in the truth table, this circuit acts as an AND gate for negative levels and as an OR gate for positive levels.

4-8.3. INVERTER

The standard inverter used throughout the computer is shown in Figure 4-10. It is a conventional common emit-

Table 4-2. AND and OR Input-Output Relationships

Pos. AND, Neg. OR Gate for Two Inputs			Neg. AND, Pos. OR Gate for Two Inputs		
A	B	Output	A	B	Output
—10V	—10V	—10V	—10V	—10V	—10V
0V	—10V	—10V	0V	—10V	0V
—10V	0V	—10V	—10V	0V	0V
0V	0V	0V	0V	0V	0V

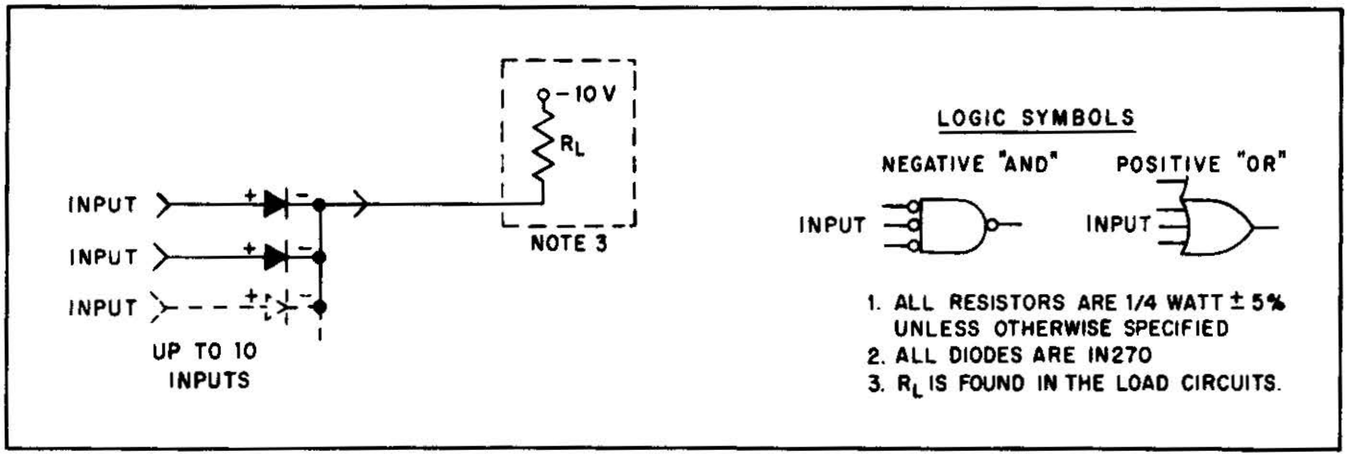


Figure 4-9. Negative AND & Positive OR Circuit

ter saturated amplifier in which signal inversion occurs. The emitter-base bias is set so that the transistor cuts off when the input is grounded and conducts when —4.5 to —10 volts is applied to the input. Therefore, when a "1" signal (ground) is applied to the input, the stage will cut off, and a "0" signal (approximately —10 volts) will be present at the output. Conversely, when a "0" signal (—4.5 to —10 volts) is applied to the input, the stage conducts at almost saturation and a "1" signal (essentially 0 volts, or ground) is present at the output. Thus, in both cases, the input signal is inverted. If the input circuit is open, the inverter will also conduct. Isolation diodes in the input circuit will effectively open the input when back-biased by a negative level. Resistor R4 is not used when a positive AND or a negative OR is connected to the input of the inverter. Resistor R1 of the positive AND and negative OR circuit (Figure 4-8) serves the same function as inverter resistor R4.

Capacitor C1 is used only on the memory section inverters. This capacitor speeds up the response of the inverter stage, since rapid response is very important in memory timing.

4-8.4. CLOCK MULTIVIBRATOR

A conventional collector-coupled, free-running, asymmetrical multivibrator (Figure 4-11) is used to generate the basic timing signals for the BI-TRAN SIX. The state of conduction of each transistor alternately changes from cutoff to saturation. Conduction of one transistor causes cutoff of the other, so that when one is on, the other is off.

When Q1 goes into conduction, the positive-going

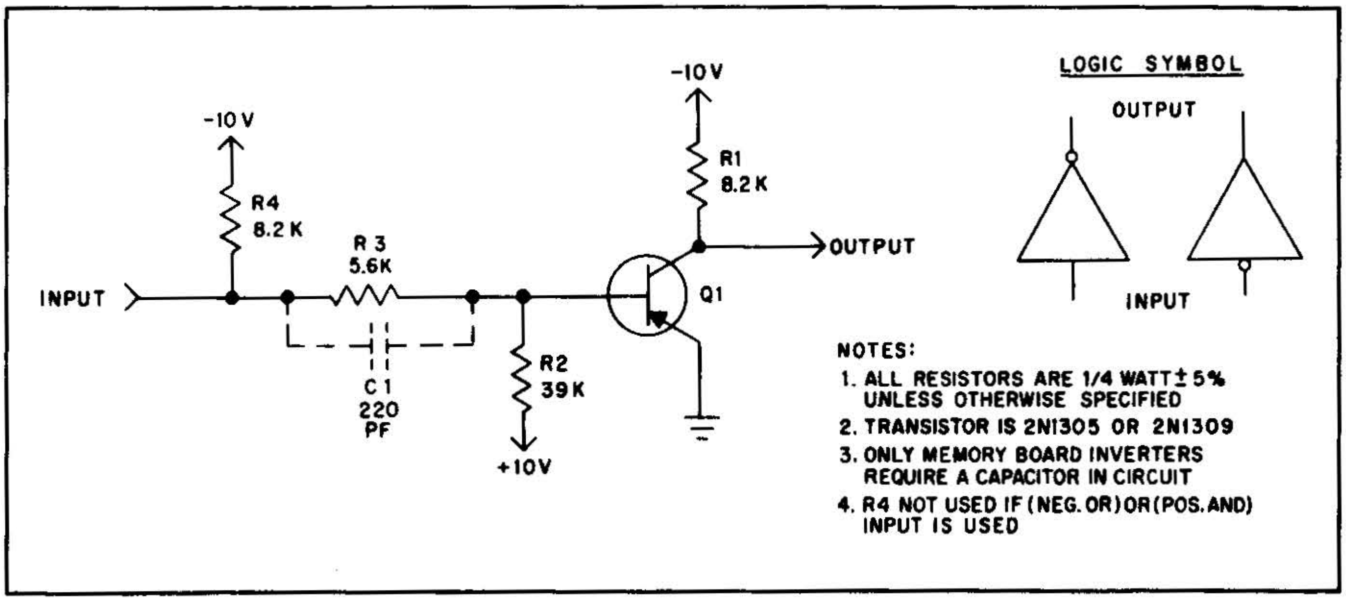
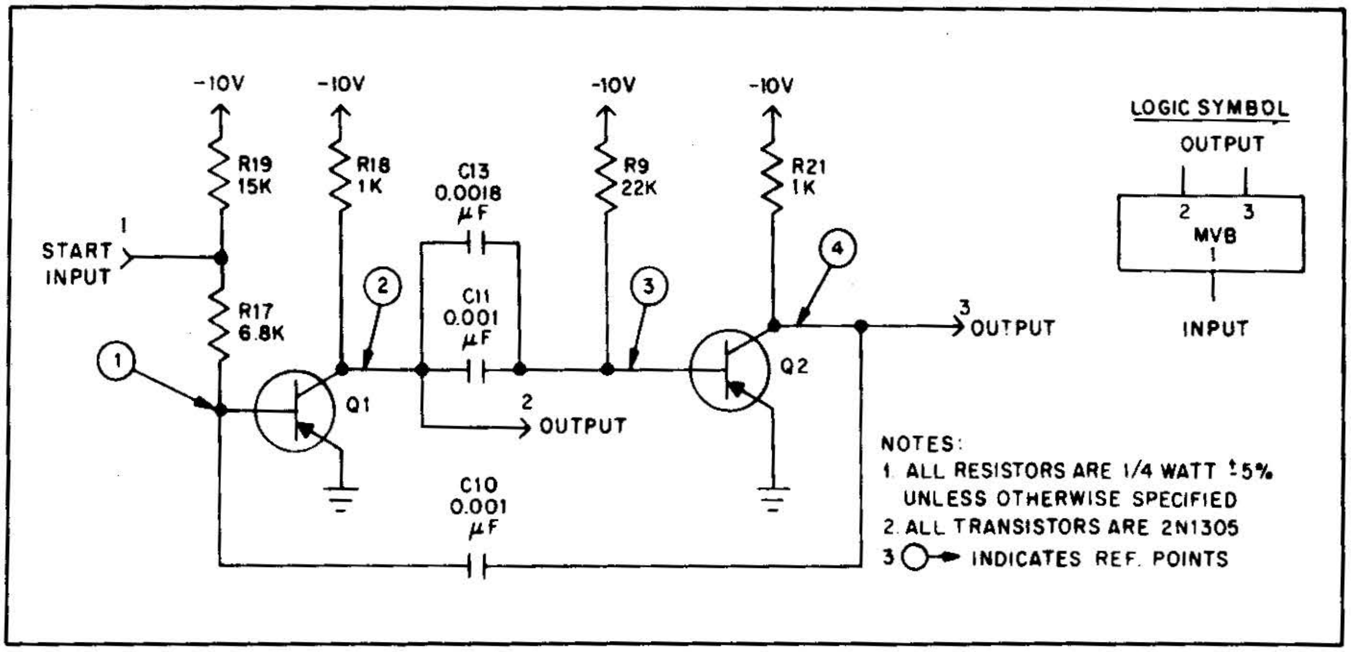

-10V
R4
8.2K
R 3
5.6K
INPUT
C 1
220
PF
R2
39K
+10V
-10V
R1
8.2K
OUTPUT
Q1
LOGIC SYMBOL
OUTPUT
INPUT
NOTES:
1. ALL RESISTORS ARE 1/4 WATT ± 5% UNLESS OTHERWISE SPECIFIED
2. TRANSISTOR IS 2N1305 OR 2N1309
3. ONLY MEMORY BOARD INVERTERS REQUIRE A CAPACITOR IN CIRCUIT
4. R4 NOT USED IF (NEG. OR) OR (POS. AND) INPUT IS USED

Figure 4-10. Inverter Circuit

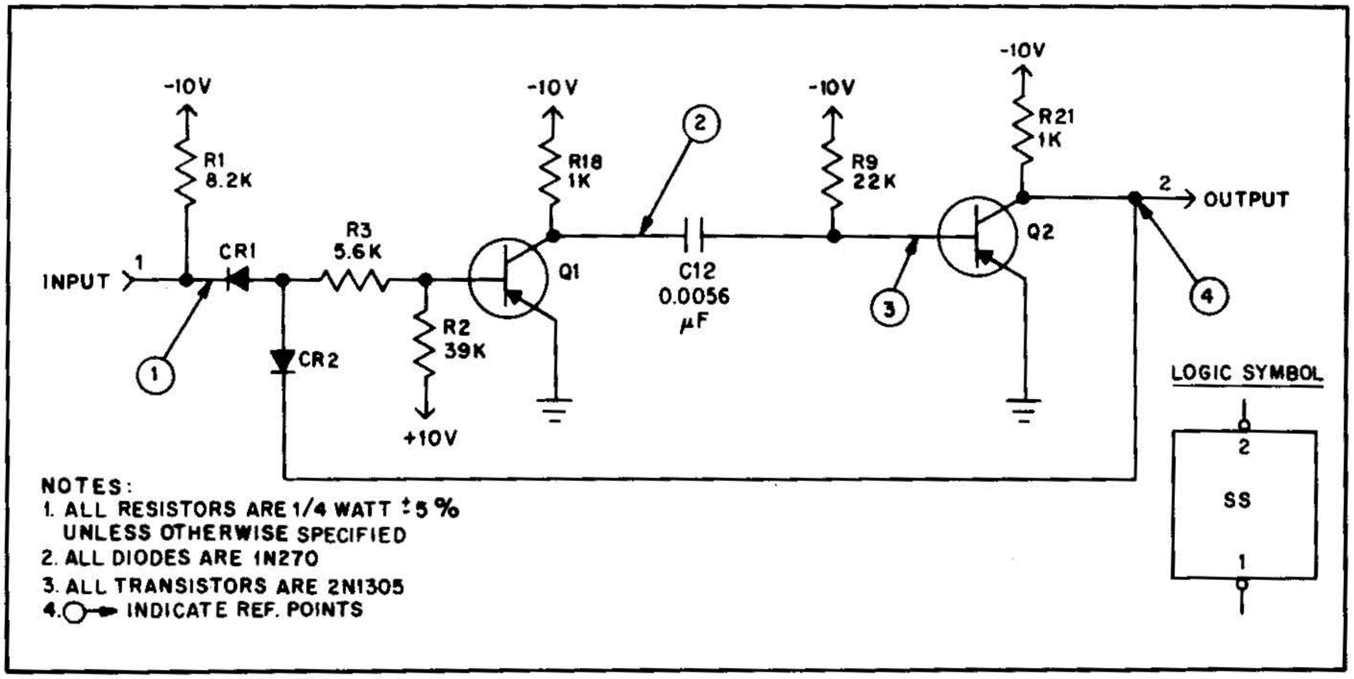


NOTES:
1. ALL RESISTORS ARE 1/4 WATT ±5% UNLESS OTHERWISE SPECIFIED
2. ALL TRANSISTORS ARE 2N1305
3. ○➔ INDICATES REF. POINTS

Figure 4-11. Multivibrator Circuit

voltage (coupled to the Q2 base through C13 and C11) turns off Q2. The positive voltage now developed by the discharge of C13 and C11 through R9 holds Q2 off. This condition will last until C13 and C11 discharge to about 0 volts (at the base of Q2). At this time, Q2 will conduct, and will turn off Q1 with a positive voltage coupled through C10. Now the discharge of C10 through R17 and R19 holds Q1 cut off.

When C10 discharges to zero volts, Q1 will once again conduct, starting a new cycle. The time of a complete cycle is determined primarily by the time constants of C13, C11, and R9 holding Q2 off, and C10, R17, and R19 holding Q1 off. These times are asymmetrical: Q1 is cut off for 15 usec, then conducts for 35 usec; Q2 is cut off for 35 usec, and then conducts for 15 usec. This gives a total cycle of 50 usec.



NOTES:
1. ALL RESISTORS ARE 1/4 WATT ±5% UNLESS OTHERWISE SPECIFIED
2. ALL DIODES ARE 1N270
3. ALL TRANSISTORS ARE 2N1305
4. ○➔ INDICATE REF. POINTS

Figure 4-12. Single-Shot Multivibrator Circuit

4-8.5. SINGLE-SHOT MULTIVIBRATOR

The single-shot multivibrator used in the BI-TRAN SIX is shown in Figure 4-12. It has one stable state (Q2 conducting and Q1 cut off) and one quasi-stable state (Q1 conducting and Q2 cut off). A negative input signal applied to the base of Q1 forward-biases the emitter-base junction and Q1 starts to conduct. This decreases the forward bias on the emitter-base junction of Q2, and turns it off. The negative output of Q2 (through CR2) now holds Q1 in conduction. This quasi-stable state will last until C12 is discharged, because the discharge of C12 through R9 is holding the base of Q2 positive. When C12 is discharged, Q2 turns on. The positive output of Q2 turns Q1 off.

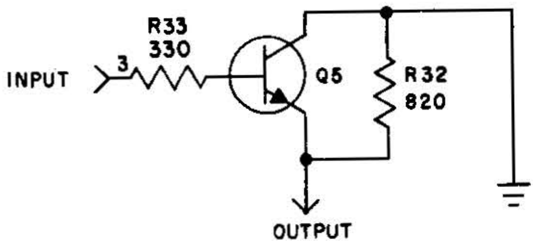
Note that diodes CR1 and CR2 form a negative OR gate. Thus, while the single-shot is in the quasi-stable state, changes in the CR1 input have no effect.

4-8.6 LAMP DRIVER

The lamp driver circuit (Figure 4-13) is a conventional emitter follower. Each lamp driver has a switch-light lamp in series with its emitter. Minus 10 volts is applied to emitter through the lamp. Therefore, when a "1" (ground) is present at the input, the circuit conducts and the lamp in the output circuit lights. With a "0" (—10 volts) present at the input, the circuit cuts off and the lamp goes out.

4-8.7 FLIP-FLOP

Figure 4-14 illustrates the typical flip-flop used throughout the computer. It is a conventional flip-flop consisting of two common-emitter saturated amplifiers coupled to each other through resistor-diode AND/OR gates and toggling circuits. All of the inputs shown in Figure 4-14 are not used in any one flip-flop. Variations are described in the notes on Figure 4-14.



NOTES:
1. RESISTORS ARE 1/4 WATT ±5% UNLESS OTHERWISE SPECIFIED
2. TRANSISTOR IS 2N1304
3. R33 USED ON SN66464 AND ABOVE

Figure 4-13. Lamp Driver Circuit

The flip-flop has two outputs, labeled "1" and "Ø". When in a quiescent state, these outputs are complementary; one of them will be —10 volts and the other zero volts. When the "1" output is zero volts, i.e., logic "1"; the flip-flop is defined as storing a "1" or as being in the "1" state. An input that serves only to put the flip-flop into the "1" state is labeled a SET input. When the "Ø" output is zero volts, the flip-flop is defined as storing a "Ø" or as being in the "Ø" state. An input that can only put the flip-flop into the "Ø" state is a CLEAR input.

One State: When in the one state, Q1 is saturated, developing a "1"-output level of zero volts and forward biasing CR4. Because the cathode of CR2 is now at approximately zero volts, the junction of R7 and R6 is approximately +1 volt and Q2 is cut off. With Q2 cut off, the "O" output level is —10 volts and CR3 is reverse biased. This allows the R4-CR1-R3-Q1 base voltage divider to drive the Q1 base negative in respect to the emitter, holding Q1 in saturation.

Zero State: When in the zero state, Q2 is saturated and develops a "Ø" output level of zero volts which forward biases CR3 and develops approximately +1 volt at the Q1 base — cutting off Q1. With Q1 cut off, the "1" output level is —10 volts and CR4 is reverse biased. This allows the R8-CR2-R7-Q2 base voltage divider to drive the Q2 base negative in respect to the emitter, holding Q2 in saturation.

NEGATIVE OR SET: When this input is used, CR1 electrically becomes a part of the negative OR gate that drives the NEGATIVE OR SET input. Therefore, the base-emitter junction of Q1 can be forward biased (setting the flip-flop) by a negative level applied either via CR1 or via similar diodes in the driving gate.

POSITIVE OR SET: When this input is used, CR4 electrically becomes a part of the positive OR gate that drives the POSITIVE OR SET input. Therefore, the base-emitter junction of Q2 can be reverse biased (clearing the flip-flop) by a positive level applied either via CR4 or via similar diodes in the driving gate.

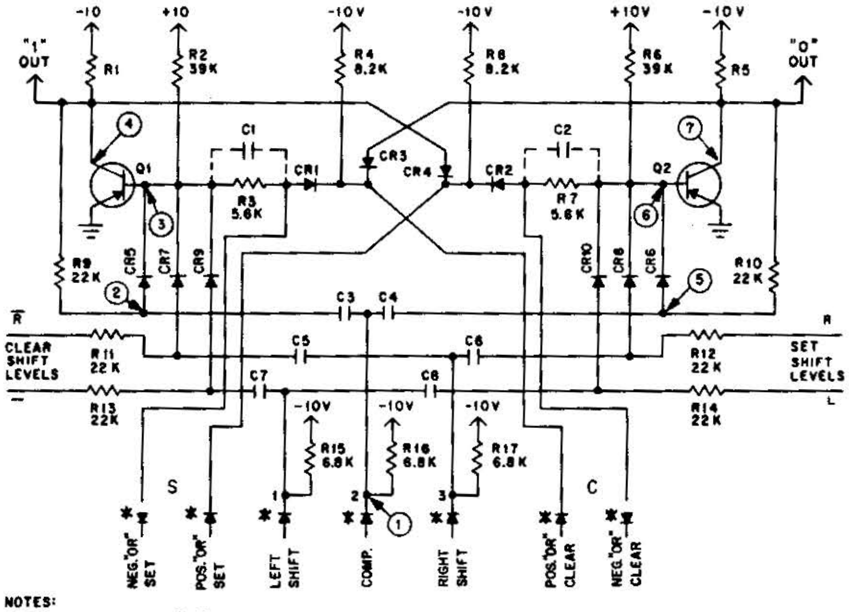
CLEAR Inputs: The CLEAR inputs operate in the same manner as the SET inputs, using CR2 (NEGATIVE OR CLEAR) and CR3 (POSITIVE OR CLEAR).

Complement: Assume the flip-flop is in the "Ø" state. Q1 is cut off, therefore the anode of CR5 has —10 volts applied via R9. Q2 is saturated, therefore the anode of CR6 has zero volts applied via R10. The cathodes of CR5 and CR6 cannot become more negative than approximately −0.3 volts because of the base limiting effects of Q1 and Q2. When a complementing pulse, rising from −10 volts to zero volts, is applied to the COMP. input, a 10 volt rise is coupled through C3 and C4. The anode of CR5 rises from −10 volts toward zero volts – therefore, CR5 stays reverse biased. The anode of CR6 rises from zero volts toward +10 volts – forward biasing CR6. Current flow through R7 drives the Q2 base positive and Q2 into cutoff. The "Ø"-output level has become −10 volts and CR3 is reverse biased – causing Q1 to saturate and the "1" output to become zero volts. The flip-flop has been complemented from the zero to the one state. Complementing from the one state is analogous to the preceding operations; in either case the COMP. trigger rise is steered to the base of the saturated transistor and drives it into cutoff.

Shift: The flip-flops in several registers are directly interconnected via their shift inputs to allow serial data transfer between logically adjacent flip-flops. A right shift transfers data toward the least significant digit flip-flop; a left shift toward the most significant digit flip-flop. Therefore, a flip-flop's R and $\overline{R}$ (right) inputs are driven by the flip-flop to its left and are used to shift data to the right. The L and $\overline{L}$ inputs are used to transfer data to the left. The operation of the shift circuitry is analogous to the complement operation. When a LEFT SHIFT trigger occurs CR9 is forward biased and Q1 cutoff if the $\overline{L}$ input is zero volts; CR10 is forward biased and Q2 cutoff if the L input is zero volts. Since the $\overline{L}$ and L inputs are driven by the "Ø" and "1" outputs, respectively, of the flip-flop to the right, data has been shifted left.

The remaining shift inputs are similar; when a shift trigger occurs, the diode (CR7, 8, 9, and 10) with a zero volt level input ($\overline{R}$, R, L, $\overline{L}$) becomes forward biased and its associated transistor cutoff – setting or clearing the flip-flop.

