

## Technical Training

ELECTRONIC COMPUTER AND SWITCHING SYSTEMS SPECIALIST

COMPUTER UNITS AND COM-TRAN 10

JUNE 1981



**USAF TECHNICAL TRAINING SCHOOL**  
**3390th Technical Training Group**  
**Keesler Air Force Base, Mississippi**

---

Designed For ATC Course Use

# C O N T E N T S

TITLE	PAGE
Chapter I - Computer Units and Programming.....	1
Computer Units and Data Flow.....	1
Data Flow.....	2
Data Flow Summary.....	3
Review Questions 1-1.....	3
Memory Units.....	4
Mode of Access.....	5
Access Time.....	5
Capacity.....	6
Permanence.....	6
Volatility.....	6
Storage Devices.....	7
Magnetic Storage Systems.....	7
Hysteresis Loop.....	7
Ferrite Core Memory.....	9
Magnetic Drum Memory.....	22
Magnetic Tape Memory.....	25
Review Questions 1-2.....	28
Terminal Equipment.....	28
Card Reader.....	29
Card Punch.....	30
Line Printer.....	30
Tape Drive Unit.....	32
Flexowriter.....	34
Display Equipment.....	34
Review Questions 1-3.....	40
Computer Operation and Familiarization.....	41
Individualism of the COM-TRAN TEN Computer System.....	43
Operating the Computer.....	45
Controls and Switches.....	45
Computer Registers and Display Panel.....	49
Hexadecimal Review.....	52
Manual Input Procedure.....	58
Manual Output Procedure.....	60
Review Questions 1-4.....	62
Programming.....	64
Type of Instruction.....	65
Instruction Repertoire.....	65
Instruction Format.....	72
Addressing Upper Memory.....	72
Indexing.....	72
Number Representation.....	73
Instructions.....	73
Review Questions 1-5.....	80
Review Questions 1-6.....	85
Review Questions 1-7.....	89
Review Questions 1-8.....	94
Review Questions 1-9.....	101
Procedures for Writing Programs.....	104
Programming Problem.....	107
Alphabetical Summary of Instructions.....	109
Numerical Summary of Instructions.....	110
Computer Terms Glossary.....	112

Supersedes KDA-3032, June 1977

	PAGE
Chapter II - Computer Units Logic Analysis.....	125
Special Components.....	125
Phantom OR-Gate.....	125
NOR-Gate Latch.....	126
D-Type Flip-Flop.....	128
Single Shot (SN74121).....	129
Positive AND-Driver.....	130
Special Component Summary.....	132
Review Questions 2-1.....	134
Computer Units.....	137
Key to Logic Diagrams.....	137
Review Questions 2-2.....	138
Clock.....	138
Review Questions 2-3.....	144
D-Register.....	144
Review Questions 2-4.....	148
Input Register.....	148
Review Questions 2-5.....	150
B-Register.....	150
Review Questions 2-6.....	152
Memory Module.....	153
Review Questions 2-7.....	154
M-Register.....	155
Review Questions 2-8.....	158
X-Register.....	158
Review Questions 2-9.....	160
P-Register.....	160
Review Questions 2-10.....	162
A-Register (Accumulator).....	162
Review Questions 2-11.....	164
ALU Module.....	164
Review Questions 2-12.....	166
Q-Register.....	166
Review Questions 2-13.....	167
C-Register.....	168
Review Questions 2-14.....	170
S-Register.....	170
Review Questions 2-15.....	172
Program Instruction Logic Analysis.....	172
Logic Timing Supplement.....	173
Acquisition Phase and DPA Pulses.....	173
Review Questions 2-16.....	176
Execution Phase.....	178
Review Questions 2-17.....	226
Chapter III - Computer System Maintenance.....	230
Computer Diagnostic Programs.....	230
Purpose of Diagnostic Programs.....	230
Basic Requirements.....	232
Review Questions 3-1.....	233
Computer System Troubleshooting.....	235
Troubleshooting Techniques.....	235
Troubleshooting Example One.....	237
Review Questions 3-2.....	249
Troubleshooting Example Two.....	252

	PAGE
Review Questions 3-3.....	264
Chapter Review.....	266
Signal Name Glossary.....	269



## CHAPTER 1

### COMPUTER UNITS AND PROGRAMMING

In past blocks you learned numbering systems and computer circuits. These are all put together to look at the computer as a whole. You will not be separating the computer into the circuits, but into functional parts. You will learn how to operate the COM-TRAN TEN trainer and how to make the computer do what you want. You will learn to program the COM-TRAN TEN. A computer can only do what it is told. Now start with the basic computer block diagram.

### COMPUTER UNITS AND DATA FLOW

Figure 1-1 is a block diagram of a basic computer. It is made up of 5 blocks. Each block has a distinct function. All digital computers are made of these five basic blocks. Different computers can use different combinations of computer circuits to do each of these five functions. Refer to figure 1-1.

- INPUT - The Input Unit of a computer accepts information in various forms and converts it to a form which can be used by other units of the computer.
- OUTPUT - The Output Unit accepts information from the computer and sends it to the output devices. These outputs can be in a form readable by man or in a form for the computer to use later.
- MEMORY - The Memory Unit stores information until it is needed by the computer. Memory locations are addressed so the machine can find the right information when it needs it, much like you would use a house address to find the right house. The most commonly used memory is magnetic core. It is a fast memory and will retain any information that it has in case of a power failure.

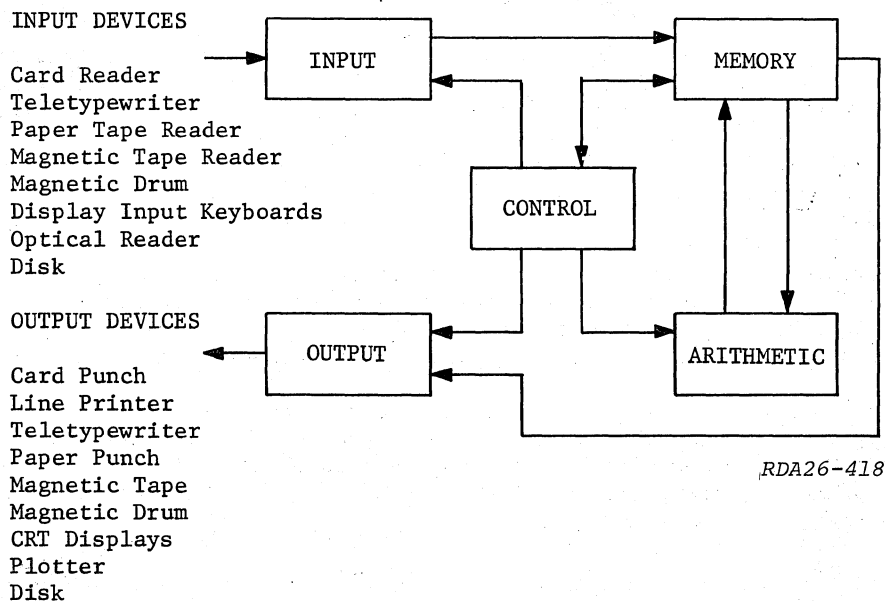


Figure 1-1

- ARITHMETIC - Arithmetic Unit performs all arithmetic and logical operations. Arithmetic units usually do nothing more than add and shift. To multiply, it does a series of adds and shifts.
- CONTROL - The Control Unit generates all the signals at the proper time to do what needs to be done. It controls all the other computer units.

#### Data Flow

Before tracing the flow of data among the five units of a computer, it is necessary to understand the definition of the following terms:

1. Computer Word - A group of binary bits, handled by the computer as a single unit. Commonly referred to as the content of a memory location.
2. Machine Cycle - The length of time which is required to acquire an instruction from memory, decode the instruction and execute it.
3. Program Time - (Acquisition Time) The portion of a machine cycle when the instruction is read from memory, decoded, and prepared for execution. *96 us*
4. Operate Time - (Execution Time) The portion of a machine cycle when the instruction is actually performed.
5. Instruction Word - A computer word having two parts; instruction code (Op-Code) and data address (Operand). The Op-Code tells the machine what to do and the Operand tells the machine where and with what data to do it.
6. Data Word - A computer word containing information or arithmetic value to be used in computations.
7. Program - A series of instruction words in logical order to solve a given problem.

With an understanding of these terms we can now discuss computer operation and data flow. Computer words, both instruction words and data words, are entered into the computer's Input unit by some external device. The Input unit puts these words into binary or some other numbering system format that is recognizable by the computer. The Input unit then transfers these computer words to the Memory unit for storage. The instruction words would be stored in logical order to form a program that would control the operations to be performed.

The Memory unit, now containing all instructions and data, can be used to output arithmetic and logical operations to the Arithmetic unit and receive the results of those operations from the Arithmetic unit. Those results can then be sent to the Output unit where they are put into a format recognized by the external output device.

Of course, all operations of the machine and the sequence of all data transfers are controlled by the Control unit. The instruction words are transferred to the Control unit from the Memory unit. The Control unit decodes these instructions into commands. The Control unit, operating under a timed sequence, causes a series of events to execute any command that it decodes.

## Data Flow Summary

Information comes into the computer through the Input unit. This Input is controlled by the Control unit. The information is put into Memory. The Control unit works with the Memory unit and the Arithmetic unit to solve whatever problem you have told it to do. When the computer gets the answer, it will put it in Memory. Then, the Control unit will control the output of the answer through the Output unit. All of this is controlled by you up to a point. You tell the computer where to get information and what to do with it. You will learn how to do this later in the block.

### REVIEW QUESTIONS 1-1

#### BASIC DIGITAL COMPUTER BLOCK DIAGRAM

#### Objective

Given a block diagram of a basic computer and a list of units and functions, label the units, match the functions to the units and trace data flow between the units.

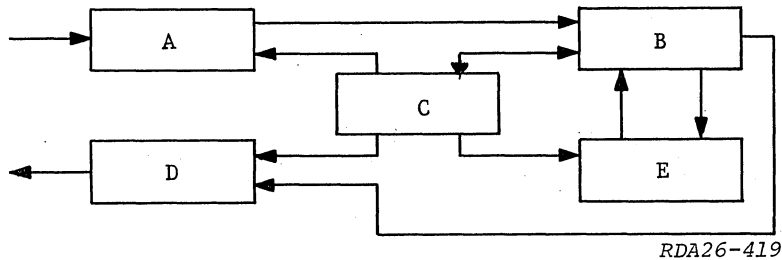


Figure 1-2

1. Analyze the basic computer block diagram (figure 1-2) and list the name of each unit to correspond with the label in the appropriate block.

- a. \_\_\_\_\_ b. \_\_\_\_\_  
c. \_\_\_\_\_ d. \_\_\_\_\_  
e. \_\_\_\_\_

2. Using the following list, match the basic computer unit with its applicable functional descriptions.

- a. Input      b. Output      c. Memory  
d. Control    e. Arithmetic

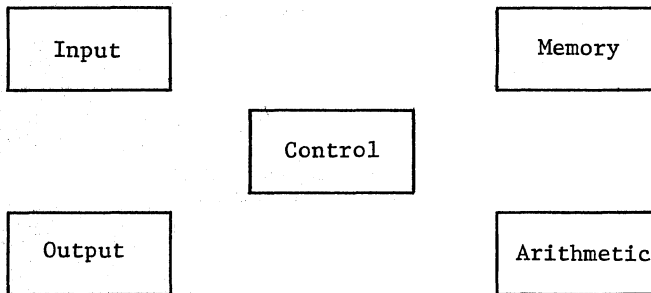
\_\_\_\_\_ Stores information until it is needed.

\_\_\_\_\_ Performs all arithmetic operations.

\_\_\_\_\_ Accepts information in various forms and converts it to a form used by the computer.

- \_\_\_\_\_ Generates signals needed to do the work.
- \_\_\_\_\_ Performs logical operations.
- \_\_\_\_\_ Accepts information from the computer and sends it to an output device.
- \_\_\_\_\_ Works with all other parts of the computer.

3. Draw all the necessary lines to properly connect the units below.



RDA26-420

4. Give the function of each of the following:

- Input-
- Output-
- Control-
- Arithmetic-
- Memory-

#### MEMORY UNITS

The difference between a memory unit and a memory (or storage) device is an important distinction. Any device which is capable of holding binary information for a period of time may be correctly called a storage device. A memory unit, on the other hand, is a complete unit composed of many storage devices and the associated circuitry which controls and operates the unit. A memory unit may be used as the memory element (central memory) of a computer system or as an auxiliary storage unit for either the input element or output element (or both elements). The most common memory devices used are magnetic cores, magnetic tapes, magnetic disk, magnetic drum, punched cards, and punched tape.

Only the central memory is used for all operations going on inside the computer. If additional data, or even a specialized program, is required, then access is made to large blocks of data in an auxiliary memory are transferred to main memory in a single operation. If main memory fills up its available storage space with intermediate results or output data, then a large block of data may be sent to auxiliary memory from the main memory. Such transfers between the central memory and an auxiliary memory are called I/O operations.

I/O operations are started by the computer through program control. After starting the transfer operation, some computers continue with their original task while the I/O operation continues. I/O transfers may be from the input element to the memory element,

from the memory element to the output element, or from the central memory to an auxiliary memory which serves both the input and output elements.

To understand the reasons why some storage units are used as central memory and other storage units are used as auxiliary, you must be able to define and use the following terms as they apply to memories: mode of access, access time, capacity, permanence, volatility, and computer word.

#### Mode of Access

Any memory unit that stores more than one item of information must have some system to identify and select a particular item for use by some other part of the computer. Normally, each separate item of data is stored in a separate "location" in the memory unit, and each location has a specific address. It is common practice to number storage locations serially in octal notation. The method used to gain access to a specific location in a storage unit (central or auxiliary) is referred to as "mode of access." There are two major modes of access: random and sequential.

**RANDOM ACCESS.** A random access memory system is one in which any location in the storage unit is equally easy to use; it takes the same amount of time to address any specific location in memory and use the data stored there. Any addressing scheme which is independent of previous addresses or that can address locations out of sequence is usually a random access system. Random access memories provide fast access to any particular item of information stored in them, and they normally have a fixed access time.

**SEQUENTIAL ACCESS.** A sequential access memory system is one in which access to memory locations occurs in series. The system must check all addresses between the present memory location and the desired location before the desired location can be used. In a sequential address system, the access time will vary depending on how many locations must be "passed through" before the needed location is found. Sequential access memories are further broken down into two groups: cyclic and progressive.

**Sequential Cyclic.** The sequential cyclic mode is a mode of access in which each location occurs in series and is available at a given fixed interval. Sequential cyclic memories normally have some rotating storage device so that the sequence of addresses and time until the needed location reappears is permanent.

**Sequential Progressive.** The sequential progressive mode is a mode of access in which each location occurs in series, but the system may move from location to location by the shortest route. A sequential progressive system does not move constantly in one direction as does the sequential cyclic mode; rather, it may "search" in either direction to locate the desired memory location. The time required to find a given address in a sequential progressive system varies depending on the distance from the starting point to the needed address.

#### Access Time

*8us*  
Access time is measured from the time information is requested to the time that information becomes available. It is the time which determines the speed of the memory system. In most applications, it is desirable to have as short an access time as possible. The central memory of a computer will always have a short access time; however, any auxiliary memories used in the system may have a relatively longer access time if they provide some other desirable feature.

In random access memory systems, the access time will be the same no matter what address (location) is selected. Sequential access system, however, will have different access times for each piece of information requested. In sequential access systems the

access time is given in maximum, minimum, and average times. For example, in a sequential cyclic mode, if the desired location is close to the starting point, the access time will be very short; if the desired location was passed just before the request was made, then the system must wait almost a complete cycle before the information is available. The average access time for a sequential system is the mean time between the minimum and maximum access times.

### Capacity

The capacity of storage may be given in terms of binary bits, characters, or computer words that can be stored. Storage devices of small capacity, such as flip-flop registers, are usually rated according to their bit capacity. When describing the storage capacity of large devices, such as magnetic tapes or drums, the "word" capacity rather than "bit" capacity is usually given. In such cases the number of bits in a word must be stated if a useful comparison is to be made between different storage units.

Access time and capacity are the two most important characteristics of any memory system. They are determined by the type of storage device used. No one memory unit, in current production, combines the desired capability of large capacity and short access time. In fact, it will be found that most large capacity storage units have a long access time; most low capacity units have a fast access time. Therefore, a combination of storage units is usually used in a computer system.

### Permanence

Permanence is the characteristic which determines whether the data in a memory unit may be erased. A magnetic memory is erasable since any selected "word" can be changed or altered without physically changing the memory unit or any of its parts. Some storage devices, however, are not erasable; that is, the stored data cannot be changed without physically replacing the storage device. For example, to alter the information stored in punched cards, new cards must be punched and used as replacements for the old cards.

### Volatility

If information is lost when power is removed from a storage unit, the memory system is said to be volatile. A flip-flop register is volatile; a punch card "deck" is non-volatile. If a computer system uses a volatile storage system, positive steps must be taken to preserve the stored information if it is not available elsewhere in the system. Therefore, valuable information is normally stored in a non-volatile unit and put into volatile storage only when it is to be operated on. In this way, critical data is preserved in the event there is a power fault. Ferrite cores are non-volatile.

Since both data (information) words and instruction words are stored in memory and are not distinguishable in form from one another, some means is needed to separate instructions from data. This can be done by restricting instructions (the program) to the area of memory or by allowing access to memory on a time-sharing basis. In a time-sharing system, memory words accessed by specific timing pulses are automatically assumed to be instructions. The time in which an instruction is transferred to the control element from memory is called the acquisition time. Acquisition time is then followed immediately by execution time in which the instruction is obeyed. If a memory word is accessed during execution time, it is automatically assumed to be data. A sequence of acquisition-execution time is called a machine cycle.

COMPUTER STORAGE DEVICES						
DEVICE NAME	CAPACITY	ACCESS MODE	ACCESS TIME	PERMANENCE	VOLATILE?	FUNCTION
FERRITE CORES	8 to 100K WORDS	RANDOM	$\frac{1}{2}$ to 10 microseconds	ERASABLE	NO	High Speed internal Central Memory
MAGNETIC DRUMS	20 to 2,000K WORDS	SEQUENTIAL CYCLIC	10 to 100 microseconds	ERASABLE	NO	Medium Speed Buffer or Bulk Storage
MAGNETIC DISCS	20 to 20,000K WORDS	SEQUENTIAL	10 to 1,000 microseconds	ERASABLE	NO	Medium Speed External Input/Output Operations
MAGNETIC TAPES	20 to 20,000K WORDS	SEQUENTIAL PROGRESSIVE	1 to 100 Seconds	ERASABLE	NO	Slow Speed External Input/Output Operations
THIN FILM	1 to 256 WORDS	RANDOM	.1 to .5 microseconds	ERASABLE	NO	Very High Speed internal Scratch-Pad Memory
DELAY LINES	5 to 10K WORDS	SEQUENTIAL CYCLIC	1 to 1,000 microseconds	ERASABLE	YES	Medium Speed internal Temporary Storage (Display)
ELECTRO-STATIC CRT	5 to 50K WORDS	RANDOM	1 to 20 microseconds	ERASABLE	YES	High Speed internal Central Memory or Buffer
PUNCHED CARDS	80 to 90 WORDS/CARD	SEQUENTIAL PROGRESSIVE	50 to 150 microseconds	PERMANENT	NO	Slow Speed external Bulk Storage or I/O
PUNCHED TAPE	1 to 1,000K WORDS	SEQUENTIAL PROGRESSIVE	10 to 150 milliseconds	PERMANENT	NO	Slow Speed external Bulk Storage or I/O
SEMI-CONDUCTOR	1 to 4,096 WORDS	RANDOM	.01 to .5 microseconds	ERASABLE	Some Types	Very High Speed internal Central Memory or Scratch

RDA26-421

Figure 1-3. Storage Device Characteristics

## STORAGE DEVICES

There are many storage devices available for use in computer systems. Some have been popular in the past and are now almost forgotten; others have been--and will continue to be--used in almost every computer system made. Some new storage devices have been invented but have not yet found their way into operational computer systems. Figure 1-3 provides information on the characteristics of many of the storage devices used in USAF computer systems.

## MAGNETIC STORAGE SYSTEMS

Ferromagnetic materials make many excellent binary storage devices for use in computers. The "polarity" (N-S direction) of a magnetic field can represent a one or zero. This fact, coupled with the close relationship between electric current and magnetism, is the reason ferromagnetic materials are used so often as computer storage devices. In fact, magnetic storage systems are the most common way of storing large amounts of computer data.

Magnetic storage systems may use tiny cores (doughnuts) of magnetic material, long strips (tapes) of magnetic material, rotating drums coated with magnetic materials, or rotating discs coated with magnetic materials. In many cases, the actual area needed to store one binary bit is as small as the head of a pin. This means that quite a lot of information can be stored in a small area. For example: a 5-inch cube of magnetic cores can store 150,000 bits of computer data; a standard 2,400-foot reel of 1/2 inch wide computer tape can hold over 2 million bits of computer data.

### Hysteresis Loop

All magnetic storage systems utilize a physical phenomenon known as residual magnetism. Residual magnetism means that a piece of ferromagnetic material will keep a given "polarity" after the magnetizing force is removed; ferromagnetic materials will store a

bit of binary data as a magnetic field. This phenomenon can be shown on a graph of magnetic flux density (B) versus magnetizing force (H). This graph is called a B-H curve and, for ferromagnetic materials, becomes a hysteresis loop. A thorough and complete understanding of the hysteresis loop, or curve, will help you to maintain ferrite core memories, tape storage units, magnetic drums, and magnetic discs. In addition, knowledge of the hysteresis loop will assist your career progression when you are faced with SKTs (Skill Knowledge Tests).

Figure 1-4 plots the values of flux density (strength and direction of magnetization) "B" versus magnetizing force "H" applied to a magnetic material. We can start our examination of the curve by locating point "X" in the center of the loop. This point represents the material in a neutral or unmagnetized condition. If a magnetizing force with a value of +Hm is then applied to the material, its flux density and direction will be forced to point +Bm. Fortunately, with most magnetic materials, any increase in the magnetizing force above +Hm will not increase the flux density above +Bm. When the material is at point +Bm, it is said to be saturated; its flux density in one direction is maximum and cannot increase. This fact will cause the hysteresis curve to become a loop. If the magnetizing force (+Hm) is now removed, the flux density of the material will drop only slightly to point +Br (residual magnetism). This slight drop, instead of the large drop common to most metals, is due to the high retentivity of ferromagnetic materials. Point +Br now represents a stable flux density and direction that can be called a binary one or zero, according to its application.

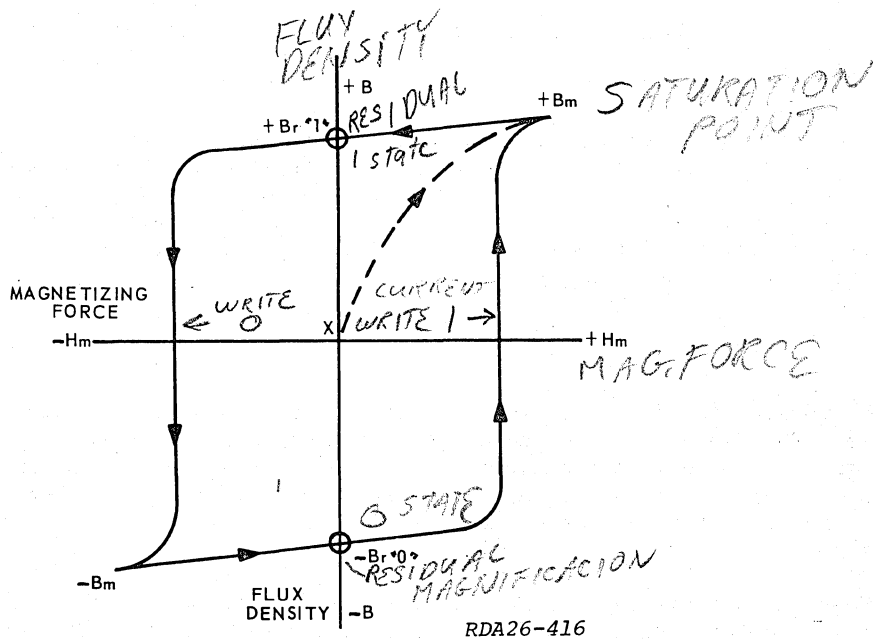


Figure 1-4. Hysteresis Loop

If we now desire to store the opposite binary number in the magnetic material, it is necessary to apply a magnetizing force of -Hm (equal in magnitude to +Hm but opposite in direction). The application of force -Hm will cause the flux density to swiftly decrease to zero and then move on to a maximum negative value, point -Bm. Again, if a force greater than -Hm is applied, the flux density cannot become greater than -Bm. The material has now become saturated with a magnetic flux density equal to +Bm but in the opposite direction. Once the force has been removed, the material will stabilize with



its residual magnetism at point  $-Br$ . If point  $+Br$  was assigned to a value of binary one, then point  $-Br$  would be a binary zero.

The application of a force of  $+H_m$  to the magnetic material at the  $-Br$  point will cause it to switch to the  $+B_m$  point. Notice that this completes the hysteresis loop and completely bypasses point  $x$ . The only way to return the magnetic material to its unmagnetized condition is to apply an AC sine wave decreasing to  $\emptyset$  volts. An examination of the hysteresis loop will reveal that, in normal operation, ferromagnetic material will remain at either point  $+Br$  (one state) or  $-Br$  (zero state). The application of magnetizing force is required to make magnetic material change states (switch from one state to another).

In some applications of magnetic storage materials, the material will be moved rapidly under a small coil; this movement of magnetic fields past a coil will induce a current flow in the coil. The direction of current flow will indicate the direction of magnetization of the material, and the magnetic field will not be changed in any way. The data stored as two different directions of magnetic fields may be "read" over and over again. This type of nondestructive readout is often used for magnetic drums, magnetic tapes, and magnetic discs. Of course, if it is desired to change the stored information, the use of the coil must be changed. To write on drums, tapes, or discs, the coil is connected to a current source; current flow through the coil produces a magnetizing force that can be used to store data in magnetic material.

One other form of magnetic storage device is a small core of ferromagnetic material that has several small wires passing through it. A single bit of information may be stored in or read from a single core. These cores are quite small, perhaps 15 to 50 thousandths of an inch in outer diameter. To produce a useful computer memory device, several thousand cores and their wires are placed in a frame (plane) about 6 inches square by 1/2 inch thick. These planes can be stacked together to make a memory array that will hold anywhere from 150,000 to 2 million bits of computer data.

#### Ferrite Core Memory

A ferrite core memory cannot be moved past a coil to sense the state of the cores; therefore, other electronic systems must be used to read out data from ferrite core arrays. Unfortunately, most of these systems involve destructive readout of the data stored in the cores. Destructive readout of ferrite cores takes advantage of the fact that the collapse of a magnetic field that surrounds a wire will induce current flow into the wire. Core memories are physically small and simple--compared to the motors and other mechanical parts needed with drums, tapes, and discs--but they have complicated electronic circuits needed to control the writing into, reading from, and restoring of data into the cores. Not all "core" memories consist of individual cores that are threaded by wires and assembled into planes. Some memory systems use ferrite plates with a series of holes through which the wires needed to read and write data pass. The plate is so constructed that each of the areas around the holes acts as a single core. The ferrite plate has characteristics similar to a plane of ferrite cores.

**COINCIDENT CURRENT ADDRESSING.** In order to use a ferrite core, the computer must be able to write information into any given core and also to read information from any given core. The techniques used for writing into or reading from a selected core or group of cores (a computer word) are called address selection techniques. The address selection technique that this SG explains is one of a number of techniques that are available to manufacturers of ferrite core memories. The technique explained is the "coincident current technique" and is the most commonly used system of getting data into or from a magnetic core.

The basic concept of the coincident current selection technique is that two wires are used to supply the magnetizing force ( $-H_m$ ) required to make a ferrite core change

states. This magnetizing force is in the form of current passing through a pair of wires. (Current flow through a wire sets up a magnetic field around the wire, which is directly proportional in strength to the current flow through the wire.) A combined current which produces a magnetizing force of  $\pm H_m$  on the hysteresis curve is called a full-select current. Each wire carries a "half-select" current; that is, a current with a magnitude of  $H/2$ . If, and only if, a core receives the effect of a "full-select" current will it completely change states. This means that a wire carrying a half-select current can be threaded through many cores, but the only core affected by this half-select current will be the one core that is receiving an additional half-select current from another wire that is also threaded through many unaffected cores.

After the ferrite core is completely saturated in one direction and the magnetizing currents removed, the retained flux will be almost that of saturation, and the direction of magnetization can be considered the one state. If and when the half-select currents are reversed to magnetize the core in the opposite direction, then the core will be considered to be in the zero state. In the coincident current addressing technique, each half-select drive current applied to the wires is of such a value that each current alone does not provide enough magnetizing force to "switch" the core or cause it to reverse flux direction. However, with both half-select drive currents applied, there is sufficient magnetizing force to cause the core to switch and retain, through its residual magnetism, the new condition after the drive currents are removed.

Assume that the cores are designed to switch when approximately 0.350 ampere drive current ( $\pm H_m$ ) is felt by the core. Each drive line carries approximately 0.250 ampere. Either of the drive lines alone does not carry a sufficient current (magnetizing force) to switch the core. However, a combination of the two drive currents (0.25 ampere + 0.25 ampere) will exceed the required switching value and cause the core to switch, providing that it is not already in the desired state.

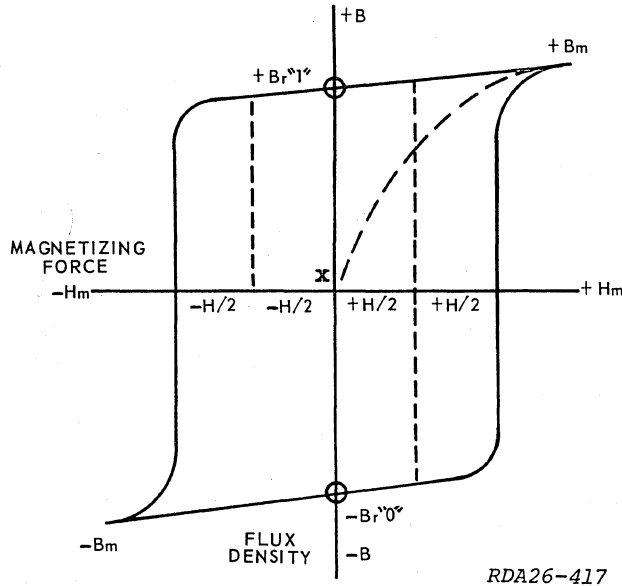


Figure 1-5. Detailed Hysteresis Loop

The operation of the coincident current addressing technique can best be explained by a more detailed study of the hysteresis loop of a ferrite core. (Refer to figure 1-5.) In the initial application of selection drive currents to the core, consider the core to be demagnetized (point "x"). As positive half-select drive currents are applied, magnetization will take place. One half-select current will not be enough to saturate the core and place it at point  $+B_m$ ; however, an additional half-select current will saturate the core and place it at point  $+B_m$ . As the drive currents fall from maximum positive to zero, the core's retentivity will allow only a slight drop in its flux density. When the drive currents fall to zero, the flux will remain stable at point  $+B_r$ . At this time the core is in the one state.

If the core is stable at point  $+B_r$  and a negative pulse of current is then applied to both half-select drive lines simultaneously, the core will switch to point  $-B_m$ . As the drive currents decay, the core flux density will come to rest at point  $-B_r$ , and the core will remain in the zero state. The important thing to notice is that if the core is in a residual state ( $\pm B_r$ ) and only a single half-select drive current ( $H/2$ ) is applied and removed, the state of the core will not be changed. The square shape of the hysteresis loop shows that a certain current or magnetizing force must be applied to a ferrite core to cause it to change states. If less than a minimum force is felt by the ferrite core, it will remain in its previously established stable state.

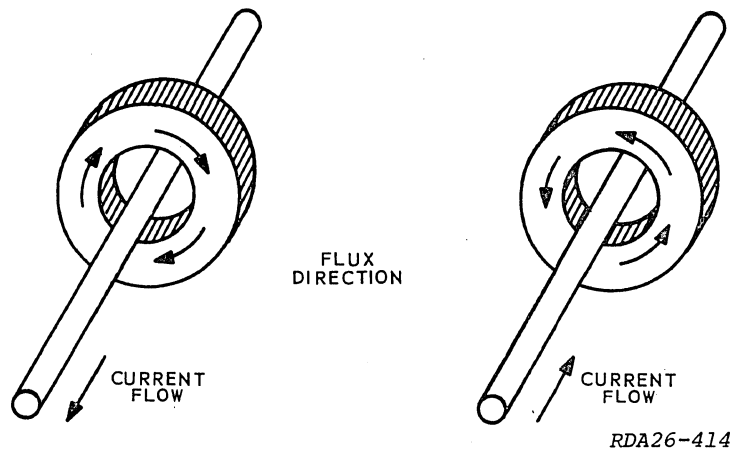


Figure 1-6. Magnetic Fields in Core

X AND Y SELECTION LINES. To understand how drive current flowing in wires threaded through a ferrite core can produce one and zero states in the core, refer to figure 1-6. This illustration shows that current flow into the page would produce cores with a counterclockwise magnetic field. Current flow out of the page (toward you) would produce cores with a clockwise magnetic field. These fields are in agreement with the "left hand rule" which was mentioned briefly when you studied the basics of magnetism in the electronic principles course. One direction of the core's magnetic field is assigned the value of binary one and the opposite direction of magnetic field is assigned the value of binary zero. Notice that the core's magnetic field completely surrounds the wire passing through it; a collapse of the magnetic field--or the switch from one direction to another--will cause current to flow in a wire passing through the ferrite core.

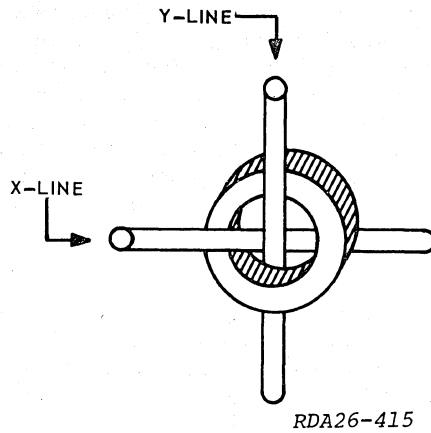


Figure 1-7. Core Address Lines

Most of the ferrite cores used to store computer data do not have a single wire applying magnetizing force to the cores. Instead they use two wires, each carrying a half-select drive current. For convenience, these address selection lines are usually called the X (horizontal) and Y (vertical) address lines. Figure 1-7 shows a single core threaded by X and Y address selection lines. Of course, each X line and each Y line passes through several other cores. Figure 1-8 shows a typical ferrite core plane with its X and Y selection lines.

The ferrite core plane in figure 1-8 contains 16 cores number 0 through 17 (octal). Each core is threaded by two address selection drive lines; each wire will carry a half-select current. Only the core that receives a magnetizing force from two wires (coincident X and Y currents) will be selected. Each core has its own unique pair of X and Y lines that will make it become a selected core. It is standard practice to label cores by their X-Y address. For example, core 11(8) would be identified as address X2Y1; core 11(8) has a location of X2Y1.

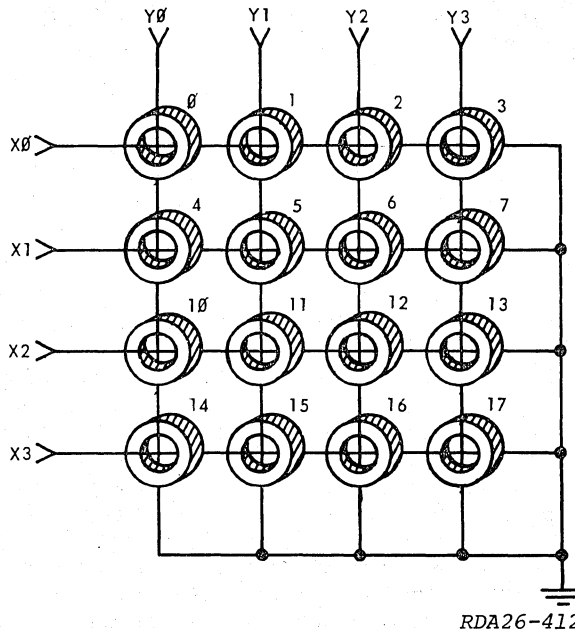


Figure 1-8. Ferrite Core Plane

Figure 1-9 shows four cores arranged in a very simple plane. Each core is threaded with two lines, X and Y. Assume that all cores are in the zero state and we want to write a one into core  $X_0Y_1$ . If a full select current with a value of  $+H_m$  is applied to the  $X_0$  line, then both cores  $X_0Y_0$  and  $X_0Y_1$  will be switched to the one state. If a full-select current of value  $+H_m$  is applied to the  $Y_1$  line, then both cores  $X_0Y_1$  and  $X_1Y_1$  will be affected. This is not quite what we want to do.

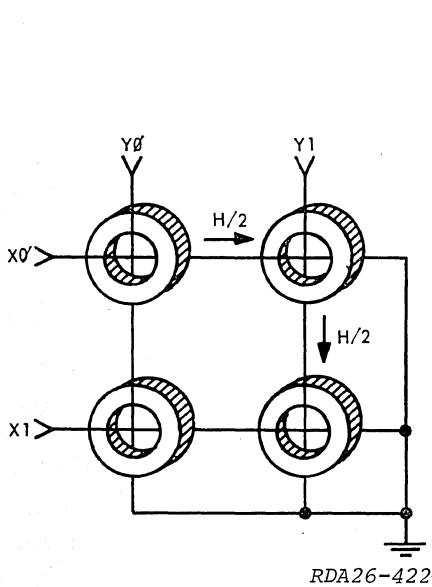


Figure 1-9. Address Selection

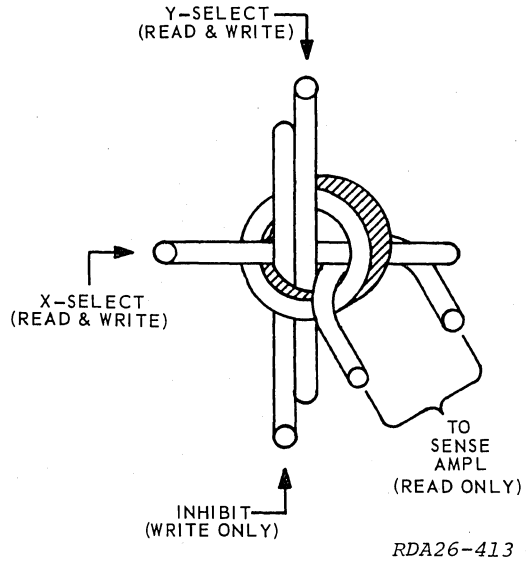


Figure 1-10. Ferrite Core

If, however, half-select currents of  $+H/2$  are applied to the  $X_0$  and  $Y_1$  lines at the same time, then only core  $X_0Y_1$  will receive the effect of a full-select current ( $+H_m$ ). Each of the other cores along the  $X_0$  and  $Y_1$  lines will receive only a half-select current. Therefore, only core  $X_0Y_1$  will be switched to the one state, and the other three cores in the plane will remain in the zero state. Two cores ( $X_0Y_0$  and  $X_1Y_1$ ) receive half-select current, which is not enough to cause them to switch, and one core ( $X_1Y_0$ ) receives no current at all. Any one of the four cores in this sample plane may be selected by pulsing the correct combination of X and Y lines with simultaneous half-select currents.

**INHIBIT WINDING.** Notice that the selection of a core that was in the zero state by positive half-select currents caused it to switch to the one state. The application of negative half-select currents to a core can be used to switch it from the one to the zero state. However, sometimes we need to select a core but leave the state of that core unchanged. This need to select, but not change, usually is the result of wanting to "write a zero." "Writing a zero" requires that we add another wire to the simple core plane, an inhibit winding. The inhibit winding will carry current that opposes one of the positive half-select currents and prevents the address selected core from switching to the one state. Since only one core from a plane can be selected at any one time, a single inhibit winding can be threaded through all the cores on a plane and energized only when it is necessary to "write a zero" into the selected core. In figure 1-10 the inhibit winding is parallel to the Y select wire.

SENSE WINDING. If information is stored in a ferrite core, we need some way of sensing or removing that information. This is usually done by adding yet another wire to the simple core plane, a sense winding (see figure 1-10). The sense winding is used to observe the current produced when a magnetic core is changed from the one to the zero state. The use of a sense winding requires that we apply negative half-select currents to the core in an attempt to put it in the zero state. If the core was in the one state and it is switched to the zero state, then the sense winding will receive the current produced by the collapse of the core's magnetic field. If the core was already in the zero state when the negative half-select currents were applied, there will be no change in the core's magnetic field and no current flow in the sense winding. Unfortunately, this sensing technique removes the stored "one" from the ferrite core; we are using what is called destructive readout. Again, since only one core on the plane can be selected at one time, only one sense winding per plane is required. This single sense winding can be looped through all the cores in a plane in such a way that the "noise" produced by the half-select current present in the plane will not affect the sense winding output.

Ferrite cores are very sensitive to temperature changes, operating ideally at room temperature (70° to 80°). The hysteresis loop changes shape as a function of temperature. The "B" dimension of the loop decreases as temperature decreases. Because of the curtailed dimension of the loop along the "B" axis, the voltage generated on the sense line would not be sufficient to represent a one. Decrease in temperature also widens the dimension of the loop along the "H" or horizontal axis. This increased loop dimension reduces the possibility of core switching by the full select current, since the large magnetizing force would be required. Eventually, the width of the "H" dimension may be large enough to make it impossible to drive the core to its new state. Temperature increases cause the "H" dimension at the loop to become narrow, and flux density along axis "B" increases. As the "H" dimension becomes shorter the core is switched by smaller currents. The core no longer discriminates, and it switches at signals below the full select current. In short, at low temperatures, the core switches less readily, and at high temperatures it is inclined to switch on any pulse.

CORE MEMORY PLANE. Figure 1-11 illustrates a complete coincident current core memory plane with all its address selection wires and windings. Any core in this plane may be selected and a one or zero written into the core. Or, the state of any core may be sensed by applying the correct half-select currents to one of the X address selection lines and one of the Y address selection lines simultaneously. It may be seen that half-select write pulses (+H/2) on both a selected X line and Y line will write a one in a single selected core. For instance, if the X<sub>0</sub> and Y<sub>3</sub> address lines are pulsed with a positive half-select current, only the core at X<sub>0</sub>Y<sub>3</sub> will receive the effect of a full write current. When a negative half-select read current is applied to a pair of X and Y selection lines, an output may be sensed by the sense windings when the selected core switches from the one to the zero state.

Read Operation. Because core memory systems use a destructive readout, it should be clear that in the operation of a coincident current core memory the read operation will occur first. The read operation makes use of the sense line threaded through all the cores of a plane to determine if the selected core was in the one state. If a negative half-select read pulse (-H/2) is applied to an X address line and a negative half-select current read pulse (-H/2) is applied to a Y address line simultaneously, the core that receives the effect of a full-select read current will be put to the zero state. If this core was in the one state, it will reverse its direction of magnetization and create a large change in flux density. This change in flux density surrounding a sense wire will cause current to flow in the sense line, resulting in a detected output by the sense amplifier. Therefore, it is only the selected core which is capable of producing a binary output. When a certain core is selected, the output of the sense amplifier will represent the state of only the selected core. If at the time of application of the half-select read current there is no output from the sense amplifier, this indicates that the core was, and still is, in the zero state; if an output is sensed, then the core was in the one state, now the core is in the zero state.

Write Operation. After the selected core has been put to the zero state by the read operation, the write operation can then either put a one into the core or leave it in the zero state. A separate inhibit line is threaded through all the cores on the plane so that a current pulse in the line will oppose one of the address lines. There is also a driver for the inhibit winding which can be gated on or off, depending on whether a zero or one is to be written into the selected core of the plane. The value of the current through the inhibit winding is the same as a half-select current used in the X and Y address lines.

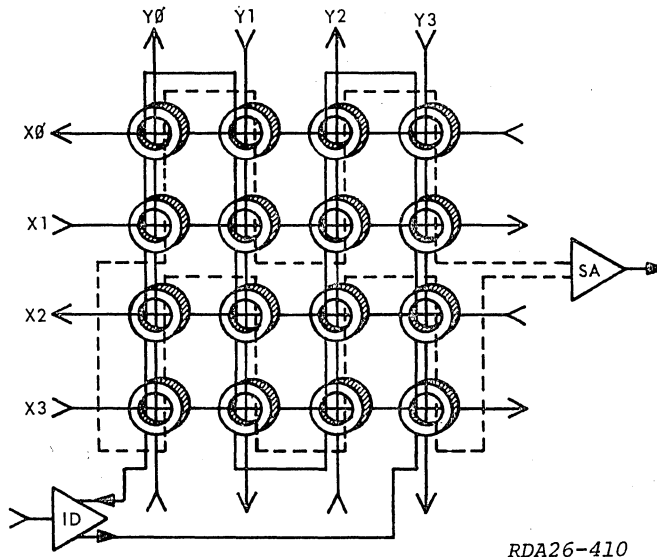


Figure 1-11. Core Memory Plane

Since the total write currents applied to the selected core by the X and Y address selection lines and the pulse from the inhibit driver oppose each other, if all these currents are applied at the same time, the total current through the selected core will only be equal to a half-select current, which is not enough to switch a core to the one state. A one or zero may therefore be written into the selected core by first clearing the core during the read operation and then turning the inhibit driver on when a zero is needed and turning it off when a one is needed, while applying half-select pulses to the X and Y address lines. Notice that this inhibit technique for writing ones and zeros into a core requires that the core be cleared to zeros before the write operation begins.

CORE MEMORY ARRAY. A complete coincident current, ferrite core memory consists of a number of planes stacked together in a rectangular array. (See figure 1-12.) The X address selection lines and the Y selection lines of each plane are connected in series. This means that a pulse fed to the X0 winding of the first plane must travel through the X0 winding of the second plane, and so on, until it passes through the X0 winding of the last plane in the array. Figure 1-12 illustrates an array with four core planes in which each plane contains 16 cores. In this array a half-select pulse (read or write) would have to travel through 16 cores, four cores on each plane.

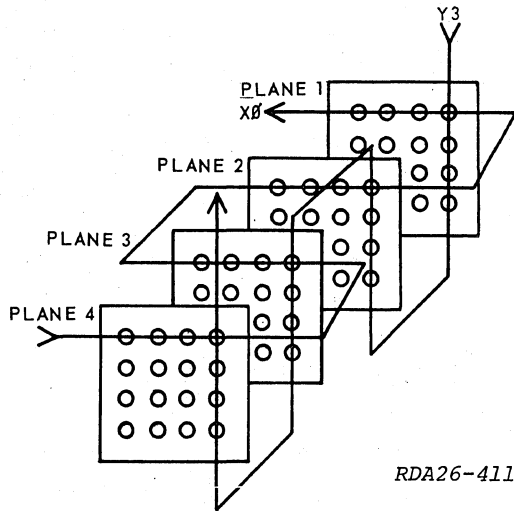


Figure 12. Core Memory Array

Each plane has its own sense winding (not shown in figure 1-12); however, the sense windings are not connected together in any way. Instead, a sense amplifier is connected to the sense winding from each plane to indicate if a one or zero was stored in the selected core of that plane. Each plane also has an inhibit winding that is not shown in figure 1-12. The inhibit winding is used during a write operation when it is necessary to "write a zero" into the selected core of that plane.

When using this type of array, there will be as many core planes in the array as there are bits in the computer word. Each plane will have its own sense line for reading the bits of the word and its own inhibit line to control the writing of ones and zeros into the bits of the word. The number of cores in any one plane will determine the number of computer words that the array can store. The array shown in figure 1-12 can hold 16 four-bit computer words.

**CORE MEMORY TIMING.** The same timing sequence is used whether the computer is to read information from the core memory or write information into the core memory. The total time taken by the complete timing sequence is called the "memory cycle," and it is one of the principal speed determining factors for a ferrite core memory. Each memory cycle consists of two portions, the first of which is called the read portion and the second of which is called the write portion. Figure 1-13 shows the sequence for all the pulses that could occur during a memory cycle. Whether we want to read from or write into memory will control which pulses are generated and used. It should be noted that the 8 microsecond memory cycle time is only an example and will vary from system to system.

The READ PULSE will be used to control the timing of the negative half-select pulses applied to the X and Y address selection lines. The READ SAMPLE PULSE will be used to turn on the sense amplifiers of each plane at the time when the flux change in the selected core of the plane will be at its greatest (if the core was in the one state). The INHIBIT PULSE is fed to the inhibit drivers for each plane; it will control the timing of the inhibit drivers if we intend to "write a zero" into the selected core on that plane. The WRITE PULSE will be used to control the timing of the positive half-select pulses applied to the X and Y address selection lines.



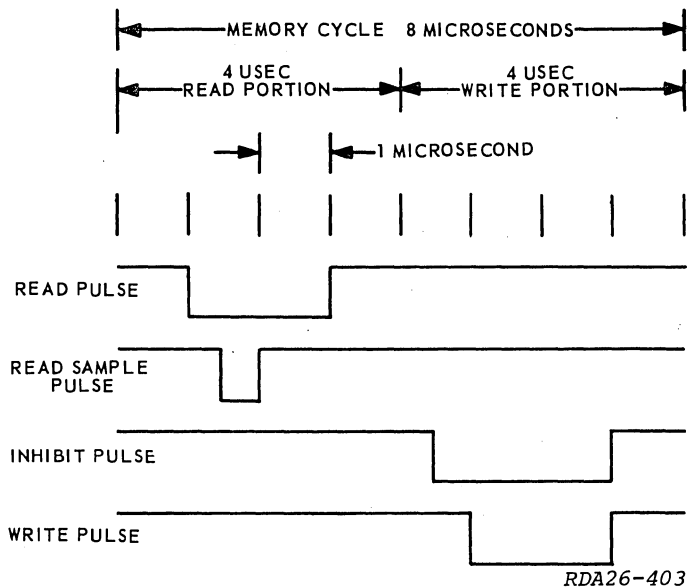


Figure 1-13. Memory Cycle Timing

Read Memory Cycle. Assume that at the beginning of the read memory cycle, core location  $X\emptyset Y3$  has been selected by external circuits to be read from. The read portion of the memory cycle in figure 1-13 will generate both the read pulse and the read sample pulse. The read pulse will cause the  $X\emptyset$  and  $Y3$  address selection lines to be pulsed with negative half-select read currents. This will cause the selected core in all four planes to be set to the zero state. If a large signal is received by the sense amplifier connected to a given plane at this time, the selected core in that plane contained a one; if a small signal is received, the selected core in that plane contained a zero. (A small current flow in the sense winding would be caused by noise or circuit unbalance and can be disregarded.) The read sample pulse "strokes" the sense amplifiers at the time when current flow in the sense windings should be at a maximum. If planes 1 and 3 produce ones and planes 2 and 4 do not, the computer word that was previously stored was (MSD) 0101 (LSD). This word will be fed out to external circuits and also be written back into the memory array. The output of each sense amplifier is used to set a storage device to the one state during the read portion of a memory cycle (providing a sense amplifier output is present), and the contents of these storage devices is then used to control the inhibit drivers during the write portion of a read memory cycle. In our example, only the inhibit drivers connected to planes 2 and 4 will be enabled by signals from their respective storage devices and conduct during the write portion. The selected cores in these planes will remain in the zero state while the selected cores in planes 1 and 3 will be set to the one state by positive half-select address selection pulses applied to all four planes as a result of the write pulse. Thus, after the write portion of the read memory cycle, the selected cores will again contain 0101, just as they did before the memory cycle began. In addition, the computer word 0101 will still be in the external storage devices and available for use by the computer.

Write Memory Cycle. Let us now change the data that is stored in location  $X\emptyset Y3$  to (MSD) 1101 (LSD). Again external circuits are used to select address lines  $X\emptyset$  and  $Y3$ , and they are then forced to generate the negative half-select read currents by the read

pulse. However, during the read portion of a write memory cycle, the read sample pulse is not generated and, as the selected core in the planes go to zero state, the sense amplifiers will not reflect the ones and zeros stored in memory. Therefore, the storage devices connected to the sense amplifiers will not be changed; the storage devices will keep the 1101 that we want to write into memory. As the write portion of the memory cycle begins the storage devices will allow the inhibit pulse to turn on the inhibit drivers. The inhibit driver for plane 2 will be turned ON and the inhibit drivers for planes 1, 3, and 4 will remain OFF. When the write pulse is generated, the positive half-select write currents applied to planes 1, 3, and 4 will cause their selected cores to go to the one state. However, in plane 2, the combination of positive half-select write currents and a negative inhibit current will leave the selected core still in the zero state. At the end of the write memory cycle, the computer word 1101 has been written into location XØY3.

Figure 1-14 shows the current flows required for the storage of information into a ferrite core during the write portion of any memory cycle (read or write). During the read portion of any memory cycle, we read all cores at the selected location by applying two negative half-select current whose timing is controlled by the read pulse. During the write portion of any memory cycle, we attempt to store ones in the selected cores by applying two positive half-select currents whose timing is controlled by the write pulse. However, the inhibit pulse is available as an opposing current flow to prevent the writing of a binary one if that is what we want. The primary difference between a read memory cycle and a write memory cycle is the generation or absence of the read sample pulse.

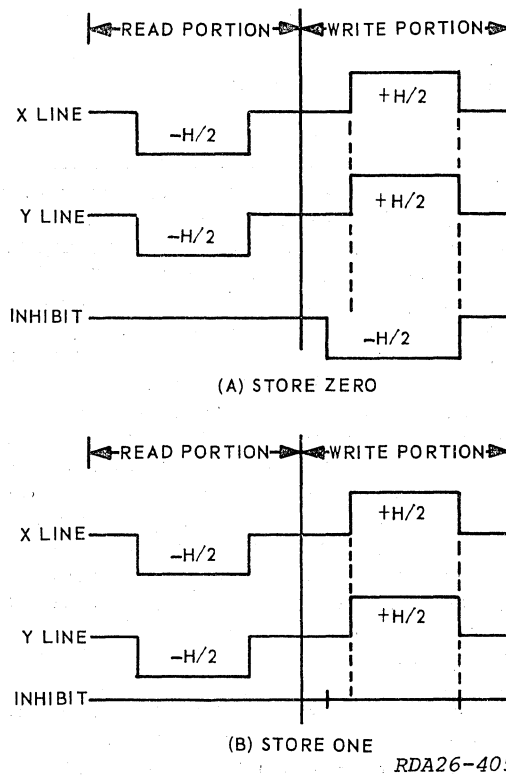


Figure 1-14. Memory Current Flows

MEMORY ADDRESS REGISTER (MAR). The computer controls the selection of specific X and Y drive lines in the ferrite core memory by use of a storage register called the memory address register (MAR). The configuration in the MAR is fed to decoders which select 1 X and 1 Y drive line in the array. The X and Y signals are combined with either the read or write pulse to produce half-select currents needed to read or write into the ferrite core memory. Intersection of these signals will pick out specific cores on a plane. The selected cores on all planes make up the computer word. Figure 1-15 illustrates, in block diagram form, the operation of an address selection system for a ferrite core memory with a sixteen word capacity.

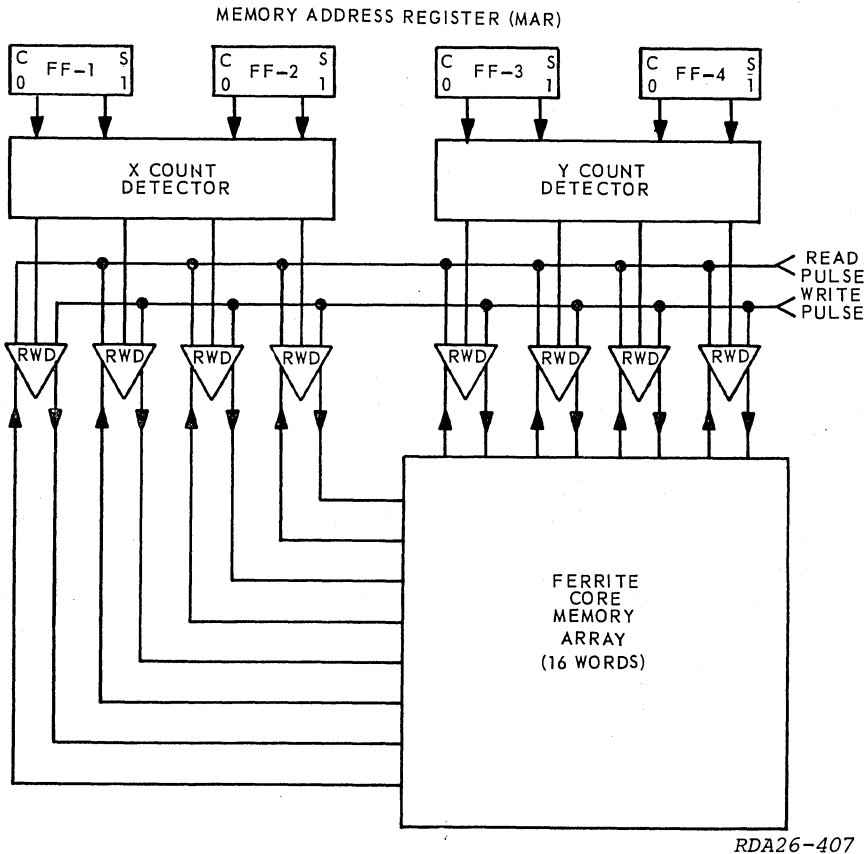


Figure 1-15. Core Memory Addressing

Flip-flops are generally used as the storage devices in the memory address register (MAR). For a memory with 16 locations (addresses), four flip-flops would be needed to give 16 different binary combinations. The input to the MAR may come from several different parts of the computer: a sequence counter (program counter) used to step through a series of memory locations from part of a previously read computer word, or from special registers. The memory address register may be divided into two parts - the least significant half to feed the X count decoders and the most significant half to feed the Y decoders.

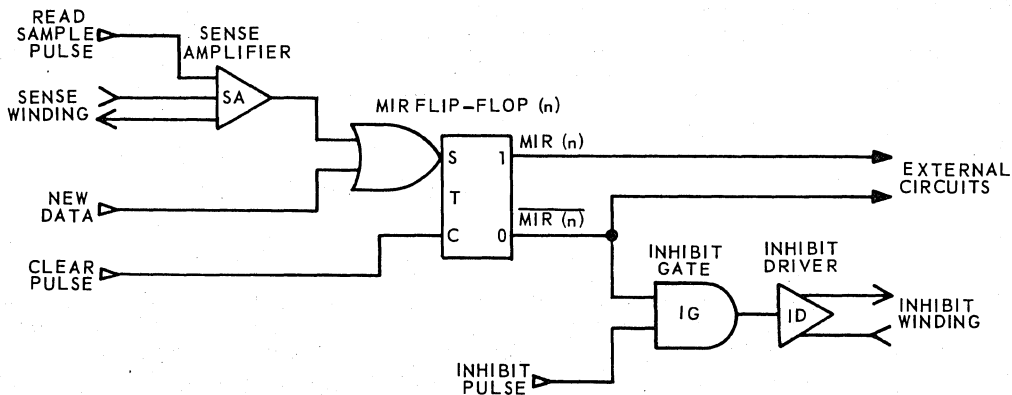
Detector Matrices. The outputs of the MAR are connected to count detectors which produce independent signals for each separate count in the memory address register.

Decoder matrices may be constructed of AND gates, transistors, diodes, or special purpose magnetic cores. Whatever device is used, the function of the detector matrices is the same; one output line is selected for each different combination of inputs from the MAR flip-flops. The X count detector decodes the two least significant bits of the MAR, and the Y count detector decodes the two most significant bits of the MAR.

The Read-Write Drivers. The Read-Write Drivers (RWD) feed a particular X and Y address selection line with both negative (read) and positive (write) half-select currents. Each RWD has three inputs: a unique X or Y count decode, a read pulse, and a write pulse. The X or Y count decode enables the read-write driver and permits it to generate current, with direction and timing determined by whether a read or write pulse is strobing the RWD. During any memory cycle, the read pulse will occur first and cause the enabled RWDs (one X and one Y) to produce negative half-select pulses of current which will put all the cores at the MAR address to the zero state. When the write pulse occurs in the second portion of the memory cycle, the enabled RWDs will produce positive half-select pulses of current that could put all the cores at the MAR address to the one state. The current supplied by a typical read-write driver is on the order of 1/4 ampere.

The operation of the memory address register and its associated circuitry causes a single core in each plane of a memory array to receive the effect of a full read current and then a full write current. The address stored in the MAR is decoded, and this decoded output is used to turn on selected read-write drivers. The timing and direction of current flow from the RWDs is controlled by read and write pulses generated in memory timing circuits.

MEMORY INFORMATION REGISTER (MIR). The memory information register (MIR) is a flip-flop register that is used to store the binary read from or to be written into a ferrite core memory. The MIR has one flip-flop for each plane in the core array; therefore, the size of the memory information register is equal to the number of bits in the core memory word. Binary data stored in the MIR as a result of a read memory cycle can be used by many different circuits in the computer; many different parts of the computer may provide the new data to be written into core memory. The MIR in the COM-TRAN TEN computer is called the Buffer or B register.



RDA26-404

Figure 1-16. MIR Flip-Flop Circuit

Figure 1-16 shows one flip-flop of a memory information register and its associated circuitry. Each flip-flop in an MIR has identical circuitry.

The individual flip-flops in a memory information register receive a clear pulse (and sometimes new information) before the actual memory cycle begins. As the read portion of the memory cycle occurs, the read sample pulse is used to "strobe" the sense amplifier and allow the sense amplifier to enter the core data into the MIR flip-flop. During the write portion of a memory cycle, the zero side output of the MIR flip-flop is used to turn the inhibit driver ON or OFF.

**Read Memory Cycle.** Before a read memory cycle begins, the location (address) of the word to be read from memory is loaded into the memory address register (MAR), and the memory information register (MIR) is cleared. During the read portion of the read memory cycle, the selected core in each plane receives two half-select read current pulses and is set to the zero state. The read sample pulse allows the transfer of the ones stored in core memory through the sense amplifiers to the MIR. If the selected core in the plane contains a one, current flows in the sense winding; this current flow is detected by the sense amplifier, and is then used to set a one in the memory information register flip-flop. If the selected core in the plane contained a zero, the current received by the sense amplifier is small and no pulse appears at the output of the sense amplifier; therefore, the MIR flip-flop remains in the zero state. The selected core in the plane is still in the zero state.

During the write portion of the read memory cycle, the selected core in the plane will receive two half-select write currents that attempt to switch the core to the one state. However, the data bit now stored in the memory information register flip-flop will control the final state of the selected core. If the MIR flip-flop contains a zero, its zero side output will turn ON the inhibit driver feeding the plane. The current from the inhibit driver will oppose the current flow through one of the address selection lines and prevent the core from switching to the one state. If the MIR flip-flop contains a one, the lack of a zero side output will prevent the inhibit driver from opposing the current flow in the address selection line, and the two half-select write currents will switch the selected core to the one state. As a result of the data placed in the MIR during the read portion of a read memory cycle, a complete read memory cycle will return the original core information back into the cores. The original core data will also remain in the memory information register for use by the external circuits of the computer.

**Write Memory Cycle.** Before a write memory cycle begins, the location of the word to be written into is loaded into the memory address register (MAR). The memory information register (MIR) is cleared and then loaded with the new data. During the read portion of the write memory cycle, the selected core in each plane receives two half-select current read pulses and is set to the zero state. However, no read sample pulse is present during the read portion of a write memory cycle. The sense current will not get past the sense amplifier, and the new data entered into the MIR before the write memory cycle began is not changed.

During the write portion of the write memory cycle, the selected core in the plane will receive two half-select write currents that attempt to switch the core to the one state. At this time, the new data loaded into the MIR before the write cycle began will control the final state of the selected cores. If the MIR flip-flop contains a one, the binary zero from the zero side output of the flip-flop will prevent the inhibit driver from producing current to oppose one of the half-select currents, and the core will switch to the one state. If the MIR flip-flop contains a zero, its zero side output will turn the inhibit driver ON, and this current flow will oppose one of the half-select write currents, preventing the core from switching. The new data placed into the memory information register before the write memory cycle began destroys the old core information, and the new data will be stored in the selected address. This new data will remain in the MIR for further use by the external circuits of the computer.

**CORE MEMORY SUMMARY.** The small memory unit described here is typical of random access, ferrite core, coincident current memories now in use by the USAF except for its size. Most memories now in use contain several thousand cores as a minimum. While the fundamentals of such memories are the same, the circuitry used to control and address them will, of necessity, differ in details.

The memory cycle times vary with different machines, but most machines have memory cycles in the 1- to 8-microsecond region. Some faster memories have access times of less than 1 microsecond and it is possible to build ferrite core memories that operate in the 0.1- to 0.3-microsecond region.

A memory cycle is divided into two portions: a read portion and a write portion. The read portion obtains data from the cores by setting them to the zero state. This is called destructive readout. Those cores that switched from the one to the zero state will produce current flow in a sense winding that can be detected and the data transferred to storage flip-flops in the MIR. The write portion of a memory cycle attempts to set all addressed cores to the one state; the data stored in the MIR will control whether ones or zeros are written into the selected cores by controlling inhibit drivers.

The selection of which cores are to be read from and written into is done by a memory address register and its associated decoders and read-write drivers. They supply half-select currents whose coincidence at particular cores decides which cores will be affected. The read-write drivers can supply both positive and negative half-select currents.

The method of reading and writing discussed here is but one of several possible ways to use a ferrite core memory. It has been described in general terms so that the principles can be applied later to a specific equipment. The terminology used in this book may be slightly different from that which will be used on actual equipment. However, if you understand the basic principles of addressing, reading, writing, and inhibiting as used in this basic ferrite core memory unit, you should have no trouble transferring this knowledge to the "real world."

#### Magnetic Drum Memory

The ferrite core memory unit uses the principle of setting an essentially bistable device to one of its two states. It provides very fast access time and sufficient capacity for use as the central memory element of a computer. Unfortunately, the ferrite core memory unit has complete electronic circuits used to read from or write into core storage. These circuits increase the cost and reduce the reliability of storing a large number of bits in a core memory unit. While large core memories which can store a million bits have been constructed, some large machines require the storage of  $10^{12}$  bits. This would require the use of 10,000 core memories. Fortunately, there are several units which can store large amounts of data and still provide reasonable access time and low cost per bit. The magnetic drum storage unit is one of these devices.

The magnetic drum storage system is presently the most common type of endless track memory. Its most important components are a rotating drum and a set of stationary heads. In general, the capacity of a drum is proportional to its surface area, but an important factor which must be taken into consideration is the head separation. Direct contact of the read-write heads and the drum surface will produce the largest output and, hence the least chance for error. However, this situation would cause great friction and the drum would have only a short life span. In moving the head slightly (.001 to .002 inch) away from the drum surface, life of the unit is increased but capacity is decreased, and the possibility of cross-talk is introduced.

**MAGNETIC DRUM CONSTRUCTION.** A magnetic drum consists basically of a rotating cylinder coated with a thin layer of magnetic material which has a hysteresis loop similar to that

of the material used in magnetic cores. A number of read-write heads are mounted along the surface of the drum. These heads are used to store information by magnetizing very small areas on the drum surface or to read information by sensing the passage of the magnetic field from previously recorded information. Figure 1-17 shows a drum with only a few read-write heads for the purposes of clarity. Standard magnetic drums used in computer systems have up to several hundred read-write heads scattered about their surface.

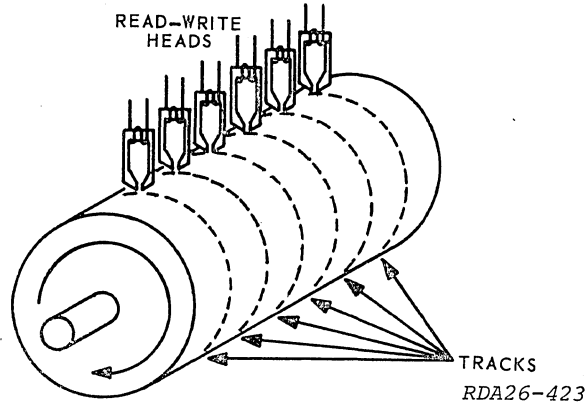


Figure 1-17. Magnetic Drum Storage

As the drum rotates, a small area continually passes under the heads. The area under a single head is known as a track or channel. One track extends completely around the circumference of the drum and can hold many bits of data. The space in a track required to store one binary bit is known as a cell. The size of a cell depends on the design of the read-write head, its spacing from the drum surface, and the speed of rotation of the drum. A group of tracks is called a field. All the cells which are under a set of read-write heads at the same time are called a register. In some drum memory units, a track is subdivided into sectors. (A sector is an angular subdivision of the circumference of the drum.) Figure 1-18 is a graphical representation of a cell, track, field, register, and sector.

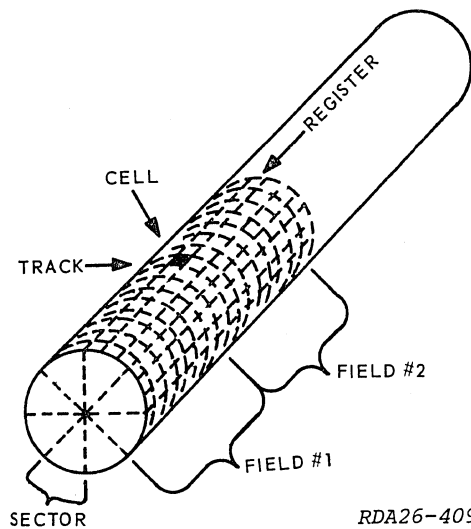


Figure 1-18. Drum Organization

Generally, one or more of the tracks is used to provide timing signals for the drum's control circuits. A series of timing signals is permanently recorded around the timing track, and each signal defines a time unit for the system. The timing track is then used to determine the location of each set of storage cells around the tracks. For instance, if the timing track is 60 inches in length and timing pulses are recorded at a density of 100 per inch, there will be 6,000 locations for bits (cells) around each of the tracks. If the drum has 30 tracks plus the timing track, the drum will have the capacity to store a total of 180,000 bits.

Information is written onto the drum by passing current through a winding on the write heads. This current causes flux to be created through the core material of the head. Some drum systems use separate heads for reading and writing, and others use combined read-write heads. The head consists of material of high permeability around which a coil is wound. When information is to be written on the surface of the drum, pulses of current are driven through the winding. The direction of flux through the head, and in turn the polarization of the magnetic field recorded on the surface of the drum, depends upon the direction of current through the coil.

The gap in the core presents a relatively high reluctance path to the flux generated by the current through the coil. Since the magnetic material on the surface of the drum is passing near the gap, some of the flux passes through this material. This causes a small area of the drum surface to be magnetized and, since the material used to coat the surface of the drum has a relatively high retentivity, the magnetic field remains after the area has passed from under the head, or the current through the coil is discontinued. It should be noted that the head does not actually touch the surface of the drum. Instead, to prevent wear, the heads are located very close to the drum surface but not touching it. The drum must, therefore, be of a very constant diameter or the distance between the heads and the drum will vary. If the head moves farther from the surface of the drum, the signal recorded will become weaker.

The signals recorded on the surface of the drum are read in a similar manner. When the areas which have been magnetized pass under the head, some of the magnetic flux is coupled into the head and changes the current flow in the head. This flux induction is changed to signals in the windings. These signals are then amplified and interpreted.

The size and storage capacities of magnetic drums vary greatly. Some drums with capacities of less than 25,000 bits have been constructed. Drums of this size generally have from 15 to 25 tracks and from 15 to 50 heads. In order to decrease access time, heads are sometimes located in sets around the periphery of the drum; a drum with 15 tracks may have 30 heads divided into two sets of 15 heads, each at a specific angular distance from the other. For very fast access time, there may be even more than two sets of heads.

Much larger drums can store up to 15 million bits and may have from 300 to 400 tracks. The larger drums are generally rotated much more slowly than small drums, and speeds vary from 120 RPM to 75,000 RPM. The access times obviously decrease as the drum speeds increase; however, there is another important factor--the packing density along the track. Most present-day drums have a packing density of from 100 to 300 bits per inch although, by maintaining the heads very close to the drum surface and rotating the drum slowly, packing densities in excess of 1000 bits per inch may be achieved.

**PARALLEL OPERATION.** It is possible to operate a drum in either a serial or parallel mode. For parallel operation, all the bits of a word may be written simultaneously and read in the same manner. If the basic computer word contains 40 bits, the drum might read from 41 tracks (one for timing) simultaneously, thus reading an entire computer word in 1-bit time. When the drum is "read from" and "written into" in parallel, a separate read and write amplifier is required for each track used.



Notice that the words in a parallel system may be located by means of a timing track. If each track contains 8192 bits, a 13-bit counter may be set to zero at the same position each time the drum revolves, and stepped by one each time a timing pulse appears. In this way, location 1096 will be the 1096th cell around the track from the zero location. If the address of the word to be read is located on a register, signals from the drum can be gated into the computer when the counter agrees with the register's content. In this way words may be located on the drum.

**SERIAL OPERATION.** A magnetic drum may also be operated in a serial mode. In this case only one track will be read from or written into at a given time. Since there are a number of tracks on each drum, the correct read-write head, as well as the location of the desired bits around the track, must be selected.

Each track is assigned a number; in addition, each track is divided into sectors, each sector containing one full computer word. For instance, if the basic computer word is 20 bits in length and 640 bits can be recorded around each track of the drum, each track would be divided into 32 sectors. Each sector would then contain one 20-bit computer word.

In order to specify the address of a word on a magnetic drum operated serially, both the track number and sector number must be given. Consider a drum with 32 tracks plus a timing track and 32 words (sectors) around each track. The address of a word on the drum in a binary machine will consist of 10 bits, 5 bits to specify the track and 5 bits the sector. When written as the address section of a computer instruction word, the address will contain 10 bits.

The five flip-flops containing the track number may be connected to a decoder matrix similar to the one used in the magnetic core memory, which will then select the correct read-write head.

Several techniques involving the timing tracks may be used to locate the selected sector. One technique involves the use of several timing tracks instead of one. One of the tracks contains a set of signals indicating the location of each bit around the tracks. The second track contains a set of pulses with a pulse at the beginning of each word time. The word time signals illustrated are 20 bits apart so the basic word would be 20 bits in length. In addition, the sector number of the next word around the drum is recorded around a third timing track. The computer reads sector numbers from this track, and when the number read agrees with the sector number in the address, the computer can then read the selected word from the next sector beginning with the next word time pulse.

**ADDRESSING.** Addressing the drum means nothing more than selecting the proper memory cell or cells at the correct time for reading or writing. Many different methods are used for addressing the drum.

### Magnetic Tape Memory

When we speak of tapes, we generally mean "magnetic tapes." Perforated tapes are used but they are less common. Another item that is brought to mind when tapes are discussed is the tape drive unit. Of course, the tapes are useless without the drive unit and vice versa. Actually, the magnetic tape is the medium where information is stored. The tape drive is the mechanism which writes information on a tape and reads it off. The process of storing information on a tape is called writing and the process of detecting stored information is called reading.

**TAPE CONSTRUCTION.** Magnetic tape is a thin flexible plastic strip with a uniform coating of ferrous oxide on one side. A typical tape is about 2,000 feet in length,

¼ to 2 inches wide, and has a word density of 40 or more computer words per inch. Information is stored on the tape in the form of a pattern of magnetic bits. In one form of tape recording, a magnetized spot or bit may represent a binary 1; a nonmagnetized spot may represent a binary 0. Another system may require that both 1's and 0's be expressed as magnetic bits. This is done by recording 1's with a north-south magnetic alignment and 0's with a south-north alignment. The number of magnetized areas across the width of the tape are called tracks. The number of recording tracks used is determined by the code that is used to represent numeric and alphabetic characters. Figure 1-19 illustrates a tape system using a six-bit character code. There are seven tracks across the tape. The seventh track is for maintaining synchronous operation between the tape drive and the computer.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	Track	
X	X	X	X	X								X				X	X	X	X	#1	
X					X	X	X	X				X	X			X	X	X		#2	
	X				X				X	X	X	X	X	X					X	#3	
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	#4 (sync)
		X				X			X				X	X	X	X			X	#5	
			X			X			X				X	X		X				#6	
				X			X			X				X			X			#7	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Characters	

RDA26-424

Figure 1-19. Magnetic Tape Six-Bit Alpha Code

**TAPE FORMAT.** Tape format may vary from system to system. For this reason we will discuss a typical tape format using a six-bit character code. If the computer's word length is 30 bits long, then a word would contain five characters (30/6 = 5). Words are written on the tape with no space between them. One or more words written together is a record and there is a recording gap between records (see figure 1-20). A group of records is called a file. Now let's apply this to what we have learned previously. A group of binary bits handled by the computer as a single unit is a computer word. In this case five characters on tape. A computer word can be either a data word or an instruction word. If instruction words were written together on tape, they would be considered a record. A series of instruction words written in logical order to solve a given problem is a program. Therefore, a record on tape could represent a program while a file could represent a group of programs.