Switch-Light: Not shown on the diagram is the switch-light input to the flip-flop. The switch's normally open contacts are connected directly to the collector of Q1. Closing the contacts applies ground to Q1, setting the flip-flop. In the C register flip-flops, the switch is connected to the collector of Q2, clearing these flip-flops when the lens (switch) is depressed.



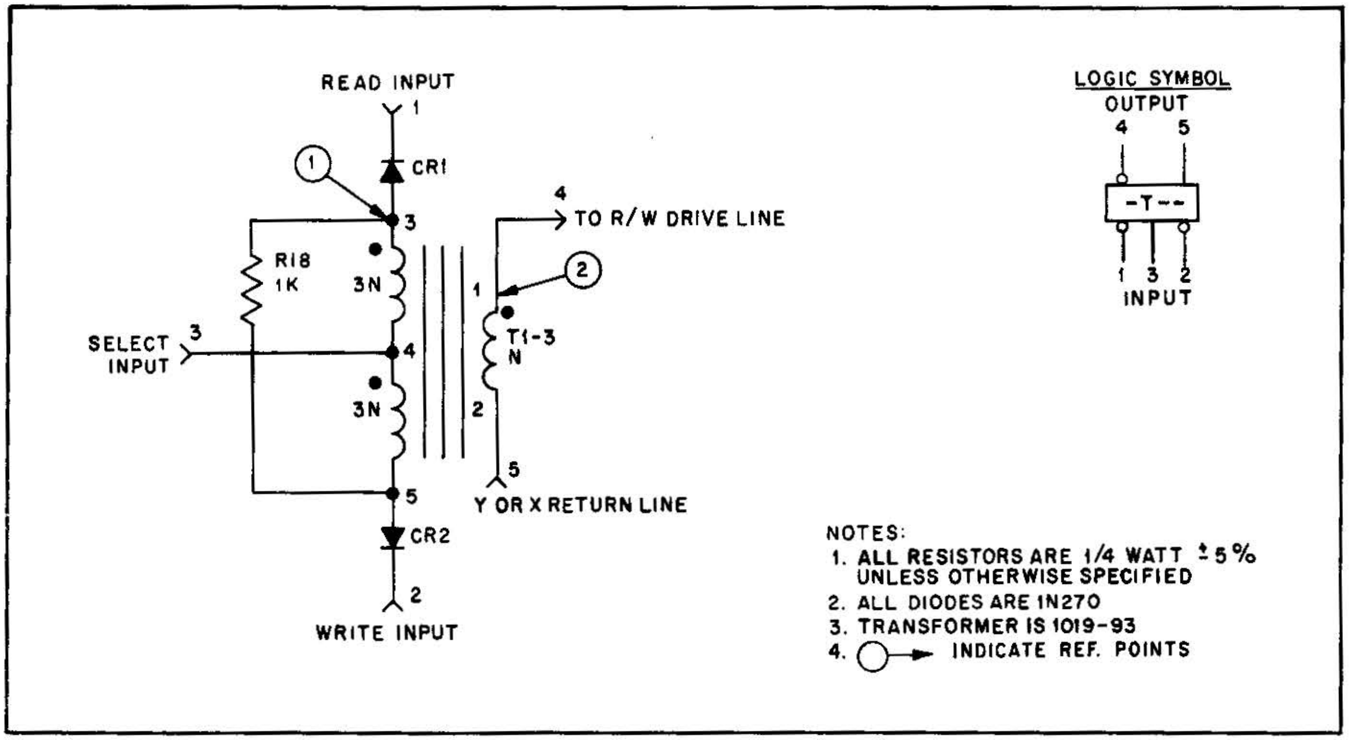
NOTES:

1. a. CR1 USED IF NEG. "OR" SET IS USED
   b. CR2 USED IF NEG. "OR" CLEAR IS USED
2. C1 AND C2 ARE USED IN MEMORY FF'S ONLY
3. CR5, CR6, R9, R10, R16, C3 AND C4 ARE USED ONLY IF TOGGLE 2 IS SHOWN ON LOGIC DIAGRAM
4. CR7, CR8, R11, R12, R17, C5 AND C6 ARE USED ONLY IF TOGGLE 3 IS SHOWN ON LOGIC DIAGRAM
5. CR9, CR10, R13, R14, R15, C7 AND C8 ARE USED ONLY IF TOGGLE 1 IS SHOWN ON LOGIC DIAGRAM
6. R1 AND R5 ARE SET FOR DIFFERENT NUMBERS OF TOGGLE CIRCUITS USED

R1 & R5	NO	TOGGLE CIRCUITS	8.2K
R1-1 & R5-1	1	TOGGLE CIRCUIT	4.7K
R1-2 & R5-2	2	TOGGLE CIRCUITS	3.3K
R1-3 & R5-3	3	TOGGLE CIRCUITS	2.7K

7. TRANSISTORS: 2N1305 OR 2N1309 AS CALLED OUT BY PARTS LIST
8. ALL DIODES ARE 1N270
9. ALL RESISTORS 1/4 WATT ± 5% UNLESS OTHERWISE SPECIFIED
10. CAPACITOR: C1 AND C2 ARE 220 PF 500V ±10%
    C3 THRU C8 ARE 470 PF 500V ±10%
11. ○→ INDICATE REF. POINTS
12. * THESE DIODES ALWAYS LOCATED IN DRIVING LOGIC

Figure 4-14. Flip-Flop Circuit and Variations



NOTES:
1. ALL RESISTORS ARE 1/4 WATT ±5% UNLESS OTHERWISE SPECIFIED
2. ALL DIODES ARE 1N270
3. TRANSFORMER IS 1019-93
4. ○→ INDICATE REF. POINTS

Figure 4-15 Selection Matrix Transformer

4-8.8. SELECTION MATRIX TRANSFORMER

The Selection Matrix Transformer (Figure 4-15) provides current for its corresponding read-write drive line when it is energized by a line selector circuit and a read-write driver. Essentially, the Selection Matrix Transformer consists of a step-down, iron-core transformer having two separate primary windings. During the read operation, ground is applied to terminal 3 of the selected drive line selection circuit by the line selector. Read current (produced by the read/write driver) then flows through the upper primary winding, inducing a voltage pulse in the secondary winding. As a result, a pulse of current flows in the read-write drive line, which is connected directly across the secondary.

The circuit operates the same during the write operation, except that write current (produced by the read/write driver) is applied to terminal 2. Consequently, current flow is through the lower primary winding. Due to the arrangement of the primary windings, this induces a voltage in the secondary that has a polarity opposite to that induced during the read operation, and current flow in the read-write drive line is in the opposite direction.

Note that current flow in the primary is toward the select input (line selector) for both the read and the write inputs. If the select input is at —10 volts, CR1 and CR2 are reverse biased and the read or write current cannot flow.

4-8.9. LINE SELECTOR CIRCUIT

The line selector circuit (Figure 4-16) controls the input to terminal 3 of each of four Selection Matrix Transformers by acting as a switch. It consists of an emitter follower (Q1) and a grounded emitter output stage (Q2). This line selector circuit delivers a ground output when its input is a logic "0" (—10 volts). Conversely, it delivers a —10-volt output when its input is a "1". When it delivers —10 volts, it is acting like an open switch to the Selection Matrix Transformer and read/write drivers. With a logic "0" at the input (terminal 1), the base of Q1 is placed at —10 volts (RP1), causing the transistor to conduct heavily. This drives the emitter of Q1 as well as the junction of resistors R3 and R22 in the emitter circuit of Q1 negative (RP2 and RP3). As a result, the base-emitter circuit of Q2 is forward biased, and Q2 can conduct heavily. It will now act as a closed switch to allow current developed by a read/write driver to pass through a Selection Matric Transformer to ground.

When ground (logic "1") is applied to terminal 1 of the line selector circuit, it decreases the forward bias on the base-emitter of Q1. This causes a decrease in Q1 collector current, with the result that the junction of resistors R3 and R22 becomes positive. Q2 therefore cuts off, and its collector voltage (terminal 2) becomes essentially —10 volts.

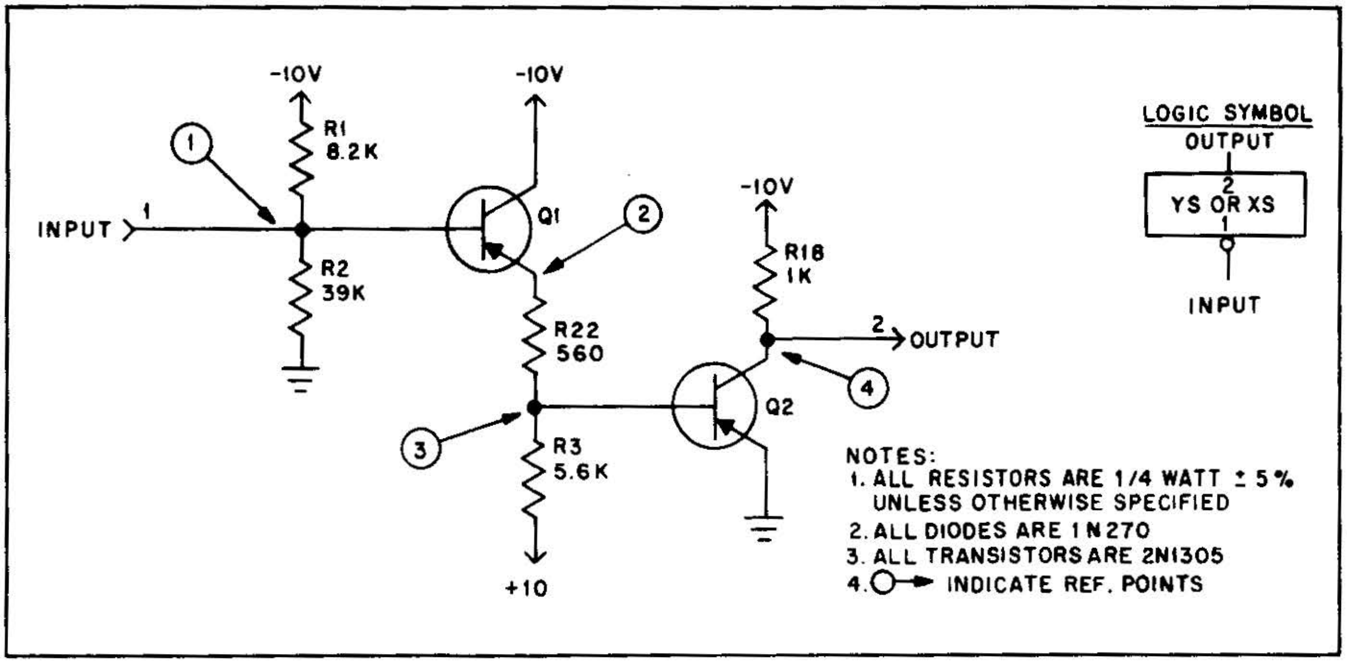


Figure 4-16. Line Selector Circuit

4-8.10. READ/WRITE DRIVER CIRCUIT AND X AND Y CURRENT LIMITER

The read/write driver (Figure 4-17) controls the logic input to terminals 1 (READ) and 2 (WRITE) of each of four Selection Matrix Transformers. Every read-write driver consists of two identical sections: one for the read operation, and one for the write operation. The READ section (transistors Q1 and Q2) will be described here. The WRITE section (transistors Q3 and Q4) operates in exactly the same manner, but not at the same time.

The input to the READ section is applied to terminal 1. The logic state ("1" or "0") of the input is determined by two bits of the memory address and the state of the READ flip-flop. When the input is "0", the READ section of the read-write drivers delivers a direct current output; and when the input is "1", the driver delivers no output whatsoever.

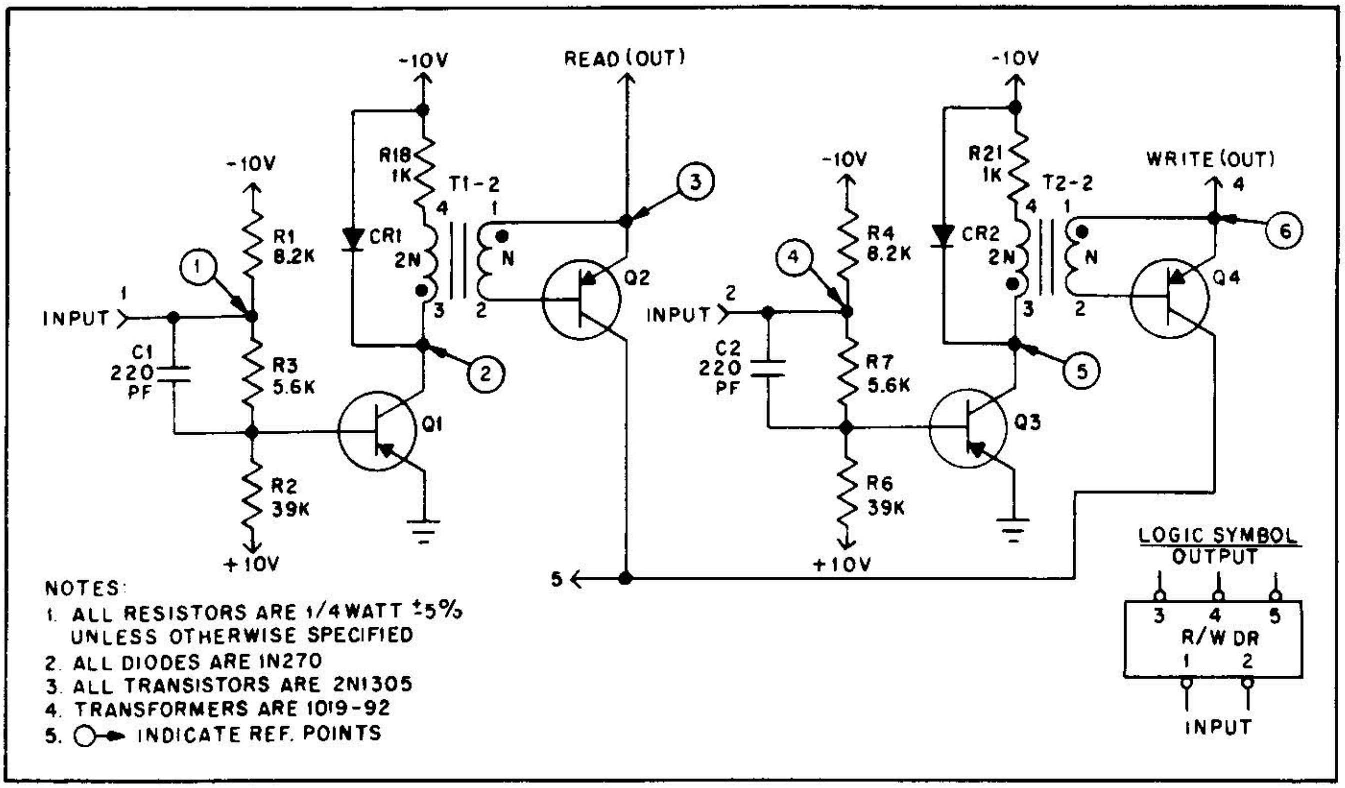
With a "1" (ground) applied to terminal 1, transistor Q1 is reverse biased, and, therefore, is cut off. No collector current flows in Q1, and as a result, no voltage is coupled to transistor Q2 by transformer T1-2. Transistor Q2, therefore, has zero base-emitter bias, and is cut off. Transistor Q2 functions as a switch, connecting the output of the driver (terminal 3) to a current limiter circuit connected to terminal 5. When Q2 is cut off, the switch is effectively open, and therefore, no output appears at terminal 3.

When a logic "0" (—10 volts) is applied to terminal 1 (RP1), it forward biases Q1, causing it to conduct. The resulting rise in collector current through the primary of transformer T1-2 causes the collector voltage of Q1 to approach ground (RP2), and induces a voltage in the secondary winding of T1-2. The polarity of this voltage is such that it forward biases Q2, causing Q2 to conduct. As a result, the output of the driver (terminal 3) is effectively connected to terminal 5, which in turn is connected to —10 volts through a current limiter circuit. Thus, with a logic "0" input to the driver, transistor Q2 conducts, effectively driving its emitter negative (RP3). Current then flows into terminal 3 of the driver, through Q2, and out terminal 5.

The amplitude of the current flow through Q2 is controlled by a current limiter circuit (Figure 4-18) connected to terminal 5 of the driver. Resistors R34 and R33 of the limiter are in series with the collector of Q2, and act as a resistance in the current path. Capacitor C14 of the limiter circuit stores energy for the high read-write current pulse. Because each core memory has its own characteristics as to the proper read-write current, the value of R33 is determined for each BI-TRAN SIX separately.

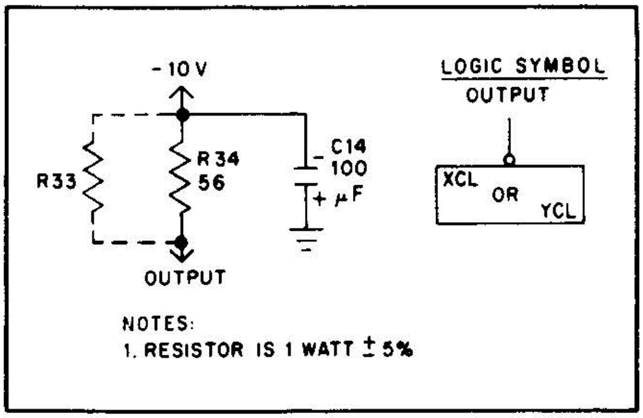
4-8.11. INHIBIT DRIVER CIRCUIT

The inhibit driver circuit (Figure 4-19) provides inhibit current to a plane to prevent writing a "1" into an addressed memory core. It provides this current during the read-write operation if its associated flip-flop in the X register is in the "0" state. With a logic "1"



NOTES:
1. ALL RESISTORS ARE 1/4 WATT ±5% UNLESS OTHERWISE SPECIFIED
2. ALL DIODES ARE IN270
3. ALL TRANSISTORS ARE 2N1305
4. TRANSFORMERS ARE 1019-92
5. ○—● INDICATE REF. POINTS

Figure 4-17. Read-Write Driver Circuit



NOTES:
1. RESISTOR IS 1 WATT ± 5%

Figure 4-18. X and Y Current Limiter Circuit

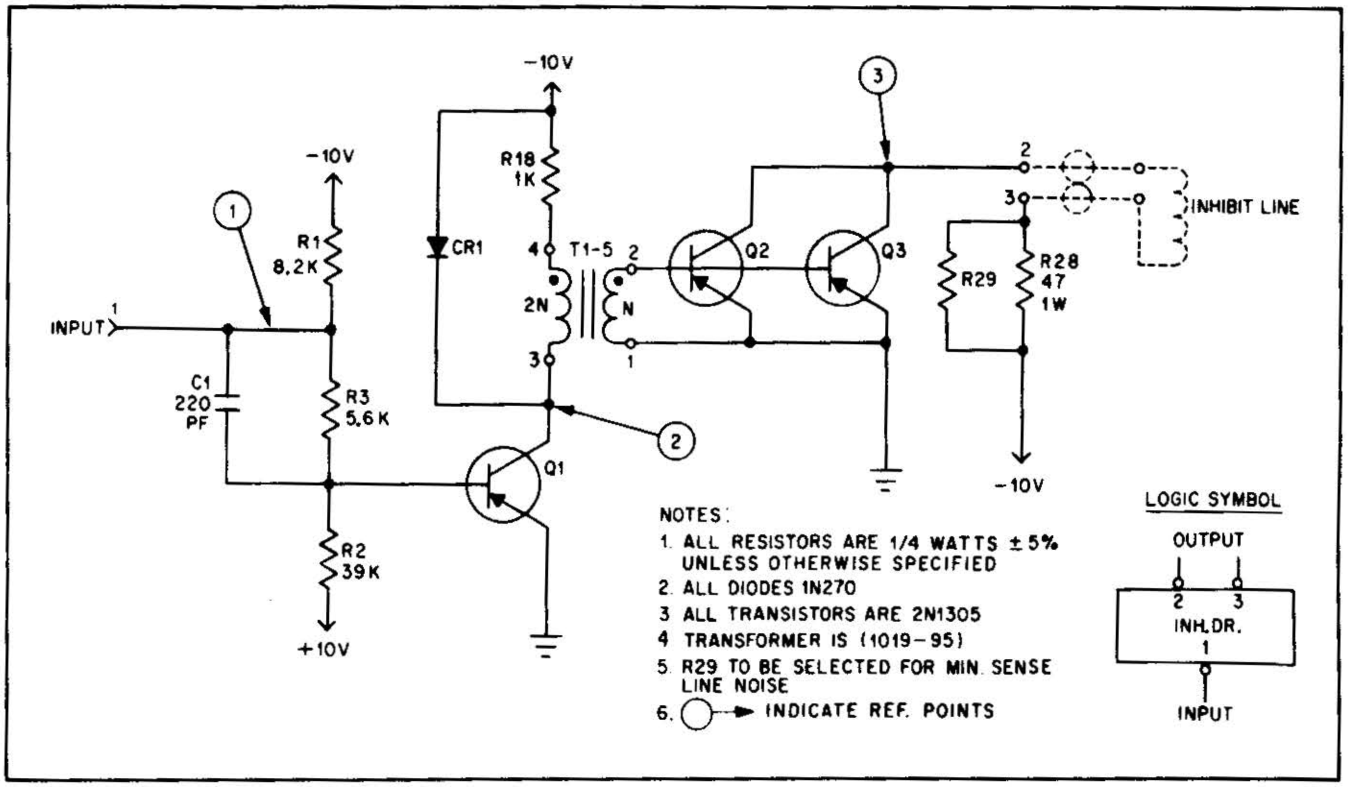
(ground) applied to terminal 1 of the inhibit driver, transistor Q1 is cut off. Therefore, no voltage is coupled to the base-emitter circuits of transistors Q2 and Q3 by transformer TI-5. Consequently, both Q2 and 3 are also cut off. Therefore, no current flows in the inhibit line, which is in series with the collector circuits of Q2 and Q3.

When a logic "0" (—10 volts) is applied to terminal 1 (RP1), it biases Q1 in the forward direction. The transistor therefore conducts, and its collector voltage approaches ground (RP2). The rise in collector current causes a voltage to be induced in the secondary of transformer TI-5. This voltage forward biases the base-emitter circuits of Q2 and Q3, which are connected in parallel. Both Q2 and Q3 then conduct, and their collector voltages are driven towards ground (RP3). When Q2 and Q3 conduct, they provide current for the inhibit line. This current is in the .form of a pulse, since when the collector current of 1 stops increasing, forward biasing voltage is no longer coupled to Q2 and Q3 by transformer TI-5. Two parallel transistors, Q2 and Q3 are used because one transistor alone could not supply the required inhibit current.

4-8.12. DELAY CIRCUIT

The delay circuit (Figure 4-20) receives logic "0" (—10 volts) pulses of 15-usec pulse width at its input, and delivers the same pulses at its output after a 15 usec delay. When the input to the delay circuit is logic "1" (ground), the input side of capacitor C9 is clamped to ground by the isolation diode. The base-emitter circuit is forward-biased by the negative voltage applied through resistor R9. Q1, therefore, conducts heavily, and its collector voltage is essentially ground.

When a logic "0" (—10 volts) is applied to the input (RP1), it effectively biases the isolation diode in the reverse direction. This isolates the delay circuit from the input. As a result, capacitor C9 begins charging; with the charging current flowing from the —10 volt supply,

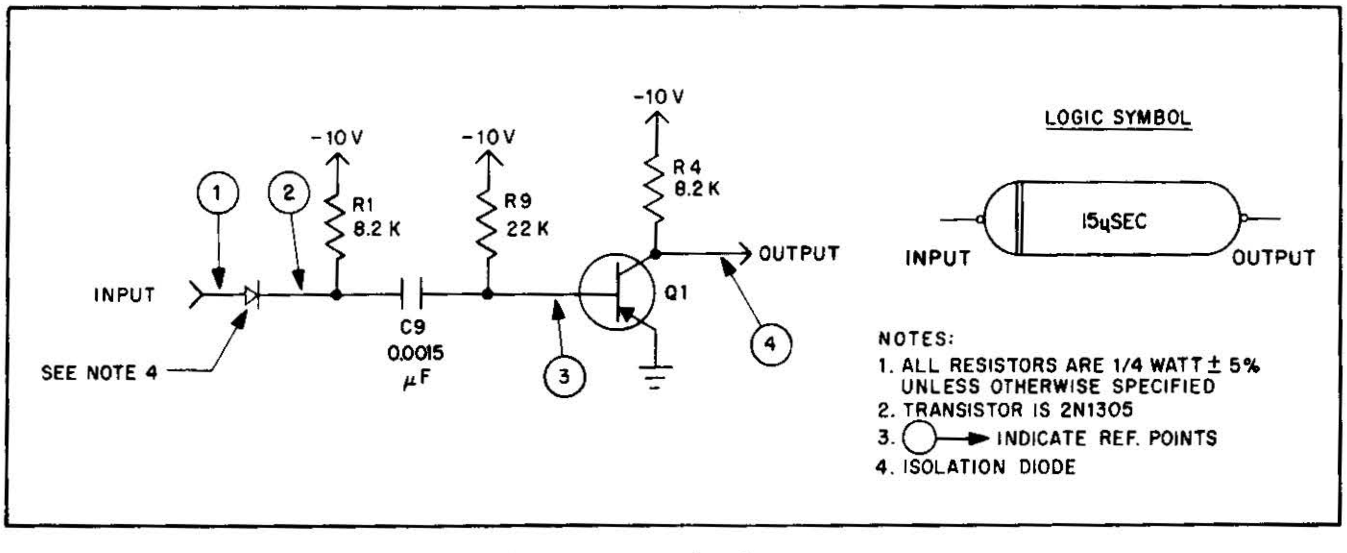


NOTES:
1. ALL RESISTORS ARE 1/4 WATTS ± 5% UNLESS OTHERWISE SPECIFIED
2. ALL DIODES 1N270
3. ALL TRANSISTORS ARE 2N1305
4. TRANSFORMER IS (1019-95)
5. R29 TO BE SELECTED FOR MIN. SENSE LINE NOISE
6. ○→ INDICATE REF. POINTS

Figure 4-19. Inhibit Driver Circuit

through resistor R1, through capacitor C9, from emitter to base through Q1, and back to the supply. The charge on C9 increases all during the time that the input is —10 volts (RP2). When the input switches again to logic "1" (ground), the input side of capacitor C9 instantly rises to ground, and C9 begins discharging through R9 and R1, driving the base positive (RP3). During the time that C9 discharges, transistor Q1 is cut off, since the discharge current removes the forward bias of Q1. Q1 does not begin conducting again until C9 is discharged. This takes approximately 15 usec.

The delay circut delivers a logic "0" (—10 volts) output while Q1 is cut off (RP4). This output is not produced until C9 begins discharging, which occurs at the termination of the logic "0" input pulse. The net effect is that the circuit delays 15-usec-width logic "0" pulses applied to its input for 15 usec.



NOTES:
1. ALL RESISTORS ARE 1/4 WATT ± 5% UNLESS OTHERWISE SPECIFIED
2. TRANSISTOR IS 2N1305
3. ○→ INDICATE REF. POINTS
4. ISOLATION DIODE

Figure 4-20. Delay Circuit

4-8.13. DELAY LINE CIRCUIT

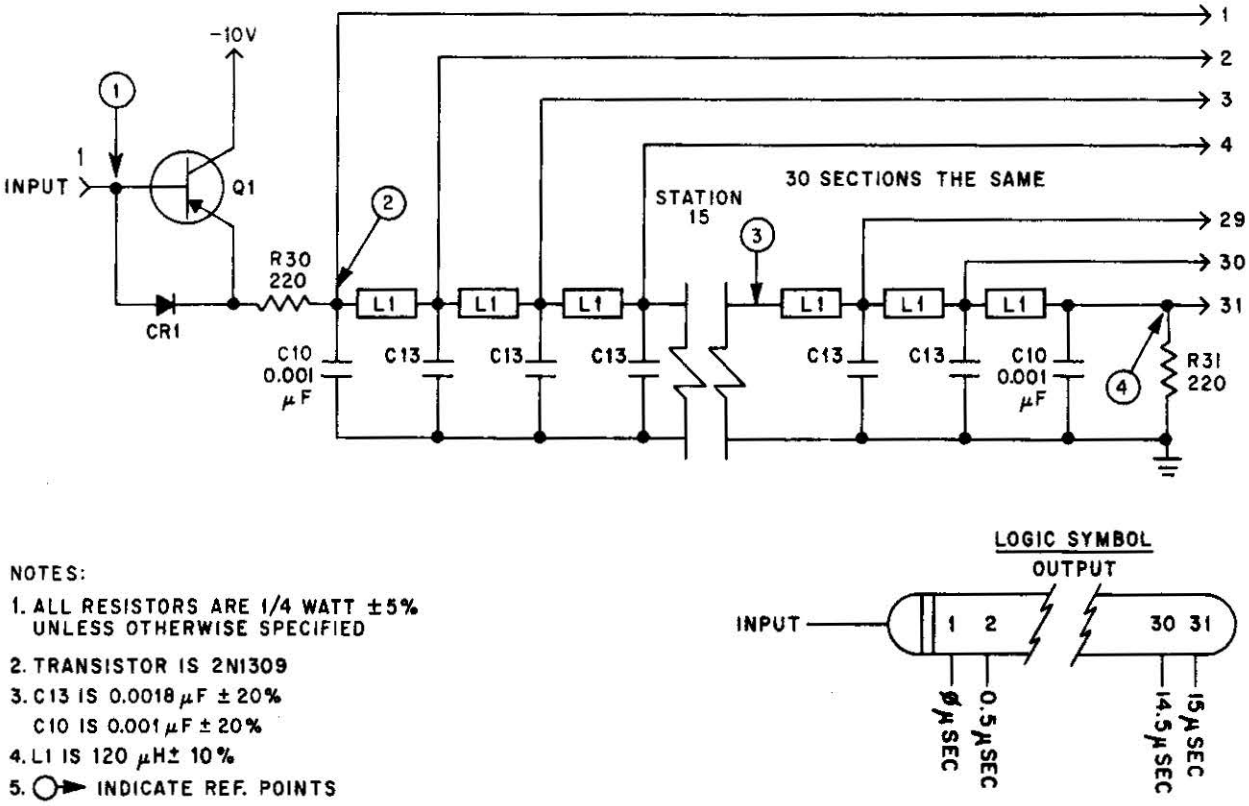
The delay line circuit (Figure 4-21) produces timing pulses for the memory. section during the read and write operations. The circuit consists of an emitter follower that has an artificial delay line as the emitter load. The delay line is made up of 30 Pi-type LC delay networks, each of which produces a delay of approximately one-half microsecond. Although 30 different delay-time pulses can be obtained from the delay circuit, only eight are used in the BI-TRAN SIX. There are 31 test points on the delay line circuit, TP1 being the input to the first delay.

Under quiescent conditions, the input applied to terminal 1 of the delay line circuit is a logic "0" (—10 volts). This forward-biases transistor Q1. When a read or write operation is initiated, a logic "1" (ground) is applied to terminal 1 (RP1). This cuts off transistor Q1, causing collector current to stop. As a result, the capacitor closest to the emitter discharges through resistor R30 and diode CR1 to ground. This causes the voltage at RP2 to rise towards ground. Consequently, the voltage on output line 1 from the delay line also rises towards ground.

As the voltage at RP2 becomes less negative, the adjacent capacitor also begins to discharge. Its discharge is delayed, however, by the inductance of its associated coil, L1. As a result, the rise in voltage on output line 2 lags behind that on line 1 by about one-half microsecond. This same action occurs for each succeeding LC delay network. Each capacitor discharges one-half microsecond after the preceding one. A total of 30 individual delays are produced for each logic "1" pulse applied to the input of the delay-line circuit. Resistors R30 and R31 are equal in value to the characteristic impedance of the delay line.

4.8.14. SENSE AMPLIFIER

The sense amplifier (Figure 4-22) produces a logic "1" (ground) output pulse during the read operation when its associated memory bit is a "1". In producing its output pulse, the sense amplifier receives three inputs. One of these comes via the sense line, from the memory core plane being read. (Because of the construction of the core plane, this sense line current can be in either direction for a "1" bit.) The other two inputs are the strobe



NOTES:

1. ALL RESISTORS ARE 1/4 WATT ±5% UNLESS OTHERWISE SPECIFIED
2. TRANSISTOR IS 2N1309
3. C13 IS 0.0018 μF ± 20%
   C10 IS 0.001 μF ± 20%
4. L1 IS 120 μH± 10%
5. ○► INDICATE REF. POINTS

Figure 4-21. Delay Line Circuit

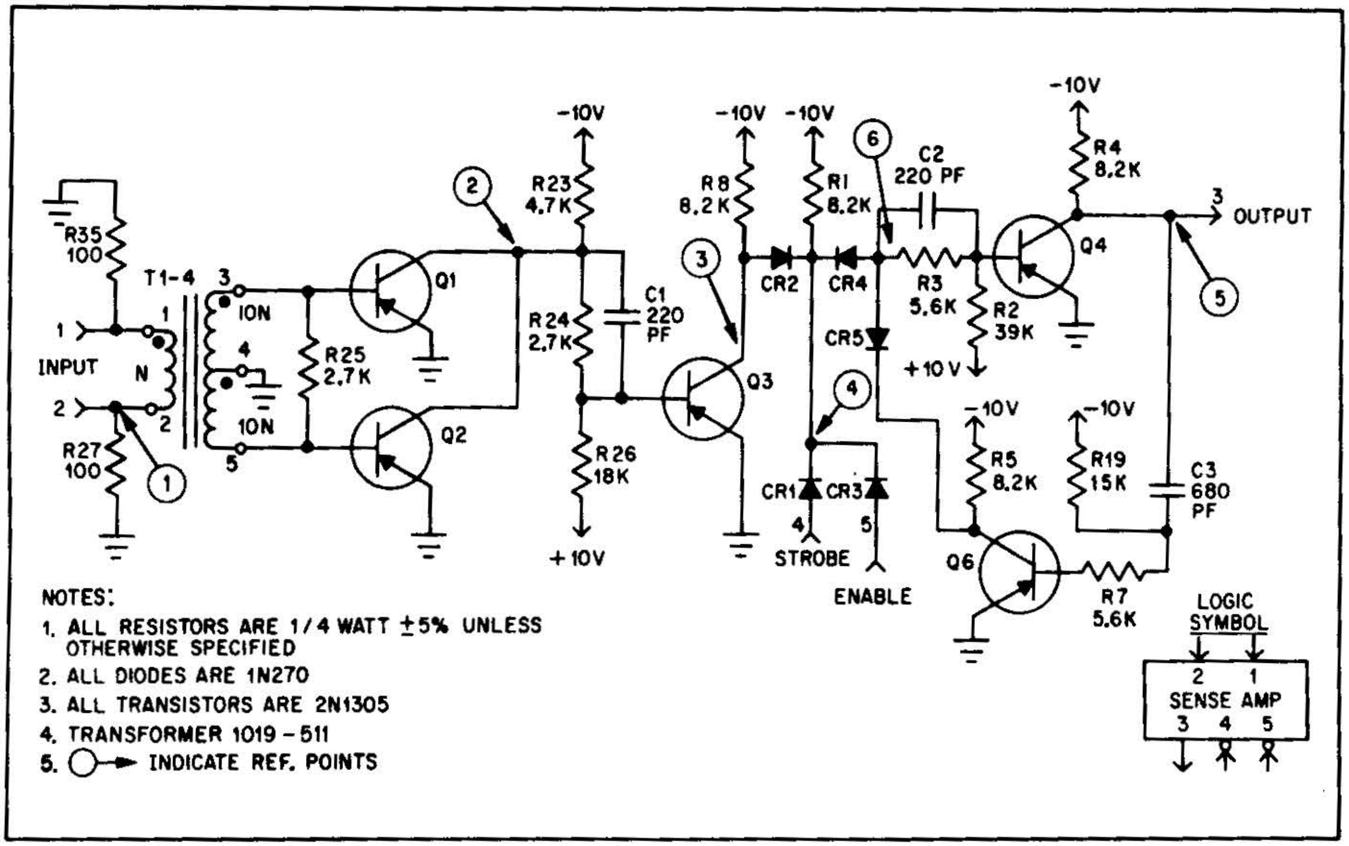


Figure 4-22. Sense Amplifier Circuit

pulse, which comes from the strobe inverter, and the enable signal, which comes from the clear side of the read/write flip-flop. Both the strobe and enable signals must be —10 volts for the sense amplifier to deliver its logic "1" output.

Input terminals 1 and 2 of the sense amplifier are connected to the primary winding of transformer TI-4. When a logic "1" is being read by the amplifier, the current through the primary is as shown on RP1. This sense line current is produced by two actions: First, the read/write driver reads the core (by clearing it), producing a current pulse on the sense line if the core was storing a "1". This "1"-bit current can be in either direction, depending upon which core in the plane is addressed. To reduce half-select noise in a plane, cores are wired so that half of them produce sense current in the opposite direction to the other half.

Next, the read/write driver writes the "1" back into the core. Although it is unwanted, the sense line develops an output at this time also. This current will be in the opposite direction to the read current.

This input current, which can be of either polarity, causes a voltage to be induced in the centertapped secondary, with one half of the voltage applied to the base-emitter circuit of transistor Q1, and the other half to the base-emitter circuit of Q2. Because of the centertap arrangement, the polarity of the induced voltage is such that Q1 and Q2 will conduct alternately. Since their collectors are connected, both Q1 and Q2 produce positive-going pulses at RP2 when they conduct. Thus, it does not matter which polarity the "1" bit produces. Each of the pulses at RP2 is applied to the base of transistor Q3. Q3 is normally conducting, but is cut off first by the positive-going "1" bit pulse, and then by the unwanted pulse. As a result, the collector of Q3 (RP3) becomes approximately —10 volts each time Q3 is cut off.

Diodes CR1, CR2, and CR3 form a negative AND gate. When Q3 is first cut off, assume both the strobe and enable signals are at —10 volts. Because all inputs to the gate are negative, the output going to CR4 is negative. Diodes CR4 and CR5 form a negative OR gate, which now outputs a negative pulse, forward-biasing Q4. Transistor Q4, which is normally cut off, therefore conducts, and its collector voltage rises to ground (RP5), for an output of a "1".

The rise in the Q4 collector voltage is coupled to the base of transistor Q6 by capacitor C3. This cuts Q6 off, applying a negative voltage to diode CR5. Diode CR5 therefore conducts, and in doing so, maintains the forward bias on the base-emitter of Q4 even though Q3 again begins to conduct. Thus, Q6 keeps Q4 conducting until capacitor C3 becomes fully charged. This is done to stretch the output pulse so it can set a flip-flop.

Note that in order for a sense amplifier to output a "1" bit, three signals must be coincident: sense line current, $\overline{\text{strobe}}$, and $\overline{\text{enable}}$. Sense line current produces a negative pulse that is ANDed with $\overline{\text{strobe}}$ and $\overline{\text{enable}}$ (CR1, CR2, CR3). The timing of the $\overline{\text{strobe}}$ pulse is such that when the unwanted pulse produced by writing occurs, the $\overline{\text{strobe}}$ input will be positive, inhibiting the gate (CR1, CR2, CR3). The $\overline{\text{enable}}$ level will be negative for the entire memory cycle, unless new data is to be put into memory from the X register. In this case, $\overline{\text{enable}}$ will be positive, inhibiting the gate, causing old data to be lost.

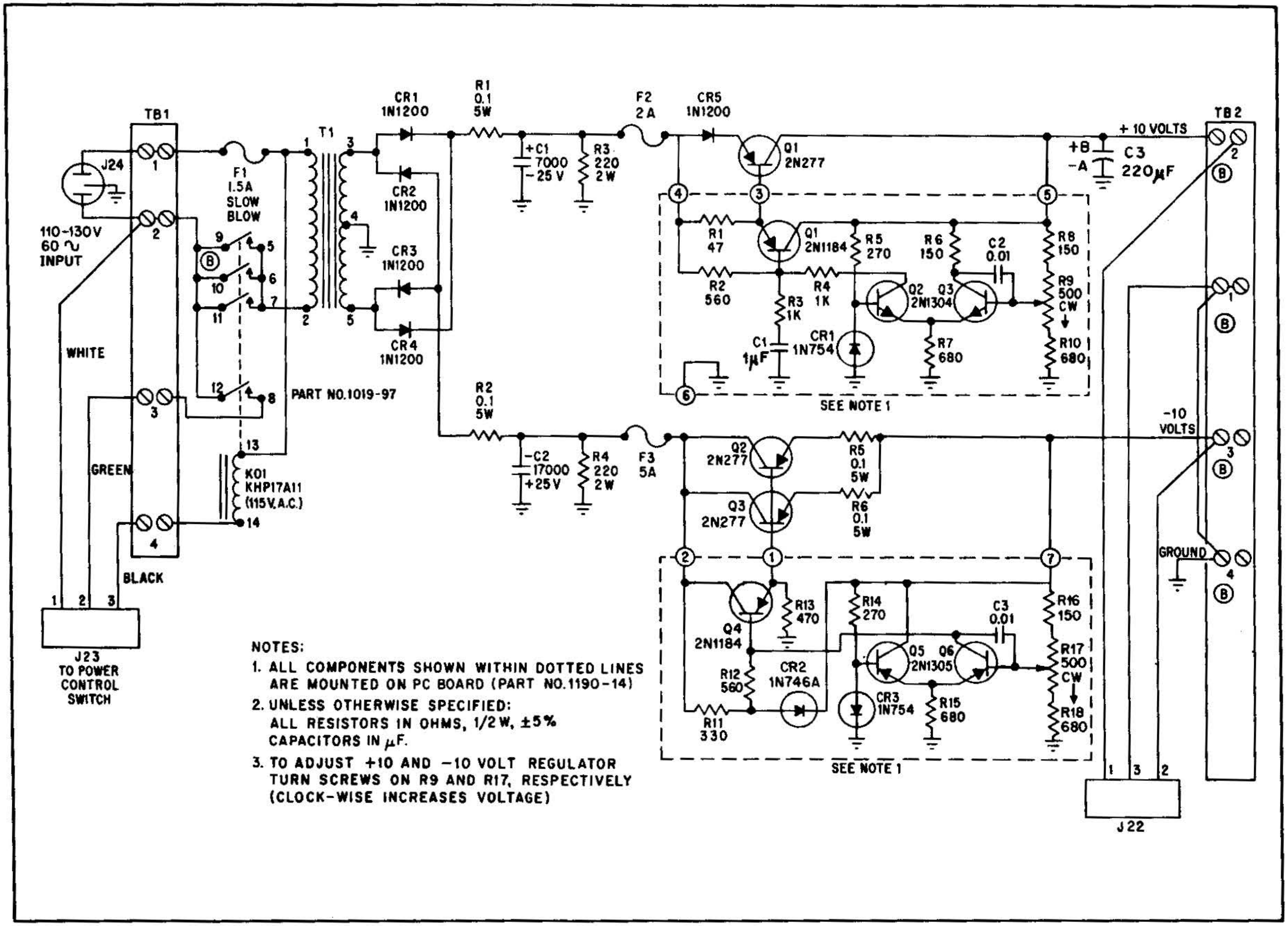
4-8.15. POWER SUPPLY

The power supply (Figure 4-23) provides all of the operating voltages for the trainer. It consists essentially of a regulated +10-volt section, a regulated —10-volt section, and a sequencing circuit. The sequencing circuit is interwired with the POWER control switch, and together they control the sequence in which power is applied to the circuits of the computer.

AC input power to the supply is applied to the primary of the transformer TI through relay KO-1 and the POWER control switch. When the POWER switch is turned to its first ON position, terminal 1 of J23 is connected to terminals 2 and 3. This places the winding of relay KO-1 across the AC line, energizing the relay and causing its contacts to close. When the POWER switch is then turned to its subsequent ON positions, terminal 1 of J23 is no longer connected to terminals 2 and 3, but terminals 2 and 3 are connected together. As a result, the winding of KO-1 is in series with closed contacts 8-12 across the AC line, so KO-1 remains energized and its contacts are kept closed. If AC power is lost, the POWER switch must be recycled to turn power back on. The input voltage applied across the primary of transformer TI is stepped down in the centertapped secondary, and applied in parallel to the negative and positive rectifier sections connected across the secondary.

The postive rectifier section consists of two diodes, CR1 and CR4, connected in a full-wave circuit. The rectified voltage is filtered by capacitor C1 and resistor R3, and applied to terminal 2 of terminal board TB2. A portion of the +10-volt output is tapped from the R8, R9, R10 voltage divider and applied to a voltage regulator circuit consisting of transtors Q1, Q2, and Q3. Variations in this tapped voltage are amplified, and applied to the base of transistor Q1, which is in series with the supply. The amplified voltage changes vary the emitter-collector resistance of Q1 in such a way that the +10-volt output is held constant.

The section of the power supply that produces the —10 volt output is similar to that which produces the +10-volt output. The AC voltage is rectified by diodes CR2 and CR3. Filtering is accomplished by capacitor C2 and resistor R4. The —10 volts is regulated by transistors Q2 through Q6. Two parallel transistors, Q2 and Q3, are used in series with the output because of the current requirements of the negative section.