**WRITING AND READING MAGNETIC TAPE.** Writing on magnetic tape occurs as the tape is moved across the magnetic gap of a recording or write head. The number of recording tracks in a write head is determined by the alphanumeric code used by the tape. Electrical pulses are sent through recording head coils at desired intervals. The oxide coating is magnetized by these pulses. These magnetized areas may be sensed as a 0 or a 1. To establish a given code, current will not flow through all the coils at the same time. These patterns represent the data sent from the computer.

The tape moves at high speeds across the write head. Typical speeds are 75 inches per second and 112.5 inches per second. The write pulses to the write heads are fast enough that the magnetized spots are almost the same as if the tape were still, for the period that the pulse is present.

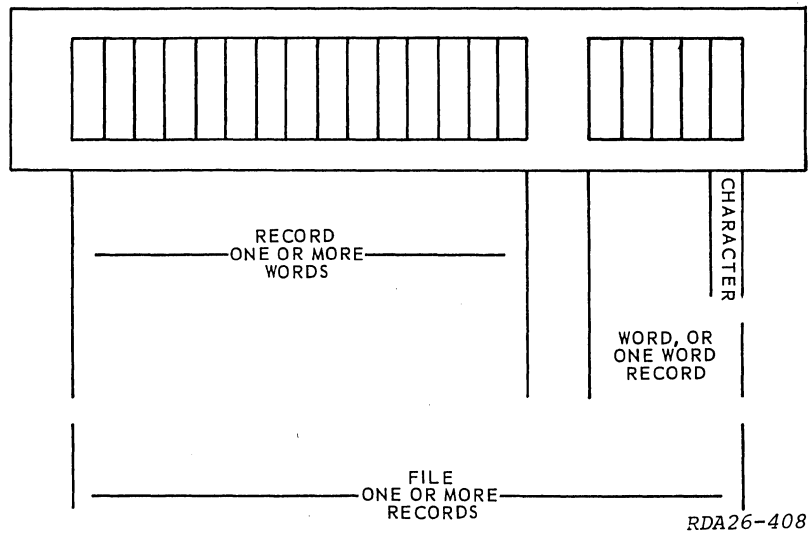


Figure 1-20. Tape Format

There are two types of read-write heads used in magnetic tape units. One type has a single gap for each channel. Both reading and writing occur at the same gap. The other newer types use two magnetic gaps for each channel. One gap is used for writing and the other is used for reading. Figure 1-21 shows both types of read-write heads.

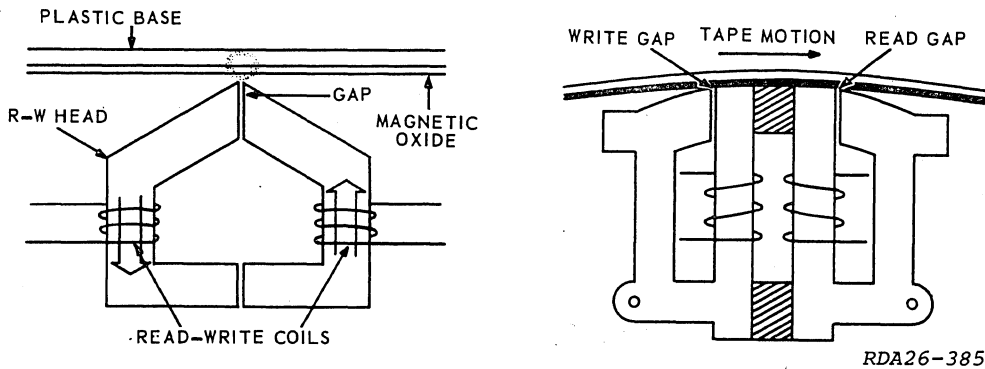


Figure 1-21. Read-Write Heads

The principles of reading and writing are the same for both type heads. However, the two-gap head has the advantage of being able to read the data shortly after it is written. This allows the data to be checked for errors. To read from the magnetic tape, the tape is passed over the read head. As the magnetized spot passes the gap, small electrical currents are generated in the coil of the read head. The pulses represent the data that is sent into the computer. Writing on the magnetic tape erases old information from the tape. Reading does not do this, so the tape can be read over and over.

Tapes are generally used as large-capacity, slow access memory storage. They may be considered input-output devices since they are used to initially load information into the computer and receive information from the computer.

#### REVIEW QUESTIONS 1-2

1. What is the difference between a storage device and a memory unit?
2. How many portions make up a memory cycle?
3. Which memory cycle is used as an input for new information?
4. What is the difference between a read and a write memory cycle?
5. Where will programs and data be stored?
6. Define volatility.
7. Why is the sense line used in a ferrite core memory?
8. What is the purpose of the inhibit line?
9. How may the number planes of a ferrite core memory be determined?
10. Which two registers are used when the computer communicates with memory?
11. Why are most core memory units called coincident current memories?
12. What is the purpose of the inhibit pulse?
13. How may a zero be written into a selected core?
14. Why are X and Y decoders used?
15. Why are line drivers used?
16. What is meant by permanence?
17. What is meant by the term "full select current"?

#### TERMINAL EQUIPMENT

Terminal equipment may be broadly defined as "all input and/or output devices." This broad definition is broken down into two categories: simple and complex. Equipment in the complex category is capable of both input and output; that is, it can send information into the memory element of a computer and receive information from the memory element. Equipment in the simple category may handle input or output but not both. This calls for another breakdown: simple input devices and simple output devices.

Simple input devices can put data into memory; simple output devices can receive data from memory. Terminal equipment may also be referred to as Peripheral equipment, or I/O equipment.

There are many types of terminal equipment. Let's list a few: card punch, card reader, line printer, magnetic tape units, magnetic drum units, various electric typewriters (Flexowriters), and communications buffers that connect to telephone lines. These are called data link buffers, and are complex pieces of terminal equipment. Any other devices which manufacturers produce that are capable of transferring digital information can become terminal equipment. Many units of terminal equipment have a compatibility package or control unit that arranges the data in proper word format or converts logic levels to insure correct data transfer between units made by different manufacturers.

Figure 1-22 lists some terminal devices and shows the classification of each. In many systems most of these devices can be controlled manually by the operator or automatically under program control by the computer. All of the equipment in figure 1-22 has dual capability except tapes and drums; these are generally controlled only by the computer.

NAME	SIMPLE INPUT	SIMPLE OUTPUT	COMPLEX
CARD READER	X		
CARD PUNCH		X	
LINE PRINTER		X	
PAPER TAPE READER	X		
PAPER TAPE PUNCH		X	
FLEXOWRITER			X
TELETYPEWRITER			X
MAG TAPE UNIT			X
MAG DRUM			X
DATA LINK BUFFER			X
CRT		X	

RDA26-425

Figure 1-22. Terminal Devices Classification

Time and space do not permit a detailed coverage of all terminal equipment. This discussion will acquaint you with a representative cross-section of terminal devices. The card reader, line printer, magnetic tape unit, and Flexowriter have been chosen for this purpose. We will take these in the order listed and discuss some of the leading particulars of each unit. We will also discuss a special input-output device used in air defense computers.

#### Card Reader

The purpose of the punch card reader is to provide a means of transferring data from punch cards to the computer system. The card reader has an input hopper for holding the cards to be read, a feed mechanism which sends the cards through the read station where the data bits are detected, and a stacker for holding the cards that have been read. The input hopper is located on the right side of figure 1-23. The cards are placed in the hopper face down, with column 1 toward the read station. The input hopper holds approximately 500 cards. The feed knife sends one card at a time into the read station. In the read station, there are 12 solar cells (each with an exciter lamp) which read the holes in the punch cards. In addition, there are two solar cells (with exciter lamps) for sensing the position of the card as it moves through the read station. After the

card is read, it drops into the stacker on the left side of the card reader. The stacker can hold approximately 500 cards. The card reader can read approximately 200 cards per minute. The lower part of the card reader cabinet has 21 storage bins for various punch card decks that are used frequently by programmers or maintenance personnel.

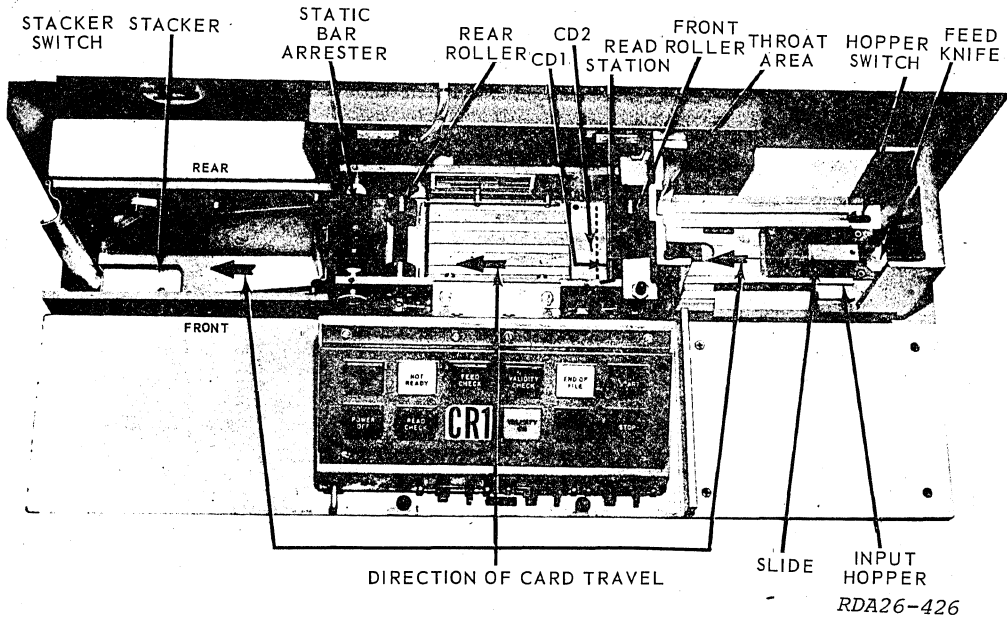


Figure 1-23. Card Reader

### Card Punch

Card punches may be operated directly by the computer to produce decks of punched cards under computer control, or they may be used by different personnel to produce punched cards that will be used to enter data into the computer.

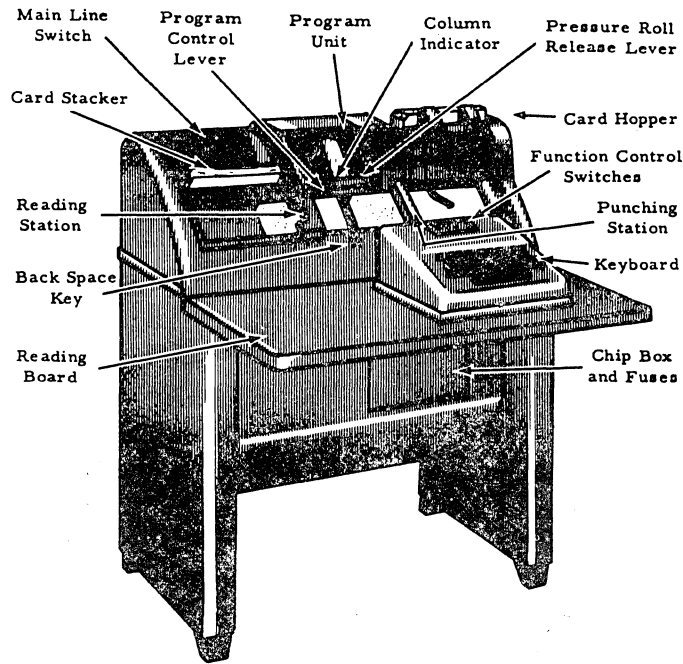
Figure 1-24 is a photograph of an IBM computer-controlled card punch used as an output device. A machine similar to this one makes the cards for your paychecks, medical appointments, and WAPS testing. It is approximately 36 inches high and 24 inches deep. The magazine is at the upper left and the stacker is the dark opening in the front center. Blank cards are placed in the magazine (hopper) and the machine is made ready by the operator.

The card punch, acting on programmed instructions from the computer, moves the cards from the hopper to the stacker. Between these two points there is a punch station with 90 punches. The card comes to rest 12 different times under the punch station. At each stop a row of the card is under the punches. The computer controls the punching action.

### Line Printer

A line printer is a "simplex" piece of terminal equipment that performs an output function only. The line printer records output information, usually in alphanumeric form. The term "line" indicates that the printer is capable of printing one entire line of characters simultaneously. The line printer does not print the line simultaneously; it ripples. Due to the printing mechanism being slightly slanted, it appears to be

simultaneous and the printing is done in a straight line across the paper. The speed of line printers varies from 100 lines per minute to 1,000 lines per minute. The number of characters (alpha or numeric) per line also varies, depending on the manufacturer. The line printer in figure 1-25 operates at approximately 600 lines per minute, printing 120 characters per line.

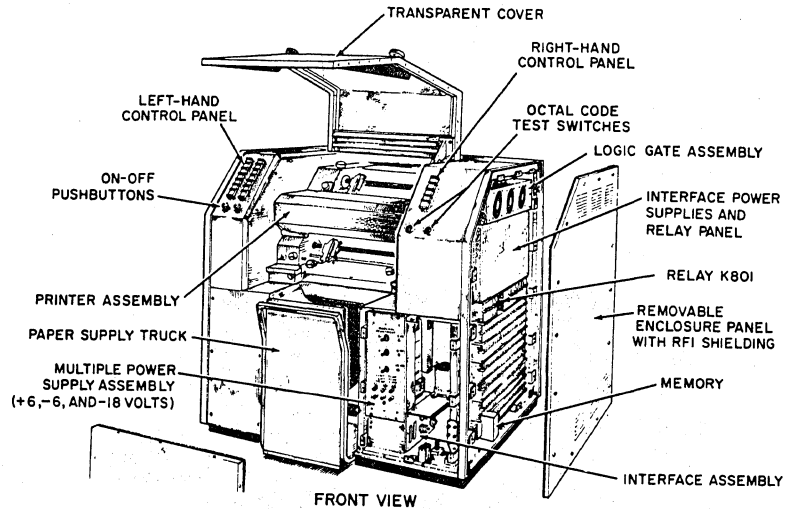


RDA26-427

Figure 1-24. Card Punch

This line printer has a print roll with fonts of characters (a font for each column of printout, 120, engraved in relief on its curved surface), and a row of solenoid actuated print hammers (a hammer for each column of printout, 120). In operation, the print roll turns continuously above the print hammers. When a required character turns into printing position, the corresponding hammer is actuated. The memory in the lower right corner of the line printer stores the data for a line of printout. One complete rotation of the print roll is required to print one line of characters. Each of the 120 hammers will be actuated only once per revolution of the print roll--when its desired character is in printing position. When the printing of one line is completed, then the

equipment's internal memory will request another line of output data from the memory element of the computer. The printer we have been discussing is the impact type printer. The non-impact printer is an electrochemical printer which uses a burn process to print. As the paper, which is a specially made paper with magnetic crystals embedded into it, passes over a stylus, electrical energy burns the outline of the characters into the paper. Printers of this type have obtained speeds of 36,000 lines per minute. While having the advantage of speed, they are very costly. The paper is expensive and they cannot make carbons.



RDA26-428

Figure 1-25. BUIC Line Printer

### Tape Drive Unit

Figure 1-26 is a typical magnetic tape drive unit. This unit is similar to a home tape recorder, but it records 7 to 9 channels instead of the 2-4 tracks used in home recorders. The tape drive unit controls the tape movement and provides the read and write operations. Since a magnetic tape may either receive or send information, it is a complex input-output terminal device. In many cases, a compatibility package or control unit (controller) is needed to synchronize timing and arrange information into the proper word formats.



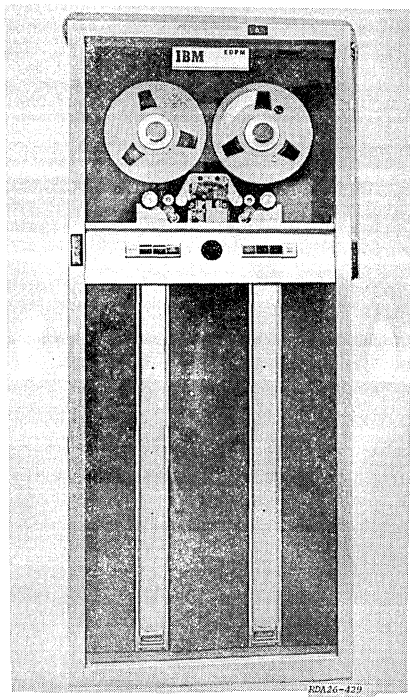
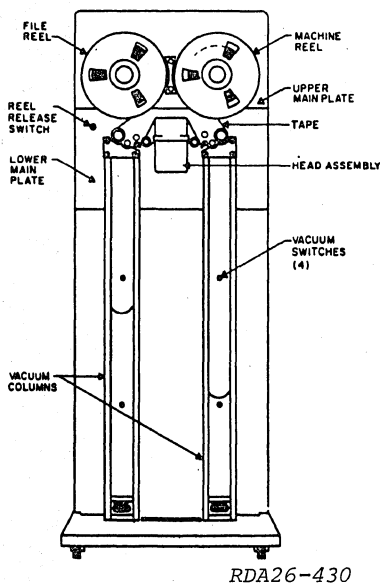
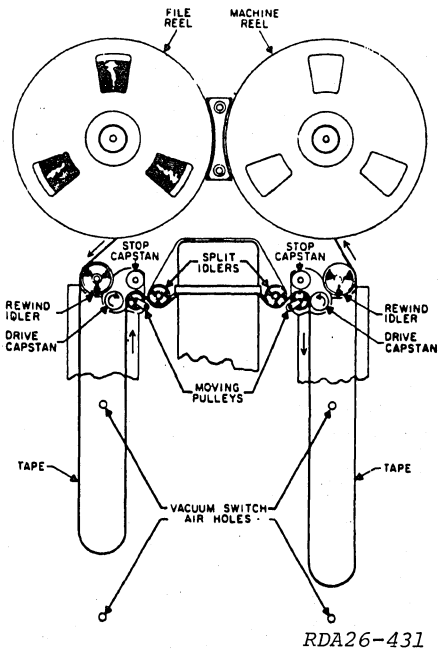


Figure 1-26. Tape Drive Unit



RDA26-430

Figure 1-27. Vital Parts of the Tape Drive Unit



RDA26-431

Figure 1-28. Tape Path

**MECHANICAL OPERATION.** Figure 1-27 gives some additional details on the mechanical functions of the tape drive. The "file reel" is one of the many reels of computer data stored in the computer room. The operator selects the desired reel from its storage file, places it in the tape drive unit, threads it through the vacuum columns and head assembly, and connects it to the machine reel. The "machine reel" is the take-up reel. The vacuum columns provide the proper slack in the tape to prevent damage during high-speed movement of the tape from reel to reel. The head assembly contains one read-write head for each of the channels on the tape. Figure 1-28 gives a detailed view of the tape path and the mechanical parts needed to move the tape through the read-write assembly.

### Flexowriter

The Flexowriter (a trademark of the Frieden Corporation) is a two-way (both input and output) complex terminal device which provides a means of communication between the operator and the computer system. The Flexowriter requires a control unit to provide a compatible interface between it and the computer input-output elements. The Flexowriter is an electro-mechanical typewriting device which provides an electrical means of communication with the computer, plus a hard copy on paper of all data exchanged.

The Flexowriter has internal circuits which convert the mechanical motion of keys into electrical signals for entry into the computer or convert electrical signals from the computer into the mechanical motion of keys to produce printed copy. In addition, it has a paper tape punch and paper tape reader on the left side to produce or read storage media. It is normally used by maintenance personnel to run maintenance programs and receive status reports from the computer. Figure 1-29 is a photograph of the Flexowriter modified for use in the BUIC System. The Teletypewriter is basically the same, it was simply made by a different manufacturer. You will become quite familiar with it through your future lab projects.

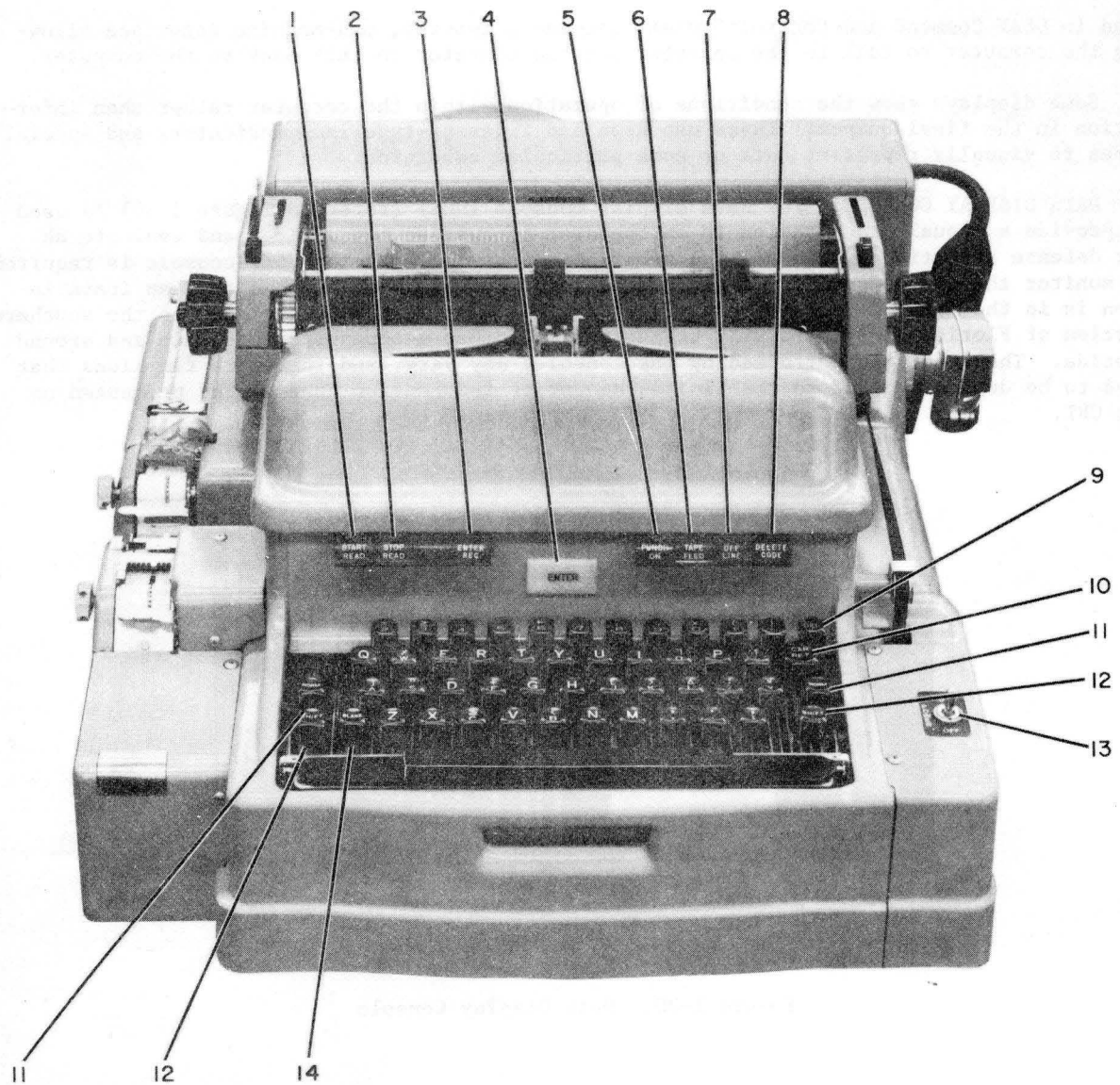
### Display Equipment

The display equipment of a computer system is part of both the input and output elements. Information may be transmitted from a digital computer and displayed visually in a direct readable form. For example, a computer used in air defense accepts air defense intelligence and evaluates or summarizes this intelligence for presentation by a display system. In addition, the display equipment provides a means for the operator to enter data into the computer.

The prime purpose of air defense is to provide flight path instructions for interceptor air weapons. To accomplish this mission effectively, a clear picture of the air situation must be available to personnel who are to direct retaliatory air defense.

A display system provides this picture. It presents relevant air surveillance intelligence on specially constructed cathode-ray tubes. Since the information from the central computer system is in a binary form, one important function of the display system is to convert such information into a form that can be easily interpreted by operating personnel. It does this by changing the binary information to visual intelligence that consists of letters, numerals, vectors, and special symbols in a prearranged format. These are visually displayed on a cathode-ray tube.

The operator of the display equipment can then use his equipment to accept, modify, or reject the information displayed. He may also enter new data into the system. Any action taken by the operator must be converted from switch actions, light pen responses, etc., back to binary form for entry into the computer. In this way, display equipment



24743/1

- |                      |                       |
|----------------------|-----------------------|
| 1. START READ switch | 8. DELETE CODE switch |
| 2. STOP READ switch  | 9. DELETE key lever   |
| 3. ENTER REQ switch  | 10. CAR RET key lever |
| 4. ENTER indicator   | 11. NORM key levers   |
| 5. PUNCH ON switch   | 12. SHIFT key levers  |
| 6. TAPE FEED switch  | 13. POWER switch      |
| 7. OFF LINE switch   | 14. BLANK key lever   |

RDA26-432

Figure 1-29. Control Panel and Keyboard Layout of "Flexowriter" Unit

used in USAF Command and Control Systems becomes a two-way, man-machine interface allowing the computer to talk to the operator and the operator to talk back to the computer.

Some displays show the conditions of operation within the computer rather than information in the final output. These use neon and filament-type lamp indicators and special tubes to visually represent data or some particular condition.

**DATA DISPLAY CONSOLE.** The data display console (DDC) (refer to figure 1-30) is used to provide a visual presentation to an operator so that he may monitor and evaluate an air defense situation within a given geographical area. More than one console is required to monitor the air defense for a given location. For example, suppose that an installation is in the area of Cape Kennedy, Florida. One console may be assigned to the southern portion of Florida and others will be assigned to other geographical areas in and around Florida. The functions performed by the consoles may vary, and there are functions that need to be duplicated due to the limited amount of information that can be presented on one CRT.

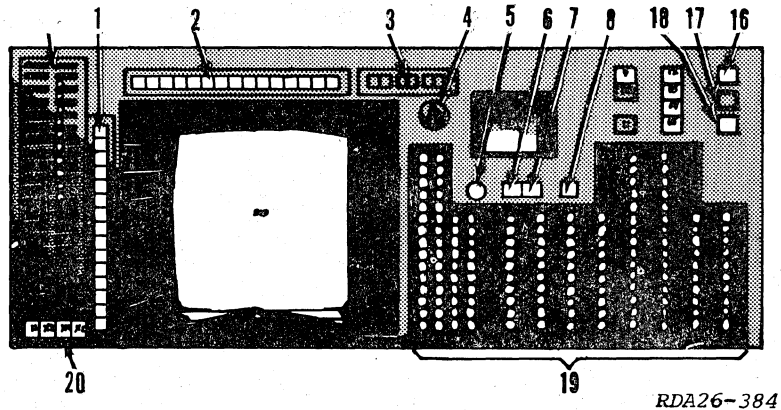


Figure 1-30. Data Display Console

A data display console can be divided into three basic sections. There are the situation display section, which consists of a large CRT (left side of figure 1-30); a tabular display section, which consists of a small CRT (center of the figure); and the manual intervention section, which is used to communicate with the computer system (lower right of figure). The situation display section is used to process and display such information as aircraft movement, boundaries, radar data, and air base locations. The tabular display section is used to process and provide the operator with the information he has requested, or provide tabulation of information that pertains to the function the operator is performing. The manual intervention section is used to interrupt the computer system when the operator requests more information or desires to insert information into the system.

All functions performed by the computer system are to process data so that it can be displayed at the data display consoles. The DDC can display two basic types of information--symbols and vectors. Symbols are numbers, letters, and special characters used to denote specific functions. Vectors are used to draw geographic boundaries and indicate the relative speed and direction of an aircraft track. Vectors will only be

displayed by the situation display CRT. All data displayed on the tabular (Tab) CRT will be symbols.

The information received at the console is either forced or selected. Forced data is received at a console and cannot be rejected by the operator. Forced data is received and displayed when new data is received at the computer and this data is needed to update the information being displayed at a console. The operator may request additional information from the computer when it is needed to interpret track messages or make a decision involving defensive action.

All data received at the data display console must first be processed by the computer system before it can be used by the console. Data received into the computer system is in the form of radar signals and the data console cannot process raw radar data as it is received from the radar site.

After proper processing, the received data is displayed in usable form on CRTs such as the typotron and charactron.

**DISPLAY TUBES.** The main component in most visual display equipment is the cathode-ray tube. The cathode-ray tube (CRT) operates on the same basic principles as the tubes commonly used for oscilloscopes or picture displays. Additional elements have been added to these CRTs to further control and direct the electron stream for character display. These are the charactron and the typotron tube.

**CHARACTRON.** Since the charactron tube presents the plan position maps of the air situation or portions of it, the display is referred to as a situation display.

Information pertaining to radar tracks, flight plans, geographical boundaries and locations, and weapons sites as presented on this tube is shown in figure 1-31 in the form of letters, numbers, special symbols, and vectors. The letters and numbers are assembled in short encoded messages that are displayed adjacent to certain points and targets to give identification and other descriptive data.

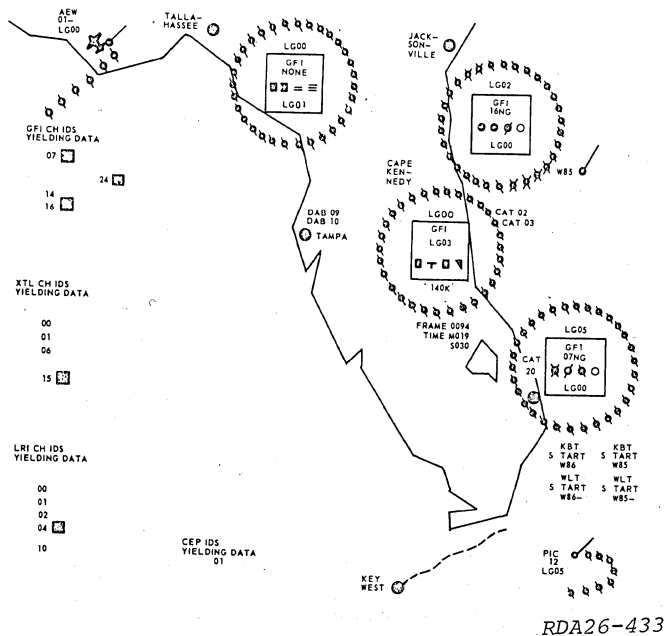


Figure 1-31. Typical Charactron Display

TYPOTRON. The typotron cathode-ray tube is similar in operation to the charactron tube. The typotron displays information, as shown in figure 1-32, that is too detailed for situation display. This type of display presents digital information in the form of charts and therefore is referred to as a digital display.

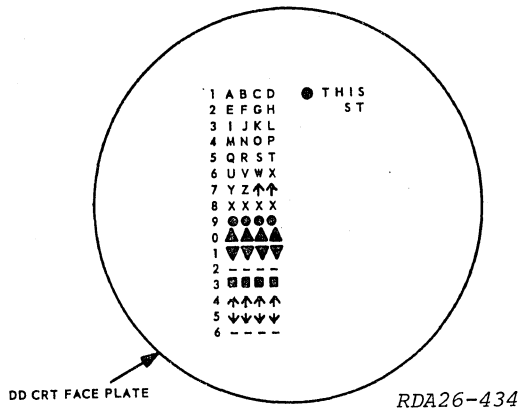


Figure 1-32. Typical Typotron Characters

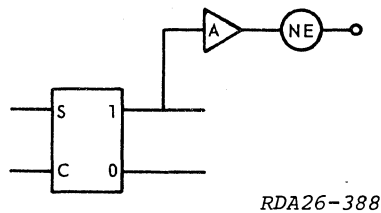


Figure 1-33. Neon Indicator

OUTPUT INDICATORS. In addition to displays intended for use by operators, most computer systems have smaller display areas intended for use by maintenance men. These displays may be gathered together in one central panel, or they may be scattered around the machine in strategic locations. Normally, these maintenance displays show the state of important flip-flops or registers in the machine. Occasionally, these displays will decode the count in significant counters or registers.

NEON INDICATORS. Neon indicators are used in computers to display directly the information contained in various registers and counters. As shown in figure 1-33, an amplifier is usually needed to increase the small voltage output of a flip-flop to a value sufficient to fire the neon lamp. The input to the neon amplifier is from the one side of the flip-flop. The amplifier will cause the neon indicator to light when the flip-flop is in the one state and to extinguish when the flip-flop is in the zero state.

INCANDESCENT INDICATORS. Many of the newer computer systems using integrated circuits have incandescent indicators rather than neon indicators. In most cases, the incandescent lamps are special-purpose bulbs that draw very little current and, therefore, do not require the use of an amplifier between the flip-flop and the indicator.

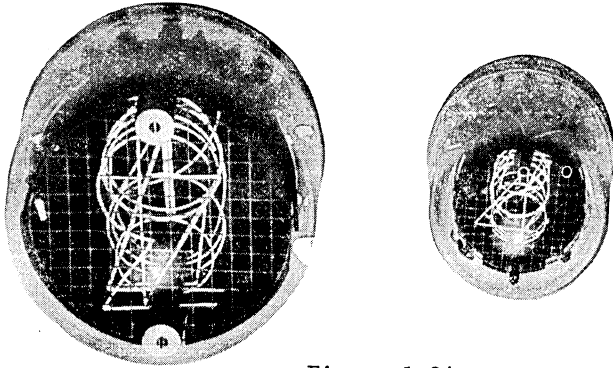


Figure 1-34. Nixie Tubes

**NIXIE TUBES.** The Nixie tube, a registered trademark of the Burroughs Corporation, is designed to display any one of several characters rather than simply indicate the state of a flip-flop.

The tube is gas filled and contains 10 cold cathodes and one common anode. Each of the cathodes is shaped to form a character, either alpha or numeric. When the correct voltage is applied between the anode and one of the cathodes, ionization of the gas occurs and causes a glow to surround the selected cathode. Because the cathode is shaped like a letter or number, we see a 6, a 2, an A, etc.

The use of Nixie tubes with cathodes in the form of letters or symbols is a design consideration and only for special purposes. For our purposes, we will consider that the cathodes are in the form of numbers only. The tubes are available in a variety of sizes ranging from  $\frac{1}{2}$  inch in diameter to several inches in diameter. Figure 1-34 shows 2 sizes of Nixie tubes.

Unlike the neon indicator which takes an output directly from a flip-flop, the Nixie tube requires a decoding system to select the desired cathode. Figure 1-35 shows the decode network for a count of five from a three stage up counter. The counter is made up of flip-flops A, B, and C. Gate 1 detects a count of five in the counter. The inverter amplifier provides a voltage of proper polarity to drive the cathode of the Nixie tube. Each cathode, except 8 and 9, would need a count detecting gate and driver like the one shown for cathode 5 to display all possible counts in the counter.

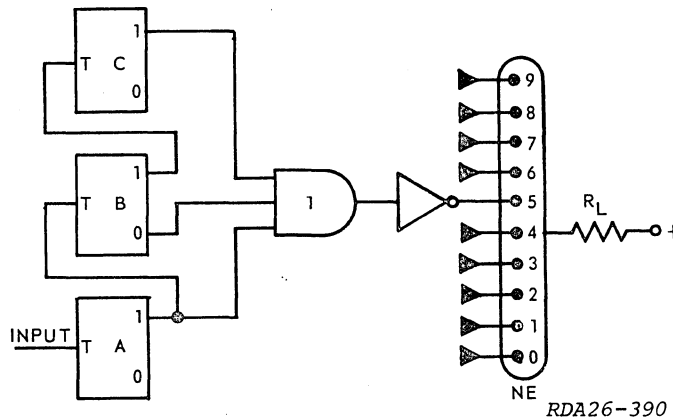


Figure 1-35. Nixie Tube Decoder Network

REVIEW QUESTIONS 1-3

1. What is the function of the input unit?
2. What is the function of the output unit?
3. What is meant by the term "ComputerWord"?
4. How does a computer distinguish an instruction word from a data word?
5. Why is a machine cycle broken down into acquisition time and execution time?
6. How many bits are used in the COM-TRAN TEN computer word?
  - a. Binary bits
  - b. Hexadecimal bits
7. What are the three positions on the main POWER switch for the teletypewriter?
  - a. \_\_\_\_\_
  - b. \_\_\_\_\_
  - c. \_\_\_\_\_
8. What position should the POWER switch be in to allow the use of the teletypewriter without accessing the COM-TRAN TEN?
9. What are the three positions of the tape reader switch?
  - a. \_\_\_\_\_
  - b. \_\_\_\_\_
  - c. \_\_\_\_\_
10. What position should the tape reader switch be in to allow reading of the tape?
11. List the four switches on the tape punch unit.
  - a. \_\_\_\_\_
  - b. \_\_\_\_\_
  - c. \_\_\_\_\_
  - d. \_\_\_\_\_
12. State the purpose of the "BACKSPACE" switch on the tape punch unit.
13. What type of code is used by the teletypewriter?
14. What is the purpose of the following keys on the keyboard of the teletypewriter?
  - a. CTRL (Control)
  - b. LINE FEED
  - c. RETURN



## COMPUTER OPERATION AND FAMILIARIZATION

In the first part of this chapter you learned the five basic blocks of all digital computers. Now, you are going to learn the block diagram of the COM-TRAN TEN. As you go through this material relate the COM-TRAN TEN block diagram to the basic block diagram. Look for the elements of the COM-TRAN TEN block diagram that make up the units of the basic block diagram. This will help you understand the flow of information through the COM-TRAN TEN block diagram.

The block diagram of the COM-TRAN TEN is foldout 1-1. First, look at the block diagram and note that there are Bus Lines. These are nothing more than conductors. They are called Bus Lines because information can be put on or taken off in many different places. There are four Buses in the COM-TRAN TEN; F-Bus, G-Bus, Y-Bus and Z-Bus. The data on the F, G, and Y Buses are in true form (High = 1 and Low = 0), while the data on the Z-Bus is in the one's complement form ( $\overline{\text{DATA}}$ ). Notice that the F and G Buses have only one route; F-Bus between the ALU and the A-Register, G-Bus between the Index Adder and the M-Register. The Y-Bus receives its data from the selector and transfers this data to many different registers. Only one register will receive this data during any of the transfers. Information can be transferred to the Z-Bus from six different registers (I-Reg, P-Reg, Memory, X-Reg, A-Reg, and Q-Reg); however, only one of these registers can transfer data to the Z-Bus at a time. This information can go to the Buffer Register, to the Two's Complementer, or the Memory Address Register. Buses are nothing more than conductors (such as wire) that carry information around inside the machine.

Look at the left-hand side of the block diagram. There you will see a block labeled Input Switch. These are sixteen switches that are used to manually input information into the computer. Figure 1-37 shows the control panel of the COM-TRAN TEN. The Input Switches are the ten switches labeled Input and sixteen Hexadecimal switches in a four by four configuration just below the Input Switch. The outputs of these switches go to the Input Register in binary form. This means your Hex inputs are changed to Binary as you input them.

The Input Register is a 10-bit storage register. It is used to hold the values which you input from the Input Switch. It can also be used to hold information that is being manually output from the computer. The Input Register can put information on the Z bus, or take it from the Y bus (Manual Output), or take it from the Input Switch. The Input Register can also transfer information through the Selector to the Y bus for manually loading the registers. Refer to figure 1-37; the switches labeled A, B, C, D, S, M, P, Q, and X are used to manually transfer the data in the Input Register to the other registers in the COM-TRAN TEN.

The Buffer Register is an 8-bit storage register, and is loaded from the Z bus. All information going to Memory or coming from Memory must go through the Buffer Register. Many of the data transfers through the computer go through the Buffer Register. Its output goes to the Selector, which decides which of three inputs will be placed on the Y bus. The three inputs are from the Buffer, the Two's Complementer, or the Input Register. The Selector output goes to the Y bus. The normal output of the Selector is the Buffer; however, the Input Register or the Two's Complementer will be selected when necessary.

The Two's Complementer is a circuit that does just what its name implies. It performs the two's complement on positive or negative numbers. It can also do a one's complement. The circuit used in the Two's Complementer is an adder. It receives data from the Z bus, and since the Z bus is in the one's complement form, a two's complement can be performed by adding a 1 and one's complement by adding a 0.

The next element of the block diagram is the Program Address Register. This is a 10-bit storage register used to hold the address of the next instruction to be done by

the computer. It can be loaded from the Input Register or the Memory Address Register. The output of the Program Address Register goes on the Z bus to the Memory Address Register.

The Op Code (S) Register is an 8-bit storage register. It holds the code for the instruction that is being performed. In other words, if you are doing an add, the code for an add will be in the Op Code Register. This will cause the computer to add two numbers together. The input to the Op Code Register comes from the Buffer or Input Register by way of the Y bus. The output goes to an Instruction Decoding Network.

The Decoder decides what instruction is to be performed, and it will generate the signals necessary to perform that instruction. If the ADD code is in the Op Code Register, the Decoder will decode this count that tells the computer to add. The output of the Decoder is sent to the proper places to make the computer add two registers together. There are many instructions: such as one to tell the computer to get a number out of Memory and put it into a register. Instructions are put together by a programmer in a logical, sequential way called a program. The program will do the job the programmer wanted done. A program is put into the computer's Memory. From Memory, the computer takes out instructions and decodes them. It then performs each instruction. The computer does this until it decodes an instruction that tells it to stop. The Program Address Register holds the Memory Address of the next instruction to be performed.

The Memory is used to store numbers, which can be decoded as instructions or data, but they must be in Hex form. If you want to add values of 26 and 73, the numbers and the add instruction must be in Memory. You also need a way to get the number when you want it. To do this, each Memory location has an address. If you were told to go get Joe Smith, but you didn't know where he was, you would have some difficulty finding him. If you were told he was in room 327 you could get him. Room 327 is like the Memory address. It tells the computer where to go to find the number you want. Note that the address and the number in that address are not necessarily the same, just as 327 is not the same thing as Joe Smith. The COM-TRAN TEN uses a random access IC chip memory made up of 1024(10) or 400(16) 8-bit words. Data input is from the Y bus. Data is output in complemented form to the Z bus. The IC chip is a volatile storage device. If power is removed from the chip, then the data in memory is no longer accurate. The COM-TRAN TEN is designed to maintain power to the IC chip memory after the power has been turned off by the power switch. However, if the machine is unplugged then the data in memory is no longer accurate.

The Memory Address "M" Register is a 10-bit register used to tell the computer what Memory location you are using. It can hold the address of instructions or data. Remember, instructions or data codes are just numbers.

The Index "X" Register is an 8-bit storage register. It is used to modify Memory addresses. A Memory address and the contents of the Index Register can be added together to give a new memory address. You will see how this is used when you get into Programming.

To add the Index to the Memory address there is an Index Adder. It takes the outputs of the Memory Address Register and the Index Register and adds the two together. The sum is placed in the Memory Address Register.

The Accumulator "A" Register is an 8-bit register. It can be shifted left or right and can transfer data in and out in parallel. The Accumulator and Buffer hold the numbers used for most arithmetic and logical operations. The results of most of these operations are then put back into the Accumulator. For example, during an add operation, the Buffer holds the addend and the Accumulator holds the augend, then the sum.

The Quotient "Q" Register is an 8-bit register used in some arithmetic operations. It holds the LSD of the product after a multiply operation, and the LSD of the dividend

and then the quotient of a divide operation. It also can be shifted left or right, and transfer data in or out in parallel.

The Arithmetic Logic Unit "ALU" takes its inputs from the Accumulator and the Y Bus. It performs all the arithmetic and some logical operations. This is where the additions and subtractions take place. The outputs of the ALU (Arithmetic Logic Unit) go to the F Bus which inputs into the Accumulator.

The Count Down Register is an 8-bit register that is used to help control the computer. It is used to help control teletype input and output to the computer, to count the number of shifts for shifting operations, and to control the number of instructions jumped in skip instructions.

The clock is basically a free-running multivibrator that produces 1-microsecond pulses every 2 microseconds.

PRT = 2  $\mu$ sec            PW = 1  $\mu$ sec            PRF = 500 KHz

The pulses are used to toggle two D-Type flip-flops which are interconnected to form a grey-code counter. The outputs of this counter are decoded as clock pulses - CP1, CP2, CP3, and ENABLE. The clock pulses control minor timing of the COM-TRAN TEN.

The D-Reg is a 4-bit up-counter, incremented by CP3, which can count in binary from 0 through 15<sub>(10)</sub>. The output of the D-Reg is decoded in conjunction with the E FF to produce distributor pulses DPO - DP15 and DPA0 - DPA15. These distributor pulses control the major timing of the COM-TRAN TEN. The E FF determines the phase (acquisition or execution) in which the COM-TRAN TEN is operating. If a word is read from memory during the acquisition time it is part of an instruction word. If it is read during execution time it is data.

You have read what each element of the block diagram does. Now we will try to put them all together into a working machine.

To write instructions and data into the computer the information goes to the Buffer Register. It can come from the Input Register or the Teletypewriter. From the Buffer it goes into the Memory.

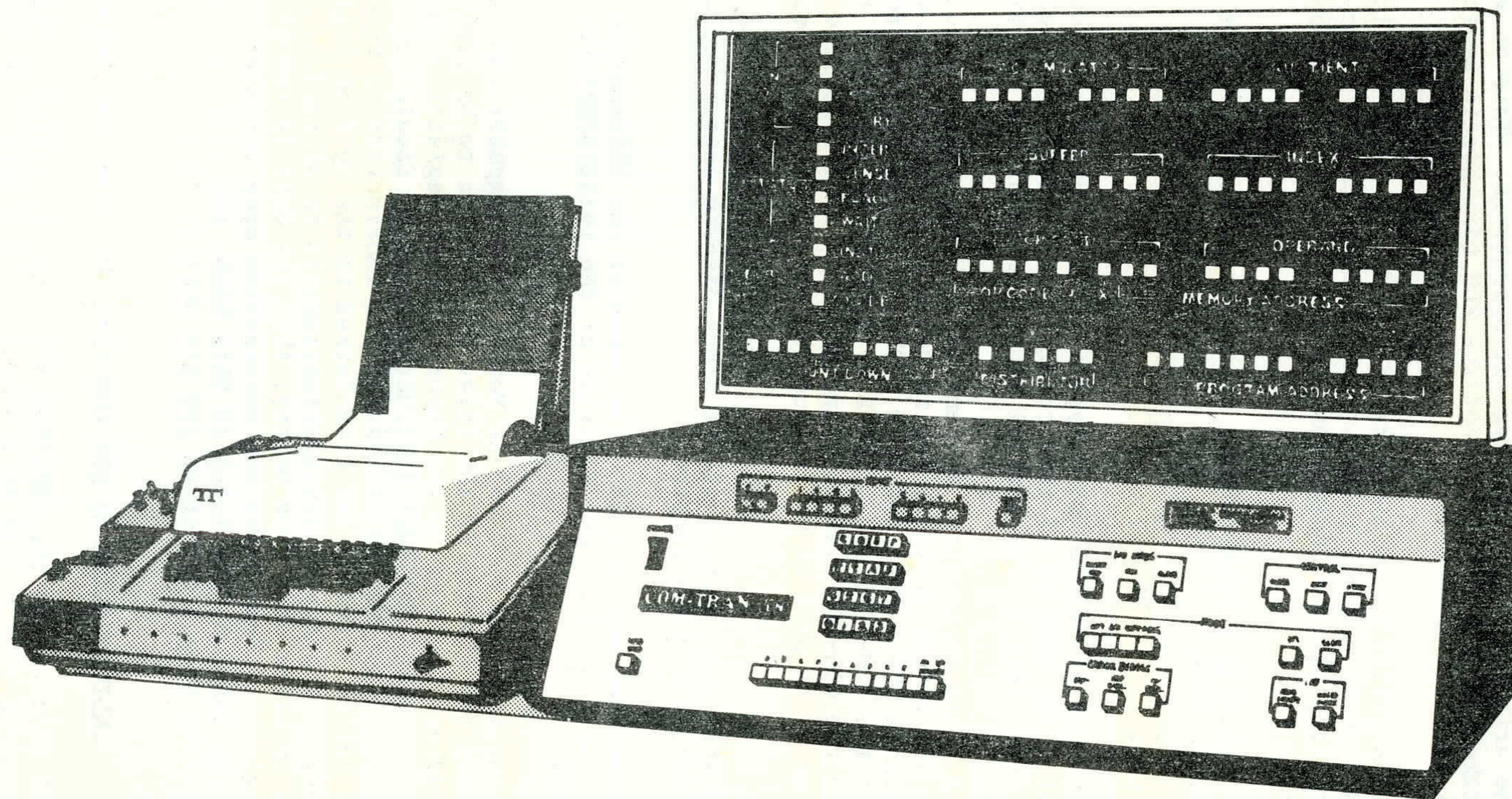
Once the instructions and data are in Memory, the computer can start working on the problem. To do this, the address of the first instruction to be performed is transferred from the Program Address Register to the Memory Address Register. Remember the Program Address Register tells where the next instruction word is located and the Memory Address Register is used to tell the Memory location. The instruction is then brought out of Memory through the Buffer Register and transferred to the Op Code Register on the Y bus. Once the Op Code Register receives the instruction, it is decoded by the Control Section. If the instruction requires data from Memory, then the data is brought out of the Memory into the Buffer Register. From there the instruction does whatever it is supposed to do. As you go through the instructions later in this book, you will need to refer to the block diagram to follow each instruction and see what it does.

#### INDIVIDUALISM OF THE COM-TRAN TEN COMPUTER SYSTEM

In figure 1-36 is pictured the COM-TRAN TEN computer system. This is the digital computer that is used in this manual as a vehicle of learning. Do not be misled by its size. It can do all the things a large scale computer can do. (It is a small computer because its word length is short and its memory size small.)

All computers have to be programmed...taught what to do and where to locate information (called data). As soon as the programming is done, high-speed computing gets





COM-TRAN TEN Educational Computer System  
Figure 1-36.



underway. The COM-TRAN TEN computer, though, does more than compute. You start out teaching the machine, as always. When computing starts, the CT-TEN education system takes on a new personality...it becomes the teacher. It reveals the exciting world of numeration, number manipulation, computer design, and logic. The activity it offers best is logic...how do we think when we solve a problem.

The INPUT section of the CT-TEN system is labelled INPUT on the CONTROL PANEL. We will also make use of a teletypewriter on which we can type into the computer's memory or we can prepare tapes and read information in this way.

The OUTPUT section of the CT-TEN computer is the display primarily. We can also direct the computer to type (or print) out answers on the teletype.

The MEMORY section of the CT-TEN computer consists of 1024 individual storage cells, each of which is addressable. Each cell is eight bits in length. This means that in any given cell we may store a number up to +127 and lower to and including -128. Instruction words are two cells in size. We refer to the size of a memory cell as a word.

The ARITHMETIC section of the CT-TEN computer can also be called accurately the LOGIC unit. There are several parts of the display that are used in the arithmetic or logic of the computer. Take note of the ACCUMULATOR, QUOTIENT, BUFFER, and INDEX. Besides the four operations of ARITHMETIC, LOGIC operations can be performed. The CT-TEN computer also records the nature of certain registers as greater than, equal to, or less than zero. Comparing numbers and acting on the results is an integral part of the nature of computers.

The CONTROL section of the CT-TEN computer can be studied in much detail through the DISTRIBUTOR MODE (see CONTROL PANEL). It takes an interaction of all the registers you see on the display for CONTROL to do its job. At present we will appreciate the work of the CONTROL portion of the computer, rather than study and understand it in detail.

#### OPERATING THE COMPUTER

The COM-TRAN TEN computer can be operated from any standard 115-volt 48-62 (220 VAC optional) Hertz AC power source. The power plug is designed for a three-prong receptacle. Check that the power plug is secure in the outlet.

The POWER switch on the CT-TEN computer is located on the left of the control panel. Press it. When lit, power is ON. Figure 1-37.

#### Controls and Switches

##### POWER

Press this switch to turn computer ON, if off. Press this same switch to turn computer OFF, if on. When switch is lit, power to the computer and its memory is on.

##### LAMP TEST

Turning ON the LAMP TEST results in all the light indicators on the display and the ten INPUT bit indicators to be illuminated.

##### MODE: DIST

DISTRIBUTOR MODE permits the operator to step through an instruction by stopping after each clock pulse. When the computer is in this mode, one clock pulse is generated each time the START switch is pressed.

A/E

ACQUISITION/EXECUTION MODE permits the operator to step through a program by stopping twice for each instruction. First the computer acquires an instruction stored in memory and stops while the contents are displayed in the OP CODE and MEMORY ADDRESS registers. Then the computer carries out the instruction and stops to display the results in the registers. The START switch must be pressed to go on to the next phase.

INST

*SINGLE STEP*

INSTRUCTION MODE allows the operator to step through a program by stopping after each instruction is carried out. The START must be pressed each time to go on to the next instruction.

PROG

PROGRAM MODE is the mode of execution for the computer to carry out instructions at its own fast speed of operation. In this mode the computer stops only if an instruction commands it.

RPT

REPEAT MODE permits the operator to have a certain phase of executing or acquiring an instruction to be repeated over again, in order that voltage levels may be constant and the registers may display the action repeatedly as a constant situation.

SENSE

SENSE MODE can be pressed at any time. If a program contains an instruction to test the SENSE switch, then one of two alternate set of instructions will be carried out.

IF A MODE SWITCH LIGHT IS ON, THAT MODE SWITCH HAS BEEN SELECTED. Press again to turn OFF.

ERROR BYPASS:

INST

The computer will stop when it encounters certain operational errors. By pressing the INSTRUCTION ERROR BYPASS the computer will NOT stop when it encounters a code that is not an instructional code.

ADD OVFL

By pressing this ADD OVERFLOW ERROR BYPASS the computer will NOT stop when the result of computation in the ACCUMULATOR register exceeds +127 or is less than -128. If this indicator is not lit, the computer does STOP when the result of addition or subtraction exceeds +127 or is less than -128.

DIV OVFL

By pressing this DIVIDE OVERFLOW ERROR BYPASS, the computer is instructed NOT to stop when the result of division is greater than the QUOTIENT register can hold. If this indicator is not lit, the computer does STOP when the QUOTIENT register does not hold the answer in division.

I/O:

Pressing one of these two switches sets up the computer to execute the I/O instruction selected. The countdown register is set to a count of FF<sub>16</sub> and the distributor register to a Hex 10 code. The operation (OP) code is set up for the proper I/O instruction.

READ INTRPT            If pressed, sets the OP code for read until interrupt operation.

WRITE BLOCK            When pressed, sets up the OP code for write data block operation.

I/O MODE:

REXMT OFF              Pressing this switch turns the teletype printer off. It will not print data from the paper tape reader or keyboard.

HEX                    Pressing this switch causes information read into or out of memory to be considered in hexadecimal form. When read into memory a colon (:) separates data words. When read out of memory, a colon (:) is supplied by the system to separate data words. The teletype remains in the I/O mode selected until a change is made on tape, on the teletype control, or on the control panel.

ALPHA                  Pressing this switch causes data to be read into or out of memory as alphabetical letters, characters, or decimal digits.

CONTROL:

CLEAR                  By pressing this switch all the register lights are cleared and become zero.

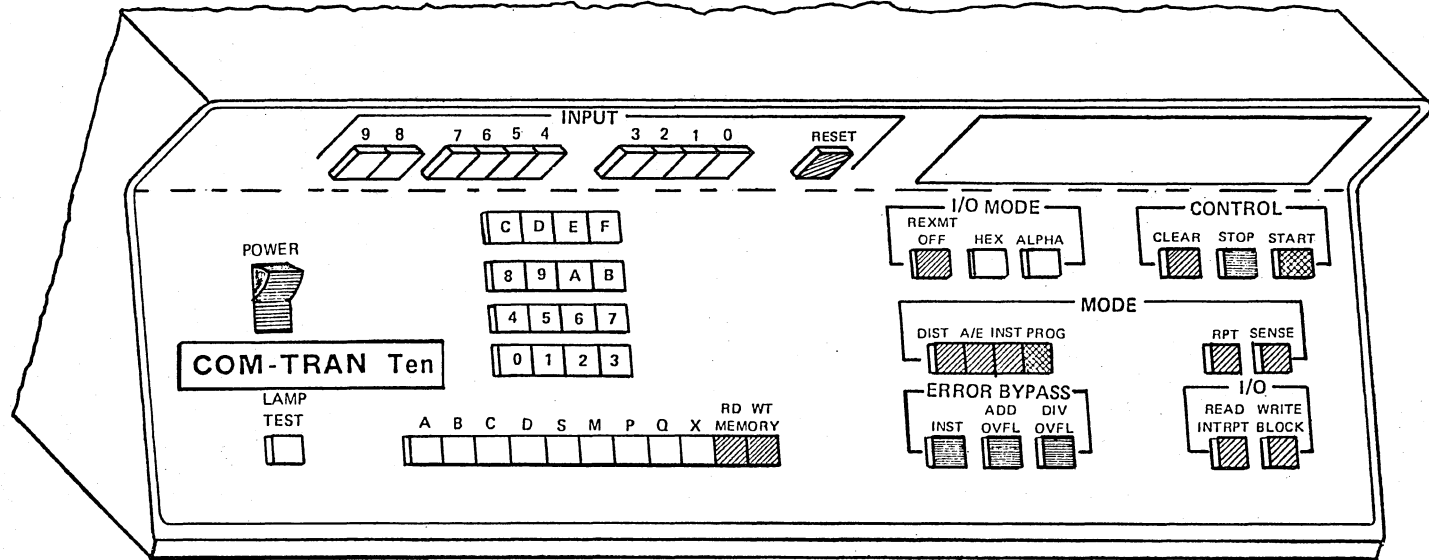
STOP                    By pressing this switch the computer will stop operating after the manual stop signal is synchronized by the computer's clock. This is the correct method of manually stopping the computer while it is operating.

START                  When pressed, the manual start signal starts the computer clock and computer operations. The computer will operate as directed by the MODE and MODE REPEAT switches until a stop is executed.

INPUT

The INPUT register consists of ten switches that can be set (lit) by pushing each individually or by using the keys numbered from 0 through F. By pressing one of the keys arranged in four rows of four keys each (hexadecimal numbers), the binary form of that hexadecimal digit will be entered on the right set of four INPUT switches; when the next hexadecimal key is pressed, the first four lights are transferred to the left before the new digit is entered in the least significant set of four lights. The RESET switch clears the ten INPUT switches. Pressing one of the following switches will transfer the contents of the INPUT REGISTER to the selected register.

Switches	
A	ACCUMULATOR register
B	BUFFER register
C	COUNTDOWN register
D	DISTRIBUTOR register
S	OP CODE register
M	MEMORY ADDRESS register (8 least significant bits are also called OPERAND)
P	PROGRAM ADDRESS register
Q	QUOTIENT register
X	INDEX register



RDA26-435

Figure 1-37. Control Panel for COM-TRAN TEN Computer



RD MEMORY	READ MEMORY switch sets up the computer to output data from memory cells indicated in the memory address register.
WT MEMORY	WRITE MEMORY switch sets up the computer to accept data through the BUFFER register and store it into memory cells beginning with the one addressed in the memory address.

### Computer Registers and Display Panel

The registers of a computer are temporary storage devices. Each is made up of a series of bistable circuits or flip-flops. Each such circuit has the ability to represent either the ZERO or the ONE state, and when connected in series they act upon one another to interpret data. Certain registers hold results before and after computation, others are a clearinghouse for the computer, still others act as the control within the computer.

The more familiar a programmer becomes with the operational characteristics of the various registers, the greater will be his ability to utilize them to best advantage when writing and running programs.

A listing of the various registers of the COM-TRAN TEN computer follows, accompanied by a description of the functional aspects of each. Figure 1-38.

#### Register and Abbreviation

#### Description and Function

ACCUMULATOR  
A

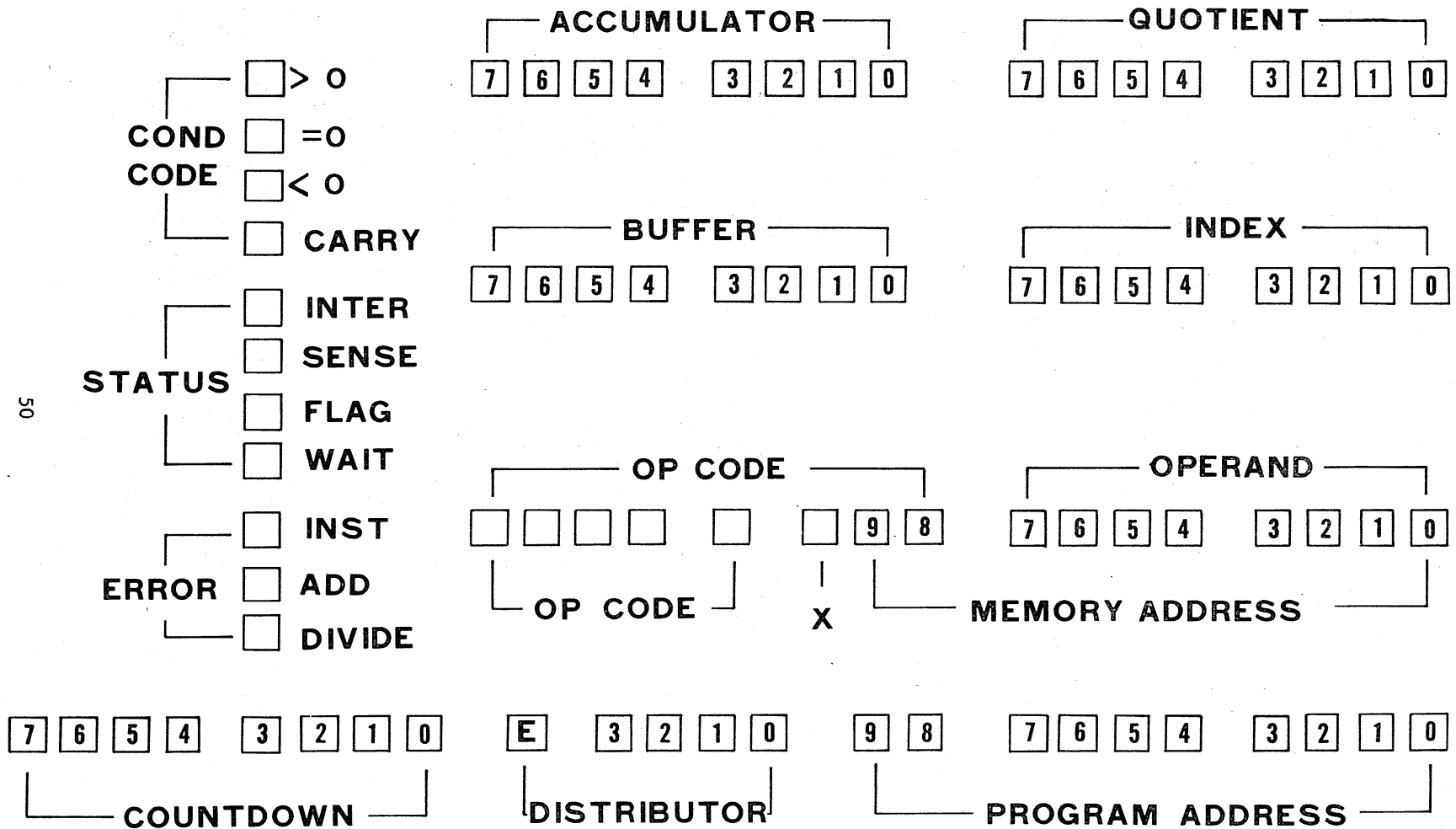
An 8-bit register; seven bits represent magnitude in most arithmetic operations and the leftmost bit represents a sign (0 is positive and 1 is negative). The register bits are numbered according to their binary integer value, i.e., A<sub>5</sub> is the 2<sup>5</sup> (32) position in the A register. A<sub>7</sub> is the sign position. The A register is used to hold:

- (1) the augend and then the sum in arithmetic 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.
- (6) the augend and then the LOGICAL sum in addition.
- (7) the first word of data and the LOGICAL result in EXCLUSIVE OR or in INCLUSIVE OR operations.
- (8) the 8-bit status word as a result of the SENSE STATUS instruction.

BUFFER  
B

An 8-bit register used to communicate with every section of the computer. It performs the following functions:

- (1) holds the addend in arithmetic addition, if this number has the same sign as the augend in the ACCUMULATOR.
- (2) holds the two's complement of the addend in arithmetic addition, if this is opposite in sign from the augend which is in the ACCUMULATOR.
- (3) holds the subtrahend in subtraction, if this number is opposite in sign from the minuend in the ACCUMULATOR.



RDA26-436

Figure 1-38. Display of CT-TEN COMPUTER

- (4) holds the two's complement of the subtrahend, if this number has the same sign as the minuend in the ACCUMULATOR.
- (5) holds the multiplicand in multiplication and the divisor in division.
- (6) holds the constant in LOGICAL operations.
- (7) acts as a data register for transferring in and out of memory.

COUNTDOWN  
C

An 8-bit decreasing counter. It holds a count of the number of process steps in multiplication and division, the number of shifts in shift instructions, the number of instructions to be skipped in skip instructions, and the number of words to be handled in Input/Output instructions. The C register flip-flops are designated according to their binary integer value, i.e., C6 is the 2<sup>6</sup> position.

DISTRIBUTOR  
D

A 5-bit increasing counter that establishes the sequence of clock pulses. Depending on the state of the E flip-flop and the decoded contents of the OP CODE register, this register controls the logic functions to be performed on initiation of each timing pulse.

OP CODE  
S

A 5- to 8-bit register that holds the operation code of the instruction being performed. In practice the most significant five bits are the instruction code. The lower three bits are added to the instruction code to allow for addressing beyond cell 'FF.' Thus three more levels of memory cells can be used. Bit S2 is used as an index.

MEMORY  
ADDRESS  
M

A 10-bit register used to locate any one of the 1024 words in the memory. The flip-flops are numbered according to their binary integer value.

PROGRAM  
ADDRESS  
P

A 10-bit register that determines the location of the instruction to be executed. It is increased during the ACQUISITION phase unless the MODE REPEAT switch is on. An instruction consists of two consecutive words: one is the OPERATION CODE and the other is the accompanying MEMORY ADDRESS (for memory referenced instructions) or OPERAND (for immediate instructions). The first instruction of any program is usually stored at an even address.

QUOTIENT  
Q

An 8-bit register; seven bits represent magnitude and the most significant bit represents the sign of the number. The bits are numbered and handled like the bits in the ACCUMULATOR. The Q register is used to hold:

- (1) the least significant bits of a product in multiplication (see AQ register).
- (2) the least significant bits of a dividend and then the quotient in division. Q<sub>7</sub> is the sign bit.

AQ

A 16-bit register consisting of the ACCUMULATOR-QUOTIENT registers. Fifteen bits hold the product after multiplication with A<sub>7</sub> being the sign of the product. The fifteen bits hold the dividend before division with A<sub>7</sub> being the sign of the dividend. This 16-bit register is affected by the ARITHMETIC shift instructions.

INDEX  
X                   The X register is an 8-bit register which is used for indexing the operand address. If S2, the index bit of an instruction OP code is set, the contents of the INDEX register will be added to the contents of the Memory Address Register during the Acquisition phase of the instruction.

COND CODE           A 4-bit register arranged in vertical position. Following each instruction this register records the nature of the register affected. After an instruction involving the ACCUMULATOR, the COND CODE records the nature of the ACCUMULATOR. After division the COND CODE records the nature of the QUOTIENT. In subtraction the two numbers are regarded as 8-bit numbers; if the subtrahend (in the BUFFER) is greater than the minuend (in the ACCUMULATOR) the CARRY bit is set.

STATUS              A 4-bit register arranged in vertical position. The bits can be set by means of an instruction. The SENSE STATUS bit can also be set from the control panel.

ERROR               A 3-bit register that is arranged in vertical position. These individual bits are set whenever the error occurs in the course of running a program. Operation will not stop, if the appropriate switch has been set to bypass the error (switch is located on control panel).

#### Hexadecimal Review

Turn ON the power switch of the computer.  
Press the CLEAR switch.  
A/E mode.  
Press RD MEMORY.  
Press the START.

At this point watch the MEMORY ADDRESS register. Press the START again. Notice that the M<sub>0</sub> light went ON. Press the START again. Record your results. (To shorten the recording process, just put down the results in the four least significant lights.)

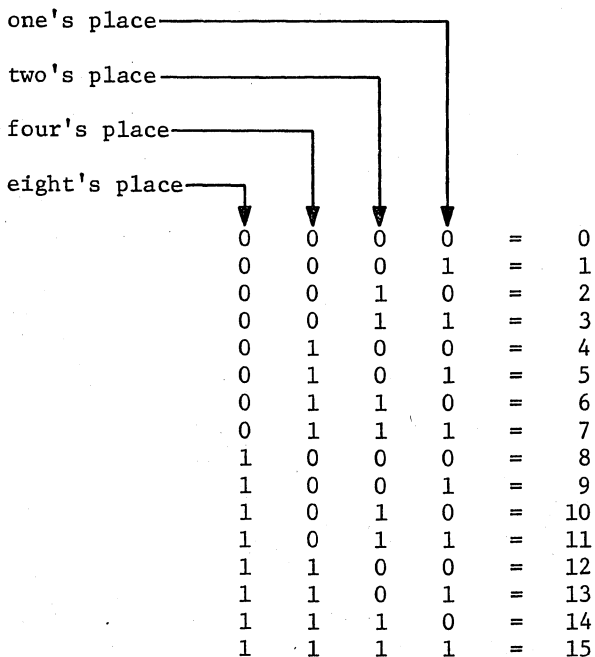
OFF	OFF	OFF	OFF
OFF	OFF	OFF	ON
OFF	OFF	ON	OFF
OFF	OFF	ON	ON
OFF	ON	OFF	OFF
OFF	ON	OFF	ON
OFF	ON	ON	OFF
OFF	ON	ON	ON
ON	OFF	OFF	OFF
ON	OFF	OFF	ON
ON	OFF	ON	OFF
ON	OFF	ON	ON
ON	ON	OFF	OFF
ON	ON	OFF	ON
ON	ON	ON	OFF
ON	ON	ON	ON

By pressing the START, the computer increased what was in the MEMORY ADDRESS by one. The computer was counting by one's in the M register. Since the computer has only two ways of showing information in any one place (not ten different ways), the computer uses a different number system. We call this system the binary numeration system. (Binary means two symbols.)

Each light is EITHER ON or OFF. When we translate these states into number symbols, we choose to let OFF be represented by "0" and ON to be represented by "1." Now the states above are represented as:

0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

You have just counted in binary from zero through fifteen. Look back over the results. We have a number system with only two symbols. The place value of each "0" or "1" becomes more significant than one may have realized. Notice that two is represented as: 0 0 1 0. Four was in this form: 0 1 0 0. Eight had this form: 1 0 0 0. A "1" in each of these special positions gives a value of 2, 4, or 8 to be added into the value of the number under consideration.



We stopped at fifteen, since we had considered only four places. Fifteen is the largest number we can represent with four places, since fifteen equals eight plus four plus two plus one. We call these places BINARY digiTs or BITS, for short.

Notice that each number from zero through fifteen had a special or unique way of representing that number. NO OTHER NUMBER could be represented in binary form as 12 is, for example. Only 12 equals 8 plus 4 and so is: 1 1 0 0. If we work with numbers greater than fifteen, we will need more places. Each place will be two times greater than the one to its right. If we consider two more places, we then have the sixteen's place and the thirty-two's place.

To emphasize and appreciate this idea that every number has a unique representation, use the BINARY SELECTION set of six charts (figure 1-39). These are labelled A, B, C, D, E, and F. Study them a few moments before turning to the following page.

Ask someone to think of a number from 0 through 63. Without telling you the number he will identify it with six YES-NO responses. He will look at each chart in turn from A through F. He will say YES, if the number is on the chart; NO if the number is not on the chart.

Suppose his responses are:

A	B	C	D	E	F
YES	YES	NO	YES	YES	NO

That could only be the number 54.

Notice the number that appears first on each chart: A has 32, B has 16, C has 8, D has 4, E has 2, and F has 1. Notice that these numbers are all a power of two (two multiplied by itself a different number of times). Converting the YES responses to 1's and the NOs to 0's, the binary form of 54 is 1 1 0 1 1 0. This means that 54 is  $32 + 16 + 4 + 2$ .

One more sample: I am thinking of a number...my responses are:

A	B	C	D	E	F
YES	NO	YES	NO	YES	YES

So that must be  $32 + 8 + 2 + 1$  which is 43.

As you work with the computer you will develop more familiarity with this system and a closely related one: hexadecimal. The conversion table on a following page will help you, so you need not master this material now.

However, if you wish to convert a number in decimal form to its equivalent binary form you would divide by two successively. Assume, for instance, that we wish to find the binary number for 53. The process is:

$53/2 = 26$ ,	with a remainder of: 1
$26/2 = 13$ ,	with a remainder of: 0
$13/2 = 6$ ,	with a remainder of: 1
$6/2 = 3$ ,	with a remainder of: 0
$3/2 = 1$ ,	with a remainder of: 1
$1/2 = 0$ ,	with a remainder of: 1

The column of remainders, when set down with the top digit at the right yields 110101 which is the binary equivalent of 53 ( $32 + 16 + 4 + 1$ ).

A		
	42	53
	43	54
32	44	55
33	45	56
34	46	57
35	47	58
36	48	59
37	49	60
38	50	61
39	51	62
40	52	63
41		

B		
16	26	
17	27	53
18	28	54
19	29	55
20	30	56
21	31	57
22	48	58
23	49	59
24	50	60
25	51	61
	52	62
		63

C		
	26	45
8	27	46
9	28	47
10	29	56
11	30	57
12	31	58
13	40	59
14	41	60
15	42	61
24	43	62
25	44	63

D		
	22	45
	23	46
4	28	47
5	29	52
6	30	53
7	31	54
12	36	55
13	37	60
14	38	61
15	39	62
20	44	63
21		

E		
2	22	
3	23	43
6	26	46
7	27	47
10	30	50
11	31	51
14	34	54
15	35	55
18	38	58
19	39	59
	42	62
		63

F		
	21	43
1	23	45
3	25	47
5	27	49
7	29	51
9	31	53
11	33	55
13	35	57
15	37	59
17	39	61
19	41	63

RDA26-437

Figure 1-39. Binary Selection

TABLE 1-1

DECIMAL-BINARY-HEXADECIMAL  
CONVERSION TABLE

HEXADECIMAL	BINARY	DECIMAL
0	00000000	00
1	00000001	01
2	00000010	02
3	00000011	03
4	00000100	04
5	00000101	05
6	00000110	06
7	00000111	07
8	00001000	08
9	00001001	09
10	00001010	0A
11	00001011	0B
12	00001100	0C
13	00001101	0D
14	00001110	0E
15	00001111	0F
16	00010000	10
17	00010001	11
18	00010010	12
19	00010011	13
20	00010100	14
21	00010101	15
22	00010110	16
23	00010111	17
24	00011000	18
25	00011001	19
26	00011010	1A
27	00011011	1B
28	00011100	1C
29	00011101	1D
30	00011110	1E
31	00011111	1F
32	00100000	20
33	00100001	21
34	00100010	22
35	00100011	23
36	00100100	24
37	00100101	25
38	00100110	26
39	00100111	27
40	00101000	28
41	00101001	29
42	00101010	2A
43	00101011	2B
44	00101100	2C

HEXADECIMAL	BINARY	DECIMAL
45	00101101	2D
46	00101110	2E
47	00101111	2F
48	00110000	30
49	00110001	31
50	00110010	32
51	00110011	33
52	00110100	34
53	00110101	35
54	00110110	36
55	00110111	37
56	00111000	38
57	00111001	39
58	00111010	3A
59	00111011	3B
60	00111100	3C
61	00111101	3D
62	00111110	3E
63	00111111	3F
64	01000000	40
65	01000001	41
66	01000010	42
67	01000011	43
68	01000100	44
69	01000101	45
70	01000110	46
71	01000111	47
72	01001000	48
73	01001001	49
74	01001010	4A
75	01001011	4B
76	01001100	4C
77	01001101	4D
78	01001110	4E
79	01001111	4F
80	01010000	50
81	01010001	51
82	01010010	52
83	01010011	53
84	01010100	54
85	01010101	55
86	01010110	56
87	01010111	57
88	01011000	58
89	01011001	59
90	01011010	5A
etc.		



On the COM-TRAN TEN we may get involved in a few problems with binary forms we will wish to convert to their decimal forms. The registers show the numbers that are the powers of two. Their equivalent values are:

$2^0 = 1$	$2^5 = 32$	$2^{10} = 1024$
$2^1 = 2$	$2^6 = 64$	$2^{11} = 2048$
$2^2 = 4$	$2^7 = 128$	$2^{12} = 4096$
$2^3 = 8$	$2^8 = 256$	$2^{13} = 8192$
$2^4 = 16$	$2^9 = 512$	

It will not be necessary to convert every number we use to binary in this way of division. Most of the numbers we will work with are codes, numbers that command the computer to do a certain task. To make things easier to express and convert them into computer language we will use a special coding form called hexadecimal.

In hexadecimal form the binary numbers are arranged into groups of four bits each. Thus the binary form of the decimal number 53 would be shown as:

0011 0101

Each place still has the same value. Therefore, we still have  $32 + 16 + 4 + 1$ . We have a total of eight places each of which has a value of (from left to right):

128    64    32    16    8    4    2    1

$8 + 4 + 2 + 1 = 15$ , the largest number we can represent with four bits. Notice that:

128 = 8 x 16  
 64 = 4 x 16  
 32 = 2 x 16  
 16 = 1 x 16.

So we can group the binary bits above and represent the decimal number 53 as '35' in hexadecimal form. This means it has a value of (see table 1-1):

$(3 \times 16) + (5 \times 1)$ .

Using this system we are able to represent numbers when we code a program with a few symbols and still be able to express quickly a number in binary form. If we wish to convert '60' (the single quote(') symbols mean the number is in hexadecimal form) to binary form, we change each digit to four binary digits:

0110 0000.

'98' would be 1001 1000  
 '20' would be 0010 0000  
 '11' would be 0001 0001  
 '48' would be 0100 1000  
 '68' would be 0110 1000

In using hexadecimal numbers, we need six more digit symbols to represent the numbers from ten through fifteen. We wish to use one symbol for each hexadecimal digit. So we invent some symbols and to make it easy to remember them we will use the first letters of the alphabet. Therefore, ten will be represented by A, eleven by B, twelve by C, thirteen by D, fourteen by E, and fifteen by F. In keeping with this:

'F0' would be 1111 0000  
 'A8' would be 1010 1000  
 'B0' would be 1011 0000  
 'D9' would be 1101 1001  
 '54' would be 0101 0100

Manual Input Procedure

One of the most important things about a computer is its ability to store information. Here we will learn how to store information in memory, where the information is stored so we can keep track of it, and what information we store.

To open the input gates to memory follow this procedure: (see figure 1-40).

1. Power ON
  2. CLEAR (Press CLEAR switch)
  3. Select WT MEMORY
  4. START (press START switch)
  5. INPUT starting address on INPUT keys
  6. Select M register (Press M register switch)
  7. INPUT "word" on INPUT keys ("word" is two hexadecimal digits)
  8. START
- ...repeat steps 7 and 8, until all information is in.

Decide where you wish to store information. The MEMORY ADDRESS register holds the location or cell at which you will store information. In step 5 above we select the first of several cells in which we will store information. In step 6 we select the M register; this opens that particular memory cell to accept information. From then on whatever we enter on the INPUT keys will be entered by pressing the START switch into each memory cell.

We refer to two hexadecimal digits or eight bits as a word. Computers differ in "word" size. The COM-TRAN TEN computer has eight bits in its ACCUMULATOR, BUFFER, QUOTIENT, INDEX, and COUNTDOWN registers. This is the maximum any one memory cell can hold also. This term "word" may represent an instruction code, an address, an operand, a constant, or a piece of data between +127 and -128.

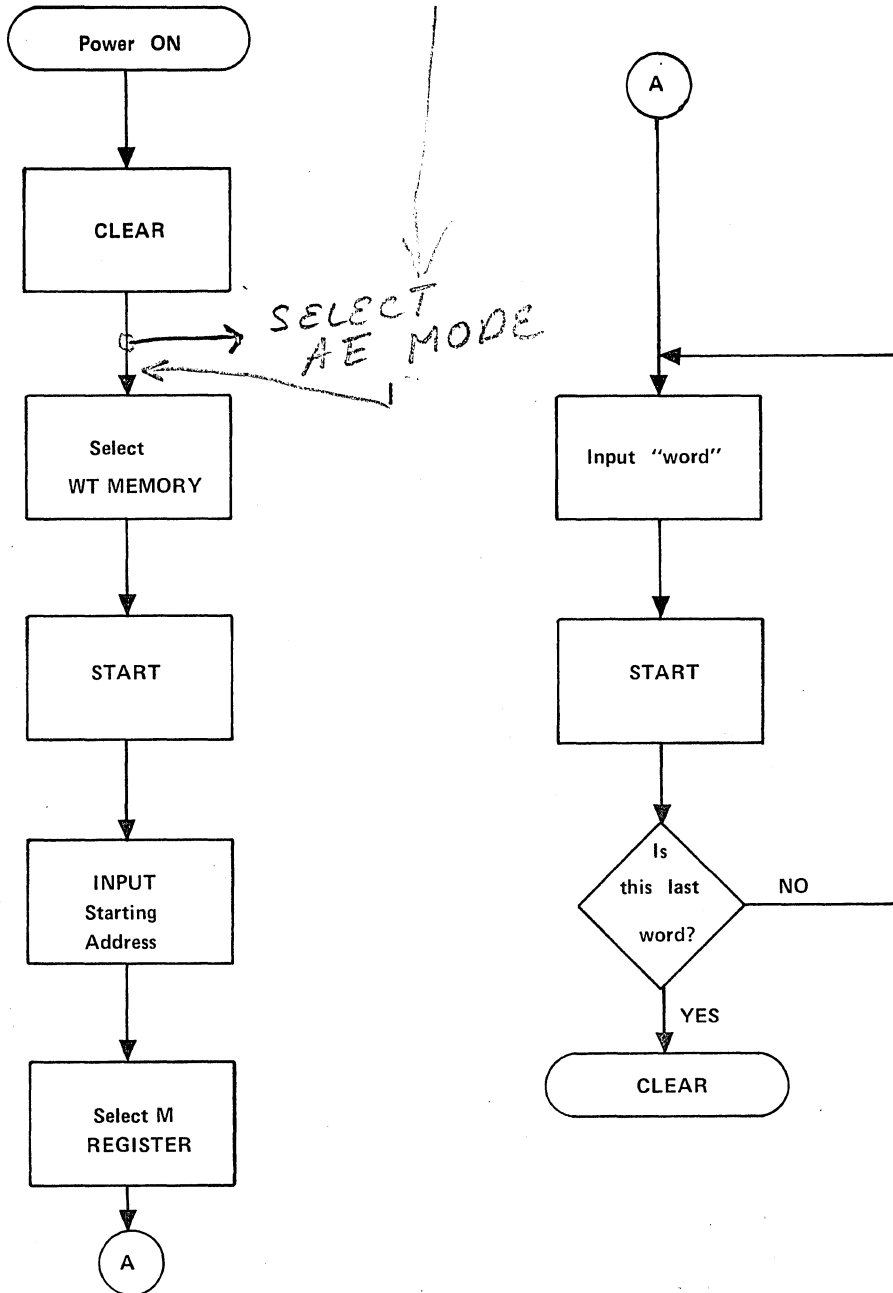
Following the procedure outlined above:

starting at MEMORY ADDRESS '10' (step 5)  
 input the following: (steps 7., 8.)

hexadecimal keys	or	INPUT binary keys
02		0000 0010
30		0011 0000
D8		1101 1000
57		0101 0111
16		0001 0110
F0		1111 0000
98		1001 1000
B8		1011 1000
04		0000 0100
A1		1010 0001

Press the RESET switch whenever you wish to erase what is on the INPUT keys. Only when you press the START is something entered into the computer.

SELECT FO FOR OP CODE  
SET C REG. TO COUNT OF FF  
SET E FF,



RDA26-438

Figure 1-40. Manual Input Procedure

## Manual Output Procedure

You have just entered ten pieces of information or "words" into the computer's memory starting at cell '10'. Now let us retrieve that information. We will look at the contents of memory cells '10' through '19' and examine what is stored there.

To open the output gates of memory follow this procedure (see figure 1-41):

1. Power ON
2. CLEAR
3. Select RD MEMORY
4. INPUT starting address on INPUT keys (that is '10')
5. Select M register
6. START

In the BUFFER register will be the contents of memory cell '10'.

Is it: 0000 0010 or '02'?

In the MEMORY ADDRESS register will be '11' and in the BUFFER will be '30'.

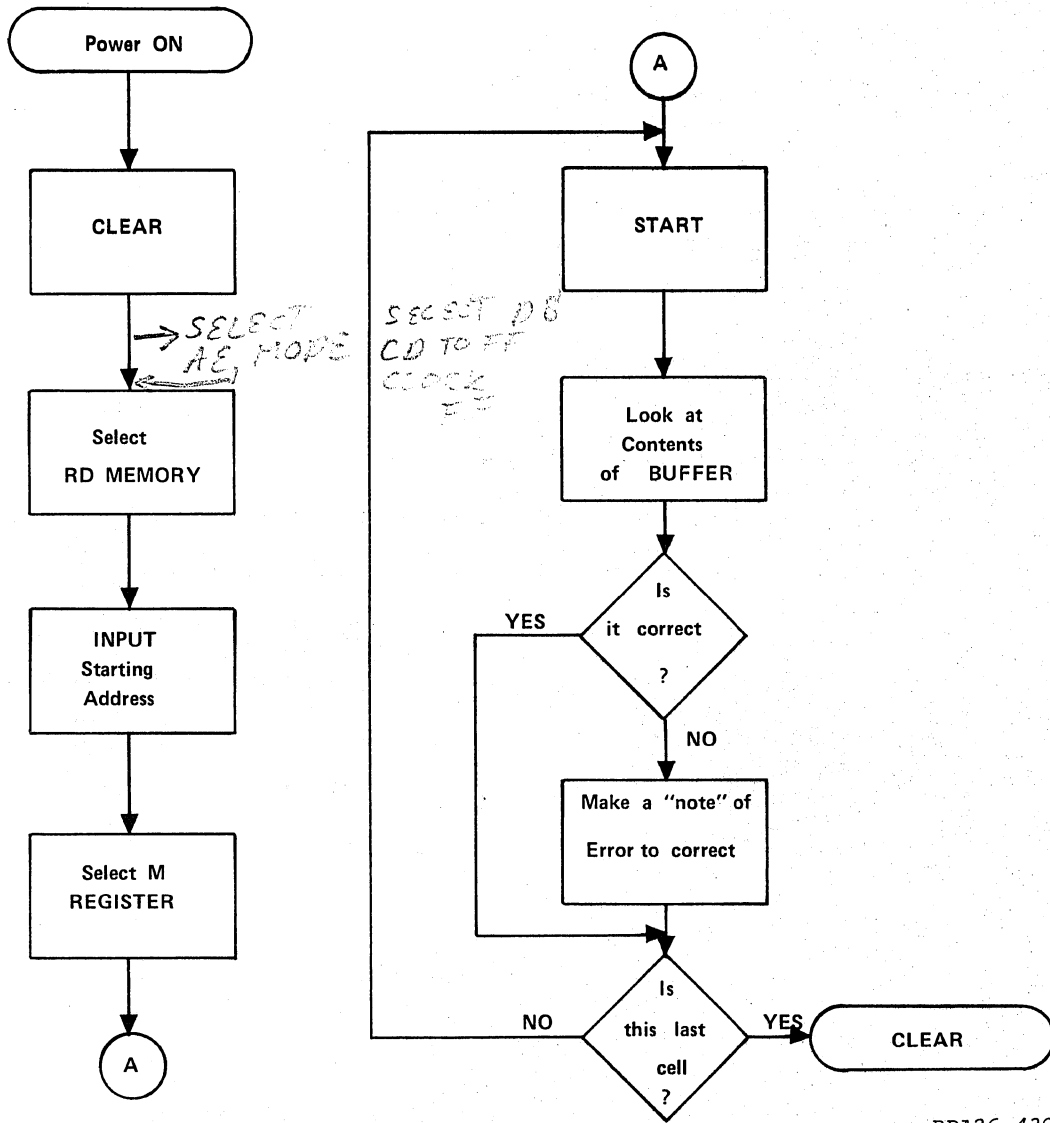
How does it look?

Continue to check out the MEMORY ADDRESS and the BUFFER.

MEMORY ADDRESS	BUFFER
12	D8
13	57
14	16
15	F0
16	98
17	B8
18	04
19	A1

We have outlined the procedure for manually inputting data into memory and manually recalling it. In summary:

MANUAL **INPUT	MANUAL ** OUTPUT
1. Power ON	1. Power ON
2. CLEAR	2. CLEAR
3. SELECT WT MEMORY	3. Select RD MEMORY
4. START	4. INPUT starting address
5. INPUT starting address	5. Select M register
6. Select M register	6. START
7. INPUT "word"	...(repeat step 6)
8. START	
...(repeat steps 7,8)	



RDA26-439

Figure 1-41. Manual Output Procedure

REVIEW QUESTIONS 1-4

1. \_\_\_\_\_ Holds the code for the instructions being performed.
  - \_\_\_\_\_ All data going into or coming from memory goes here.
  - \_\_\_\_\_ Performs all arithmetic operations.
  - \_\_\_\_\_ Used to modify a memory address.
  - \_\_\_\_\_ Used to manually input hex numbers.
  - \_\_\_\_\_ It holds the quotient after a divide.
  - \_\_\_\_\_ Used to control the timing.
  - \_\_\_\_\_ Conductors used to carry information many different places.
  - \_\_\_\_\_ Used to store numbers.
  - \_\_\_\_\_ Used to help control input and output.
  - \_\_\_\_\_ Used to take the two's complement of a number.
  - \_\_\_\_\_ Used to hold the results of most arithmetic and logical operations.
  - \_\_\_\_\_ Used to hold values that are manually input.
  - \_\_\_\_\_ Used to hold the address of the next instruction to be performed.
  - \_\_\_\_\_ Used to add the Memory Address Register and the Index Register.
  - \_\_\_\_\_ Used to tell the computer what memory location you are using.
  - \_\_\_\_\_ Many of the data transfers through the computer go through the register.
  - \_\_\_\_\_ Performs some logical operations.
  - \_\_\_\_\_ Used to hold the results of most logical operations.
- a. Bus Lines
  - b. Input Switches
  - c. Input Register
  - d. Buffer Register
  - e. Two's Complementer
  - f. Program Address Register
  - g. OP Code Register
  - h. Memory
  - i. Memory Address Register
  - j. Index Register

- k. Index Adder
- l. Accumulator
- m. Quotient Register
- n. Arithmetic Logic Unit
- o. Countdown Register
- p. Distributor

Answer the following questions in your own words.

2. Information to be written into the computer memory can come from which two places?
3. The information goes into what register before it goes into the Memory?
4. To execute a program, the address of the first instruction must be put in which register?
5. Where is an instruction transferred after it is brought from Memory to the Buffer?
6. If an instruction requires data, where does the data come from?
7. What is the fastest way to enter data into the Input Register?
8. To enter data into the "OP CODE" register from the Input Register which switch is pushed?
9. The "RD" switch next to the Register Selection Switch is used to do what?
10. What does the acquisition phase of an instruction cycle do?
11. What does the execution phase of an instruction cycle do?
12. What is the purpose of the display panel?
13. After a divide, where does the remainder appear?

14. Which element (unit) of the COM-TRAN TEN performs all arithmetic and most logical operations?
15. Which CT-TEN register keeps a running total of the result of arithmetic operations?
16. Can the ALU handle both positive and negative numbers during a subtraction operation?
17. What three CT-TEN registers are used to hold the operands of multiply operation?
18. Why does the COM-TRAN TEN use the two's complement to express a negative difference to a subtraction problem?
19. What is the two's complement of the binary number 0101 1010?
20. During what operations would the C Register of the CT-TEN be used?
21. Which occurs first during a machine cycle, acquisition time or execution time?
22. Which bits of an instruction code determine high order memory locations?
23. Which bit(s) of an instruction code determine indexing?
24. In the CT-TEN, which register must the operand pass through to reach the M register?

#### PROGRAMMING

The attributes which have contributed to the growth and importance of modern digital computers include the following:

1. Ability to operate at high speeds.
2. Capability of storing data permanently.
3. Operation in the stored-program mode.
4. Ability to handle "decision"-type operations.



5. Accuracy in repeated performances.
6. Capability of changing data and instructions as programmed.
7. Repetitive operations as instructed.
8. Indication of errors in number magnitude and in programming instructions.
9. Alphanumeric operations whereby messages can be stored and printed out.

Despite all these qualifications, however, the digital computer must still be told what to do, how to do it, and what must be done with the end results. Trained personnel who know how to communicate with computers and get them to process data are known as Computer Programmers. Computer programmers communicate with the computer through a medium called program language (Fortran, Cobol, RPG, etc.). This program language has to be changed to machine language (coded commands which tell the computer what to do) by a compiler, before it can be used by the computer. Computer technicians do not need to know a program language; however, they must fully understand the machine language.

Machine language which is made up of coded commands or instructions tells the computer what to do, how to do it, and where to store the results. When these instructions are sequenced to perform a specific job to reach a desired end, they become a program. In order for a computer technician to detect and locate any existing malfunctions, it is necessary for him to know the content of any register anytime during a program run.

In this section you will learn the type of instruction, instruction repertoire, instruction word format, data word format and the use of each instruction in the COM-TRAN TEN. You should relate the flow of each instruction and/or data to the basic block diagram and the block diagram of the COM-TRAN TEN.

#### Type of Instruction

The 44 discrete instructions used by the COM-TRAN TEN can be broken into various categories as follows:

1. Load - 7 instructions
2. Store - 3 instructions
3. Arithmetic - 9 instructions
4. Logical - 5 instructions
5. Branch - 11 instructions
6. Input/Output - 9 instructions

#### Instruction Repertoire

Following is a list of the COM-TRAN TEN instructions with descriptions. The lower case letters following the symbolic code of each instruction have the following meaning:

1. m - Memory instruction
2. k - Immediate instruction
3. x - Indexed instruction

INSTRUCTION REPERTOIRE

HEX CODE	SYMBOLIC	DESCRIPTION	NOTES
		LOAD	
12	LX1,k	LOAD INDEX IMMEDIATE Load INDEX Register with a count of k.	3
01	LC1,k	LOAD COUNTDOWN IMMEDIATE Load COUNTDOWN Register with a count of k.	3
02	LA1,k	LOAD ACCUMULATOR IMMEDIATE Load ACCUMULATOR with the value k.	3,14
20	LDA,m,x	LOAD ACCUMULATOR Load ACCUMULATOR with contents of memory address m.	1,2,14
30	LCC,m,x	LOAD CONSECUTIVE Transfer the contents of memory address m to memory address m + 1	1,2
38	LAN,m,x	LOAD ACCUMULATOR NEGATIVE Load ACCUMULATOR with the two's complement of the contents of m.	1,2,14
40	LDQ,m,x	LOAD QUOTIENT REGISTER Load QUOTIENT Register with the contents of memory address m.	1,2
		STORE	
48	STA,m,x	STORE ACCUMULATOR Store contents of the ACCUMULATOR at memory address m.	1,2
50	STX,m,x	STORE INDEX Store contents of the INDEX Register at memory address m.	1,2
58	STQ,m,x	STORE QUOTIENT REGISTER Store the contents of the QUOTIENT Register at memory address m.	1,2

HEX CODE	SYMBOLIC	DESCRIPTION	NOTES
ARITHMETIC			
60	ADD,m,x	ADD  Add the contents of memory address m to the contents of the ACCUMULATOR leaving the result in the ACCUMULATOR. Conditionally set carry and add overflow.	1,2,11 12,14
68	SUB,m,x	SUBTRACT  Subtract the contents of memory address m from the contents of the ACCUMULATOR leaving the result in the ACCUMULATOR. Conditionally set carry and add overflow.	1,2,11 12,14
70	MPY,m,x	MULTIPLY  Multiply the contents of the ACCUMULATOR by the contents of memory address m, leaving a double length product in the AQ Register.	1,2,14
78	DIV,m,x	DIVIDE  Divide the double length number in the AQ Register by the contents of memory address m, leaving the quotient in the Q Register and the remainder will be in the ACCUMULATOR. Set Condition code according to the sign of the quotient. The sign of the remainder will be the same as the sign of the dividend. Conditionally set divide overflow.	1,2, 13,14
80	RAO,m,x	REPLACE ADD ONE  Add 1 to the contents of memory address m. If the contents of memory address m is FF, this instruction will cause the contents of m to be set to zero and the carry bit will be set. The ACCUMULATOR contains the result of the addition.	1,2,4 11,14
88	RSO,m,x	REPLACE SUBTRACT ONE  Subtract 1 from the contents of memory address m. If the contents of memory address m is zero, this instruction will cause the contents of m to be set to FF and the carry bit will be set. The ACCUMULATOR contains the result of the subtraction.	1,2,4 11,14
03	INX,k	INCREASE INDEX  Increase the contents of the INDEX Register by k.	3

HEX CODE	SYMBOLIC	DESCRIPTION	NOTES
		LOGICAL	
0B	SLA,k	SHIFT LEFT ARITHMETIC Shift the AQ Register left k places, filling in zeros on the right.	3
10	SRA,k	SHIFT RIGHT ARITHMETIC Shift the AQ Register right k places, propagating sign bits on the left.	3
13	SLL,k	SHIFT LEFT LOGICAL Shift the ACCUMULATOR left k places, filling in zeros on the right.	3
18	SRL,k	SHIFT RIGHT LOGICAL Shift the ACCUMULATOR right k places, filling in zeros on the left.	3
19	AND,k	AND Form the bit-by-bit logical product of k and the contents of the ACCUMULATOR, leaving the result in the ACCUMULATOR.	3
1A	IOR,k	INCLUSIVE OR Form the bit-by-bit Inclusive OR of k and the contents of the ACCUMULATOR, leaving the result in the ACCUMULATOR.	3
1B	XOR,k	EXCLUSIVE OR Form the bit-by-bit Exclusive OR of k and the contents of the ACCUMULA- TOR, leaving the result in the ACCUMULATOR.	3
		BRANCH	
90	BUN,m,x	BRANCH UNCONDITIONAL Branch to memory address m.	1,2
98	BST,m,x	BRANCH AND STOP Branch to memory address m and stop.	1,2
A0	BSB,m,x	BRANCH TO SUBROUTINE Store BUN op code in location m. Store the contents of the P Register (next instruction address) in location m + 1. Branch to location m + 2.	1,2

HEX CODE	SYMBOLIC	DESCRIPTION	NOTES
A8	BPS,m,x	BRANCH ON POSITIVE  Branch to memory address m if the condition code is greater than zero.	1,2
B0	BZE,m,x	BRANCH ON ZERO  Branch to memory address m if the condition code is equal to zero.	1,2
B8	BNG,m,x	BRANCH ON NEGATIVE  Branch to memory address m if the condition code is less than zero.	1,2
C0	BNC,m,x	BRANCH ON NO CARRY  Branch to memory address m if the condition code is not carry.	1,2
C8	BXZ,m,x	BRANCH ON INDEX ZERO  Branch to memory address m if the INDEX Register is zero.	1,2
08	SKI,k	SKIP ON INTERRUPT  Skip k instructions or 2k words if interrupt is set.	3
09	SKS,k	SKIP ON SENSE SWITCH  Skip k instructions or 2k words if the sense switch is on.	3
0A	SKF,k	SKIP ON FLAG  Skip k instructions or 2k words if flag is set.	3
		INPUT/OUTPUT	
D0	WDB,m,x	WRITE DATA BLOCK  Transfer the contents of consecutive memory locations, starting with address m to the selected external device. Continue until the C Register is zero. The number of words transferred from memory will be one more than the initial count in the C Register.	1,2, 10

HEX CODE	SYMBOLIC	DESCRIPTION	NOTES
D8	MNO,m,x	MANUAL OUTPUT  Transfer the contents of consecutive memory locations to the Input Register. Continue until the C Register is zero. The number of words transferred from memory will be one more than the initial count in the C Register.	1,2,15
E0	RDB,m,x	READ DATA BLOCK  Store data from the selected external device in consecutive memory locations starting with address m. Continue until C Register is zero. The number of words stored in memory will be one more than the initial count in the C Register.	1,2,7,9
E8	RDI,m,x	READ UNTIL INTERRUPT  Store data from the selected external device in consecutive memory locations starting with address m. Continue until interrupt is set.	1,2,7,8,9
F0	MNI,m,x	MANUAL INPUT  Transfer the contents of the Input Register to consecutive memory locations 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.	1,2
11	OCD,k	OUTPUT COMMAND  Transmit the operand k to the external device addressed by the three low order bits in k.	3,6,9
00	SST,k	SENSE STATUS  The previously addressed device sends an 8-bit status word which is transferred to the ACCUMULATOR.	
28	FLC,k	FLAG CLEAR  Clear/flag to zero	5
F8	FLS,k	FLAG SET  Set flag to one.	5

NOTES for Instruction Repertoire

1. To address memory locations above ( $FF_{16}$ ); 1, 2, or 3 is added to the instruction code.
2. Setting the index bit in on instruction OP CODE will cause the X Register to be added to the memory address.
3. Value k is limited to ( $FF_{16}$ ).
4. Arithmetic in RAO and RSO instruction handles an 8-bit unsigned number.
5. Operand is not used in FLAG CLEAR or FLAG SET instructions.
6. OCD instruction operand bits have the following significance:
  - Bits 2, 1, 0 select one of eight peripheral units
  - Bit 3 selects HEX mode
  - Bit 4 selects APH mode
7. RDB and RDI instructions automatically generate and transmit the XON character to the teletype. (XON turns on the tape reader.)
8. When using the teletype tape reader, the tape must end with XOFF followed by CONTROL 1. (XOFF turns the reader off after reading one more character; Control 1 sets the Interrupt Bit.)
9. When using the teletype keyboard CONTROL H sets HEX mode; CONTROL A sets ALPHA mode. In HEX mode a colon (:) loads the preceding two HEXadecimal characters into memory.
10. When using the teletype peripheral in HEXadecimal mode, output colons (:) are automatically inserted, and line feeds and carriage returns are automatically generated.
11. CARRY Condition Code may be set by ADD, SUB, RAO, or RSO instructions. The CARRY bit indicates that a carry or borrow took place from the 8-bit arithmetic result. The 8-bit numbers are handled as unsigned binary numbers.
12. ADD overflow is an error condition. This bit is set whenever the magnitude of the result exceeds the largest signed number that can be represented by 8-bits. The 8-bit numbers are handled as signed two's complement numbers.
13. DIVIDE overflow is an error condition. This bit is set whenever the magnitude of the result exceeds the largest signed number that can be represented by 8-bits. The 8-bit number in the Quotient register is a signed two's complement number as is the remainder in the Accumulator.
14. CONDITION CODE reflects the status of the Accumulator at the end of each instruction. ~~In addition to~~ the following:
  - EXCEPT*
    - a. Indicates status of the Quotient Register upon completion of a Divide instruction.
    - b. Indicates status of the double length AQ Register after Multiply and Arithmetic Shift instructions. (*MULT, SRA, SLA*)
15. On the MANUAL OUTPUT instruction, the contents of the indicated memory location will also be displayed on the Buffer Register. The computer will stop with the WAIT light on, after each word is displayed. To continue, press the START switch.

## Instruction Format

An instruction consists of two consecutive 8-bit words. The first word located at a program address is displayed in the S Register. The second word is located at the next consecutive address and is displayed in the M Register.

For immediate instructions not requiring memory access for addition data, the first word is the op code and the

```
┌---OP CODE---┐      ┌---OPERAND---┐
  x x x x x x x x      x x x x x x x x
```

second word is the operand of the instructions.

For instructions requiring memory access, the second word and the two lower bits of the

```
  x x x x x x x x      x x x x x x x x
└Op Code┘ ┌---Memory Address---┘
           |
           └Index
```

first word contain the 10-bit operand address word. The operand address is displayed in the Memory Address Register. The five upper bits of the first word represent the op code and the sixth bit of the first word indicates Indexing.

## Addressing Upper Memory

The COM-TRAN TEN has 400 hex memory locations, numbered 0 - 3FF. These 400 locations may be broken into "pages" of memory as follows:

```
  0 - FF
 100 - 1FF
 200 - 2FF
 300 - 3FF } upper memory
```

In order to get into upper memory, the "page" number must be added to the op code. For example: LDA 15 = 20 15. To get to upper memory, we must modify the op code.

21 15 would load A into location 115

22 15 would load A into location 215

23 15 would load A into location 315

## Indexing

There are times when it is desirable to modify the operand address of a given instruction without actually changing the data stored in memory. Indexing is an easy way to accomplish this.

Setting S2, the index bit, to a 1 will cause the contents of the Index Reg to be added to the contents of the M Reg during Acquisition time.



For example:

	Op Code	M Reg	Index Reg
Hex	24	013	16
Binary	0010 0100 00	0001 0011	0001 0110

After indexing, the contents of the M Reg will be:

029  
00 0010 1001

\*Remember that bits S0-S1 and bits M8-M9 are the same two bits used for high order addressing.

### Number Representation

The COM-TRAN TEN uses a two's complement number system. Positive numbers are treated in their normal binary representation. Thus, decimal 10 would have the binary equivalent of 0000 1010.

The negative representation for this number is the two's complement of it. The two's complement of any number is derived by complementing the number and adding one to it.

BINARY NUMBER	=	0000 1010
COMPLEMENT	=	1111 0101
+1	=	<u>0000 0001</u>
TWO'S COMPLEMENT	=	1111 0110

This number is the negative representative for decimal 10. A simple rule for two's complementation:

START WITH THE LSB POSITION

COPY THE BITS UP TO AND INCLUDING THE FIRST ONE

THEN COMPLEMENT ALL REMAINING BIT POSITIONS

MSB of any number is the sign bit and is a "1" for negative numbers. The largest negative number in two's complement is one larger in magnitude than the largest positive number.

1 0 0 0	0 0 0 0	=	-128 <sub>10</sub>
0 1 1 1	1 1 1 1	=	+127 <sub>10</sub>
↑			
sign bit			

### Instructions

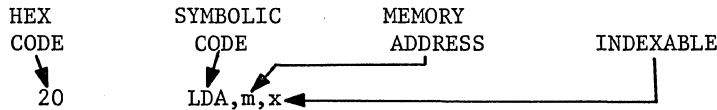
Now you are ready to get into the instruction set of the COM-TRAN TEN. Keep in mind that these instructions are for this computer only. Other computers will have similar instructions and codes but not necessarily the same ones.

There are six different categories of instructions in the COM-TRAN TEN. You will not learn all of the instructions in one category and then go to the next. We will be taking instructions from different categories so that we can start writing small programs.

The load group is used to put data into the registers. The store group is used to take data from the computer registers and put the data in memory. The arithmetic instructions are used to do the arithmetic operations such as add and subtract. The logical group performs such things as shifting, OR, exclusive OR, and AND functions. The branching group allows control of the program. It allows a change in the sequence of the program. The Input/Output (I/O) group allows data in from the teletype or out to the teletype. As you go through the instructions in each group you will get a better idea of what each instruction does.

Each instruction has a symbolic name which generally is an abbreviation of what it does. It also has a Hexadecimal number code. This code is what will be put into the computer memory in binary. The code is called machine language. You will need to know some terms to understand the explanation of the instructions. Instructions will be laid out in one of 2 formats for explanation.

The First Format is as follows:



The 20 is the Hexadecimal code for the Load the Accumulator instruction. The m tells you that the instruction must go to a memory location (m) to get the number it is going to put into the accumulator. The X indicates that the memory address (m) can be indexed. Indexing will be explained later. For now, note that it can be indexed.

The Second Format is as follows:

02 (Hex Code)            LAI (Symbolic Name)            K (Constant)

The 02 is the Hex code for the Load Accumulator immediately instruction. The K represents the number that will be put into the Accumulator. Note, the difference in the two instructions. The LDA instruction must have a memory address to tell where the number is coming from. The LAI uses no address. Instead, the number to be loaded is a part of the Instruction.

Example:    LDA 25;    LAI 25

The LDA 25 means that the contents of memory location 25 will be put into the Accumulator. This memory location may contain any 8-bit number. The LAI 25 means that the number 25<sub>(16)</sub> will be put into the Accumulator.

Note, the difference between immediate instructions and memory instructions.

### Instruction Word Format

All instructions in the COM-TRAN TEN take 2 memory (8 bits each) locations. The first location will hold the hexadecimal code for the instruction to be used. The second location will hold either the constant K or the memory location m's address. (Both K and m must be in hexadecimal.)

Those instructions that use memory addresses may sometimes need modification. Using 8 bits, the highest number we can get is a FF. This is 127 in decimal. Our computer has 400 (16) memory locations. We have to add numbers to our Hexadecimal instruction code if we want an address larger than FF. To address memory location 14B, we must add the 1 to the Hex instruction code and put the 4B in the second memory location.

Examples:

SYMBOLIC CODE	HEX CODE	
	<u>First memory location</u>	<u>Second memory location</u>
LDA 35	20	35
LDA 135	21	35
LDA 235	22	35
LDA 335	23	35

Each of these instructions loads the accumulator with the contents of different memory locations.

Micro Steps:

When you look at the programmers reference card, KDA-3020 you will find a column by the instructions labeled "Micro Steps." These steps are there as a reminder of what each instruction does. They do not tell you everything that happens during that instruction. They will serve as a quick reference so you will not have to search through the book. An explanation follows:

REGISTERS

- A - Accumulator
- C - Countdown
- M - Memory Address
- P - Program Address
- Q - Quotient
- AQ - Combined accumulator and quotient
- I - Input
- X - Index
- K - Constant

CONDITION CODES

- CARRY-Carry or borrow
- (>0) greater than zero
- (=0) equal to zero
- (<0) less than zero

SPECIAL SYMBOLS

- Goes to
- c() the contents of
- ⇨(K) shift right K places
- ⇦(K) shift left K places
- + add
- subtract
- X multiply
- ÷ divide
- + or
- and
- ⊕ exclusive or
- NOT or compliment of

Examples:

ADD,m             $A + c(M) \rightarrow A$

A register added to the contents of memory location m goes to the A.

SUB,m             $A - c(M) \rightarrow A$

A register minus the contents of memory location m goes to the A register.

The following seven instructions are the first basic group you will learn. There is a detailed explanation of each instruction with it.

Hex Code	Symbolic Name	Load Accumulator
20	LDA,m,x	

LDA is a load accumulator instruction. The contents of memory location m will be put into the accumulator. This instruction can be indexed which will be explained later in the book. Memory location (m) is unchanged.

Example:

Before: LDA 3E Accumulator = F3 Memory Location 3E = 27

After: LDA 3E Accumulator = 27 Memory Location 3E = 27

Note, that what was in the accumulator (F3 in this case) was destroyed and the contents of location 3E (27 in this case) was put into the accumulator.

48 STA,m,x Store Accumulator

This instruction is used to put the contents of the accumulator into a memory location (m). The contents of the memory location before the instruction is executed will be destroyed. The accumulator contents will be unchanged.

Example:

Before: STA 43 Accumulator = 7B Memory Location 43 = D1

After: STA 43 Accumulator = 7B Memory Location 43 = 7B

The D1 that was in memory location 43 is destroyed by the STA 43 instruction.

60 ADD,m,x ADD

This instruction is used to ADD the contents of Memory location m to the contents of the accumulator. The contents of Memory location will be unchanged, but the accumulator will contain the sum of the two numbers.

Example:

Before: ADD 1F Accumulator = 08 Memory Location 1F = 05

After: ADD 1F Accumulator = 0D Memory Location 1F = 05

Note, that the 08 that was in the accumulator was destroyed by the ADD.

## 68 SUB,m,x SUBTRACT

This instruction is used to subtract the contents of memory location m from the accumulator leaving the difference in the accumulator. All negative numbers are in two's complement form. If the computer comes up with a negative answer (such as subtracting a larger number from a smaller number), then it will be in two's complement form. If you enter a negative number it must be in two's complement.

Example:

Before: SUB 71 Accumulator = 43 Memory Location 71 = 21

After: SUB 71 Accumulator = 22 Memory Location 71 = 21

## 13 SLL,k Shift Left Logical

This instruction is used to shift the accumulator to the left. It will be shifted the number of times specified by the constant k. Remember, k will be specified in Hexadecimal. The constant k can be any number from 0 to FF. It would not be practical to shift it more than 8 times because after 8 shifts the accumulator will be zero.

Example:

Before: SLL 03 Accumulator = 0101 1100 (2) = 5c

After: SLL 03 Accumulator = 1110 0000 (2) = E0

Note, zeros shift in on the right as the number moves left.

## 18 SRL,k Shift Right Logical

This instruction is used to shift the contents of the accumulator to the right. The number of places the accumulator will be shifted is specified by the constant k. Again, k can be any number between 0 and FF, but after 8 shifts the accumulator will be zero. Sign bit is shifted along with the number.

Example:

Before: SRL 01 Accumulator = 1011 0100 (2) = B4

After: SRL 01 Accumulator = 0101 1010 (2) = 5A

Note, zeros are shifted in on the left as the number is shifted right. Both the SRL and the SLL used a constant k. Logical shifts only affect the accumulator. Remember, that this is not an address of memory.

## 98 BST,m,x Branch and Stop

This instruction is used to stop the computer. You will put this instruction at the end of most of your programs. The BST instruction will load the Program Counter (the register that stores the next instruction address) with the address of memory location m. When the computer will halt, the Program Counter is the only register that changed.

Example:

Before: BST 36 Program Counter = 74

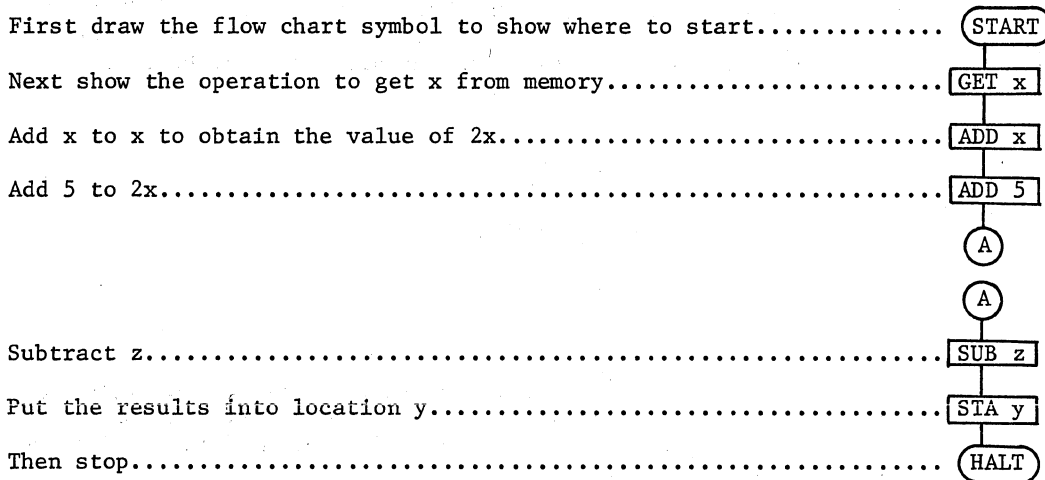
After: BST 36 Program Counter = 36 and the computer has halted

Now we will put these seven instructions to work. Suppose we wanted to figure out this formula:  $y = 2x + 5 - z$ .

Our first step in writing the program would be to analyze the problem. We know, for instance, that we want the answer  $y$ . Therefore,  $y$  is an unknown. To find  $y$  we must know  $x$  and  $z$ . We can assume that those two will be given to us. We can also see that we are going to have to add, subtract, and multiply to get the answer. At least now we know what we must do to find the answer.

The second step in writing a program is to devise a method to solve the problem. This has been done for us since we are given the formula. All we have to do is tell the computer how to work the formula. If, however, we had been given a problem to determine some mathematical process, then we would have to find or derive a formula.

Our third step is to develop a flow chart. Each step is explained in the following example.



Now that we have a flow chart that will work we need to write the program. To do this all we need to do is find an instruction or a group of instructions that will do each thing in the flow chart. The first thing we need to do is to assign  $x$ ,  $y$ ,  $z$  and 5 memory locations where they can sit until we are ready for them. These locations can be anywhere in memory that we are not using for something else, like our program. So let's put  $x$  in location 20,  $y$  in location 21,  $z$  in location 22, and 5 in location 23. We could have put them anywhere, but once we have assigned these memory locations, we must use them.

Find an instruction that will get  $x$  out of memory and put it in the accumulator. It just so happens that the LDA instruction does just that. So our first instruction is LDA 20. To make  $x$  become  $2x$ , we must add another  $x$ . The instruction used for this is ADD 20. To add 5 we must do ADD 23. We now need a way to subtract  $z$ . We can use the SUB instruction. So to subtract  $z$  we do a SUB 22. Now, we have our final result. We want to put this in  $y$  so we must do a STA 21. The last thing we must do is stop the computer. We can do this with a BST 00. The address portion of this instruction usually causes a branch back to the beginning of the program or to an area in the memory where data is stored so you can look at the results.

Now we have our program.

```
LDA 20 Load x
ADD 20 Add x
ADD 23 Add 5
SUB 22 Subtract z
STA 21 Store y
BST 00 STOP
```

Our program must also go into memory. Usually you will put the program into memory starting at location 00. Programs can, of course, start at any location in memory just as data can be located in any available memory address. It was mentioned earlier that all instructions take two memory locations. One location holds the instruction code while the next successive location holds either the address location or operand (constant).

If we put the LDA 20 into memory locations 00 and 01 then the ADD 20 instruction must go into memory locations 02 and 03. The hex code for LDA (20) goes into memory location 00. The memory address 20 which is the location for x, goes into memory location 01. Each instruction needs two memory locations. The program must follow in sequential memory locations. This is known as a straight line program.

The fifth step in writing a program is to code it into machine language. In memory locations 00 and 01 we will put the LDA 20 instruction. In locations 02 and 03 we will put the ADD 20 instructions and so on. Our format will be as follows:

Memory locations	Symbolic code	Hex Code	Remarks
00, 01	LDA 20	20 20	Load Accumulator with x
02, 03	ADD 20	60 20	Add x to accumulator
04, 05	ADD 23	60 23	Add 5 to accumulator
06, 07	SUB 22	68 22	Subtract z from accumulator
08, 09	STA 21	48 21	Store accumulator in y
0A, 0B	BST 00	98 00	Stop with 00 in program count

Notice first that all memory locations are in hex. If the first instruction starts in an even memory location, so do the remaining instructions.

The first number in the hex column is the hexadecimal code for each instruction. The second number is either the memory address or a constant (operand) depending on the type of instruction.

The sixth and seventh steps would be performed by inserting the instructions into the computer and executing the program. You do not need to do these steps for this program. You will be doing them on the programs you write.

In the program above x and z can be any numbers capable of fitting into one memory location. That means that we can find y for any number of different x's and z's. All we would have to do is to put a new x and a new z in and run the program again.

If we let x = 4 and z = 1 our program would perform as follows:

<u>PROGRAM</u>	<u>ACCUMULATOR</u>	<u>REMARKS</u>
LDA 20	04	Load the 4
ADD 20	08	Add 4 to it
ADD 23	0D	Add 5 to it
SUB 22	0C	Subtract 1
STA 21	0C	Store answer in y
BST 00	0C	STOP

If we looked in memory location 21 we would find that it contains a 0C. If you have any questions about the 7 instructions on this program, see your instructor.

#### REVIEW QUESTIONS 1-5

1. Match the following statements with the step in which it would be performed.

- \_\_\_ This step is sometimes performed by the computer.
  - \_\_\_ Figure how to take this given material and come up with the answer.
  - \_\_\_ If the program does not run correctly this step must be done.
  - \_\_\_ Make a pictorial representation of how to solve the problem.
1. Analyze the problem.
  2. Devise a general method to solve the problem.
  3. Develop a flow chart.
  4. Write the program.
  5. Put symbolic coding into machine language.
  6. Test the program.
  7. Revise the program.

Match the following instructions with their functions:

- \_\_\_ Subtracts the contents of the memory location specified from the Accumulator.
  - \_\_\_ Shifts the Accumulator right the number of places specified.
  - \_\_\_ Puts the contents of the Accumulator into memory.
  - \_\_\_ Adds the contents of the memory location specified to the Accumulator.
  - \_\_\_ Puts the contents of the memory location specified into the Accumulator.
  - \_\_\_ Puts the address portion into the Program Address Register and halts the computer.
  - \_\_\_ Shifts the Accumulator left the number of places specified.
1. LDA
  2. STA
  3. ADD
  4. SUB
  5. SLL
  6. SRL
  7. BST



3. Analyze the following programs. Give the contents of the Accumulator after each instruction. If you have trouble, go to the computer, load the program and run it one instruction at a time.

Memory Location	Symbolic Code	Accumulator
00,01	LDA 08	
02,03	SRL 03	
04,05	ADD 09	
06,07	BST 00	
08	1A	
09	05	

Memory Location	Symbolic Code	Accumulator
40,41	LDA 4E	
42,43	ADD 4F	
44,45	STA 4E	
46,47	SLL 01	
48,49	SUB 4F	
4A,4B	STA 4F	
4C,4D	BST 4E	
4E	07	
4F	02	

Now we can go into seven more of the instructions. The same format will be used to discuss these as was used with the first seven.

40 LDQ,m,x Load Q

This instruction is used to load the contents of the memory location m into the Q (quotient) register. The contents of memory location m are not changed.

Before: LDQ 07 Q Register = 21 Memory Location 07 = 12

After: LDQ 07 Q Register = 12 Memory Location 07 = 12

No other register is affected. In other words, the accumulator was not changed, just the Q register.

### 58 STQ,m,x Store Q

This instruction is used to put the contents of the Q register in memory location m. The contents of the Q are not affected. Upon completion of the STQ instruction, memory location m will contain the same information as the Q register.

#### Example:

Before: STQ C0 Q Register = 3C Memory Location C0 = FF

After: STQ C0 Q Register = 3C Memory Location C0 = 3C

Note, that the only thing affected is the contents of memory location m.

### 38 LAN,m,x Load Accumulator Negative

This instruction is used to put the two's complement of the contents of memory location m into the accumulator. The accumulator will hold the two's complement of what was in memory location m.

#### Example:

Before: LAN 4F Accumulator = 36 Memory Location 4F = 05

After: LAN 4F Accumulator = FB Memory Location 4F = 05

FB is the two's complement of 05. Actually FB is the 16's complement of 05 but hexadecimal is only a binary shorthand.

### 70 MPY,m,x Multiply

This instruction is used to multiply the contents of the accumulator by the contents of memory location m. The product of this multiply is placed back into the combined AQ register. The A register will hold the 8 most significant bits, while the Q will hold the 8 least significant bits. If you multiply two numbers whose product is greater than 7F, you would get an overflow. This would severely limit the numbers you can multiply. To avoid this problem, the answer is put into two registers. In that way you can never exceed the modulus (get an overflow) when you do a multiply.

#### Example 1:

Before: MPY 27 Accumulator = 05 Q Register = 34 Memory Location 27 = 07

After: MPY 27 Accumulator = 00 Q Register = 23 Memory Location 27 = 07

$5 \times 7 = 0023$  or in decimal  $5 \times 7 = 35$   $23(16) = 35(10)$

#### Example 2:

Before: MPY 62 A Register = 43 Q Register = 00 Memory Location 62 = 7C

After: MPY 62 A Register = 20 Q Register = 74 Memory Location 62 = 7C

$43 \times 7C = 2074$  or in decimal  $67 \times 124 = 8308$   $2074(16) = 8308(10)$

Even though in Example 1 the answer only needed 8 bits, the accumulator is still part of the answer.

## 78 DIV,m,x Divide

This instruction is used to divide the contents of the combined AQ registers by the contents of memory location m. The answer is put into the Q register and the remainder is put into the A register. Notice that it has a remainder not a fraction. In other words if you divide 5 by 2, the answer will be 2 in the Q reg and 1 in the A reg. You will not get a 2.5 answer. Note also that the answer must be contained in 8 bits. That is if you divide one number by another and the answer is greater than 7F you will get an error.

The accumulator and the Q register are considered as one 16-bit register when doing a divide. The accumulator will hold the sign of the number. If the number to be divided by is FB (-5), then the accumulator will have to be FF and the Q register will be FB. That is, the combined AQ register must have the two's complement of the number, not just the Q register. Also, the remainder (A register) will have the sign of the number you are dividing into unless there is no remainder. Therefore, if you are dividing into a negative number the remainder, if there is one, will be in two's complement form.

In the following examples, we will use the same values with different signs to help point out the fact that the remainder has the sign of the number you are dividing into.

### Example 1: Dividing 5 by 2

Before: DIV 57 A Register = 00 Q Register = 05 Memory Location 57 = 02

After: DIV 57 A Register = 01 Q Register = 02 Memory Location 57 = 02

5 - 2 = 2 remainder 1

### Example 2: Dividing 5 by -2

Before: DIV 57 A Register = 00 Q Register = 05 Memory Location 57 = FE

After: DIV 57 A Register = 01 Q Register = FE Memory Location 57 = FE

5 - FE(-2) = FE(-2) remainder 1

### Example 3: Dividing -5 by 2

Before: DIV 57 A Register = FF Q Register = FB Memory Location 57 = 02

After: DIV 57 A Register = FF Q Register = FE Memory Location 57 = 02

FF FB (-5) - 2 = FE (-2) remainder FF (-1)

### Example 4: Dividing -5 by -2

Before: DIV 57 A Register = FF Q Register = FB Memory Location 57 = FE

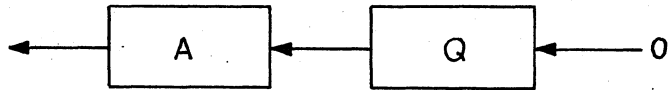
After: DIV 57 A Register = FF Q Register = 02 Memory Location 57 = FE

FFFB (-5) - FE (-2) = 2 remainder FF (-1)

## OB SLA,k Shift Left Arithmetic

This instruction shifts the combined AQ register left the number of places specified by the constant k. The constant k (operand) can be any value from 0 to FF but after 10 hex shifts the AQ register contains all zeros.

Zeros are used to fill in the Q LSB. The Q MSB is shifted into the A LSB. A pictorial diagram is below.



RDA26-440

Example:

Before: SLA 09 A Register = 0B Q Register = 42

After: SLA 09 A Register = 84 Q Register = 00

10 SRA,k Shift Right Arithmetic

This is used to shift the combined AQ register right the number of places specified by the constant k. The sign bit of the accumulator is shifted in from the left. In other words, if bit 7 (sign bit) of the A register is 0 then zeros will be shifted in. If bit 7 of the A register is 1 then ones will be shifted in.

Example 1:

Before: SRA 08 Accumulator = 05 Q Register = F3

After: SRA 08 Accumulator = 00 Q Register = 05

Example 2:

Before: SRA 08 Accumulator = FB Q Register = F3

After: SRA 08 Accumulator = FF Q Register = FB

The following is a short program that used some of the instructions you have learned. Read it carefully and keep track of the A Register and the Q Register after each instruction. In other words, write down what would be in the A and Q registers after the computer runs each step.

<u>MEMORY LOCATIONS</u>	<u>SYMBOLIC CODE</u>	
00.01	LDA 50	
02.03	LDQ 51	
04.05	DIV 52	
06.07	STA 60	
08.09	STQ 61	
0A.0B	LAN 53	c(50) = FF
0C.0D	LDQ 51	c(51) = F5
0E.0F	DIV 52	c(52) = 03
10.11	STA 62	c(53) = 01
12.13	STQ 63	
14.15	LDA 51	
16.17	SRA 08	
18.19	DIV 52	
1A.1B	STA 64	
1C.1D	STQ 65	
1E.1F	BST 00	

Here is the program again with the contents of the A and Q shown after each step. See if your's compares to this:

<u>MEMORY LOCATIONS</u>	<u>SYMBOLIC CODE</u>	<u>A REGISTER</u>	<u>Q REGISTER</u>
00.01	LDA 50	FF	00
02.03	LDQ 51	FF	F5
04.05	DIV 52	FE	FD
06.07	STA 60	FE	FD
08.09	STQ 61	FE	FD
0A.0B	LAN 53	FF	FD
0C.0D	LDQ 51	FF	F5
0E.0F	DIV 52	FE	FD
10.11	STA 62	FE	FD
12.13	STQ 63	FE	FD
14.15	LDA 51	F5	Fd
16.17	SRA 08	FF	F5
18.19	DIV 52	FE	FD
1A.1B	STA 64	FE	FD
1C.1D	STQ 65	FE	FD
1E.1F	BST 00	FE	FD

c(50) = FF  
 c(51) = F5  
 c(52) = 03  
 c(53) = 01

There are a few things that need to be said about the program. First, there are three routines that all do the same thing. This program divides -11 by 3. It does it three times, each in a different way. Steps 00-08 do it by first loading the A directly with FF, loading the Q with F5, and then dividing by 3. The second group, steps 0A-12, does it by loading A negative with 1 which puts FF into the A register. Steps 14-1c does it by loading the A with F5 (-11) and shifting it right. That shifts ones into the A because the sign bit was set. Therefore, it leaves FFs in the A register after the shift.

There are many ways to approach the same problem, as was just shown. No one way is better than another. A programmer should try to make his program as short as possible and still satisfy the requirements. The reason is that memory space in a computer is usually very precious. You need to take up as little space as possible with your program. This leaves room for data and other programs.

#### REVIEW QUESTIONS 1-6

1. Analyze the following programs and give the contents of the Accumulator and the Quotient after each step. If you have trouble, go to the computer, load the program and run it one instruction at a time.

Memory Location	Symbolic Code	Accumulator	Quotient
00,01	LDA 0A		
02,03	SRA 08		
04,05	DIV 0B		

Memory Location	Symbolic Code	Accumulator	Quotient
06,07	BST 00		
0A	57		
0B	03		

Memory Location	Symbolic Code	Accumulator	Quotient
00,01	LDA 0C		
02,03	MPY 0D		
04,05	DIV 0E		
06,07	SLA 08		
08,09	MPY 0D		
0A,0B	BST 00		
0C	6		
0D	3		
0E	2		

Memory Location	Symbolic Code	Accumulator	Quotient
00,01	LDA 14		
02,03	MPY 14		
04,05	SLA 08		
06,07	MPY 15		
08,09	SLA 08		
0A,0B	SUB 16		
0C,0D	SUB 16		
0E,0F	ADD 17		
10,11	STA 18		
12,13	BST 00		
14	05		
15	02		
16	06		
17	03		
18	00		

2. What are the contents of memory location 18 when this program stops?

### Branching and Indexing

#### 02 LAI,k Load Accumulator Immediate

This instruction will load the accumulator with the constant k. K can be any value between 0 and FF. Notice that no memory location is used for the data. K is the data, not the address of the data.

Example:

Before: LAI 4B Accumulator = 37 Memory Location 4B = 00

After: LAI 4B Accumulator = 4B Memory Location 4B = 00

Notice that 4B was put into the accumulator, not the contents of memory location 4B.

#### 80 RAO,m,x Replace and Add One

This instruction will load the accumulator with the contents of memory location m, add 1 to it, and store it back into memory location m. Notice here that the contents of the A register will be destroyed.

Example:

Before: RAO 21 Accumulator = 32 Memory Location 21 = 42

After: RAO 21 Accumulator = 43 Memory Location 21 = 43

#### 88 RSO,m,x Replace and Subtract One

This instruction loads the accumulator with the contents of memory location m, subtracts one from it, and stores the results back into memory location m. This instruction will destroy the contents of the accumulator.

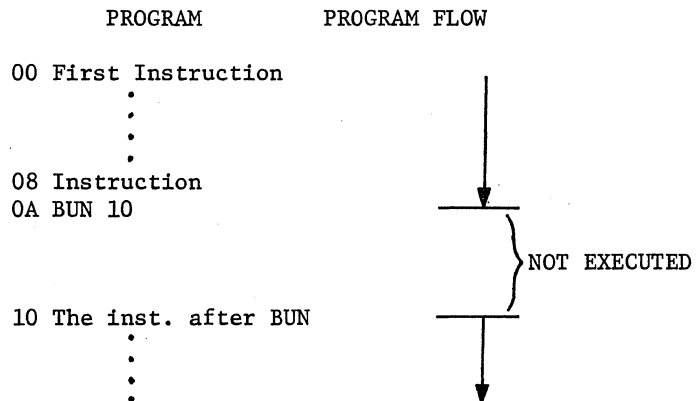
Example:

Before: RSO 3E Accumulator = 10 Memory Location 3E = 50

After: RSO 3E Accumulator = 4F Memory Location 3E = 4F

#### 90 BUN,m,x Branch Unconditionally

This instruction will cause the sequence of the program to jump to the instruction at memory location m. In other words, the next instruction to be performed will be the instruction in memory location m. Below is a diagram of what this instruction will do.



Steps 00 through 08 will be performed sequentially, that is one right after the other. Then the BUN 10 will be performed at step 0A. This will cause control of the computer to transfer to the instruction at step 10. The steps between steps 0A and 10 are jumped over and they are not performed. Before we can talk about some other branching instructions, we need to know about condition codes.

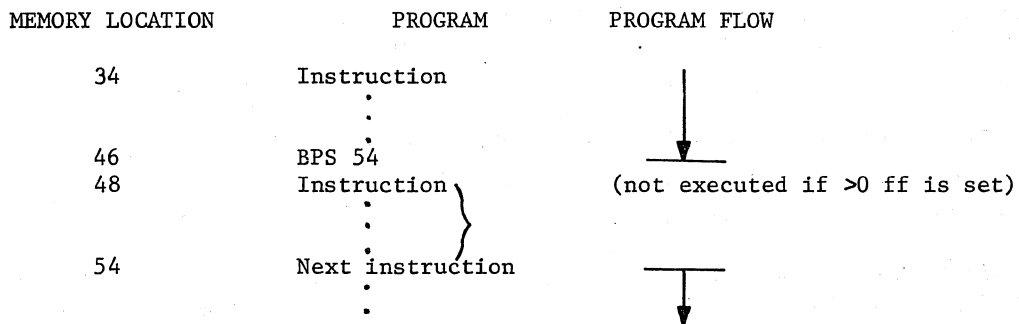
There are four flip-flops that the computer uses as condition codes. These usually give the status of the accumulator except after a divide, in which case the status of the Quotient will be given. The condition code set after a multiply instruction indicates the status of the combined AQ register.

If the accumulator (or quotient in the case of a divide) is zero, the =0 flip-flop will be set. If the result is negative, the (less than zero) flip-flop will be set. After an add or subtract, if the instruction caused an overflow or borrow because of the limited size of the accumulator, the carry flip-flop will be set. During a divide, the condition codes will be set by checking the Q register. In other words, if Q = 0, the = 0 flip-flop will be set, if Q<0 then <0 will be set, and if Q>0 (greater than zero) then >0 will be set. The computer can check these condition codes and make decisions depending upon their state. All decisions made by the computer, are made by checking either these conditions or others that will be discussed later.

**A8 BPS,m,x Branch on Positive**

This instruction will cause the sequence of the program to change to the instruction at memory location m if the >0 condition code is set. In other words, if the condition code >0 ff is set, the normal step-by-step process will be altered and the process will start back up at memory location m.

Example:



In this example, the program will step sequentially up to step 46. It is at this if the condition code >0 is set then the next instruction will be at step 54. If >0 is not set, the next instruction will be step 48. The computer can decide, by checking the >0 condition code, whether to do step 54 next or to do step 48.

**B0 BZE,m,x Branch on Zero**

This instruction will cause the sequence of the program to change to the instruction at memory location m if the condition code =0 is set. If the condition code =0 is set, this instruction is performed. The normal step-by-step process will be changed and the process will be started again at memory location m. This instruction allows the computer to make decisions on what to do by checking for an answer being zero. Remember, the condition codes are not set by the condition of the accumulator only.

**B8 BNG,m,x Branch on Negative**

This instruction is used to change the sequence of the program to memory location m if the condition code <0 is set. If the <0 condition code is set, then the normal flow



of the program will be changed. The flow will pick up again at memory location m. This allows the computer to make decisions based on the <0 condition code.

Now that you have learned seven more instructions, you can see that the branching instructions are what gives the computer its "brain". They allow logical decisions to be made which is essential to figuring most problems. These instructions are the major difference between a computer and a calculator.

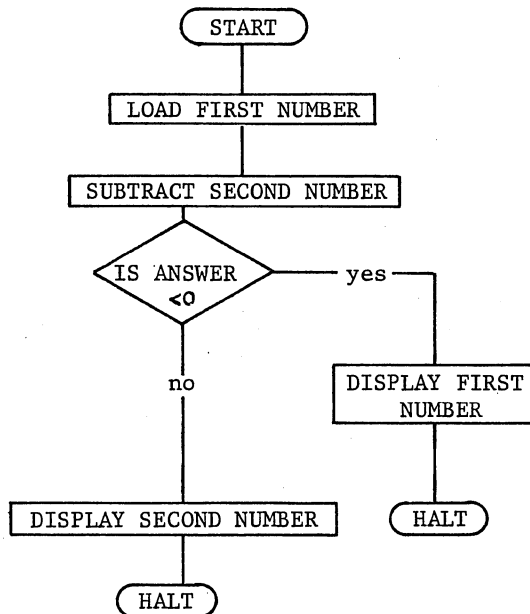
The following is a program for finding the larger of two numbers. There are many ways to work this problem. This is just one of the ways.

```

00 LDA 20
02 SUB 21
04 BNG 0A
06 LDA 20
08 BST 00
0A LDA 21
0C BST 00

20 first number
21 second number

```



RDA26-441

Notice that subtracting and using a branch on negative could tell us which number was largest. If the answer is negative, then the second number must have been larger than the first.

#### REVIEW QUESTIONS 1-7

1. Match the following instructions with their functions.

- \_\_\_ Unconditionally causes a change in the sequence of a program.
  - \_\_\_ Adds one to the memory location specified.
  - \_\_\_ Cause a change in the sequence of a program if the "greater than zero" condition code is true.
  - \_\_\_ Subtracts one from the memory location specified.
1. LAI
  2. RAO

3. RSO
4. BUN
5. BPS
6. BZE
7. BNG

Answer the following questions in your own words.

2. What is meant by the term "unconditional" branch?
3. What is meant by the term "conditional" branch?
4. What happens if the condition being checked by a conditional branch instruction is a one (true)?
5. What happens if the condition being checked by a conditional branch instruction is a zero (false)?
6. Analyze the following program. Give the contents of the Accumulator. Be sure and leave the registers blank beside any instructions not performed. If you have trouble, go to the computer, load the program and run it one instruction at a time.

Memory Location	Symbolic Code	Accumulator
00,01	LDA 0C	
02,03	ADD 0D	
04,05	SLL 04	
06,07	BNG 0A	
08,09	BST 00	
0A,0B	BST 01	
0C	05	
0D	06	

7. Which program step did not get executed?
8. If the BNG 0A at step 06 was changed to BPS 0A which step would not get executed?
9. Analyze the following program and give the contents of the Accumulator and Quotient registers after each step. If you have trouble, go to the computer, load the program and run it one instruction at a time.

Memory Location	Symbolic Code	Accumulator	Quotient
00,01	LDA 20		
02,03	SRA 08		
04,05	DIV 21		
06,07	BNG 0C		
08,09	STQ 22		
0A,0B	BST 00		
0C,0D	LDA 20		
0E,0F	STA 22		
10,11	BST 01		
20	0A		
21	02		
22	00		

10. If location 06 was changed to a BZE, where would this program stop?

11. If memory location 20 had FC in it when this program was run, where would the program stop?

12. If memory location 20 had FC in it when this program was run, what would be in memory location 22 when the program stopped?

19 AND,k AND

This instruction will AND the contents of the accumulator with the constant k. The results will be put into the accumulator. A logical AND of two numbers works just like an AND gate. In other words, both inputs must be one to get one output.

Example:

Before: AND 35 Accumulator = 61(16) 0110 0001

After: AND 35 Accumulator = 21(16) 0010 0001

If we match the two numbers in binary maybe you can see what happens.

Accumulator = 0110 0001

We ANDED with 0011 0101

Result 0010 0001

Both bit 0's are ones. AND says both A and the number must be ones to get a one. So if both A and the number has a one in the same position then a one will be put back into the Accumulator in that position.

1A IOR,k Inclusive OR

This instruction will do a bit by bit inclusive OR of the Accumulator and the constant k putting the results back into the accumulator. In other words, it works like an OR gate. If either the A or the constant has a one in that bit position the result will have a one there.

Acc = 1001 0101  
k = 0101 0011  
Res = 1101 0111

1B XOR,k Exclusive OR

This instruction does a bit by bit exclusive OR of the accumulator and the constant k and puts the results back into the accumulator. An exclusive OR means if one or the other (but not both) has a one in that bit position then the result will have a one in that bit position.

Acc = 0101 1100  
k = 0100 1011  
Res = 0001 0111

Earlier you were told not to worry about the instructions that had "x" placed after them. You were told that it meant the instruction could be indexed and indexing would be explained later. Now you will learn indexing.

Indexing is nothing more than a convenient way to modify or change a memory address. An instruction that has been indexed will use the memory location m as a starting point. Then it will add to this address the contents of the index register. This will give a result which will be the new memory location. Let's go through an example to help explain it.

LDA,x 30

This says to load the accumulator with the contents of some memory location. Because of the x, we can not be sure of what location we will be loading until we know what is in the index register. If the index register has 00 in it then we will load the accumulator with the contents of location 30. How did I get that? 30 (the memory location) + (00 the contents of the index register) = 30 (the address we will load).

If the index register has 05 in it then we will load the accumulator with the contents of memory location 35. 30 + 5 = 35. The index register can hold any value 00 through FF.

In order for the computer to know you want to index, you will have to modify the hexadecimal code for the instruction. If you remember how to modify codes to get into "high memory," this process is the same. Instead of adding 1, 2, or 3 as we did to get into high memory this time we will add a 4 to get indexing.

SYMBOLIC	HEX CODE
LDA,x 23	24 23

You can index high memory by adding the 1, 2, or 3 and adding the 4.

SYMBOLIC	HEX CODE
LDA,x 23B	26 3B

Refer to the Instruction Repertoire on page 111.

Naturally if we want to be able to use this in any programs we must be able to work with the index register. That is, we must be able to load the index with a known value. The following instructions work with the index register.

#### 12 LXI,k Load Index Immediate

This instruction will load the index register with the constant k. No other registers will be affected.

Example:

Before: LXI 07 index register = 00

After: LXI 07 index register = 07

#### 50 STX,m,x Store Index

This instruction is used to put the contents of the index register into memory location m. Whatever was in memory location m will be destroyed and replaced by the contents of the index register. The index register will not be affected.

Example:

Before: STX 42 Index register = 16 Memory location 42 = 66

After: STX 42 Index register = 16 Memory location 42 = 16

#### 03 INX,k Increase Index

This instruction is used to change the index register by adding a value to it. If the constant k is positive the contents of the index register will be increased. If k is negative the contents of the index register will be decreased.

Example:

Before: INX 03 Index Register = 05

After: INX 03 Index Register = 08

The positive 3 was added to the 5 given for a total of 8.

Example:

Before: INX FE Index register = 05

After: INX FE Index register = 03

The FE, which is negative 2, is added to the 05. When a negative is added, it is the same thing as subtracting. Therefore,  $-2 + 5 = 3$ .

While using the INX instruction we can increment or decrement the index register.

#### C8 BXZ,m,x Branch on Index Equal to Zero

Branch on Index Equal to Zero. This instruction will cause a change in sequence of the program if the index register is equal to zero. This instruction does not check the condition codes, it checks the index register. If the index register is zero, then the next instruction to be performed will come from memory location m instead of the location following the BXZ instruction.

Below is a short program that uses indexing. No explanation of why you would want indexing has been offered. Remember, how these instructions are used is up to the programmer. The following program uses these instructions to hunt through memory and count the words that have bit 4 set.

```

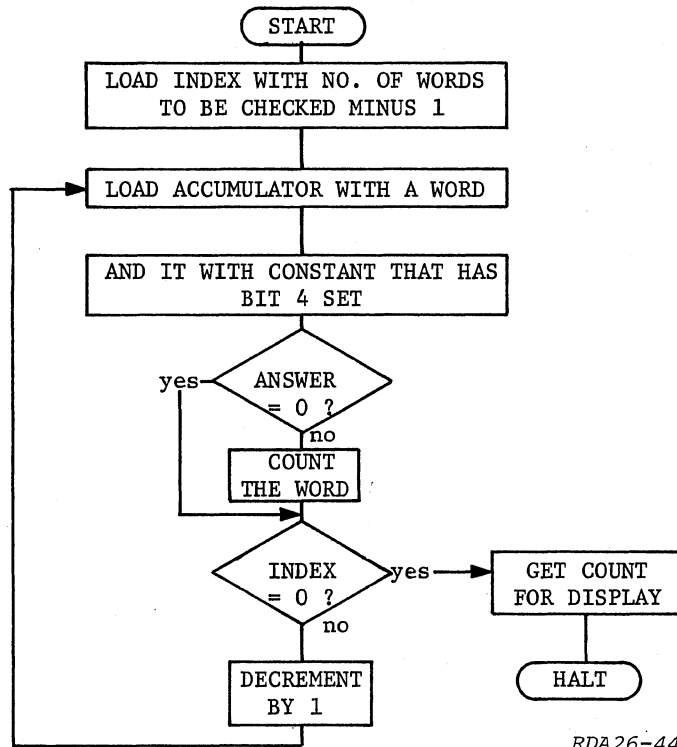
00 LXI  04
02 LDA,X 30
04 AND  10
06 BZE  0A
08 RAO  35
0A BXZ  10
0C INX  FF
0E BUN  02
10 LDA  35
12 BST  00

```

```

c(30) = 27
c(31) = 3A
c(32) = 75
c(33) = FE
c(34) = 61
c(35) = 00

```



RDA26-442

This program will check locations 30-34 and see how many of them have bit 4 set. The answer will be in location 35. Notice that you must execute steps 02-0E five times before the program is finished.

The program starts by loading the index register with one less than the number of locations to be checked. Then it loads the accumulator with 30 indexed. The first time this will actually load location 34. ANDing the accumulator with 10 will either leave a 10 or a 0 in the accumulator. If a 0 is left then bit 4 of the word in memory is clear. If 10 is in the accumulator then bit 4 set so we count that word. Then we check to see if we are finished by checking the index register for zero. If it is not zero, we subtract 1 from it and go through the loop again. If it was zero then we display the count and halt.

#### REVIEW QUESTIONS 1-8

1. Match the following instructions with their functions.

- \_\_\_ Used to store the contents of the Index Register into the memory location specified.
- \_\_\_ Used to change the sequence of the program if the Index Register contains zero.
- \_\_\_ Used to Inclusive OR the contents of the Accumulator with a constant.

\_\_\_ Used to increase the Index Register by the number specified.

\_\_\_ Used to load the Index Register with the constant specified.

\_\_\_ Used to AND the contents of the Accumulator with a constant.

\_\_\_ Used to Exclusive OR the Accumulator with a constant.

1. AND

2. IOR

3. XOR

4. LXI

5. STX

6. INX

7. BXZ

2. Analyze the following program. Give the contents of the Accumulator, Quotient Register, and Index Register after each step. If you have trouble, go to the computer, load the program and run it one instruction at a time.

	ACCUMULATOR	QUOTIENT	INDEX
00,01 LAI	00		
02,03 STA	1C		
04,05 LXI	04		
06,07 LDA,X	1F		
08,09 ADD	1C		
0A,0B STA	1C		
0C,0D BXZ	12		
0E,0F INX	FF		
10,11 BUN	06		
12,13 LDA	1C		
14,15 SRA	08		
16,17 DIV	1D		
18,19 STQ	1E		
1A,1B BST	00		
1C	00		
1D	05		

	ACCUMULATOR	QUOTIENT	INDEX
1E	00		
1F	05		
20	07		
21	06		
22	02		
23	0A		

3. Which steps in this program make up the loop?

NOTE: It might be helpful to draw a flow chart of the program.

4. How many times will the loop be performed before the program will get out of the loop?

5. Analyze the following program. Give the contents of the Accumulator and Index Register after each step. If you have trouble, go to the computer, load the program and run it one instruction at a time.

Memory Location	Symbolic Code	Accumulator	Index
00,01	LAI 00		
02,03	STA 18		
04,05	LXI 08		
06,07	LDA,X 19		
08,09	XOR 36		
0A,0B	BZE 12		
0C,0D	BXZ 16		
0E,0F	INX FF		
10,11	BUN 06		
12,13	STX 18		
14,15	BST 00		
16,17	BST FF		
18	00		
19	30		
1A	B1		



Memory Location	Symbolic Code	Accumulator	Index
1B	B2		
1C	33		
1D	B4		
1E	35		
1F	36		
20	B7		
21	B8		
22	39		

6. How many times does this program go through the loop?

7. If the constant in the XOR at step 08 was changed from a 36 to a B8, what would be stored in memory location 18 after the program runs?

#### Input/Output Programming

##### F8 FLS,k Flag Set

This instruction is used to set a flip-flop called the flag. This can be used to remind us of a condition that existed earlier in our program. In other words, you could check a condition and if that condition was true, set the flag. Then later in the program you can check the flag to determine prior conditions.

##### 28 FLC,k Flag Clear

This instruction is used to clear the flag. It is used like the FLS. Both the FLS and FLC have a constant k. This constant has no meaning for these instructions.

##### 0A SKF,k Skip on Flag

This instruction is used to check the condition of the flag. If the flag is set, this instruction will cause the computer to skip (jump over) the next k sequential instructions. Remember, each instruction takes 2 memory locations. The skip instruction causes the computer to skip k instructions. If k is 1 then the computer will skip 1 instruction or 2 memory locations.

Example: 00 FLS 00  
02 SKF 01  
04 BST 00  
06 FLC 00  
08 BST 00

In this program the SKF 01 will cause the computer to skip over the BST 00 at location 4. The next instruction to be performed would be the FLC at location 06. All the skip instructions work this way. Their only difference is what they check.

09 SKS,k Skip on Sense

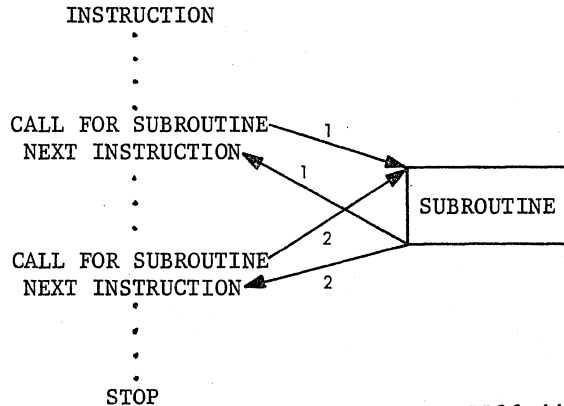
This instruction causes the computer to skip k instructions if the sense switch is set (pushed). This allows the operator to make some decisions for the computer.

08 SKI,k Skip on Interrupt

This instruction causes the computer to skip k instructions or 2k memory locations if the interrupt bit is set. This bit can be set from the teletype by pressing the control button and the letter I. You will learn more about that in the next series of instructions.