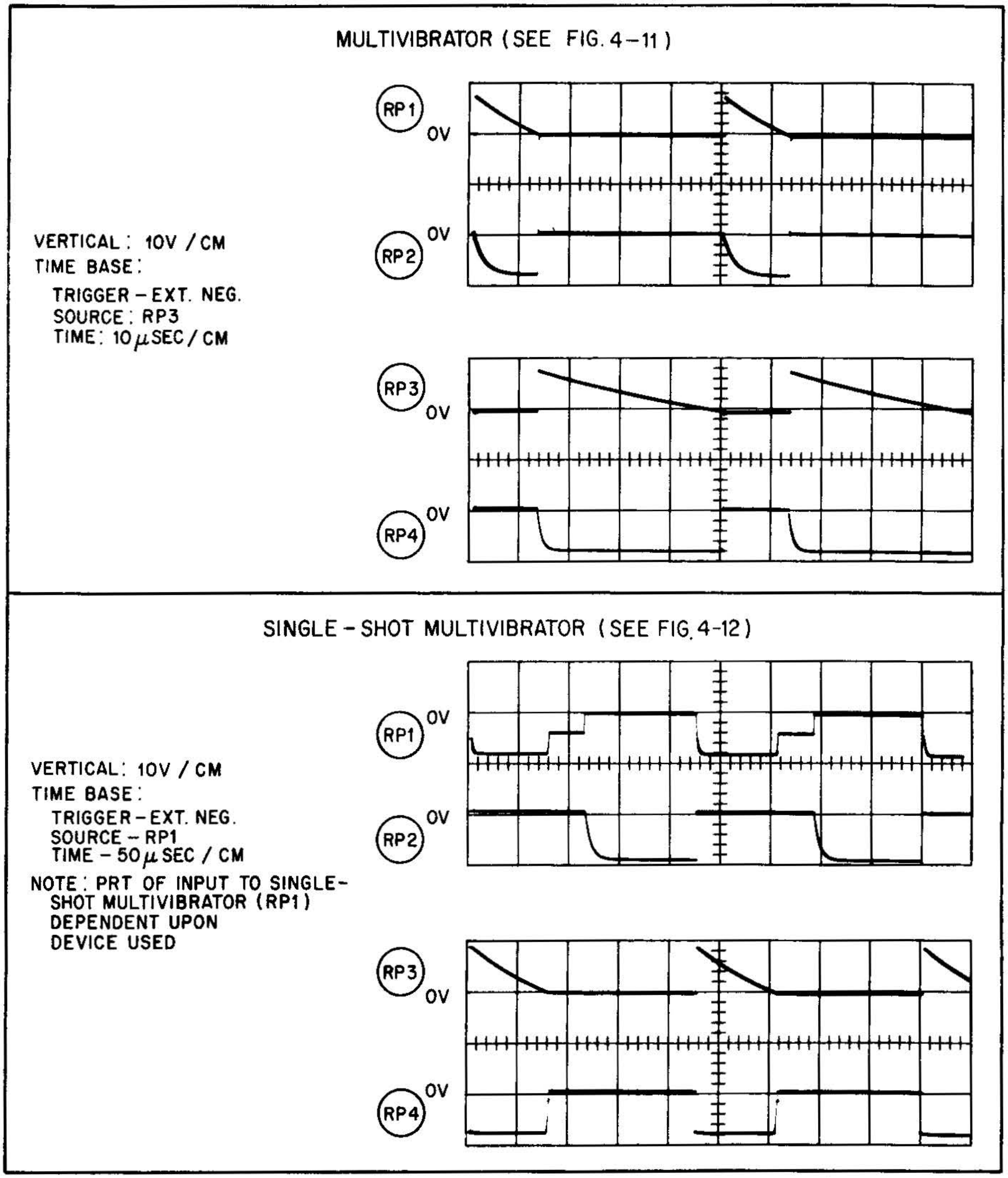
NOTES:

1. ALL COMPONENTS SHOWN WITHIN DOTTED LINES ARE MOUNTED ON PC BOARD (PART NO.1190-14)
2. UNLESS OTHERWISE SPECIFIED:
   ALL RESISTORS IN OHMS, 1/2 W, ±5%
   CAPACITORS IN μF.
3. TO ADJUST +10 AND -10 VOLT REGULATOR TURN SCREWS ON R9 AND R17, RESPECTIVELY (CLOCK-WISE INCREASES VOLTAGE)

Figure 4-23. Power Supply Circuit

WAVEFORMS

The wave-forms are keyed to the circuit diagrams that appear earlier in this section.



MULTIVIBRATOR (SEE FIG. 4–11)

VERTICAL: 10V / CM
TIME BASE:
TRIGGER – EXT. NEG.
SOURCE: RP3
TIME: 10μSEC / CM

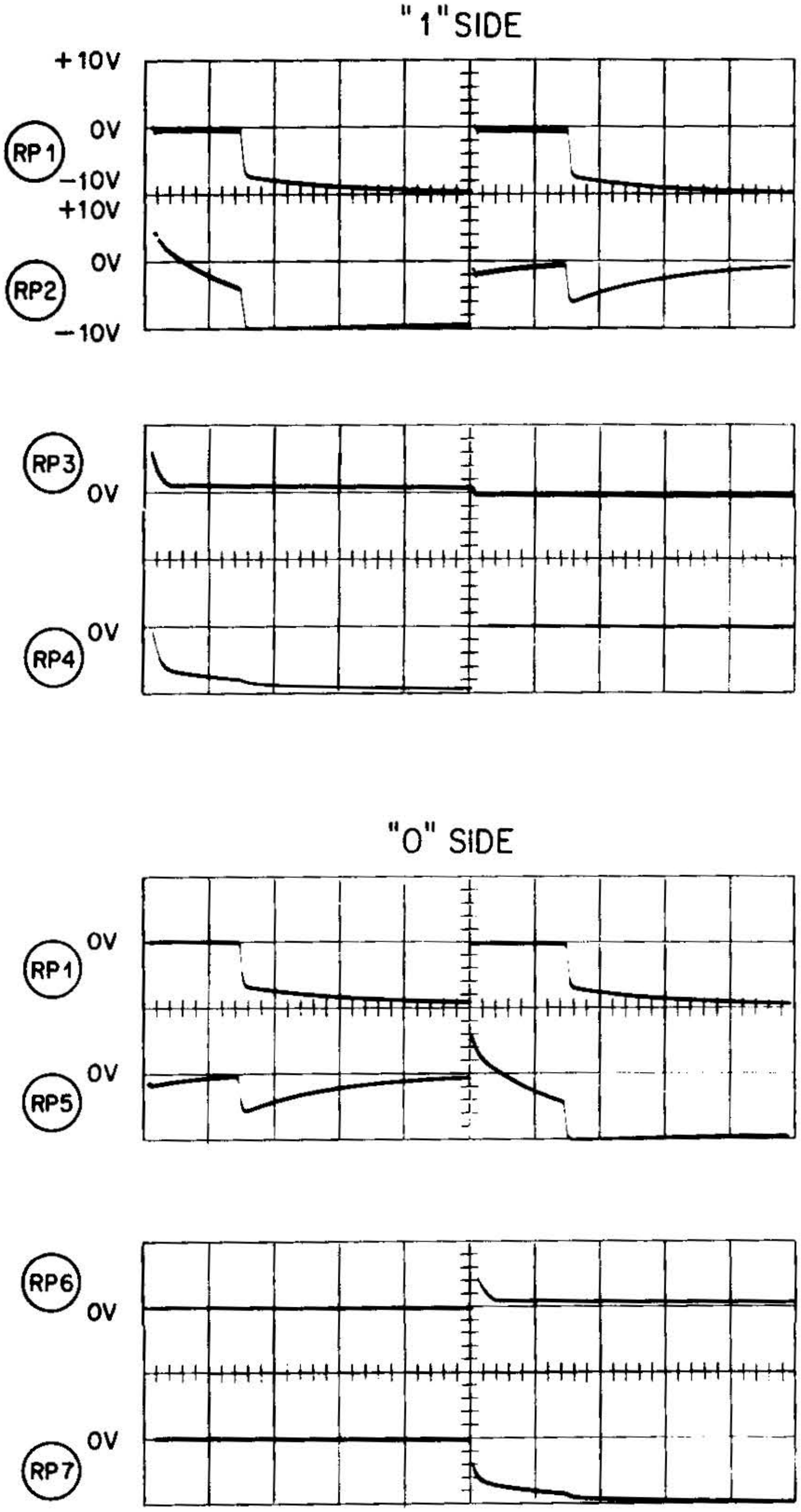
SINGLE – SHOT MULTIVIBRATOR (SEE FIG. 4-12)

VERTICAL: 10V / CM
TIME BASE:
TRIGGER – EXT. NEG.
SOURCE – RP1
TIME – 50μ SEC / CM

NOTE: PRT OF INPUT TO SINGLE-SHOT MULTIVIBRATOR (RP1) DEPENDENT UPON DEVICE USED

Figure 4-24. Circuit Waveforms (Sheet 1 of 7)

FLIP-FLOP-A_5 (SEE FIG. 4-14)



VERTICAL: 10V / CM
TIME BASE:
TRIGGER-EXT. POS.
SOURCE- $\overline{A_5}$
TIME-10 μ SEC / CM

Figure 4-24. Circuit Waveforms (Sheet 2 of 7)

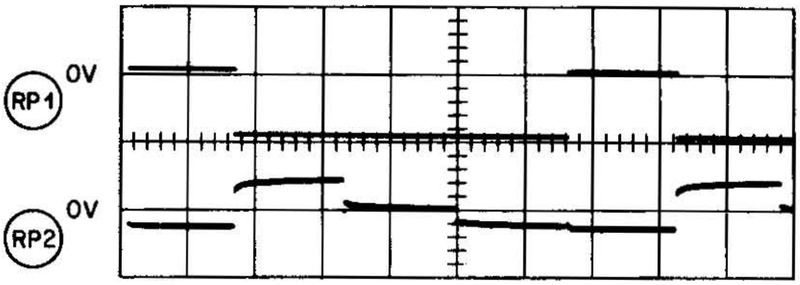
SELECTION MATRIX TRANSFORMER-YT_{00} (SEE FIG 4-15)

VERTICAL: RP1 = 10V / CM
RP2 = 2V / CM

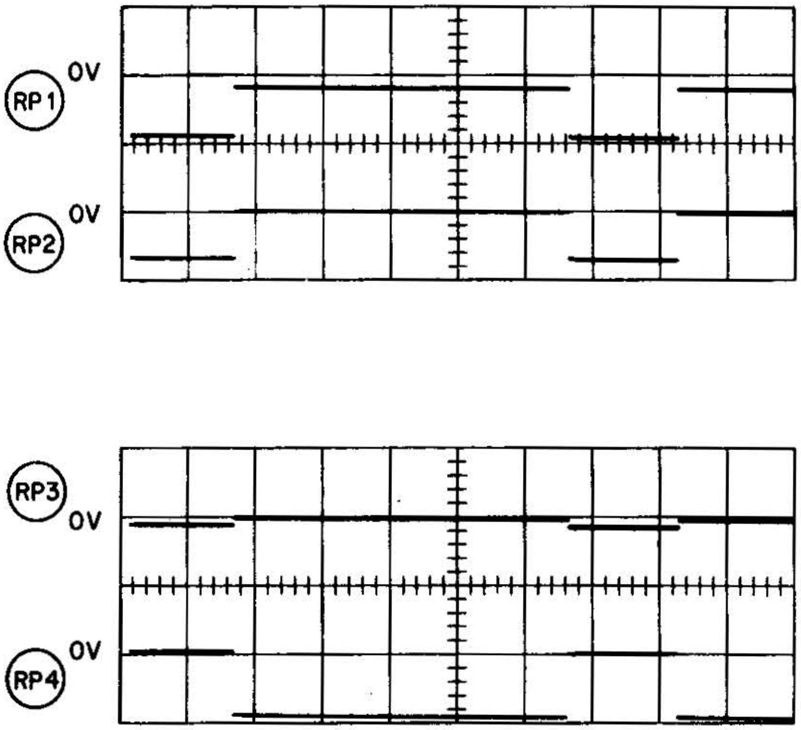
TIME BASE:

TRIGGER-EXT. NEG.
SOURCE-M_0
TIME-0.5 M SEC/CM

NOTE: AEMR
AE = 1
I REG. = 70



LINE SELECTOR-YS_0 (SEE FIG. 4-16)



VERTICAL: 10V / CM

TIME BASE:

TRIGGER-EXT. NEG.
SOURCE-M_0
TIME-0.5 M SEC / CM

NOTE: AEMR
AE = 1
I REG. = 70

Figure 4-24. Circuit Waveforms (Sheet 3 of 7)

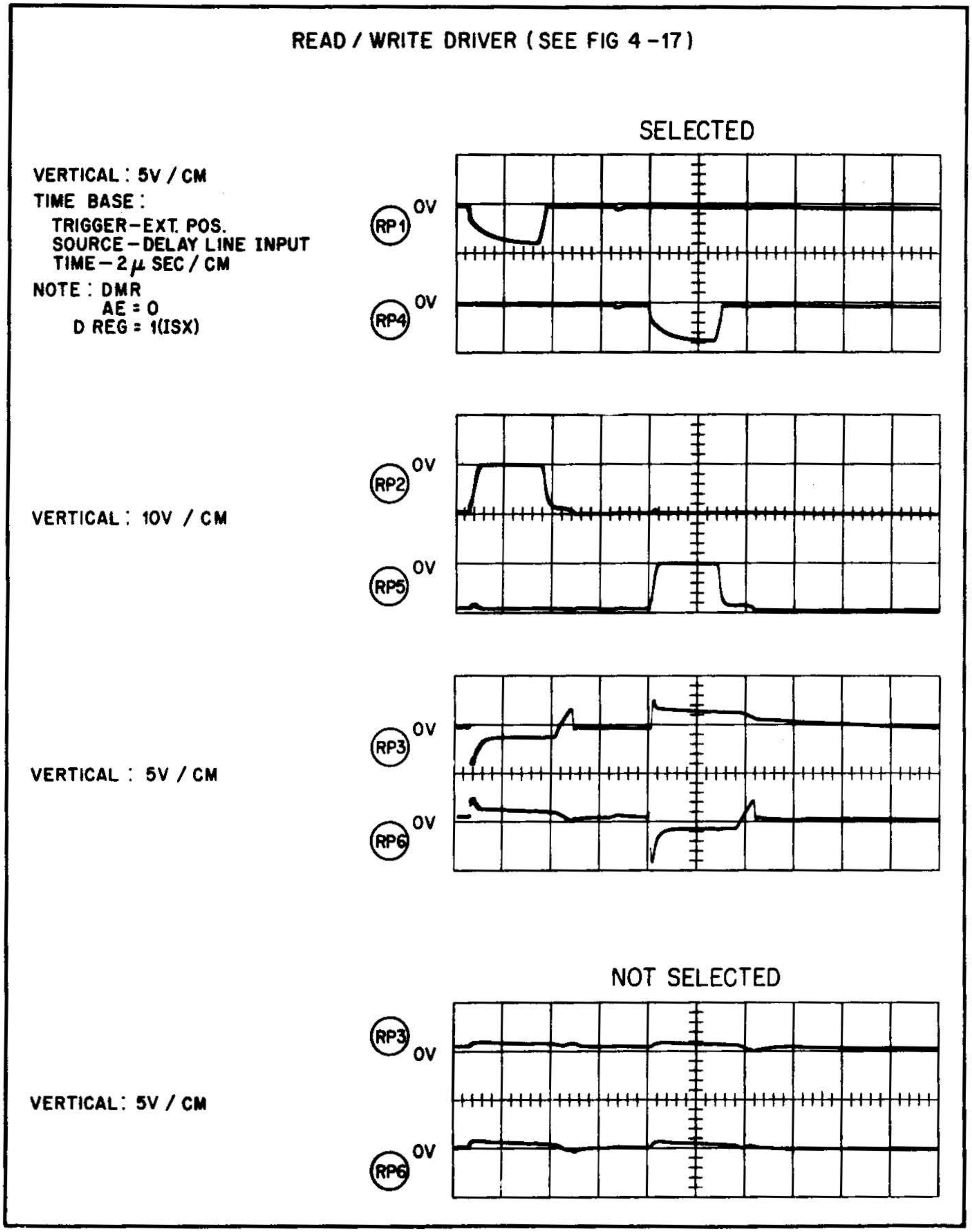


Figure 4-24. Circuit Waveforms (Sheet 4 of 7)

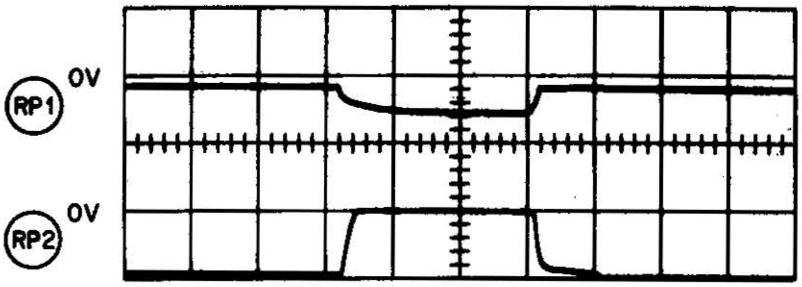
INHIBIT DRIVER (SEE FIG. 4-19)

VERTICAL: 10V / CM

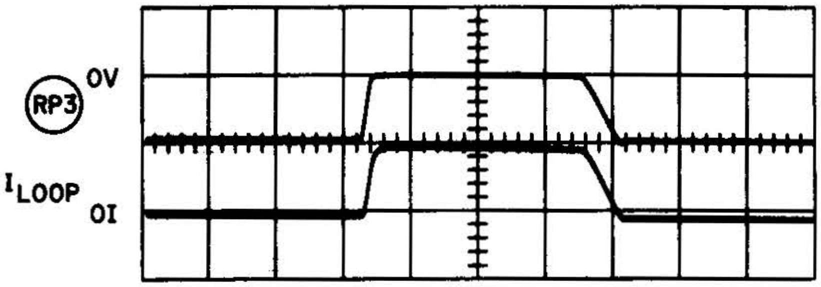
TIME BASE:

TRIGGER-EXT. POS.
SOURCE-DELAY LINE INPUT
TIME-2 μ SEC / CM

NOTE: DMR
. AE = 0
D REG = 1(ISX)



I_{LOOP} = 200 MA / CM VERTICAL DEFLECTION

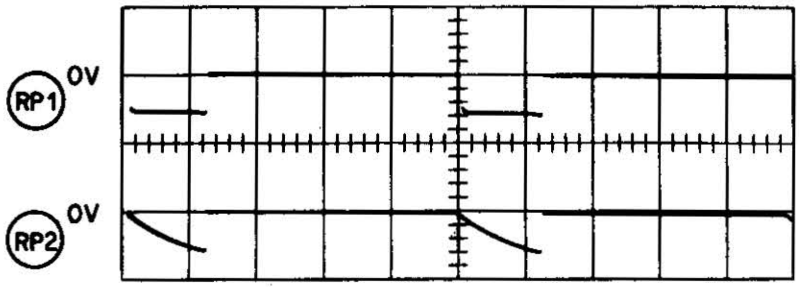


DELAY CIRCUIT (SEE FIG. 4-20)

VERTICAL: 10V / CM

TIME BASE:

TRIGGER - EXT. NEG.
SOURCE-RP1
TIME-10 μ SEC / CM



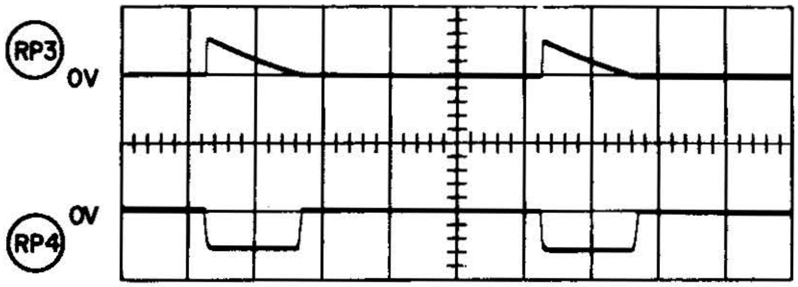


Figure 4-24. Circuit Waveforms (Sheet 5 of 7)

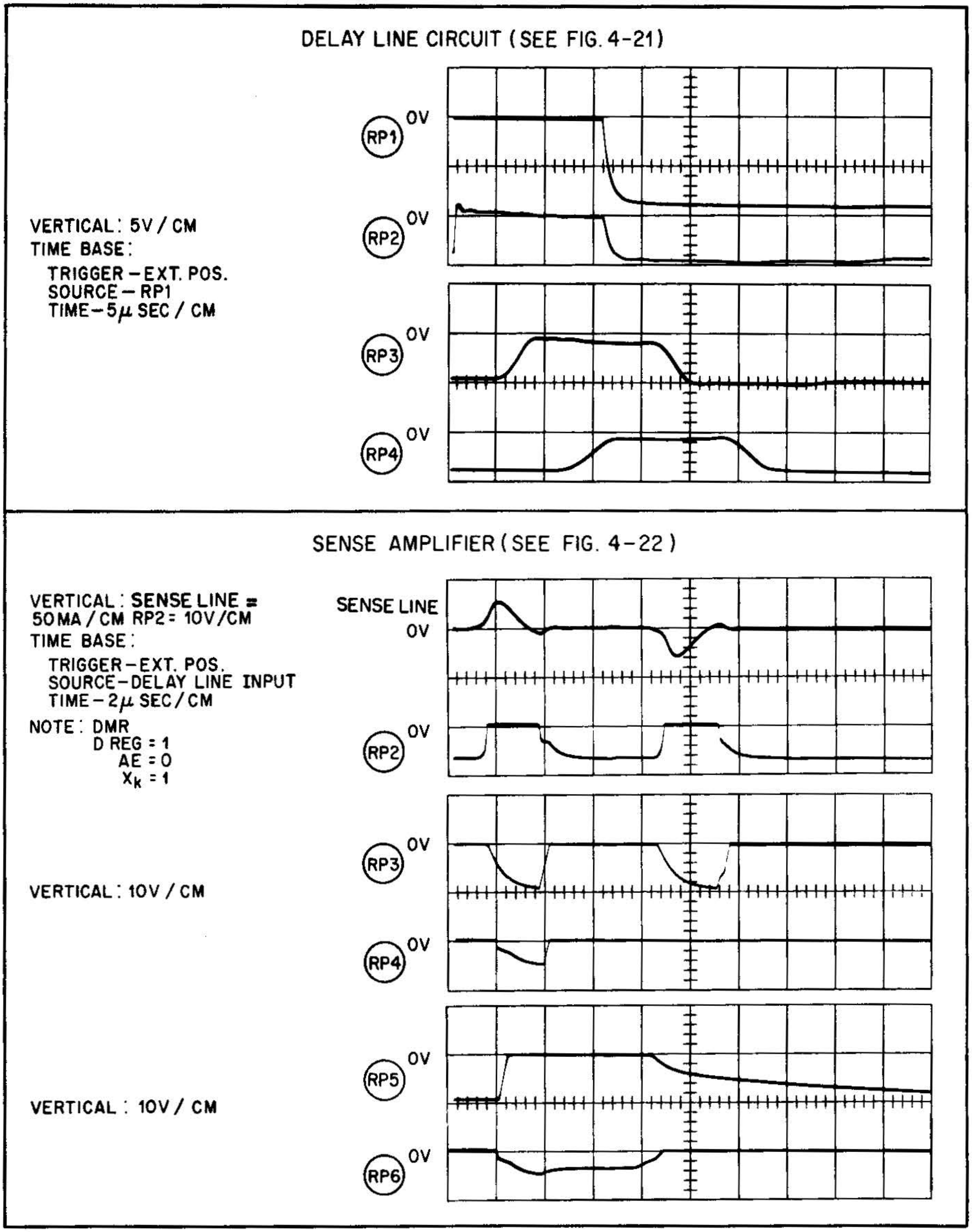
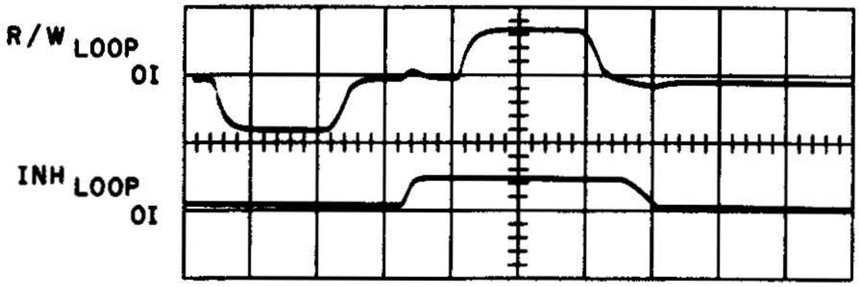