A subroutine is any group of instructions that performs a specific task. Programs are usually made up of different subroutines. If a subroutine is only needed once in a program it would be just as easy to put the instructions in the place where they are needed. If a subroutine is needed many times within a program it might be easier to put the subroutine down once and branch to it each time it is needed in order to save memory. The problem with this approach is that the subroutine must branch back to where it was "called" from.

Example:



RDA26-443

Notice that the subroutine must jump back to 2 different places. That makes it difficult to program because the subroutine doesn't know where it was called from.

AO BSB,m,x Branch to Subroutine

This instruction causes a branch unconditionally (90) to be put in memory location m. Then it puts the contents of the program address register (the address of the next instruction) into memory location m + 1. Then it unconditionally branches to the instruction at m + 2.

By storing the BUN to the program address in memory locations m and m + 1, the subroutine has a method of finding its way back to where it was called from.

Example:

```
00 BSB 10
02 BST 00
```

```

10 Nothing is put here
12 ' ' ' '
14 ' ' ' ' } INSTRUCTIONS
16 ' ' ' '
18 BUN 10

```

When step 00 is performed, the BSB will store a BUN 02 in memory location 10. That is why location 10 has nothing put there. Then the program will branch to 12 to start the subroutine. The subroutine can be as long or short as needed. Then at the end of the subroutine (step 18 in this case) there is a branch back to the beginning of the subroutine (10) which will branch the program back to location 02. Working this way the subroutine can be called from anywhere in memory and when it is finished it will take the program back to where it was called from.

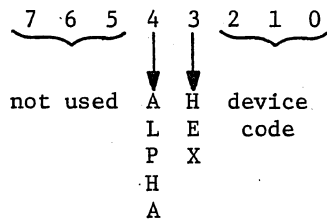
No computer can really be useful without some way of communicating with other devices. It must be able to communicate with humans. So far, you have done this by way of the switches on the front control panel. The group of instructions that allow the computer to communicate with the teletypewriter are referred to as I/O instructions.

11 OCD,k Output Command

This instruction is used to set up the interface between the peripheral equipment (teletypewriter in our case) and the computer. This does not start any data transfers, but must be executed before any I/O can be performed. The constant in this case has two functions. First it selects whether the data transfers will be in Hex or Alpha mode. Hex mode means that each transfer will be considered a number. In alpha mode each transfer will be considered an ASCII character. The mode depends on the OCD constant. Bit 3 of the constant specifies Hex mode. Bit 4 of the constant specifies Alpha mode.

If neither are set, you cannot determine what the mode will be and you might get something you did not want.

The second function of the constant is to determine which I/O device is used. In our system we only have one I/O device and it is not wired to detect any specific count. Bits 0-2 are used to determine which device will be used. Since our teletype is not wired to decode it, any number (0-7) can be put in these 3 bits. If our computer had more than one I/O device, each would have a designated number. The constant k for the OCD is set up like this:



Remember, all I/O operation must be preceded by the OCD instruction. If you do not change modes or devices within your program then one OCD will be enough for the program. Each time you want to change, you must use another OCD.

01 LCI,k Load Countdown Immediate

This instruction is used to put the constant k into the countdown register. The countdown register is used to control I/O transfers. The countdown always has a count of one less than the number of words to be transferred. If you load the countdown with a 4 before an I/O operation, then 5 words will be transferred. If you forget to load the countdown and it has 00 in it then 1 word will be transferred.

#### EO RDB,m,x Read Data Block

This instruction will cause data to be stored into memory location m. The I/O device and the mode (alpha or hex) are determined by the OCD. The first word read into the computer will be stored in memory location m. The second word will go into memory location m + 1 and so on. This continues until the countdown register is equal to zero. If the countdown starts with a 10(16) in it, then the computer will read in 11(16) words storing them in memory location m through m + 10(16). The tape reader will supply the data to be read when a tape is loaded. If there is no paper tape loaded, the input will be from the keyboard. Notice that there is no way of designating whether you want the input from the paper tape reader or the keyboard. You will have to tell the person using your program what is expected from him.

Often times the programmer may not know how much data is to be read. If people are to enter their names, you do not know if the name will be Bob or Robert. You don't know how many words you want so you cannot load the countdown. The RDB will wait until the countdown is zero. If you put 05 in the countdown and do a RDB then the computer will just sit there until it has 6 inputs. That is all right as long as data is going to be a certain length each time such as social security numbers. This creates a problem in the case of a name. To get around this problem we have another read instruction.

#### E8 RDI,mx Read until Interrupt

This instruction will cause the computer to read from the device selected by the OCD. The computer will read until the interrupt bit is set. Data will be stored in memory location m,m+1, m + 2, and so on until the interrupt bit is set. The countdown has no effect on this read. The interrupt can be set by typing a Control I on the teletype. Now the machine can read a human input but it must also be able to write.

#### DO WDB,mx Write Data Block

This instruction will cause the computer to write the data starting in location m to the selected device. The device and mode are selected by an OCD. The number of words written will be determined by the count in the countdown just as it is for the TDB. Formatting your data to be written out will probably be the hardest part of writing programs that use I/O. You must put every letter, space, carriage return, and line feed into memory. In other words, the format by which data is printed is totally dependent on the programmer when in Alpha mode. In Hex mode, the computer converts each memory location to Hex and writes it out. It puts colons after each number and puts 16 numbers across on a line. There is no programmer formatting in Hex mode.

#### 30 LCC,m,x Load Consecutive

This instruction takes the data in memory location m and stores it in memory location m + 1. No registers are affected and memory location m is not affected.

#### Example:

Before: LCC 30 Memory Location 30 = A6 Memory Location 31 = 2C

After: LCC 30 Memory Location 30 = A6 Memory Location 31 = A6

In a small loop this instruction could be used to initialize a block of memory to a certain value.

#### 00 SST, Sense Status

This instruction causes a status word sent by the selected I/O device to be put into the accumulator. The teletype does not have a status word so this instruction cannot be

used with it. A status word tells the computer things like parity error, out of paper, end of tape and so on. Again, this is of no use to us since the teletype has no status word.

CO BNC,m,x Branch on no Carry

This instruction will cause a change in the program sequence if the carry flip-flop is clear. The carry is set by an overflow in addition or an underflow in subtraction (borrow from a nonexisting bit position). This is what sets the carry. The BNC checks for no carry or the carry clear.

FO MNI,m,x Manual Input

This instruction halts the computer to allow the operator to put a value into the I register (input). The word in the input register will be stored in memory location m when the operator pushes the start pushbutton. This will continue until the countdown register is zero. Each time the start is pushed, the word in the input register will be stored in memory location m, m + 1, m + 2, and so on until the countdown is equal to zero.

D8 MNO,m,x Manual Output

This instruction will cause the contents of memory location m to be displayed in the I register. Each time the start is pushed the next memory location will be displayed. This will continue until the countdown is equal to zero. Remember, as in the RDB and WDB, the number of words transferred with the MNI and MNO is always one more than the count in the countdown.

You have covered the block diagram of a computer. This 5 block diagram will pertain to any computer that you may come across. If, while you are studying other machines you will try to break them down into their five basic blocks, you will find that learning the other machines will be easier.

You also studied the block diagram of the COM-TRAN TEN. This will be a big help in the blocks to come.

Last you studied programming and the instructions of the COM-TRAN TEN. The purpose of this course is to make you a maintenance person, not a programmer. It is very necessary that you know the instructions of any computer on which you work. If you do not know what the computer is supposed to do when it decodes an instruction, you cannot tell if the computer did it right or wrong. That is an essential part of maintenance.

#### REVIEW QUESTIONS 1-9

1. Match the instructions to their functions.

- \_\_\_ Causes the computer to skip the number of instructions specified if the Sense Switch is set.
- \_\_\_ Causes the computer to store a branch unconditional at the address specified and then branches to the instruction after the address is specified.
- \_\_\_ Used to clear the Flag flip-flop.
- \_\_\_ Causes the computer to skip the number of instructions specified if the Flag is set.
- \_\_\_ Used to set the Flag flip-flop.

\_\_\_\_ Causes the computer to skip the number of instructions specified in the Interrupt flip-flop is set.

1. FLS
2. FLC
3. SKF
4. SKS
5. SKI
6. BSB

2. Analyze the following program. Give the contents of the Accumulator after each step. Any steps not performed leave blank. If you have trouble, go to the computer, load the program and run it one instruction at a time.

Memory Location	Symbolic Code	Accumulator
00,01	FLS 00	
02,03	LDA 1C	
04,05	BNG 10	
06,07	ADD 1D	
08,09	STA 1E	
0A,0B	SKF 01	
0C,0D	BSB 14	
0E,0F	BST 00	
10,11	FLC 00	
12,13	BUN 06	
14,15	00 00	
16,17	LAN 1E	
18,19	STA 1E	
1A,1B	BUN 14	
1C	10	
1D	2E	
1E	00	

3. If the contents of memory location 1C was changed to F5 and the program was run, what would be stored in memory location 1E after the program was finished?

4. Match the following instructions with their functions.

- \_\_\_ Used to read the amount of data specified by the Countdown Register from the selected device (teletypewriter) into memory starting at the address specified.
- \_\_\_ Used to select a device and tell it what mode (Hex or Alpha) to work in.
- \_\_\_ Used to write the amount of data specified by the Countdown Register from memory to the specified device (teletypewriter).
- \_\_\_ Used to read data from the specified device (teletypewriter) into memory until an interrupt is detected.
- \_\_\_ Used to load the Countdown Register with the number specified.

1. OCD
2. LCI
3. RDB
4. RDI
5. WDB

5. Analyze the following program. Give the written output that would be printed after the program has run. If you have trouble, go to the computer, load the program and run it. Be sure and turn the teletype to "line" when running the program.

Memory Location	Symbolic	Remarks		
00,01	OCD 10	Sets Alpha Mode		
02,03	LCI 09	One less than what is to be printed		
04,05	WDB 10	Write Data		
06,07	LCI 0D	One less than what is to be printed		
08,09	WDB 1F	Write Data		
0A,0B	BST 00			
			ASCII CHARACTER	
			HEX CODE	
10	CARRIAGE RETURN		8D	
11	LINE FEED		0A	
12	T		D4	
13	H		48	
14	I		C9	
15	S		53	What will this program print out?
16	Space		A0	
17	I		C9	
18	S		53	
19	Space		A0	
1A	F		C6	
1B	U		55	
1C	N		4E	
1F	A		41	
20	Space		A0	
21	T		D4	
22	E		C5	
23	S		53	

	ASCII CHARACTER	HEX CODE
24	T	D4
25	Space	A0
26	P	50
27	R	D2
28	O	CF
29	G	47
2A	R	D2
2B	A	41
2C	M	4D

6. Match the following instructions with their functions.

\_\_\_ Used to manually input information through the Input Register.

\_\_\_ Used to check the status of the selected I/O device.

\_\_\_ Used to move data from the specified memory location into the next memory location.

\_\_\_ Will cause a change in the sequence of the program if the Carry flip-flop is clear.

1. LCC

2. SST

3. BNC

4. MNI

5. MNO

#### PROCEDURES FOR WRITING PROGRAMS

In this section you will learn basic techniques used in developing a program for the machine to use. There are distinct steps which you must go through to write a program. You will learn these steps and use them.

Before you get started there are some terms you need to know. These will be used throughout the remainder of the block. As they are used they will be explained, but here is a quick definition of some of the terms.

**Instruction** A code that the computer can decode, and because of it, can perform a preset sequence of events.

**Program** A sequence of instructions that perform a specific job to reach the desired end.

**Subroutine** A sequence of instructions that performs a specific task of a job. Sub-routines are used within programs.

**Flow-Chart** A diagram that visually shows what needs to be done within the program to get a job done.

There are seven distinct steps involved in writing a program. You will not always do all seven steps on paper. Some of these steps will be done mentally. In fact,



sometimes you may not even realize that you did the step. Each step is performed whether you realize it or not. The first step is to analyze the problem.

Analyzing the problem is usually the most difficult part of writing a program. If it is done properly the other six steps will be easy.

This step involves finding out exactly what the desired results of the program are to be. We all know that we want our answer, but do we want it printed out, and, if so, in what format? Is our answer going to be used by other programs? If so, where are we to put our answer so the other programs can find it? Questions like these must be answered. Also in this step, we must determine what we have to work with. If we are to design a building, we must be told the number of rooms, the length of each wall, and the height of the building. We've got to find out what is known in our problem. We must also determine where these knowns are stated, or if they are going to come from I/O device, such as the Teletypewriter.

A good understanding of the problem is a must in determining how to solve it.

Let it be noted now that a program is never written for just one situation. You never write a program that is to be used only once. It would be ridiculous to write a program that will take the square root of 37. It would not be so ridiculous to write a program that will take the square root of any number. After a programmer has taken all the time to figure out how to solve the problem, he will have the answer for any value to the one time program. By the time the program is written he will know the square root of 37. If the program, however, will find the square root of any number, and you do not know the square root of all numbers, then the program has value.

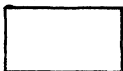
The programs and programming problems that you will have in this course are to help teach you what the machine is doing and how it does it.

The second step in writing a program is to select a method to solve the problem. This means that the programmer must figure out how he can take the knowns or the given information, and come up with the desired answer. He must be able to work the problem himself. That does not mean he must be a mathematician or an engineer, but simply means he must be able to look at the formula and determine what it takes to do that formula. There is normally a number of different ways to solve a given problem. The programmer must be able to look at the problem and determine which way is the best way.

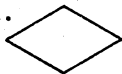
After this has been determined, the programmer will go on to the third step. This step is to develop a flow chart. A flow chart is a pictorial representation of how you are going to solve the problem. In flow charting there are a few basic symbols you will need to know. These are not all of the symbols, but these are the major ones.

A rectangle is used to show an action that is to take place. This action might be adding two numbers, or reading something from the teletypewriter, or taking something out of memory.

The symbol looks like this.



Another symbol we will use is a diamond.



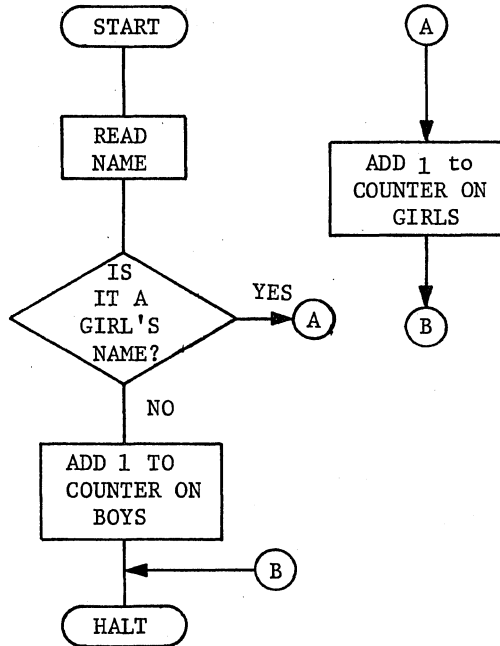
This will be used to show when a question is to be asked and the different paths that can be followed depending on the answer to the question. It is called a decision block.

The next symbol is an oval. It is called a terminal symbol and it is used to show starting and stopping points of the program.



Last is the connector symbol. ○

This symbol is used when it would be confusing to draw a line or when you go from one page to the next. Label the connector out and the connector in the same way to show that these points are the same. An example of a flow chart using these symbols is shown below.



RDA26-444

Remember, a flow chart is a programmer's tool. How detailed the flow chart is depends on the programmer, but the more detailed it is the easier it will be to write the program.

The fourth step in writing a program is to write the program. Using the flow chart, the programmer must write the instructions that will cause the action the flow chart describes. For us this means write the program using the symbolic name for the instructions that the COM-TRAN TEN uses. You can see that the more detailed the flow chart the easier this step will be.

The fifth step is to put the symbolic coding into machine language. This is usually done by the programmer putting his symbolic program on punched cards and letting the computer code it into machine language for him, but, in our case, we will do this by looking up the hexadecimal code for each instruction.

The sixth step is to test the program. Here you will load the program into memory and run it to see if it gives the correct answer. This means that you will have to use known values so you will know what the answer is supposed to be.



e. Write the program; then code the program for loading in the COM-TRAN TEN Computer.

f. Call the instructor to check your work. \_\_\_\_\_

ALPHABETICAL  
SUMMARY OF INSTRUCTIONS

60	ADD,m,x	ADD (ACC) + (m) → (ACC)	12	LXI,k	Load INDEX immediate k → (X)
19	AND,k	AND k AND (ACC) → (ACC)	F0	MNI,m,x	Manual input (INPUT) → (B) → (m)
C0	BNC,m,x	Branch on No Carry CARRY = 0 → m → (PA)	D8	MNO,m,x	Manual Output (m) → (B) → (INPUT)
B8	BNG,m,x	Branch on Negative <0 → m → (PA)	70	MPY,m,x	Multiply (ACC) * (m) → (AQ)
A8	BPS,m,x	Branch on Positive >0 → m → (PA)	11	OCD,k	Output Command k → external device
A0	BSB,m,x	Branch to Subroutine 90 (BUN) → (m) (PA) → (m + 1) m + 2 → (PA)	80	RAO,m,x	Replace Add One (m) + 1 → (m)
98	BST,m,x	Branch & Stop m → (PA) & STOP	E0	RDB,m,x	Read Data Block Data → (m)
90	BUN,m,x	Branch Unconditional m → (PA)	E8	RDI,m,x	Read until Interrupt Data → (m)
C8	BXZ,m,x	Branch on INDEX Zero (X) = 0 → m → (PA)	88	RSO,m,x	Replace Subtract One (m) - 1 → (m)
B0	BZE,m,x	Branch on Zero = 0 → m → (PA)	0A	SKF,k	Skip on FLAG k instructions
78	DIV,m,x	Divide (AQ) ÷ (m) → (Q) remainder → (ACC)	08	SKI,k	Skip on Interrupt k instructions
28	FLC,k	FLAG clear 0 → FLAG bit	09	SKS,k	Skip on Sense k instructions
F8	FLS,k	FLAG set 1 → FLAG bit	0B	SLA,k	Shift LEFT Arithmetic (AQ) to left k places
03	INX,k	Increase INDEX (X) + k → (X)	13	SLL,k	Shift LEFT Logical (ACC) to left k places
1A	IOR,k	Inclusive OR k OR (ACC) → (ACC)	10	SRA,k	Shift RIGHT Arithmetic (AQ) to right k places
02	LAI,k	Load ACCUMULATOR immediate k → (ACC)	18	SRL,k	Shift RIGHT Logical (ACC) to right k places
38	LAN,m,x	Load ACCUMULATOR Negative - (m) → (ACC)	00	SST,k	Sense Status 8-bit status word → (ACC)
30	LCC,m,x	Load Consecutive (m) → (m + 1)	48	STA,m,x	Store ACCUMULATOR (ACC) → (m)
01	LCI,k	Load COUNTDOWN immediate k → (C)	58	STQ,m,x	Store QUOTIENT (Q) → (m)
20	LDA,m,x	Load ACCUMULATOR (m) → (ACC)	50	STX,m,x	Store INDEX (X) → (m)
40	LDQ,m,x	Load QUOTIENT register (m) → (Q)	68	SUB,m,x	Subtract (ACC) - (m) → (ACC)
			1B	XOR,k	Exclusive OR k EOR (ACC) → (ACC)
			D0	WDB,m,x	Write Data Block (m) → external device

NUMERICAL  
SUMMARY OF INSTRUCTIONS

00	SST,k	Sense status 8-bit status word → ACC	60	ADD,m,x	ADD (ACC) + (m) → (ACC)
01	LCI,k	Load COUNTDOWN immediate k → (C)	68	SUB,m,x	SUBtract (ACC) - (m) → (ACC)
02	LAI,k	Load ACCUMULATOR immediate k → (ACC)	70	MPY,m,x	Multiply (ACC) * (m) → (AQ)
03	INX,k	Increase INDEX (INDEX) + k → (INDEX)	78	DIV,m,x	DIVide (AQ) ÷ (m) → (Q) remainder → (ACC)
08	SKI,k	Skip on INTERRUPT k instructions	80	RAO,m,x	Replace ADD One (m) + 1 → (m)
09	SKS,k	Skip on SENSE k instructions	88	RSO,m,x	Replace Subtract One (m) - 1 → (m)
0A	SKF,k	Skip on FLAG k instructions	90	BUN,m,x	Branch UNconditional m → (PA)
0B	SLA,k	Shift LEFT Arithmetic (AQ) to left k places	98	BST,m,x	Branch & Stop m → PA & STOP
10	SRA,k	Shift RIGHT Arithmetic (AQ) to right k places	A0	BSB,m,x	Branch to Subroutine 90 (BUN) → (m) (PA) → (m + 1) (m + 2) → (PA)
11	OCD,k	Output Command k → external device	A8	BPS,m,x	Branch on Positive >0 → m → (PA)
12	LXI,k	Load INDEX immediate k → (INDEX)	B0	BZE,m,x	Branch on Zero = 0 → m → (PA)
13	SLL,k	Shift LEFT Logical (ACC) to left k places	B8	BNG,m,x	Branch on Negative <0 → m → (PA)
18	SRL,k	Shift RIGHT Logical (ACC) to right k places	C0	BNC,m,x	Branch on No Carry CARRY = 0 → m → (PA)
1A	IOR,k	Inclusive OR k OR (ACC) → (ACC)	C8	BXZ,m,x	Branch on INDEX Zero (INDEX) = 0 → m → (PA)
19	AND,k	AND k AND (ACC) → (ACC)	D0	WDB,m,x	Write Data Block (m) → external device
1B	XOR,k	Exclusive OR k EOR (ACC) → (ACC)	D8	MNO,m,x	Manual Output (m) → (INPUT) & (BUFFER)
20	LDA,m,x	Load ACCUMULATOR (m) → (ACC)	E0	RDB,m,x	Read Data Block Data → (m)
28	FLC,k	FLAG clear 0 → FLAG bit	E8	RDI,m,x	Read until Interrupt Data → (m)
30	LCC,m,x	Load Consecutive (m) → (m + 1)	F0	MNI,m,x	Manual Input (INDEX) & (BUFFER) → (m)
38	LAN,m,x	Load ACCUMULATOR Negative - (m) → (ACC)	F8	FLS,k	FLAG Set 1 → FLAG bit
40	LDQ,m,x	Load QUOTIENT register (m) → (QUO)			
48	STA,m,x	Store ACCUMULATOR (ACC) → (m)			
50	STX,m,x	Store INDEX (INDEX) → (m)			
58	STQ,m,x	Store QUOTIENT (QUO) → (m)			

COM-TRAN TEN  
INSTRUCTION  
REPERTOIRE

MEMORY

PAGE =    1    2    3    0    1    2    3    0    1    2    3    0    1    2    3

INDEXED =                    X   X   X   X                    X   X   X   X

LSD+0    1    2    3    4    5    6    7    8    9    A    B    C    D    E    F

MSD

↑

0	SST	LCI	LAI	INX	INVALID	INST		SKI	SKS	SKF	SLA	INVALID	INST				
1	SRA	ØCD	LXI	SLL	INVALID	INST		SRL	AND	IØR	XØR	INVALID	INST				
2	LDA	LDA	LDA	LDA	LDA	LDA	LDA	LDA	FLC	FLC	FLC	FLC	FLC	FLC	FLC	FLC	FLC
3	LCC	LCC	LCC	LCC	LCC	LCC	LCC	LCC	LAN	LAN	LAN	LAN	LAN	LAN	LAN	LAN	LAN
4	LDQ	LDQ	LDQ	LDQ	LDQ	LDQ	LDQ	LDQ	STA	STA	STA	STA	STA	STA	STA	STA	STA
5	STX	STX	STX	STX	STX	STX	STX	STX	STQ	STQ	STQ	STQ	STQ	STQ	STQ	STQ	STQ
6	ADD	ADD	ADD	ADD	ADD	ADD	ADD	ADD	SUB	SUB	SUB	SUB	SUB	SUB	SUB	SUB	SUB
7	MPY	MPY	MPY	MPY	MPY	MPY	MPY	MPY	DIV	DIV	DIV	DIV	DIV	DIV	DIV	DIV	DIV
8	RAØ	RAØ	RAØ	RAØ	RAØ	RAØ	RAØ	RAØ	RSØ	RSØ	RSØ	RSØ	RSØ	RSØ	RSØ	RSØ	RSØ
9	BUN	BUN	BUN	BUN	BUN	BUN	BUN	BUN	BST	BST	BST	BST	BST	BST	BST	BST	BST
A	BSB	BSB	BSB	BSB	BSB	BSB	BSB	BSB	BPS	BPS	BPS	BPS	BPS	BPS	BPS	BPS	BPS
B	BZE	BZE	BZE	BZE	BZE	BZE	BZE	BZE	BNG	BNG	BNG	BNG	BNG	BNG	BNG	BNG	BNG
C	BNC	BNC	BNC	BNC	BNC	BNC	BNC	BNC	BXZ	BXZ	BXZ	BXZ	BXZ	BXZ	BXZ	BXZ	BXZ
D	WDB	WDB	WDB	WDB	WDB	WDB	WDB	WDB	MNØ	MNØ	MNØ	MNØ	MNØ	MNØ	MNØ	MNØ	MNØ
E	RDB	RDB	RDB	RDB	RDB	RDB	RDB	RDB	RDI	RDI	RDI	RDI	RDI	RDI	RDI	RDI	RDI
F	MNI	MNI	MNI	MNI	MNI	MNI	MNI	MNI	FLS	FLS	FLS	FLS	FLS	FLS	FLS	FLS	FLS
LSD+0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F		

## COMPUTER TERMS GLOSSARY

These terms are defined only as they relate to computer programming. Specifically, where technicalities arise, the reference is to the COM-TRAN TEN computer system. Not all these terms are issues in this manual, but they are included here as a more overall source.

abacus	... device for calculating by sliding beads or counters along strings
absolute coding	... instruction written in machine language; language that the computer understands and so no translating process is needed
access time	... length of time required to move a word from a specified memory location or to put a word into a specified memory location.
accumulator	... special part of the arithmetic-logic unit of the computer where temporary storage is handled and algebraic sums are formed
address	... identification of any location in computer where information is stored
ALGOL	... ALGOritmic Language; an algebraic language
algorithm	... step-by-step procedure to accomplish a given result, method of solving a problem in a given system
alphanumeric characters	... generic term for alphabetic, numeric, and special symbols
analog computer	... computer which outputs as a result of measured input; both input and output are of a continuous nature
applications	... problems in any discipline to which the computer is directed
argument	... variable upon whose value the value of a function depends; the function is a subprogram in the computer and the argument is listed in parentheses after the function name
arithmetic operation	... one of the four basic operations of arithmetic; addition, subtraction, multiplication or division
arithmetic section	... function part of a digital computer that performs the arithmetic operations required to solve a problem
arithmetic statement	... type of statement which specifies numerical computation and assigns the value to some variable
array	... series of items arranged in rows and columns



assembler	... computer program that operates on symbolic-coded instructions to produce a machine-coded program that the computer can execute
automation	... predetermined process for self-movement and self-control
base (radix)	... number of different digit-symbols used to form numbers in a number system that uses positional notation; the decimal number system is written to the base ten and has ten digits: 0,1,2,3,4,5,6,7,8,9; the octal number system is written to the base eight and has eight digits: 0,1,2,3,4,5,6,7; the binary number system is written to the base two and uses two digits: 0 and 1; the hexadecimal number system is written to the base sixteen and uses sixteen digits: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
BCD (Binary Coded Decimal)	... system of writing decimal numbers in binary coded form in which each decimal digit is represented by a binary number: 256 is 0010-0101-0110 and 718 is 0111-0001-1000
batch processing	... technique by which information is coded and collected into groups before processing
binary number	... number written in the binary number system, that is using only 0's and 1's
binary digit	... either 0 or 1 which represents one of two conditions: on or off
binary point	... character to separate units from fractional representation in binary notation
bit	... binary digit; either a 0 or 1
blank	... absence of any information; character which specifically represents no information
Boolean algebra	... mathematical system with two elements and unary and binary operators
bootstrap	... technique by which a device brings itself into a desired state by acting on itself
branch	... depart from the normal execution of instructions in sequence
buffer	... extremely temporary storage device of relatively small capacity that can receive and transmit data at different speeds
byte	... sequence of adjacent binary digits that are acted upon as a unit; usually eight bits
call	... branching to a specified subroutine

card punch ... piece of peripheral equipment for punching holes in cards

card reader ... mechanical or photoelectric device used to read the pattern of holes on a punched card

cell ... one location in the memory section; each cell has its own location and can store one "word"

central processor ... computer proper that contains the circuits that control and perform the execution of instructions

character ... symbol used to represent one digit of a number or one letter of the alphabet

clear ... instruction that erases the contents of storage by storing zeros

closed-subroutine ... subprogram which is not stored with sequence of main program but is entered by a branch instruction and after execution, control returns to main program

COBOL ... COmmon Business Oriented Language; specific language for expressing business data and procedures in everyday language

code ... binary number that represents a computer operation

coding ... act of preparing the code a list of specific instructions directing the computer to solve a certain problem

coding sheet ... form on which instructions are prepared in a certain language

command ... an instruction or control signal in machine language

compiler ... program-making routine which translates an instruction into a subroutine in machine-language, assigns addresses to data and instructions, executes the program, and retranslates into pseudo-instructions and data

complement ... in the binary system, the number formed by replacing 1's with 0's and by replacing 0's with 1's and then adding one to the result: 0001 1011 and the complement of 0101 0011 is 1010 1101 in the decimal system the complement of 526 is 474 and the complement of 169 is 831

computer ... device capable of accepting information, processing the information according to prescribed laws and controls, and supplying the results of the processing

console ... unit of peripheral equipment where the control keys and special devices are located

constant ... quantity of message that is not subject to change in a given program

control statements ... command in a program that directs the next instruction to be executed

core ... a very small toroid shaped object made of ferrous (iron) material capable of permanently maintaining either one of two definite states of magnetism

core memory ... storage device made of ferrite cores

counter ... binary counter displays the count in binary notation and a decimal counter displays the count in decimal form

cybernetics ... field of technology involved in the comparative study of man and machines in their control and intracommunication of information

data ... any information fed into the computer that is used by the computer in performing some operation

data processing ... any procedure for accepting information and producing a specified result

debug ... locate and correct errors in a computer program

decimal ... numeration system in base ten

decision ... process of determining one or more choices; usually by comparison to determine if a certain condition is met

decrement ... quantity by which a variable decreases

delete ... leave out

delimiter ... character that limits a string of characters

device ... anything formed by design

digital computer ... computer that processes information that is expressed in discrete quantities

dimension ... statement that sets the maximum number of memory cells to be reserved for variables with a subscript

direct addressing ... naming the location of information by its machine-coded address

direct mode ... on-line working with a special language system

disk-storage ... device that stores information on the surface of flat magnetic disks

divide fault	... error that occurs when a large number is divided by a small number and the quotient cannot contain all the digits of the result
DO loop	... a sequence of instructions that are repeated for variables that increase until a specified limit is reached
documentation	... information that presents, organizes and communicates all necessary facts to understand behavior of a problem in a given situation
double precision	... increased word size in a computer that allows for results to be twice as large as input data
drum	... storage device with a magnetic surface on a circulator cylinder
dummy variable	... character introduced within a program to store data within the processing of the program
E-format	... exponential representation of a number; significant digits are expressed as a number between 1 and 0 multiplied by a power of ten
electronics	... branch of science that deals with behavior of electrons
= equal symbol	... is replaced by
error message	... symbol or information that communicates that the language in the program cannot be understood and executed
execution	... carrying through a prescribed instruction or group of instructions
explicit parentheses	... parentheses around an expression in a statement
exponentiation	... expression that involves powers of a given or understood base
expression	... valid series of functions, variables, and constants that may be connected by operations
fetch	... part of a computer cycle that determines what the next instruction is to be executed
fixed point	... arithmetic notation for a number that has predetermined the position of the decimal point
flag	... character that indicates some condition
flip-flop	... an electronic device capable of maintaining either one of two states, usually designated as a '1' or a '0'

floating point	... arithmetic notation for a number that requires expressing each time the position of the decimal point; notation that involves digits and a power of ten
flow chart	... pictorial representation of the solution of a problem by showing the steps in a sequential order
format	... predetermined arrangement of words, characters, numbers, symbols, lines, etc.
FORTRAN	... FORMula TRANslation; specific language that uses algebraic and English notation to solve scientific and mathematical problems
function	... special purpose or characteristic action; relation of one item from a set with items from another set
hard copy	... printed-out pages from output device
hardware	... mechanical, magnetic, electrical, and electronic devices or components of a computer
heuristics	... intuitive trial and error method of approaching a problem; exploratory method of problem solving that arrives at solutions by examining progress toward the final answer
hexadecimal	... numeration system based on sixteen digits
hierarchy of operations	... order in which mathematical operations are carried out
hybrid computer	... combination of an analog computer with a digital computer; analog computer offers speed, flexibility, and direct communication while the digital computer contributes the memory, logic, and accuracy
IBM card	... paper card that may have information on it in the form of holes that can be read by a computer
illegal characters	... symbol or combination of bits that the computer does not accept
implicit parentheses	... parentheses that are not printed in the statement but result in certain operations as if they were present
increment	... quantity added to a variable
index	... number operated upon in a special register or cell to count or modify a procedure
indirect addressing	... method of assigning memory cells in machine language to instructions or data in another language

information retrieval ... branch of computer science relating to the techniques of storing large amounts of information and searching for appropriate sections as needed

initialize ... set a counter, switch, or address to zero or some other beginning number

input/output ... equipment that communicates with the computer

instruction ... numerically coded command (in machine language) that commands the computer to perform some operation; an instruction has two parts: the operation code number (code) and an operation data address (operand)

iteration ... technique of repeating a group of instructions

jump ... instruction or signal which conditionally or unconditionally tells the computer the location of the next instruction and directs the computer to that instruction, for example an unconditional jump instruction causes the computer to go to a specified place for its next command and to do this as soon as it is decoded; a conditional jump instruction causes the computer to go to a specified place for its next command ONLY if a certain condition exists

keyboard ... device to input data; device encodes data into a binary form which is changed to signals to the computer

language ... defined set of symbols, characters, words, and rules that combine to a meaningful communication

library ... organized collection of standard routines

limit ... capacity of a system; specified number that stops a loop; number to which a series converges

linear programming ... technique used in mathematics and operations research to find the BEST solution to a problem (there is no RIGHT answer)

list ... items written in a meaningful format that are transmitted to input/output print every relevant item of input data

load ... to put instructions or data into the computer's memory or into a register

location ... unit of storage in the internal memory of the computer; place in main memory or an auxiliary storage where data or information is kept until called for

logic	... branch or part of philosophy that studies human thinking and the processes of reasoning; it investigates the validity of conclusions reached
logical design	... overall plan of a digital computer expressed in terms of the five basic logic units: AND, OR NOT, flip-flop, delay
logic diagrams	... representation of the working relation between the parts of a system in terms of symbolic logic
logical decision	... choice between alternatives; usually the response is either yes or no
loop	... repeated execution of a series of instructions
machine cycle	... length of time required to fetch and execute a single instruction; time required to get an instruction from memory, interpret it, and carry it out
machine language	... set of symbols and characters with certain rules for combining them that convey instructions or information that the computer can understand and execute
magnetic tape	... external storage device that retains information in the form of magnetically polarized spots
magnitude	... size of a number without regard to direction or sign
matrix	... rectangular array of numbers or variables
mechanical	... physical quantities that operate with masses, gears, rods, etc
memory	... a storage device used to hold instructions or data until the computer calls for it
microsecond	... one-millionth of a second
mnemonic operation codes	... technique to assist one to remember; computer instructions written in an abbreviation that is meaningful
mode	... method of operation
model	... mathematical representation of a concept, device, process, or technique
multiprocessor	... computer with multiple logic and arithmetic units for simultaneous use
nanosecond	... one-billionth of a second
nesting	... including a routine within a larger routine
network	... series of interconnected points

nonexecutable	... state in which the computer cannot operate
object program	... machine language program which is the final output of a coding system
octal	... numeration system with eight digits
off-line unit	... peripheral equipment not under the direct control of the central processor of the computer
on-line unit	... peripheral equipment under direct control of the central processor of the computer
open subroutine	... separately coded sequence of instructions that is inserted into the main program directly where it would be executed
operand	... any quantity entering into or arising from an operation
operation code	... symbols which direct the computer to perform a defined action
operators	... characters that designate mathematical operations
overflow	... result of an arithmetic operation that exceeds the capacity of the register to receive it
parameter	... constant or independent variable which is not an argument and upon which an operation depends
parentheses	... special character: ( )
peripheral equipment	... units or machines used in connection or conjunction with a computer but are not part of it
positional notation	... procedure in some numeration systems whereby the value of a digit is determined by its symbol and its position relative to the radix point
power	... expression involving some base used as a factor a specified number of times
printer	... device that outputs data
processor	... device capable of receiving data, manipulating it, and supplying results usually of an internally-stored program
program	... plan for the solution of a problem which includes a step-by-step set of instructions that the computer can understand with all necessary data
program counter	... binary counter located in the control section of the computer that keeps track of the location of the next instruction to be executed
programmer	... person who prepares the set of instructions for the computer to solve the required problem



programming system	... method of programming problems consisting of a language and its processor, other than machine language
pseudo instruction	... group of characters having the same general form as a computer instruction, but never executed by the computer as an actual instruction
punched tape	... tape on which information is recorded by punched holes
quantity	... constant, variable, expression, or function name
radian	... unit of measure for angles; pi radians = 180 degrees
radix	... base of a number system
random access	... ability of a memory device to obtain the contents of any memory location immediately
range of a DO loop	... values that the variable in a DO loop has to execute the loop
read	... transmit information from an input device to a computer
real-time	... method of processing data so quickly that there is no time between inquiry and result; pertains to performance of a computation during a related physical event to obtain results that guide the event
register	... a temporary storage device which is capable of holding binary coded data until the data is called for; the register is made up of several flip-flops, each of which can hold one binary digit (bit)
A REGISTER (ACCUMULATOR)	... in the CT-TEN computer, an 8-bit register in which the results of the arithmetic operation of add and subtract are formed
AQ REGISTER	... in the CT-TEN computer a combined register formed from the 8-bits of the Q Register and the sign and magnitude bits of the Accumulator; the AQ Register is used during multiplication or division
B REGISTER (BUFFER)	... in the CT-TEN computer, an 8-bit register which holds all data to be loaded into, or which has been unloaded from, the memory
C REGISTER (COUNTDOWN)	... in the CT-TEN computer, an 8-bit decreasing counter; the C Register is used as a decision counter in arithmetic operations, and as a down counter during the input or output of data
D REGISTER (DISTRIBUTOR)	... in the CT-TEN computer, a 5-bit increasing counter which controls the computer timing

M REGISTER (MEMORY)	... in the CT-TEN COMPUTER, a ten-bit register used to determine from which memory address the next instruction or datum is to be taken
P REGISTER (PROGRAM)	... in the CT-TEN computer, a ten-bit register which holds the address location of the instructions as the instruction is performed; this allows the computer to follow a program in sequence
Q REGISTER (QUOTIENT)	... in the CT-TEN computer, an 8-bit register which stores the quotient upon completion of a divide operation; this register also holds the least significant digits in the product of a multiplication
S REGISTER (STORAGE)	... in the CT-TEN computer, an 8-bit register which holds the OP CODE of an instruction, the index bit and the two high order bits of the memory address register
X REGISTER (INDEX)	... in the CT-TEN computer an 8-bit register in which a count can be loaded, changed by addition, and compared to zero
repertoire	... a complete list of instructions which the computer is able to perform
right-justify	... position the right-hand digit or character so that it occupies the allotted right-most space
root	... result of a power expression with the power greater than zero and less than one
routine	... set of step-by-step instructions fed to a computer to solve a particular problem or do a certain job
rub out	... erase from memory
run	... execute
scale	... range of values dictated by computer word-length or routine being executed
scientific notation	... expression in which a quantity is a fractional number and a power of ten
sense switch	... switch on the console of some computers which may be set UP or DOWN; by setting this switch a program may be tested for certain conditions
shift	... movement of bits, digits, or characters to the right or left
sign	... binary indicator that distinguishes negative quantities from positive quantities
simulator	... device that uses an experimental technique with a physical or mathematical model to behavior of a real-world problem

slide rule ... device based on logarithms that multiplies numbers by adding distances; an example of a hand analog computer

software ... internal programs and routines prepared to simplify programming and computer operations which also extend the capabilities and functions of the hardware

soroban ... biquinary abacus used by Japanese merchants

sort ... distribute into groups according to a set of rules

source program ... program coded in some language other than machine language that must be translated into machine language before use

statement ... instruction directing the computer to perform some sequence of operations

statement number ... number identifying a single statement so that the statement can be referenced within the program

step ... one of the simple operations required to solve a complex problem

storage ... device in which data and instructions can be stored

store ... place information in a location in storage so that it can be recovered later

stored program computer ... computer in which instructions to be executed are stored in memory

subprogram ... program which defines desired operations and which may be included in another program

subroutine ... subprogram

subscript ... notation used to specify a particular member of an array where each member is referenced only in terms of the array

subscripted variable ... variable followed by one or more subscripts enclosed in parentheses

symbolic coding ... writing programs in any language other than absolute machine language

tape unit ... device upon which a magnetic tape is mounted for reading or writing

TEACHWARE™ ... materials that implement the hardware and software in a classroom situation

teletype ... teletypewriter; device that communicates with computer for input or output from a paper tape or a keyboard

time-sharing	... letting a device for two or more purposes
transfer	... terminate one sequence of instructions and begin another sequence or move information from one location to another; move a "word" from one register to another
transfer instruction	... instruction which causes a transfer for any or no condition
truncate	... cut off the number of digits in an answer by dropping them
unary	... operation that acts on a single number of expression
unload	... to remove instructions or data from memory
variable	... symbol whose numeric value changes from one iteration of the program to the next; any expression whose value changes within a program
word	... group of digits handled by the computer as a single unit
word size	... set number of digits that a storage cell may hold; largest number the computer can handle
write	... transfer information to an output device

## CHAPTER 2

### COMPUTER UNITS LOGIC ANALYSIS

A heart specialist knows "what" the heart does, it pumps blood through the body. However, in order for him to properly diagnose heart problems and repair them, the "what" alone is not enough. He must, also, know "how" the heart works. The same is true for a computer maintenance specialist. As a computer repairman knowing "what" the computer does is necessary, but "what" by itself is not enough. You must, also, know "how" it does its job. When a computer system fails for some reason, knowing the "how" will let you go through the circuits, step by step, in the same order the computer does. This is the best way for you to find the problem and then repair the system.

You will start learning the "how" of the COM-TRAN 10. There are some special components used in the COM-TRAN 10 which will be explained first. Then you will learn the "how" of each UNIT of this system, by tracing input signals to determine their effects and the output signals generated.

### SPECIAL COMPONENTS

#### Phantom OR-Gate

This gate is called "phantom" because it is not a real, physical OR-gate which you could find and look at or remove. However, due to the way the circuit is wired the OR-function does exist at these points. Actually a phantom OR-gate is just a common point, that is, a point with several parallel inputs, and one output. This is shown in figure 2-1.

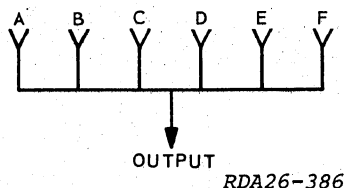
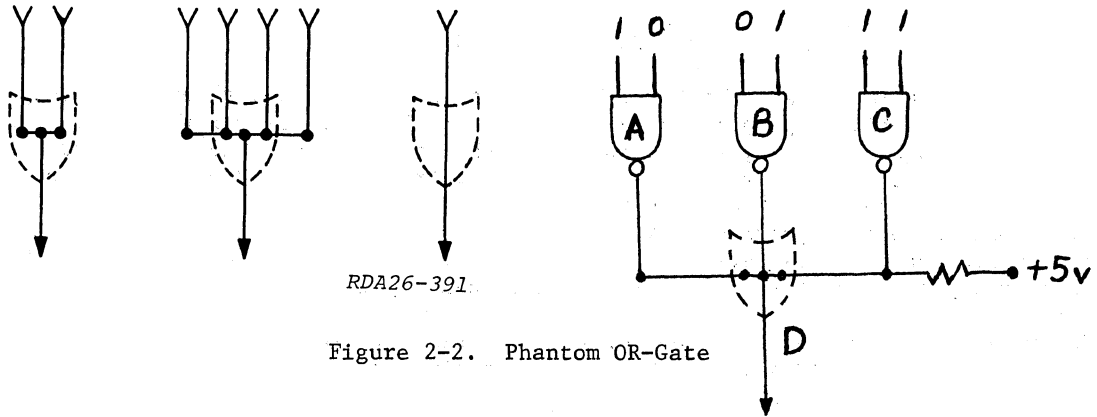


Figure 2-1

Since these are nothing but conductors, all points are electrically the same and if any one of the inputs were to be connected to ground, all the other inputs and the output would feel that ground. This, then, is acting like an OR-gate, if any input goes low, all inputs and the output go low. Therefore, the logic symbol for this type of wiring arrangement shows an OR-gate made with dotted lines. (See figure 2-2.) Each phantom OR-gate symbol shows the inputs to the gate for the particular page of logic it appears on ONLY, and it must be remembered that there may be several other inputs affecting that output elsewhere in the circuit diagrams.

The COM-TRAN 10 uses phantom OR-gates for two purposes. One, it is used to show the generation of a signal called SPD15 (see sheet 21 of KDA-3034). Notice that this symbol shows 7 inputs (these are all the inputs affecting this gate) and if any one of them goes



RDA26-391

Figure 2-2. Phantom OR-Gate

low, all the other inputs and the output, SPD15 will go low. For example, you can see on the logic sheet that there are 6 AND-gates and 1 inverter feeding the inputs to the phantom OR-gate. If any one of these were to become satisfied by its input signals, its output would go low. This LOW would then be felt on the phantom OR-gate input and it would cause the output and all the other inputs of the phantom OR-gate to go LOW. If you were to then measure the output of any of the 6 AND-gates or the inverter, you would measure a LOW. You would not be able to tell which gate had produced the LOW input which satisfied the phantom OR-gate. This phenomena must be kept in mind when you try to troubleshoot a computer which contains phantom OR-circuits. The second use for this circuit in the COM-TRAN 10 is to indicate the inputs to the Z-bus. A bus is a common conductor among a number of locations, which is used to transmit data between different parts of the computer. There are several buses in the COM-TRAN 10 and "Z" is the name used for this particular one. The Z-bus is made up of 8 bits or 8 separate common conductors, and therefore it takes 8 phantom OR-gates to represent it. Several of the COM-TRAN 10 registers have outputs to the Z-bus; these are indicated by the 8 phantom OR-gates whose outputs are called  $\overline{Z0}$ ,  $\overline{Z1}$ , ..... $\overline{Z7}$ . (See sheets 1, 6, 8, 10, 12 and 14 of KDA-3034.) Notice that each symbol shows only the inputs on that sheet of logic, but a LOW input to any one of the gates would cause the inputs and output to go LOW on all of the gates representing that bit.

In summary, a phantom OR-gate is just a common conductor with many parallel inputs which functions like an OR-gate. If any one of its inputs go LOW all of its inputs and its outputs go LOW. This type of circuit is symbolized by a dotted line OR-gate logic symbol.

#### NOR-Gate Latch

A latch circuit is the most basic type of flip-flop. They can be constructed using AND, OR, NAND, or NOR gates; or some combination of these gates. The COM-TRAN 10 uses one type of NOR-gate latch in its circuits. Before going into the actual operation of that circuit, review the three basic characteristics shared by all flip-flops:

1. The flip-flop is a bistable device, that is, a circuit with only two stable states. These states are called the 0 or clear state and the 1 or set state.

2. The flip-flop circuit can store a binary bit of information because of its bi-stable property. The flip-flop responds to inputs, and if an input causes it to go to its 1 state, it will stay there and store that 1 until another input causes it to go to its 0 state. The same would be true for the 0 state.

3. The flip-flop has two output signals, with one of them normally the complement of the other.

The NOR-gate latch is made from two NOR-gates with the output of each gate tied to the input of the other. (See figure 2-3.) Notice that the diagram shows only one external input to each gate, however, there can be any number of inputs. With no signals applied to the external inputs they will be in the normal condition, which is HIGH. As with all flip-flops when power is first applied to a circuit containing a latch circuit it will randomly go to one state or the other, and stay in that state until a specific input is applied.

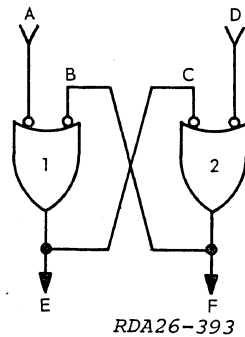


Figure 2-3. NOR-Gate Latch

Look at figure 2-3 and assume that a LOW is applied to A. This will satisfy gate 1 and will give a HIGH out at E, which is tied back to C. There is a HIGH at D since there is no input signal applied at that point. With only HIGH inputs, gate 2 is not satisfied and a LOW output is felt at F. This LOW at F is coupled back to input B, and serves to keep gate 1 satisfied even though the LOW input to A may no longer be present. Now the latch will remain in this state, E = 1 and F = 0, until a LOW is applied to input D to cause the circuit to switch states. The following chart shows what effect different input combinations would have on the NOR-gate latch shown in Figure 2-3.

CHART 2-1

TRUTH TABLE FOR NOR-GATE LATCH

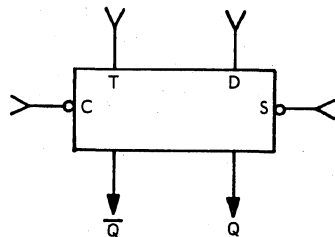
A	D	OUTPUT
1	1	No effect. The latch stays in the state it was in since neither input is satisfied.
0	1	Gate 1 satisfies. Latch outputs go to E = 1, F = 0.
1	0	Gate 2 satisfies. Latch outputs go to E = 0, F = 1.
0	0	Both gates satisfy producing an unstable condition. Latch outputs go to E = 1, F = 1. Not a normal input or condition.

The main use of a latch circuit is to store a bit of binary data. It cannot be used in a counter or shift register since it has no provision for clocking inputs. This is the main disadvantage of latch circuits.

In summary, a NOR-gate latch is a basic flip-flop. Its external inputs determine which state it will be in at any given time. The logic symbol shows two NOR gates with the output of each gate coupled to the input of the other. NAND gates may be used instead of NOR gates to obtain the same latch flip-flop.

#### D-Type Flip-Flop

It was pointed out in the last section that a latch-type flip-flop cannot be used for counters or shift registers. Therefore, another type of flip-flop which can perform these functions must be used. Of the many types available, the designers of the COM-TRAN 10 decided on the D-type flip-flop for use in places where the latch would not be satisfactory. The D-type flip-flop is probably the simplest of the group of clocked flip-flops. It has two direct inputs and one data input with trigger. The first direct input, called SET, forces the flip-flop to the set 1 state. The second input, called CLEAR, forces it to the clear 0 state. These two inputs are called direct because they function independently from the clock. The fact that it needs only one data input line makes it unique from most clocked flip-flops. The data input is dependent on the clock input to pulse it into the flip-flop. The logic symbol (see figure 2-4) shows these four inputs with labels as follows: set - S, clear - C, data input - D and clock - T. The outputs are labeled Q which is the one side and  $\bar{Q}$  which is the zero side. Q and  $\bar{Q}$  are standard symbols used to represent the outputs on most types of flip-flops.



RDA26-389

Figure 2-4. D-Type Flip-Flop

The state indicators on the S and C input lines mean that it takes a LOW signal to satisfy the input and thus set or clear the flip-flop. The trigger or clock input is satisfied by an up-clock. This up-clock will cause the data input to be transferred to the Q side output. If the data line (D) has a 1 input when the clock input up-clocks, the 1 will be transferred to the Q side output (1 state). If the D line has a 0 input when the clock input up-clocks, the 0 will be transferred to the Q side output (0 state).



The  $\bar{Q}$  side is always the complement of the Q side. When an input is on the D line, whether HIGH or LOW, the output will not change until an up-clock occurs on the trigger input.

In summary, the D-type flip-flop is a clocked flip-flop with 2 direct inputs and one data input with trigger. The direct inputs are forced inputs which are used to set or clear the flip-flop independent of the clock. The data line contents will transfer to the Q side output on the up-clock of the clock input.

### Single Shot (SN74121)

The COM-TRAN 10 uses this particular single shot (monostable multivibrator) to produce pulses used to control the generation of clock pulses. Before covering the operation of the circuit, review these basic single-shot principles.

The single shot is a circuit which has one stable state and one unstable state. The unstable state is sometimes called the semi-stable state. The single shot can be triggered into its unstable state but will not stay there. The RC time constant built into the circuit determines how long it will stay in its unstable state. After that period of time, the single shot will go back to its stable state and stay there until it is triggered again.

The operation of single-shot circuits was covered in previous blocks. The one used in the COM-TRAN 10 has a few differences. Its triggering circuit is made up of three inputs. The logic symbols for this single shot are shown in figure 2-5. A(1), A(2) and B are the three inputs which are used to trigger a single shot. Each single shot has two possible outputs, Q and  $\bar{Q}$ , but only one will be wired into the rest of the circuit. The logic symbol will only show the one being used.

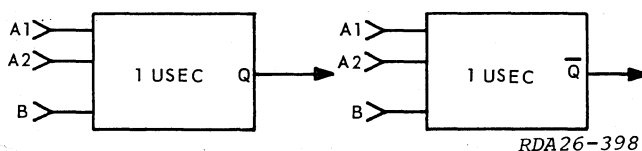






Figure 2-5. SN74121, Single Shot

The trigger inputs work together to fire the circuit outputs. The output pulse will be produced when the A(1), A(2) and B inputs meet the required conditions. Look at figure 2-6, the triggering sequences for this circuit, and refer to it while going through the triggering sequence required to produce an output pulse. A(1) and A(2) inputs are always tied together in the COM-TRAN 10. Look on sheet 7 of the C & D's to see how this is accomplished.

No other conditions will fire the single shot. Note that the output is the same regardless of which triggering sequence is used. If either or both of the A(1) or A(2) inputs are already in the logical 0 state and the B input changes from a 0 to a 1 state,

there will be a 1-microsecond pulse produced at the output. If the B input is already in a logical 1 state and either or both of the A(1) or A(2) inputs change from a 1 to a 0 state, a 1-microsecond output pulse will be produced. The output signal pulse will be a 1-microsecond positive pulse if the circuit is using the Q output. If the circuit is using the  $\bar{Q}$  output, a 1-microsecond negative pulse will be produced. This is true regardless of which triggering sequence is used to produce the output pulse.

INPUTS		OUTPUTS	
A(1) or A(2)	B	Q	$\bar{Q}$
0	upclock ( $\uparrow$ )		
downclock ( $\downarrow$ )	1		

RDA26-446

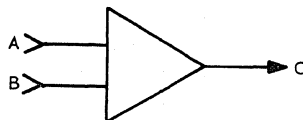
Figure 2-6. Triggering Sequences for the SN74121 Single Shot

The SN74121 does the same thing any other single shot does - produces a single output pulse each time it is triggered. Its three input triggering circuit is the only "different" feature it has.

#### Positive AND-Driver

The positive AND-driver is a very simple circuit, and is used basically as an electronic switch. In the COM-TRAN 10, it is used to turn the indicator lights off and on.

At first glance, the logic symbol (see figure 2-7 for a positive AND-driver) tends to be misleading because it is very similar to the logic symbol for an amplifier. A closer look, however, points out that it has two inputs, A and B, which distinguishes it from an amplifier symbol. The positive AND-driver operates just like an AND-gate; that is two HIGHS in give a HIGH out, while any LOW in gives a LOW out. The thing that makes the circuit different is the fact that due to its application in the COM-TRAN 10, the LOW output generates an ON condition. Its normal condition inputs are A and B both HIGH, giving a HIGH out which keeps the light turned off. Look at figure 2-8 to see why this is true. Notice that one side of the light is tied to the positive AND-driver, and the other side is tied to a positive voltage.



RDA26-395

Figure 2-7. Positive AND-Driver

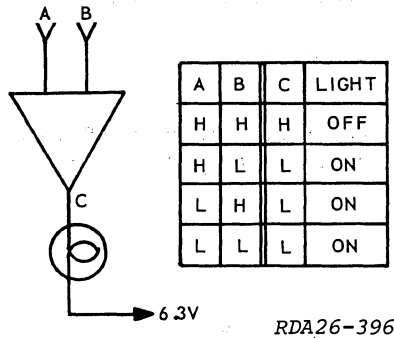


Figure 2-8. Lamp Connection and Truth Table

This means that a HIGH output from the positive AND-driver would NOT produce a current large enough to turn on the light. However, a LOW output from the positive AND-driver would allow a large enough current flow to turn the light ON.

The internal logic circuitry is very easy to understand. It consists of one AND-gate whose output biases a transistor (see figure 2-9).

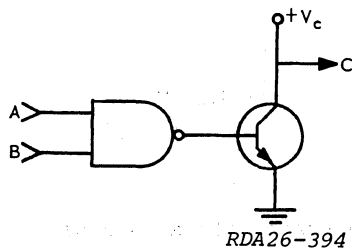


Figure 2-9. Internal Logic for Positive AND-Driver

With two HIGH inputs the AND-gate would be satisfied giving a LOW output. This LOW is fed to the base of the transistor, and since it is an NPN the LOW will reverse bias the transistor and cut it off. A transistor in cutoff acts like an open, so the output at C is  $+V_{CC}$ . If either input or both inputs go LOW the AND-gate is inhibited, putting a HIGH output on the base of the transistor. This HIGH will forward bias the transistor into saturation, causing it to act like a short. The output at C will be 0V. This produces a large enough current flow to turn the light on.

In summary, the positive AND-driver functions just like an AND-gate - two HIGHS in produce a HIGH out, and any LOW in produces a LOW out. The normal condition is two HIGHS in, giving a light OFF condition. Any LOW in gives a light ON condition. The logic symbol looks like an amplifier with two inputs. One of the inputs is from lamp test and the other is from a flip-flop.

SPECIAL COMPONENT SUMMARY

SPECIAL SYMBOL TRUTH TABLES

PHANTOM OR-GATE

A LOW, or OV, applied to any point of a phantom OR-gate, produces a LOW at the output and every input. The static state of a phantom OR-gate is all high inputs and a high output.

NOR-GATE LATCH

Inputs		Outputs		
Gate 1	Gate 2	Gate 1	Gate 2	
1	1	No change		
1	0	0	1	
0	1	1	0	
0	0	1	1	- Unstable state

RDA26-447

D-TYPE FLIP-FLOP

A LOW on the C input to a D-type flip-flop clears the flip-flop. A LOW on the S input to a D-type flip-flop sets the flip-flop. For clocking inputs, the following applies:

T	D	Q	$\bar{Q}$
↑	1	1	0
↓	0	0	1

RDA26-448

POSITIVE AND-DRIVER

A LOW turns the lamp ON. The static state is a high, with the lamp OFF.

Inputs		Output	Lamp
1	1	1	OFF
1	0	0	ON
0	1	0	ON
0	0	0	ON

RDA26-449

TRIGGERING SEQUENCES FOR THE SN74121 SINGLE SHOT

A1 or A2	B	Q	$\bar{Q}$
L	↑	□	□
↓	H	□	□

RDA26-450

SELECTOR TRUTH TABLES

B REGISTER SELECTOR

Pin 2	Pin 14	Selects	
L	L	B Reg	Static state
H	L	I Inputs	
L	H	Two's Comp	

RDA26-451

M REGISTER SELECTOR

Pin 2	Pin 14	Selects	
L	L	S0 & S1	Static state
H	L	P8 & P9	
H	H	I8 & I9	

RDA26-452

ALU CONTROL INPUTS

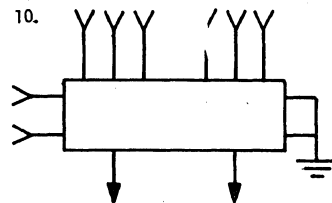
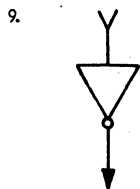
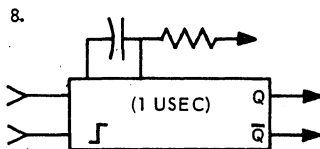
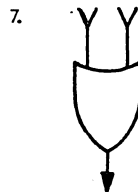
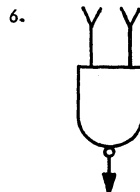
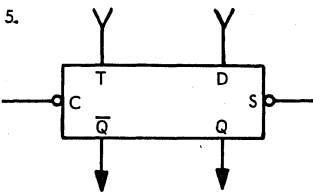
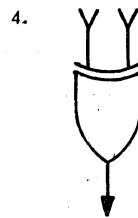
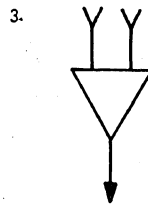
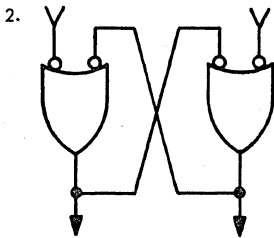
INPUT SIGNAL	CONDITION CODES					RESULTING ACTION
	S3	S2	S1	S0	M	
DECA	H	H	H	H	L	Decrease A input by 1
INA	H	L	L	H	L	Add A input to B input
INS	L	H	H	L	L	Subtract B input from A input
INCA	L	L	L	L	L	Increase A input by 1
IAND	H	L	H	H	H	Logical AND of A with B
IORI	H	H	H	L	H	Logical OR of A with B
IEX	L	H	H	L	H	Logical Exclusive OR of A with B
No inputs (static state)	H	L	H	L	H	Parallel transfer of B to F bus

RDA26-453

REVIEW QUESTIONS 2-1

1. Identify each symbol by placing its number in the blank.

- a. D-type flip-flop
- b. NOR-Gate latch
- c. Positive AND-Driver
- d. Exclusive OR-Gate
- e. Phantom OR-Gate
- f. Monostable Multivibrator



RDA26-402

2. Match the logic symbol names with the correct logic function by placing the letter of the function in the blank provided.

\_\_\_1. D-Type Flip-Flop

\_\_\_2. NOR-Gate Latch

\_\_\_3. Positive AND-Driver

\_\_\_4. Phantom OR-Gate

\_\_\_5. Monostable Multivibrator

a. A common point having several inputs but only one output; where a function exists but a physical component does not.

b. This device can store a binary one or zero; when an up-clock is applied to the T input, the data level on the D input is transferred to the Q output.

c. This circuit is also called a single shot. When the B input up-clocks and the A inputs are low, the circuit will produce one pulse.

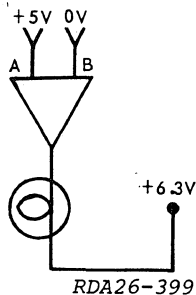
d. When the B inputs downclock and the A inputs are high, this circuit produces one pulse.

e. This device selects one of three inputs for its output depending on two select levels.

f. Any low into this circuit gives a low out that is usually used to turn on a lamp (provide a ground for current flow).

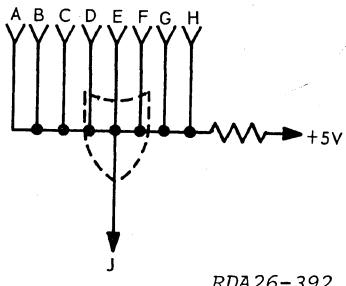
g. This simple flip-flop has no trigger input; a low input to one side produces a high output on that side.

3.



With the applied voltages to positive AND-driver, is the lamp on or off? \_\_\_\_\_

4.



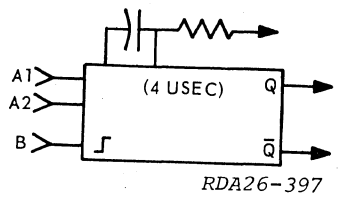
RDA26-392

With nothing applied to the inputs of the phantom OR-gate, what is the output voltage? \_\_\_\_\_

What is the output if input A is zero volts? \_\_\_\_\_

If all the inputs are +5 volts, what is the output? \_\_\_\_\_

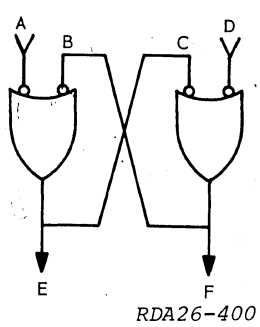
5.



RDA26-397

With the B input high and an input pulse applied to A1 every 3 seconds, what is the output at Q? \_\_\_\_\_

6.



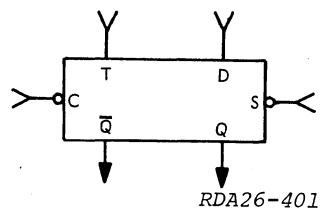
RDA26-400

If a 1-microsecond negative pulse is applied to the A input and the D input is high, what is felt at output F? \_\_\_\_\_

What is felt at input C? \_\_\_\_\_

What is felt at input B? \_\_\_\_\_

7.



RDA26-401

If a high is felt at Q, what single input would cause the flip-flop to change states? \_\_\_\_\_

If a low is felt at Q-bar, what input would be needed along with the trigger to change the state of the flip-flop? \_\_\_\_\_



## COMPUTER UNITS

### Key to Logic Diagrams

In order to trace signals through the various computer units and determine their effects, it will be necessary to use the COM-TRAN 10 Circuits and Diagrams (KDA-3034). So that you can use them effectively, the "Key to Logic Diagrams" will be covered first. The information contained in the logic sheets will be explained. Now turn to sheet 3, KDA-3034. III

The signals shown across the top of the sheet are the signal inputs to that sheet. The little numbers in parentheses after each input signal show which sheet of logic that signal came from. All output signals are shown across the bottom of the page, with the number in parentheses indicating to which sheets that signal goes. For example, turn to sheet 7, look in the lower right-hand corner, and locate the signal MLD. The number it shows is (17), which means that this signal goes to sheet 17. Turn to sheet 17 and you will find MLD somewhere across the top, as an input with an indication that it came from sheet 7.

Each component and each register has three alphanumeric codes indicated in its logic symbol. The first code is always a "U" followed by a number. It is used to identify the type of integrated circuit. The second code is a number followed by a letter. It identifies the IC chip's physical location on the printed circuit board in the computer. For example, if the code was 3C you would find the IC chip in column 3, row C in the computer. This information will be important to you during the lab projects in this block. Look at sheet 3 at four AND-gates on the lower right-hand side. Notice that they are all labeled U3 and 2C. This tells you that the IC chip containing these gates is an SN7403 and that it is located in column 2, row C. In this case, all four AND-gates are in the same chip (2C). Thus, a problem would occur if we tried to refer to only one of these gates during our discussions. This is where the third code (a single letter) comes into use. When used with the second code, it identifies a particular component on the logic diagram. So if you were talking about the AND-gate at the far right you would say, "AND-gate 2CD", and the "D" would identify exactly which one of the AND-gates on IC chip 2C you were indicating. Normally the third code is found on the third line; however, some components combine codes two and three onto one line. You can always recognize this condition as being used, anytime you see two letters on code 2. For example, see sheet 11, the D-type flip-flops use this method; 11JA means column 11, row J, flip-flop A. If only one component exists on a chip, the third code is not used, but rather is identified by code 2 only.

Every input and output to all components has a small number indicated beside it. These tell you the pin number of that input or output line. The pin numbers are also indicated on the IC chip in the computer. Therefore, suppose you wanted to check the input to inverter 2DB (sheet 3) with the oscilloscope. First, you would locate 2DB on the logic sheet, and check the input to find out what the pin number for its input is (Pin 3). Then, you would go to the computer, locate the IC chip (column 2, row D), and connect the oscilloscope input to pin 3 of that chip. The signal observed on the oscilloscope would be the input waveshape to inverter 2DB.

Although it is not shown on the "Key to Logic Diagrams" there is a coordinate system used throughout the logic diagrams to help you find the general location of any particular item. Refer to sheet 3 in KDA-3034. The top of the sheet is divided into ten sections numbered right to left from 1 to 10; and the left side is divided into six sections, labeled bottom to top A to F. When you are trying to locate a specific item such as, AND-gate 6DA, it is much easier if you know that its sheet coordinates are 4D. This means that you should find 4 across the top and D down the side; then look at the area where these two intersect and you will find AND-gate 6DA. During the discussions, you

will be given the sheet coordinates in parentheses following the component identification code. For example, the AND-gate just located would be identified as 6DA (4D).

As you use the logic diagrams you will become familiar with the information and codes they contain and using them will become second nature to you.

### REVIEW QUESTIONS 2-2

Circle the letter of the correct answer to the following questions.

1. What type of IC chip is a U4 (sheet III, KDA-3034)?
  - a. AND-OR INVERT Gate
  - b. Hex Buffer/Driver
  - c. Inverter
  - d. Quadruple 2 - Input Positive NOR-Gate

2.



In the figure at the left, what does 3C indicate?

- a. pin number of component signal
- b. IC type identity (Positive NAND-Gate)
- c. Page cross reference
- d. component location on cartesian grid

3.



What does the signal at the left mean when it appears at the bottom of a logic page?

- a.  $\overline{Q4}$  comes from sheet 9 and is statically high
- b.  $\overline{Q4}$  goes to nine different logic sheets
- c.  $\overline{Q4}$  goes to sheet 9 and is statically high
- d.  $\overline{Q4}$  comes from nine different logic sheets

### Clock

Every computer has a clock circuit to control its timing. This unit is very important because it makes sure that every single step, major or minor, occurs exactly in the right order and at exactly the right time within the computer.

The logic diagram for the COM-TRAN 10 clock circuit is on sheet 7 (KDA-3034), refer to this diagram during the following discussion. The clock can actually be separated into four major parts: (1) the switching network, (2) the pulse generator, (3) the grey-code counter, and (4) the decoding network. (See figure 2-10.) The switching network is made up of 2 NOR-gate latches, 3 AND-gates, 2 OR-gates, and 2 inverters. Its purpose is to turn the clock on or turn it off. The  $\overline{\text{RESM}}$  signal or the  $\overline{\text{START}}$  signal will start the clock, while the  $\overline{\text{SPSW}}$ ,  $\overline{\text{SPCK}}$ , or  $\overline{\text{CL}}$  signals will stop the clock. The two single shots make up the pulse generator, which produces the timing pulses necessary to trigger the counter. The grey-code counter is composed of two D-type flip-flops. The counter counts from 0(10) to 3(10); however, it counts in grey-code, that is, 00 (gc), 01 (gc), 11 (gc), 10 (gc). The reason for using a grey-code count is that only one flip-flop at a time changes state. If both flip-flops had to change states, as in binary from 01 (2), to 10 (2), and if the transition times were not exactly the same then the counter might, for an instant, produce a 00 (2) or a 11 (2) instead of 10 (2). With only one flip-flop changing at a time this problem is eliminated. The decoder determines what count is in the counter and produces the appropriate clock pulse,  $\overline{\text{ENABLE}}$ , CP1, CP2, or CP3. The decoder is made up of 5 AND-gates, and 5 inverters. When the clock operates, all four major parts work together to produce the clock pulses necessary for timing and control.

While going through the clock operation be sure to refer to the logic diagram (sheet 7) and the waveshapes (figure 2-11) and relate the explanations to them.

This explanation assumes that the clock is in an off condition and will trace the signals through the clock circuit to determine what happens when it is started. Anytime the clock is stopped, the grey-code counter will contain a count of 00 (gc), therefore, the clock will always start from 00 (gc). The clock can be started by either one of two signals,  $\overline{\text{START}}$  or  $\overline{\text{RESM}}$  (8F). The  $\overline{\text{START}}$  signal is generated by pressing the START switch on the COM-TRAN 10 front panel, and the  $\overline{\text{RESM}}$  is generated by the teletype interface under certain conditions.

Assume that the start switch is pressed. This makes the signal  $\overline{\text{START}}$  (8F) go low. This low is felt at NOR-gate latch 9HA, pin 1, and NOR-gate latch output (pin 3) goes high. This high is felt at one input to NAND-gate 9HD, pin 13 (8D) satisfying that input. The  $\overline{\text{START}}$  signal also produces a low at one input to NOR-gate 9HC, pin 9 (8E) satisfying that gate and causing its output (pin 8) to high. This high is felt at the other input to NAND-gate, 9HD, pin 12 (8D), satisfying that input. Therefore, both inputs to NAND-gate 9HD are satisfied and its output goes low. The low is felt at one of NOR-gate latch inputs, 9GA pin 1 (7D) which causes pin 3 to go high and NOR-gate latch output 7JA pin 12 (7D) to go low. This low is felt at the input to AND-gate 8JA pin 1 inhibiting that input and giving us a low output at pin 3. The low output is felt at the input to NOR-gate 9JB pin 5 (7C) inhibiting that input. The other input to that NOR-gate is kept inhibited by inverter 6JB pin 4 (7D). Therefore, when pin 5 of NOR-gate 9JB (7C) goes from high to low, its output (pin 4) goes from low to high providing an up-clock to the single-shot B input, 6H pin 5 (7C).

In order to see how the  $\overline{\text{RESM}}$  signal would produce the same effect, you must first understand that the  $\overline{\text{RESM}}$  signal can only be produced when the computer is running, and therefore the  $\overline{\text{START}}$  signal has already been produced and set the NOR-gate latch (8E) to a high or a one output at pin 3. The  $\overline{\text{RESM}}$  then satisfies NOR-gate 9HC (8E) and produces a high output at pin 8. From this point on the sequence is the same as it was when using the  $\overline{\text{START}}$  signal. Thus, in order to start the clock, the switching circuit has provided an up-clock to the pulse generator.

The two single-shots 6H and 7H (6B and 7B) which make up the pulse generator are the SN74121 types discussed earlier in this chapter. Pin 5 is the B input and pins 3 and 4 are the A1 and A2 inputs. Notice that single-shot 6H (7B) has a  $\overline{\text{Q}}$  output, which means that the output stays high until the circuit is triggered and then it goes low for 1 microsecond. However, the other half of the pulse generator, single-shot 7H (6B) has a Q output which means that its output stays low until it is triggered and then it goes

140

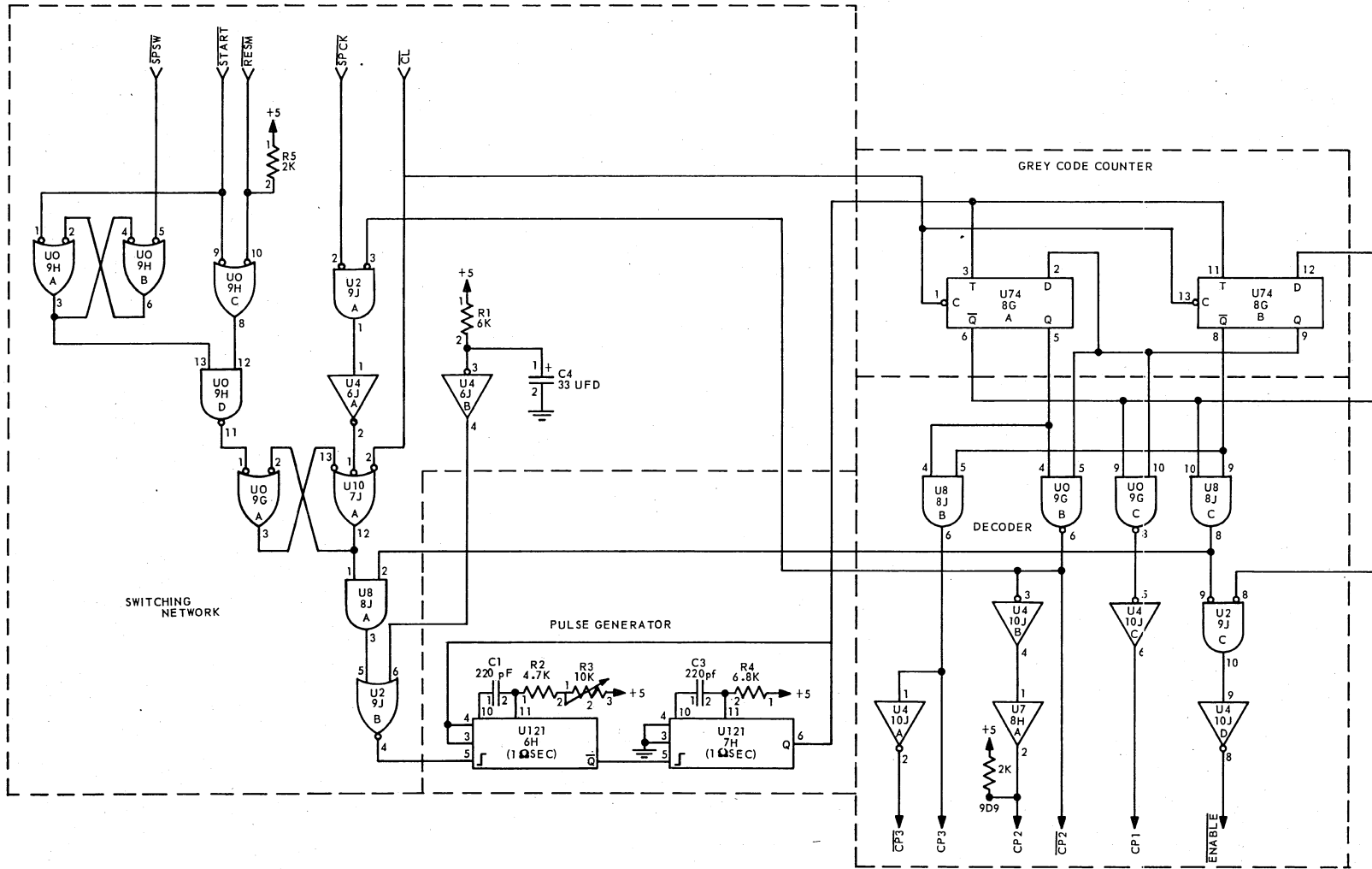
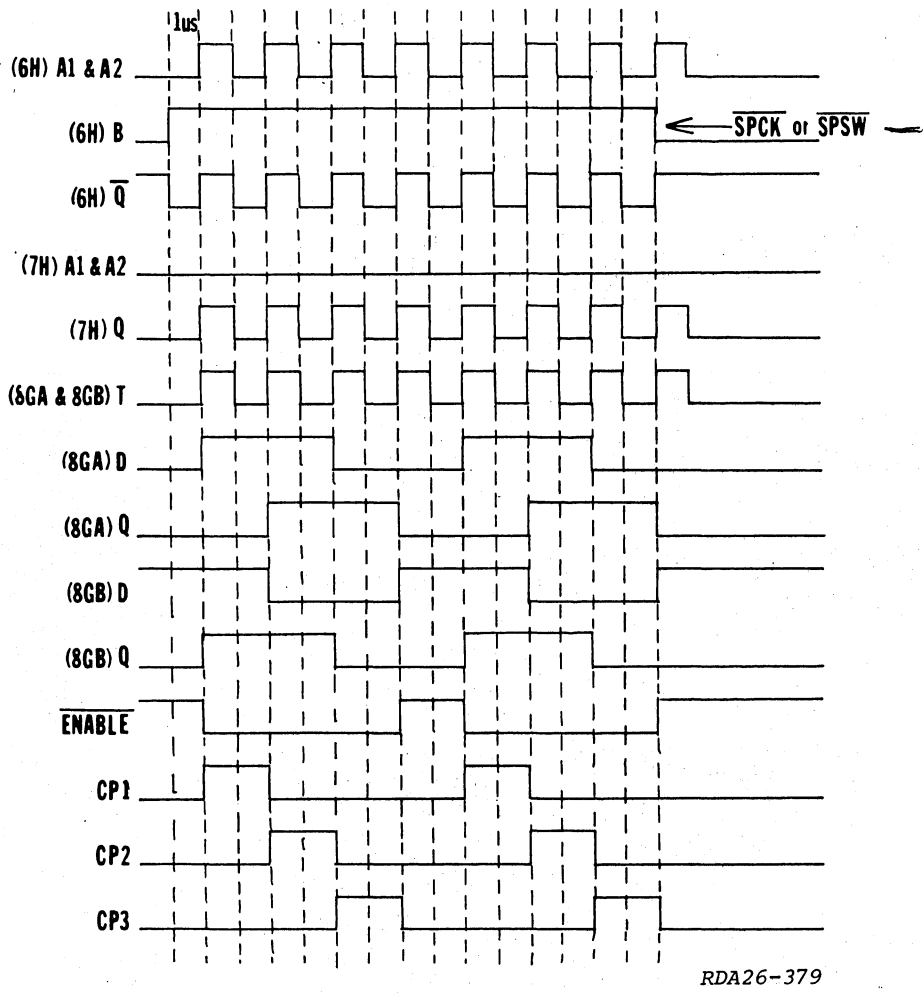


Figure 2-10. Clock Circuit Breakdown



RDA26-379

Figure 2-11. Clock Circuit Waveshapes

high for 1 microsecond. Since the clock is in the off condition, there are no timing pulses being generated by the pulse generator, and therefore the Q output from 7H is at a constant low. This output is fed back to the A1 and A2 inputs of single-shot 6H, holding them at a steady low while the clock is not running. Then when the switching circuit causes the B input (pin 5) to 6H to go from low to high, the single-shot is triggered and a 1-microsecond low output is produced at  $\bar{Q}$  of 6H. This output is directly connected to the B (pin 5) input of single-shot 7H. Before determining the B input's effect, notice that the A1 and A2 (pins 3 and 4) inputs to single-shot 7H are shorted to ground and are therefore always at a low. Now, when the 1-microsecond low output of  $\bar{Q}$  goes back high, the conditions are met to trigger single-shot 7H, that is A1 and A2 (pins 3 and 4) already low and B going from low to high. This produces a 1-microsecond positive output pulse at Q of 7H. This pulse is the first timing pulse to the grey-code counter.

The problem now is how to keep the pulse generator going, so that a steady series of pulses is generated. To understand how this is accomplished, go back and take a look at the B input (pin 5) to single-shot 6H. The switching circuit caused this input to go from a low to a high. The input will stay high once this transition occurs. The START pulse caused the NOR-gate latch output 7JA pin 12 (7D) to go to a low, inhibiting AND-gate 8JA (7C), etc., giving us the high at NOR-gate 9JB, pin 4 (7C). Due to the operation of a latch circuit, that low output will remain low at 7JA pin 12 (7D) until another type of input causes it to change states. Therefore, the B input to single-shot 6H will remain high once it goes high, until caused to go low (this is done to stop the clock and is not important at this time). The Q output of single-shot 7H (6C) is fed back to the A1 and A2 (pins 3 and 4) inputs of single-shot 6H (7C). Therefore when the first timing pulse is produced it is felt at these A1 and A2 inputs, causing them to go high for 1 microsecond and then back low. The B input is already high, this is one of the two triggering sequences for an SN74121 single-shot and causes a second 1-microsecond low pulse to be generated at the  $\bar{Q}$  output of 6H. That pulse again triggers single-shot 7H and it produces another 1-microsecond positive output at Q. This cycle will continue and the pulse generator will produce 1-microsecond positive output pulses every 2 microseconds until stopped. So far, a START signal has caused the switching circuit to trigger the pulse generator, and the pulse generator then continues to trigger itself through a feedback line and produce a steady output of 1-microsecond positive pulses, spaced 2 microseconds apart.

The output pulses from the pulse generator are fed to the trigger inputs of the two D-type flip-flops 8GA and 8GB (3E, 4E) which make up the grey-code counter. Remember that the counter always starts with 00 (gc) in the counter. Flip-flop 8GB is the LSD bit and 8GA is the MSD bit. Notice the wiring arrangement of the counter, the data input to flip-flop 8GB (pin 12) comes from the  $\bar{Q}$  output of flip-flop 8GA (pin 6), and the data input to 8GA (pin 2) comes from the Q output of 8GB (pin 9).

Each time the trigger inputs upclock (every 2 microseconds from the pulse generator output), the data inputs will be transferred to the Q side outputs. Remember, both flip-flops start in the zero state, so the data input to 8GB will be a high and the data input to 8GA will be a low. When the first trigger pulse upclocks, the high (1) will transfer to the Q side of 8GB putting it in the one state, and the low (0) will transfer to the Q side of 8GA leaving it in the zero state. Thus, the counter has clocked from 00 (gc) to 01 (gc). This means that the data input levels to both 8GA and 8GB are now highs or ones. So when the next trigger input upclocks, it will transfer ones to the Q side of both flip-flops causing the counter to clock from 01 (gc) to 11 (gc). The counter will continue to count from 11 (gc) to 10 (gc) and from 10 (gc) to 00 (gc) at which point the counting begins again. Each count remains in the counter for 2 microseconds since the input PRT is 2 microseconds between upclocks. The counter will continue to count, cycling through 00 (gc) to 10 (gc) until the clock is stopped. These counts provide the 2-microsecond clock pulses used throughout the COM-TRAN 10.

Now all the clock has left to do is decode the counts. The decoder is made up of 4 AND-gates 8JB, 9GB, 9GC, 8JC (4D, 5D). By checking the input lines to AND-gate 8JC you can see that it will be satisfied when the counter contains a count of 00 (gc) and will produce a high (1) output at 8JC pin 8 (4D). This high is fed to the input of AND-gate 9JC pin 9 (4C) inhibiting that input leg and producing a low at its output (pin 10). The low output then goes through inverter 10JD (4C) giving a high output, called ENABLE, for 2 microseconds. If you follow this same process for the other three AND-gates 9GC, 9GB, and 8JB (4D, 5D) you will see that 9GC decodes 01 (gc) and produces a high output called CP1; 9GB decodes 11 (gc) and produces a high output called CP2 and a low called CP2; while 8JB decodes 10 (gc) producing outputs called CP3 and CP3. Each of these output pulses have a pulse width of 2 microseconds. However, in the case of ENABLE the 6-microsecond negative portion of the signal is the part used. Therefore, its pulse width is usually thought of as being 6 microseconds. The clock pulses are sent to all parts of the computer and are used to control the timing and sequencing of various operations.

To start the clock the START or RESM signals must be present. This causes the switching circuit to generate an upclock which starts the pulse generator. This in turn generates 1-microsecond pulses every 2 microseconds to trigger the grey-code counter, causing it to count from 00 (gc) to 10 (gc). The decoder detects these counts, producing 2-microsecond clock pulses. Once the clock is started, it will continue with this cycling process until another signal is generated to cause it to stop.

The clock can be stopped by any one of three signals, SPSW, which is generated by pressing the stop switch on the control panel; CL which is generated by pressing the clear switch on the control panel; or SPCK which is generated internally by some of the COM-TRAN 10 instructions, and also by the stop switch. Assume you want to manually stop, you press the stop switch, generating a low at SPSW (8F). This low will satisfy the input (pin 5) of NOR-gate latch 9HB (8E). The function of the NOR-gate latch will produce a low at its output, 9HA pin 3 (8E) in response to the SPSW low input. The NOR-gate latch output is fed to the input of NAND-gate 9HD pin 13 (8D) and since it is a low it will inhibit the NAND-gate so that a RESM signal cannot restart the clock. Anytime the SPSW signal is low it produces the low signal SPCK which actually stops the clock. SPCK is input to NAND-gate 9JA pin 2 (7E) satisfying that leg. The other leg of that gate (pin 3) is connected to the decoder output of CP2 (9GB pin 6, 4D). It will be low when the counter contains a count of 11 (gc) or CP2. Therefore, NAND-gate 9JA will be satisfied during clock pulse 2 and will generate a high output (pin 1), which is then inverted to a low through inverter 6JA (7D) and fed to one of the NOR-gate latch inputs, 7JA pin 1 (7D). The NOR-gate latch output 7JA pin 12 will go high and satisfy one input to AND-gate 8JA pin 1 (7C). The other input to this AND-gate comes directly from the decoder gate 8JC pin 8 (4D) which produces a high at a count of 00 (gc). The AND-gate 8JA (7C) will be satisfied when the counter reaches 00, and will produce a high output 8JA pin 3. This output in turn will satisfy NOR-gate 9JB pin 5 and produce a low output (pin 4). Since this output is the B input (pin 5) to single-shot 6H (7C), the single-shot will no longer be satisfied and will stop producing its 1-microsecond output pulses. With no pulses from the pulse generator the counter stops counting and remains at 00 until started again with the start switch. If the SPCK only had been used to stop the clock (as in the I/O instructions), the clock could be re-started by the start switch or by the RESM since the SPCK only would not change the state of NOR-gate latch 9HA (8E). If CL is used to stop the clock, it will stop immediately because the CL signal goes to the clear input of the flip-flops in the grey-code counter 8GA pin 1, and 8GB pin 13 (4D, 3D) setting it to count of 00 (gc). The CL also goes through the switching circuit to produce the low input required to shut down the pulse generator. As we stated, the clock always stops with 00 (gc) in the counter.

You should now understand how the clock works, what its function is, and be able to relate the input signals to the output signals. Remember, the clock is the heart of the computer. Without it, nothing else would function.

## REVIEW QUESTIONS 2-3

Refer to logic sheet 7 to answer the following questions.

1. What is the count in the two D-type flip-flops (8GA and 8GB) when  $\overline{CP2}$  is low?
  - a. 00 (gc)
  - b. 01 (gc)
  - c. 10 (gc)
  - d. 11 (gc)
  
2. To stop the clock, a low must be felt on pin 5 of U121-6H (coordinate 7C). When can this low occur?
  - a.  $\overline{ENABLE}$  is low
  - b. The two D-type flip-flops are in the 1 state
  - c. The two D-type flip-flops are in the 0 state
  - d.  $\overline{SPCK}$  is high
  
3. What allows the two single-shots to output continually to trigger the CP counter?
  - a. A low on pin 5 and an upclock on pins 3 and 4
  - b. A low on pin 5 and a downclock on pins 3 and 4
  - c. A high on pin 5 and an upclock on pins 3 and 4
  - d. A high on pin 5 and a downclock on pins 3 and 4

### D-Register

The D-Register (Distributor) is a 4-bit upcounter which is part of the timing and control circuits in the COM-TRAN 10. Its function is similar to that of the clock in that it provides timing pulses needed to synchronize and control the machine operation. When the COM-TRAN 10 is carrying out an instruction, the operation is divided into major steps. For example, when executing an LDA instruction, first data is transferred from memory to the buffer, then from the buffer to the A-register, and finally one of the various condition codes is set before starting the next instruction. Within each of these major steps, there are various minor steps. For example, during the memory to buffer transfer, the computer first reads a word from memory, then it transfers that word to the Z bus, and finally transfers it from the Z bus to the buffer. Thus, there are two timing requirements, one to tell the machine which major step it should do, and another to tell it which minor step within that major step to perform. The D-register produces counts from 00 (10) to 15 (10) called distribution pulses which time and control the major steps, while the clock produces clock pulses used to time and control the minor steps. Each distribution pulse lasts 6 microseconds, and three clock pulses, CP1, CP2, and CP3 occur during each distribution pulse. The clock and the distributor have basically the same function to control, time, and properly sequence the steps required during machine operations.



Refer to the logic diagram of the D-register (sheet 5 KDA-3034) during this discussion. There are two peculiar things about this circuit that you must understand before going through the actual operation. First, if you look across the bottom of sheet 5 you will see five D-type flip-flops which make up the counter. This is a 4-bit upcounter; the fifth (E) flip-flop, 1GA (7B) is not part of the counter but is used to tell the computer whether it is in Acquisition or Execution phase. If the true output of the E flip-flop is low (0) the computer knows it is in Acquisition phase and the distribution pulses are called DPA0, DPA1, etc. However, if it is high (1) the machine knows that it is in Execution phase and the distribution pulses are called DPO, DP1, etc.

The second peculiarity of the distributor has to do with the counter. It functions as an upcounter, however, the flip-flops are actually counting down from 15 (10) to 00 (10). Look at the outputs from this counter, and you will discover that they are reversed. The true condition, D0, D1, D2, D3, is taken from the  $\bar{Q}$  sides, while the false condition is taken from the Q sides. Since the outputs determine the type of counter, and since the true outputs are counting up, it is called an upcounter. Keep this in mind as you study the distributor, so that you do not become confused - the counter output is always the complement of the actual flip-flop state. This is also true for flip-flop E.

You should now be ready to learn the operation of the D-register by tracing the logic signals through it. If you were getting ready to operate the computer, you would first push the stop switch, then the clear switch, and finally the start switch. Assuming the stop switch has been pressed, when the clear switch is pressed the signal  $\bar{CL}$  (9F) goes low. This signal enables OR-gates 1JB and 1JC (9D), producing a low output from 1JB pin 6 and 1JC pin 8. The low output from 1JB goes directly to the S input of the E flip-flop, 1GA (7B), causing the flip-flop to set producing a low (0) output at E. Remember, this tells the computer that it is in Acquisition phase. At the same time, the low output from OR-gate 1JC (9D) is fed to the S input of each of the counter flip-flops 1GB, 1FA, 1FB, 1EA (6B, 5B, 2B) causing them all to go to the set state. This gives all lows (0s) out of the counter for a count of 00 (10). Thus, the CL signal has prepared the computer to start in the Acquisition phase at DPA0. Pressing the start switch will start the clock, producing CP1, CP2, and CP3.

Clock pulse one is used by the distributor only if the computer is being manually loaded; a read or a write being executed by a panel switch. Its purpose is to set the distributor to a count of zero. Clock pulse two is used to set the E flip-flop to the Execution phase and also used during a transfer from the input register to the distributor. Its purpose is to transfer data on the Y bus into the counter and the E flip-flop. Thus, you can see that CP1 and CP2 are not used by the distributor except in special cases. Therefore, this discussion will mainly be concerned with CP3 and its purpose in the D-register. CP3 (F8) satisfies one input to AND-gate 2HA (8D). Pin 2 of that gate is always satisfied, that is, high, unless you are doing a manual load or using the REPEAT function in the Distributor Mode. Under any other conditions OR-gate 1JD (8D) is inhibited, producing a high at the input pin 2 of AND-gate 2HA. Assuming normal operation in program Mode, both inputs are satisfied on AND-gate 2HA anytime the computer clock produces clock pulse three. Therefore, AND-gate 2HA is inverted to a low through the inverter 2KA (8C). The low output of the inverter is then fed to the trigger (T) input of each of the four D-type flip-flops which make up the counter. You know from previous discussions that CP3 will be high for 2 microseconds and then it goes back low. When this transition back to its low condition occurs, AND-gate 2HA (8D) will be inhibited at pin 1, causing its output to go back to a low condition. The low will then be inverted to a high through the inverter 2KA, and then fed to all the trigger inputs. Thus, on the upclock of CP3 the trigger inputs go low, then 2 microseconds later on the downclock of CP3 the trigger inputs go high. (See figure 2-12.) This produces the upclock at the trigger which is required for the data transfer which you learned about in the previous discussion of the D-type flip-flops. Therefore, you can see that every CP3 will clock the counter.

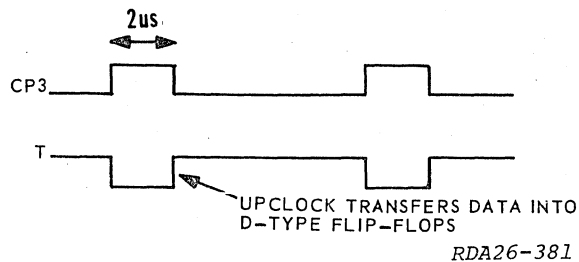
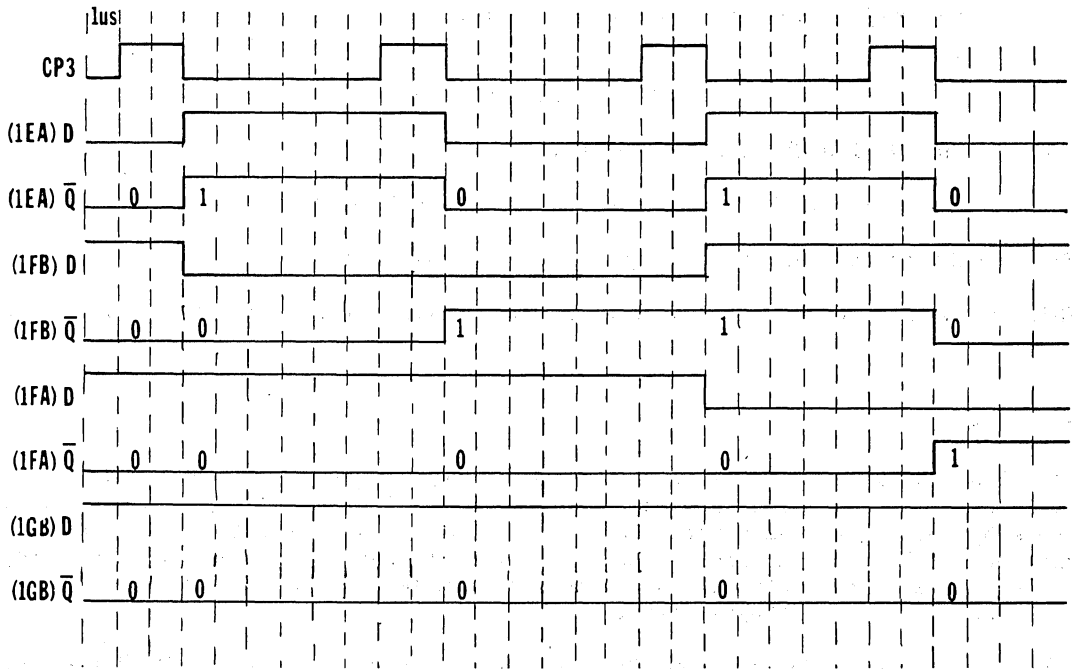


Figure 2-12. Waveshape Comparison of CP3 to T-Inputs

Take a look at what data is being clocked into the counter at these times. Remember, at the beginning the machine was cleared, setting all the Distributor counter flip-flops, producing all lows out at the  $\bar{Q}$  sides. Start tracing the signals with LSD output, D0, which is the  $\bar{Q}$  output of flip-flop 1EA (2B). The low output, D0, is fed around to inverter 2JA (2D), inverted to a high which satisfies one input to AND-gate 2GE (2D). The other one input to AND-gate comes from inverter 2KB (7D) and is always high unless one of the special count signals ( $\overline{SPDI}$ ,  $\overline{SDP4}$ ,  $\overline{SDP6}$ , etc.) (7F) is present. Assuming normal counting for now, none of these are present. Therefore, both inputs to AND-gate 2GE are satisfied and it produces a high output which in turn satisfies OR-gate 2GD (2C). The OR-gate then gives a low output which is felt at the D input to flip-flop 1EA (2B). The LSD output, D0, also goes to the Exclusive OR-gate 3FA (3D) making that input a low. The other input to the Exclusive OR-gate comes from the  $\bar{Q}$  output, D1, of flip-flop 1FB (3B). This signal is also low because there is a count of 0000(2) in the counter. Two low inputs to the Exclusive OR-gate inhibit the gate producing a low output, which is an input to AND-gate 2GB (3D). A low input at that gate will inhibit it, generating a low output, which is an input to AND-gate 2GB (3D). A low input at that gate will inhibit it, generating a low output which is input to OR-gate 2GC. With the OR-gate also being inhibited, there is a high output at pin 6. Thus, a high is felt at the D input of flip-flop 1FB (3C). By similarly tracing the D inputs to flip-flops 1FA and 1GB (6C,5C), you will find that with a count of 0000 (2) they would both be high. Therefore, with a count of 0000 (2), the D inputs are 1GB = 1, 1FA = 1, 1FB = 1, and 1EA = 0. When CP3 occurs these inputs will be transferred to the Q side outputs, thus producing a Q side count of 0001 (2). In other words, the counter was clocked from an output of 0000(2) to 0001(2) or from DPA0 to DPA1. Refer to figure 2-13, the waveshapes for the distributor counter, and trace the signals to observe how the counter clocks from 0000(2) to 0100(2).

The counter will count this way whether it is in the Acquisition or the Execution phase. The only difference is whether the E flip-flop output is high or low. At the end of each of these phases the signal CKE (8F) is generated. CKE is the input to pin 4 of AND-gate 2HB (8D) satisfying that input. The other input to that gate is always high, unless you are operating in the AE Mode with the Repeat switch set. Therefore, AND-gate 2HB is satisfied, producing a high at pin 2 of AND-gate 1LA (7C). The other input to AND-gate 1LA is satisfied when CP3 occurs, and a low output is felt at pin 3 of 1LA for 2 microseconds. This output is fed to the trigger input of the E flip-flop 1GA (7B). When CP3 is over, the T input will go back to high, thus producing the upclock required to transfer data to the Q side output. Notice that the Q side output is returned to the D input of the E flip-flop. Therefore, when it is triggered the condition of  $\bar{Q}$ , high or low, will be transferred to the Q side, causing the  $\bar{Q}$  side to reverse. Remember, this



RDA26-454

Figure 2-13. Waveshapes for the Distributor Counter

occurs at the end of the Acquisition and the Execution phases so that the computer will know that it is time to change phases.

Sometimes, an instruction does not need all the distributor counts, and it will skip to the next count it needs rather than count sequentially. The required special count signal will be generated, and the counter will be set to that count. As an example, SDP6 (7F) will produce a count of 0110(2) rather than the next count in order. SDP6 satisfies OR-gate 3HA (7E) producing a high output. This high goes to one leg of AND-gates 1HA, 1HF, 2GA, and 2GF (6D, 5D, 3D, 2D) enabling each of those inputs. The high output of OR-gate 3HA is also inverted through 2KB (7D) to a low and fed to one input of AND-gates 2HB, 1HE, 2GB, and 2GE (6D, 5D, 3D, 2D) inhibiting each of these gates. You should recall that these gates are the ones used to sequentially upcount the counter, and they are now inhibited so that a special count of 6 can be entered. SDP6 also goes to the inputs of OR-gates 3GA and 3GC (6E, 3E) satisfying these gates and producing highs out. The high output of 3GA is inverted to a low through 2JC (6D) and then used to inhibit AND-gate 1HA (6D). This produces a low output used to inhibit OR-gate 1HC producing a high at the D input to flip-flop 1GB (6B). Tracing from OR-gates 3GB, 3GC, and 3KA (5E, 3E, 2E) to their respective D inputs, you see that the D inputs are 1FA = 0, 1FB = 0 and 1EA = 1. Thus on CP3 1001(2) is transferred from the D inputs to the Q side of the flip-flops, producing a count of 6 (10). The other special count signals work the same way to produce their respective counts.

You should now understand the basic operation of the D register. It counts sequentially from 0000(2) to 1111(2), unless a special count signal is generated. The output condition of the E flip-flop determines whether these counts are interpreted as DPA

pulses or DP pulses. These pulses are all used in timing and control, so that the computer knows what major step it should be executing.

#### REVIEW QUESTIONS 2-4

1. What is the PW of a distribution pulse?
2. The true conditions, D0, D1, D2, and D3 are taken from which flip-flop output?
3. What clocks the D-register through its sequential counts?
4. Which flip-flop in the D-register will have a high (1) on the D input when  $\overline{SDPI5}$  goes low?

#### Input Register

The Input Register (I-register) is a 10-bit register that has three main functions. First, it can be used to manually input data into the A, B, C, D, S, M, P, Q, and X Registers by loading the data into the Input Register and pressing one of the respective switches below the HEX switches. Next, it is used to manually load memory by using the Manual Input (MNI) instruction. The third function is to display data from memory by using the Manual Output (MNO) instruction.

The logic diagram for the COM-TRAN 10 Input Register is on sheet 10 (KDA-3034). Refer to that diagram for the following discussion. In row C you will see the ten D-type flip-flops that make up the I-register. The clear switch on the front of the COM-TRAN 10 clears all register indicators except the Input Register. To clear the Input Register you must press the RESET switch located to the right of the input indicators. When this switch is pressed, you generate RST (10F) which is then low and clears the Input Register. RST goes to OR-gate 11MA (10C) pin 1 and produces a low on pin 3. This output goes to the clear input of all the flip-flops in the Input Register and puts them in the clear condition.

There are three ways to load the Input Register. One is by the individual Input switches labeled 0 through 9. Another is by the HEX switches 0 through F. The third way is by transferring inputs from the "Y" bus. The first method to follow in logic is the use of the individual data input switches, 0 through 9. When you press one of these switches, you generate a low signal which is fed to the SET input of the respective flip-flop. To follow this in logic, assume that the Input Register is cleared and you pushed input switches 0, 2, 5, 7, and 9. This would generate the signals  $\overline{IS0}$  (2F),  $\overline{IS2}$  (3F),  $\overline{IS5}$  (6F),  $\overline{IS7}$  (8F), and  $\overline{IS9}$  (9F). When these signals come in as lows, they set the respective flip-flops. With these flip-flops in the set condition and the others clear, the Input Register would now be loaded with 10 1010 0101.

The second way to load the Input Register is by means of the sixteen HEX switches located on the front panel below the Input Register indicators. By pressing one of these HEX switches you will generate one of the signals  $\overline{HI}$  through  $\overline{HF}$  (2-6F) and another signal called ICLK (9F). ICLK is a delayed one (1) microsecond positive pulse that will occur when any one of the HEX switches are pressed. If you follow this signal in logic, you will notice it goes to the TRIGGER input of every flip-flop in the Input Register. This will then be used to gate in whatever is on the Data input.

To follow this operation in logic, assume that the Input Register is still loaded with 2A5 of the previous operation. Now, pressing HEX switch "A" you will generate  $\overline{HA}$  (4F) and the signal ICLK (9F). When  $\overline{HA}$  (4F) goes low, this low will be felt on pin 4 of OR-gate 16KA (5D), and pin 2 of OR-gate 14KA (3D). The outputs of these OR-gates then go high. These highs are felt at the DATA inputs of both flip-flops 13MA (4C) and

12MA (3C). Since OR-gate 15KA (4D) and 13KA (2D) have no input going low, their inputs are at the static condition of a high. This will cause their outputs to be low and these lows will be felt at the DATA inputs of flip-flops 13MB (3C) and 12MB (2C). Before bringing in the ICLK (9F) and clocking in this data to the four LSB flip-flops notice that the Q outputs of these flip-flops are feeding back to the DATA inputs of the next group of four flip-flops. The Q output of Flip-Flop 12MB (2C) feeds back to the DATA input of 14MB (5C), the Q output of 12MA (3C) feeds back to the DATA input of 14MA (6C), the Q output of 13MB (3C) feeds back to the DATA input of 15MB (7C), and the Q output of 13MA (4C) feeds back to DATA input of 15MA (8C). Notice that the Q outputs of 14MA (6C) and 14MB (5C) also provide feedback to the DATA inputs of 16MA (9C) and 16MB (8C). Remember, whatever is felt at the DATA input of these flip-flops when the TRIGGER comes in will be felt on the Q output. Before HEX switch A was pressed the Input Register contained 2A5. Now, with the present conditions of 2A5 in the Input register and DATA inputs of 10 0101 1010, bring in ICLK (9F). This was generated by pressing the HEX A switch and is a delayed one (1) microsecond positive pulse. With the flip-flops triggered the Input Register now contains whatever was on the DATA inputs or 10 0101 1010. Notice what was in the six least significant bits is now transferred to the six high order bits, and A was placed in the four least significant bits. Each time one of the HEX switches is pressed the action just described will take place.

It was stated that there are sixteen HEX switches on the front panel. Only fifteen of these generate a signal to the decoder network at 2 through 5EF on the logic. HEX switch 0 is different in that it does not come to this decoder network, therefore, the output will be at a low static condition and place low's on the DATA inputs of the 4 least significant flip-flops. However, HEX switch 0 will generate ICLK (9F) to trigger the flip-flops and place zeros in the four LSB flip-flops. The same transfer for the other bits will take place as previously described.

The third way to load the Input Register is from the "Y" bus. This loading will occur when using the Manual Output (MNO) instruction. When reading from memory data transfers to the "Z" bus and into the Buffer. The static condition of the Selector allows this true data to be felt on the "Y" bus. This "Y" bus data comes into one input of the eight AND-gates located at 2C through 7C on the logic. Notice, that if the input is a high coming in, this input on the AND-gate, will satisfy that leg. The other input to the eight AND-gates comes from TYI (10F) which is gated on another logic sheet by MNO and DP2. TYI will be present for six microseconds, so at CP1 (10F) AND-gate 11LA (10C) will be satisfied and produce a low output. This low goes to the input of OR-gate 11MA (10C) and its output goes low to clear all the flip-flops in the Input Register. Data is clocked into the B Register at CP2 time so at that time its true output will be on the "Y" bus (2F through 10F). With TYI (9F) still present and data on the "Y" bus, this information will now be gated to the SET input of the 8 LSB flip-flops. For example, if the "Y" bus is high this will satisfy one leg of the respective AND-gate, and TYI will satisfy the other. The AND-gate would then produce a low output to the SET input of the flip-flop. If data on the "Y" bus was low, the AND-gate would not be satisfied and the flip-flop would remain in its clear condition.

Now that you know the three ways to load data into the Input Register, you need to look at how it transfers this data to other registers. The Q side of all ten flip-flops transfers out as I0 through I9 to the "P" Register (sheet 12). Thus, all bits in the Input Register can transfer to the "P" Register. The eight LSB bits I0 through I7 are felt at the Selector input (sheet 3) and, if the Input Register is selected, these bits will transfer to the "Y" bus. Bits I8 and I9 transfer to the Memory Register Selector (sheet 11, 8E), and if the transfer of Input Register to Memory Register occurs, these bits will be loaded in the "M" Register along with the "Y" bus data. Data in the Input Register can also go to the Buffer Register but this will transfer by way of the "Z" bus. The Q output of the eight low order bits is felt at one input of eight AND-gates (sheet 10, 1B through 7B). The other input to each of these gates is TIB (sheet 10, 7F) which will place DATA or the complement data on the Z bus.

You should now know the purpose of the Input Register and the three ways it can be loaded. Also, you should be able to explain the operation of the Input Register and develop the output required.

#### REVIEW QUESTIONS 2-5

Refer to logic sheet 10 to answer the following questions.

1. List the ways to load the Input Register with information.
2. What produces the D input to flip-flop I6?
3. What clock pulse clears the I-Register when it is being loaded from the Y-bus?
4. Which hex input switch is not shown on sheet 10 but generates ICLK when pressed?

#### B-Register

The B-register, more commonly known as the Buffer, is the register which is used as the link between memory and the rest of the computer. Anything that goes into memory or comes out of memory must go through the Buffer. This register is made up of two 4-bit parallel load storage registers. The logic diagram (sheet 3) actually shows three separate circuits; the B-register, the Two's Complementer, and the Selector. The heavy dash lines divide the logic into these three circuits, identifying each on the left side of the logic diagram. This discussion will cover the B-register and the Selector circuits completely, and the Two's Complementer briefly.

The B-register logic is found in the upper section of sheet 3, KDA-3034. Be sure that you use this logic diagram to trace the signals as you read the operational explanation. Notice that each of the 4-bit storage registers 7E (7E, 8E) and 7D (3E, 4E) are identical, with four data inputs across the top of each register, two control inputs at the left of each register, and eight outputs across the bottom of each register. The data inputs to the Buffer always come from the Z-bus, with  $\overline{Z0}$  through  $\overline{Z3}$  feeding register 7D (3E, 4E) and bits  $\overline{Z4}$  through  $\overline{Z7}$  feeding the other register 7E (7E, 8E). Since the Z-bus data is all NOT signals, the B-register inputs are actually the one's complement of the data you are trying to store. However, the B-register inverts the inputs internally and therefore has the true data at its outputs. The control inputs, pin 4 and 13, to the B-register are used to tell it when to load the input data. Even though there are two control inputs, they actually function as one, because they are shorted together. CP2 and TZB (Transfer the Z-bus to the Buffer) are the two inputs to AND-gate 8F (9F). If both of these signals are present, the AND-gate would be satisfied, producing a low output at pin 3. This low is felt at one input to OR-gate 8FB (9E) causing it to satisfy and produce a high output. The output of the OR-gate is felt at the control inputs of both registers 7E (8E, 7E) and 7D (3E, 4E). Whenever these control inputs upclock, the B-register will load the data on the Z-bus into the B-register.

Take another look at OR-gate 8FB (9E) and notice that its output is always low until one of its inputs is satisfied, and then the output goes high. This change from low to high is an upclock condition. (It was previously mentioned that the OR-gate 8FB can be satisfied by the CP2 and TZB signals being present.) It can also be satisfied by the  $\overline{CL}$  signal (9F). The  $\overline{CL}$  signal is generated when you press the clear switch on the COM-TRAN 10 control panel, and will satisfy the pin 5 input to the OR-gate causing it to produce the required upclock to load the Buffer. Pressing the Clear switch causes the Z-bus to clear (this is done elsewhere in the logic circuitry) so that when the  $\overline{CL}$  signal produces the upclock, the B-register loads all zeros from the Z-bus. Therefore, any data entering the B-register will be clocked in from the Z-bus.

The Output-Transfer circuitry is made up of the B-register output lines, and the Selector circuit. Across the bottom of each of the B-register chips, 7E (7E, 8E) and 7D (3E, 4E) you will see eight output pins. There are two outputs for each input, one with a State indicator and one without, for example, on chip 7D, pins 1 and 16 are both outputs for the data entering on pin 2  $\overline{Z0}$ . The output lines without state indicators, pins 9, 10, 15 and 16, produce the complemented output signals called  $\overline{B0}$ ,  $\overline{B1}$ ,  $\overline{B2}$ , etc. As in other registers these complemented signals are used to control the lights on the COM-TRAN 10 display board. The output lines with the state indicators, pin 1, 14, 11, and 8, feed the true data from the B-register to the Selector circuit.

The Selector circuit is made of four IC chips, 9F, 8E, 8D, and 8C (row C), called Dual 4 to 1 Data Selectors. This means that for every 4 input lines there is one output line; however, in the COM-TRAN 10 only three input lines are used for each output line. Each Selector chip has two output lines, pins 7 and 9, and, therefore, two sets of three input lines. These inputs come from the B-register, the I-register, and the Two's Complementer. Look at chip 8C (2C); pins 10, 11, and 12 are the three inputs for output pin 9. Pin 10 comes from the B-register true value output for  $B0$ , pin 11 comes from the Two's Complementer, and pin 12 comes from signal  $I0$ , which is bit zero of the I-register. Notice that each of the Selector circuits outputs go through an amplifier and produce output signals called  $Y0$  through  $Y7$ , this is the Y-bus.

Now consider how the Selector knows which input signal to place at the output. Each Selector chip has two control signal inputs at the left end, pins 2 and 14. The signal which feeds the pin 2 control input is TIY (9F), Transfer Input Register to the Y-bus. If TIY is present, it will place a high on pin 2 and leave pin 14 low, which tells the Selector chip to transfer the I inputs (pins 4 and 12) to the output. The COMP signal (10F), Complement, feeds the pin 14 control input of each Selector chip. If this signal is present, it will place a high on pin 14 leaving pin 2 low, and the selector will transfer the Two's Complementer inputs (pins 5 and 11) to the output pins. If neither of these two control signals is present, then both pins 2 and 14 would be low and the Selector would transfer the B-register inputs (pins 6 and 10) to the output pins. This means that with no inputs applied to the selector (static state), the B-register contents are felt on the Y-bus. To summarize, if pin 2 is high and pin 14 is low, the I-register inputs will be selected; if pin 2 is low and pin 14 is high the Two's Complementer inputs will be selected; and if both pins 2 and 14 are low the B-register inputs will be selected (see figure 2-14). Regardless of which inputs are selected the outputs will be to the Y-bus.

Pin 2	Pin 14	Selects
L	L	B Register
H	L	I inputs
L	H	Two's Comp

RDA26-455

Figure 2-14. Selector Truth Table

So far you have traced two types of outputs from this sheet of logic, the Y-bus outputs from the Selector, and the Complemented B outputs  $\overline{B0}$ ,  $\overline{B1}$ ,... $\overline{B7}$ , directly from the B-register outputs. There is one more output signal called  $\overline{BZ}$  (4B) which is used to indicate that the B-register contains all zeros. The  $\overline{BZ}$  signal is produced by AND-gate

10CA (4B). There are eight inputs to this gate, one from each of the Complemented B-register outputs. If all of the inputs are high, which would mean that the B-register contains all zeros, then AND-gate 10CA would be satisfied producing a low on output  $\overline{BZ}$ .

In the center of this sheet of logic, Row D, you will see a series of AND-gates and Exclusive OR-gates. This is the Two's Complementer. Notice that its inputs come from the Z-bus, not the B-register. Remember in order to two's complement a number you first take the one's complement and add a one to it, and since the Z-bus is already in the one's complement form ( $\overline{DATA}$ ), it is only necessary to add this one. The Two's Complementer circuit is nothing more than an adder, which will add a one to the Z-bus if the signal TWO's (2F) is high (Two's complement) or a zero if the signal TWO's is low (one's complement). The output from this circuit feeds one of the Selector inputs, and would be transferred to the Y-bus only when signal COMP goes high (10F).

This section has covered three circuits: The B-register, the Two's Complementer, and the Selector. The B-register and the Selector are the most significant at this time. The B-register inputs come from the Z-bus, while its outputs produce the complement signals to control the panel lights, and the true signals which are fed through the Selector to the Y-bus. The Selector circuit takes three inputs, and by interpreting its control signals determines which one should go to the Y-bus at any specified time.

#### REVIEW QUESTIONS 2-6

To answer the following questions refer to KDA-3034, sheet 3.

1. Where would the scope lead be placed to observe I7 going into the selector?
  - a. 9F pin 4
  - b. 9F pin 12
  - c. 8E pin 12
  - d. 8E pin 4
  
2. If the A and B inputs are low to the selector U153-9F (sheet 3, 8C), what is felt at pin 7?
  - a. pin 4 input
  - b. pin 5 input
  - c. pin 6 input
  - d. one's complement input
  
3. If the A and B inputs to the selector are low (normal condition), what is on the Y-bus?
  - a. The I-register
  - b. The B-register
  - c. The B-register complement
  - d. The I-register complement



## Memory Module

The memory module is where the COM-TRAN 10 stores all the information you have given it. Computers cannot think, they can only follow the instructions which they are given. Therefore, when you want the machine to perform some function, you must write a program and store it in memory. The computer will then go to its memory and get one instruction at a time and execute it. There are many different types of memories used in computers, some use magnetic core memory, some use electrostatic memory, and some use IC memory. The COM-TRAN 10 uses an IC memory, which is similar to using flip-flops to hold the information. The only problem with this is that should the machine lose power the memory information will be lost. This does not mean turning the power switch off, for that has no effect, but should the computer become unplugged, or should a loss of power to the building occur, the memory information would be lost and your program would have to be reloaded. As you follow the logic operation of the memory module, you should refer to sheet 8 in KDA-3034, and relate the material to the logic diagrams.

The large block in the center of the logic diagram labeled MODEL 104, IC Memory, contains the actual timing and control circuitry of the memory. However, as a Maintenance specialist you have no control over that circuitry and no access to the circuitry, since it is contained in an IC chip. Therefore, you will not be concerned with the internal workings, but only with those external signals and their effects which can be observed, measured, and controlled. The memory module is used anytime the machine is required to read or write information. In the upper left-hand corner of the memory chip, you will see an input called Read/Write (7E). In order to write information into memory this input must be low, and to read information from memory it must be high. Across the top of the chip, you will see a series of ten input lines labeled  $\overline{M0}$  through  $\overline{M9}$  (4F, 5F). These ten signals are used to identify which memory location, 000(16) through 3FF (16), from which you are reading or to which you are writing. The address input lines and the READ/WRITE input line are the only ones used during both a read and write operation. All other circuitry is used for one or the other. Each operation will be discussed separately.

First, assume that during the machine operation, it is required to write a word into memory. Prior to initiating the write, certain conditions must have already been accomplished. The address of the memory location into which you are writing, is already present in complemented form on the address input lines, and that information will be decoded internally by the IC chip. The data to be written into memory is already present on the data input lines Y0 through Y7 (3F). Therefore, the memory knows what to write and where to write it before you actually start the write process. When the computer gets ready to actually begin writing, it will generate the  $\overline{I\overline{B}S}$  signal, Initiate Buffer to Storage. This signal is felt at input pin 4 of AND-gate 9LB (7E) as a low. The low will inhibit the AND-gate, and produce a low output, pin 6. The output of the AND-gate is fed to the READ/WRITE input, and since it is a low, the memory chip executes a write. It takes the data from the Y input lines and stores it in the location identified by the  $\overline{M}$  address lines. After six microseconds, one distribution pulse, the  $\overline{I\overline{B}S}$  goes back to its normal high condition, and the write sequence is ended.

The read operation is a little more complex. First, take a look at AND-gate 9LB (7E) again. Since a write sequence is not being executed, the  $\overline{I\overline{B}S}$  has not been generated. Therefore, pin 4 of the AND-gate feels a high input which satisfies that leg. The input to inverter 7 MB (8E) is tied to a +5V, which is a high, and which is inverted to a low through the inverter. The low output of the inverter is felt at the input of OR-gate 8MB (8E) satisfying the gate and producing a high output, which in turn is felt at the input of AND-gate 9LB (7E). This means that both inputs to the gate are high at all times except when a write signal is generated. The AND-gate produces a high output since it is in a satisfied condition, which is felt at the READ/WRITE input telling the chip to perform a read sequence. Thus the memory is continually reading the locations identified by the  $\overline{M}$  address lines. However, this does not mean that the computer is continually reading. It only means that the memory chip is continually providing data at the data output lines (along the bottom of the chip).

In order for the computer to accomplish a read sequence, the data must be transferred from the data output lines to a computer register. This is accomplished by a transfer circuit, which you will find below the memory module. It is made up of 8 NOR-gate latches and 8 AND-gates. When the computer is ready, it generates the  $\overline{ISB}$  signal, Initiate Storage to Buffer. The  $\overline{ISB}$  signal comes through inverter 7MA (9D), where it is inverted from a low to a high. The high is then felt on one input to AND-gate 8MC (9C) satisfying that leg, and also on one input to each of the transfer AND-gates (Row B), satisfying each of those legs. CP1 feeds the other input of AND-gate 8MC (10C). Thus when CP1 occurs the AND-gate is satisfied producing a 2-microsecond negative pulse at its output, pin 8. The low output is fed to one input of each of the NOR-gate latch circuits (Row C), causing the outputs of each latch to go to a high. The purpose of this is to temporarily set all the latch circuits, so that at the end of CP1 when the input to the latches goes back high the data output lines from memory can load the proper value into the transfer circuits. The data output lines which have a 0, low, on them will reset the latches, while the lines carrying a 1, high, will have no effect, leaving the latches set as they were by CP1, thus coding the transfer circuit with the proper information from memory.

As the data is loaded into the NOR-gate latches their outputs are felt on the transfer AND-gates. Those latches with 1 outputs will satisfy their respective AND-gates producing low outputs to the Z-bus, and those latches with 0 outputs will inhibit their respective AND-gates producing high outputs to the Z-bus. Thus as CP1 ends, that is downclocks, the Z-bus is loaded with the complement of the data read from memory. This is necessary because the Z-bus lines are "NOT" signals,  $\overline{Z0}$ ,  $\overline{Z1}$ ,...which means they must be opposite the true condition. At this time the read sequence is completed as far as the memory module's part is concerned.  $\overline{ISB}$  enables the transfer AND-gates, and combines with CP1 to set the latch circuits. Then as CP1 downclocks, the data lines reset those latches necessary to give the proper binary code, and then the AND-gates transfer the data to the Z-bus, and the read sequence is ended.

In summary, the memory is used for all read or write sequences. The read/write input and the memory address lines are common to both processes. The Y-bus data lines are used to input data during a write. The data output lines and the transfer circuits are used during a read to transfer data from the memory to the Z-bus. The Z-bus always receives the complement of the actual data.

#### REVIEW QUESTIONS 2-7

Refer to sheet 8 to answer the following questions.

1. Which of the following signals would not be used to write information in memory?
  - a.  $\overline{ISB}$
  - b. Y BUS input
  - c.  $\overline{M0}$  through  $\overline{M9}$
  - d.  $\overline{ISB}$
  
2. If the data word being read from memory was FF(16), what voltage level would be felt at gate 7K-A pin 3 (B7)?
  - a. 0 volts
  - b. +5 volts

3. How long is the  $\overline{\text{ISB}}$  signal low?
  - a. 2 microseconds
  - b. 6 microseconds
  - c. 8 microseconds
  - d. as long as the LDA instruction is in the S-register

### M-Register

The Memory Address Register, M-register, is a 10-bit register used to locate any one of the 400(16) words in memory. This address selection is necessary anytime you want to read or write. The M-register has 10 flip-flops which can be loaded in several different ways. You will find the M-register logic diagram on sheet 11 of the KDA-3034. Notice that this register has a fairly complex Input-Control circuit, but an extremely basic Output circuit.

There are four sources for loading the M-register: the B-register, P-register, I-register, and the Index Adder. These inputs actually enter the M-register through either the Y-bus, Z-bus or G-bus. Addresses for instructions are transferred to the M-register from the P-register through the Z-bus. When addresses for operands are loaded into the M-register, the lower 8 bits are transferred from the B-register through the Y-bus, while the two most significant bits are determined by S0 and S1 of the OP Code register. To manually load an address into the M-register, the contents of the I-register are transferred through the Y-bus for the 8 least significant bits, while M9 and M8 are loaded from I9 and I8 through the selector. Whenever the computer is executing an indexed instruction, the indexed address (M + X) is loaded into the M-register from the Index Adder through the G-bus.

Before actually tracing the signals for these various loading methods, you must take a look at chip 13J (8E) which is a Dual 4 to 1 data selector. This chip is identical to the ones you studied in the selector portion of the B-register. You will recall from that discussion that the COM-TRAN 10 uses these chips as Dual 3 to 1 data selectors. There are three inputs for each output, and the control inputs to the chip are used to determine which input is transferred to the output. This chip is required by the M-register because it is a 10-bit register. As noted above, the inputs to this register sometimes come through the Y-bus and Z-bus. These two busses are only 8-bit bus lines, and therefore cannot carry the two most significant bits on an M-register input. To solve this problem the data selector has six inputs, three for each output, which are I9, P9, and S1 for one output (pin 9), and I8, P8, and S0 for the other (pin 7). If the computer is transferring from the I-register to the M-register, the control signals tell the data selector 13J (8E) to transfer I9 and I8 to its output. If the transfer is a P to M, inputs P8 and P9 will be selected; and for a B to M, inputs S0 and S1 are selected. The only other transfer possible is the X-register to M-register through the G-bus, and since the G-bus is a 10-bit bus, the data selector is not required.

Since the G-bus input is the least complex it will be traced first. Across the bottom of sheet 11 you will see 10 D-type flip-flops (Row B), which make up the storage circuit for the M-register. The D input to each of these flip-flops is fed directly from one of the bits of the G-bus GO through G9. In order to transfer this data into the flip-flops, the signal TGM (9F) must be present. TGM is one of the inputs to AND-gate 15HB (9C), and when it is present it will satisfy that input leg. The other input to this gate is CP3. When CP3 is generated, CP3 goes low satisfying that leg of the AND-gate. When the AND-gate inputs are both satisfied, its output goes from a static condition of low to its satisfied condition of high. This causes an upclock to be fed to the

T input of each of the flip-flops. When the T input upclocks, the data on line D is transferred into the flip-flop. If TGM is present, on the downclock of  $\overline{CP3}$ , an upclock occurs at the T input to each flip-flop, loading the flip-flops with data from the G-bus. This is the only type of input to the M-register which uses the D input to load.

When the computer executes an operation which requires the M-register to be loaded from the P, I, or B-registers, the first thing it must do is clear the M-register. The  $\overline{TBM}$  (Transfer B to M), the  $\overline{TIM}$  (Transfer I to M), and the  $\overline{DPA8}$  (Transfer B to M) signals are the three inputs to OR-gate 15JB (9F). If any one of these signals are present the OR-gate will be satisfied, and produce a high output. This high output is fed to one input of OR-gate 15HA (8E) satisfying that gate and producing a low output. The  $\overline{TPM}$  (Transfer P to M) signal is the only input to inverter 13GA (10F). When this signal is present, the inverter inverts it from a low to a high and feeds it to the other input leg of OR-gate 15HA (8E), satisfying that gate so that it produces a low output. Thus, if any of the signals  $\overline{TBM}$ ,  $\overline{DPA8}$ , or  $\overline{TPM}$  are present OR-gate 15HA (8E) will satisfy and produce a low output. Then this output is fed to one input of OR-gate 14JA (9D) satisfying that gate to produce a high output, which is fed in turn to AND-gate 14JB (10D) pin 5 satisfying that leg. The other leg (pin 4) is fed by CP1 and will satisfy when the clock produces CP1. With both inputs satisfied, the gate is satisfied and produces a low output, which is fed to input pin 9 of OR-gate 14HC (10C). This will satisfy the OR-gate and it will produce a low output which is fed to the C inputs of flip-flops 11JA (9B) and 11JB (8B). When these C inputs go low, the flip-flops are cleared to the zero state. Now, if you go back to the output of OR-gate 15HA (8E), you will see that its low output is also fed to inverter 14FA (9D). The inverter will invert the signal from a low to a high and feed it to AND-gate 14JC (9D), where it is ANDed with CP1. When the AND-gate satisfies, it will produce a low output which in turn satisfies one input to OR-gate 14HD (9C). Notice that the output of this OR-gate is tied to the C inputs of all the other D-type flip-flops, so when it is satisfied and its output goes low, the 8 least significant flip-flops will be cleared. For any of the transfers, P to M, I to M, or B to M, the first step (that is CP1) will clear all ten of the M-register flip-flops.

All of these transfers will also strobe the data selector 13J (8E) during CP2. In order for the data selector to interpret the control signals and make the right data output selection, the strobe input (pins 1 and 15) must go low. When you traced the logic to clear the flip-flops, you found that if any of the signals  $\overline{TPM}$ ,  $\overline{TIM}$ ,  $\overline{DPA8}$ , and  $\overline{TBM}$  are present, OR-gate 14JA (9D) will be satisfied and produce a high output. This high is fed to AND-gate 14JD (9C) where it is ANDed with CP2. Therefore, when CP2 is present during one of these transfers, the AND-gate will be satisfied, producing a low output. This output is tied directly to the strobe inputs of the data selector 13J (8E). Thus, during CP2 the data selector strobe inputs will be conditioned so that it can interpret the control signals and output the proper data.

The control signal inputs are pins 2 and 14 of the data selector. The input to pin 2 comes from the output of OR-gate 14GA (10F) and the input to pin 14 comes from the output of inverter 13GB (9F). If  $\overline{TBM}$  or  $\overline{DPA8}$  are present, pins 2 and 14 will both be low and the selector will transfer S0 and S1 to the output pins. If  $\overline{TIM}$  is present, the two control inputs will both be high causing I8 and I9 to be transferred to the output. If  $\overline{TPM}$  is present, pin 2 will be high and pin 14 will be low, causing the selector to transfer P8 and P9 to the output (see figure 2-15). These outputs are then inverted through inverters 13GE and 13GF (8C) and fed to the S input of the most significant bit flip-flops, 11JA (9C) and 11JB (8C). If either selector output is high then the inverters will invert it to a low, satisfying the low, satisfying the flip-flop so that it will go to the set or 1 state. If there is a low (0) out of the selector, it will invert to a high and have no effect on the flip-flop. This would leave it in the clear or 0 state in which it was put during CP1.

So far, it has been shown that the B, I, or P to M transfers all cause the 10 flip-flops of the M-register to be cleared during CP1. They also cause the proper control signals and the strobe signal to be fed to the data selector during CP2, causing the M8

Pin 2	Pin 14	Selects
L	L	S0 & S1
H	L	P8 & P9
H	H	<u>I8 &amp; I9</u>

Figure 2-15. Selector Truth Table

and M9 flipflops to be loaded. Now all that remains is to see how the other 8 flip-flops (M0-M7) are loaded during these transfers. The  $\overline{\text{TPM}}$  signal causes these flip-flops to load data from the Z-bus.  $\overline{\text{TPM}}$  is inverted to a high through inverter 13GA (10F). This high is fed to AND-gate 14HB (7D), where it is ANDed with CP2. When  $\overline{\text{TPM}}$  and CP2 are both present, the AND-gate will be satisfied, producing a high output. This output is one of the inputs to each of the AND-GATES 12JA, 12JF, 12HA, 12HF, 12GA, 12GF, 13FA, and 13FF (all located in Row D). The other input to each of these AND-gates is coming from the Z-bus through an inverter. These data bits are inverted because the Z-bus is a complemented bus and therefore must be inverted before it is loaded. In order to see how this data is loaded, assume that the Z-bus data is 1110 1011(2). Each bit is first inverted by the inverters in Row E, so that the data is now fed to the AND-gates as 0001 0100(2). This means that AND-gates 12GF and 12HF feel a high from the Z-bus inputs and from  $\overline{\text{TPM}}$  input and will be satisfied, producing a high output. The high outputs are fed to the OR-gates 12GD (3C) and 12HD (4C), respectively. These OR-gates will be satisfied and will produce a low output which is fed to the S inputs of flip-flops 11FB (3C) and 11GB (5C). When the S inputs feel the low, the flip-flops will be set to the 1 state. The rest of the flip-flops will remain in the 0 state, thus loading the 8 least significant flip-flops with 0001 0100(2).

If any of the signals  $\overline{\text{TBM}}$ ,  $\overline{\text{TIM}}$ , or  $\overline{\text{DPA8}}$  are present the transfer will occur approximately the same way as the  $\overline{\text{TPM}}$  transfer except that the 8 flip-flops are loaded from the Y-bus rather than from the Z-bus. Notice across the top of sheet 11, that the Y inputs are all true inputs (Y0, Y1, Y2...) not complemented inputs like the Z inputs ( $\overline{\text{Z0}}$ ,  $\overline{\text{Z1}}$ ,  $\overline{\text{Z2}}$ ,...). This means that the data coming from the Y-bus does not require a set of inverters like the Z-bus data. The Y-bus data (Y0 - Y7) feed directly to AND-gates 12JB, 12JE, 12HB, 12HE, 12GB, 12GE, 13FB, and 13FE. If  $\overline{\text{TIM}}$ ,  $\overline{\text{DPA8}}$ , or  $\overline{\text{TBM}}$  are present OR-gate 15JB (9F) will be satisfied, producing a high output. The output is tied to one input of AND-gate 14HA (7E). The other input to this AND-gate comes from CP2 (9F), and when both inputs are present the AND-gate will satisfy, producing a high output. This high output will be fed to one input of each of the Y-bus input AND-gates listed above. Thus, if any one of the Y-bus bits is a one, its AND-gate will satisfy, and will set its respective flip-flop to the one state through the proper OR-gate. This procedure is identical to the Z-bus transfer during  $\overline{\text{TPM}}$ .

In summary, there are 4 input possibilities for the M-register. One is to transfer the G-bus data into the M-register. The G-bus is a 10-bit bus and is transferred directly into the M-register using the D and the T inputs of the M-register flip-flops. The other three possibilities are P to M ( $\overline{\text{TPM}}$ ), I to M ( $\overline{\text{TIM}}$ ), or B to M ( $\overline{\text{DPA8}}$  or  $\overline{\text{TBM}}$ ). Any of these signal inputs will clear all 10 flip-flops of the M-register during CP1. They will then produce the proper control and strobe signals for the data selector during CP2, and thus load the two MSB flip-flops. Also during CP2,  $\overline{\text{TPM}}$  will load the 8 LSB flip-flops from the Z-bus;  $\overline{\text{TIM}}$ ,  $\overline{\text{DPA8}}$ , or  $\overline{\text{TBM}}$  will load from the Y-bus. You should now be able to trace any of the four possible M-register inputs.

The M-register outputs are very simple. Notice that each of the 10 flip-flops has two outputs, one from the Q side and one from the  $\bar{Q}$  side. The Q side outputs are called M0, M1, ... M9, and are true value outputs. The  $\bar{Q}$  side outputs are called  $\bar{M}0$ ,  $\bar{M}1$ , ...  $\bar{M}9$ , and are the complemented outputs. These complemented outputs are used to address the memory module and control the front panel lights. You should be able to determine the effect on the M-register and the outputs generated, when given specific input conditions.

#### REVIEW QUESTIONS 2-8

Refer to logic sheet 11 to answer the following questions.

1. What signal is necessary to step the Memory Address Register by one?
  - a.  $\overline{TGM}$
  - b.  $\overline{TPM}$
  - c.  $\overline{TIM}$
  - d.  $\overline{TBM}$
2. What select levels are needed on pins 2 and 14 of data selector 13J, to transfer S0 and S1 to M8 and M9?
  - a. pin 2 high and pin 14 high
  - b. pin 2 high and pin 14 low
  - c. pin 2 low and pin 14 high
  - d. pin 2 low and pin 14 low
3. From what register does the M-register receive information at DPA8?
  - a. S-register only
  - b. I-register via the Y-bus
  - c. B-register and S-register
  - d. P-register via the Z-bus
4. What signal will load the M-register from the Z-bus?
  - a.  $\overline{DPA8}$
  - b.  $\overline{TBM}$
  - c.  $\overline{TIM}$
  - d.  $\overline{TPM}$

#### X-Register

The Index Register is an 8-bit storage register used for indexing the operand address. This means that any time an instruction is used as an index instruction

the X-register contents will be added to the M-register contents. The logic diagram for the X-register and the X Adder are located on sheet 14 of KDA-3034.

The X-register is made up of two 4-bit Selector/Storage chips, 14E (8F) and 15D (4F). These are the same type of chips that are used in the P-register. There are two inputs for each output on these chips. Across the top of the chips you will see that the two input choices to the X-register are the M-register (M0 through M7) and the Y-bus (Y0 through Y7). Each chip has a clock input (pin 10) which must downclock in order to transfer the selected input to the output pins. Each chip also has a MODE input (pin 9). If this input is high the Y-bus data will be loaded, but if it is low the M-register will be loaded. Anytime the computer must load the X-register, the signal LX is generated. LX is ANDed with CP2 at AND-gate 15CA (9F). If both of the signals are present the AND-gate satisfies and produces a low output. This low will satisfy the OR-gate 15CB (9F) causing its output to go high. The output of this OR-gate is tied to the clock input of each 4-bit Selector/Storage chip. So when the OR-gate satisfies, the clock input goes high. At the end of CP2, pin 1 of AND-gate 15CA (9F) will be inhibited and the output will go back to high. This, in turn, inhibits OR-gate 15CB (9F) causing its output to go from high to low. This downclock causes the chips to load the selected data. With only the LX signal generated the Y-bus will be the selected input, since the static condition of the MODE input is high. In order to load the M inputs, the signal  $\overline{TMX}$  (8F) must also be present. This input is tied directly to the MODE input of both chips, and when  $\overline{TMX}$  is generated the MODE input goes low. Then when LX and CP2 combine to generate the downclock for the clock input, the chips will load the M-register inputs. These are the only two ways X-register can be loaded.

Once the chips have clocked the selected input data to the output pins, there are three possible output signals for each bit of the X-register. Only one bit will be traced, as all work the same way. Let's take pin 13 of chip 15D (4F), for example. This pin is tied to inverter 14DA (4E), which will invert the signal and produce the complemented output  $\overline{X1}$ . Pin 13 is also tied to one input leg of AND-gate 14CC (4E). The other leg is fed by the signal TXB, Transfer X to B, which causes the X-register to be loaded on to the Z-bus. If TXB is present one leg of each Z-bus transfer gate will be enabled, including AND-gate 14CC. Now if X1 is high (1) the gate will be satisfied and will output a low (0) as  $\overline{X1}$ , but if X1 is low (0) the AND-gate will be inhibited and will produce a high (1) as  $\overline{X1}$ . Therefore, a complemented signal is transferred to the Z-bus. The other possible output is generated in this case by AND-gate 12CD (4D). Pin 12 of this gate is enabled by the signal AXM (9F), Add X to M. Then if X1, the other input to the AND-gate, is high (1) the gate will produce a high output; if not, it produces a low output. This output is tied to pin 8 of chip 11D (3B), which is part of the X adder and receives the X input when it needs to add X to M.

In summary, the three possible outputs from each bit of the X-register are the complemented output ( $\overline{X0}$ ,  $\overline{X1}$ , ...,  $\overline{X7}$ ), the Z-bus output ( $\overline{Z0}$ ,  $\overline{Z1}$ , ...,  $\overline{Z7}$ ), and the X Adder signals required for the Add X to M operation. There is also a possible  $\overline{XZ}$  signal generated when the X-register contains zeros. It is generated by AND-gate 11CA (4B). It functions just like zero contents signal in the other registers, previously covered.

Now look at the X Adder circuit and notice that it is made up of two 4-bit Adder chips, 11E (7B) and 11D (3B). These chips are very simple in operation. They do not require any control signals or any transfer output signals. Each chip has eight inputs, 4 from the M-register inputs and 4 from the X-register outputs. These 4-bit Adder chips continually add these two data inputs together and place the results out on the G-bus. Unless the AXM signal is present, the X inputs to the Adder chips are always 0000 0001(2). Thus you can see the X Adder is a very simple circuit in operation.

You should now be familiar with both the X-register and the X Adder operations. Given specific input conditions you should be able to determine the resulting effects and the resulting output signals.

## REVIEW QUESTIONS 2-9

Refer to logic sheet 14 to answer the following questions.

1. What signals are necessary to add the Index Register to the M-register and transfer the results to the M-register?
  - a. TXB
  - b.  $\overline{\text{TMX}}$  and CP2
  - c. LX and CP2
  - d. AXM,  $\overline{\text{TGM}}$ , and  $\overline{\text{CP3}}$
2. What happens if the sum of the Index Register and the M-register exceeds 10-bit places?
  - a. The Index Register end carries
  - b. The overflow bits are lost
  - c. The carry light comes on
  - d. The ADD overflow light comes on
3. What logic levels are felt on pins 1, 3, 8, and 10 of IC's 11E and 11D when the AXM signal is low?
  - a. Contents of the X-register
  - b. 11111110(2)
  - c. 00000001(2)
  - d. 00000000(2)

### P-Register

The Program Address Register (P-register) is a 10-bit storage register. It is used by the computer to determine the next instruction to be executed. It is made up of three 4-bit Selector/Storage chips plus the required Input/Control and Transfer/Output circuitry. See sheet 12 of the KDA-3034, Circuits and Diagrams, for the logic diagram of the P-register.

In Row E of sheet 12, you will see three chips, 16J, 16H, and 16G. These are the 4-bit Selector/Storage chips. Notice the inputs to these chips (across the top of each chip) are M0 through M9 and I0 through I9. This tells you that the data inputs to the P-register come from either the M-register or the I-register. Look at chip 16G (3E), and you will see that it has 8 input lines (across the top) but only 4 output lines (across the bottom). There are two inputs for each output. That is why the chip is called a 4-bit Selector; it must select which of the two inputs it is to transfer to the output. As you go from right to left across the top of the chip you will see that there is one I-register input, then one M-register input, then one I-register input, etc. These are grouped I0, M0, etc., and when the chip is loaded it will either load all the I inputs or all the M inputs, never M and I inputs mixed. The Selector knows which input to load by the condition of its Mode input, pin 9. If the Mode input is



high it will load the I-register inputs, but if the Mode input is low it will load the M-register inputs. Notice chip 16J (8E) has only 4 inputs and 2 outputs, but it works just like the other two chips, two inputs per output. There are two input transfer signals,  $\overline{TMP}$ , Transfer M to P, and  $\overline{TIP}$ , Transfer I to P. Both of these signals are inputs to OR-gate 16FA (9F). If either signal is present, the OR-gate will be satisfied and produce a high output. The high output is ANDed with CP2 at AND-gate 16FB (9E). Therefore, if the computer is transferring data into the P-register, CP2 will cause the AND-gate to satisfy, producing a low output which is fed to the input of OR-gate 16FC (9E). This will satisfy the OR-gate, generating a high output which is tied to the clock input (pin 10) of each of the three 4-bit Selector/Storage chips. Notice that this clock input pin indicates that it is activated by a downclock. This means that the selected data will be transferred to the outputs when the clock input gets a downclock. We have just seen that during any  $\overline{TMP}$  or  $\overline{TIP}$ , when CP2 occurs the clock input to the chips goes high. Then when CP2 downclocks 2 microseconds later, AND-gate 16FB (9E) will be inhibited causing its output to go high. This inhibits OR-gate 16FC (9E) and its output will go back low. Since the OR-gate output is tied to the three chip clock inputs, when it goes from a high output back to low these inputs are satisfied by a downclock. Thus at the end of CP2, the selectors will be clocked and will transfer the selected data to the outputs of the 4-bit Selector/Storage chips. If the transfer signal present is  $\overline{TMP}$  (9F) it will have no effect on the MODE input to the chips. Therefore, it will remain in its static condition of low, and when the chips are clocked the M inputs will transfer to the output pins. However, if  $\overline{TIP}$  (9F) is the transfer signal used, it will be inverted to a high through inverter 16EA (9E) and fed to the MODE input pins of all three chips. The MODE input being high will cause the I inputs to be transferred to the output pins when the chips are clocked. So there are two possible ways to load data into the P-register.

Once the input data has been transferred to the output pins of the 4-bit Selector/Storage chips, there are two possible outputs from the P-register. One of these outputs is always present, it is the complemented output signal called  $\overline{P0}$ ,  $\overline{P1}$ , ...  $\overline{P9}$ . For example, look at the output pins of chip 16H (6E). Each of these outputs is fed through an inverter. The output at pin 11, is inverted through inverter 15GF and becomes the P-register output,  $\overline{P4}$ . By tracing each output pin, you will find that each is inverted to become a P-register complemented output. These outputs are used to control the front panel lights.

The other P-register output is the Z-bus outputs  $\overline{Z0}$  through  $\overline{Z7}$ . There is no  $\overline{Z8}$  and  $\overline{Z9}$  output because the Z-bus is only an 8-bit bus line. In addition to being tied to an inverter each output is also tied to the input of an AND-gate. For example, pin 14 of 16G (3E) is tied to pin 5 of AND-gate 16DB (3C). If you check the other output pins, you will find that each one is tied to an AND-gate input pin. The other input to the 8 LSB AND-gates comes from the signal TPLB, Transfer the P-register Lower Order Bits, (10). Anytime this signal is present it will enable one leg of each of these AND-gates, 15EA, 15EB, 15EC, 15ED, 16DA, 16DB, 16DC, 16DD. Then if the output pin is a high (1) its AND-gate will be satisfied, producing a low at its Z-bus position; if the output pin is low (0), its AND-gate will be inhibited, putting a high (1) of its Z-bus position. Therefore, if TPLB occurs, whatever is in the 8 LSB bits of the P-register will be placed on the Z-bus in complemented form.

There is one other output to be considered, and that is the true signal output for the two MSB bits P8 and P9. The two output pins of chip 16J (8E) have a direct line output called P8 and P9. Remember these signals are required by the M-register during the P to M transfer. So the complemented output signals for all 10 bits, the Z-bus outputs for the 8LSB bits, and the true output for the 2 MSB bits, are all available from the P-register.

You should now be able to take any given input conditions and determine the P-register output signals, as well as the general resulting effect on the circuit.

## REVIEW QUESTIONS 2-10

Refer to logic sheet 12 to answer the following questions.

1. What signals are needed to load the P-register from the M-register?
  - a.  $\overline{\text{TMP}}$  low,  $\overline{\text{TIP}}$  high, and CP2
  - b.  $\overline{\text{TMP}}$  high,  $\overline{\text{TIP}}$  high, and CP2
  - c.  $\overline{\text{TIP}}$  low, TPLB, and CP2
  - d.  $\overline{\text{TMP}}$  low,  $\overline{\text{TIP}}$  low, and CP2
  
2. Which bits of the P-register can be transferred to the M-register without the use of the Z-bus?
  - a.  $\overline{\text{P0}}$ ,  $\overline{\text{P1}}$ ,  $\overline{\text{P2}}$ ,  $\overline{\text{P3}}$ ,  $\overline{\text{P4}}$ ,  $\overline{\text{P5}}$ ,  $\overline{\text{P6}}$ , and  $\overline{\text{P7}}$
  - b.  $\overline{\text{P0}}$  and  $\overline{\text{P1}}$  only
  - c. P8 and P9 only
  - d. All bits
  
3. If we are transferring the contents 3FF from the P-register to the M-register, what inputs are felt by the selector and AND-gates of the M-register?
  - a. P8 and P9 high,  $\overline{\text{Z0}}$  through  $\overline{\text{Z7}}$  low
  - b. All inputs high
  - c. All inputs low
  - d. P8 and P9 low,  $\overline{\text{Z0}}$  through  $\overline{\text{Z7}}$  high

### A-Register (Accumulator)

The A-register or Accumulator is the register used in arithmetic operations to hold the augend, minuend, multiplicand, and two high order bytes of the dividend before the divide operation; and the sum, difference, two high order bytes of the product, and the remainder after such operations. This register is an 8-bit, left shift, right shift, parallel load register. The logic diagram for the A-register can be found on sheet 1 of the Circuits and Diagrams book, KDA-3034. Refer to this diagram during the following discussion. On sheet 1, in coordinate sections 8F, 8E, 7F, and 7E, you will see 4 AND-gates, 4KA, 4KB, 4JA, 4JB; and one OR-gate 4HA; and in coordinate section 8C one AND-gate, 4HB. These gates are not part of the A-register, but rather are part of the control circuitry used during an ADD or SUB instruction. Therefore, these gates will be ignored during this discussion. The explanation of the A-register operation will be divided into two major parts, the Input-Control circuitry and operation and the Output-Transfer circuitry and operation.

All of the data input signals and all of the control signals are fed directly to IC chip 5G (4E, 5E). This chip is the actual register, with the various storage devices contained in it. However, as in the memory module, the internal workings of the chip are NOT relevant to you as a maintenance technician and, therefore, this discussion will be limited to the signals feeding the chip and their effects on the chip. There are three sources of data inputs to the A-register, the left shift input, the right shift

input, and the parallel input. The left and right shift inputs are serial, that is, loaded one bit at a time. The signal DSLA, which feeds pin 22 of 5G (4E, 5E), is the left shift data input; and the signal DSRA, which feeds pin 2 of 5G, is the right shift data input. If the data input line is high, then a one will be shifted in, and if it is low, then a zero will be shifted into the accumulator. Across the top of IC chip 5G (4E, 5E) you will see 8 input lines labeled F0 through F7, which are also called the F-bus. These lines are the parallel data inputs; that is, all eight bits of data are loaded into the A-register at one time.

Now that you know where the data comes in, the next step is to continue with an explanation of how you control the A-register so that it knows which type of loading it should do, and when it should do it. On the lower left side of chip 5G (4E, 5E) there are four control line inputs. These are pin 13, clear; pin 23, shift left; pin 1, shift right; and pin 11, clock. The clear signal is generated when you press the clear switch on the COM-TRAN 10 control panel.  $\overline{CLA}$  (6F) goes low satisfying the clear input which clears the A-register to all zeros. If the shift left control input goes high, the register knows to shift the data in left; if the shift right control input goes high, the register knows to shift data in right. When both of these inputs go high at the same time, then the register knows to execute a parallel load. These two input controls tell the register how to load the data. The purpose of the clock input (pin 11) is to tell the register when to load the data. When the clock input upclocks the data loads, either serially shifting or straight parallel as indicated. Notice the clock input signal comes from  $\overline{CP2}$ , (6F), therefore when CP2 is generated by the clock circuit,  $\overline{CP2}$  will downclock, then at the end of CP2, 2 microseconds later,  $\overline{CP2}$  will upclock back to its static condition, and the register will execute what it has been told to do. The clear signal is not dependent on the clock, but rather clears as soon as the  $\overline{CLA}$  is generated. Thus there are three data inputs which make up the Input-Control circuitry.

Now that you know how to load data in, consider how you get data out. The 8 pins across the bottom of IC chip are the output pins, and whatever data is in the A-register will be present at these output pins. Each of the outputs produce three possible output signals. This is accomplished by the transfer circuit. All the inverters, AND-gates, and phantom OR-gates in Rows B, C, and D make up the transfer circuitry (except AND-gate 4HB, sheet 1, 8D which was previously excluded). Notice that each output pin is connected to an inverter, an AND-gate, and has one direct line output. For example, pin 20 has a direct line output labeled A0, which is bit 0 of the A-register; an output through inverter 6GF (2D) labeled  $\overline{A0}$ , which is the complement of bit 0 of A-register and is used to control the indicator lights of the computer's front panel; and is fed to AND-gate 7GD (2C) as an input. The other input to the AND-gate comes from signal TAZ (9F), Transfer the A-register to the Z-bus. When this signal is present the gate would be satisfied if the A-register bit was a one, producing a low output on the Z-bus called  $\overline{Z0}$ . It would be inhibited if the A-register bit was a zero and a high output would appear on the Z-bus as  $\overline{Z0}$ . This circuitry works the same way for each output bit. Each A-register bit produces a true condition as A0, A1; a complemented output through inverters as  $\overline{A0}$ ,  $\overline{A1}$ , etc.; and a complemented output through the AND-gates, as  $\overline{Z0}$ ,  $\overline{Z1}$ , etc., if TAZ has been generated by the computer.