Figure 4-24. Circuit Waveforms (Sheet 6 of 7)

MEMORY READ/WRITE CURRENT LOOP AND INHIBIT CURRENT LOOP

VERTICAL: 500MA/CM
TIME BASE:
TRIGGER – EXT. POS.
SOURCE – DELAY LINE INPUT
TIME – 2μSEC/CM
NOTE: DMR
AE = 0
D REG. = 1



MEMORY TIMING (READ, STROBE, WRITE, AND INHIBIT)

VERTICAL: 10V/CM
TIME BASE:
TIGGER – EXT. POS.
SOURCE – DELAY LINE INPUT
TIME – 2μSEC/CM
NOTE: DMR
AE = 0
D REG. = 1

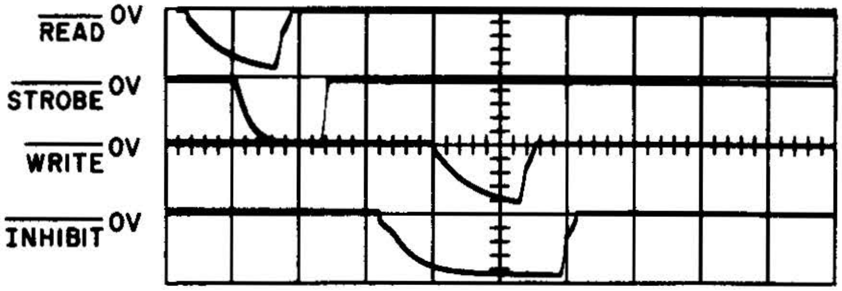


Figure 4-24. Circuit Waveforms (Sheet 7 of 7)

Section V
MAINTENANCE

5-1. PERIODIC INSPECTION AND PREVENTIVE MAINTENANCE

The BI-TRAN SIX requires no special care. However, it should be inspected and cleaned at least twice a year according to the following procedure:

(1) With the POWER switch ON, check each switch-light and switch as described in the preliminary operating inspection, paragraph 2-1.2. Remove dirt and dust from the front panel and cabinet with a damp cloth.

(2) With the POWER switch OFF and the trainer disconnected from the AC power source, remove the front panel (paragraph 5-3), raise all the circuit boards, and clean the inside of the cabinet with a soft brush or vacuum cleaner.

(3) Lower and raise each board to check that it does not bind and that it locks firmly in its raised position. Lubricate slide assembly if necessary. Replace the front panel.

(4) Check the power supply output voltages (paragraph 5-2).

WARNING

While no dangerous voltages are present on the circuit boards, 115 volts, which can be dangerous, is present in the power supply and on the power switch.

5-2. POWER SUPPLY ADJUSTMENT

The power supply has a +10-volt DC regulated output and a —10-volt DC regulated output. For proper operation, these voltages should be within ±1 volt of nominal voltage. For the power supply to regulate properly, the AC line voltage must be between 110 and 130 volts AC. To check the +10-volt and —10-volt DC outputs, perform the following procedure:

(1) With a voltmeter rating of at least 20K ohms/volt, measure the +10-volt output at terminal 2 on terminal board TB2, or preferably on the +10-volt bus on Board 1.

(2) Rotate the +10 VOLT ADJUST until the proper output voltage is obtained.

(3) Measure the —10-volt output at terminal 3, or preferably on the —10-volt bus on Board 1.

(4) Rotate the —10 VOLT ADJUST until the proper output voltage is obtained.

5-3. REMOVING FRONT PANEL

(1) Turn the POWER switch to OFF and disconnect the computer from the AC power source.

(2) Grasp the bar that separates the upper and lower sections of the front panel, and lift the panel straight up as far as it will go. Slide the bottom of the panel toward you until it is clear of the cabinet; then lower the whole panel, being careful not to strain the panel-to-cabinet cables.

(3) If it is necessary to completely remove the front panel, disconnect the four connectors of the panel-to-cabinet cables.

(4) Replace the panel in the opposite manner, making sure that no cables are caught between panel and cabinet, and that the panel is properly seated.

5-4. REMOVING POWER SUPPLY

(1) Turn the POWER switch to OFF and disconnect the computer from the AC power source.

(2) Remove the cover on the rear of the cabinet.

(3) Remove the bolts that hold the power supply in place.

(4) Disconnect the power supply connectors.

(5) Carefully slide the power supply out of the cabinet.

(6) Replace the power supply in the opposite manner. Be sure to insert the bolts that hold the power supply in place.

5-5. REMOVING FUSES

Three fuses protect the computer's circuits from current overload damage. They are located on the rear of the computer. To remove one, depress and rotate the screw cap half a turn counterclockwise. When replacing a defective. fuse, make sure that the replacement has the identical current rating. Never replace a fuse with one having a higher current rating.

5-6. SWITCH-LIGHT LAMP AND SWITCH-LIGHT REMOVAL AND REPLACEMENT

5-6.1. SWITCH-LIGHT LAMP REMOVAL AND REPLACEMENT

Lamps may be removed from the switch-lights and replaced as follows:

(1) Remove the lens by grasping it with your fingers and pulling.

(2) Remove the lens spring.

(3) Remove the lamp by grasping it with the lamp tool and pulling.

(4) Insert a new lamp.

(5) Insert the spring with the widest spiral out.

(6) Insert the lens. If the switch contact or the spring-shoulder has been removed from the lens, reinsert before inserting lens. The contact must not be installed in the wrong slot, as this can damage the lamp driver. Insert the contact so it will be to the left of the lamp when the lens is installed. Do not bend the contact when inserting.

5-6.2. SWITCH-LIGHT REMOVAL

(1) Remove the lamp driver from the switch-light by pulling them apart.

(2) Remove the switch-light from the panel by compressing the springs while pushing it out.

5-7. TROUBLESHOOTING TECHNIQUES

5-7.1. TROUBLE INDICATIONS

The following may be considered typical machine failures:

(1) Loss or gain of data in register to register transfers.

(2) Loss or gain of data in memory to X, or X to memory, transfers.

(3) Incorrect memory addressing, i.e., data from locations other than the one specified by the M register to and from the X register.

(4) Loss or gain of data during I/O transfers.

(5) Loss of Control;

(a) D, AC, C, M, or P registers fail to count correctly.

(b) I register is decoded incorrectly.

(c) D register is decoded incorrectly.

(d) Commands, subcommands or DP's fail or are generated at the wrong time or during the wrong instructions. (Related to items (b) and (c)).

(6) Sums are formed in the A register incorrectly.

5-7.2. OPERATOR ERRORS

Generally, machine failures will be discovered when a given program does not operate as intended. Before troubleshooting the machine, the program must be examined for errors, including:

(1) Arithmetic errors:

(a) Numbers whose sum or quotient exceeds ±31 are added or divided.

(b) Numbers are not properly scaled.

(2) Instruction coding errors:

(a) Incorrect jump addresses (jump address = ½ actual address.)

(b) Incorrect operand addresses.

(c) Incorrect code for instructions; use of an even code when an odd code is needed, etc.

(3) Program coding:

It is difficult to list typical program coding errors. In general, all but very simple programs will have errors when first developed and must be debugged on the machine.

(4) Operator errors:

(a) Machine set up incorrectly for loading program.

(b) Incorrect response to manual inputs in program.

(c) I/O equipment not set up incorrectly.

(d) Operating switches on front panel not set correctly.

5-7.3. TROUBLESHOOTING: GENERAL

When it is determined that the machine and not the program is failing, several steps must be taken to localize the malfunction:

(1) Determine which instruction or instructions in the program are failing.

(2) Determine the nature of the failure:

(a) Is only one instruction failing or are several?

(b) Do all instructions fail?

(c) Are the instructions and operands properly acquired from memory?

(d) Is the problem in the adder or arithmetic control? Does the failure only occur with a certain combination of numbers?

(3) Through the process of elimination, use of the logic timing charts, and logic analysis, determine the component that is failing. This will require the use of test equipment and, in some cases, maintenance programs, unless the trouble need only be isolated to a single board. In this case, interpretation of the register indicators and use of the programs and timing charts will be sufficient.

5-7.4. DETERMINING FAILING INSTRUCTIONS

When a program (known to be error-free) fails, first determine where the failure occurred. If an error stop occurs, the failing instruction will be in the I register. If the failing instructions do not cause an error stop, the program must be operated manually: set the MODE switch to INSTRUCTION, and run the program. As each instruction is executed, interpret the register indicators and note any errors.

If this method doesn't locate the failing instructions, there are several other possibilties:

(1) The failure may only occur with certain combinations of numbers in the arithmetic section. Run the arithmetic maintenance program in order to use all combinations.

(2) The failure may be intermittent. Troubles of this type are the most difficult to locate. Useful techniques include margining the supply voltages and/or careful interpretation of the registers to determine which instruction may have failed and caused the evident symptoms. It is in this area that maintenance programs can be most useful.

(3) The failure may only occur in PROGRAM mode. This usually indicates memory problems.

5-7.5. LOCATING MALFUNCTIONS

5-7.5.1. General

Once it is known which instructions fail, the cause of the failure must be determined. The method of troubleshooting depends upon the nature of the trouble. The general technique is to place the computer in DISTRIBUTOR mode and single-step through the failing instructions, using the logic timing chart to determine which subcommands are not being carried out. Further analysis should reveal the cause.

5-7.5.2. Instruction Analysis Technique

When the computer is single stepped through an instruction, the results of each subcommand must be interpreted to define the malfunction. The acquisition phase is used here to illustrate the general technique. Since it is common to all instructions, however, it would serve no purpose to troubleshoot the acquisition phase unless all instructions, or a group of related instructions, fail.

(1) *Setup:*

(a) Press master CLEAR switch.

(b) Set P register to address of stored instruction unless it is not desired to check the memory and X register.

(c) Set MODE switch to DISTRIBUTOR.

(d) Set 77_8 into the M, X, and I registers, and set the INSTRUCTION ERROR flip-flop by pressing the INSTRUCTION ERROR switch-light.

(2) *Procedure:* Check that the following operations occur each time the START switch is pressed:

D = 0—The M register will clear and the contents of P will transfer into M_{1-6}. For a complete check, repeat DP_0 using the combination $P = 0$, $M = 77$, and the combination $P = 77$ and $M = 0$; then reset M to the desired address.

D = 1—The X register will clear and the contents of the location specified by M will load X. If X is not loaded with the anticipated data, there are several malfunction possibilities;

(a) The memory addressing circuits (including M) are malfunctioning, and a location other than the one specified by M has been read out.

(b) The memory circuits (including X) are malfunctioning, and bits have been gained or dropped.

(c) The desired data was not originally stored in the cell being addressed.

D = 2—The I register and IE should clear. If any flip-flop stays set, the trouble is associated with that flip-flop. If all flip-flops stay set, the CLI subcommand is suspect.

D = 3—The contents of X should transfer into I. For a complete check, repeat DP3 using the combination X = 0 and X = 77. Once again, if individual bits are dropped or gained, the trouble is probably associated with the affected flip-flop. If all bits are dropped, the subcommand TXI is probably failing.

D = 4—IE should set only with an illegal code (0 through 7 or 30 through 37) in I.

D = 5—P should increment. For a complete check of P, MODE REPEAT should be used.

D = 6-11—No subcommands generated.

D = 12—M_0 should set.

D = 13—The X register should clear and the contents of the location specified should load X. Refer to DP = 1.

D = 14 & 15—No subcommands generated.

D = 16—These subcommands have no function during the acquisition phase, but can be checked by setting the flip-flops before generating DP16.

D = 17—M should clear, and the contents of X should transfer into the six least significant bits of M, while the contents of I_0 transfers into M_6. For a complete check, repeat DP17 using the combinations $I_0 = 1$ and $X = 77$, $M = 0$, and $I_0 = 0$, $X = 00$, $M = 77$. AE should complement. CLEL has no function during acquisition.

5-8. MALFUNCTION ANALYSIS

Generation of a given subcommand causes an action to occur in the computer. The occurrence of this action will usually be shown on the register indicators. When it is apparent from the register indicators that a subcommand was not carried out correctly, or that a given action occurred when its associated subcommand should not have been generated, the cause of the malfunction must be determined. Typically, the trouble will be:

(1) The subcommand was not generated, due to a failure in the I register decoding or the logic control circuits.

(2) The subcommand was generated correctly, but the circuits being actuated failed.

(3) The subcommand was generated when it should not be present due to a failure in the I register decoding or logic control circuits.

(14) The subcommand was not generated, but a given action occurs anyway, due to a failure in the circuits being actuated.

5-9. ANALYSIS OF TYPICAL DATA TRANSFER FAILURES

The X register will serve to illustrate typical data transfer troubles. Refer to the Board 6 logic diagram. Because the logic configuration of all X registers flip-flops is identical, with the exception of the CX_0 and SX_5 inputs, X_5 and its input gates will be the reference. Thus, gate 6-1-1 typifies gates 6-1-5, 9, 13, 17, and 21.

(1) *All 1 bits lost, any-register to X register transfer.* This is probably a malfunction of an input common to all X flip-flops. The inputs TEX, TMX, TAX, and TQX are common to all flip-flops. However, because of the isolating properties of the gates, any one of these inputs (regardless of its condition) could not inhibit all transfers. It is not likely that all four subcommands would be lost either. Gates 6-1-54, etc., OR these inputs and the possibility exists of all inputs to a given flip-flop being lost here, but not to all "X" flip-flops. The positive OR inputs to each X flip-flop are also independent of each other and can be disregarded. The complement input cannot hold a flip-flop set or clear, and unwanted pulses here would cause the register to complement. The clear input is also common to all flip-flops. If gate 6-1-25 always outputs 0 volts, the X register will be held clear. This is the probable malfunction. It should be noted that when a flip-flop is being held clear, a negative pulse into the negative OR set input will drive the set side into conduction, and both sides will output 0 volts while the negative set and positive clear levels are both present.

(2) *All "1" bits lost, given-register-to-X register transfer.* Assume all "1" bits from Q are lost on the Q to X transfer. A malfunction common to all bits of the Q to X transfer path is indicated. The subcommand TQX is the only input common to all bits in the Q to X path and probably is not being generated, causing this malfunction.

(3) *One bit lost, on all registers-to-X transfers.* Assume that X_5 cannot be set by any data transfer. A malfunction common to all of the X_5 input gates is indicated. The only component common to all X_5

inputs is gate 54. It is not likely that gate 54 is the source of this trouble, however, because all five of its diodes would have to be open. Therefore, the trouble is probably within flip-flop X_5.

(4) *One bit lost from a given register transfer to X.* Assume that X_5 cannot be set by subcommand TQX when Q_5 is set. A malfunction common only to the Q_5 to X_5 transfer path is indicated. If the subcommand TQX was inoperative, all bits would be lost. If gate 4 has an open diode, data would be gained rather than lost. Gate 54 is probably bad. Specifically, diode 6-1-54.4 is open.

(5) *One bit gained during a specific register-to-X register transfer.* Assume X_5 sets whenever subcommand TQX is generated, regardless of the state of Q_5. Gate 4 is probably inoperative. The function of any AND gate is to inhibit the passage of signals unless all inputs are present. In this case, diode 6-1-4.2 being open would allow a negative output whenever TQX was generated, setting F-F X_5.

(6) *A given bit transferred into the X register, when no transfer subcommand is generated.* Assume X_5 sets whenever Q_5 is set, and the TQX input to gate 4 is 0 volts. The trouble probably is that diode 6-1-4.1 is open, allowing a negative level to pass (setting X_5), whenever Q_5 is set.

5-10. ANALYSIS OF CONTROL MALFUNCTIONS

Since the control section is concerned with the generation of subcommands, malfunctions in the control section will be evident when subcommands are not generated or when they are generated out of order. The failure to generate subcommands correctly can be caused by malfunctions in the subcommand-generating (logic control) circuits, the command-generating (I register decoding) circuits, or in the D register and DP generating circuits.

(1) If all subcommands for a given instruction are lost, or if the subcommands for two instructions are generated simultaneously, the trouble is probably in the command generating circuits on Board 1. Also, if a group of subcommands in related instructions malfunction, the command decoding circuit are suspect. An example of related instructions would be those that generate command ADG. If ADG failed, certain subcommands in the instruction LDA, LDN, ADD, and SUB would fail.

(2) When a given subcommand fails, the subcommand generating (logic control) circuits, located primarily on Board 3, are suspect.

(3) When all subcommands produced by a given DP fail, the DP generating circuits on Board 2 are suspect.

(4) The D register is, of course, subject to the same malfunctions as the other register/counters. Troubles in the D register will generally affect all instructions.

5-10.1. COMMAND GENERATION MALFUNCTIONS

(1) *General:* An open diode in any of the command decoding gates on Board 1 will cause commands to be generated when they should be inhibited. Inoperative inverters on Board 1 will either inhibit given commands, or will enable the wrong command, for a given code. Note that F-F I_0 is not used in command generation.

(2) *Malfunction Examples:* Assume Diode 1-2-62.3 is open. The command CAS and its associated subcommands would also be generated whenever the following instructions are executed.

CAS (74, 75)
SRE (64, 65)
MPY (54, 55)
ADD (44, 45)
EXI (24, 25)
NAE (14, 15)

Assume Diode 1-2-11.3 is open. Command SL (66, 67) will also be generated during the instruction EXO (26, 27). Command SR (64, 65) will be generated during instruction EXI (24, 25); STQ (62, 63) during MNO (22, 23); and command STA (60, 61) during MNI (20, 21).

5-10.2. SUBCOMMAND GENERATION MALFUNCTIONS

(1) *General:* Execution phase subcommands are generated by decoding command levels, distributor pulses and, in some cases, conditional levels such as C Reg = "0". Acquisition phase subcommands are generated by decoding AE and certain distributor pulses. Those subcommands which are used by more than one instruction or more than once in any instruction are encoded as part of the generating process. Decoding and encoding is performed by appropriate use of AND/OR gates. Inverters are used for signal amplification. Trouble, therefore, can be caused by open diodes or inoperative inverters. For

a given subcommand, an open diode in an associated OR gate would cause the loss of the subcommand, and an open diode in an associated AND gate would cause the subcommand to be generated when it should be inhibited. Inoperative inverters can cause either type of trouble, depending upon the location of the inverter and the output ("1" or "0") of the inverter.

(2) *Malfunction examples:* Subcommand AQL should be generated at DP_7 for command DIV and at DP_{14} for command SL. The following malfunctions could be caused by open diodes:

(a) AQL occurs every DP_7, regardless of the command. Diode 3-2-58.2 is open.

(b) AQL occurs every DP_{14}, regardless of the command. Diode 3-2-57.2 is open.

(c) AQL is present as a "1" level during the entire divide execution phase. Diode 3-2-58.1 is open.

(d) AQL is present as a "1" level during the entire execution phase of the shift left instruction. Diode 3-2-57.1 is open.

(e) AQL isn't generated during a divide command, but is generated during the shift left. Diode 3-2-59.2 is open.

(f) AQL isn't generated during a shift left, but is generated during the divide. Diode 3-2-59.1 is open.

Note that inverters 8 and 9 being inoperative could have caused the last two troubles listed. Also, if either inverter 8 or 9 was malfunctioning so as to output —10 volts, regardless of its input, AQL would always be present as a "1" level. Inverter 10 could malfunction so as to always output a "1" or a "0" level.

5-11. ADDER MALFUNCTION ANALYSIS

Although the adder consists of the same types of circuits found throughout the computer, its logical configuration complicates the troubleshooting procedure. Most troubles within the computer can be localized through point-to-point signal tracing, once it is determined which subcommand is not carried out or which flip-flops do not set, etc. As a general rule, intelligent interpretation of the register indicators and other symptoms localize the malfunction to a few possibilities, and analysis of these possibilities with test equipment can be rapidly accomplished. Basically, this is also true of trouble analysis within the adder. However, because there are so many variables used in generating the adder outputs, analyzing an adder malfunction before attempting to use test equipment is of greater importance and more difficult than it is in other sections. Also, it is not always readily apparent whether trouble is in the adder or in the A register.

5-11.1. ADDER ANALYSIS TECHNIQUES

The initial steps in troubleshooting the adder are determined by the conditions that indicate it may be failing. Failures that are apparent during adder control only are almost certainly within the adder or the AC or A registers. If an arithmetic instruction fails, it should be executed in DISTRIBUTOR mode. Obviously, if a subcommand such as ISX is not carried out, the adder is not involved in the failure. The failure of a subcommand, such as INCA or CAM, to be carried out could be due to a malfunction within the adder, or due to a malfunction in the logic control circuits.

Example: Assume I = 44 (ADD), D = 7, A = 01 and OV = 1, the computer is in DISTRIBUTOR mode. Upon start, INCA is not carried out. Sums that do not produce end carries are formed correctly.

Analysis: Repeat DP_7 with AM = 37. If some of the AM flip-flops complement, then INCA is being generated and the trouble is probably in the adder. If none of the AM bits complement, INCA probably was not generated by logic control. Since the AM flip-flops toggle correctly when end carry is not involved, they must be all right.

5-11.2. WRONG SUM IN THE ACCUMULATOR

If it is determined that the sum formed in the accumulator is not correct for a given value of AM and XM, the adder should be exercised to determine all the configurations of AM and XM that will not add correctly. Since there are 1024 possible combinations, the adder test program should be used. The initial contents of AM and XM and the sum should be recorded for each failing combination. Once the failing combinations are known, it can be determined which adder output malfunctions.