There is one other output signal generated by AND-gate 5KA (5B). There are eight inputs to this gate, one from the complement of each A-register bit. If the A-register contained all zeros then the complements would all be ones or highs and the AND-gate would be satisfied, thus producing a low output. The low output would then be inverted through inverter 5HC (5B) to a high output called AZ. This signal is used elsewhere in the computer to indicate that the A-register contains all zeros.

You should now be able to determine what the A-register will do, and what the outputs will be if you know what inputs and what control signals are present.

## REVIEW QUESTIONS 2-11

Refer to sheet 1 to answer the following questions.

1. What signals are needed to parallel load from the F-bus to the A-Register?
  - a. SAR, SAL and  $\overline{CP2}$
  - b. CLA and CP2
  - c. SAL and  $\overline{CP2}$
  - d. SAR, SAL, and DSLA
  
2. The F-bus inputs are 01010101(2) TAZ and pins 23 and 1 are high. When  $\overline{CP2}$  upclocks, what is felt on the Z-bus?
  - a. 01010101(2)
  - b. 10101010(2)
  - c. 00101010(2)
  - d. 01010100(2)
  
3. The A-register contains 11110000(2) and pins 22 and 23 are high. After 3 upclocks on pin 11, what is felt on the complemented lines,  $\overline{A0}$  through  $\overline{A7}$ ?
  - a. 10000111(2)
  - b. 11111110(2)
  - c. 01111000(2)
  - d. 11110000(2)

### ALU Module

The Arithmetic Logic Unit, ALU, is the module which performs all the arithmetic operations. The logic diagrams for the ALU is on sheet 2, KDA-3034. The ALU is made up of two 4-bit arithmetic and logic unit chips, 5F and 5E. These two chips are tied together so that they operate as one unit using 8-bit values. The COM-TRAN 10 uses the ALU for seven different arithmetic and logic functions. This unit is actually capable of doing 48 separate functions, but in the COM-TRAN 10 it is only wired for seven of these. Therefore, this discussion is concerned with these seven functions only and will cover these seven only.

The ALU has two sets of inputs, the A inputs, which come from the A-register, and the B inputs, which come from the Y-bus. Look at sheet 2, and across the top of the 4-bit ALU chips, you will see these two sets of inputs. What the ALU module does with these inputs is determined by the control signal inputs, pins 3, 4, 5, 6, and 8 of the chips. These pins are fed by the outputs of OR-gates 5DA (9E), and 4CB (8E) and the inverted outputs of OR-gates 4CA (9E), 5CA (8E), and 5DB (7E). These OR-gates act like a decoder circuit and set up the proper control signals for the input signal present. With the proper code at the five control input lines, the ALU will perform the following arithmetic or logic operations:

Arithmetic:

1. Add A input to B input.
2. Subtract B input from A input.
3. Increase A input by one.
4. Decrease A input by one.

Logic:

1. The logical AND of A input with B input.
2. The logical OR of A input with B input.
3. The logical Exclusive OR of A input with B input.

The relationship between the input signal, the control code, and the resulting action is shown in the ALU truth table (see figure 2-16). You should be able to trace each signal through the OR-gate decoder to see how each code is set. If none of the seven input signals are present, the ALU is in its static condition and will automatically do a parallel transfer of the B input (Y-bus) to the ALU's output pins. These are the eight possible actions of the ALU as it operates. Seven of them are arithmetic or logic function, and the other is a straight transfer.

Input Signal	Condition Code					Resulting Action
	S3	S2	S1	S0	M	
$\overline{DECA}$	H	H	H	H	L	Decrease A input by one
$\overline{INA}$	H	L	L	H	L	Add A input to B input
$\overline{INS}$	L	H	H	L	L	Subtract B input from A input
$\overline{INCA}$	L	L	L	L	L	Increase A input by one
$\overline{IAND}$	H	L	H	H	H	Logical AND of A with B
$\overline{IORI}$	H	H	H	L	H	Logical OR of A with B
$\overline{IEX}$	L	H	H	L	H	Logical Exclusive OR of A with B
No Inputs Static state	H	L	H	L	H	Parallel Transfer of B to F-bus

RDA26-456

Figure 2-16. ALU Control Inputs

The output pins of the ALU produce direct output signals called F0, F1,...F7. This is the F-bus and it goes directly to the A-register inputs. Notice that the F7 output signal is also fed through inverter 4DD (sheet 2, 6B) to produce  $\overline{F7}$ .

You should now be able to determine what the ALU will do and what its outputs will be when you are given the input signals present.

#### REVIEW QUESTIONS 2-12

1. What are the selection inputs to the ALU for an ADD instruction?
  - a. S0 and S3 high, S1 and S2 low, mode low
  - b. S0 and S3 low, S1 and S2 high, mode low
  - c. S0, S1, and S3 high, S2 low, mode high
  - d. S0, S1, S2, and S3 low, mode low
2. With the following selection, S3, S1, and M high, S0 and S2 low, what is felt on the F-bus?
  - a. Y-bus
  - b. Accumulator
  - c. Accumulator added to the Buffer
  - d. Accumulator ANDeD with the Buffer
3. What register is both an ALU input and an ALU output?
  - a. Buffer
  - b. Accumulator
  - c. Quotient
  - d. None of the above

#### Q-Register

Like the A-register, the Q-register, or Quotient register, is an eight-bit, left shift, right shift, parallel load register. In fact, if you take a look at the logic diagram for the Q-register (sheet 6, KDA-3034) you will see that it is identical to the A-register, except for some of the signal names. The Q-register is used in arithmetic operations to hold the two lower bytes of the dividend during a divide, the quotient after a divide, and the two lower bytes of the product after a multiply. Since the operation of this register is just like the A-register, the explanation will be brief.

The Input-Control circuitry for the Q-register functions, as in the A-register, with three data input lines and four control lines. The left shift data comes into pin 22 of 3E (4E), from a signal called DSLQ. The right shift data comes into pin 2 of the same chip from a signal called A0. This is bit zero of the A-register. Whenever the Q-register is shifting to the right, it is actually loading data from bit A0 of the A-register. Across the top of chip 3E (6E, 5E) you find the parallel input lines. Notice

that the Q-register parallel inputs come from the Y-bus, Y0 through Y7, while the A-register's parallel inputs came from the F-bus. The four control signals are identical to those on the A-register, and work the same way. Pin 13 is the clear input and when  $\overline{CL}$  is present, the Q-register will be cleared to all zeros. SQR (7F, 8F) are signals which tell the register how to load. If SQR goes high, the register shifts right, if SQL goes high, the register shifts left, and if both signals go high together, the register loads in parallel. The clocking input tells the register when to execute the identified loading procedure. It is controlled by  $\overline{CP2}$ , and when  $\overline{CP2}$  upclocks (at the end of CP2) the register loads the data. The Q-register operates just like the A-register for input data and control, with the only exception being where some of the signals come from.

Across the bottom of chip 3E you will see the output pins and output lines. Each bit has only two possible outputs, while the A-register has three. The Q-register does not produce direct output signals, but rather produces one complemented output per bit ( $\overline{Q0}$ ,  $\overline{Q1}$ , etc.) through inverters, and if the signal TQZ (Transfer Q to the Z-bus) is present, produces a transfer to the Z-bus ( $\overline{Z0}$ ,  $\overline{Z1}$ , etc.) through the transfer AND-gates. The one exception to this is bit 7 of the Q-register, which does produce one direct output called Q7 (8B), which is the true condition for bit 7 of the Q-register. The Q-register also produces one signal,  $\overline{Q6Q0}$ , which the A-register does not produce. The complemented signals from each bit Q0 through Q6 are fed as inputs to AND-gate 1CA (5B), and if they are all high, (the Q-register contains zeros in all those bits) then the gate is satisfied and produces a low output called  $\overline{Q6Q0}$ . This signal is used during the DIV instruction. The output of that gate is also fed to the input of AND-gate 1DA (6B) where it is ANDed with Q7. If Q7 is low, that is, a zero, and  $\overline{Q6Q0}$  is low meaning all the other bits are zero, then the gate is satisfied and produces signal QZ, which indicates that the Q-register contains all zeros.

You should now be able to determine what effect the input-control signals will have on the Q-register and what its outputs will be for any given input conditions.

#### REVIEW QUESTIONS 2-13

1. The Q-register contains FF (16) and pins 1 and 2 are both high. After 8  $\overline{CP2}$  pulses are supplied, what is felt at pin 1 of AND-gate 1DA and pin 8 of AND-gate 1CA?
  - a. QZ is high  $\overline{Q6Q0}$  is low
  - b. QZ is low  $\overline{Q6Q0}$  is low
  - c. QZ is low  $\overline{Q6Q0}$  is high
  - d. QZ is high  $\overline{Q6Q0}$  is high
  
2. If the accumulator contains 0A (16) and The Q-register contains 55 (16), what will the Q-register contain after executing an SRA 04 instruction?
  - a. 0000101(2)
  - b. 11110101(2)
  - c. 01000101(2)
  - d. 10100101(2)

3. What single instruction allows the programmer to load the Q-register with the contents of the A-register?

- a. LDQ
- b. STQ
- c. SRA
- d. SRL

#### C-Register

The C-register, or Countdown Register, is an 8-bit down-counter used in certain sequenced instructions (multiply, divide, shift, and skip) and during Input/Output instructions. It is called a decision counter because it is used to determine when the computer has repeated a certain series of steps within an instruction the right number of times. For example, if the computer was executing a SRA instruction and you had specified that it was to shift 3 places, then the computer would load the C-register with a count of 3. It would then start shifting and after each shift it would check to see if the C-register contained all zeros. If the C-register did not contain all zeros, the computer would decrease the C-register by one and jump back to the DP pulse which caused the shift. It will repeat this process until the C-register contains all zeros. The register is made up of 8 D-type flip-flops, which make up the down-counter circuit, and the necessary Input-Control circuitry and Output circuitry. Use sheet 4 of KDA-3034, Circuits and Diagrams, so that you can relate the following explanations to the logic diagram.

In the center of sheet 4, you will see the 8 D-type flip-flops (Row D) that make up the down-counter circuit, with the LSB on the right, MSB on the left. Its maximum count is FF (16); however, it can be loaded with any number less than FF (16). Regardless of the count it is loaded with, it will always countdown to zero. In order to explain how the down-counter works, assume the counter is loaded with FF (16). Notice the wiring arrangement between the flip-flops, the  $\bar{Q}$  side output of each flip-flop is fed back to its D (data line) input, and with the exception of the MSB flip-flop 5MA (9D) the Q side output of each flip-flop is tied to the T (trigger line) input of the next higher bit flip-flop. Remember that D-type flip-flop operates so that an upclock at the T input causes the data on the D input to be transferred to the Q side output. Assuming a count of FF (16), all flip-flops are in the one state; i.e., a high (1) at the Q side output and low (0) at the  $\bar{Q}$  side output. This means that there is a low being fed back to the D input of each flip-flop. Since the LSB flip-flop 3MB (2D) obviously cannot be triggered by the next lower bit flip-flop, it must be triggered by other means. The two inputs to AND-gate 4LA (2D) are the signals DEC (Decrease) and CP3. If DEC is present when CP3 is produced, the AND-gate 4LA will be satisfied and will produce a high output, which is tied to the T input of flip-flop 3MB (2D). Prior to generation of CP3, AND-gate 4LA was inhibited and its output was low. As CP3 upclocks to a high, the AND-gate output will go from a low to a high (assuming that DEC is already present). That high is felt at the T input of flip-flop 3MB as an upclock. This will trigger the flip-flop and the low at the D input will be transferred to the Q side output. Thus the Q side goes from a high (1) to a low (0) and the  $\bar{Q}$  side goes from a low (0) to a high (1). It was already pointed out that the Q side output is tied to the T input of the next higher bit position flip-flop, in this case 3MA (3D). Remember the LSB flip-flop Q side output went from a high to a low, thus feeding a down-clock to the T input of flip-flop 3MA. Since the D-type flip-flop requires an upclock to activate, the down-clock has no effect, and flip-flop 3MA remains in the state it was in, Q side = high (1) and  $\bar{Q}$  side = low (0). Therefore, all other flip-flops in the counter will remain the same and the new count will be 1111 1110(2) or FE (16), counting down by one. By tracing logic in this manner, you will see that every time DEC is present, the counter will decrease its value by one on every CP3.



Now that you understand how the counter counts, you need to take a look at how the original count, or starting place, gets loaded into the counter in the first place. Actually there are three methods of loading the C-register with its original count, by generating the signal  $\overline{SE}$ , or the signal  $\overline{SC8}$ , or by loading from the Y-bus. Generating the signal  $\overline{SE}$  always loads the counter with FF (16). This signal is generated during all manually controlled input/output operations; that is, anytime you push one of the four switches labeled RD MEMORY, WT MEMORY, READ INTRPT, or WRITE BLOCK on the COM-TRAN 10 control panel. The  $\overline{SE}$  signal is an input to OR-gate 6MA pin 1 (10E). When  $\overline{SE}$  is generated, it will satisfy the OR-gate which will in turn produce a high output at pin 6. The high output is felt at one of the inputs to AND-gate 5LA satisfying that leg. Then when CP1 (which is the other input to the AND-gate) is produced, the AND-gate will be satisfied, producing a low output. This low output is fed to the S (set) input of each D-type flip-flop in the counter, which sets them all to the 1 state, for a count of 1111 1111(2) or FF (16). If the C-register counter is loaded using signal  $\overline{SC8}$  (9F), then the count loaded will be 08 (16). This signal is used during multiply and divide operations. Notice that  $\overline{SC8}$  does exactly the same thing as  $\overline{SE}$ . It comes into OR-gate 6MA (10E) producing a high output which is ANDed with CP1 through AND-gate 5LA, and placing a low at the S input to all flip-flops of the counter. These S input signals are present for 2 microseconds since CP1 is a 2-microsecond pulse. At the same time  $\overline{SC8}$  is also an input to OR-gate 5LB (9E). This will satisfy the gate and produce a high output, which in turn is tied to one input leg of each of the 8 OR-gates found in Row D, except OR-gate 1NB (5D). When the high is felt on these OR-gates, they will all be satisfied and will produce low outputs which are fed to the C (clear) input of each of the corresponding flip-flops. This will cause the flip-flops to clear, or go to the zero state. Since OR-gate 1NB does not have  $\overline{SC8}$  as an input, flip-flop 2MA (4D) does not clear. In brief then,  $\overline{SC8}$  and CP1 set all the flip-flops to the one state, and then  $\overline{SC8}$  clears all the flip-flops except 2MA. If you check the count, you will find 0000 1000(2), which is 08 (16).

The final way in which you could load a count is to load from the Y-bus. To do this the signal  $\overline{TYC}$ , Transfer Y to C, (10F) is used. You should notice that  $\overline{TYC}$ , just like  $\overline{SE}$  and  $\overline{SC8}$ , comes into OR-gate 6MA (10E) producing a high output which is ANDed with CP1 through AND-gate 5LA (10D). This places a low at the S input of each flip-flop, causing them all to go to the one state.  $\overline{TYC}$  is also ANDed with CP2 through AND-gate 3LA (9F). When  $\overline{TYC}$  and CP2 are both present the gate will satisfy, producing a high output which is inverted to a low through inverter 2LA (9E). The low output is then fed to one input of each of the 8 AND-gates found in Row E, satisfying those inputs. The other input to each of these AND-gates is one of the Y-bus bits, Y0 through Y7. If any of the Y bits are low (0), then the AND-gate which it feeds will be satisfied and will produce a high output. For example, if Y3 = 0 AND-gate 1NA (5E) will be satisfied producing a high output. This output is, in turn, felt on input pin 6 or OR-gate 1NB (5D) satisfying that gate. It will then produce a low output which will be felt at the C input to flip-flop 2MA (5D) causing the flip-flop to clear. The same sequence will occur for all Y-bus bits which equal 0, through their own respective AND- and OR-gates. For example, if 1011 0110(2) was on the Y-bus, the  $\overline{TYC}$  signal would set all the flip-flops on CP1, and then at CP2 it would clock through the 0's on Y6, Y3, and Y0 to clear their three flip-flops, leaving all the others set. This would load the counter with 1011 0110(2). Regardless of which one of the three ways is used to load the counter, once it is loaded with a value, it will count down each time DEC and CP3 are present.

The C-register has only two types of output signals. As in all the other registers it has a complemented output for each bit which is used to control the COM-TRAN 10 front panel lights. These outputs,  $\overline{CO}$ ,  $\overline{CI}$ , etc., come from the  $\overline{Q}$  side of each of the C-register flip-flops. The other output signal from the C-register is called CZ (6B). This signal is used to determine whether or not the register contains all zeros. Remember at the beginning of this section, it was mentioned that the C-register is used to determine when the computer has repeated some particular sequence of steps the right number of times. Therefore, whenever the COM-TRAN 10 uses the C-register, it is only concerned with whether or not it equals zero; the actual remaining count is not important. Looking at

the logic diagram you will see the  $\bar{Q}$  side of each flip-flop, in addition to producing the complemented bits, is also connected to an input leg of AND-gate 1MA (6B). If all the  $\bar{Q}$  outputs were equal to a one (high), the AND-gate would be satisfied and would produce a low output. This output is inverted to a high through inverter 2LB (6B) and called CZ (C-register equal zero).

In summary, the C-register is an 8-bit down-counter used when instructions or operations call for a repeating sequence. It can be loaded three ways - with FF (16) using  $\overline{SE}$ , with 08 (16) using SC8, and with any count, 00 (16) to FF (16), from the Y-bus using signal  $\overline{TYC}$ . Once it is loaded the register will begin to count down once each time the signals CP3 and DEC are both present. It produces two types of output signals, the complemented bits  $\overline{C0}$ ,  $\overline{C1}$ , etc., for light control, and CZ to determine when the register has reached zero.

#### REVIEW QUESTIONS 2-14

Refer to logic sheet 4 to answer the following questions.

1. What clock pulse triggers the C-register to its next count?
  - a. CP1
  - b.  $\overline{CP2}$
  - c. CP3
  - d.  $\overline{SC8}$
2. What is the content of the C-register after the  $\overline{SC8}$  has been applied?
  - a. 11111111(2)
  - b. 00000000(2)
  - c. 00001000(2)
  - d. 11110111(2)
3. What register supplies the count that is loaded into the C-register during an LCI instruction?
  - a. M-register
  - b. B-register
  - c. B-register Complement
  - d. Accumulator

#### S-Register

The Operation Code Register, S-register, is an 8-bit storage register. The S-register is used to hold the operation code of the instruction to be executed. If the instruction is one which requires memory access, then the two lower order bits of the S-register (S0, S1) are used as the high order bits of the M-register (M8, M9). This allows a 10-bit memory address capability. Use sheet 13 of KDA-3034 for reference to follow the explanation. The S-register is made up of 8 NOR-gate latches, which act

together as a storage register, and the required Input/Control circuitry. The S-register outputs are all direct outputs and, therefore, no Output/Transfer circuitry is required.

The most common method of loading data into the S-register is to load from the Y-bus. This is either during  $\overline{DPA2}$ , or during a manual load from the Input (I) register to the S-register. There are also four special signals, which are generated during any manual read or write, that will load a code directly without using the Y-bus. The S-register will be loaded with the data on the Y-bus if either the  $\overline{TIS}$  signal, Transfer I to S, or the  $\overline{DPA2}$  signal is present. Both of these signals are inputs to OR-gate 20GA (9E), and if either one is present the OR-gate will be satisfied. When it satisfies, it will produce a low output which is fed to OR-gate 20FA (9E). Notice that the other input to OR-gate 20FA comes from signal  $\overline{SE}$  (9F). This signal is generated by the special read or write signals mentioned above. Thus, anytime the S-register is being loaded, this OR-gate will be satisfied. It will then produce a high output which is ANDed with CP1 at AND-gate 20FB (9D). The AND-gate will be satisfied at CP1, during any S-register input sequence, and will produce a low output. This low output is fed to one input of OR-gate 20GB (9D) satisfying that gate and producing a low output from it. The output of OR-gate 20GB is tied to one input leg of the NOT side of each NOR-gate latch (Row C). Therefore, when the OR-gate's output goes low, it conditions every latch to the clear state; that is, a high (1) output from the NOT side ( $\overline{S7}, \overline{S6}, \dots, \overline{S0}$ ) and a low (0) output from the TRUE side ( $S7, S6, \dots, S0$ ). So during any input to the S-register, CP1 will cause all the latches to clear. Now, go back to OR-gate 20GA (9E), this gate is satisfied by either  $\overline{TIS}$  or  $\overline{DPA2}$ . Notice that this gate's output is also tied to AND-gate 19GA pin 3 (8E). Therefore, when OR-gate 20GA is satisfied, it will produce a low output, which will in turn satisfy pin 3 input to AND-gate 19GA. The other input to this AND-gate comes from CP2 (8F). Thus when CP2 is generated the AND-gate will be satisfied causing it to produce a high output. The output of AND-gate 19GA is tied to one input leg of AND-gates 20FC, 20FD, 20E8, 20EC, 20ED, 19EA, and 19EB (all in Row E).

When the output of AND-gate 19GA (8E) goes to its satisfied condition of high, it will enable one input leg to each of these Y-bus input transfer gates. If any of Y-bus data inputs is high (1), then its transfer AND-gate will be satisfied, producing a low output. This output is fed to the TRUE side of the NOR-gate latch, and would cause it to switch to the set state, that is a high (1) output from the TRUE side and a low (0) output from the NOT side. However, if the Y-bus data bit is a low (0), then its input transfer AND-gate will be inhibited and the latch it feeds would remain in its cleared condition. In summary, to load the S-register from the Y-bus data lines, either  $\overline{TIS}$  or  $\overline{DPA2}$  must be present. These signals will clear all the NOR-gate latches at CP1. Then during CP2 the Y-bus data will be transferred into the latches.

The four special input signals mentioned at the beginning of the section are called  $\overline{IRB}$  (6F),  $\overline{IRD}$ ,  $\overline{IWT}$ , and  $\overline{IWB}$  (5F). The special signals are generated for any manual read or write. They are set up to automatically load the proper operation code. For example, if you are going to manually load the computer from the teletype, you would press the READ INTRPT switch. Pressing the switch will generate the  $\overline{IRB}$  and the  $\overline{SE}$  signals. You have already seen that during CP1 the  $\overline{SE}$  signal will clear the latch circuits. Then the  $\overline{SE}$  signal is present as an input to latch 18DA (8C) and 18DB (7C). These inputs will cause the two latches to go to the one state. Signal  $\overline{IRB}$  (6F) is fed to the inputs of latches 19DA (6C) and 19DB (4C). Thus, when  $\overline{IRB}$  is present it will cause these two latches to go to the one state. Therefore, the S-register is loaded with 1110 1000(2), which is E8 (16). This is the Hex Code for the instruction Read Until Interrupt. If you trace the other three special inputs you will find that they are also directly wired to load the proper code for their function. The S-register, therefore, can be loaded by the Y-bus or by the direct inputs of the special signals.

The output signals from the S-register are very simple. Each latch has an output from the TRUE side called  $S0, S1, \dots, S7$ . These are the true signal outputs. All except  $S0$  and  $S1$  also have an output from the NOT side called  $\overline{S2}, \overline{S3}, \dots, \overline{S7}$ . These are the complemented outputs used to control the front panel lights. The signals from  $S0$  and  $S1$

are not needed because those two lights are controlled by  $\overline{M8}$  and  $\overline{M9}$ . Thus, the S-register has only direct outputs, and does not require any output transfer circuits.

You should now understand the S-register operation, and be able to trace through logic, any given inputs, to determine their effects on the register and their generated outputs.

#### REVIEW QUESTIONS 2-15

To answer the following questions, refer to logic (sheet 13).

1. When using the pushbuttons for the RD, WT, RDI, or WDB instructions, what NOR-gate latches are always set?
  - a. S5 and S3
  - b. S5 and S4
  - c. S7, S6, and S4
  - d. S7 and S6
2. Which NOR-gate latches of the S-register do not condition lamp drivers?
  - a. S7 and S6 do not condition lamp drivers
  - b. S0 and S1 do not condition lamp drivers
  - c. None of the latches condition lamp drivers
  - d. All of the latches condition lamp drivers
3. Where are the output voltages of the S-register felt?
  - a. M-register high order bits and Y-bus
  - b. Buffer and M-register high order bits
  - c. Instruction decoder (Control Logic) and high order bits of M-register
  - d. Z-bus and Instruction Decoder Circuits

#### PROGRAM INSTRUCTION LOGIC ANALYSIS

You are now familiar with the "how" of each unit of the COM-TRAN 10. This is similar to having all the pieces to a puzzle. You have analyzed each piece, and you know what it contains. However, until you put all the pieces together, you cannot see the overall picture. This section will take all the units and put them together to see how the whole machine operates as a functioning computer. This will be done by tracing the logic signals for selected instruction.

On sheet I of KDA-3034, you will find a chart called the Logic Timing Chart. This chart will be very important to you as you trace the instructions through the computer logic. The first column lists the instructions by their mnemonic codes, the second by their hex code, and the third by their binary code. The fourth column gives a brief description of what the instruction does. The next 16 columns represent the sixteen distribution pulses. The mnemonic indicated in each of these columns tells you what

takes place during that distribution pulse (DP) for that instruction. For example, locate the instruction STX in column one. The hex code is 50 (16), the binary code is 0101 0000(2), and it means Store the Index. By looking at the DP columns, you will see that at DPO the action will be TXB, Transfer the X-register to the B-register. Then at DPl, it will perform an IBS, Store the B-register in Memory. As you trace logic using this chart you find that the meanings of the DP mnemonics will become very familiar to you. The top line, Acquisition, has no Hex code. Acquisition precedes the Execution Phase of every instruction, and its DP pulses are decoded as DPA pulses.

#### Logic Timing

Using the same instruction, STX, look in the timing section of this chapter for the instruction STX. (The STX mnemonic symbol is located at the top of the page.) The first thing you will see is the name of the instruction does, and its mnemonic symbol. This is followed by a brief explanation of what the instruction does, and what its hex OP code is. Next, you will see a copy of the instruction's Timing Chart line exactly as it appears in the Logic Timing Chart. Now, you come to the breakdown of each DP in detail. Look at this breakdown for DPO of the instruction STX. It tells you that during DPO the signal TXB is generated and that it means Transfer Index to Buffer. This signal is then broken down into all of its subparts. First STX and DPO are ANDed to generate TXB, and if you want to see the logic, it tells you to look at sheet 23 of KDA-3034. All page references in this book correspond to the sheet numbers in KDA-3034. The only thing you need to be aware of particularly is that all signals are referred to as true signals. When you look up the logic for these, however, you may find that the circuit is actually using the complemented form of the signal.

After a little practice, you will find this supplement extremely valuable. It will lead you page by page, signal by signal, through the logic for any instruction.

#### Acquisition Phase and DPA Pulses

The COM-TRAN 10 carries out each of its instructions in two phases: Acquisition and Execution. In the Acquisition Phase the instruction is brought from memory to the computer processor. Acquisition Phase is the same for all instructions. It was already stated, during discussion of the D-register, that the E flip-flop is used to tell the computer which Phase it is in. During Acquisition the E flip-flop has a low output at E, and the panel light will be off. At this time all distribution pulses are decoded as DPA pulses, DPA0, DPA1,...DPA15.

On the following pages you will find the complete breakdown of the Acquisition Phase Logic Signals. This is laid out by DPA pulse, and a total breakdown of each signal generated. Following this chart, trace the logic through the Circuits and Diagrams, while relating it to the Logic Timing. You should become thoroughly familiar with what happens in the computer during the Acquisition Phase, so that, given any particular DPA pulse, you could determine its effects and the signals it generates.

The logic timing that permits the COM-TRAN 10 to carry out instructions is controlled by the D-register, E flip-flop, S-register, and control logic. It consists of two distinct phases: (1) Acquisition, and (2) Execution. Each phase in turn consists of 16 Distributor phases (numbered decimally, 0 through 15) 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. Subcommands are not issued during all Acquisition and Execution distributor phases for any given instruction.

COMMAND GENERATION. In the COM-TRAN 10, commands are the decoded outputs of the S-register. A command usually represents one instruction, and will be labeled as such. These are indicated in Logic Diagrams. Thus, the command MPY is present (+5 volts) only when the OP code for multiply is in the S-register.

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 E flip-flop is in a reset state, subcommands are produced by the DPA Distributor pulses. 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 TQB causes data to be transferred from Q to B.

The Logic Timing Chart (KDA-3034, Sheet I) illustrates the sequence of subcommands for each instruction.

The Acquisition phase precedes the Execution phase of each instruction in normal operation. This operation allows us to retrieve the instruction code from its storage location in memory and transfer it to the Operation Code (S) Register. It then retrieves the Address from memory and transfers it to the Memory Address Register.

Once this has occurred, the computer will initiate the Execution phase and execute the instruction present in the Operation Code (S) Register.

Indexing also takes place in this phase if the instruction to be executed requires that information be transferred to or from memory. If the Index bit is a one, the contents of the Index Register will be added to the contents of the Memory Address Registers. The results will appear in the Memory Address Register.

ALL PAGE NUMBER REFERENCES CORRESPOND TO SHEET NUMBERS IN THE COM-TRAN 10 CIRCUITS AND DIAGRAMS, KDA-3034

DPA0	DPA1	DPA2	DPA3	DPA4	DPA5	DPA6	DPA7	DPA8
TPM TPB CLERR	ISB	TBS	SERI STOP IF NOT INEB	INCM	ISB	INCM	IF NOT RPT(AE + INST) TMP	TBM

DPA9	DPA10	DPA11	DPA12	DPA13	DPA14	DPA15
IF S2 = 1 AXM	SDP15					CLERR CKE

RDA26-457

DPA0 (TPM) Transfer Program to Memory Register

Transfer the contents of the Program Address Register to the Memory Address Register. See page 24. Transfer the contents of the Program Address Register to the Buffer Register. See pages 20, 19, and 3. (CLERR) Clear the Instruction Error flip-flop (ERI) unconditionally. See page 19.

- DPA1 (ISB) Initiate Storage to Buffer
- The contents of the memory location addressed in DPA0 are transferred to the Buffer Register. See pages 3 and 24.
- DPA2 (TBS) Transfer Buffer to S-Register
- Transfer the contents of the Buffer Register to the Operation Code (S) Register. The S-Register now contains the code of the instruction to be executed during the Execution phase. See page 13.
- DPA3 (SERI) Set Instruction Error  
(INEB) Instruction Error Bypass
- Verify that the HEX code present in the S-Register is a valid instruction code. See page 17. If it is not a valid code, set the Instruction Error flip-flop (ERI) with SERI. See page 19.
- If the Instruction Error Bypass switch, (INEB), is not pressed, a Stop signal will be generated by ERI. See page 20. The stop signal enables the Stop Clock (SPCK) signal. See page 18. This signal stops the computer clock. See page 7.
- DPA4 (INCM) Increment the Memory Address Register
- Add one to the contents of the Memory Address Register. INCM, see page 24, is used to enable a transfer of the G-bus to the Memory Address Register (TGM). See page 20. The G-bus data is generated by the Index Adder. During INCM time the Index Register outputs are inhibited to the Index Adder. Thus, the Index Register inputs to the adder are all zeros except in the case of bit 0 which is inverted. This enables the adder to add 1 to the contents of the Memory Address Register. See page 11.
- DPA5 (ISB) Initiate Storage to Buffer
- The contents of the memory location addressed in DPA4 are transferred to the Buffer Register. See page 24.
- DPA6 (INCM)
- See description of DPA4.
- DPA7 If not RPT (AE + INST), TMP
- Transfer the contents of the Memory Address Register to the Program Address Register. If the computer is in Repeat, and in the Acquisition/Execution or Instruction mode, the transfer (TMP) will not take place. See page 21. TMP clocks the contents of the Memory Address Register into the Program Address Register. See page 12. The Program Address Register now contains the address of the next instruction to be executed.
- DPA8 (TBM) Transfer Buffer to Memory Register
- Transfer the contents of the Buffer Register, which was retrieved from memory during DPA5, to the Memory Address Register. The data present in the Buffer Register is also available on the Y-bus. The signal DPA8 strobes this data into the Memory Address Register. See page 11.

DPA9 (AXM) Add the X-Register contents to the M-Register contents

The third least significant bit of the S-Register (S2) is designated as the Index bit. If this bit position is a one, indicating Index, we add the contents of the Index Register to the content of the Memory Address Register and leave the results in the Memory Address Register.

A logical AND of DPA9 and S2 generate the Add Index to Memory (AXM) signal. See page 23. This signal strobes the Index Register contents into the Index adder where it is added to the Memory Address Register contents. The adder outputs to the G-bus. See page 14.

AXM also generates a transfer G to M (TGM) signal. See page 20. This signal strobes the contents of the G-bus into the Memory Address Register. See page 11.

DPA10 (SDP15) Set Distributor Pulse 15

DPA10 enables the SDP15 signal. See page 21. SDP15 sets the Distributor Register to a count of 15. See page 5.

DPA11  
through  
DPA14

No Operations

DPA15 (CKE) Clock the E flip-flop  
(CLERR) Clear Errors

Two operations take place during this Distributor pulse. First, the Add, Divide, and Sign flip-flops are reset. See pages 19 and 23. This is done since these three flip-flops are used for detecting certain conditions during the Execution phase of an instruction, and therefore, must be reset prior to starting execution.

Secondly, the Execution flip-flop is clocked. This sets the computer in the Execution phase of the instruction. The signal Clock E flip-flop (CKE) is generated by either DP15 or DPA15. See page 18. This signal toggles the E flip-flop to the Execution phase. See page 5.

The Distributor also resets to zero at this time. At Clock Pulse 3 (CP3) of DPA15 the Distributor flip-flops are toggled. Since the data inputs (D) of the flip-flops are HIGH at this time they will reset to zero. See page 5.

The Execution phase is now enabled and the Distributor is set to zero. Thus, the computer is ready to begin executing the instruction it has just acquired.

#### REVIEW QUESTIONS 2-16

Match the operation with its associated timing pulse by placing the letter of the pulse in the blank preceding the operation.

1. \_\_\_ Reads the OP CODE from memory.
2. \_\_\_ Clears the ADD error and DIV error latches.
3. \_\_\_ Causes indexing if S2 is set.



4. \_\_\_ Loads address of instruction to be acquired into the M-register.
5. \_\_\_ Upcounts M-register by one, to read Operand or memory address.
6. \_\_\_ Transfers OP CODE from B-register to S-register.
7. \_\_\_ Stores address of next instruction to be acquired into the P-register.
8. \_\_\_ Reads Operand or memory address into the B-register.
9. \_\_\_ Checks for invalid instruction.
10. \_\_\_ Clears instruction error latch.
 

a. DPA0	e. DPA4	i. DPA8
b. DPA1	f. DPA5	j. DPA9
c. DPA2	g. DPA6	k. DPA10
d. DPA3	h. DPA7	l. DPA15

Circle the letter of the correct answer.

1. What is a function of the  $\overline{\text{DPA}}4$  pulse (sheet 18)?
  - a. Produces the  $\overline{\text{TGM}}$  signal
  - b. Produces the AXM signal
  - c. Given the address for the next OP CODE
  - d. Allow the distributor to be set to a count of 15
2. Which of the following is not a function of DPA15 (sheet 18)?
  - a. Clear divide error
  - b. Clear add error
  - c. Clear instruction error
  - d. Clear sign flip-flop
3. What conditions must be met at DPA3 to produce an instruction error ( $\overline{\text{SERI}}$ )? Reference sheet 17, 3E.
  - a. S2 set and OP CODE is between 20 (16) and FF (16)
  - b. S2 set and OP CODE is between 0 (16) and 1F (16)
  - c. S2 clear and OP CODE is between 20 (16) and 1F (16)
  - d. S2 clear and OP CODE is between 20 (16) and FF (16)
4. Which of the following would prevent DPA15 from clocking the E flip-flop to execution phase (sheet 5, B7)?
  - a. Distributor mode only
  - b. Instruction mode and RPT being used

- c. Program mode and RPT being used
- d. A/E mode and RPT being used

Execution Phase

The execution phase of an instruction is the phase during which the actual instruction steps or subcommands are executed. The Execution phase of all instructions begins at Distributor pulse zero (DP0) and ends at Distributor pulse 15 (DP15).

During DP15 certain operations take place regardless of the instruction being executed. These operations will be explained now and the individual instruction explanations will be referenced to this paragraph in regards to DP15.

Two operations occur during DP15. First, the appropriate Condition Code is set (STCC). In all instructions, other than multiply, arithmetic shifts, and Divide, the Condition Code will reflect the status of the Accumulator. In the Divide instruction the Condition Code reflects the Quotient Register status. In multiply, Shift Right Arithmetic and Shift Left Arithmetic, the Condition Code reflects the status of the AQ Register.

Condition Code Less Than Zero (CCLT) is set if the Accumulator MSB (A7) is a one or in the case of Divide, if the Quotient Register MSB (Q7) is a one. Since the MSB is the sign bit, having a one in the MSB indicates that the data in the register represents a negative value and is therefore less than zero. See page 19.

Condition Code Greater than Zero (CCGT) is set if A7 or Q7, in the case of Divide, is zero and AZ or QZ, is not set. AZ and QZ are signals which are enabled if the Accumulator or Quotient Registers, respectively, contain zeros in all bit positions. Thus, CCGT is sensing that the data in the Accumulator or Quotient Registers is not negative or zero. See page 19.

Condition Code Equal to Zero (CCEQ) is set if AZ or QZ in the case of Divide is a one. See page 19.

The second operation that takes place during DP15 is the clocking of the Execution flip-flop (CKE). When the Distributor reaches a count of 15 in either Acquisition or Execution phase, it is necessary to clock to the other phase; i.e., Acquisition to Execution or Execution to Acquisition. The CKE signal is generated by either DP15 or DPA15. See page 18. CKE is ANDed with Clock Pulse 3 (CP3) which is the last clock pulse of a Distributor pulse. This AND generates the signal which toggles the E flip-flop. See page 5.

LOAD C IMMEDIATE - LC1

This instruction loads the contents of the Buffer Register into the Countdown Register. It must be understood that the Buffer contents were retrieved from memory during the previous Acquisition phase at DPA5.

OP CODE - 01<sub>16</sub>

DP0	DP1	DP2	DP3 - DP14	DP15
TBC		SDP15		CKE
				STCC

RDA26-458

DPO (TBC) Transfer Buffer to Countdown

During this Distributor pulse the contents of the Buffer Register are transferred to the Countdown Register. Since the data contained in the Buffer Register is also present on the Y-bus, a transfer Y-bus to the Countdown Register (TYC) signal is generated. See page 22. This signal acts as the strobe which transfers the data from the Y-bus into the Countdown Register during Clock Pulse 2 time (CP2). See page 4.

DP1 No operation

DP2 (SDP15) Set Distributor Pulse 15

The LCl signal is ANDed with DP2 to generate a SDP15 pulse. See page 21. This signal sets the Distributor to a count of 15. See page 5.

DP3 through DP14 No operation

DP15 (CKE) Clock Execution flip-flop  
(STCC) Set Condition Code

LOAD A - LDA

This instruction loads the Accumulator with data which is stored in memory at the location addressed by the Memory Address Register.

OP CODE - 20<sub>16</sub>

DPO	DP1	DP2	DP3	DP4 - DP14	DP15
ISB	TBA		SDP15		STCC
					CKE

DPO (ISB) Initiate Storage to Buffer

The data stored at the memory location addressed by the Memory Address Register is transferred to the Buffer Register. ISB is generated by a Load Group (LDG) signal, which defines LDA and LAN instructions, and DPO. See pages 18 and 24. ISB initiates a memory cycle in which the data is taken from memory and placed on the Z-bus during CP1. See page 8. ISB also generates a transfer of the Z-bus to the Buffer Register (TZB). See page 19. TZB is ANDed with CP2 and this signal strobes the data into the Buffer Register. See page 3.

DP1 (TBA) Transfer Buffer to Accumulator

The data present in the Buffer Register is transferred to the Accumulator. The data present in the Buffer Register is available on the Y-bus. Since the computer is not doing an arithmetic operation, data on the Y-bus is transferred through the Arithmetic Logic Unit (ALU), without the use of a strobe pulse. Thus, the data in the Buffer is also available at the output of the ALU or the F-bus. See page 2.

The AND of LDG and DP1 generates two signals: Shift the Accumulator Right (SAR), and Shift the Accumulator Left (SAL). See page 23. Both these signals are enabled for the complete Distributor pulse.

When CP2 occurs during DP1, a parallel load of the F-bus into the Accumulator takes place if SAR and SAL are enabled. Thus, a transfer of data from the Buffer Register to the Accumulator has occurred. See page 1.

DP2 No operation

DP3 (SDP15) Set Distributor Pulse 15

LDG and Distributor pulse 3 (DP3) are ANDed to generate a SDP15 pulse. See page 21. This signal sets the Distributor to a count of 15. See page 5.

DP4 through DP14 No operation

DP15 CKE  
STCC

LOAD A IMMEDIATE - LAI

This instruction loads the contents of the Buffer Register into the Accumulator. It must be understood that the Buffer contents were retrieved from memory during the previous Acquisition phase at DPA5.

OP CODE - 02<sub>16</sub>

DP0	DP1	DP2	DP3	DP4 - DP14	DP15
	TBA		SDP15		STCC
					CKE

DPO No operation

DP1 (TBA) Transfer Buffer to Accumulator

The data present in the Buffer Register is transferred to the Accumulator. The data present in the Buffer Register is available on the Y-bus. Since the computer is not doing an arithmetic operation data on the Y-bus is transferred through the Arithmetic Logic Unit (ALU), without the use of a strobe pulse. Thus, the data in the Buffer is also available at the output of the ALU or the F-bus. See page 2.

The AND of LAI and DP1 generates two signals: Shift the Accumulator Right (SAR) and Shift the Accumulator Left (SAL). See page 23. Both these signals are enabled for the complete Distributor pulse.

When CP2 occurs during DP1 a parallel load of the F-bus into the Accumulator takes place if SAR and SAL are enabled. Thus, a transfer of data from the Buffer Register to the Accumulator has occurred. See page 1.

DP2 No operation

DP3 (SDP15) Set Distributor Pulse 15

LDG and Distributor pulse 3 (DP3) are ANDED to generate a SDP15 pulse. See page 21. This signal sets the Distributor to a count of 15. See page 5.

DP4  
through  
DP14 No operation

DP15 CKE, STCC

LOAD X IMMEDIATE - LXI

This instruction loads the contents of the Buffer Register into the Index Register. It must be understood that the Buffer contents were retrieved from memory during the previous Acquisition phase at DPA5.

OP CODE - 12<sub>16</sub>

DPO	DP1	DP2	DP3 - DP14	DP15
TBX		SDP15		STCC
				CKE

DPO (TXB) Transfer Buffer to Index

The data in the Buffer Register is also available on the Y-bus. The logical AND of DPO and LXI generate the signal Load Index (LX). See page 22. This signal strobes the data on the Y-bus into the Index Register at CP2. See page 14. Thus, the data present in the Buffer Register has been transferred to the Index Register.

DP1 No operation

DP2 (SDP15) Set Distributor Pulse 15

LXI and DP2 are ANDED to generate SDP15. See page 21. This signal sets the Distributor to a count of 15. See page 5.

DP3  
through  
DP14 No operation

DP15 STCC  
CKE

LOAD CONSECUTIVE - LCC

This instruction acquires data from a given memory location and loads it into the following memory location.

OP CODE - 30<sub>16</sub>

DPO	DP1	DP2	DP3	DP4 - DP14	DP15
ISB	INCM	IBS	SDP15		STCC CKE

DP0 (ISB) Initiate Storage to Buffer

The data stored at the memory location addressed by the Memory Address Register is transferred to the Buffer Register. LCC and DP0 are ANDed to generate an ISB signal. See page 24. ISB initiates a memory cycle in which the data is taken from memory and transferred to the Z-bus during CP2. See page 8. ISB also generates a signal to transfer the Z-bus data to the Buffer Register (TZB). See page 19. TZB along with CP2 strobes the data into the Buffer Register. See page 3.

DP1 (INCM) Increment Memory

During INCM, the Index Register contents are inhibited from the Index adder. Thus, the Index Register inputs to the Adder are zeros, except in the case of bit 0, which is inverted. Thus, the Adder will add one to the contents of the Memory Address Register and the results of the add will be on the G-bus. See page 14.

INCM is also used to enable a transfer of the G-bus to the Memory Address Register (TGM). See page 20. The Memory Address Register will now contain a count one greater than at the beginning of the instruction. See page 11.

DP2 (IBS) Initiate Buffer to Storage

Store the data present in the Buffer at the Memory location addressed by the Memory Address Register. This signal is generated by LCC and DP2. See page 24. IBS generates a memory cycle in which the data present in the Buffer Register is stored in memory via the Y-bus. See page 8.

DP3 (SDP15) Set the Distributor to a count of 15

LCC and DP3 combine to generate SDP15. See page 21. This signal sets the Distributor to a count of 15. See page 5.

DP4  
through  
DP14

No operation

DP15

STCC  
CKE

LOAD A NEGATIVE - LAN

This instruction loads the Accumulator with the two's complement of the data acquired from memory.

OP CODE - 38<sub>16</sub>

DPO	DP1	DP2	DP3	DP4 - DP14	DP15
ISB	TBA	2's COMP A	SDP15		STCC  CKE

DPO (ISB) Initiate Storage to Buffer

The data stored at the memory location addressed by the Memory Address Register is transferred to the Buffer Register. ISB is generated by a Load Group (LDG) signal, which defines LDA and LAN instructions, and DPO. See pages 18 and 24. ISB initiates a memory cycle in which the data is taken from memory and placed on the Z-bus during CP2. See page 8. ISB also generates a transfer of the Z-bus to the Buffer Register (TZB). See page 19. TZB is ANDed with CP2 and this signal strobes the data into the Buffer Register. See page 3.

DP1 (TBA) Transfer Buffer to Accumulator

The data present in the Buffer Register is available on the Y-bus. Since the computer is not doing an arithmetic operation, data on the Y-bus is transferred through the Arithmetic Logic Unit (ALU), without the use of a strobe pulse. Thus, the data in the Buffer is also available at the output of the ALU or the F-bus. See page 2.

The AND of LDG and DP1 generates two signals: Shift the Accumulator Right (SAR) and Shift the Accumulator Left (SAL). See page 23. Both these signals are enabled for the complete Distributor pulse.

When CP2 occurs during DP1 a parallel load of the F-bus into the Accumulator takes place if SAR and SAL are enabled. Thus, a transfer of data from the Buffer Register to the Accumulator has occurred. See page 1.

DP2 (COMP A) Complement the Accumulator

During this Distributor pulse a two's complement is executed on the Accumulator contents. LAN is ANDed with DP2 to generate COMP A. See page 20. COMP A generates TAZ. See page 23. TAZ is the strobe which gates the Accumulator contents to the Z-bus. See page 1. COMP A also generates COMP. See page 20. A TWO's signal is normally HIGH due to the use of mismatch logic. See page 22.

Z-bus data is available at the Two Complementer. Since the TWO's signal is HIGH, the output of the Two's Complementer contains the two's complement of the Z-bus data. The COMP signal steers the Selector so that the output of the Complementer is strobed onto the Y-bus. See page 3.

COMP A also enables SAL and SAR. See page 23. Since no arithmetic operations are taking place in the Arithmetic Logic Unit (ALU), the Y-bus data is transferred to the F-bus without the need of a strobe. See page 2. At CP2 the F-bus is transferred in parallel form into the Accumulator since SAL and SAR are both HIGH. See page 1.

During the Distributor pulse data has been taken from the Accumulator, a Two's complement has been executed, and the result has been replaced in the Accumulator.

DP3 (SDP15) Set Distributor Pulse 15

LDG and Distributor pulse 3 (DP3) are ANDed to generate a SDP15 pulse. See page 21. This signal sets the Distributor to a count of 15. See page 5.

DP4 through DP14 No operation

DP15 STCC  
CKE

LOAD Q - LDQ

This instruction acquires data from the memory location addressed by the Memory Address Register and transfers it to the Quotient Register.

OP CODE - 40<sub>16</sub>

DP0	DP1	DP2	DP3	DP4 - DP14	DP15
ISB		TBQ	SDP15		STCC CKE

DP0 (ISB) Initiate Storage to Buffer

Data is retrieved from memory and placed on the Y bus. LDQ and DP0 generate the ISB signal. See page 24. ISB initiates a memory cycle in which the data in the location addressed by the Memory Address Register is transferred to the Z-bus during DP2. See page 8.

ISB also generates a signal which allows the Z-bus data to be transferred to the Buffer Register (TZB). See page 19.

The Z-bus is available at the input to the Buffer Register. TZB and CP2 strobe the Z-bus data into the Buffer Register and to the Y-bus. See page 3.

DP1 No operation

DP2 (TBQ) Transfer Buffer to Quotient

The data on the Y-bus is transferred to the Quotient Register. LDQ and DP2 are ANDed to generate SQR and SQL. See page 22. At CP2 of DP2, the



Y-bus data is strobed into the Quotient Register in parallel since SQL and SQR are both HIGH. See page 6.

DP3 (SDP15) Set the Distributor to count of 15

LDQ and DP3 generate a SDP15 pulse. See page 21. This signal sets the Distributor to a count of 15. See page 5.

DP4 through DP14 No operation

DP15 STCC  
CKE

STORE A - STA

This instruction stores the Accumulator contents in the memory location addressed by the Memory Address Register.

OP CODE - 48<sub>16</sub>

DP0	DP1	DP2	DP3 - DP14	DP15
TAB	IBS	SDP15		STCC CKE

DP0 (TAB) Transfer Accumulator to Buffer

STA and DP0 combine to generate a transfer Accumulator to Buffer signal (TAB). A signal which transfers the Accumulator contents to the Z-bus (TAZ) is also generated at this time. See page 23. TAZ gates the Accumulator contents to the Z-bus. See page 1. TAB enables a signal which transfers to Z-bus to the Buffer Register (TZB). See page 19. At CP2, TZB strobes the Z-bus data into the Buffer Register and on to the Y-bus. See page 3.

DP1 (IBS) Initiate Buffer to Storage

The data in the Buffer Register will now be stored in memory. Store Group (STG) is generated by a logical OR of STA, STX, or STQ instructions. See page 18. STG is ANDed with DP1 to generate a signal which allows the data to be stored in memory (IBS). See page 24. IBS initiates a memory cycle in which the data on the Y-bus is stored in memory at the location addressed by the Memory Address Register. See page 8.

DP2 (SDP15) Set Distributor Pulse 15

STG is ANDed with DP2 to generate SDP15. See page 21. This signal sets the Distributor to a count of 15. See page 5.

DP3 through DP14 No operation

DP15 STCC, CKE

STORE X - STX

This instruction stores the Index Register data in the addressed memory location

OP CODE - 50<sub>16</sub>

DPO	DP1	DP2	DP3 - DP14	DP15
TXB	IBS	SDP15		STCC
				CKE

DPO (TXB) Transfer Index to Buffer

STX and DPO are ANDed to generate TXB. See page 23. TXB gates the Index Register contents to the Z-bus. See page 14. TXB also generates TZB which gates the Z-bus contents to the Buffer Register and Y-bus. See pages 3 and 19.

DP1 (IBS) Initiate Buffer to Storage

The data in the Buffer Register will now be stored in memory. Store Group (STG) is generated by a logical OR of STA, STX, or STQ instructions. See page 18. STG is ANDed with DP1 to generate a signal which allows the data to be stored in memory (IBS). IBS initiates a memory cycle in which the data on the Y-bus is stored in memory at the location addressed by the Memory Address Register. See page 8.

DP2 (SDP15) Set Distributor Pulse 15

STG is ANDed with DP2 to generate SDP15. See page 21. This signal sets the Distributor to a count of 15. See page 5.

DP3 through DP14 No operation

DP15 STCC, CKE

STORE Q - STQ

This instruction stores the contents of the Quotient Register at the memory location addressed by the Memory Address Register.

OP CODE - 58<sub>16</sub>

DPO	DP1	DP2	DP3 - DP14	DP15
TQB	IBS	SDP15		STCC
				CKE

DP0 (TQB) Transfer Quotient to Buffer

STQ and DP0 are ANDed to generate TQB and TQZ. See page 22. TQZ transfers the Quotient Register contents to the Z-bus. See page 6. TQB generates TZB which transfers the Z-bus to the Buffer Register and Y-bus. See pages 3 and 19.

DP1 (IBS) Initiate Buffer to Storage

The data in the Buffer Register will now be stored in memory. Store Group (STG) is generated by a logical OR of STA, STX, or STQ instructions. See page 18. STG is ANDed with DP1 to generate a signal which allows the data to be stored in memory (IBS). IBS initiates a memory cycle in which the data on the Y-bus is stored in memory at the location addressed by the Memory Address Register. See page 8.

DP2 (SDP15) Set Distributor Pulse 15

STG is ANDed with DP2 to generate SDP15. See page 21. This signal sets the Distributor to a count of 15. See page 5.

DP3 through DP14 No operation

DP15 STCC, CKE

ADD - ADD

This instruction adds the contents of the Accumulator to the contents of the addressed memory location and places the sum in the Accumulator.

OP CODE - 60<sub>16</sub>

DP0	DP1	DP2	DP3	DP4 - DP14	DP15
ISB	INA	IF ADD OVFL	SDP15		STCC
CLCR	IF OVFL SAOV	AND NOT BYPASS STOP			CKE
	IF EC SCARY				

DP0 (CLCR) Clear Carry flip-flop  
(ISB) Initiate Storage to Buffer

The ADD or SUB instructions enable the Arithmetic group signal (ARG). See page 18. ARG and DP0 clear the CARY flip-flop. See page 19.

ARG and DP0 also generate an initiate storage to buffer (ISB). See page 24. ISB initiates a memory cycle in which the data in the location addressed by the Memory Address Register is transferred to the Z-bus during CP2. See page 8.

ISB also generates a signal which allows the Z-bus data to be transferred to the Buffer Register (TZB). See page 19.

The Z-bus is available at the input to the Buffer Register. TZB and CP2 strobe the Z-bus data into the Buffer Register and to the Y-bus.

DP1 (INA) Initiate Add

If there is overflow, set the Add Overflow flip-flop (SAOV). If a carry exists (EC), set the Carry flip-flop (SCARY).

An initiate add signal (INA) is generated by ADD and DP1. See page 22. INA is applied to the steering gates of the Arithmetic Logic Units (ALU). This initiates an add of the Y-bus and the Accumulator contents. The result is available on the F-bus. See page 2. INA also generates SAR and SAL. See page 23. These signals are both HIGH at this time. Thus, CP2 of DP1 clocks the F-bus data into the Accumulator. Overflow is also sensed at this time. If overflow exists, SAOV will be enabled. See page 1. SAOV will set the Add Error flip-flop (ERA). See page 19.

If the ALU generates an End Carry, the CARY flip-flop will be set. See page 19.

DP2 The computer clock will be halted by a STOP pulse if ERA, ARG, and ADEB are ANDed together. Mismatch logic is used in this case generating STOP. See page 20.

DP3 (SDP15) Set Distributor Pulse 15

ARG and DP3 generate a SDP15 pulse. See page 21. This pulse sets the Distributor to a count of 15. See page 5.

DP4 through DP14 No operation

DP15 STCC, CKE

SUBTRACT - SUB

This instruction subtracts the contents of the addressed memory location from the contents of the Accumulator and places the remainder in the Accumulator.

OP CODE - 68<sub>16</sub>

DP0	DP1	DP2	DP3	DP4 - DP14	DP15
CLCR	INS	IF ADD OVFL	SDP15		STCC
ISB	IF OVFL SAOV	AND NOT BYPASS			CKE
	IF EC SCARY	STOP			

DPO (CLCR) Clear the Carry flip-flop  
(ISB) Initiate Storage to Buffer

The ADD or SUB instructions enable the Arithmetic group signal (ARG). See page 18. ARG and DPO clear the CARY flip-flop. See page 19.

ARG and DPO also generates an initiate storage to buffer (ISB). See page 24. ISB initiates a memory cycle in which the data in the location addressed by the Memory Address Register is transferred to the Z-bus during CP2. See page 8.

ISB also generates a signal which allows the Z-bus data to be transferred to the Buffer Register (TZB). See page 19.

The Z-bus is available at the input to the Buffer Register. TZB and CP2 strobe the Z-bus data into the Buffer Register and to the Y-bus.

DP1 (INS) Initiate Subtract

If overflow exists, set Add Overflow (SAOV). If a carry (borrow) exists, (EC), set the Carry flip-flop (SCARY).

An initiate subtract signal, (INS) is generated by SUB and DP1. See page 22. INS is applied to the steering gates of the Arithmetic Logic Units (ALU). This initiates a subtraction of the Y-bus from the Accumulator contents. The result is available on the F-bus. See page 2. INS also generates SAR and SAL. See page 23. These signals are both HIGH at this time. Thus, CP2 of DP1 clocks the F-bus data into the Accumulator.

Overflow is also sensed at this time. If overflow exists, SAOV will be enabled. See page 1. SAOV will set the Add Error flip-flop (ERA). See page 19.

If the ALU generates an End Carry, the CARY flip-flop will be set. See page 19.

DP2 The computer clock will be halted by a STOP pulse if ERA, ARG, and ADEB are ANDed together. Mismatch logic is used in this case to generate STOP. See page 20.

DP3 (SDP(15) Set Distributor Pulse 15

ARG and DP3 generate a SDP15 pulse. See page 21. This pulse sets the Distributor to a count of 15. See page 5.

DP4 through DP14 No operation

DP15 STCC

CKE

MULTIPLY - MPY

This instruction multiplies the contents of the Accumulator by the contents of the addressed memory location. The product appears as a double length number in the Accumulator and Quotient Register.

OP CODE - 70<sub>16</sub>

DP0	DP1	DP2	DP3	DP4	DP5	DP6
ISB	TBQ					
SC8	IF B7=0 SDP4	2's COMP A	2's COMP Q	TAB	CLA	If Q0=0 SDP9

DP7	DP8	DP9	DP10	DP11	DP12 - DP14	DP15
INA	SSNF=B7	SAQR SNF → A7 AO → Q7	DEC	IF C≠0 SDP6		STCC CKE

DP0 (ISB) Initiate Storage to Buffer  
(SC8) Set the C-Register to a count of 8

MPY and DP0 generate the ISB signal. See page 24. ISB initiates a memory cycle in which the data in the location addressed by the Memory Address Register is transferred to the Z-bus during CP2. See page 8.

ISB also generates a signal which allows the Z-bus data to be transferred to the Buffer Register (TZB). See page 19.

The Z-bus is available at the input to the Buffer Register. TZB and CP2 strobe the Z-bus data into the Buffer Register and to the Y-bus.

MPY and DP0 also generate a signal which sets the Countdown Register to a count of 8 (SC8). See page 22. SC8 causes the Countdown Register to be set to all ones during CP1. At the end of CP1, SC8 clears all the flip-flops except C3. Thus, the Countdown Register contains a HEX count of 8. See page 4.

DP1 (TBQ) Transfer the Buffer Contents to the Q-Register  
(SDP4) Set Distributor Pulse 4 (If B7 = 0)

The data on the Y-bus is transferred to the Quotient Register. MPY and DP1 are ANDed to generate SQR and SQL. See page 22. At CP2 of DP1 the Y-bus data is strobed into the Quotient Register in parallel since SQL and SQR are both HIGH. See page 6.

If B7 = 0, denoting a positive value in the Buffer Register, the Distributor is set to a count of 4, skipping DP2 and DP3. SDP4 is generated by the AND of MPY,  $\bar{B}7$ , and DP1. See page 21. SDP4 sets the Distributor to a count of 4. See page 5.

DP2 (COMP A) Complement the Accumulator

During this Distributor pulse a two's complement is executed on the Accumulator contents. MPY is ANDed with DP2 to generate COMP A. See page 20. COMP A generates TAZ. See page 23. TAZ is the strobe which gates the Accumulator contents to the Z-bus. See page 1. COMP A also generates COMP. See page 20. A TWO's signal is normally HIGH due to the mismatch logic. See page 22.

Z-bus data is available at the Two Complementor. Since the TWO's signal is HIGH, the output of the Two's Complementor contains the two's complement of the Z-bus data. The COMP signal steers the Selector so that the output of the Complementor is strobed onto the Y-bus. See page 3.

COMP A also enables SAL and SAR. See page 23. Therefore, since no arithmetic operations are taking place on the Arithmetic Logic Unit (ALU), the Y-bus data is transferred to the F-bus without the need of a strobe. See page 2. At CP2 the F-bus is transferred in parallel form into the Accumulator since SAL and SAR are both HIGH.

During this Distributor pulse data has been taken from the Accumulator, a two's complement has been executed, and the result has been replaced in the Accumulator.

DP3 (COMP Q) Complement the Quotient

During this Distributor pulse a two's complement will be executed on the contents of the Quotient Register.

MPY and DP3 generate COMP Q. See page 20. COMP Q enables TQZ which transfers the Quotient Register contents to the Z-bus. See pages 6 and 22.

COMP Q also generates COMP. See page 20. This signal selects the output of the two's complementor and clocks it onto the Y-bus. See page 3.

COMP Q also forces SQL and SQR HIGH. See page 22. At CP2 of the Distributor pulse, with the above signals HIGH, the Y-bus data will be loaded in parallel fashion into the Quotient Register.

DP4 (TAB) Transfer Accumulator to Buffer

MPY and DP4 combine to generate a transfer Accumulator to Buffer signal (TAB). A signal which transfers the Accumulator contents to the Z-bus (TAZ) is also generated at this time. See page 23. TAZ gates the Accumulator contents to the Z-bus. See page 1. TAB enables a signal which transfers the Z-bus to the Buffer Register (TZB). See page 19. At CP2, TZB strobes the Z-bus data into the Buffer Register and onto the Y-bus. See page 3.

DP5 (CLA) Clear the Accumulator

DP5 and MPY combine to generate CLA. See page 23. This pulse clears the Accumulator. See page 1.

DP6 (SDP9) Set Distributor Pulse 9 (If Q0 = 0)

If the LSB of the Quotient Register is zero, the Distributor will be set to a count of 9.  $\overline{Q_0}$ , DP6 and MPY are gated together to enable an SDP9. See page 21. This pulse sets the Distributor to a count of 9. See page 5.

DP7 (INA) Initiate Add

INA is generated by MPY and DP7. See page 22. INA is applied to the steering gates of the Arithmetic Logic Units (ALU). This initiates an add of the Y-bus and the Accumulator contents. The result is available on the F-bus. See page 2. INA also generates SAR and SAL. See page 23. These signals are both HIGH at this time. CP2 of DP1 clocks the F-bus data into the Accumulator.

DP8 (SSNF = B7) Set the Sign flip-flop equal to B7

MPY, DP8 and B7 set the Sign flip-flop (SNF). See page 23.

DP9 (SAQR) Shift the Accumulator and Quotient Right

The contents of the Accumulator and the Quotient Register will be shifted to the right one place. The contents of A0 will be shifted into Q7.

MPY and DP9 generate shift AQ right (SAQR). See page 22. SAQR enables SAR and SQR. See pages 22 and 23.

SAR shifts the Accumulator right one bit at CP2. The sign flip-flop (SNF) contains the sign of the result of the addition which took place in DP7. See page 23. SNF and MPY generate DSRA. See page 23. If this signal is HIGH, indicating that the addition resulted in a negative sum, a one will be shifted into A7. See page 1.

SQR shifts the Quotient Register right one bit at CP2. This is an open-ended shift. The contents of A0 will be shifted into Q7 at this time. See page 6.

DP10 (DEC) Decrement the C-Register

The Countdown Register will be decreased by one count during this Distributor pulse. MPY and DP10 generate a decrement Countdown (DEC) signal. See page 24. At CP3, DEC decreases the Countdown Register by one. See page 4.

DP11 (If C ≠ 0 SDP6) If the C-Register does not contain zero, set the Distributor to a count of 6

If the Countdown Register is completely reset, a CZ signal is generated. See page 4.  $\overline{CZ}$ , MPY, and DP11 are gated together to enable an SDP6 signal. See page 19. This signal will set the Distributor to a count of 6. See page 5.

DP12  
through  
DP14

No operation

DP15 STCC, CKE

DIVIDE - DIV

This instruction divides the double length number in the Accumulator and Quotient Register by the contents of the addressed memory location.

The quotient will be contained in the Quotient Register and the remainder will be contained in the Accumulator.



DP0	DP1	DP2	DP3	DP4	DP5	DP6
ISB	IF B=0 SERD & STOP	IF A7=0 SDP6	IF Q=0 2's COMP A	2's COMP Q	IF A7=1 SERD STOP	IF C=0 SDP11
SC8	IF A7≠B7 SSNF		IF Q≠0 1's COMP A			

DP7	DP8	DP9	DP10	DP11	DP12	DP13	DP14	DP15
DEC	IF B7 = 1 INA		IF B7 = 0 INS		IF B7 SNF=1, 2s COMP A	IF SNF=1 2's COMP Q		STCC  CKE
	SAL	SQL $\overline{F7} \rightarrow Q0$	IF F7=0 TFA SDP6	IF Q7 (SNF + Q6-Q0), SET ERD STOP				

DP0 (ISB) Initiate Storage to Buffer  
(SC8) Set the C-Register to a count of 8

DIV and DP0 generate the ISB signal. See page 24. ISB initiates a memory cycle in which the data in the location addressed by the Memory Address Register is transferred to the Z-bus during CP2. See page 8.

ISB also generates a signal which allows the Z-bus data to be transferred to the Buffer Register (TZB). See page 19.

The Z-bus is available at the input to the Buffer Register. TZB and CP2 strobe the Z-bus data into the Buffer Register and to the Y-bus. See page 3.

DIV and DP0 also generate a signal which sets the Countdown Register to a count of 8 (SC8). See page 22. SC8 causes the Countdown Register to be set to all ones during CP1. At the end of CP1, SC8 clears all the flip-flops except C3. Thus, the Countdown Register contains a HEX count of 8. See page 4.

DP1 A check of the Buffer Register contents is made. If it is zero, a Divide error is present and the computer stops. If not, a check of the signs of the divisor and dividend is made. If they differ, the sign flip-flop (SNF) is set.

If the Buffer Register contains zero, BZ is generated. See page 3. BZ, DP1, and DIV are ANDed to enable SERD. See page 24. SERD sets the Divide Error (ERD). See page 19. ERD will generate a STOP pulse which will stop the computer clock. (If not DVEB.) See pages 7 and 20.

The sign flip-flop (SNF) is set if A7 does not equal B7. See page 23.

- DP2 (If  $A7 = 0$  SDP6) If the Dividend is positive, set Distributor Pulse 6  $\overline{A7}$ , DIV, and DP2 generate a SDP6 pulse. See page 19. This sets the Distributor to a count of 6. See page 5.
- DP3 If the Quotient Register contains zero a two's complement will be executed on the Accumulator contents. If the Quotient Register does not contain zero a one's complement will be executed on the Accumulator contents.
- DP3 and DIV generate COMP A. See page 20. COMP A generates TAZ. See page 23. TAZ is the strobe which gates the Accumulator contents to the Z-bus. See page 1. COMP A also generates COMP. See page 20. TWO's signal is HIGH if the Q-Register equals 0. See page 22.
- Z-bus data is available at the Two Complementor. Since the TWO's signal is HIGH, the output of the Two's Complementer contains the two's complement of the Z-bus data. The COMP signal steers the Selector so that the output of the Complementor is strobed onto the Y-bus. See page 3.
- COMP A also enables SAL and SAR. See page 23. Therefore, since no arithmetic operations are taking place in the Arithmetic Logic Unit (ALU), the Y-bus data is transferred to the F-bus without the need of a strobe. See page 2. At CP2 the F-bus is transferred in parallel form into the Accumulator since SAL and SAR are both HIGH.
- If the Quotient Register does not contain zero, TWO's is not generated. See page 22. This signal forces the complementor to execute a one's complement rather than a two's complement. See page 3.
- DP4 During the Distributor pulse a two's complement will be executed on the contents of the Quotient Register.
- DIV and DP4 generate COMP Q. See page 20. COMP Q enables TQZ which transfers the Quotient Register contents to the Z-bus. See pages 6 and 22.
- COMP Q also generates COMP. See page 20. This signal selects the output of the two's complementor and clocks it onto the Y-bus. See page 3.
- COMP Q also forces SQL and SQR HIGH. See page 22. At CP2 of the Distributor pulse, with the above signals HIGH, the Y-bus data will be loaded in parallel fashion into the Quotient Register.
- DP5 If the Dividend is negative, set Divide error and Stop. (If not DVEB.)
- A7, DIV, and DP5 will generate a Divide Error signal (SERD). See page 24. This pulse will set the Divide Error (ERD). See page 19. ERD will generate a STOP pulse which will stop the computer clock. See pages 7 and 20.
- DP6 If the C-Register is zero, set the Distributor to a count of 11.
- If the Countdown Register contains zero, a CZ pulse is enabled. See page 4. CZ, DIV, and DP6 enable SDP11. See page 21. This signal sets the Distributor to a count of 11. See page 5.

- DP7 (DEC) Decrease the count contained in the C-Register
- DIV and DP7 generate a decrement Countdown signal (DEC). See page 24. At CP3, DEC counts the Countdown Register down by one. See page 4.
- DP8 through DP11 Are decoded as Divide Group, (DVG). See page 18. During this time either an addition or subtraction of the accumulator and buffer takes place depending on the state of the Buffer MSB.
- DP8 (SAL) Shift Accumulator Left
- Shift Accumulator Left is generated by DP8 and DIV. See page 23. This causes the contents of the Accumulator to be shifted left one bit position. See page 1. The contents of Q7 will be shifted into A0 due to the mismatch logic used in generating DSLA. See page 24.
- DP9 (SQL) Shift Quotient Left
- Shift the Quotient Register left (SQL) is generated by DIV and DP9. See page 22. This causes the contents of the Quotient Register to be shifted one bit position to the left. If the addition or subtraction results caused the MSB of the ALU output bus (F7) to be a one, a zero will be shifted into Q0. See pages 6 and 20.
- DP10 (TFA) Transfer F-bus to Accumulator  
(SDP6) Set Distributor pulse 6
- If the results of the addition or subtraction result in F7 being zero, the output (F-bus) of the ALU will be transferred to the Accumulator. Otherwise it will be ignored.
- $\overline{F7}$ , DIV, and DP10 generate SAR and SAL. See page 23. At CP2 of DP10 the F-bus will be transferred to the Accumulator. See page 1.
- DP10 and DIV enable SDP6. See page 19. This signal forces the Distributor to a count of 6.
- DP11 If the Q-Register contains a negative value (Q7) and if either the Sign flip-flop (SNF) is not set or the Q-Register contains a positive value (Q6 through Q0 $\neq$ 0), a division error is detected.
- Q7,  $\overline{Q6Q0}$ , DP11,  $\overline{SNF}$ , and DIV are ANDed to enable SERD. See page 24. This signal generates Divide Error (ERD). See page 19. ERD generates a STOP signal which stops the computer clock. See pages 7 and 20.
- DP12 If the divisor is negative (B7), the remainder is two's complemented (COMP A).
- The Buffer Register MSB (B7), DP12,  $\overline{SNF}$  and DIV are gated together to generate COMP A. See page 20.
- COMP A generates TAZ. See page 23. TAZ is the strobe which gates the Accumulator contents to the Z-bus. See page 1. COMP A also generates COMP. See page 20. TWO's signal is normally HIGH due to the use of mismatch logic. See page 22.

Z-bus data is available at the Two Complementor. Since the TWO's signal is HIGH, the output of the Two's Complementor contains the two's complement of the Z-bus data. The COMP signal steers the Selector so that the output of the Complementor is strobed onto the Y bus. See page 3.

COMP A also enables SAL and SAR. See page 23. Therefore, since no arithmetic operations are taking place on the Arithmetic Logic Unit (ALU), the Y-bus data is transferred to the F-bus without the need of a strobe. See page 2. At CP2 the F-bus is transferred in parallel form into the Accumulator since SAL and SAR are both HIGH.

During the Distributor pulse data has been taken from the Accumulator, a Two's Complement has been executed, and the result has been replaced in the Accumulator.

DP13 If the Sign flip-flop (SNF) is set, the Quotient is Two's Complemented (COMP Q).

The Sign flip-flop output (SNF) together with DP13, and DIV generate COMP Q. See page 20. COMP Q enables TQZ which transfers the Quotient Register contents to the Z-bus. See pages 6 and 22.

COMP Q also generates COMP. See page 20. This signal selects the output of the two's complementor and clocks it onto the Y-bus. See page 3.

COMP Q also forces SQL and SQR HIGH. See page 22. At CP2 of the Distributor pulse, with the above signals HIGH, the Y-bus data will be loaded in parallel fashion into the Quotient Register. See page 6.

DP14 No operation

DP15 STCC, CKE

REPLACE ADD ONE - RAO

This instruction retrieves data from the addressed memory location, adds one to the data, and replaces it into the same memory location.

OP CODE - 80<sub>16</sub>

DP0	DP1	DP2	DP3	DP4	DP5	DP6
CLCR		INCA				
ISB	TBA	IF EC SCARY		TAB	IBS	

DP7	DP8-DP14	DP15
SDP15		STCC CKE

DPO (CLCR) Clear the Carry flip-flop  
(ISB) Initiate Storage to Buffer

RAO or RSO enable the replace group signal (RPG). See page 18. RPG and DPO clear the carry flip-flop. See page 19. RPG and DPO also generate an initiate storage to Buffer signal (ISB). See page 24. ISB initiates a memory cycle in which the data is taken from memory and transferred to the Z-bus during CP2. See page 8. ISB also generates a signal to transfer the Z-bus data to the Buffer Register (TZB). See page 19. TZB along with CP2 strobes the data into the Buffer Register. See page 3.

DP1 (TBA) Transfer Buffer to Accumulator

The data present in the Buffer is available on the Y-bus. Since the computer is not doing an arithmetic operation, data on the Y-bus is transferred through the Arithmetic Logic Unit (ALU) without the use of a strobe pulse. Thus, the data in the Buffer is also available at the output of the ALU or the F-bus. See page 2.

The AND of RPG and DP1 generates two signals: Shift the Accumulator Right (SAR) and Shift the Accumulator Left (SAL). See page 23. Both these signals are enabled for the complete Distributor pulse.

When CP2 occurs during DP1 a parallel load of the F-bus into the Accumulator takes place if SAR and SAL are enabled. Thus, a transfer of data from the Buffer Register to the Accumulator has occurred. See page 1.

DP2 (INCA) Increment Accumulator  
(EC) End Carry  
(SCARY) Set Carry (If EC = 1)

During this Distributor pulse, the contents of the Accumulator will be increased by one, and a carry condition will be detected if it exists.

RAO and DP2 are ANDed to enable INCA. See page 22. INCA is applied to the steering logic of the Arithmetic Logic Unit (ALU) and causes it to add one to the Accumulator output. The result is available on the F-bus. See page 2. INCA also causes SAL and SAR to be in HIGH state. At CP2 the F-bus contents will be transferred into the Accumulator. See pages 1 and 23.

If an End Carry (EC) was generated by the ALU, the Carry flip-flop will be set. See page 19.

DP3 No operation

DP4 (TAB) Transfer Accumulator to Buffer

RPG and DP4 combine to generate a transfer Accumulator to Buffer signal (TAB). A signal which transfers the Accumulator contents to the Z-bus (TAZ) is also generated at this time. See page 23. TAZ gates the Accumulator contents to the Z-bus. See page 1. TAB enables a signal which transfers the Z-bus to the Buffer Register (TZB). See page 19. At CP2, TZB strobes the Z-bus data into the Buffer Register and onto the Y-bus. See page 3.

DP5 (IBS) Initiate Buffer to Storage

The data in the Buffer Register will now be stored in memory. Replace Group (RPG) and DP5 generate IBS, which initiates a memory cycle which

stores the information on the Y-bus at the addressed memory location. See pages 8 and 24.

DP6 No operation

DP7 (SDP15) Set Distributor Pulse 15

RPG and DP7 enable SDP15. See page 21. This signal sets the Distributor to a count of 15. See page 5.

DP8  
through  
DP14

No operation

DP15 STCC, CKE

REPLACE SUBTRACT ONE - RSO

This instruction retrieves data from the addressed memory location, subtracts one from it and replaces it into the same memory location.

OP CODE - 88<sub>16</sub>

DPO	DP1	DP2	DP3	DP4	DP5	DP6
CLCR	TBA	DECA		TAB	IBS	
ISB		IF EC SCARY				

DP7	DP8 - DP14	DP15
SDP15		STCC
		CKE

DP0 (CLCR) Clear the Carry flip-flop  
(ISB) Initiate Storage to Buffer

RAO or RSO enable the replace group signal (RPG). See page 18. RPG and DPO clear the carry flip-flop. See page 19. RPG and DPO also generate an initiate storage to Buffer signal (ISB). See page 24. ISB initiates a memory cycle in which the data is taken from memory and transferred to the Z-bus during CP2. See page 8. ISB also generates a signal to transfer the Z bus data to the Buffer Register (TZB). See page 19. TZB along with CP2 strobes the data into the Buffer Register. See page 3.

DP1 (TBA) Transfer Buffer to Accumulator

The data present in the Buffer Register is transferred to the Accumulator. The data present in the Buffer Register is available on the Y-bus. Since the computer is not doing an arithmetic operation, data on the Y-bus is transferred through the Arithmetic Logic Unit (ALU) without the

use of a strobe pulse. Thus, the data in the Buffer is also available at the output of the ALU or the F-bus. See page 2.

The AND of RPG and DP1 generates two signals: Shift the Accumulator Right (SAR) and Shift the Accumulator Left (SAL). See page 23. Both these signals are enabled for the complete Distributor pulse.

When CP2 occurs during DP1 a parallel load of the F-bus into the Accumulator takes place if SAR and SAL are enabled. Thus, a transfer of data from the Buffer Register to the Accumulator has occurred. See page 1.

DP2 (DECA) Decrement the Accumulator  
(EC) End Carry  
(SCARY) Set Carry (If EC = 1)

During this Distributor pulse, the contents of the Accumulator will be decreased by one, and a carry condition will be detected if it exists.

RSO and DP2 are ANDed to enable DECA. See page 22. DECA is applied to the steering logic of the Arithmetic Logic Unit (ALU) and causes it to subtract one from the Accumulator output. The result is available on the F-bus. See page 2. DECA also causes SAL and SAR to be HIGH. At CP2 the F-bus contents will be transferred into the Accumulator. See pages 1 and 23.

If an End Carry (EC) was generated by the ALU, the Carry flip-flop will be set. See page 19.

DP3 No operation

DP4 (TAB) Transfer Accumulator to Buffer

RPG and DP4 combine to generate a transfer Accumulator to Buffer signal (TAB). A signal which transfers the Accumulator contents to the Z-bus (TAZ) is also generated at this time. See page 23. TAZ gates the Accumulator contents to the Z-bus. See page 1. TAB enables a signal which transfers the Z-bus to the Buffer Register (TZB). See page 19. At CP2, TZB strobes the Z-bus data into the Buffer Register and onto the Y-bus. See page 3.

DP5 (IBS) Initiate Buffer to Storage

The data in the Buffer Register will now be stored in memory. Replace Group (RPG) and DP5 generate IBS, which initiates a memory cycle which stores the information on the Y-bus at the addressed memory location. See pages 8 and 24.

DP6 No operation

DP7 (SDP15) Set the Distributor to a count of 15

RPG and DP7 enable SDP15. See page 21. This signal sets the Distributor to a count of 15. See page 5.

DP8  
through  
DP14

No operation

DP15 STCC, CKE

INCREASE INDEX - INX

This instruction increases the count of the Index Register by the value present in the Buffer Register, Instruction Operand.

OP CODE - 03<sub>16</sub>

DP0	DP1	DP2	DP3	DP4 - DP6	DP7
	TBM	AXM	TMX		SDP15

DP8 - DP14	DP15
	STCC
	CKE

DP0 No operation

DP1 (TBM) Transfer the Buffer contents to the M-Register

INX and DP1 enable a transfer of the Buffer Register to the Memory Address Register (TBM). See page 20. TBM strobes the Y-bus data into the Memory Address Register. See page 11.

DP2 (AXM) Add the Index Register contents to the M-Register contents

INX and DP2 generate an add Index to Memory Address signal (AXM). See page 23. AXM strobes the Memory Address Register contents to the Index Adder where they are summed. The result is present on the G-bus. See page 14.

AXM also enables TCM. See page 20. TCM strobes the G-bus data into the Memory Address Register during CP3. See page 11.

DP3 (TMX) Transfer the Memory Address Register contents to the X-Register

INX and DP3 generate a Transfer M to X (TMX) and Load X (LX). See page 22. TMX selects the Memory Address Register inputs at the Index Register, and LX at CP2 strobes the M data into the Index Register. See page 14.

DP4  
through  
DP6

No operation

DP7 (SDP15) Set the Distributor to a count of 15

INX and DP7 enable an SDP15 pulse. See page 21. This signal sets the Distributor to a count of 15. See page 5.

DP8  
through  
DP14

No operation



DP15 STCC, CKE

SHIFT LEFT ARITHMETIC - SLA

This instruction shifts the double length number in the Accumulator and Quotient Register to the Left one bit position for each count of the Countdown Register.

OP CODE - 0B<sub>16</sub>

DP0	DP1	DP2	DP3	DP4 - DP14	DP15
TBC	IF C=0	SAQL	DEC		STCC
	SDP15		SDP1		CKE

DP0 (TBC) Transfer the contents of the Buffer to the C-Register

SLA, SRA, SLL, or SRL enable the Shift Group (SFG) signal. See page 18. SFG and DP0 generate TYC which transfers the Buffer Register contents via the Y-bus to the Countdown Register. TYC sets the Countdown Register during CP1 and then loads it with the Buffer Register contents during CP2. See pages 4 and 22.

DP1 If the C-Register contains zero set Distributor pulse 15

If the Countdown Register contains zero, CZ will be enabled. See page 4. CZ, SFG, and DP1 generate SDP15 which sets the Distributor to a count of 15. See pages 5 and 21.

DP2 (SAQL) Shift the AQ Register to the Left

SLA and DP2 enable SAQL and SQL. See page 22. SAQL also enables SAL. See page 23. SQL causes the contents of the Quotient Register to shift one bit position to the left at CP2. See page 6. SAL will cause the Accumulator contents to shift one bit position to the left at CP2. See page 1. The Quotient Register MSB (Q7) will be shifted into the Accumulator LSB (A0). Q7 enables DSLA. See page 24. DSLA causes a logic one to be shifted into A0 at CP2. See page 1.

DP3 (DEC) Decrement the C-Register  
(SDP1) Set the Distributor to a count of 1

SFG and DP3 generate a DEC pulse which decreases the value of the Countdown Register by one. See pages 4 and 24. SFG and DP3 generate an SDP1 pulse which sets the Distributor to a count of 1. See pages 5 and 24.

DP4  
through  
DP14 No operation

DP15 STCC, CKE

SHIFT RIGHT ARITHMETIC - SRA

This instruction shifts the double length number in the Accumulator and Quotient Register to the right one bit position for each count of a Countdown Register.

OP CODE - 10<sub>16</sub>

DPO	DP1	DP2	DP3	DP4 - DP14	DP15
TBC	IF C=0 SDP15	SAQR	DEC SDP1		STCC CKE

DPO (TBC) Transfer the Buffer contents to the C-Register

SLA, SRA, SLL, or SRL enable the Shift Group (SFG) signal. See page 18. SFG and DPO generate TYC which transfers the Buffer Register contents via the Y-bus to the Countdown Register. TYC sets the Countdown Register during CP1 and then loads it with the Buffer Register contents during CP2. See pages 4 and 22.

DP1 If the C-Register contains zero set the Distributor to a count of 15

If the Countdown Register contains zero, CZ will be enabled. See page 4. CZ, SFG, and DP1 generate SDP15 which sets the Distributor to a count of 15. See pages 5 and 21.

DP2 (SAQR) Shift the AQ Register to the Right

SRA and DP2 generate SAQR and SQR. See page 22. SAQR also enables SAR. See page 23. SQR causes the contents of the Quotient Register to be shifted one bit position to the right at CP2. The Accumulator LSB (A0) will be shifted into the Quotient Register MSB (Q7). See page 6. SAR causes the Accumulator contents to be shifted one bit position to the right at CP2. See page 1. A7 enables DSRA. See page 23. DSRA causes A7 to remain unchanged during SRA.

DP3 (DEC) Decrease the C-Register count by one  
(SDP1) Set the Distributor to a count of 1

SFG and DP3 generate a DEC pulse which decreases the value of the Countdown Register by one. See page 4 and 24. SFG and DP3 also generate an SDP1 pulse which sets the Distributor to a count of 1. See pages 5 and 24.

DP4 through DP14 No operation

DP15 STCC, CKE

SHIFT LEFT LOGICAL - SLL

The Accumulator contents will be shifted one bit position to the left for each count of the Countdown Register.

OP CODE - 13<sub>16</sub>

DP0	DP1	DP2	DP3	DP4 - DP14	DP15
TBC	IF C=0 SDP15	SAL	DEC SDP1		STCC CKE

DP0 (TBC) Transfer the contents of the Buffer to the C-Register

SLA, SRA, SLL, or SRL enable the Shift Group (SFG) signal. See page 18. SFG and DP0 generate TYC which transfers the Buffer Register contents via the Y-bus to the Countdown Register. TYC sets the Countdown Register during CP1 and then loads it with the Buffer Register contents during DP2. See pages 4 and 22.

DP1 If the C-Register is zero, set Distributor pulse 15

If the Countdown Register contains zero, CZ will be enabled. See page 4. CZ, SFG, and DP1 generate SDP15 which sets the Distributor to a count of 15. See pages 5 and 21.

DP2 (SAL) Shift the Accumulator contents Left

SLL and DP2 generate SAL. See page 23. SAL causes the Accumulator contents to be shifted to the left one bit position at CP2. See page 1. DSLA is disabled by  $\overline{\text{SLL}}$ , causing a zero to be shifted into A0. See page 24.

DP3 (DEC) Decrease the count of the C-register by one  
(SDP1) Set Distributor pulse 1

SFG and DP3 generate a DEC pulse which decreases the value of the Countdown Register by one. See pages 4 and 24. SFG and DP3 also generate an SDP1 pulse which sets the Distributor to a count of 1. See pages 5 and 24.

DP4 through DP14 No operation

DP15 STCC, CKE

SHIFT RIGHT LOGICAL - SRL

The Accumulator contents will be shifted one bit position to the right for each count of the Countdown Register.

OP CODE - 18<sub>16</sub>

DP0	DP1	DP2	DP3	DP4 - DP14	DP15
TBC	IF C=0 SDP15	SAR	DEC SDP1		STCC CKE

DPO (TBC) Transfer the B-Register contents to the C-Register

SLA, SRA, SLL, or SRL enable the Shift Group (SFG) signal. See page 18. SFG and DPO generate TYC which transfers the Buffer Register contents via the Y-bus to the Countdown Register. TYC sets the Countdown Register during CP1 and then loads it with the Buffer Register contents during CP2. See pages 4 and 22.

DP1 Set Distributor Pulse 15 if the C-Register contains zero

If the Countdown Register contains zero, CZ will be enabled. See page 4. CZ, SFG, and DP1 generate SDP15 which sets the Distributor to a count of 15. See pages 5 and 21.

DP2 (SAR) Shift the A-Register contents Right

SRL and DP2 generate SAR. See page 23. SAR causes the Accumulator contents to be shifted one bit position to the right at CP2. See page 1. DSRA is not enabled, therefore, a zero is shifted into A7. See page 23.

DP3 (DEC) Decrease the Count of the C-Register  
(SDP1) Set the Distributor to a Count of 1

SFG and DP3 generate a DEC pulse which decreases the value of the Countdown Register by one. See pages 4 and 24. SFG and DP3 also generate an SDP1 pulse which sets the Distributor to a count of 1. See pages 5 and 24.

DP4 through DP14 No operation

DP15 STCC, CKE

AND - AND

This instruction performs a bit by bit logical AND of the contents of the Buffer Register and the Accumulator, leaving the result in the Accumulator.

OP CODE - 19<sub>16</sub>

DPO	DP1	DP2	DP3 - DP14	DP15
IAND		SDP15		STCC CKE

DPO (IAND) Initiate AND

AND and DPO enable IAND. See page 22. This signal is applied to the steering gates of the Arithmetic Logic Units (ALU). The ALU performs a bit by bit logical AND of the Buffer Register and Accumulator contents. The result is available on the F-bus. See page 2.

IOR, AND, or XOR instructions enable the Logic Group signal (LCG). See page 18. LCG and DPO cause SAR and SAL to be enabled for the complete

Distributor pulse. See page 23. At CP2, with SAL and SAR enabled, the F-bus is loaded into the Accumulator. See page 1.

DP1 No operation

DP2 (SDP15) Set Distributor pulse 15

LCG and DP2 generate an SDP15 pulse. See page 21. This signal sets the Distributor to a count of 15. See page 5.

DP3 through DP14 No operation

DP15 STCC, CKE

INCLUSIVE OR - IOR

This instruction performs a bit by bit Inclusive OR of the contents of the Buffer Register and the Accumulator, leaving the result in the Accumulator.

OP CODE -  $1A_{16}$

DPO	DP1	DP2	DP3 - DP14	DP15
IOR1		SDP15		STCC CKE

DPO (IORI) Initiate Inclusive OR

IOR and DPO enable IORI. See page 22. This signal is applied to the steering gates of the Arithmetic Logic Units (ALU). The ALU performs a bit by bit Inclusive OR of the Buffer Register and Accumulator contents. The result is available on the F-bus. See page 2.

IOR, AND, or XOR instructions enable the Logic Group signal (LCG). See page 18. LCG and DPO cause SAR and SAL to be enabled for the complete Distributor pulse. See page 23. At CP2 with SAL and SAR enabled, the F-bus is loaded into the Accumulator. See page 1.

DP1 No operation

DP2 (SDP15) Set the Distributor to a count of 15

LCG and DP2 generate an SDP15 pulse. See page 21. This signal sets the Distributor to a count of 15. See page 5.

DP3 through DP14 No operation

DP15 STCC, CKE

EXCLUSIVE OR - XOR

This instruction performs a bit by bit Exclusive OR of the contents of the Buffer Register and the Accumulator, leaving the result in the Accumulator.

OP CODE - 1B<sub>16</sub>

DPO	DP1	DP2	DP3 - DP14	DP15
1EX		SDP15		STCC CKE

DPO (1EX) Initiate Exclusive OR

XOR and DPO enable 1EX. See page 22. This signal is applied to the steering gates of the Arithmetic Logic Units (ALU). The ALU performs a bit by bit Exclusive OR of the Buffer Register and Accumulator contents. The result is available on the F-bus. See page 2.

IOR, AND, or XOR instructions enable the Logic Group signal (LCG). See page 18. LCG and DPO cause SAR and SAL to be enabled for the complete Distributor pulse. See page 23. At CP2 with SAL and SAR enabled, the F-bus is loaded into the Accumulator. See page 1.

DP1 No operation

DP2 (SDP15) Set Distributor pulse 15

LCG and DP2 generate an SDP15 pulse. See page 21. This signal sets the Distributor to a count of 15. See page 5.

DP3 through DP14 No operation.

DP15 STCC, CKE

BRANCH UNCONDITIONAL - BUN

This instruction causes the program to branch to the location designated by the operand which is contained in the Memory Address Register.

OP CODE - 90<sub>16</sub>

DPO	DP1	DP2 - DP14	DP15
TMP	SDP15		STCC CKE

DPO (TMP) Transfer the contents of the M-Register to the P-Register

BUN and DPO generate TMP. See page 21. TMP strobes the Memory Address Register contents into the Program Register at CP2. See page 12.

DP1 (SDP15) Set the Distributor to a count of 15

BUN and DP1 enable SDP15. See page 21. This pulse sets the Distributor to a count of 15. See page 5.

DP2  
through  
DP14 No operation

DP15 STCC, CKE

BRANCH AND STOP - BST

This instruction causes the program to branch to the location designated by the operand, which is contained in the Memory Address Register, and stop.

OP CODE - 98<sub>16</sub>

DPO	DP1	DP2 - DP14	DP15
TMP	SDP15		STCC
STOP			CKE

DPO (TMP) Transfer the contents of the M-Register to the P-Register

BST and DPO generate TMP. See page 21. TMP strobes the Memory Address Register contents into the Program Register at CP2. See page 12.

BST and DPO also generate STOP. See page 20. STOP enables a stop clock signal (SPCK) which stops the computer clock. See pages 7 and 18.

DP1 (SDP15) Set Distributor pulse 15

BST and DP1 generate SDP15. See page 21. This signal sets the Distributor to a count of 15. See page 5.

DP2  
through  
DP14 No operation

DP15 CKE, STCC

BRANCH TO SUB-ROUTINE - BSB

This instruction stores an Unconditional Branch operation code in the addressed memory location. It then stores the next instruction address at the next consecutive memory address. It then causes the program to branch to the next consecutive memory location.

DP0	DP1	DP2	DP3	DP4	DP5	DP6
TPHB	IBS	INCM	TPLB	IBS	INCM	TMP

DP7	DP8 - DP14	DP15
SDP15		STCC CKE

DP0 (TPHB) Transfer the Two High-Order Bits of P (P9 and P8) to the B-Register (B1 and B0) Set B4 and B7

BSB and DP0 generate TPHB. See page 20. TPHB enables Z7 and Z4. It also transfers the data present in P8 and P9 to Z0 and Z1, respectively. See page 12. TPHB also generates TZB. See page 19. This signal transfers the Z-bus data into the Buffer Register at CP2. See page 3.

The Buffer Register now contains 1001 0000. This is the Operation code for Branch Unconditional. The two low order bit positions contain page bits if they were present in the Program Register.

DP1 (IBS) Initiate Buffer to Storage

BSB and DP1 or DP4 generate IBS. See page 24. IBS initiates a memory cycle which stores the contents of the Buffer Register via the Y-bus at the addressed memory location. See page 8.

DP2 (INCM) Increment the M-Register

BSB and DP2 or DP5 generate INCM (page 24) which enables a transfer of the G-bus by Memory Address Register (TGM). See page 20. The G-bus data is generated by the Index Adder. During INCM time the Index Register contents are inhibited from the Index Adder. Thus, the Index Register inputs to the adder are all zeros except in the case of bit 0 which is inverted. This enables the adder to add 1 to the contents of the Memory Address Register. See page 11.

DP3 (TPLB) Transfer the 8 low order bits of the P-Register to the Buffer

TPLB is enabled by BSB and DP3. See page 20. This signal strobes P0 - P7 to the Z-bus. See page 12. TPLB also generates TZB. See page 19. TZB strobes the Z-bus data into the Buffer Register at CP2. See page 3.

DP4 See DP1 description.

DP5 See DP2 description.

DP6 (TMP) Transfer the contents of the M-Register to the P-Register

BSB and DP6 enable TMP. See page 21. This signal strobes the Memory Address Register contents into the Program Register at CP2. See page 12.



DP7 (SDP15) Set Distributor pulse 15

BSB and DP7 enable SDP15. See page 21. SDP15 sets the Distributor to a count of 15. See page 5.

DP8 through DP14 No operation

DP15 STCC, CKE

BRANCH ON POSITIVE - BPS

If the Condition Code is greater than zero, the program will branch to the location designated by the operand, which is contained in the Memory Address Register.

OP CODE - A8<sub>16</sub>

DPO	DP1	DP2 - DP14	DP15
IF CC>0 TMP	SDP15		STCC CKE

DPO If the Condition Code is greater than zero (CC>0) transfer the contents of the M-Register to the P-Register (TMP)

CCGT, BPS, and DPO enable TMP. See page 21. TMP strobes the Memory Address Register contents into the Program Register at CP2. See page 12.

DP1 (SDP15) Set Distributor pulse 15

BPS and DP1 generate SDP 15. See page 21. This signal sets the Distributor to a count of 15. See page 5.

DP2 through DP14 No operation

DP15 STCC, CKE

BRANCH ON ZERO - BZE

If the Condition Code is equal to zero, the program will branch to the location designated by the operand, which is contained in the Memory Address Register.

OP CODE - B0<sub>16</sub>

DPO	DP1	DP2 - DP14	DP15
IF CC=0 TMP	SDP15		STC CKE

**DP0** If the Condition Code is equal to zero (CC=0) transfer the contents of the M-Register to the P-Register

CCEQ, BZE, and DP0 enable TMP. See page 21. TMP strobes the Memory Address Register contents into the Program Register at CP2. See page 12.

**DP1** (SDP15) Set Distributor pulse 15

BZE and DP1 generate SDP15. See page 21. This signal sets the Distributor to a count of 15. See page 5.

DP2 through DP14 No operation

DP15 STCC, CKE

**BRANCH ON NEGATIVE - BNG**

If the Condition Code is less than zero, the program will branch to the location designated by the operand, which is contained in the Memory Address Register.

**DP CODE - B8<sub>16</sub>**

DP0	DP1	DP2 - DP14	DP15
IF CC<0 TMP	SDP15		STCC CKE

**DP0** If the Condition Code is less than Zero (CC<0) transfer the contents of the M-Register to the P-Register (TMP)

CCLT, BNG, and DP0 enable TMP. See page 21. TMP strobes the Memory Address Register contents into the Program Register at CP2. See page 12.

**DP1** (SDP15) Set Distributor pulse 15

BNG and DP1 generates SDP15. See page 21. This signal sets the Distributor to a count of 15. See page 5.

DP2 through DP14 No operation

DP15 STCC, CKE

**BRANCH ON NO CARRY - BNC**

If the Condition Code is carry, the program will branch to the location designated by the operand, which is contained in the Memory Address Register.

OP CODE - C0<sub>16</sub>

DPO	DP1	DP2 - DP14	DP15
IF CARRY = 0 TMP	SDP15		STCC  CKE

DPO            If the Condition Code is not Carry, transfer the contents of the M-Register to the P-Register (TMP)

$\overline{\text{CARY}}$ , BNC, and DPO enable TMP. See page 21. TMP strobes the Memory Address Register contents into the Program Register at CP2. See page 12.

DP1            (SDP15) Set Distributor pulse 15

BNC and DP1 generate SDP15. See page 21. This signal sets the Distributor to a count of 15. See page 5.

DP2  
through  
DP14            No operation

DP15            STCC, CKE

BRANCH ON INDEX ZERO - BXZ

If the Index Register contains zero, the program will branch to the location designated by the operand which is contained in the Memory Address Register.

OP CODE - C8<sub>16</sub>

DPO	DP1	DP2 - DP14	DP15
IF X=0 TMP	SDP15		STCC  CKE

DPO            If the X-Register is zero (X = 0) transfer M to P (TMP)

If the Index Register contains zeros in all bit locations, Index Zero (XZ) will be enabled. See page 14.

BXZ, XZ, and DPO enable TMP. See page 21. TMP strobes the Memory Address Register contents into the Program Register at CP2. See page 12.

DP1            (SDP15) Set Distributor pulse 15

BXZ and DP1 generate SDP15. See page 21. This signal sets the Distributor to a count of 15. See page 5.

DP2  
through  
DP14            No operation

DP15            STCC, CKE

SKIP ON INTERRUPT - SKI

This instruction causes the program to skip k number of instructions (2k words) if the interrupt bit is set. The interrupt bit is cleared.

OP CODE - 08<sub>16</sub>

DPO	DP1	DP2	DP3	DP4	DP5	DP6
IF INT=0 SDP15	CLINT	TBC	TPM	IF C=0 SDP15	INCM DEC	INCM

DP7	DP8 - DP14	DP15
TMP		STCC
SDP4		CKE

DPO            If INT=0, SDP15

If the Interrupt bit is not set, the instruction will be disregarded. SKI, DPO, and  $\overline{INT}$  are gated to enable SDP15. See page 21. SDP15 sets the Distributor to a count of 15. See page 5.

DP1            (CLINT) Clear the Interrupt bit

SKI and DP1 are gated to reset the Interrupt flip-flop. See page 19.

DP2            (TBC) Transfer the Buffer contents to the C-Register

SKI, SKS, or SKF enable the Skip Group signal, (SKG). See page 18. Since the Buffer Register data is also present on the Y-bus SKG and DP2 generate a TYC signal. See page 22. This signal strobes the Y-bus data into the Countdown Register at CP2. See page 4.

DP3            (TPM) Transfer the P-Register contents to the M-Register

SKG and DP3 generate TPM. See page 24. TPM enables TPLB. See page 20. TPM strobes the two high order bits of the Program Register into the high order bit positions of the Memory Address Register. TPLB strobes the remaining 8 bits of the Program Register into the Memory Address Register at CP2. See pages 11 and 12.

DP4            If the C-Register equals zero (C=0) set Distributor pulse 15 (SDP15)

If the Countdown Register contains zero, CZ will be enabled. See page 4. SKG, DP4 and CZ enable SDP15. See page 21. This signal sets the Distributor to a count of 15. See page 5.

DP5 (INCM) Increment the M-Register  
(DEC) Decrement the C-Register

SKG and DP5 generate INCM. See page 24. INCM is used to enable a transfer of the G-bus to the Memory Address Register (TGM). See page 20. The G-bus data is generated by the Index Adder. During INCM time the Index Register contents are inhibited from the Index Adder. Thus, the Index Register inputs to the Adder are all zeros except in the case of bit 0 which is inverted. This enables the Adder to add 1 to the contents of the Memory Address Register. See pages 11 and 14.

SKG and DP5 also generate a DEC pulse which decreases the value of the Countdown Register by one. See pages 4 and 24.

DP6 (INCM) Increment the M-Register

SKG and DP6 generate INCM. See page 24. INCM is used to enable a transfer of the G-bus to the Memory Address Register (TGM). See page 20. The G-bus data is generated by the Index Adder. During INCM time the Index Register contents are inhibited from the Index Adder. Thus, the Index Register input to the Adder are all zeros except in the case of bit 0 which is inverted. This enables the Adder to add 1 to the contents of the Memory Address Register. See pages 11 and 14.

DP7 (TMP) Transfer the contents of the M-Register to the P-Register  
(SDP4) Set the distributor to a count of 4

DP7 and SKG generate TMP. See page 21. TMP strobes the Memory Address Register contents into the Program Register at CP2. See page 12. SKG and DP7 also enable SDP4. See page 21. SDP4 sets the Distributor to a count of 4. See page 5.

DP8 through DP14 No operation

DP15 STCC, CKE

SKIP ON SENSE SWITCH - SKS

This instruction causes the program to skip k number of instructions (2k words) if the Sense Switch is set.

OP CODE - 09<sub>16</sub>

DP0	DP1	DP2	DP3	DP4	DP5	DP6
IF SENSE=0 SDP15		TBC	TPM	IF C=0 SDP15	INCM DEC	INCM

DP7	DP8 - DP14	DP15
TMP SDP4		STCC DKE

- DP0      If Sense = 0 SDP15
- If the Sense Switch is not set, the instruction will be disregarded. SKS, DP0, and  $\overline{SEN}$  are gated to enable SDP15. See page 21. SDP15 sets the Distributor to a count of 15. See page 5.
- DP1      No operation
- DP2      (TBC) Transfer the Buffer contents to the C-Register
- SKI, SKS, or SKF enable the Skip Group signal (SKG). See page 18. Since the Buffer Register data is also present on the Y-bus SKG and DP2 generate a TYC signal. See page 22. This signal strobes the Y-bus data into the Countdown Register at CP2. See page 4.
- DP3      (TPM) Transfer the P-Register contents to the M-Register
- SKG and DP3 generate TPM. See page 24. TPM enables TPLB. See page 20. TPM strobes the two high order bits of the Program Register into the high order bit positions of the Memory Address Register. TPLB strobes the remaining eight bits of the Program Register into the Memory Address Register at CP2. See pages 11 and 12.
- DP4      If the C-Register contains zero (C=0) set Distributor pulse 15 (SDP15)
- If the Countdown Register contains zero, CZ will be enabled. See page 4. SKG, DP4, and CZ enable SDP15. See page 21. This signal sets the Distributor to a count of 15. See page 5.
- DP5      (INCM) Increment the M-Register  
          (DEC) Decrement the C-Register
- SKG and DP5 generate INCM. See page 24. INCM is used to enable a transfer of the G-bus to the Memory Address Register (TGM). See page 20. The G-bus data is generated by the Index Adder. During INCM time the Index Register contents are inhibited from the Index Adder. Thus, the Index Register inputs to the Adder are all zeros except in the case of bit 0 which is inverted. This enables the Adder to add 1 to the contents of the Memory Address Register. See pages 11 and 14.
- SKG and DP5 also generate a DEC pulse which decreases the value of the Countdown Register by one. See pages 4 and 24.
- DP6      (INCM) Increment the M-Register
- SKG and DP6 generate INCM. See page 24. INCM is used to enable a transfer of the G-bus to the Memory Address Register (TGM). See page 20. The G-bus data is generated by the Index Adder. During INCM time the Index Register contents are inhibited from the Index Adder. Thus, the Index Register inputs to the Adder are all zeros except in the case of bit 0 which is inverted. This enables the Adder to add 1 to the contents of the Memory Address Register. See pages 11 and 14.
- DP7      (TMP) Transfer the Memory Address Register contents to the P-Register  
          (SDP4) Set Distributor pulse 4
- DP7 and SKG generate TMP. See page 21. TMP strobes the Memory Address Register contents into the Program Register at CP2. See page 12.

SKG and DP7 also enable SDP4. See page 21. SDP4 sets the Distributor to a count of 4. See page 5.

DP8  
through  
DP14           No operation

DP15           STCC, CKE

SKIP ON FLAG - SKF

This instruction causes the program to skip k number of instructions (2k words) if the Flag is set.

OP CODE - 0A<sub>16</sub>

DP0	DP1	DP2	DP3	DP4	DP5	DP6
IF FLAG=0 SDP15		TBC	TPM	IF C=0 SDP15	INCM  DEC	INCM

DP7	DP8 - DP14	DP15
TMP		STCC
SDP4		CKE

DP0           If FLAG = 0 SDP15

If the Flag is not set, the instruction will be disregarded. SKF, DP0 and FLG are gated to enable SDP15. See page 21. SDP15 sets the Distributor to a count of 15. See page 5.

DP1           No operation

DP2           (TBC) Transfer the B-Register contents to the Countdown Register

SKI, SKS, or SKF enable the Skip Group Signal (SKG). See page 18. Since the Buffer Register data is also present on the Y-bus, SKG and DP2 generate a TYC signal. See page 22. This signal strobcs the Y-bus data into the Countdown Register at CP2. See page 4.

DP3           (TPM) Transfer the contents of the P-Register to the M-Register

SKG and DP3 generate TPM. See page 24. TPM enables TPLB. See page 20. TPM strobcs the two high order bits of the Program Register into the high order bit positions of the Memory Address Register. TPLB strobcs the remaining eight bits of the Program Register into the Memory Address Register at CP2. See pages 11 and 12.

DP4           If C is zero (C=0) set Distributor pulse 15 (SDP15)

If the Countdown Register contains zero, CZ will be enabled. See page 4. SKG, DP4, and CZ enable SDP15. See page 21. This signal sets the Distributor to a count of 15. See page 5.

DP5 (INCM) Increment the M-Register  
(DEC) Decrement the Countdown Register

SKG and DP5 generate INCM. See page 24. INCM is used to enable a transfer of the G-bus to the Memory Address Register (TGM). See page 20. The G-bus data is generated by the Index Adder. During INCM time the Index Register contents are inhibited from the Index Adder. Thus, the Index Register inputs to the Adder are all zeros except in the case of bit 0 which is inverted. This enables the Adder to add 1 to the contents of the Memory Address Register. See pages 11 and 14.

SKG and DP5 also generate a DEC Pulse which decreases the value of the Countdown Register by one. See page 4 and 24.

DP6 (INCM) Increment the M-Register contents

SKG and DP6 generate INCM. See page 24. INCM is used to enable a transfer of the G-bus to the Memory Address Register (TGM). See page 20. The G-bus data is generated by the Index Adder. During INCM time the Index Register contents are inhibited from the Index Adder. Thus, the Index Register inputs to the Adder are all zeros except in the case of bit 0 which is inverted. This enables the Adder to add 1 to the contents of the Memory Address Register. See pages 11 and 14.

DP7 (TMP) Transfer M to P  
(SDP4) Set Distributor pulse 4

DP7 and SKG generate TMP. See page 21. TMP strobes the Memory Address Register contents into the Program Register at CP2. See page 12.

SKG and DP7 also enable SDP4. See page 21. SDP4 sets the Distributor to a count of 4. See page 5.

DP8 through DP14 No operation

DP15 STCC, CKE

WRITE DATA BLOCK - WDB

This instruction allows the computer to output data to an external device. Data will be taken from memory starting from the memory location addressed by the Memory Address Register.

OP CODE - D0<sub>16</sub>

DP0	DP1	DP2	DP3	DP4	DP5	DP6
INW	ISB		INWD WAIT	IF C=0 SDP15		INCM DEC SDP1

DP7 - DP14	DP15
	STCC CKE



DP0 (INW) Initiate Write

INW is generated by WDB and DP0. See page 20. This signal sets up the Interface to receive data from the computer.

DP1 (ISB) Initiate Storage to Buffer

WDB generates Write Group (WG). See page 18. WG and DP1 enable an initiate storage to Buffer signal (ISB). See page 24. ISB initiates a memory cycle in which the data is taken from memory and placed on the Z-bus during CP2. See page 8. ISB also generates a transfer of the Z-bus to the Buffer Register (TZB). See page 19. TZB is ANDed with CP2 and this signal strobes the data into the Buffer Register. See page 3.

DP2 No operation

DP3 (INWD) Initiate Write Data  
(WAIT)

INWD is generated by DP3 and WDB. See page 20. This signal triggers the Interface clock, allowing the Interface to accept the data present on the Y-bus. INWD also enables the WAIT signal. See page 23. WAIT generates a stop clock (SPCK) signal. See page 18. This signal halts the computer clock. The clock restarts upon receiving a resume (RESM) pulse from the Interface, signaling that it has processed the received data and is ready to accept another data word. See page 7.

DP4 (SDP15) If the C-Register equals zero (C=0) set Distributor pulse 15

If the Countdown Register contains zero, CZ will be enabled. See page 4. WG, DP4, and CZ enable SDP15. See page 21. This signal sets the Distributor to a count of 15. See page 5.

DP5 No operation

DP6 (INCM) Increment the M-Register  
(DEC) Decrement the C-Register  
(SDP1) Set Distributor pulse 1

WG and DP6 generate INCM. See page 24. INCM is used to enable a transfer of the G-bus to the Memory Address Register (TGM). See page 20. The G-bus data is generated by the Index Adder. During INCM time the Index Register contents are inhibited from the Index Adder. Thus, the Index Register inputs to the Adder are all zeros except in the case of bit 0 which is inverted. This enables the Adder to add 1 to the contents of the Memory Address Register. See pages 11 and 14.

WG and DP6 also generate a DEC pulse which decreases the value of the Countdown Register by one. See pages 4 and 24. WG and DP6 generate an SDP1 pulse which sets the Distributor to a count of 1. See pages 5 and 24.

DP7  
through  
DP14 No operation

DP15 STCC

CKE

MANUAL OUTPUT - MNO

This instruction transfers the data at the addressed memory location to the Buffer and Input Register. One memory location is read for each manual START the computer receives.

OP CODE - D8<sub>16</sub>

DP0	DP1	DP2	DP3	DP4	DP5	DP6
	ISB	TBI WAIT		IF C=0 SDP15		INCM DEC SDP1

DP7 - DP14	DP15
	STCC
	CKE

DP0 No operation

DP1 (ISB) Initiate Storage to Buffer

MNO generates Write Group (WG). See page 18. WG and DP1 generate an initiate storage to Buffer signal (ISB). See page 24. ISB initiates a memory cycle in which the data is taken from memory and placed on the Z-bus during CP2. See page 8. ISB also generates a transfer of the Z-bus to the Buffer Register (TZB). See page 19. TZB is ANDed with CP2 and this signal strobes the data into the Buffer Register. See page 3.

DP2 (TBI) Transfer the contents of the Buffer to the Input Register (WAIT)

MNO and DP2 generate TYI. See page 23. This signal, gated with the Y-bus sets the proper flip-flops in the Input Register. The output of these flip-flops enable the Input Register indicators on the front panel. See page 10. MNO and DP2 also enables the WAIT signal. See page 23. WAIT generates a stop clock (SPCK) signal. See page 18. This signal halts the computer clock. The clock restarts upon receiving a manual START pulse from the control panel. See page 7.

DP3 No operation

DP4 (SDP15) If the C-Register contents are zero (C=0) set Distributor pulse 15

If the Countdown Register contains zero, CZ will be enabled. See page 4. WG, DP4, and CZ enable SDP15. See page 21. This signal sets the Distributor to a count of 15. See page 5.

DP5 No operation

DP6 (INCM) Increment the M-Register  
 (DEC) Decrement the C-Register  
 (SDP1) Set Distributor pulse 1

WG and DP6 generate INCM. See page 24. INCM is used to enable a transfer of the G-bus to the Memory Address Register (TGM). See page 20. The G-bus data is generated by the Index Adder. During INCM time the Index Register contents are inhibited from the Index Adder. Thus, the Index Register inputs to the Adder are all zeros except in the case of bit 0 which is inverted. This enables the Adder to add 1 to the contents of the Memory Address Register. See pages 11 and 14.

WG and DP6 also generate a DEC pulse which decreases the value of the Countdown Register by one. See pages 4 and 24. WG and DP6 generates an SDP1 pulse which sets the Distributor to a count of 1. See page 5 and 24.

DP7 through DP14 No operation

DP15 STCC, CKE

READ DATA BLOCK - RDB

This instruction allows the computer to read data furnished from an external device. The number of data words that will be read is dependent upon the value contained in the Countdown Register.

OP CODE - E0<sub>16</sub>

DP0	DP1	DP2	DP3	DP4	DP5	DP6
INR	INRD	TEB	IBS	IF C=0 SDP15		INCM
WAIT	WAIT					DEC
						SDP1

DP7	DP14	DP15
		STCC
		CKE

DP0 (INR) Initiate Read  
 (WAIT)

RDB or RDI and DP0 enable INR. This signal sets up the selected Interface to input data to the computer. See pages 20 and 28. INR also enables the WAIT signal. See page 23. WAIT generates a stop clock (SPCK) signal. See page 18. This signal halts the computer until it receives a resume (RESM) pulse from the Interface, signaling that it is ready to process data.

DP1 (INRD) Initiate Read Data

INRD is generated by RDB or RDI and DP1. See page 20. This signal triggers the Interface clock, allowing the Interface to accept data from the external device. INWD also enables the WAIT signal. See page 23. WAIT generates a stop clock (SPCK) signal. See page 18. This signal halts the computer clock. The clock restarts upon receiving a resume (RESM) pulse from the Interface, signaling that it has processed the received data and is ready to accept another data word. See page 7.

DP2 (TEB) Transfer External Data to the Buffer

RDB or RDI and DP2 generate TEB which transfers the data from the Interface, via the Z-bus, to the Buffer Register. See page 20. TEB also enables TZB which strobes the data into the Buffer Register. See pages 3 and 19.

DP3 (IBS) Initiate Buffer to Storage

TDB, TDI, or MNI enable Read Group RDG. See page 20. RDG and DP3 enable an initiate Buffer to storage signal (IBS). See page 24. IBS initiates a memory cycle which stores the contents of the Buffer Register, via the Y-bus, at the addressed memory location. See page 8.

Buffer Register. See page 3.

DP4 (SDP15) If the C-Register equals zero (C=0) set Distributor pulse 15

If the Countdown Register contains zero, CZ will be enabled. See page 4. DP4, RDB, and CZ enable SDP15. See page 21. This signal sets the Distributor to a count of 15. See page 5.

DP5 No operation

DP6 (INCM) Increment the M-Register  
(DEC) Decrement the Value in the C-Register  
(SDP1) Set Distributor Pulse 1

RDG and DP6 generate INCM. See page 24. INCM is used to enable a transfer of the G-bus to the Memory Address Register (TGM). See page 20. The G-bus data is generated by the Index Adder. During INCM time the Index Register contents are inhibited from the Index Adder. Thus, the Index Register inputs to the Adder are all zeros except in the case of bit 0, which is inverted. This enables the Adder to add 1 to the contents of the Memory Address Register. See pages 11 and 14.

RDB and DP6 generate a DEC Pulse which decreases the value of the Countdown Register by one. See pages 4 and 24. RDG and DP6 generate an SDP1 pulse which sets the Distributor to a count of 1. See pages 5 and 24.

DP7  
through  
DP14 No operation

DP15 STCC

CKE

READ UNTIL INTERRUPT - RDI

This instruction allows the computer to read data furnished by an external device until it receives an interrupt pulse which acts as an end-of-data signal.

OP CODE - E8<sub>16</sub>

DP0	DP1	DP2	DP3	DP4 - DP5	DP6
INR	INRD	TEB	IBS		INCM
WAIT	WAIT	IF INT=1 SDP15			SDP1

DP7 - DP14	DP15
	STCC
	CKE

DP0 (INR) Initiate Read  
(WAIT)

RDB or RDI and DP0 enable INR. This signal sets up the selected Interface to input data to the computer. See pages 20 and 28. INR also enables the WAIT signal. See page 23. WAIT generates a stop clock (SPCK) signal. See page 18. This signal halts the computer clock. The clock restarts upon receiving a resume (RESM) pulse from the Interface, signaling that it has processed the received data and is ready to accept another data word. See page 7.

DP1 (INRD) Initiate Read Data  
(WAIT)

INRD is generated by RDB or RDI and DP1. See page 20. This signal triggers the Interface to accept the data present on the Y-bus. INRD also enables the WAIT signal. See page 23. WAIT generates a stop clock (SPCK) signal. See page 18. This signal halts the computer clock. The clock restarts upon receiving a resume (RESM) pulse from the Interface, signaling that it has processed the received data and is ready to accept another data word. See page 7.

DP2 (TEB) Transfer external data to Buffer  
(SDP15) If the Interrupt bit is set (INT=1) set Distributor pulse 15

RDB or RDI and DP2 generate TEB which transfers the data from the Interface, via the Z-bus, to the Buffer Register. See page 20. TEB also enables TZB which strobes the data into the Buffer Register. See pages 3 and 19.

INT, DP2, and RDI enable SDP15. See page 21. This signal sets the Distributor to a count of 15. See page 5.

DP3 (IBS) Initiate Buffer to Storage

RDB, RDI, or MNI enable Read Group (RDG). See page 20. RDG and DP3 enable an initiate Buffer to storage signal (IBS). See page 24. IBS initiates a memory cycle which stores the contents of the Buffer Register, via the Y-bus, at the addressed memory location. See page 8.

DP4 through DP5 No operation

DP6 (INCM) Increment the M-Register  
(SDP1) Set Distributor pulse 1

RDG and DP6 generate INCM. See page 24. INCM is used to enable a transfer of the G-bus to the Memory Address Register (TGM). See page 20. The G-bus data is generated by the Index Adder. During INCM time the Index Register contents are inhibited from the Index Adder. Thus, the Index Register input to the Adder are all zeros except in the case of bit 0 which is inverted. This enables the Adder to add 1 to the contents of the Memory Address Register. See pages 11 and 14. RDG and DP6 generate an SDP1 pulse which sets the Distributor to a count of 1. See pages 5 and 24.

DP7 through DP14 No operation

DP15 STCC, CKE

MANUAL INPUT - MNI

This instruction allows the entering of data directly into Memory from the Input Register on the control panel. The number of data words which are entered is one more than the initial count in the Countdown Register.

OP CODE - FO<sub>16</sub>

DP0	DP1	DP2	DP3	DP4	DP5	DP6
	WAIT	TIB	IBS	IF C=0 SDP15		INCM DEC SDP1

DP7 - DP14	DP15
	STCC
	CKE

DP0 No Operation

DP1 (WAIT)

MNI and DP1 enable the WAIT signal. See page 23. WAIT generates a clock stop (SPCK) signal which halts the computer clock. See pages 7 and 18.

The clock will restart when it receives a manual START pulse from the Control Panel. See page 15.

DP2 (TIB) Transfer the Input Register contents to the Buffer

The data present in the Input Register is transferred to the Buffer Register. MNI and DP2 enable TIB. See page 24. TIB enables TZB which strobes the data into the Buffer Register. See page 3.

DP3 (IBS) Initiate Buffer to Storage

RDB, RDI, or MNI enable Read Group (RDG). See page 20. RDG and DP3 enable an initiate Buffer to storage signal (IBS). See page 24. IBS initiates a memory cycle which stores the contents of the Buffer Register via the Y-bus at the addressed memory location. See page 8.

DP4 (SDP15) If the C-Register contains zero (C=0) set Distributor pulse 15

If the Countdown Register contains zero, CZ will be enabled. See page 4. DP4, MNI, and CZ enable SDP15. See page 21. The signal sets the Distributor to a count of 15. See page 5.

DP5 No operation

DP6 (INCM) Increment the M-Register  
(DEC) Decrement the C-Register  
(SDP1) Set Distributor pulse 1

RDG and DP6 generate INCM. See page 24. INCM is used to enable a transfer of the G-bus to the Memory Address Register (TGM). See page 20. The G-bus data is generated by the Index Adder. During INCM time the Index Register contents are inhibited from the Index Adder. Thus, the Index Register input to the Adder are all zeros except in the case of bit 0 which is inverted. This enables the Adder to add 1 to the contents of the Memory Address Register. See pages 11 and 14.

MNI and DP6 generate a DEC pulse which decreases the value of the Countdown Register by one. See pages 4 and 24. RDG and DP6 generate an SDP1 pulse which sets the Distributor to a count of 1. See pages 5 and 24.

DP7 through DP14 No operation

DP15 STCC, CKE

OUTPUT COMMAND - OCD

This instruction selects the external device which will Interface with the computer.

OP CODE - 11<sub>16</sub>

DP0	DP1	DP2	DP3 - DP14	DP15
INOC		SDP15		STCC CKE

DPO (INOC) Initiate Output Command

OCD and DPO enable initiate output command (INOC). See page 20. This signal selects and enables the Interface addressed by the instruction operand.

DP1 No operation

DP2 (SDP15) Set Distributor pulse 15

OCD and DP2 enable SDP15. See page 21. This signal sets the Distributor to a count of 15. See page 5.

DP3 through DP14 No operation

DP15 STCC, CKE

SENSE STATUS - SST

This instruction allows the previously selected external device to send an 8-bit status word to the Accumulator.

OP CODE - 00<sub>16</sub>

DPO	DP1	DP2	DP3	DP4 - DP14	DP15
INSS	TEB	TBA	SDP15		STCC CKE

DPO (INSS) Initiate Sense Status

SST and DPO enable Initiate Sense Status (INSS). See page 20. This signal enables the Interface to generate a Status word.

DP1 (TEB) Transfer external data to the Buffer

DP1 and SST generate TEB. See page 20. TEB transfers the data word, which is stored in the output registers of the Interface to the Buffer Register via the Z-bus TEB also enables TZB which strobes the data into the Buffer Register. See pages 3 and 19.

DP2 (TBA) Transfer the Buffer contents to the Accumulator

The data present in the Buffer Register is transferred to the Accumulator. The data present in the Buffer Register is available on the Y-bus. Since the computer is not doing an arithmetic operation, data in the Y bus is transferred through the Arithmetic Logic Units (ALU) without the use of a strobe pulse. Thus, the data in the Buffer is also available at the output of the ALU or the F-bus. See page 2.

SST and DP2 enable two signals: Shift the Accumulator Right (SAR) and Shift the Accumulator Left (SAL). See page 23. Both these signals are enabled for the complete Distributor pulse.



When CP2 occurs during DP2, a parallel load of the F-bus into the Accumulator takes place if SAR and SAL are enabled. Thus, a transfer of data from the Buffer to the Accumulator has occurred. See page 1.

DP3 (SDP15) Set Distributor pulse 15

SST and DP3 enable SDP15. See page 21. This signal sets the Distributor to a count of 25. See page 5.

DP4 through DP14

No operation

DP15 STCC, CKE

CLEAR FLAG - FLC

This instruction clears the Flag flip-flop under program control.

OP CODE - 28<sub>16</sub>

DP0	DP1	DP2	DP3 - DP14	DP15
CLF		SDP15		STCC CKE

DP0 (CLF) Clear the Flag flip-flop

FLC and DP0 are ANDED to clear the Flag flip-flop. See page 19.

DP1 No operation

DP2 (SDP15) Set Distributor pulse 15

FLC and DP2 enable SDP15. See page 21. This signal sets the Distributor to a count of 15. See page 5.

DP3 through DP14

No operation

DP15 STCC, CKE

SET FLAG - FLS

This instruction sets the Flag bit under program control.

OP CODE - F8<sub>16</sub>

DP0	DP1	DP2	DP3 - DP14	DP15
STF		SDP15		STCC CKE

DP0 (STF) Set Flag flip-flop  
 FLS and DP0 are ANDED to set the Flag flip-flop. See page 19.

DP1 No operation

DP2 (SDP15) Set Distributor pulse 15  
 FLS and DP2 generate SDP15. See page 21. This signal sets the Distributor to a count of 15. See page 5.

DP3 through DP14 No operation

DP15 STCC, CKE

#### REVIEW QUESTIONS 2-17

Match the operation with its associated timing pulses.

1. \_\_\_ Reads the OP CODE from memory
2. \_\_\_ During execution time causes the index register to be added to the operand
3. \_\_\_ Sets the condition codes >0, = 0, <0
4. \_\_\_ Transfer the P-register high order bits to the Buffer and sets B7 and B4
5. \_\_\_ Sets the distributor (D-register) to a count of 11
6. \_\_\_ Changes the distributor pulses from acquisition to execution phase
7. \_\_\_ Causes indexing if bit 2 of the OP CODE register (S-register) is set
8. \_\_\_ Checks the countdown register during a multiply
9. \_\_\_ Transfers the Buffer to the Q-register
10. \_\_\_ Transfer the P-register to the M-register during execution of a skip instruction
  - a. DP0
  - b. DP2
  - c. DP6
  - d. DP11
  - e. DPA1
  - f. DPA9
  - g. DPA15
  - h. DP15
  - i. DP3

Circle the letter of the correct answer.

1. What is a function of the  $\overline{DPA4}$  pulse (sheet 18)?
  - a. Produces the  $\overline{TGM}$  signal
  - b. Produces the AXM signal
  - c. Given the address for the next OP CODE
  - d. Allow the distributor to be set to a count of 15

2. Which of the following is not a function of  $\overline{\text{DPA15}}$  (sheet 18)?
  - a. Clear divide error
  - b. Clear add error
  - c. Clear instruction error
  - d. Clear sign flip-flop
3. When will DP15 set the condition code less than zero CCLT (sheet 19)?
  - a. Q7 is 0 and doing a divide
  - b. Q7 is 1 and doing a divide
  - c. A7 is 1 and doing a divide
  - d. A7 is 0 and not doing a divide
4. When does the TW0s signal (sheet 3, 2F) go low?
  - a. DP2 of a multiply instruction if A doesn't equal 0
  - b. DP3 of a multiply instruction if Q doesn't equal 0
  - c. DP4 of a divide instruction if Q doesn't equal 0
  - d. DP3 of a divide instruction when Q doesn't equal 0
5. Which signal is not produced by DP0 of an STA instruction (sheet 23, D5)?
  - a. TAZ
  - b. TXB
  - c. TZB
  - d.  $\overline{\text{TAB}}$
6. Which of the following signals is produced by the STX instruction at DP0 (sheet 23, 1D)?
  - a.  $\overline{\text{TMX}}$
  - b.  $\overline{\text{IBS}}$
  - c. TZB
  - d.  $\overline{\text{SDP15}}$
7. During DP1 of an LDA instruction, which of the following signals are produced (sheet 18, D8 and C6)?
  - a. SAR and SAL
  - b.  $\overline{\text{TAB}}$  and TAZ
  - c.  $\overline{\text{ISB}}$  and TZB
  - d.  $\overline{\text{CLA}}$

8. What conditions must be met at DPA3 to produce an instruction error ( $\overline{\text{SERI}}$ )? Reference sheet 17, 3E.
- S2 set and OP CODE is between 20(16) and FF(16)
  - S2 set and OP CODE is between 0(16) and 1F(16)
  - S2 clear and OP CODE is between 20(16) and 1F(16)
  - S2 clear and OP CODE is between 20(16) and FF(16)
9. If 78(16) is in the S-register, which decoder on sheet 17 is conditioned, and what output pin is low?
- 21G pin 1
  - 21C pin 1
  - 21G pin 17
  - 21L pin 17
10. Which of the following signals will not produce ( $\overline{\text{SPCK}}$ ) a stop clock (sheet 18, 7A)?
- $\overline{\text{DPI5}}$  in A/E mode
  - DPA15 in instruction mode
  - Every pulse in distributor mode
  - $\overline{\text{DPI5}}$  in instruction mode
11. Which of the following would prevent DPA15 from clocking the E flip-flop to execution phase (sheet 5, B7)?
- Distributor mode only
  - Instruction mode and RPT being used
  - Program mode and RPT being used
  - A/E mode and RPT being used
12. Which instruction uses the signal  $\overline{\text{Q6Q0}}$  (sheet 6 coordinate A5)?
- Multiply
  - Divide
  - Add, SLA, and SRA
  - Subtract, SLA, and SRA
13. Which of the following instructions will produce the TPLB signal during execution phase (sheet 12, 10F)?
- BST if B-register is clear
  - SKF if flag is clear
  - SKI if interrupt is set
  - SKS if sense is clear

14. There are 35 lamps on sheet 16; 8 for the index register, 10 for the program address, 10 for the memory address, 6 for the OP CODE register, and one additional lamp. What causes the extra lamp?
- Index bit
  - Sense switch
  - Sign bit for condition codes
  - Lamp test light
15. Which of the following conditions will not produce the  $\overline{\text{SAOV}}$  signal (sheet 1, A8)?
- Subtract a positive from a negative and get a positive difference
  - Subtract a negative from a positive and get a positive difference
  - Add a negative to a negative and get a positive sum
  - Add a positive to a positive and get a negative sum
16. What instruction will cause pin 8 of the ALU chip 5E (sheet 2, 3E) to be low?
- AND
  - IOR
  - SUB
  - XOR
17. The COMP signal on sheet 3, 10F, can be produced by all the following except:
- DIV
  - MPY
  - LAN
  - SLA
18. Which instruction cannot produce an SDP15 signal during execution phase?
- SKI
  - MNO
  - DIV
  - BST
19. What will be in the accumulator after DP2 of LAN instruction if the accumulator contains 55 and the Buffer contains F0 after DP0?
- AA
  - AB
  - OF
  - 10

## CHAPTER 3

### COMPUTER SYSTEM MAINTENANCE

#### COMPUTER DIAGNOSTIC PROGRAMS

To better guard our country in today's electronic world, the Air Force must be in constant readiness and be able to act with speed to any change in the world situation. The need for speed and reliability has placed strict requirements on both equipment and manpower to maintain and repair this equipment. The mission of the Air Force requires its electronic equipment to be reliable 100 percent of the time. As with all electronic equipment, computers are subject to malfunctions that must be repaired.

#### Purpose of Diagnostic Programs

When considering the enormous number of circuits and the numerous functions performed by some of the large general-purpose digital computers, one realizes that previously used methods of trouble isolation and detection are impractical and inadequate. Circuits designed to provide reliable data processing and computation, with speed and accuracy, require a more scientific method of preventive maintenance, as well as corrective maintenance.

The magnitude of the number of circuits and the many functions which they perform is an indication of the number and types of failures that can occur in equipment as complex as modern weapons systems. The basic circuits and combinations of these circuits are designed to perform specific operations rapidly, accurately, and reliably. If one of these circuits should fail to produce its specified output, the entire mission could be lost. Therefore, special preventive maintenance is necessary.

The maintenance program is one of the major tools used to maintain some of the complex systems. A maintenance program is any program designed to indicate whether the computer is able to correctly perform its intended design function. If improper operation occurs, the maintenance program should be able to specify the equipment which is responsible for the error(s). The primary function of any maintenance program is to maintain computer reliability--that is, to locate any existing or impending failure during scheduled maintenance periods so that no machine failures occur during operating time. Therefore, the maintenance program must attempt to test computer circuits as thoroughly as possible.

**FAILURE ISOLATION.** Maintenance programs used for trouble isolation and detection must check for many types of troubles. However, one can see that it would be impractical, if not impossible, to check every trouble possibility. In order to check the maximum number of conditions, the programs are written to detect classes of troubles so that further isolation of trouble within a class can be accomplished.

System failures have been classified into three states as follows:

- Catastrophic failures
- Intermittent failures
- Machine state failures

The catastrophic failure is one that is continuously present until it is repaired. It is the easiest to detect and locate. Programs designed to locate catastrophic failures usually exercise each unit of the equipment by using several different approaches to its operation. These approaches or techniques will be discussed later.

The intermittent failure is one that is not continuously present but that can produce inconsistent symptoms. An intermittent failure can both appear and disappear at random. Due to the inconsistent and erroneous symptoms that may occur during the program run, this type of failure is extremely difficult to isolate.

Machine state failures are similar to the intermittent failures in the inconsistency of indications that they may present. Machine failures, however, occur only under certain specific conditions. These failures may occur as the result of a unique sequence of instructions or as the result of a particular instruction being followed by a specific delay in time. Machine failures may occur only at a particular machine duty cycle, or at a particular pulse repetition rate.

**PROGRAMMING TECHNIQUES.** The function of a maintenance program is to help the computer repairman find and fix failures quickly. In order to do this, a programmer may use several techniques or a combination of techniques when writing a program. Five of the major techniques are listed below.

Start small, as the name implies, starts its operation by checking a small number of key circuits. These circuits are then used to check another small group of circuits. This process uses a continually expanding group of proven circuits to check out other groups of circuits until the total area within the scope of the program is checked. The start small is thorough and well suited to diagnosing catastrophic failures and some types of intermittent and machine state failures.

Start big is the technique used to test the computer while operating in a manner similar to its operational mission. Since the entire computer is under test, the start big would detect certain catastrophic, intermittent, and machine state failures (which would normally escape the attention of a start small program).

Marginal checking is a must in many of our older computer systems. Basically, it is a method of preventive maintenance that simulates component aging by using voltage variations to aggravate current operation. Since component values normally change with age, the marginal check is a valid indication of how soon a component will need replacing. The most widely used method of marginal checking are the variation of AC vacuum-tube filament voltage and the variation of DC supply voltage.

Multiple clue approach may be used to isolate a malfunction. Once an error is detected, a program using multiple clue approach attempts to obtain the same error using varying sequences of instructions. If the error can be detected in a variety of ways, it is only necessary to locate the common conditions in isolating the error.

You will find that certain errors in the computer system are very difficult to analyze. However, it is possible for a maintenance program using the process of elimination to aid you in locating errors. Some circuits cannot be directly checked by the maintenance program, but if we validate each circuit in an area, then we could infer the trouble to the area not checked.

**CAUSES OF SYSTEM FAILURE.** There are many causes of system failures that could be listed; however, we will only list a few. These failures are listed to enable you to recognize the need for maintenance program versatility to the extent that they can detect and isolate a variety of failures.

- Failing circuit cards (opens, shorts, etc.)
- Loss of voltage
- Timing
- Interface problems

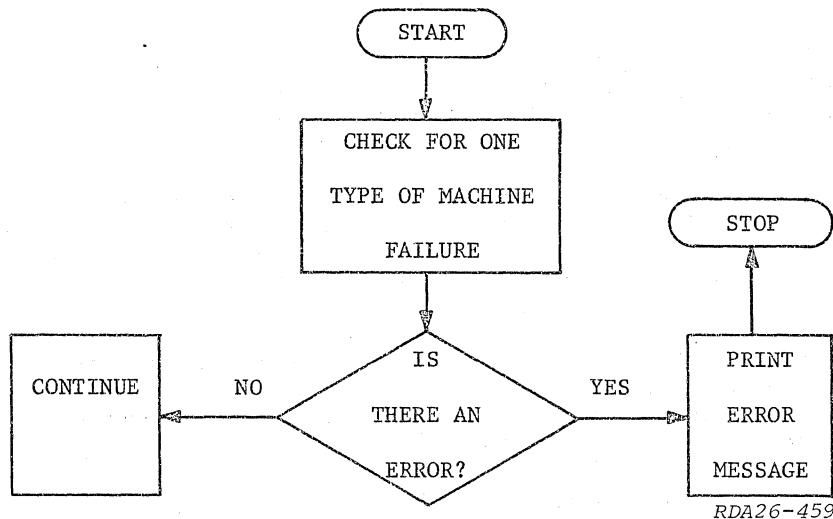
- ⓐ Spurious or continuous outputs
- ⓑ Operator error
- ⓒ Program

Basic Requirements

There are three requirements of a maintenance program to make it valid. First, it must accurately check the equipment, locate all the errors, and give this information to the repairman. Second, it must consistently find any and all errors. Third, it must be comprehensive. It must find errors without regard to type, nature, frequency or location. Maintenance programs must completely test the equipment. Programs must detect any abnormal circuit or system consistently and completely. (The above approach is, of course, idealistic. These three requirements are goals we strive to reach in all maintenance programs. Because programs cannot always be written to find every conceivable problem, maintenance personnel are needed to insure continuous, reliable operation.)

To insure a computer meets the demands of actual operation, the program must exercise the computer at least as much as actual operation. These maintenance programs are known as Reliability Programs because they determine how reliable the computer is in actual operation.

One other type of checking program was developed to find known machine failures. This was done to confine the error to the smallest possible area. These programs are called Operational Diagnostic Programs. Diagnostics are designed for checking the computer and all of its terminal equipment. A diagnostic program can be split into two parts. The first part checks for an error. The second part gives the results of these checks to the maintenance man. Usually these results are in a printed form. This flow chart represents the basic setup for a diagnostic program.



These two separate maintenance programs had certain disadvantages. The reliability programs could determine an error occurred, but a diagnostic program was needed to determine where it occurred. Too many programs were needed to isolate an error. The reliability programs were not comprehensive and missed many errors. The two types of programs have been combined into a single program that overcomes these problems. This



program is designed to not only locate every possible error, but also to trace any error detected to its location. These programs are what we refer to today by the title "Diagnostic" programs. The word diagnostic is like the word diagnose used in medicine. Diagnose is the step a doctor goes through to determine a patient's ailment or disease. In the computer field, it means to check for errors in either the computer or in its terminal equipment. It also means to isolate a machine failure to a small area of logic. Some of the newest computer system diagnostics can tell you which circuit card has failed. The key idea here is that it checks the computer equipment for errors, and tells you where the error occurred.

SUMMARY. Because of the speed of computers, new methods were developed to help find and fix machine failures. Because these programs helped to maintain the computer they were called maintenance programs. They were separated into two types. The first, a reliability program, checked for errors. The second, a diagnostic, narrowed an error down to a small area. These two types had disadvantages which caused them to be combined into a single program usually called a "diagnostic." To be good, a diagnostic program must accurately check the equipment and consistently and comprehensively locate errors.

#### Exercise 3-1

#### Purpose of Diagnostics

#### INSTRUCTIONS

There are four types of questions in the exercise. Essay, multiple choice, fill in the blank, and true-false. Answer the essay questions with a short paragraph. For the multiple choice question, circle the letter which corresponds to the correct answer. The "fill in the blanks" need one or more words to make a complete sentence. In the true-false questions, circle whether the question is true or false.

1. What major requirement does the Air Force make of its electronic equipment?
2. Why didn't the old repair techniques work in computer repair?
3. The main things we check in determining correct computer operation are:
  - a. Speed of computer operation.
  - b. Accuracy and reliability of computer operation.
  - c. Speed of printer operation.
  - d. Temperature of computer cabinet.
4. The two types of maintenance programs are:
  - a.
  - b.
5. Why were computer programs developed to check the accuracy of the computer operation?

6. What were these older checking programs called?
7. How is locating a computer error similar to what a doctor does in determining what is making you ill?
8. What does a reliability diagnostic program determine about the operational status of a computer?
9. What shortcomings of earlier maintenance programs brought about present-day diagnostics?
10. The computer operational diagnostic program has the purpose of:
  - a. Managing computer input/output transfers.
  - b. Verifying the computer would function under operational loads.
  - c. Determining the operational status of the computer.
  - d. Test the teletype for on line operation.
11. Which operational diagnostic program would be best? One that prints.
  - a. X instruction failed.
  - b. Replace circuit cards 1, 2, 4, 5.
  - c. IC 36 circuit card 4 failed.
  - d. X instruction failed, check logic pages 6, 8, 10. Check signals A3, B6, A2.
12. Why is it better to have a diagnostic which prints as much information as possible?
13. Which of the older maintenance programs was used to isolate an error to a small area?
  - a. Reliability programs
  - b. Consistent programs
  - c. Main frame programs
  - d. Diagnostic programs
14. The older diagnostic programs were designed to exercise the computer to the expected operational levels and detect all errors. (True) (False)

15. The two types of diagnostic programs which were previously used in computer maintenance to detect and isolate computer malfunctions are:

- a.
- b.

16. Select all of the following which are requirements of a valid maintenance program.

- a. It must be consistent
- b. It must test only the I/O portions of the computer
- c. It must require the use of the standby equipment
- d. It must measure the power fluctuations of the main power unit
- e. It must locate all errors regardless of location or type
- f. It must be comprehensive

#### COMPUTER SYSTEM TROUBLESHOOTING

The general maintenance function performed by most 305XXs fall in the category of preventive maintenance routines and the repairing or replacing of electronic components. Preventive maintenance is the maintenance of a computer system which attempts to keep equipment in top operating condition and to preclude failures during production runs. Preventive maintenance inspections (PMIs) are scheduled maintenance actions which are utilized to insure that the end item is operating within specified tolerances. Some of the tasks performed are service, inspections, operational/performance checks, etc.

Even though preventive maintenance is an important task, we will devote the rest of this chapter to the repair or isolation of faulty computer circuits.

#### Troubleshooting Techniques

As a computer repair technician, you will use diagnostic programs to quickly locate failures, and restore correct machine operation. You will learn the different ways a computer can fail, and recognize each type of error.

Before tracing a logic error, we need to be aware that some apparent machine failures can be caused by improper computer operation. A computer operator can cause apparent errors by setting wrong switches. A computer is extremely fast, but it cannot reason. A computer does what it is told. Apparent errors also result if a programmer makes mistakes in writing or coding the program. When the program and the operation of the program are both correct, the problem is a machine failure.

Locating the exact circuit failure requires a knowledge of troubleshooting techniques. These techniques will give the basis to theoretically locate and trace errors to the failing circuit. The job of finding where a machine failure exists is made up of three main parts.

1. Running the program to observe the error indications.
2. Asking questions about the error indications and program operation that limit the problem to a small area of logic.

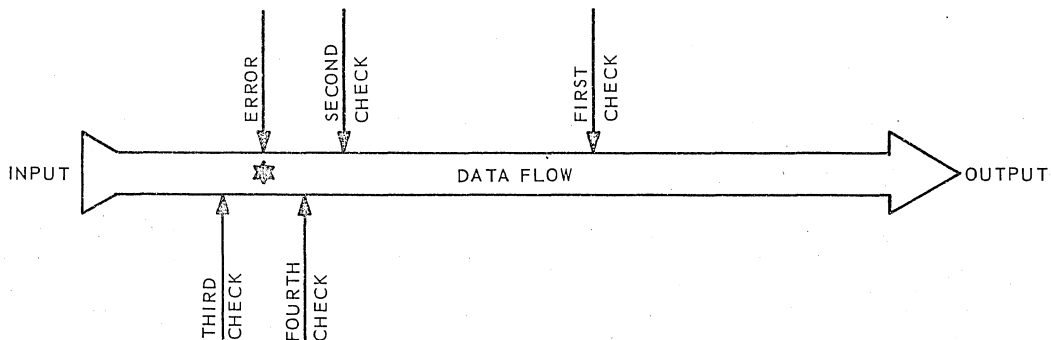
3. Checking for the correct signal in the suspected circuit until the failure is found.

We will examine each of these parts, one at a time, to see what each involves.

Part One: Running the Program. When you troubleshoot, you should always compare what the broken machine does with what a working machine does. The actions of a correctly operating computer are given for each instruction in the logic and timing charts. You will start troubleshooting the COM-TRAN 10 by loading a program into the computer. You will then need to check for correct operation. After you have observed correct machine operation, the instructor will insert an error. Run the program in program mode to see what goes wrong. Check all used registers and memory locations for their correct values after the program stops. Run the program in instruction mode checking again for error indications. Keep track of each error indication. Now run the program in acquisition/execution mode to see what error indications occur. Finally, run the program in distribution mode and check the operation of each DPA and DP pulse for error indications.

Part Two: Questioning. The questions you ask will be different for each program and error. Your ability to ask questions will develop as your troubleshooting experience grows. The questions you ask will be about the error indications you saw from part one. What is the program doing that it should not do? Which instruction is failing? Where is the instruction failing? In acquisition or execution? In which pulse do you see the error indication? In what instruction do you see the first error indication? The loss of what signals could cause this error indication? What do the failing instructions have in common? The answers to these questions should give us a good idea of where to start tracing logic.

Part Three: Error Tracing. The tracing of errors or breakdowns in any system can be approached from three directions. You can start at the input and work to the output or you can start at the output and work back to the input. The final approach is to start in the center. When you start in the center and all signals are good, you know that the error is between you and the output. You continue to divide the remaining distance the signal travels until you find the error. This is the fastest method. Here are a few examples of its use. We use a straight line to represent the signal flow.



Example #1

RDA26-460

Our first check shows an error in the signals. This means the error is upstream. The second check also finds errors, so we again divide the distance in half and do a third check. Finding the correct signals, we know the problem is toward the point where we made our second check. This has narrowed the error location down to a small area in just three checks. After the fourth check we have narrowed the area of the error to between the points where we made the third and fourth check. If you find no error using this method, the problem is upstream in the data flow.

When you make your checks with the oscilloscope, there are a few things you need to keep in mind. Where should you trigger the scope to see the signals? How do you get the computer to loop on an instruction, so you can see the signals. It is best to have the scope trigger on the lowest frequency. When you are looking at an instruction, you trigger on the instruction signal. When you are ready to trace signals, you first decide which signal has the lowest frequency. Set your scope to trigger on this signal. Set the computer so it loops through the instruction. Do this with an unconditional branch instruction (BUN). The program below shows how to examine an instruction:

Memory locations	Mnemonic
00,01	LDA 10
02,03	STA 10 Error found in this location
04,05	BST 04

To look at the signals generated during the STA 10 instruction, alter the program so it will loop on that instruction. Do this by placing a BUN instruction immediately following the failing instruction. The program will continue to branch back to that instruction. Examine the following program.

Memory locations	Mnemonic
00,01	LDA 10
02,03	STA 10
04,05	BUN 02 Branches back to the STA instruction.

To examine the LDA 10 instruction, use the following program.

Memory locations	Mnemonic
00,01	LDA 10
02,03	BUN 00 Branches back to location 00 and the LDA 10 instruction.

#### Troubleshooting Example One

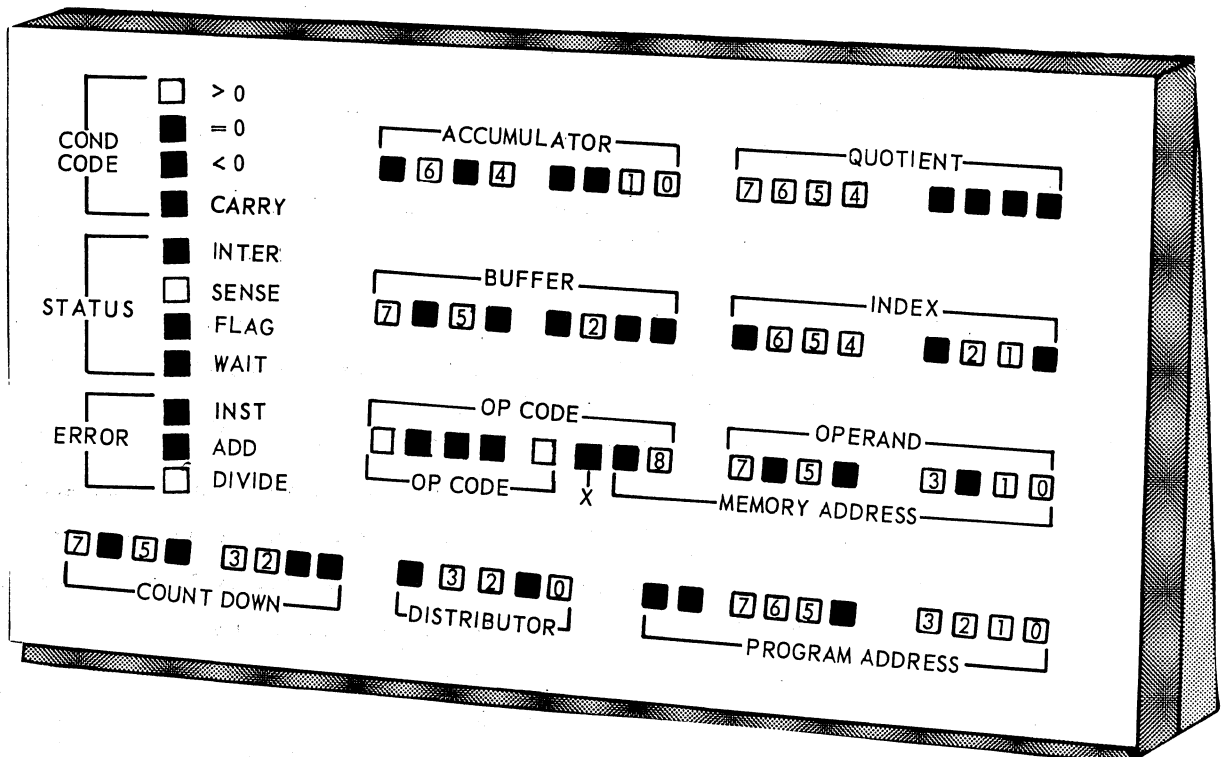
Let's look at the first sample troubleshooting problem. Refer to KDA-3033, Exercise 1-6, page 18 and fill in appropriate blanks.