Example: Assume that all sums formed from numbers that have both A_4 and X_4 set (before INA) are incorrect except for the combinations that also have A_5 and X_5 set, which add correctly. All other combinations add correctly:

Typical sums formed

$$\begin{array}{lll}
\text{A. } A = 00010 & \text{B. } A = 00010 & \text{C. } A = 00011 \\
\phantom{\text{A. }} X = 00010 & \phantom{\text{B. }} X = 01110 & \phantom{\text{C. }} X = 00011 \\
\hline
\phantom{\text{A. }} A = 00000 & \phantom{\text{B. }} A = 01100 & \phantom{\text{C. }} A = 00110 \\
\\
\text{D. } A = 00011 & \text{E. } A = 00010 & \text{F. } A = 00010 \\
\phantom{\text{D. }} X = 00011 & \phantom{\text{E. }} X = 00110 & \phantom{\text{F. }} X = 01110 \\
\hline
\phantom{\text{D. }} A = 00100 & \phantom{\text{E. }} A = 00100 & \phantom{\text{F. }} A = 01100
\end{array}$$

The sums formed in A_4 and A_5 are always correct. Examination of the failing combinations reveals there is no carry propagated to A_3 (Ex. A + B), or to $A_{3\&2}$ (Example E), or to $A_{3,2,\&1}$ (Example F), when the combination of $A_4 \cdot X_4$ is to generate the carry. This is group carry G_4. Note that G_4 is evaluated at T_2, after addition with inner-group carries. The failing function of G_4, therefore is $X_4 \cdot \overline{A_4}$. This function is decoded by AND gate 5-1-52. An open diode here would not inhibit G_4, however. Either inverter 5-1-5 outputs 0 volts, regardless of its input, or diode 5-1-53.2 is open.

5-11.3. DETERMINATION OF FAILING ADDER CIRCUITS

When the symptoms have been analyzed, as above, normal troubleshooting techniques are used. Naturally, all troubles cannot be localized to a specific gate, etc., merely by interpreting the failing combinations. To determine the failing circuits:

(1) Determine which A register flip-flops are not being toggled correctly with the failing configurations of A and X.

(2) Use the Boolean equations and the logic diagrams to analyze the possibilities.

(3) Set the computer to DISTRIBUTOR mode, and set

$$\begin{array}{l}
I = 44 \\
D = 5 \\
\left.\begin{array}{l} X = \\ A = \end{array}\right\} \text{failing combination}
\end{array}$$

(4) Determine which T_k is not being carried out correctly.

(5) Set the MODE REPEAT switch, and set AC = T_k where the failure occurs. Examine the circuits with an oscilloscope. Since the A register is an input to the adder, and the outputs from the adder complement the A register flip-flops, it may be necessary to manually hold the A register in the desired configuration.

5-12. MEMORY TROUBLESHOOTING

5-12.1. GENERAL

Memory problems are often difficult to analyze because:

(1) The memory system contains the most electronically complex circuits in BI-TRAN SIX.

(2) In memory, relative current amplitudes or directions are the determining factor as to whether a "1" or a "0" is present. Generating and detecting these currents requires more precise control, with less allowable variation, than in the rest of the computer.

(3) Ambient electromagnetic fields can affect memory operation.

(4) Some memory wave shapes, such as the outputs of the line selectors, are difficult to interpret.

5-12.2. INTERPRETING MEMORY TROUBLES

When a trouble occurs in memory, it should be analyzed as follows: Is one bit or are all six bits of an address affected? Which addresses fail? Is the trouble random, intermittent, or catastrophic? Before troubleshooting memory, the supply voltages should be checked. Variations in these voltages will usually influence memory operation before affecting the rest of the computer.

5-12.2.1. Failure of the Write (IXS) Cycle Only—all bits, all addresses

(1) Subcommand IXS not generated, or is lost at the OR gate input to the delay line.

(2) The read/write flip-flop fails to clear, but IXS does actuate the delay line. A read cycle will occur, with logical addition of memory data and the data in the X register.

5-12.2.2. Failure of the Read (ISX) Cycle Only—all bits, all addresses

(1) Subcommand ISX is not generated, or is lost at the OR gate input to the delay line.

(2) The read/write flip-flop fails to set, but ISX does actuate the delay line. A write cycle will occur. Since CLX will also be generated, the addressed cell will be cleared.

(3) Read flip-flop does not set, or $\overline{\text{READ}}$ does not

actuate the R/W drivers. Although this trouble would also occur during the write (IXS) cycle, it would have no effect.

(4) Strobe flip-flop doesn't set, or $\overline{\text{STROBE}}$ does not actuate the sense amplifiers. See above item.

5-12.2.3. Failure of the Read (ISX) and the Write (IXS) Cycles

(1) *All "1" bits lost.* Write flip-flop does not set, or $\overline{\text{WRITE}}$ does not actuate the R/W drivers. Although this would allow the read cycle to obtain data from a given cell, the data would not be restored, and all "1" bits in all addresses would be lost.

(2) *All "0" bits lost.* Inhibit flip-flop does not set, or $\overline{\text{INHIBIT}}$ does not actuate the inhibit drivers. All "1's" would be written into the selected cell.

(3) *A given bit always read as "0"—all addresses.* Assume that bit 0 will not set X_0. Any of the following could be the trouble:

(a) Sense amplifier 0 inoperative.

(b) Inhibit driver 0 always generates inhibit current. This could be caused by its input gate X_0 diode being open.

(c) The S_0 input diode to X_0 open.

(d) The read/write current is too low, possibly caused by the —10-volt supply output being less than —10 volts.

(4) *A given bit always read as "1"—all addresses*

(a) The associated inhibit driver is inoperative.

(b) Too high a R/W current, possibly caused by the —10-volt supply output being greater than —10 volts.

(c) The associated sense amplifier not sensing zeros.

5-12.2.4. Addressing Failures

The addresses of the cells that fail will define the circuit that is probably malfunctioning. Figure 5-1 illustrates the relationship of the address selection circuits to the core matrix. For example:

(1) Addresses 000 through 037 cannot be read out. Probable failure is R/W driver X_0, as it is the only circuit common to all addresses 000 - 037.

(2) Addresses 130 - 137 cannot be read out. Probable failure is XT13.

(3) Addresses 030 - 037, 070 - 077, 130 - 137, and 170 - 177 cannot be read out. Probable failure is XS3.

Open diodes in the M register decoding gates on Board 8 would cause the selection of two cells at the same time. For example, if the M_0 input diode to R/W driver X_0 (read gate) was open, X_0 would also be actuated when R/W driver X_2 was selected.

5-12.3. MEMORY TEST PROCEDURES

(1) *LDC Quick-Check*

Set-Up:

MODE	= AE
MODE RPT.	= ON
AE	= 1
I	= 70

Depress START. As computer runs, load pattern into X. This is a quick check of the memory. If bits are lost or gained it will be apparent on the X register indicator. Try patterns of all "1's", all "0's", alternate "1's" and "0's" ($X_0 = 1$, $X_1 = 0$, etc.), and five "1's" with one "0", i.e., $X_0 = 0$, X through $X_5 = 1$, then $X_0 = 1$, $X_1 = 0$, X_2 through $X_5 = 1$, then $X_0 = 1$, $X_1 = 0$, X_2 through $X_5 = 1$, each time having a different bit = "0". If a failing combination is found, stop the computer and use the MNO instruction to determine the failing addresses.

(2) *Pattern Checks*

Punch a paper tape with the desired pattern. Read it in (EXI) and then print or punch it out. Compare the input with the output. For a comprehensive check of the M register decoding gates, load the numbers 00 - 77 into addresses 000 - 077, and load numbers 77 - 00 into addresses 100 - 177. These patterns can also be entered and read using MNI and MNO.

(3) *Intermittent Troubles*

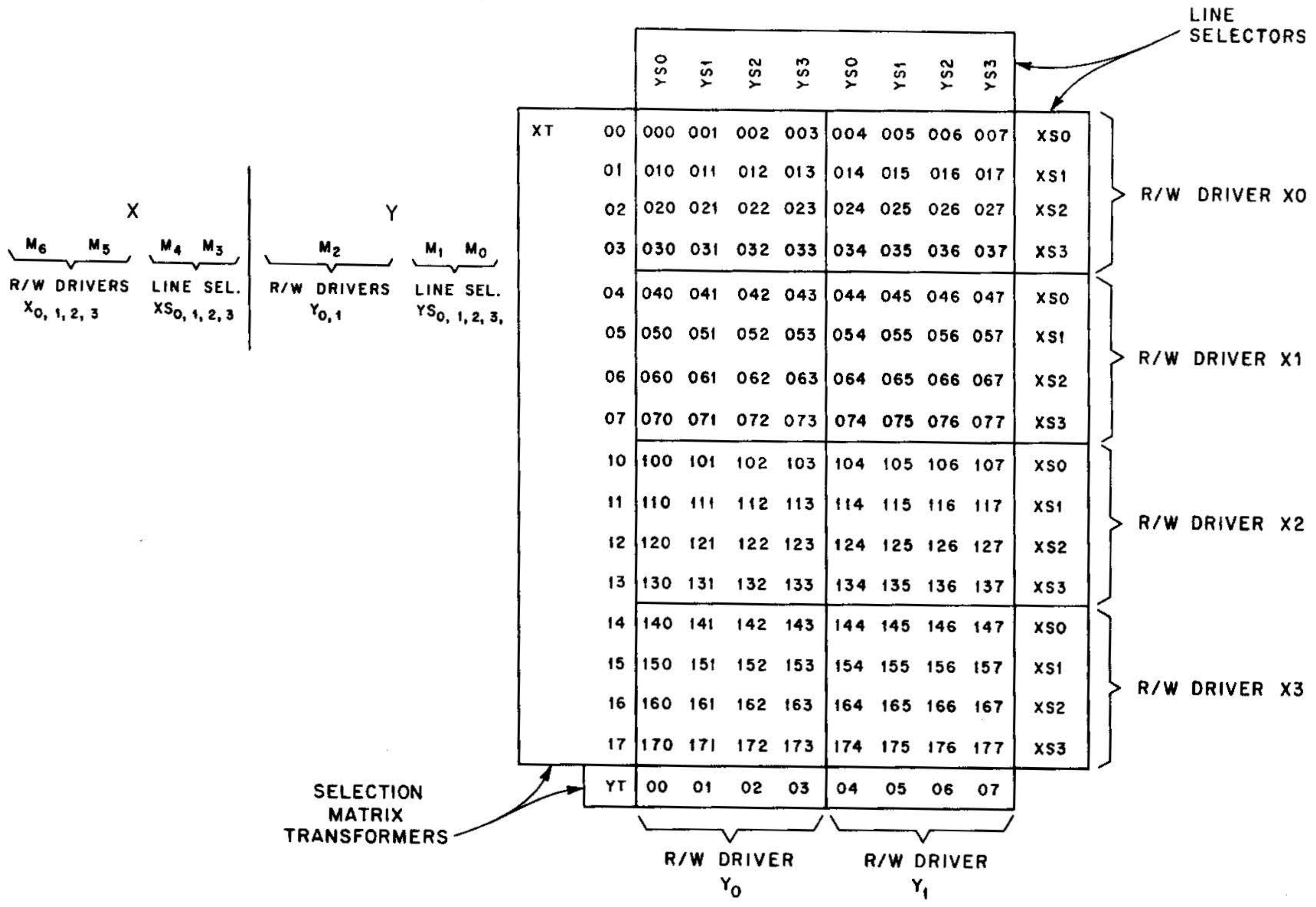
(a) Try to make the trouble occur constantly by varying the supply voltages.

(b) Closely examine the memory timing, including the relationship of R/W currents to inhibit current, and strobe to the core output, etc.

(c) Marginal operation of a R/W driver may only affect one bit, so if a given bit fails intermittently and the sense amplifier looks all right, determine if the trouble only occurs for certain addresses.

(4) *Waveshape Analysis*

The waveshapes shown in Section 4 may vary between computers. The best check is to compare the



XT		YS0	YS1	YS2	YS3	YS0	YS1	YS2	YS3	
	00	000	001	002	003	004	005	006	007	XS0
	01	010	011	012	013	014	015	016	017	XS1
	02	020	021	022	023	024	025	026	027	XS2
	03	030	031	032	033	034	035	036	037	XS3
	04	040	041	042	043	044	045	046	047	XS0
	05	050	051	052	053	054	055	056	057	XS1
	06	060	061	062	063	064	065	066	067	XS2
	07	070	071	072	073	074	075	076	077	XS3
	10	100	101	102	103	104	105	106	107	XS0
	11	110	111	112	113	114	115	116	117	XS1
	12	120	121	122	123	124	125	126	127	XS2
	13	130	131	132	133	134	135	136	137	XS3
	14	140	141	142	143	144	145	146	147	XS0
	15	150	151	152	153	154	155	156	157	XS1
	16	160	161	162	163	164	165	166	167	XS2
	17	170	171	172	173	174	175	176	177	XS3
	YT	00	01	02	03	04	05	06	07	

Figure 5-1. Memory Addressing Circuits Organization

output of the suspected circuit against circuits that are known to be functioning correctly. For instance, if R/W driver X_0 appears to be malfunctioning, compare its output to X_1, etc.

5-13. MAINTENANCE PROGRAMS

BI-TRAN SIX maintenance programs are in Section 6. More maintenance programs will be published periodically. Each program has a specific function, and care must be exercised not to misinterpret success or failure indications. For instance: The addition check does not completely exercise the adder, therefore a successful run does not mean that the adder is functioning correctly. It does indicate that the ADD instruction is being decoded. The adder check will completely exercise the adder, but it does not test all possible combinations of signs. The arithmetic control check will check all combinations of signs, but the assumption is made by this program that the adder works correctly and the ADD instruction is being decoded. Thus, to completely check the addition circuits, all three programs must be run.

Section VI

BI-TRAN SIX MAINTENANCE PROGRAMS

JUMP TEST (Even)

This program checks the even jump instructions. UNE and SBE are tested for their ability to jump; NAE and NZE are tested under no jump conditions. UNO is used to permit an optional programmed stop.

Operating procedure

1. Load the program: 20_8 words, first location-000.
2. Master clear.
3. Set JUMP STOP switch to ON if desired.
4. START.

Success Indications

The decision capability of NAE and NZE can be checked by loading "A" and starting with P = 006 for NZE or P = 010 for NAE. In these cases, stop 01 or 02 would be success indications.

```
a0:     UNE     a4                    00:    12    02
a2:     SBE     a6                    02:    10    03
a4:     UNO     a2                    04:    13    01
a6:     NZE     a14                   06:    16    06
a10:    NAE     a16                   10:    14    07
a12:    UNE     a0                    12:    12    00
a14:    STP     01                    14:    76    01
a16:    STP     02                    16:    76    02
```

JUMP TEST (Odd)

This program is similar to Jump Test (even) except that the odd jumps are tested. With the JUMP STOP switch ON, the computer should stop during every jump. Aside from this, the success indications and the operating procedure are the same as for the even test.

```
00:   13   02
02:   11   03
04:   13   01
06:   17   06
10:   15   07
12:   13   00
14:   76   03
16:   76   04
```

ADDITION TEST

This program checks the LDA, ADD, and CAS instructions. A successful run indicates that these instructions are decoded properly. However, the adder and arithmetic control circuits are not completely exercised, and malfunctions such as the failure of subcommands CAM, CA_0, SAV, and INCA will not be uncovered. The combinations added will not form carrys; therefore, the carry-generating circuits are not tested. The test is as follows: the number N is loaded into "A" (LDA), changed to -N (CAS); and then -N+N = 0 is formed (Add). If a sum other than zero is produced, an error stop occurs. The UNO permits an optional programmed stop.

Operating Procedure

1. Load the program: 16_8 words, first location - 000.
2. Master clear.
3. Set the JUMP STOP switch to ON if desired.
4. START.
5. Manually input N (any value), and re-start.

Success Indications

With the JUMP STOP switch ON, the program stops with P = 014. With the JUMP STOP switch off, the program loops.

Failure Indications

STOP 05 indicates -N+N $\neq$ 0.

```
a0:   MNI  a5             00:  20  05
a2:   LDA  a5  ←          02:  40  05
a4:   CAS  --             04:  74  00
a6:   ADD  a5             06:  44  05
a10:  NZE  a14            10:  16  06
a12:  UNO  a2             12:  13  01
a14:  STP  05  ←          14:  76  05
```

SUBTRACTION TEST

This program is similar to the Addition Test except the LDN, SUB, and CAS are used to form -N, then +N, and then N - (+N) = 0. The procedures and other comments for the Addition Test apply, except that the failure stop is STOP 06, which indicates N - (+N) ≠ 0.

```
a0:    MNI   a5                        00:   20   05
a2:    LDN   a5            -N          02:   42   05
a4:    CAS   --            +N          04:   74   00
a6:    SUB   a5            +N-N        06:   46   05
a10:   NZE   a14           (+N-N)≠0?   10:   16   06
a12:   UNO   a2                        12:   13   01
a14:   STP   06                        14:   76   06
```

SHIFT TEST

This program checks the SRE and SLE instructions. The operator enters a word to be shifted and the number of shifts desired. The program then loads "A" with the word, shifts right, then left, and compares the new contents of "A" with the original word. If "A", after shifting, is not the same as it was before shifting, a malfunction has occurred. The UNO permits an optional programmed stop.

Operating Procedures

1. Load the program: 25_8 words, first location - 000.
2. Master clear.
3. Set JUMP STOP switch to ON if desired.
4. START.
5. MNI - Word to be shifted, re-start.

*6. MNI - Number of right shifts; re-start.

*7. MNI - Number of left shifts; re-start.

*NOTE: Number of shifts entered must be equal (N_{SL} - N_{SR}) and must not shift the word out of AQ.

Success Indications

With the JUMP STOP switch ON, the program stops with P = 022. With the JUMP STOP switch off, the program loops continually.

Failure Indications

STOP 07 indicates data was lost or gained in shifting.

```
a0:   MNI   x0                  00:   20   24
a2:   MNI   a11                 02:   20   11
a4:   MNI   a13                 04:   20   13
a6:   LDA   x0                  06:   40   24
a10:  SRE   00                  10:   64   00
a12:  SLE   00                  12:   66   00
a14:  SUB   x0                  14:   46   24
a16:  NZE   a22                 16:   16   11
a20:  UNO   a6                  20:   13   03
a22:  STP   07                  22:   76   07
a24:  x0                        24:   00   --
```

MULTIPLY-DIVIDE TEST

This program tests the MPY and DIV instructions. A number (X) is manually loaded; then the equation $\left(\frac{X \cdot X}{X}\right) - X = 0$, is formed. A solution other than zero indicates a malfunction. The UNO permits an optional programmed stop.

Operating Procedure

1. Load the program: 23_8 words, first location - 000.
2. Master clear.
3. Set the JUMP STOP switch to ON if desired.
4. START.
5. MNI - Number must be ≠ 0; re-start.

Success indications

With the JUMP STOP switch ON, the program stops with P = 20.

With the JUMP STOP switch off, the program continually loops.

Failure Indications

STOP 10 (P = 22) indicates; $\frac{X \cdot X}{X} - X \neq 0$.

```
a0:   MNI   x0                         00:   20   22
a2:   LDA   x0                         02:   40   22
a4:   MPY   x0         X · X           04:   54   22
                      (X · X)
a6:   DIV   x0        -------          06:   56   22
                         X
a10:  SLO   05                         10:   67   05
a12:  SUB   x0         X - X           12:   46   22
a14:  NZE   a20       (X - X)≠0?       14:   16   10
a16:  UNO   a2                         16:   13   01
a20:  STP   10                         20:   76   10
a22:  x0    --                         22:   00   --
```

STORE TEST

This program tests the STA and STQ instructions. A number is manually loaded into memory location 26. The program tests the store instructions by transferring this number as follows:

$$(a_{26}) \longrightarrow \text{"A"} \longrightarrow \text{"Q"} \longrightarrow (a_{27}) \longrightarrow \text{"A"} \longrightarrow (a_{30}) \longrightarrow \text{"A"}$$

The contents of a_{26} and "A" are then compared. If they are not equal, the STOP is executed. The UNO permits an optional program stop.

Operating Procedure

1. Load the program: 30_8 words, first location - 000.
2. Master clear.
3. Set the JUMP STOP switch to ON, if desired.
4. START.
5. MNI - Number to be stored. Number must be positive and greater than zero. Re-start.

Success Indications

With the JUMP STOP switch ON, the program stops with P = 24.

With the JUMP STOP switch off, the program continually loops.

Failure Indications

STOP 10 (P = 26) indicates the number originally loaded was modified.

```
a0:    MNI    x0             00:   20   26
a2:    LDA    x0             02:   40   26
a4:    SRO    05             04:   65   05
a6:    STQ    x1             06:   62   27
a10    LDA    x1             10:   40   27
a12:   STA    x2             12:   60   30
a14:   LDA    x2             14:   40   30
a16:   SUB    x0             16:   46   26
a20:   NZE    a24            20:   16   12
a22:   UNO    a2             22:   13   01
a24:   STP    11             24:   76   11
a26:   x0     x1             26:   00   00
a30:   x2     --             30:   00   --
```

RAU & RSU TEST

This program tests the ability of RAU and RSU to modify the contents of a memory location. RAU is not tested for 7-bit (stored operand address) modification. A number (N) is manually loaded, then RAU and RSU (in turn) modify it. If (N + 1 - 1) ≠ N, the stop is executed. UNO permits an optional program stop.

Operating Procedure

1. Load the program: 26_8 words, first location - 000.
2. Set the JUMP STOP switch to ON if desired.
3. START
4. MNI - Number to be modified. Number must be positive and less than 37_8. Re-start.

Success Indications

With the JUMP STOP switch ON, the program stops with P = 22.

With the JUMP STOP switch off, the program continually loops.

Failure Indications

STOP 12 (P = 24) indicates (N + 1 - 1) ≠ N.

```
a0 :   MNI   x0    N→a25          00:  20  25
a2:    LDA   x0                   02:  40  25
a4:    STA   x1    N→a24          04:  60  24
a6:    RAU   x0    N + 1          06:  50  25
a10:   RSU   X0    (N+1)-1        10:  52  25
a12:   LDA   x0    N→A            12:  40  25
a14:   SUB   x1    N-(N+1-1)      14:  46  24
a16:   NZE   a22                  16:  16  11
a20:   UNO   a2                   20:  13  01
a22:   STP   12                   22:  76  12
a24:   x1    x0                   24:  00  00
```

RAU TEST

This program tests the ability of RAU to modify a 7-bit stored operand address. The following conditions are checked:

RAU of a location storing 37_8 forms 40 (-0 in the A register)

RAU of an odd location storing 77_8 forms 00 and sets the LSD in the previous (even) location.

The UNO permits an optional program stop.

Operating Procedure

1. Load the program: 50_8 words, first location - 000.
2. Set the JUMP STOP switch to ON if desired.
3. START.

Success Indications

With the JUMP STOP switch ON, the program stops with P = 32.

With the JUMP STOP switch off, the program continually loops.

Failure Indications

STOP 13 (P = 34) indicates that RAU of a location storing 37_8 doesn't form AM = 0.

STOP 14 (P = 36) indicates that RAU of a location storing 37_8 formed AM = 0, but did not set A_0.

STOP 15 (P = 40) indicates that RAU of a location storing 77_8 doesn't form 00.

STOP 16 (P = 42) indicates that RAU of an odd location storing 77_8 doesn't set the LSD in the previous (even) location.

RAU TEST

a_0:	LDA	x_0		00:	70	42
a_2:	LDA	x_2		02:	40	44
a_4:	STA	x_5		04:	60	47
a_6:	RAU	x_1	37 + 1	06:	50	43
a_{10}:	NZE	a_{32}	$(37 + 1) \neq 0$?	10:	16	15
a_{12}:	RAU	x_1	40 + 1	12:	50	43
a_{14}:	ADD	x_3	- 1 + 1	14:	44	45
a_{16}:	NZE	a_{34}	$(-1+1) \neq 0$?	16:	16	16
a_{20}:	RAU	x_5	77 + 1	20:	50	47
a_{22}:	NZE	a_{36}	$77 + 1 \neq 0$?	22:	16	17
a_{24}:	RSU	x_4	1 - 1	24:	52	46
a_{26}:	NZE	a_{40}	$(1-1) \neq$?	26:	16	20
a_{30}:	UNO	a_0		30:	13	00
a_{32}:	STP	13		32:	76	13
a_{34}:	STP	14		34:	76	14
a_{36}:	STP	15		36:	76	15
a_{40}:	STP	16		40:	76	16
a_{42}:	x_0	x_1		42:	37	00
a_{44}:	x_2	x_3		44:	77	01
a_{46}:	x_4	x_5		46:	00	00

x_0:	37
x_1:	(37, 40, or 41)
x_2:	77
x_3:	01
x_4:	00 or 01
x_5:	77

ADDER TEST

This program exercises the adder, but doesn't test the arithmetic algorithm control. Therefore, this program could run successfully even through a subcommand such as CAM was failing. The basic test is as follows: four numbers (a, b, c, and d) are generated and a = d, b = c. These numbers are added to form sums S_1 and S_2:

$$\begin{array}{c} a \\ b \\ \hline S_1 \end{array} \qquad \begin{array}{c} c \\ d \\ \hline S_2 \end{array}$$

If the addition was correct, $S_1 = S_2$. A given number is first used as the addend to form S_1, and then as the augend to form S_2. The initial values of a, b, c and d are manually loaded. Then, the program modifies these values.

Assume N = 0 is manually entered. The first test will be:

$$\begin{array}{lcl} a & = & 0 \\ b & = & 0 \\ \hline S_1 & = & 0 \end{array} \qquad \begin{array}{lcl} c & = & 0 \\ d & = & 0 \\ \hline S_2 & = & 0 \end{array} \qquad S_2 - S_1 = 0?$$

Then b and c will be increased and the test performed again:

$$\begin{array}{lcl} a & = & 0 \\ b & = & 1 \\ \hline S_1 & = & 1 \end{array} \qquad \begin{array}{lcl} c & = & 1 \\ d & = & 0 \\ \hline S_2 & = & 1 \end{array} \qquad S_2 - S_1 = 0?$$

This test sequence,increasing b and c - then adding and comparing sums, will be repeated until b and c are both = 37_8. Then a, b, c, and d will all be set to 1, and the test sequence (increase b and c, form S_1 and S_2, $S_2 - S_1 = 0?$). resumed:

$$\begin{array}{lcl} a & = & 1 \\ b & = & 1 \\ \hline S_1 & = & 2 \end{array} \quad \begin{array}{lcl} c & = & 1 \\ d & = & 1 \\ \hline S_2 & = & 2 \end{array} \qquad S_2 - S_1 = 0? \qquad \begin{array}{lcl} a & = & 1 \\ b & = & 2 \\ \hline S_1 & = & 3 \end{array} \quad \begin{array}{lcl} c & = & 2 \\ d & = & 1 \\ \hline S_2 & = & 3 \end{array} \qquad \text{etc.}$$

As the values of a and d are increased, the maximum values of b and c are decreased so overflow will not occur. The final test will be:

$$\begin{array}{lcl} a & = & 17 \\ b & = & 20 \\ \hline S_1 & = & 37_8 \end{array} \qquad \begin{array}{lcl} c & = & 20 \\ d & = & 17 \\ \hline S_2 & = & 37_8 \end{array} \qquad S_2 - S_1 = 0?$$

Operating Procedure

1. Load the program: 123 words, first location - 000.

2. Set the JUMP STOP switch to ON if desired.

3. START.

4. MNI - N = 17 or less. Re-start. If N = 0, all meaningful combinations of a, b, c, and d will be generated. If a test of all numbers is not desired, set N to the lowest value to be tested. Thus, MNI of N = 6 would test the following combinations:

$$\begin{array}{l} a = 6 \\ \underline{b = 6} \\ S_1 = 14_8 \end{array} \quad \begin{array}{l} c = 6 \\ \underline{d = 6} \\ S_2 = 14_8 \end{array} \quad \text{through} \quad \begin{array}{l} a = 17 \\ \underline{b = 20} \\ S_1 = 37_8 \end{array} \quad \begin{array}{l} c = 20 \\ \underline{d = 17} \\ S_2 = 37_8 \end{array}$$

Success Indications

1. With the JUMP STOP switch ON, the program runs approximately 6 seconds if N = 0.

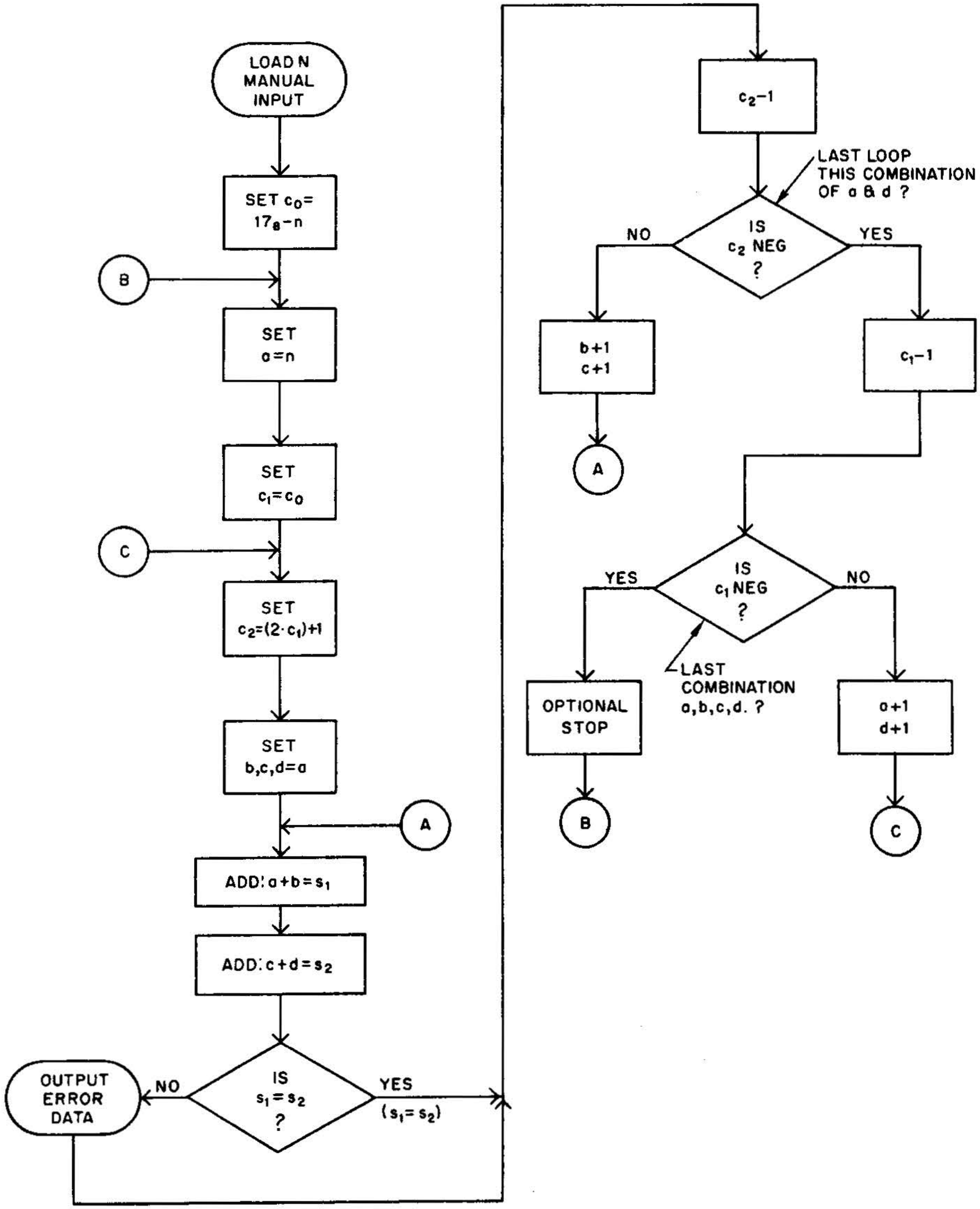
2. With the JUMP STOP switch off, the program continually loops.

Failure Indications

1. The program stops with P = 106 (I = 23-MNO). This indicates $S_2 - S_1 \neq 0$. Depressing the start switch five times will output a, b, c, d, S_1 and S_2 in that order. Depressing start again will re-start the program with the next sequential combination of a, b, c, d.

2. The program runs for other than 6 seconds. This indicates the adder is failing and, in turn, causing RSU to fail.

ADDER TEST



LOAD N MANUAL INPUT
SET c_0= 17_8−n
B
SET a=n
SET c_1=c_0
C
SET c_2=(2·c_1)+1
SET b,c,d=a
A
ADD: a+b=s_1
ADD: c+d=s_2
IS s_1=s_2 ?
NO
OUTPUT ERROR DATA
YES
(s_1=s_2)
c_2−1
LAST LOOP THIS COMBINATION OF a & d ?
IS c_2 NEG ?
NO
YES
b+1 c+1
A
c_1−1
IS c_1 NEG ?
YES
NO
LAST COMBINATION a,b,c,d. ?
OPTIONAL STOP
B
a+1 d+1
C

ADDER CHECK PROGRAM - SYMBOLIC

```
a_0:    MNI   m_3     n                                     m:      15_10
a_2:    LDA   m                                             m_0:    c_0
a_4:    SUB   m_3     15_10 - n                             m_1:    c_1
a_6:    NAE   a_0     error (15-n < 0)                      m_2:    c_2
a_10:   STA   m_0     c_0 = 15_10 - n                       m_3:    n
a_12:   LDC   m_3     a = n                                 m_4:    a
a_14:   LDC   m_0     c_1                                   m_5:    b
a_16:   LDA   m_1                                           m_6:    c
a_20:   SLE   01                                            m_7:    d
a_22:   STA   m_2     2c_1                                  m_10:   S_1 = a + b
a_24:   RAU   m_2     c_2 = 2c_1 + 1                        m_11:   S_2 = c + d
a_26:   LDC   m_4
a_30:   LDC   m_5
a_32:   LDC   m_6
a_34:   LDA   m_4
a_36:   ADD   m_5                         c_1 = 0
a_40:   STA   m_10
a_42:   LDA   m_6
a_44:   ADD   m_7                         c_2 >= 0
a_46:   STA   m_11
a_50:   SUB   m_10                        b_0:   LCT   05
a_52:   NZE   b_0     ERROR               b_2:   MNO   m_4
a_54:   RSU   m_2                         b_4:   UNE   a_54
a_56:   NAE   a_66
a_60:   RAU   m_5
a_62:   RAU   m_6     c_2 < 0
a_64:   UNE   a_34
a_66:   RSU   m_1
a_70:   NAO   a_100
a_72:   RAU   m_4
a_74:   RAU   m_7
a_76:   UNE   a_16
a_100:  UNO   a_12    c_1 < 0
```

ADDER CHECK PROGRAM - OCTAL

00:	21	14		
02:	41	10		15_{10}
04:	47	14	$c_0 < 0$	$c_0 = 15_{10} - n$
06:	14	00		
10:	61	11		c_0
12:	71	14		$n \rightarrow a$
14:	71	11		$c_0 \rightarrow c_1$
16:	41	12		c_1
20:	66	01		$2c_1$
22:	61	13		$2c_1 \rightarrow c_2$
24:	51	13		$2c_1 + 1 \rightarrow c_2$
26:	71	15		$a \rightarrow b$
30:	71	16		$b \rightarrow c$
32:	71	17		$c \rightarrow d$
34:	41	15		a
36:	45	16		$a + b$
40:	61	21		$S_1 + a + b$
42:	41	17		c
44:	45	20	$(c_1 < 0)$ $(c_1 \geq 0)$	$c + d$
46:	61	22		$S_2 = d + c$
50:	47	21	$(c_2 \geq 0)$	$S_2 - S_1$
52:	16	41		
54:	53	13		$c_2 - 1 \rightarrow c_2$
56:	14	33		
60:	51	16		$b + 1 \rightarrow b$
62:	51	17	$c_2 < 0$	$c + 1 \rightarrow c$
64:	12	16		
66:	53	12		$c_2 < 0$ $c_1 - 1 \rightarrow c_1$
70:	14	40		
72:	51	15		$a + 1 \rightarrow a$
74:	51	20	$c_1 < 0$	$d + 1 \rightarrow d$
76:	12	07		
100:	13	05		$c_1 < 0$
102:	72	05		$S_2 - S_1 \neq 0$
104:	23	15		Error Control
106:	12	26		

110:	17
111:	c_0
112:	c_1
113:	c_2
114:	n
115:	a
116:	b
117:	c
120:	d
121:	$S_1 = a + b$
122:	$S_2 = c + d$

UPPER MEMORY TEST

This program tests all cells in locations 100_8 through 177_8. All cells are sequentially tested with a given data word. The program then modifies the data word by adding one (1) to it and again tests all cells. This routine of modifying the data word and sequentially testing all cells is repeated until the data word 77_8 has been used. At this time there is a UNO that allows the program to start over with the first data word or allows it to be stopped.

Operating Procedure

1. Load the program: 100_8 words, first location - 000.
2. Set the JUMP STOP switch to ON if desired and set ERROR STOP to AV BYPASS.
3. START. If the JUMP STOP switch is ON, the program will stop before running as the first instruction is UNO.

Success Indications

1. With the JUMP STOP switch ON, the program runs for approximately 67 seconds and stops with P = 02.
2. With the JUMP STOP switch OFF, the program continually loops.

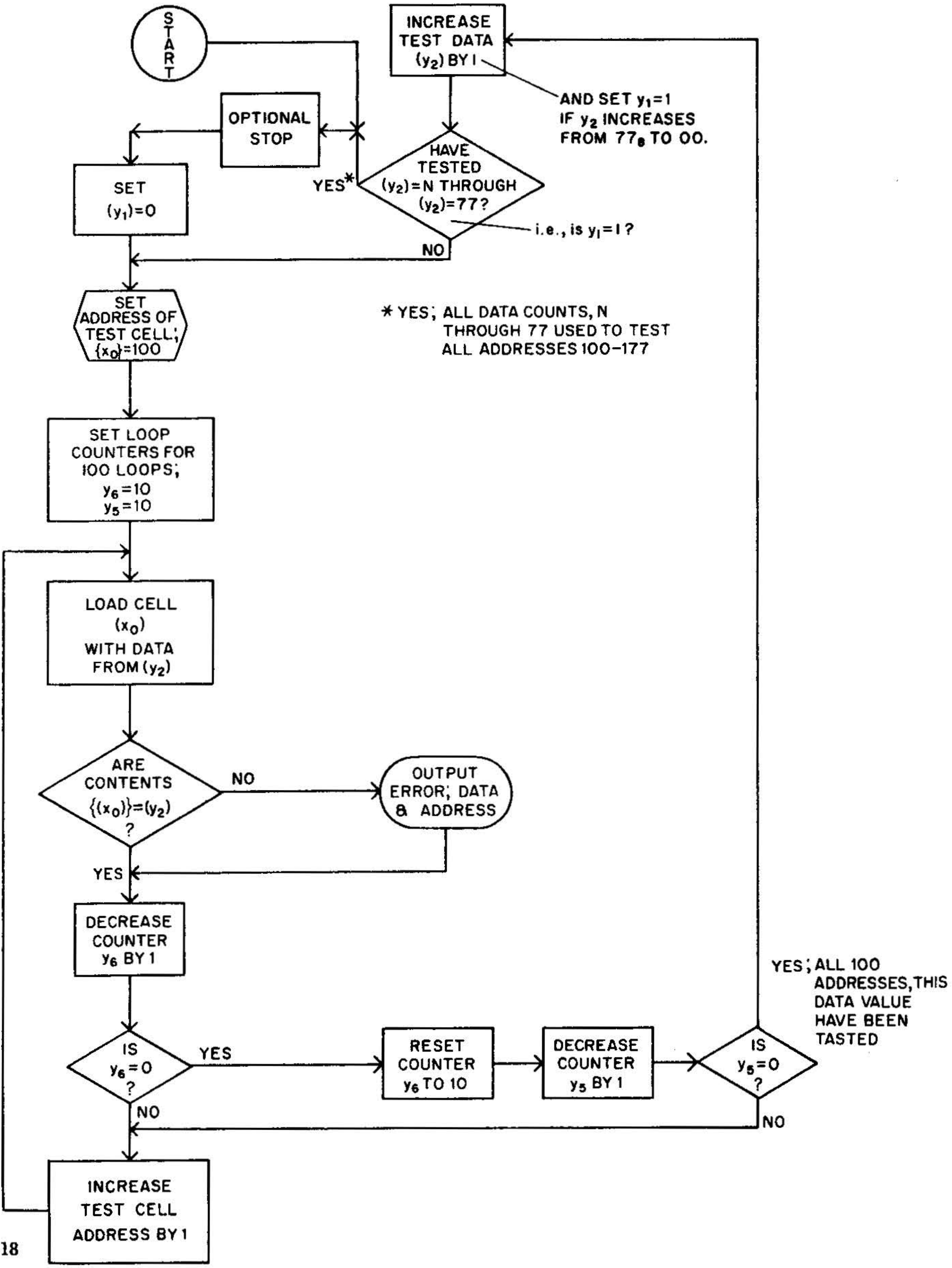
Failure Indications

1. If the program doesn't take approximately 67 seconds to run, there is a malfunction affecting the lower memory, arithmetic, or control section.
2. MNO (P = 66), (M = 25)

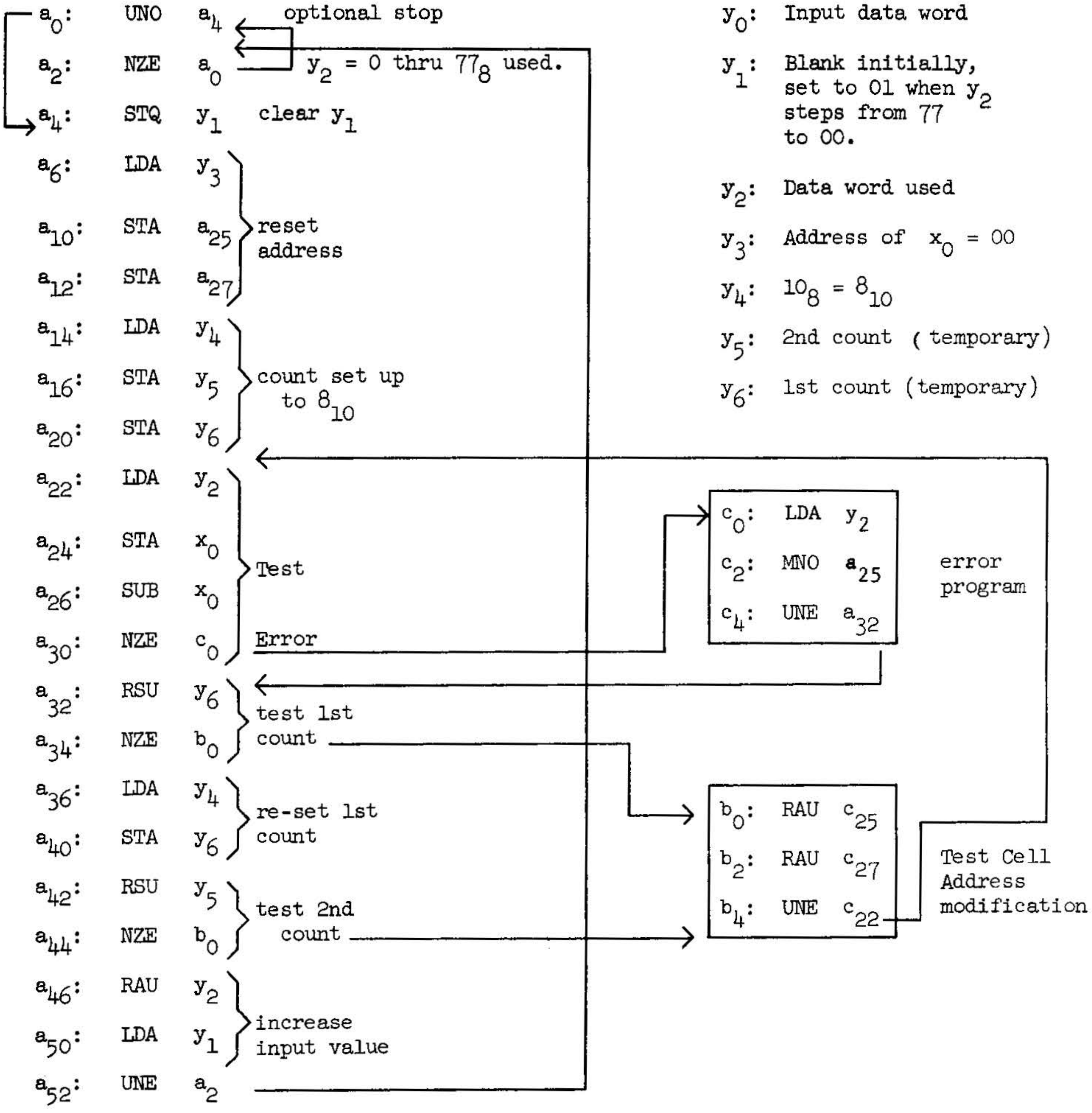
   The word in X is the address (in upper memory) of the failing cell. The word in the ACCUMULATOR is the failing combination. For example: an ERROR STOP with X = 41 and A = 37 indicates the number 37_8 was not stored correctly in cell 141_8.

   START. The test will continue.

UPPER MEMORY TEST


START
INCREASE TEST DATA (y2) BY 1
AND SET y1=1 IF y2 INCREASES FROM 77₈ TO 00.
OPTIONAL STOP
HAVE TESTED (y2)=N THROUGH (y2)=77?
YES*
NO
i.e., is y1=1?
SET (y1)=0
SET ADDRESS OF TEST CELL; {x0}=100
* YES; ALL DATA COUNTS, N THROUGH 77 USED TO TEST ALL ADDRESSES 100-177
SET LOOP COUNTERS FOR 100 LOOPS; y6=10 y5=10
LOAD CELL (x0) WITH DATA FROM (y2)
ARE CONTENTS {(x0)}=(y2) ?
NO
OUTPUT ERROR; DATA & ADDRESS
YES
DECREASE COUNTER y6 BY 1
IS y6=0 ?
YES
RESET COUNTER y6 TO 10
DECREASE COUNTER y5 BY 1
IS y5=0 ?
YES; ALL 100 ADDRESSES, THIS DATA VALUE HAVE BEEN TASTED
NO
NO
INCREASE TEST CELL ADDRESS BY 1

UPPER MEMORY TEST PROGRAM - SYMBOLIC



```
a0:   UNO  a4    optional stop
a2:   NZE  a0    y2 = 0 thru 77(8) used.
a4:   STQ  y1    clear y1
a6:   LDA  y3  }
a10:  STA  a25 } reset
a12:  STA  a27 } address
a14:  LDA  y4  }
a16:  STA  y5  } count set up
a20:  STA  y6  }   to 8(10)
a22:  LDA  y2  }
a24:  STA  x0  }
a26:  SUB  x0  } Test
a30:  NZE  c0  } Error
a32:  RSU  y6  } test 1st
a34:  NZE  b0  } count
a36:  LDA  y4  } re-set 1st
a40:  STA  y6  } count
a42:  RSU  y5  } test 2nd
a44:  NZE  b0  }   count
a46:  RAU  y2  }
a50:  LDA  y1  } increase
a52:  UNE  a2  } input value
```

y_0: Input data word

y_1: Blank initially, set to 01 when y_2 steps from 77 to 00.

y_2: Data word used

y_3: Address of x_0 = 00

y_4: $10_8 = 8_{10}$

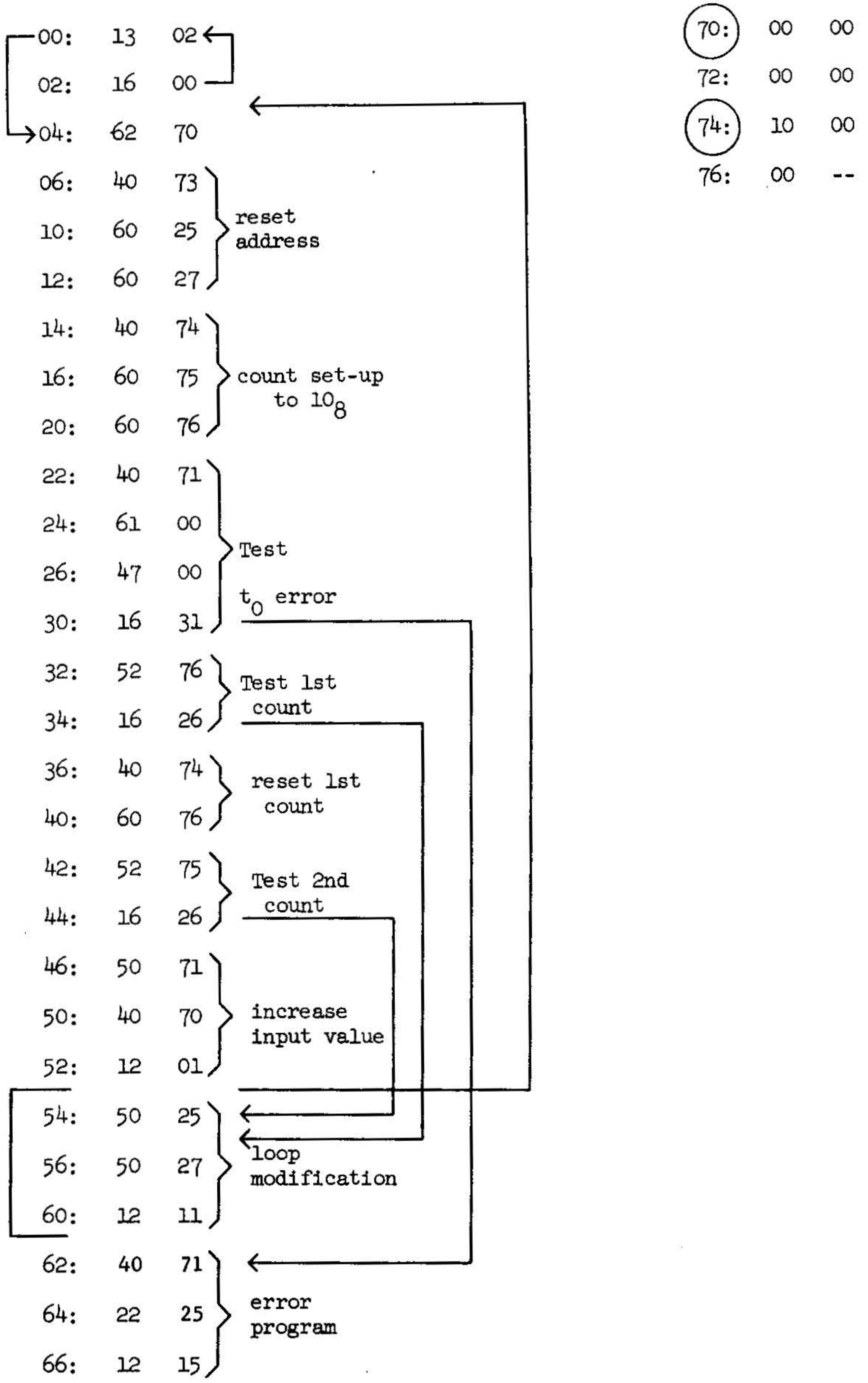
y_5: 2nd count (temporary)

y_6: 1st count (temporary)

```
c0:  LDA  y2
c2:  MNO  a25     error
c4:  UNE  a32     program
```

```
b0:  RAU  c25
b2:  RAU  c27     Test Cell
b4:  UNE  c22     Address
                  modification
```

UPPER MEMORY TEST PROGRAM - OCTAL



```
00:  13  02                              70:  00  00
02:  16  00                              72:  00  00
04:  62  70                              74:  10  00
06:  40  73  \                           76:  00  --
10:  60  25   > reset
12:  60  27  /  address
14:  40  74  \
16:  60  75   > count set-up
20:  60  76  /     to 10_8
22:  40  71  \
24:  61  00   |
26:  47  00   > Test
30:  16  31  /  t_0 error
32:  52  76  \ Test 1st
34:  16  26  /  count
36:  40  74  \ reset 1st
40:  60  76  /  count
42:  52  75  \ Test 2nd
44:  16  26  /  count
46:  50  71  \
50:  40  70   > increase
52:  12  01  /  input value
54:  50  25  \
56:  50  27   > loop
60:  12  11  /  modification
62:  40  71  \
64:  22  25   > error
66:  12  15  /  program
```

Section VII

TECHNICAL GLOSSARY OF ABBREVIATIONS FOR THE BI-TRAN SIX

Symbols Used:

$\vert \ \vert$	The magnitude of. Thus $\vert A\vert$ is the magnitude of the contents of the accumulator.
—	Dashes following letter indicate any register, flip-flop, etc. Thus, T— means Transfer from one register to another, and generalizes specific statements like TAX or TMX.
$-_k$	Subscripts identify specific flip-flops within a register, or specific pulses within a cycle, etc. Thus M_2 identifies a specific flip-flop of the memory address register.
——	A bar (vinculum) above letters abbreviating a function complements the function and, hence, inverts the signal.
A	Accumulator. A 6-bit register with associated additive properties.
$\vert A\vert \neq 0$	The contents of AM is not equal to zero.
$\vert A\vert = 37$	The contents of AM is equal to 37_8.
AC	Adder control register. A 2-bit increasing counter.
ADG	Addition group command defined by the function (LDA + LDN + ADD + SUB).
AE	Acquisition/execution. A 1-bit control flip-flop register.
AEM	Acquisition/execution mode.
AEMR	Acquisition/execution mode and mode repeat.
AM	"A" magnitude. A 5-bit register consisting of the 5 bits of magnitude of A.
AQ	AQ register. An 11-bit register consisting of A and QM.
AQL	Subcommand causing a shift of the AQM register left by 1.
AQM	AQM register. A 10-bit register consisting of AM and QM.
AQR	Subcommand causing a shift of the AQM register right by 1.
ARG	Arithmetic group command defined by the Boolean function (LDA + LDN + ADD + SUB + RAU + RSU + MPY + DIV).
AV	Add overflow control flip-flop.
AVBP	Add overflow bypass.
BIS	Bi-octal start.
BP	Bypass.
C	Count down register. A 7-bit decreasing counter.

Symbols Used:

C—	Complement the register or flip-flop indicated.
CL	Clear all flip-flops in the system. The master clear.
CL—	A logic clear of the register or flip-flop indicated.
$C = 0$	The count down register has a count of 0.
CP	Clock pulse for equipment timing.
CP_1	Delayed clock pulse.
D	Distributor register. A 4-bit increasing counter.
DEC	Subcommand to decrease the count of the C register.
DM	Distributor mode.
DMR	Distributor mode and mode repeat.
DP_k	One of the sixteen distributor pulses.
DV	Divide overflow flip-flop.
E_k	One of the six external input lines.
EIR	External input resume.
EL	External control flip-flop.
EOR	External output resume.
EXCL	Level that indicates if the sign bits of A_0 and X_0 are different (exclusive OR).
I	Instruction register. A 6-bit register.
IE	Instruction error flip-flop.
IEDP	Instruction error distributor pulse.
IM	Instruction mode.
IMR	Instruction mode and mode repeat.
INA	Initiate adder control subcommand.
INC·	Subcommands that increase the count of the indicated register.
INHIBIT	A memory control flip-flop.
INL	Initiate subcommand for loading the X register from external equipment.
INS	Initiate start.
INSS	External input resume single-shot.
INU	Initiate subcommand for unloading the X register to external equipment.
ISX	Initiate subcommand for unloading storage into the X register.
IXS	Initiate subcommand for loading storage from X.
J1	External input jack.
J2	External output jack.
J3	Bi-octal jack.
J4	Front panel jack.
J5	Remote panel jack.
J6 - J21	Jacks located on boards 1 through 8, two per board in order of count.

Symbols Used:

J22	DC power jack.
J23	AC control jack.
J24	AC input power jack.
JMP	Boolean function defined by (SB + UN + SJ + ZJ).
JSE	Jump stop enable.
LAN	Command defined by (LDA + LDN).
LNS	Command defined by (LDN + SUB).
LSTP	The logic stop control function.
LZ	Logic zero flip-flop. A 1-bit register.
M	Memory address register. A 7-bit increasing counter.
MCL—	Manual clear of the indicated register or flip-flop.
MIN	Manual input enable.
MOUT	Manual output enable.
MS—	Manual set of the indicated register or flip-flop.
MVB	The generator of clock pulses for the system's basic timing.
NO BP	No bypass.
OTSS	External output resume single-shot.
OV	Overflow flip-flop. A 1-bit register.
P	Program address register. A 6-bit increasing counter.
PSTP	Logic stop function for peripheral equipment.
Q	Quotient register. A 6-bit register.
QM	QM register. A 5-bit register consisting of the 5 bits of magnitude of Q.
RCS	Remote console start.
READ	A memory control flip-flop.
RPT	Mode repeat.
RUN 1	Run 1 control flip-flop.
RUN 2	Run 2 control flip-flop.
R/W	Read/write control flip-flop.
S—	Logic set of the indicated register or flip-flop.
EXCEPTIONS:	
SC5	Subcommand which sets the count down register to an octal count of 5.
SD7, SD13, SD16	Subcommands which set the distributor pulse register to the octal count indicated.
SS	Single-shot.
S_k	An output of a sense amplifier.
SI	Sign flip-flop. A 1-bit register.

Symbols Used:

STROBE A memory control flip-flop.

STSS Start single-shot.

T_k The adder control register timing pulse outputs.

T— Subcommands to transfer data from the first indicated register to the second. (For example, TAX causes data to be transferred from the accumulator to the Exchange register.) Transfer between 6-bit registers occur in parallel: the rightmost bit is transferred to the right most bit and the leftmost to the leftmost.

TRANSFERS BETWEEN 6-BIT AND 7-BIT REGISTERS ARE MADE IN THE FOLLOWING MANNER:

TMX Transfers the most significant 6 bits of the memory address registers to the X register. M_0 is not transferred.

TPM Transfers the 6 bits of the program address register to the most significant 6 bits of the memory address register. (Note that the P_0 switch-light indicator has no associated flip-flop.)

TXC Transfers the X register to the least significant bits of the C register. I_0 is transferred to C_6.

TXM Transfers the X register to the least significant bits of the M register. I_0 is transferred to M_6.

TIC Transfer of I_0 to C_6.

TIM Transfer of I_0 to M_6.

WRITE A memory control flip-flop.

X Exchange register. A 6-bit register.

XM A 5-bit register consisting of the 5 bits of magnitude of X.

Section IX

PARTS LIST

Symbol Number	Description		Fabri-Tek Part Number
Q1, Q2, Q3, Q4, Q6	Transistor	2N1305	021-0001-00
Q1-1, Q2-1	Transistor	2N1309	021-0018-00
Q5	Transistor	2N1304	021-0003-00
CR	Diode	1N270	022-0002-00
R1, R4, R5, R8	Resistor	8.2K, ¼ Watt, 5% Fixed	108-0822-00
R2, R6	Resistor	39K, ¼ Watt, 5% Fixed	108-0393-00
R3, R7	Resistor	5.6K, ¼ Watt, 5% Fixed	108-0562-00
R19, R20	Resistor	15K, ¼ Watt, 5% Fixed	108-0153-00
R9, R10, R11, R12, R13, R14	Resistor	22K, ¼ Watt, 5% Fixed	108-0223-00
R15, R16, R17	Resistor	6.8K, ¼ Watt, 5% Fixed	108-0682-00
R18, R21	Resistor	1K, ¼ Watt, 5% Fixed	101-0102-00
R27, R35	Resistor	100 ohm, ¼ Watt, 5% Fixed	108-0101-00
R23, R1-1, R5-1	Resistor	4.7K, ¼ Watt, 5% Fixed	108-0472-00
R24, R25, R1-3, R5-3	Resistor	2.7K, ¼ Watt, 5% Fixed	108-0272-00
R1-2, R5-2	Resistor	3.3K, ¼ Watt, 5% Fixed	108-0332-00
R22	Resistor	560 ohm, ¼ Watt, 5% Fixed	108-0561-00
R26	Resistor	18K, ¼ Watt, 5% Fixed	108-0183-00
R30, R31	Resistor	220 ohm, ¼ Watt, 5% Fixed	108-0221-00
R32	Resistor	820 ohm, ¼ Watt, 5% Fixed	108-0821-00
R28	Resistor	47 ohm, 1 Watt, 5% Fixed	103-0470-00
R34	Resistor	62 ohm, 1 Watt, 5% Fixed	103-0620-00

Symbol Number	Description		Fabri-Tek Part Number
R33	Resistor	330 ohm, ¼ Watt, 5% Fixed	108-0331-00
C3, C4, C5, C6, C7, C8	Capacitor	470 pf. disc Type CE Ceramic, 500 V 20%	023-0244-00
C9	Capacitor	.0015 uf. disc CK 60 Ceramic, 500V, 20%	023-0133-00
C3 – Sense Amp Only	Capacitor	680 pf. disc Type CE Ceramic, 500 V, 20%	023-0282-00
C13	Capacitor	.0018 uf. disc DD-182 Ceramic, 1000 V, 20%	023-0280-00
C10, C11	Capacitor	.001 uf. disc CK 60 Ceramic, 500 V,20%	023-0132-00
C1, C2	Capacitor	220 pf. disc CK 60 Ceramic, 600 V, 20%	023-0116-00
C14	Capacitor	100 uf. Mallory TT 25 X 100 Electrolytic 25V	023-0281-00
C12	Capacitor	.0056 uf. ±10% 200 V.	023-0301-00
C15	Capacitor	15 uf. 20 V. DC Tantulum	023-0105-00
L2	Choke	120 microhenry	019-0036-00
L1	Choke	Delay Line Choke	019-0091-00
T1-2, T2-2	Transformer	R/W Driver	019-0092-00
T1-3	Transformer	Selection	019-0093-00
T1-4	Transformer	Sense Amp	019-0511-00
T1-5	Transformer	Inhibit Driver	019-0095-00
SW-1–SW-15	Switch	PB TEC MBS-22-0	025-0092-00
SW 21, SW 22	Switch	Rotary 2-pole, 4-position Centralab PA 2003	025-0018-00
SW 23	Switch	Rotary 4-pole, 4-position Centralab PA 2010	025-0097-00
SW 16, SW 18**	Switch	Lever, Non-locking, 3-position, 1-C, 1-C Contacts, Switchcraft LT 41306	025-0142-00
SW 17*	Switch	Lever, Nonlocking, C Contacts Switchcraft 28203 (modified)	025-88-1
SW 17**	Switch	Lever, Non-locking, 2-position, 1-C Contacts, Switchcraft LT 41203	025-0140-00
SW 19, SW 20*	Switch	Lever, Locking C Contacts Switchcraft 28203L (modified)	025-87-1
SW 19, SW 20**	Switch	Lever, Locking, 2-position, 1-C Contacts, Switchcraft LT 41203L	025-041-00
SW 16, SW 18*	Switch	Lever, Locking 2 C Contacts 3-position, Switchcraft 28306 (modified)	025-99-1

Symbol Number	Description		Fabri-Tek Part Number
All SL except SL 6, SL 12 SL 64, SL 65,. SL 60, SL 61, SL 62, SL 63, SL 56, SL 57, SL 66	Switch Light	White TEC-S-1778A	025-0100-01
SL 6, SL 12, SL 64, SL 65, SL 56, SL 66	Switch Light	Green TEC-S-1778A	025-0100-02
SL 60, SL 61, SL 62, SL 63, SL 57	Switch Light	Red TEC-S-1778A	025-0100-03
	Lamp Driver		190-0685-00
	Lamp Driver, C Register		190-0686-00
**	Lamp Driver, Modified – (Reg. Clear Pull-down resistor)		190-0687-00
J6–J21	Connector	(Card) Viking VH 28/1an5 or Cannon PBAD 56-2AE2G	024-0419-00
J1, J2, J3	Connector	Female (Cable) Amphenol MinRac 17-10150	024-0466-00
P22	Connector	Male CJ P-304-CCT-L	024-0468-00
P23	Connector	Male CJ P-303-CCT-L	024-0467-00
	Socket	Transistor, Augat 8058-1G23	024-0458-00
	Jumper	Cinch Jones 141-J	043-0154-00
Logic Board 1	Circuit Assy.	Instruction Register	999-5029-15
Logic Board 2	Circuit Assy.	Pulse Distributor & P Register	999-5029-16
Logic Board 3	Circuit Assy.	Logic Control	999-5029-17
Logic Board 4	Circuit Assy.	Arithmetic Control	999-5029-18
Logic Board 5	Circuit Assy.	X, C, and Q Register	999-5029-19
Logic Board 6	Circuit Assy.	A Register & Adder	999-5029-20
Logic Board 7	Circuit Assy.	Main Control & M Register	999-5029-21
Logic Board 8	Circuit Assy.	Core Memory	999-5029-23
		POWER SUPPLY	999-5014-08
Q1, Q2, Q3	Transistor	2N277	021-0011-00
CR1 thru CR5	Diode	1N1200	022-0112-00
R1, R2, R5, R6	Resistor	.1 ohm, 5 Watt, 5%	106-0042-00

Symbol Number	Description		Fabri-Tek Part Number
R3, R4	Resistor	220 ohm, 2 Watt, 5%	105-0221-00
C1	Capacitor	7000 uf. 25 V	023-0247-00
C2	Capacitor	17000 uf. 25 V	023-0188-00
C3	Capacitor	220 uf. 15 V, 20%	023-0354-00
T1	Transformer	Power	019-0097-00
KO1	Relay	KH P17A11 (115 VAC)	029-0040-00
	Fuse Holder		070-0019-00
J24	AC Power Jack	Amphenol 160-5	024-0092-00
J22	Connector	Female, 4 Pin, Cinch Jones S-304-CCT-K	024-0470-00
J23	Connector	Female, 3 Pin, Cinch Jones S-303-CCT-K	024-0469-00
		POWER SUPPLY REGULATOR ASSEMBLY	190-0014-00
Q1, Q4	Transistor	2N1184	021-0059-00
Q2, Q3	Transistor	2N1304	021-0003-00
Q5, Q6	Transistor	2N1305	021-0001-00
CR2	Diode	1N746A	022-0009-00
CR1, CR3	Diode	1N754A	022-0011-00
R6, R8, R16	Resistor	150 ohm, ½ Watt, 5%	101-0151-00
R7, R15, R10, R16	Resistor	680 ohm, ½ Watt, 5%	101-0681-00
R5, R14	Resistor	270 ohm, ½ Watt, 5%	101-0271-00
R3, R4	Resistor	1K ohm, ½ Watt, 5%	101-0102-00
R2, R12	Resistor	560 ohm, ½ Watt,	101-0561-00
R1	Resistor	47 ohm, ½ Watt, 5%	101-0470-00
R13	Resistor	470 ohm, ½ Watt, 5%	101-0471-00
R11	Resistor	330 ohm, ½ Watt, 5%	101-0331-00
R9, R17	Resistor	500 ohm Bourns Trimpot 200 P-1-5001	026-0047-00
C2, C3	Capacitor	.01 uf 50 V +80% -20%	023-0074-00
C1	Capacitor	1 uf. 35 V 10%	023-0105-00

Symbol Number	Description		Fabri-Tek Part Number
	MISCELLANEOUS		
	Power Cord	Belden No. 17460-S	085–0004–00
F3 (-10 Volt)	Fuse	5A 250 V STD 3AG	070-0038-00
F2 (+10 Volt)	Fuse	2A 250 V STD 3AG	070-0026-00
Fl (AC)	Fuse	1.5A 250 V SLO BLD	070-0028-00
	Removal Tool, Bulb Microswitch MS-15PA7 (not supplied)		
	Bulb	397	052-0084-00
	Knob	Raytheon DS 90-3-2	045-0012-00

* All Units Up to SN66464
** Unit SN66464 and after

DIGIAC®
CORPORATION

TEACHWARE ™ by

DIGIAC
CORPORATION