Memory locations	Hex Code	Mnemonic	Comments
00,01	21 01	LDA 101	Brings first value to Accumulator
02,03	61 02	ADD 102	Add second value to Accumulator
04,05	49 03	STA 103	Store sum in memory location 103
06,07	98 06	BST 06	Stop at location 06
c(101)=5			
c(102)=5			
c(103)=0			

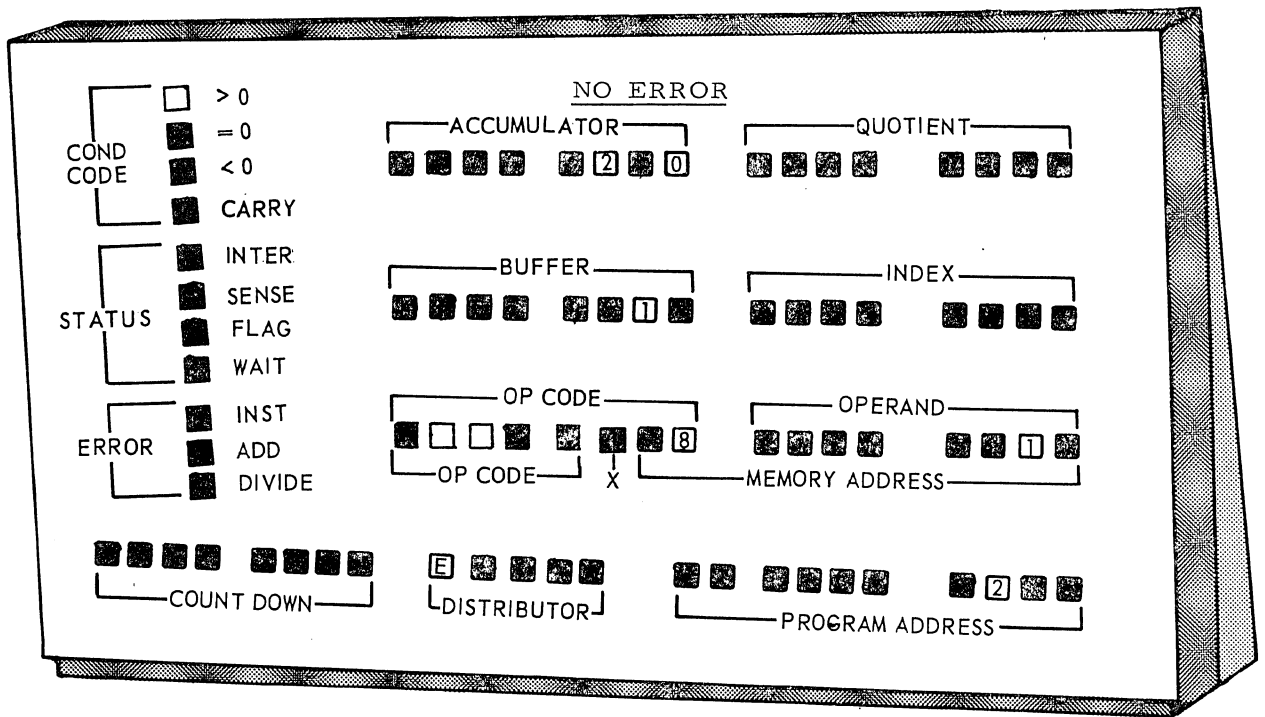
This program should be carefully loaded into the computer and checked to be sure it runs correctly. Memory location 103 will contain the sum of the two numbers after the program stops. You will need to clear this location each time you run the program. You cannot be sure the correct sum was stored unless this step is completed.

Before the instructor puts an error into the computer, be sure you know what the computer does without an error. When sure of the correct operation, have the instructor put an error in the computer. You now begin part one of troubleshooting. The purpose of part one is to find the error indications. Run the program in program mode and check to see if the correct value has been stored in memory location 103 when the program stops. We find 00, which is an error. Go to instruction mode and step through the program one instruction at a time. Remember that in instruction mode, the first time we press the start switch, the first instruction is executed and the second instruction is acquired. This book shows the display lights as we step through the program. The display at the top of each page will show the correct values of each register; the display at the bottom of each page shows the error indications. An example figure shown below will give you an idea how the display panel will be presented after each instruction has been executed. The darkened squares represent binary zeros (lights off). The binary counts on the display panel for the following registers and indicators are:

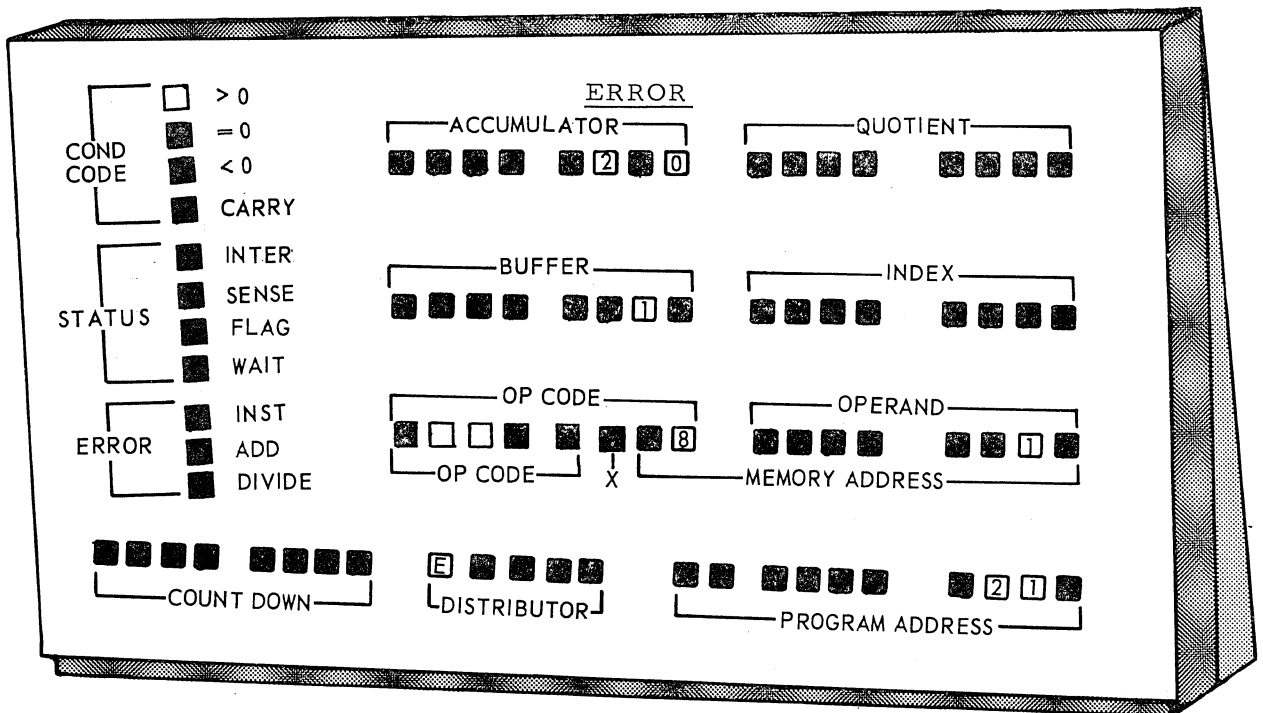
Accumulator	53	Count Down	AC
Quotient	F0	Distributor	0D
Buffer	A4	Program Address	EF
Index	76	Cond Code	>0
Op Code	89	Status	Sense
Operand	AB	Error	Divide

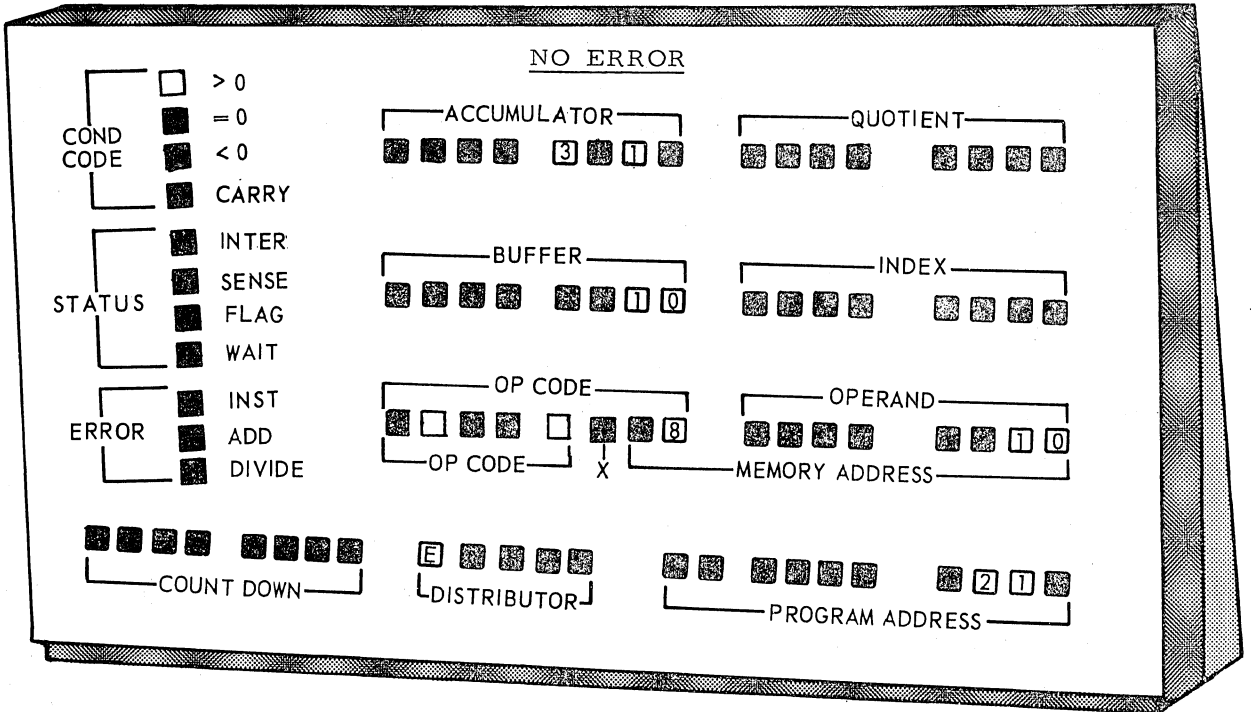


RDA26-461

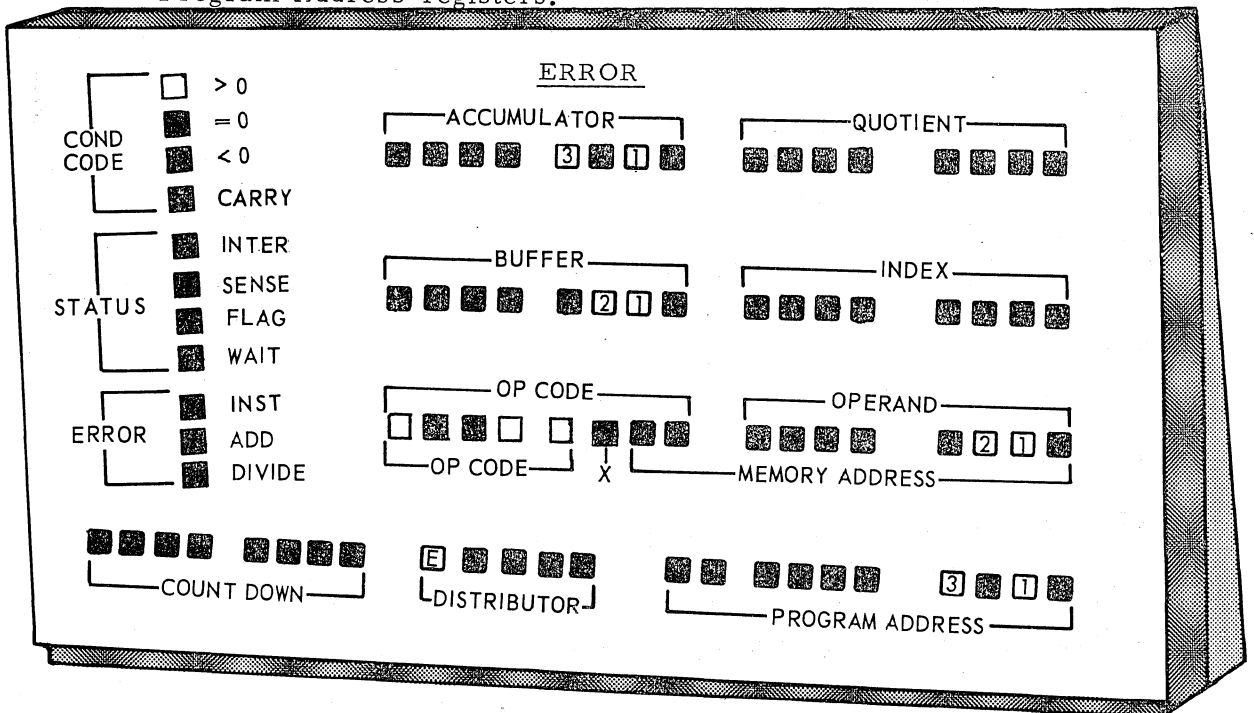


The display panels shows the registers after execution of the LDA 101 instruction. The first detectable error is shown below. The Program Address Register should contain 004 and not 006. The Op Code Register now contains the code for an ADD instruction.

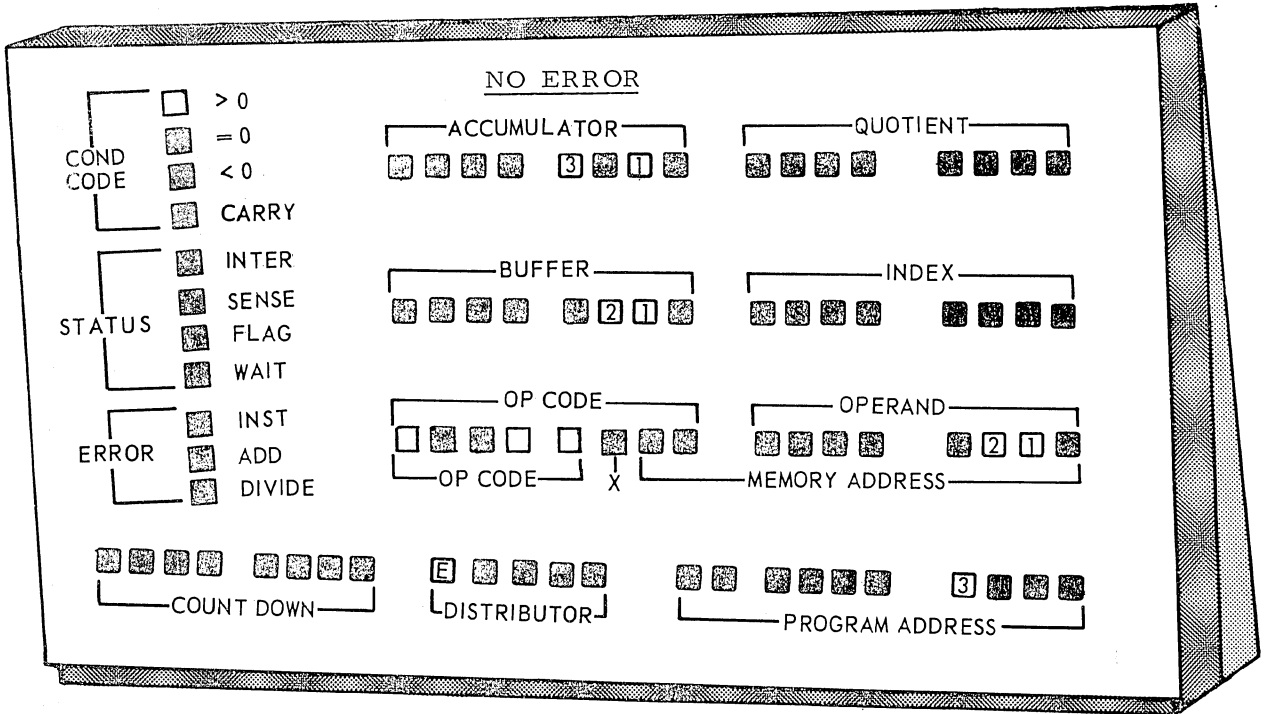




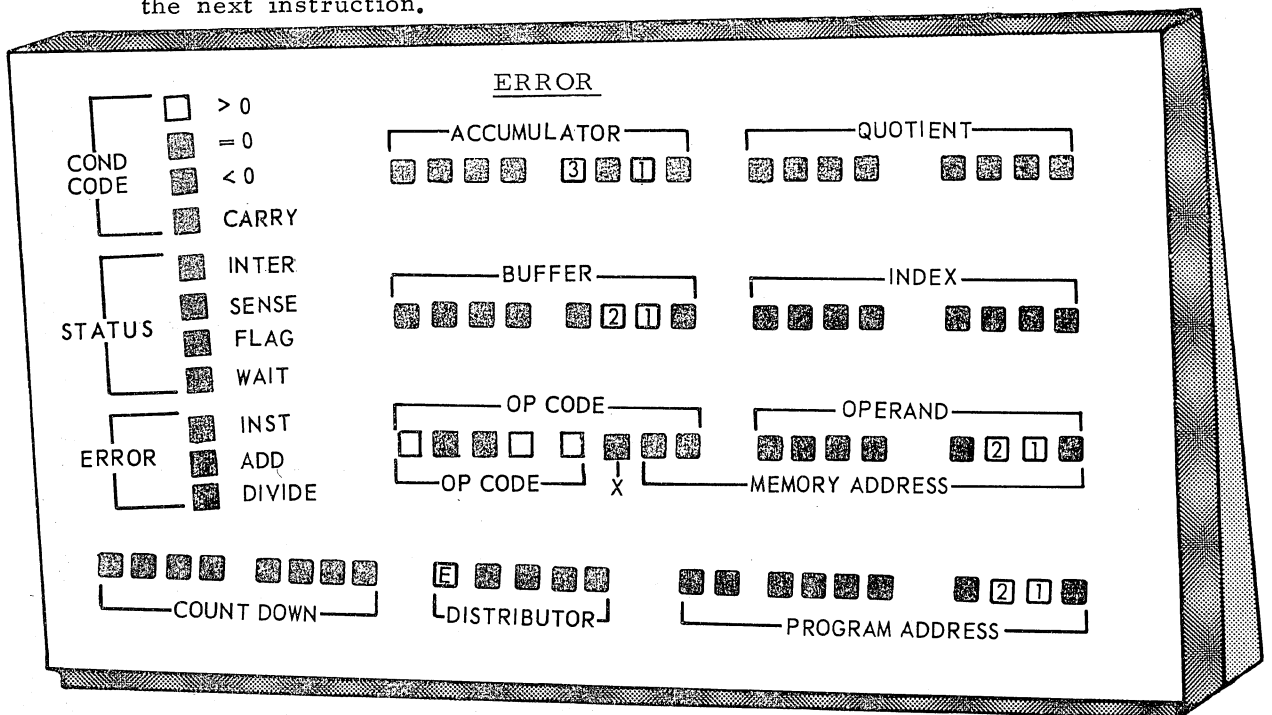
The START pushbutton has been pushed for the fourth time and the third instruction has been executed. The fourth instruction has been acquired and is in the Op Code register. Note the error indications in Buffer, Op Code register, Operand, and Program Address registers.

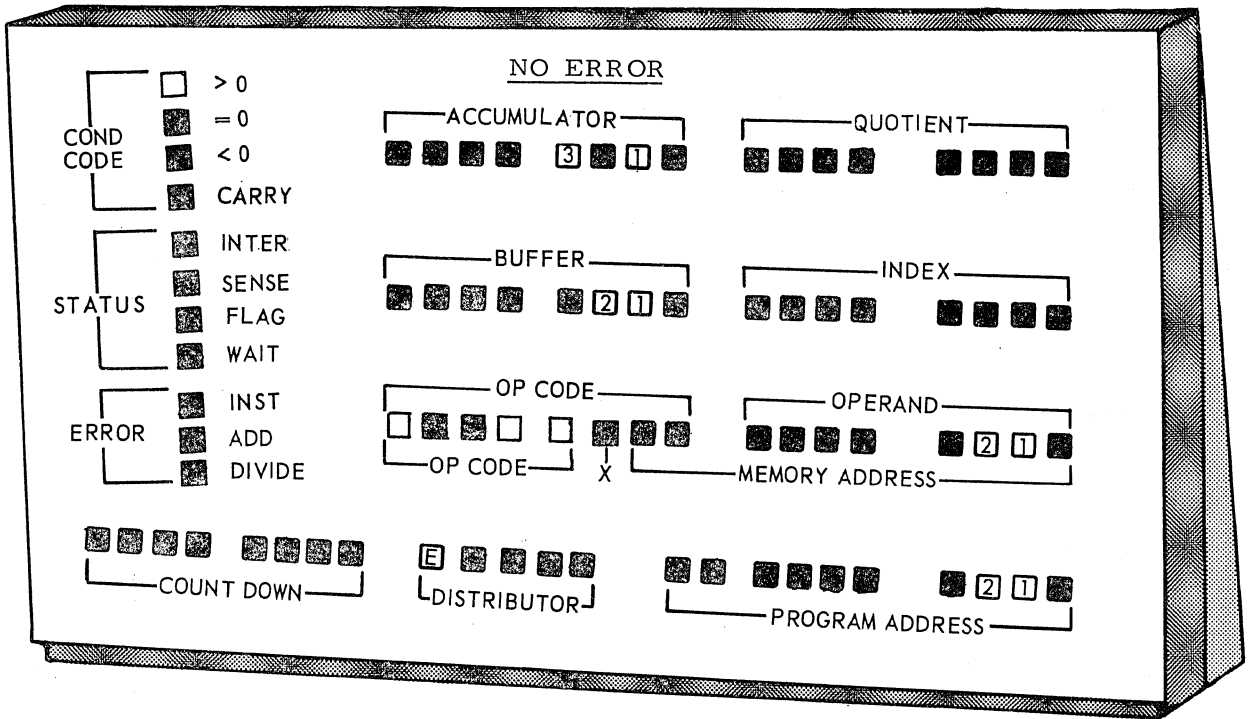




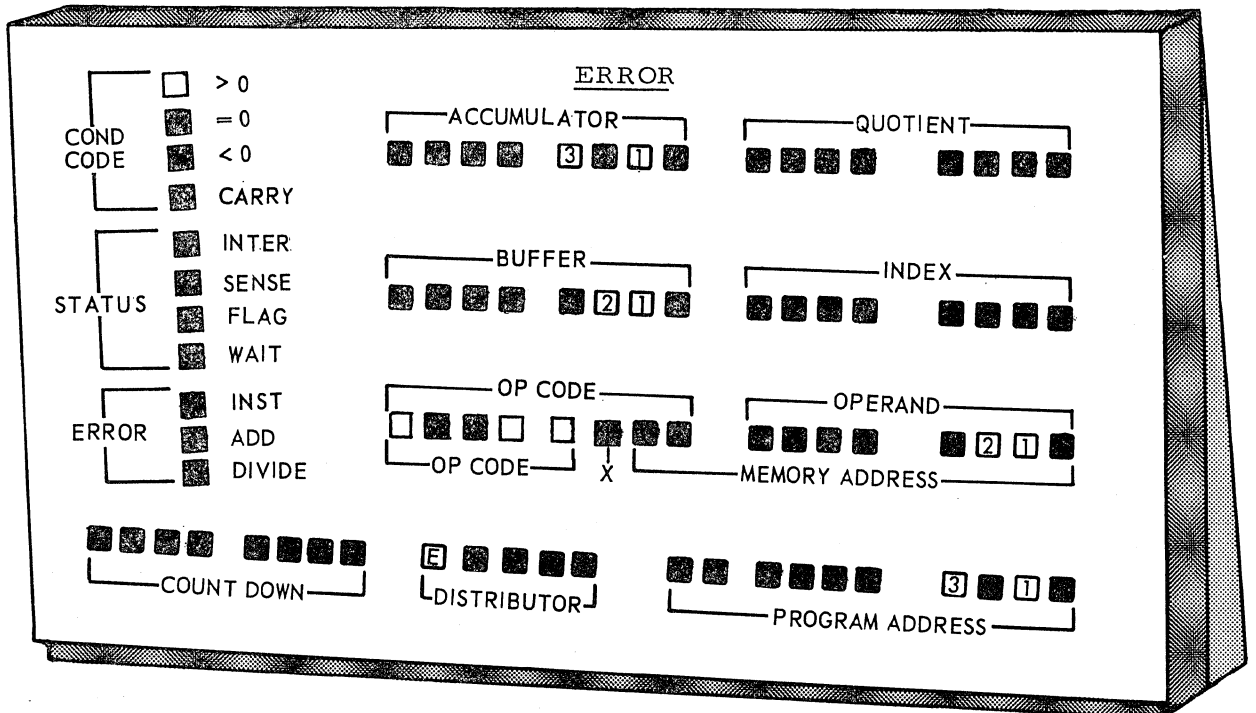


After execution of the fourth instruction the only detectable difference in register contents is the Program Address register. The display panel below shows the error results and it must be remembered that the BST was executed and then acquired for the next instruction.



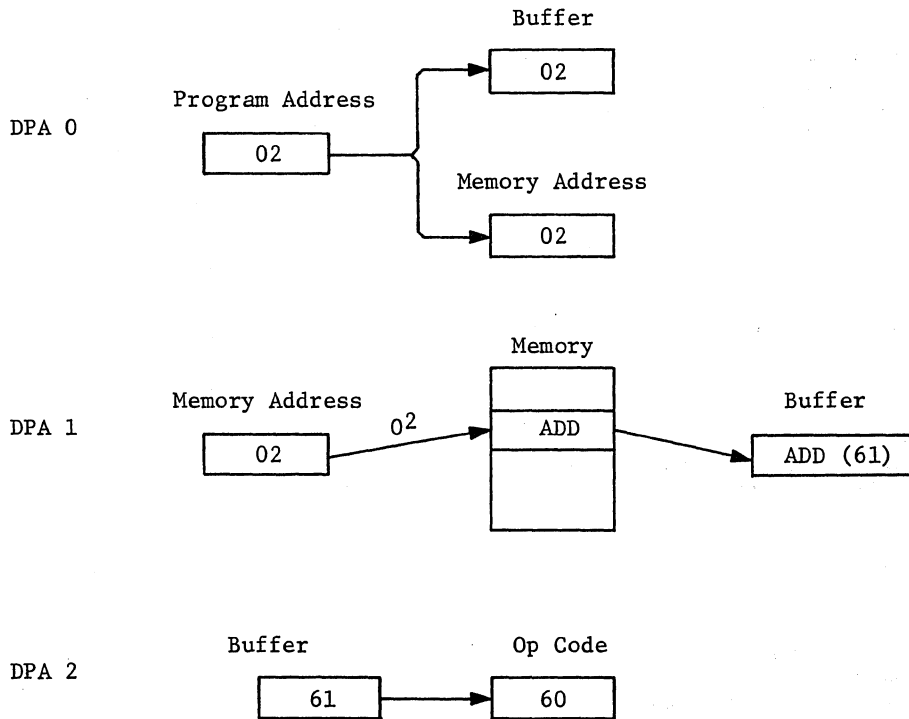


When the BST instruction was executed in the good machine the only detectable change was the change in the Program Address back to 006. Notice that the machine with the error advanced to 00A which is another error indication.

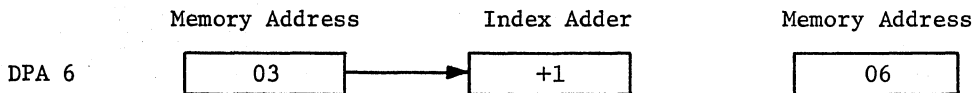
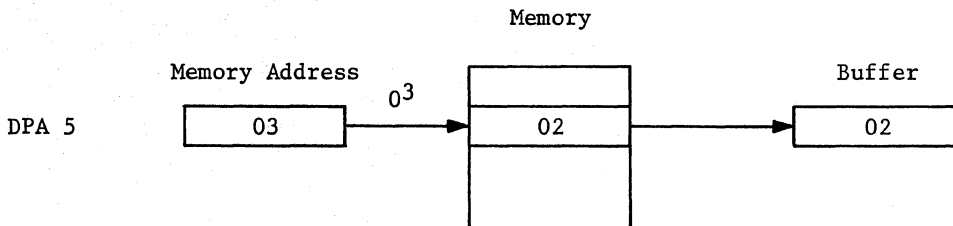
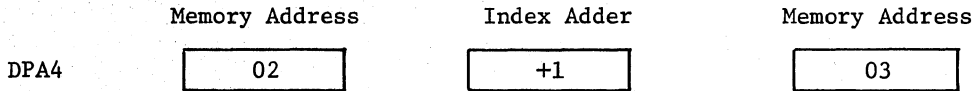


There are enough indications to narrow our search. We are not quite ready to leave part one, but we can start asking ourselves questions. Does the STA 103 instruction do what it should? By finding that location 103 still contains 00 we know that a STA did not work correctly. The answer to our first question is no. Do the LDA and ADD instructions do what they should? We look at the accumulator and see that the correct values were loaded, and the correct sum generated, so the answer to this question is yes. Notice that we are combining steps one and two of troubleshooting here. We continue to look for error indications and to ask questions that will narrow down our search. During what instruction is the first error indication found? We see that the Program Address Register has the wrong value after the Acquisition of the ADD instruction, so the answer is the ADD instruction. The Program Address Register contains 006 instead of 004. How will the incorrect value in the Program Address Register affect the operation of the computer? This will cause the program to go to memory location 006 for the (BST) instead of location 004 for the STA instruction. This is why the correct sum isn't stored into memory location 103.

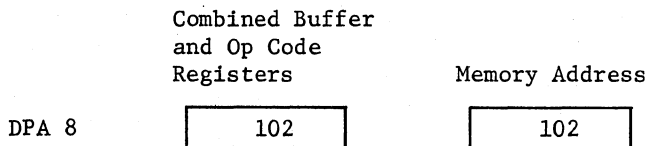
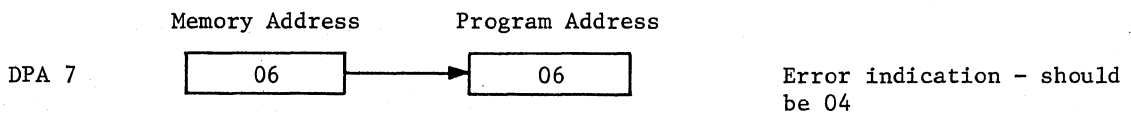
Since the first error indication occurs during the acquisition phase of the ADD instruction, go to the distribution mode and step through the acquisition phase of the ADD instruction.



. DPA 3      Check for invalid instructions



$03$   
 $+ \frac{1}{04}$  Does not equal 06. First error indication.

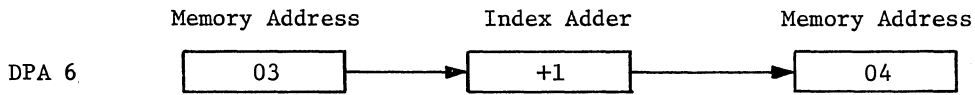


DPA 9 No action - not an indexed instruction.

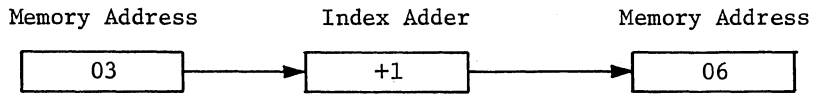
DPA 10 Set distributor to DPA 15.

DPA 15 Set distributor to execution mode.

Going back to DPA 6, the correct action would be



but the action we saw was:



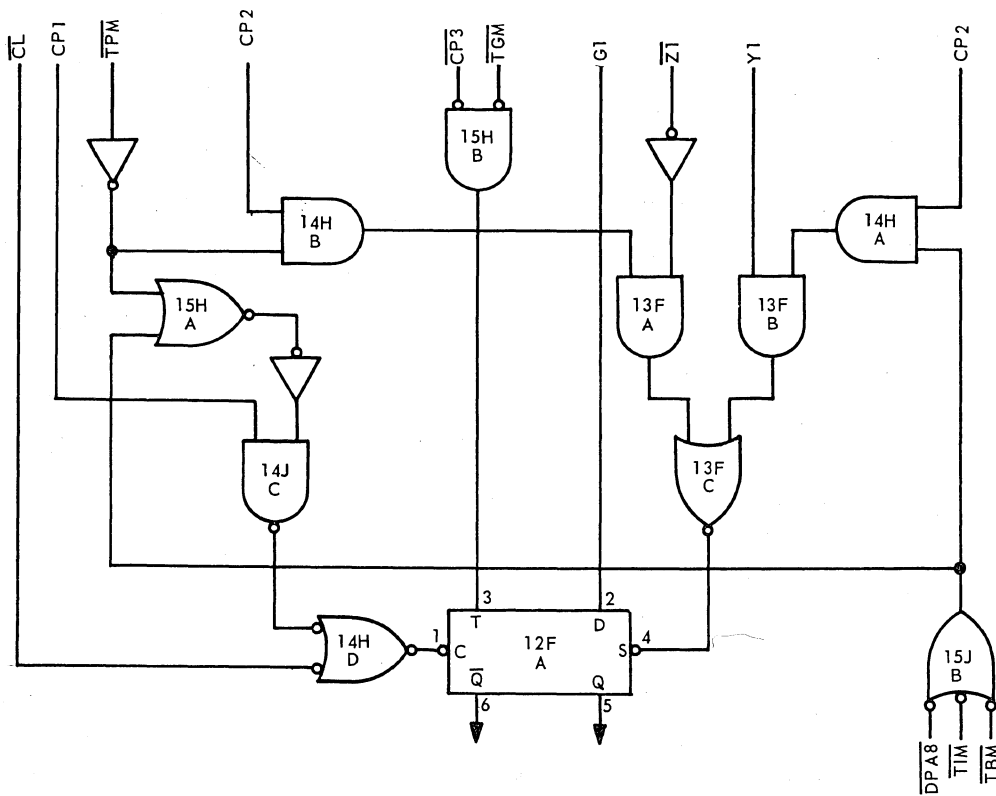
What is the difference between the error condition and the expected action?

CORRECT VALUE

$$\begin{array}{r} 011 \\ + \quad 1 \\ \hline 100 \end{array}$$

ERROR VALUE

$$\begin{array}{r} 011 \\ + \quad 1 \\ \hline 110 \end{array}$$



RDA26-377

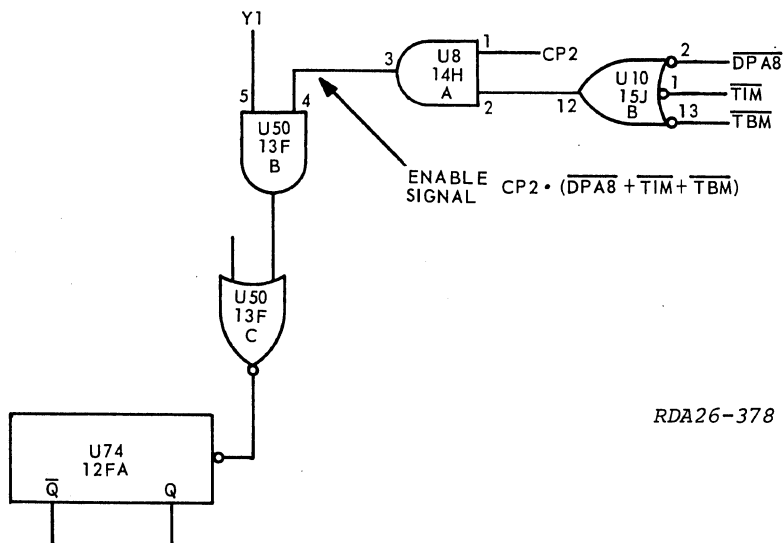
Figure 3-1

Bit M1 is set when it shouldn't be. We now have the indications we need and continue with out questions that will limit the area of our search. To which area of logic should we turn? Since the problem occurs in the Memory Address Register, that would be a good place to start. Sheet 11 of KDA-3034 is the logic for the M-register. Turn to that page of logic before you continue reading.

With which part of the logic on sheet 11 should we concern ourselves? Bit M1 is incorrect. The D flip-flop which generates the M1 signals is a good place to start. This is flip-flop B2 - 12FA on sheet 11 or figure 3-1.

Which of the three inputs to the D flip-flop (B2 - 12FA) will cause the error at DPA6? The input G1 to the flip-flop is from the index adder. The index adder is being used at DPA6, so this input is a possible source of our error. The fact that the LDA instruction worked correctly during DPA4 and DPA6 of the LDA acquisition makes a failure on the G1 input very remote. The Z bus input is used whenever we transfer the Program Address Register into the Memory Address Register. We know that this transfer works because the acquisition phase for the LDA instruction worked correctly.

This leaves the Y bus input. The Y bus Y1 signal enters gate 2D13FB through pin 5. This Y1 signal will be gated through AND-gate 2D13FB by the high enable signal to pin 4 of that same gate. See figure 3-2.

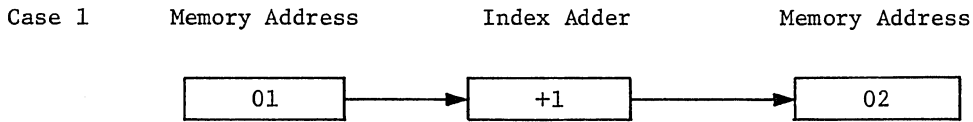


RDA26-378

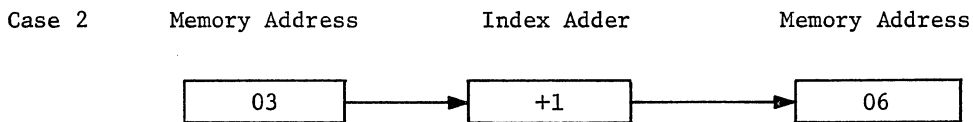
Figure 3-2

This enable signal to pin 4 is generated by AND-gates E7 14HA and F9 15JB. It is generated by the logic equation CP2 (DPA8+TIM+TBM).

By the questions we have asked, and the error indications from part one, we are reasonably sure the problem exists on the input of the Y1 bus into the Memory Address Register bit position M1. The Buffer and Input Register are the only registers to feed the Y bus. During program execution only the Buffer register is used. The problem is first detectable at DPA6 of the ADD acquisition phase. Reviewing the acquisition phase of the LDA instruction, at DPA6 we have:



...but at DPA6 for the ADD acquisition:



Why isn't the computer doing the second case like it does the first? What is the difference between the two?

Since the Y bus comes from the Buffer register, we need to see what the contents of the Buffer were at the time of each case.

Case 1  
LDA acquisition DPA6

Buffer

01

Memory Address

01

+1

Memory Address

02

Case 2  
ADD acquisition DPA6

Buffer

02

Memory Address

03

+1

Memory Address

06

Error

We are looking at the Y bus input to the Memory Address Register, which comes from the Buffer Register. In case 2 the Buffer bit B1 is set, and bit M1 of the Memory Address Register doesn't clear like it should.

Going back to logic sheet 11, if the Y1 line is true (bit B1 is set), and we get the enable signal to pin 4 of AND-gate D2 13FB, the value on the Y1 line will be loaded into the M1 flip-flop.

The logic equation for the enable signal to pin 4 is CP2 (DPA8 + TIM + TBM). At the ADD acquisition DPA6, we are not at DPA8, we are not doing a Buffer to Memory Address Register transfer (TBM), and we are not doing a manual input to the Memory Address Register (TIM). There shouldn't be an enable signal present at pin 4 of D2 13FB.

Is there an incorrect enable pulse being generated and applied to pin 4 of AND-gate D2 13FB on sheet 11?

We have confined the problem to a small area of logic. Now we can trace signals with the oscilloscope. Place the oscilloscope on internal trigger. Set the computer to do the acquisition phase of the ADD instruction. Set the distributor to a count of eight (8) hexadecimal and depress the REPEAT switch. Place a probe on pin 3 of AND-gate 14HA (coordinate E7). (See figure 3-3.) Watch the level as we step through the DPA pulses. At DPA8 the enable signal is generated correctly. Move down stream in the data flow and check the enable signal at pin 4 of AND-gate 13FB (coordinate D2) (see figure 3-3). Put the probe on pin 4 of the AND-gate and set the computer to do the ADD acquisition phase. Watch the CRT for the enable level as we step through the DPA pulses. Here the signal is not present.

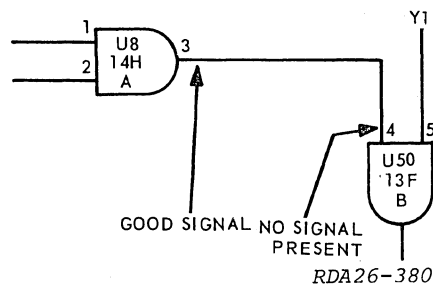


Figure 3-3

What would cause pin 4 of D2 13FB to not receive the signal?

The input to any transistor transistor logic (TTL) integrated circuit will be the same as applying a positive five (+5) volts if input is open or is not connected. Somewhere between pin 3 of E7 14HA and pin 4 of D2 13FB there is an open on the line.

We have located the cause of the machine failure. The enable is always high. Whenever there is a one stored in the Buffer bit 1 position, it will be force fed into the Memory Address Register bit 1 position.

We have traced our sample problem to an open at pin 4 of AND-gate D2 13FB on sheet 11. We have used the following steps:



1. Load the program correctly.
2. Check it to learn what the program caused the machine to do. Use this knowledge to determine error conditions.
3. Insert the error.
4. Run the program in program mode, instruction mode, acquisition/execution mode, and distribution mode looking for error indications.
5. Ask questions about the error indications that narrow the area of search.
6. When the problem is isolated to a small area, use the oscilloscope to trace and find the error.

### REVIEW QUESTIONS 3-2

#### Instructions

There are 12 troubleshooting problems for you to solve. Each gives you the error indications, and four possible causes. Circle the one which will cause the given error indication.

1. Upon execution of the following program, the Accumulator and Memory location 016 contain 1E, and the C-register contains the count 04. What could have caused this error to occur?

Memory Locations	Hex Code	Mnemonics
000, 001	02 7F	LAI 7F
002, 003	68 15	SUB 15
004, 005	13 05	SLL 05
006, 007	48 16	STA 16
008, 009	98 00	BST 00

$$C(015) = 70$$

- a. Pin 1 of AND-gate 4LA open (sheet 4-2D).
  - b. Pin 4 of flip-flop 4MA shorted to ground (sheet 4-6D).
  - c. Pin 4 of flip-flop 4MA shorted to positive 5 volts (sheet 4-6D).
  - d. Pin 1 of AND-gate 1M open (sheet 4-6C).
2. After completion of the following program, the "AQ" register and Memory Location 050 contain 34. The malfunction which probably caused this is:

Memory Locations	Hex Code	Mnemonics
000, 001	20 4E	LDA 4E
002, 003	70 4F	MPY 4F C(4E) = 5 C(4F) = 4
004, 005	58 50	STQ 50
006, 007	98 00	BST 00

- a. Pin 13 of Arithmetic Logic Unit 5E open (sheet 2).
- b. Pin 10 of Arithmetic Logic Unit 5F shorted to ground (sheet 2).

c. Pin 1 of OR-gate 5D open (sheet 2-9E).

d. Pin 4 of Inverter 4D open (sheet 2-8E).

3. During execution of the following program, the computer halts at DPA4 with 002 in the "P"-register. What could have caused this?

Memory Locations	Hex Code	Mnemonics
000, 001	12 05	LXI 05 C(10) = 2
002, 003	24 10	LDA,x 10 C(11) = 1
004, 005	60 11	ADD 11 C(15) = 10
006, 007	4C 12	STA,x 12 C(16) = 2
008, 009	98 00	BST 00

a. Pin 5 of OR-gate 24K open (sheet 17-6E).

b. Pin 4 of Inverter 25F shorted to ground (sheet 18-2D).

c. Pin 12 of NOR-gate latch 19F open (sheet 13-4C).

d. Pin 2 of AND-gate 23H open (sheet 17-3E).

4. If during DPA4 and DPA6 the Memory Address did not increment, a possible cause would be (assume the X-register contains all zeros).

a. Pin 10 U83 11D (sheet 14 2-3B) open.

b. Pin 1 U0 13CA (sheet 14 3-4C) open.

c. Pin 9 U74 12FB (sheet 11 2B) open.

d. Pin 11 U8 14HD (sheet 119C) open.

5. If the E flip-flop did not toggle at all, a likely point to check is (D-register is counting).

a. Pin 17 U154 21E (sheet 18 4-5E).

b. Pin 17 U154 21J (sheet 18 3E).

c. Pin 11 U0 24LD (sheet 18 7E).

d. Pin 3 U8 18JA (sheet 15 3B).

6. At DPA10 you find that the D-register is set to 14. A possible problem could be:

a. Pin 4 U40 3KA (sheet 5 2D) open.

b. Pin 3 U4 2KB (sheet 5 7D) open.

c. Pin 3 U3 23GA (sheet 21-6C) open.

d. Pin 4 U8 2HB (sheet 5 8C) open.

7. You find that only Bits M8 and M9 never get cleared when transferring new data into the M-register, the fault could be:

- a. Pin 1 U2 15HA (sheet 11 8D) open.
- b. Pin 1 U4 14FA (sheet 11 9D) open.
- c. Pin 1 U0 14JA (sheet 11 9D) open.
- d. Pin 8 U8 14HC (sheet 11 9C) open.

8. If the COM-TRAN 10 will acquire an instruction but will not execute that instruction, what is the most probable cause of the malfunction?

- a. Gate U0 24LD pin 11 is grounded (sheet 18-7E).
- b. Integrated circuit U154 21J is inoperative (sheet 18-3E).
- c. Gate U30 3HA pin 8 is grounded (sheet 5-7E).
- d. Pin 3 U451 3BA has a continuous high (sheet 9-6B).

9. After execution of the following program, the Accumulator contains 17 and Memory location 012 contains 1F. What malfunction could have caused the error?

```
000 LDA 10
002 ADD 11
004 STA 12
006 BST 00
```

C(010) = 0A C(011) = 0D

- a. Pin 6 of inverter 6GC open (sheet 1-6D).
- b. Pin 13 of Arithmetic Logic Unit 5E shorted to ground (sheet 2).
- c. Pin 1 of AND-gate 7GA open (sheet 1-5C).
- d. Pin 12 of AND-gate 5JD shorted to positive 5 volts (sheet 1-6C).

10. While attempting to run the following program, the computer halts at DPA4 with 001 in the P-register. What malfunction could cause this?

```
000 LDA 17
002 XOR 18 C(017) = X C(018) = Y
004 BZE 08
006 BST 00
008 BST 01
```

- a. Pin 8 of AND-gate 27GC open (sheet 21-1D).
- b. Pin-7 of OR-gate 26CA open (sheet 24-4C).
- c. Pin 7 of Decoder 21J open (sheet 18-3E).
- d. Pin 7 of OR-gate 26CA shorted to positive 5 volts (sheet 24-4c).

11. After execution of the following program, Memory location 222 contains 70. Which malfunction caused this erroneous answer?

```

200 LXI 020
202 LDA,x 201
204 IOR 050 C(221) = 25
206 STA 222
208 BST 000

```

- a. Pin 3 of AND-gate 29HA open (sheet 23-2D).
- b. Pin 3 of AND-gate 29HA shorted to positive 5 volts (sheet 23-2D).
- c. Pin 1 of AND-gate 27JA open (sheet 22-2D).
- d. Pin 5 of OR-gate 30LB shorted to ground (sheet 20-5E).

12. After completion of the following program, the Q-register and Memory location 012 contain positive 6. Which malfunction listed below would cause this error?

```

000 LAN 010
002 MPY 011
004 STQ 012
006 BST 000
C(10) = 3 C(11) = 2

```

- a. Pin 8 of Decoder 21L open (sheet 17-6E).
- b. Pin 10 of OR-gate 26HC open (sheet 18-8D).
- c. Pin 5 of OR-gate 31JB open (sheet 20-7E).
- d. Pin 16 of Decoder 21L open (sheet 17-7E).

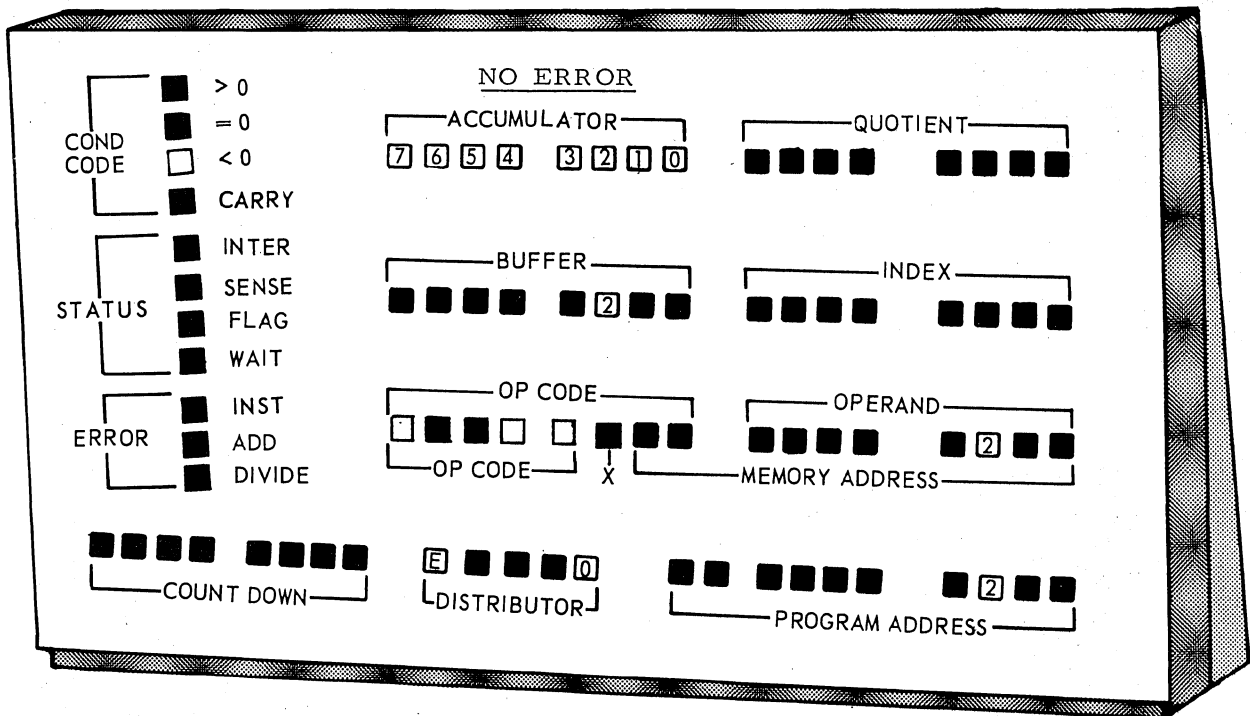
#### Troubleshooting Example Two

Let's locate a second machine failure. Refer to KDA-3033, Exercise 2-14, page 21 and fill in appropriate blanks.

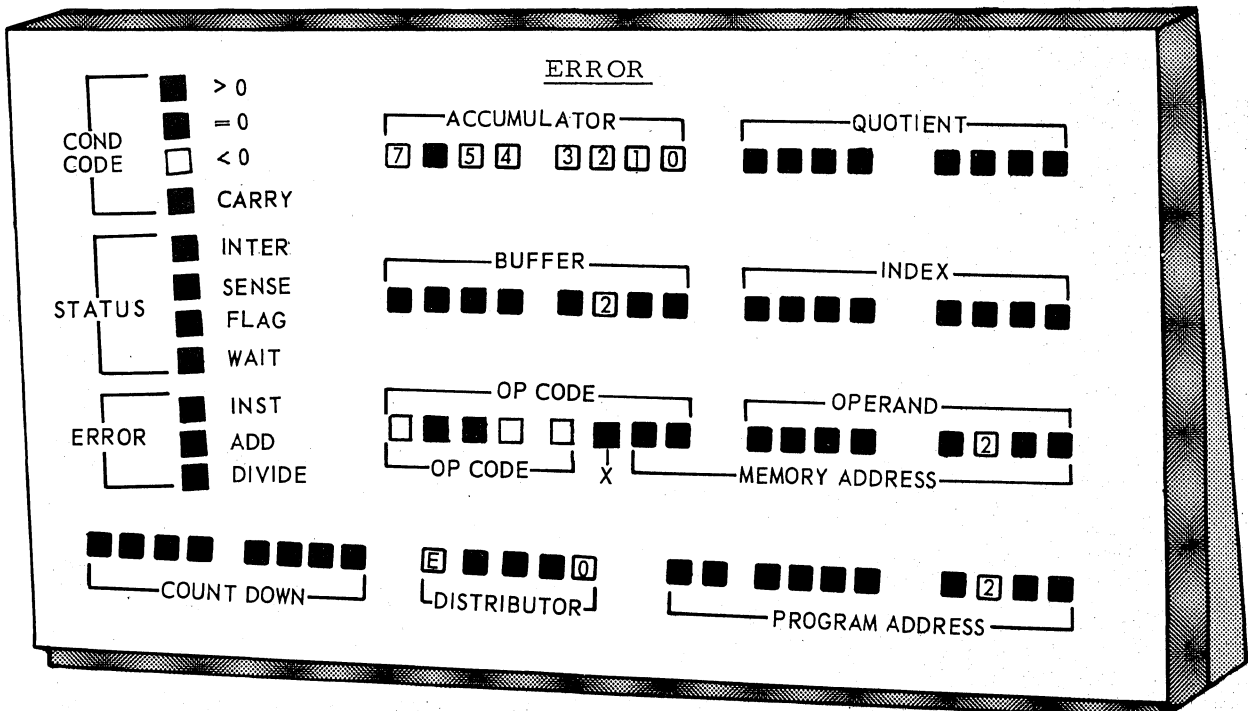
<u>Memory Locations</u>	<u>Hex Code</u>	<u>Mnemonics</u>	<u>Comments</u>
00,01	21 00	LDA 100	Brings value in memory location 100 to the Accumulator
02,03	49 01	STA 101	Stores Accumulator into memory location 101
04,05	98 04	BST 04	Stops at location 04 in Program Register
c(100)	FF		
c(101)	00		

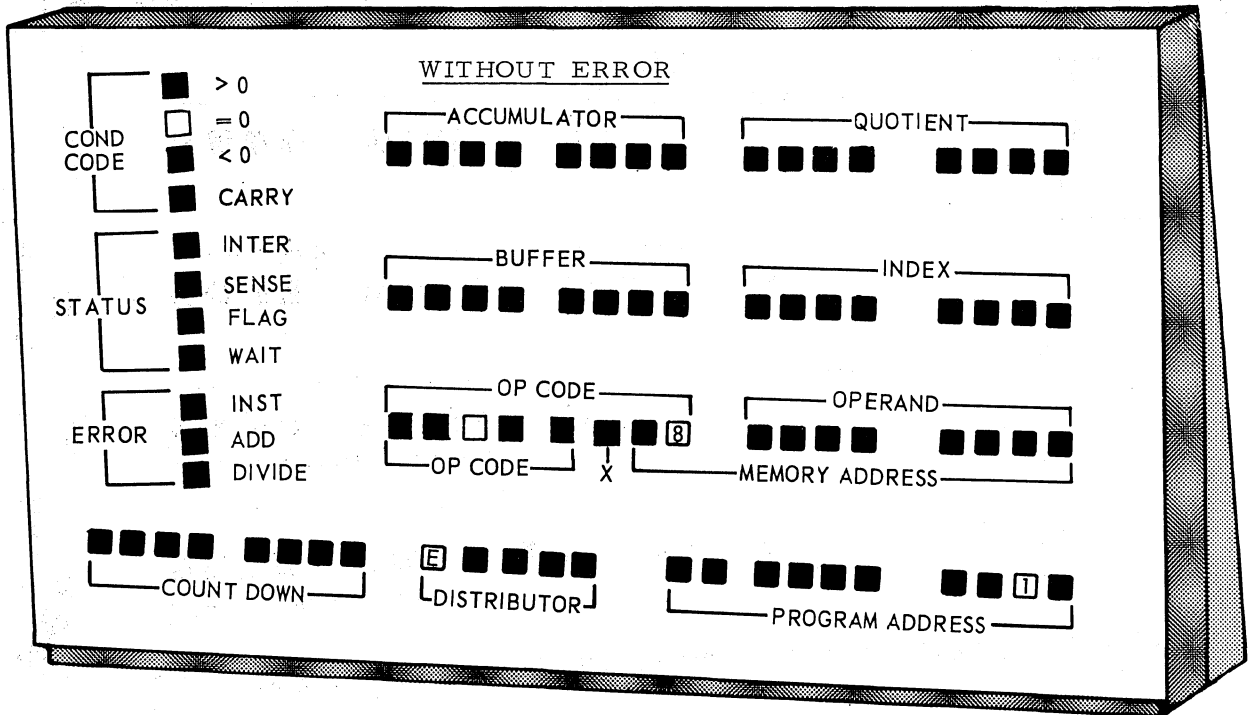
Load this program into the computer and be sure the machine performs it correctly. This program takes the value FF from memory location 100 and stores it into location 101. Memory location 101 must be cleared before repeating the program. This is done to determine if the STA (store accumulator) instruction is being performed correctly. It is important that we know what the program does when working correctly, since we are using the correct operation of the program as our comparison. Call the instructor to place an error in the computer when you are sure of correct operation.

Begin the process of looking for error indications which is the first part of troubleshooting. Set the computer to program mode and run the program. Compare the values in the registers with the correct results previously obtained. The comparison should yield the following:

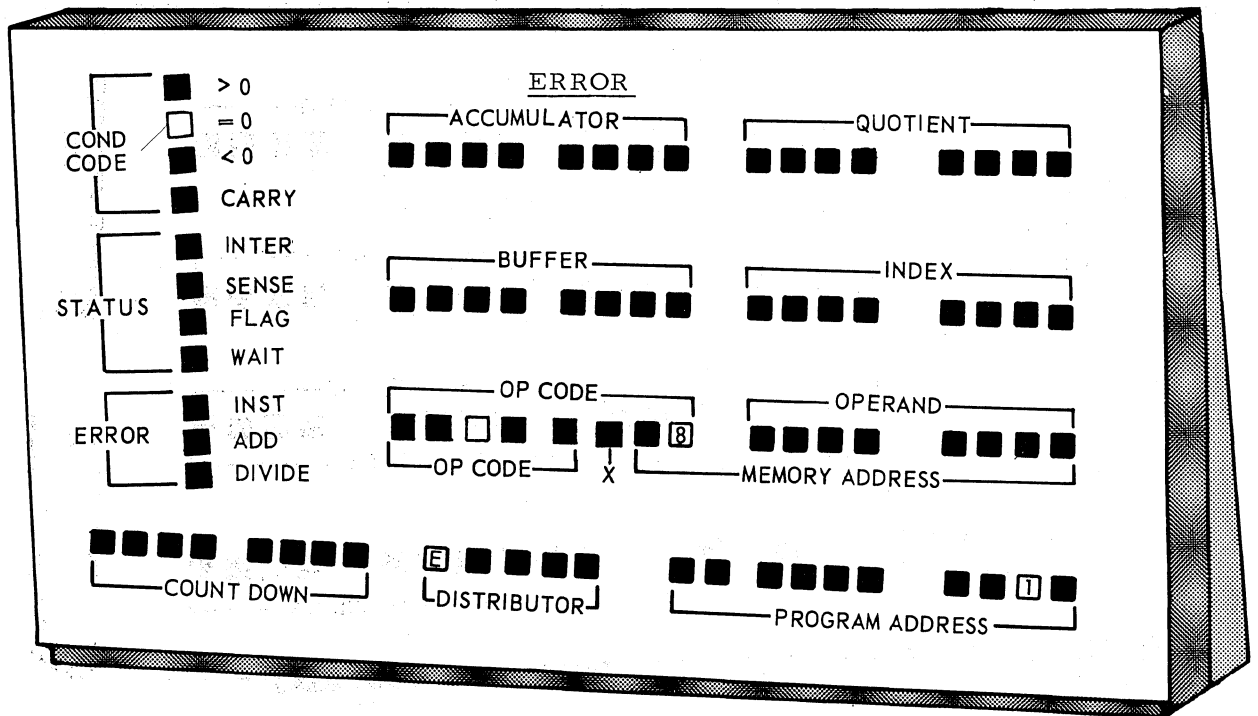


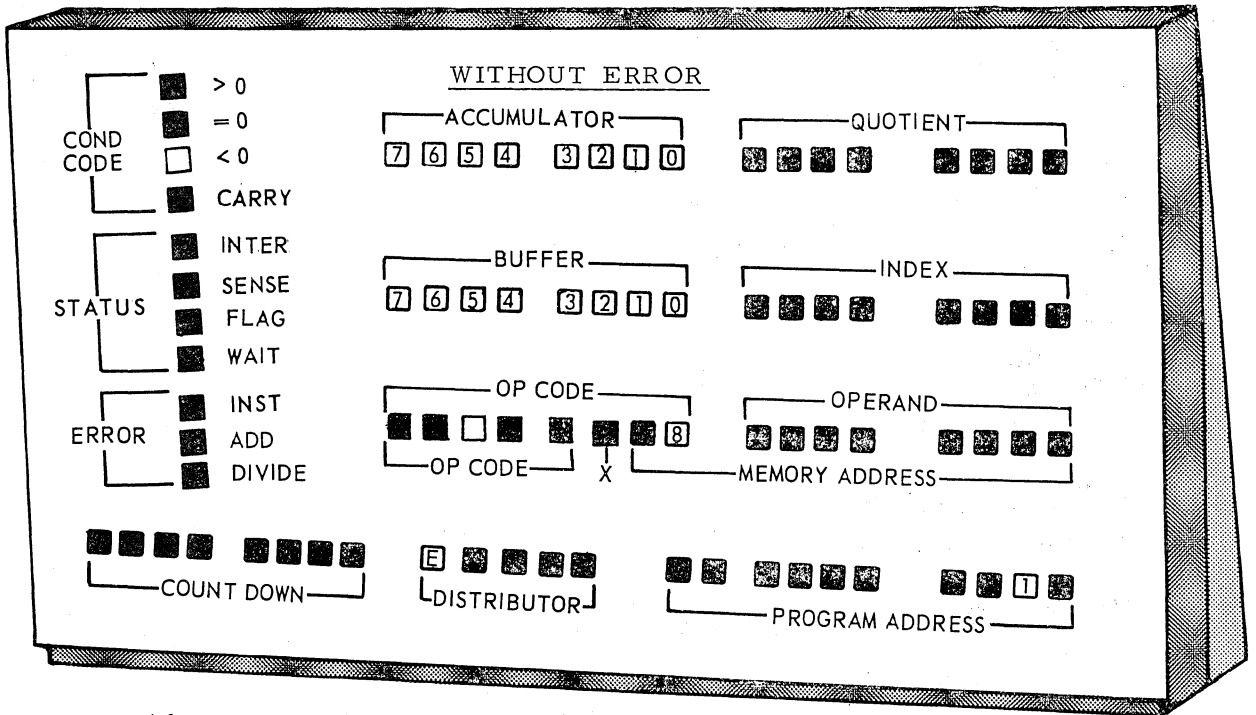
After running the program through we find BF in the accumulator instead of FF. Checking location 101 we find that BF was also stored instead of FF. This is our second error indication. Step through the program in A/E mode to narrow the error further.



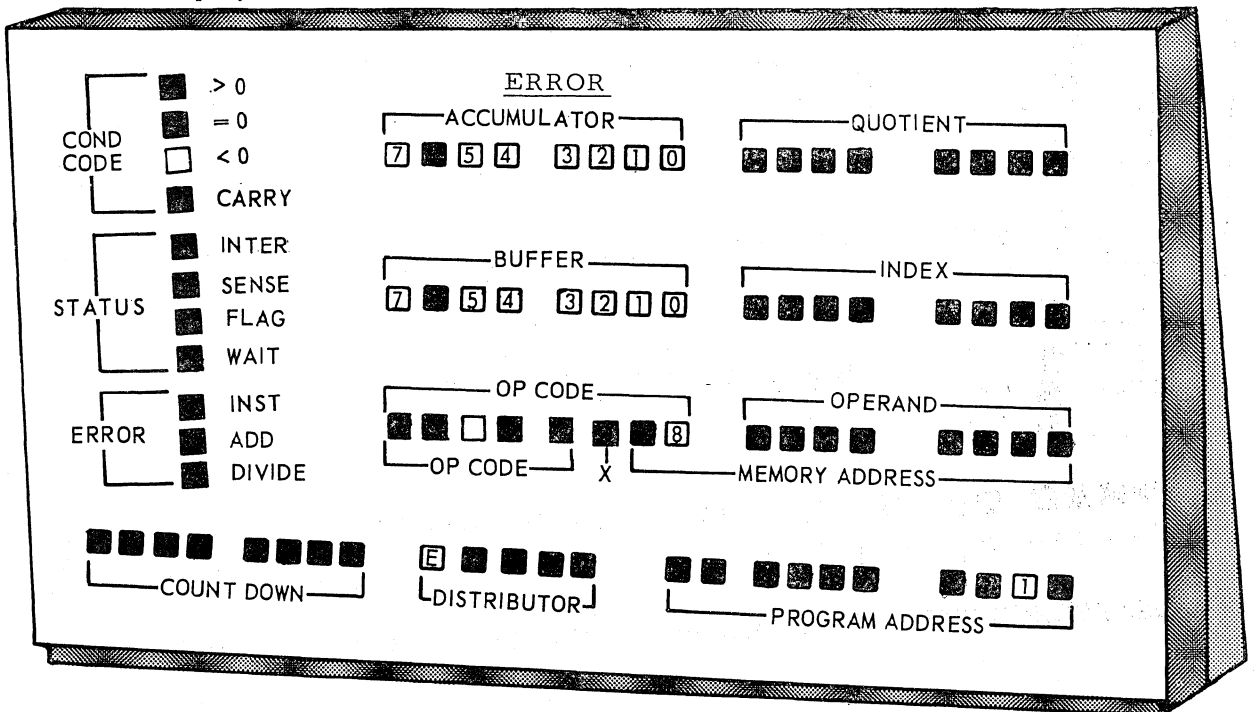


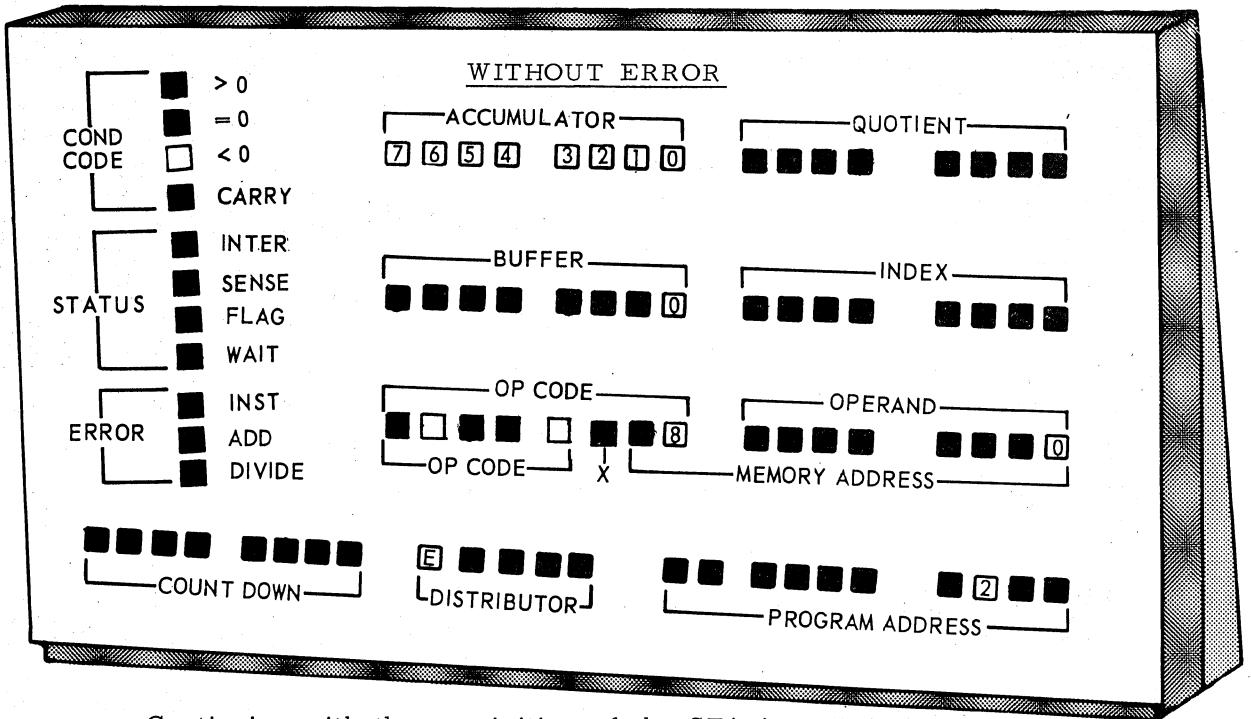
After acquisition of the LDA instruction both displays are identical. We will continue to step through the program and look for errors.



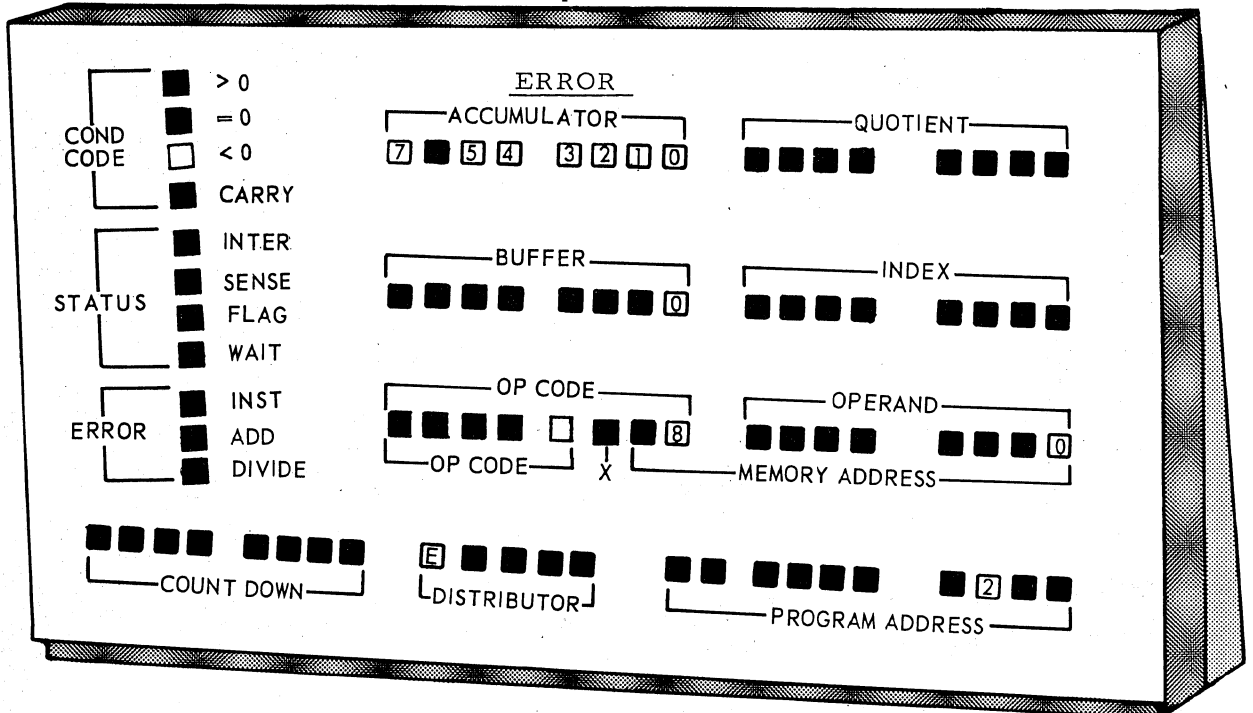


After execution of the LDA we notice that the contents of both the Buffer and Accumulator contain BF instead of FF. It would appear that location 100 was not loaded correctly. Position 6 of the Buffer and Accumulator was dropped. Look closely at the display below that shows these two errors.

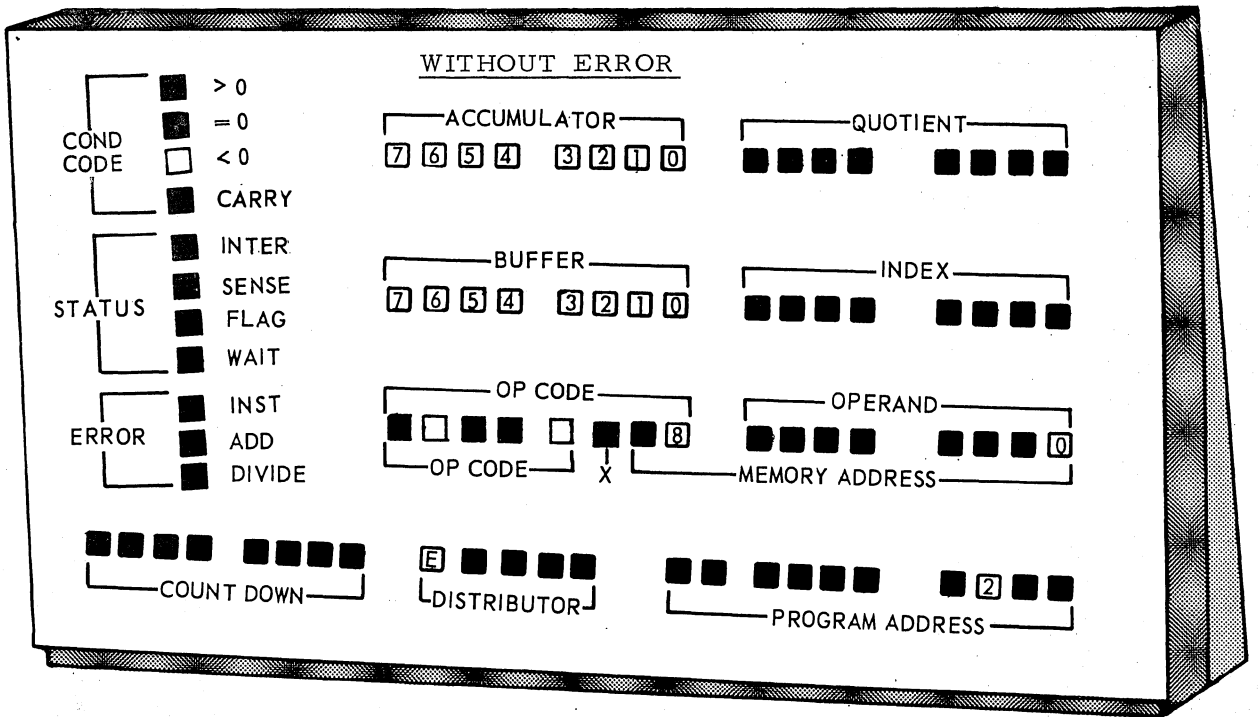




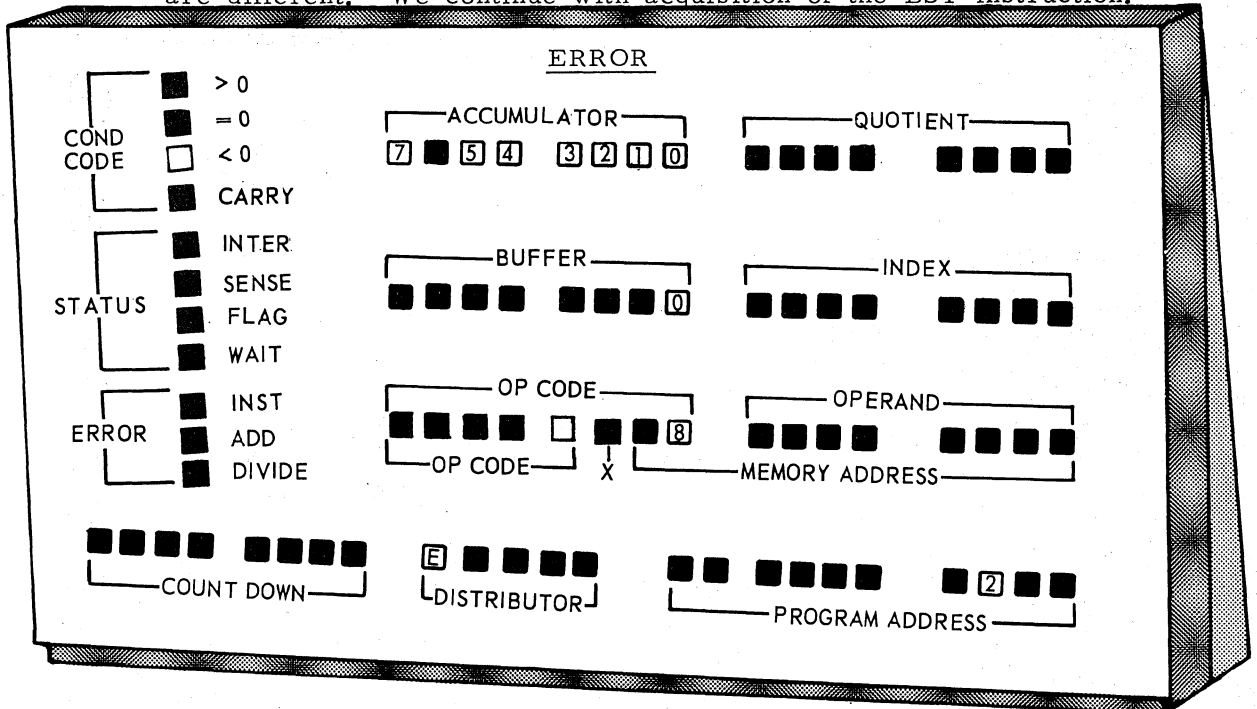
Continuing with the acquisition of the STA instruction we notice another error indication. Instead of acquiring 49 in the Op Code register we have 09 (this is the code for skip on sense). It again appears that we are dropping bit position 6 from memory. We continue with the execution phase.

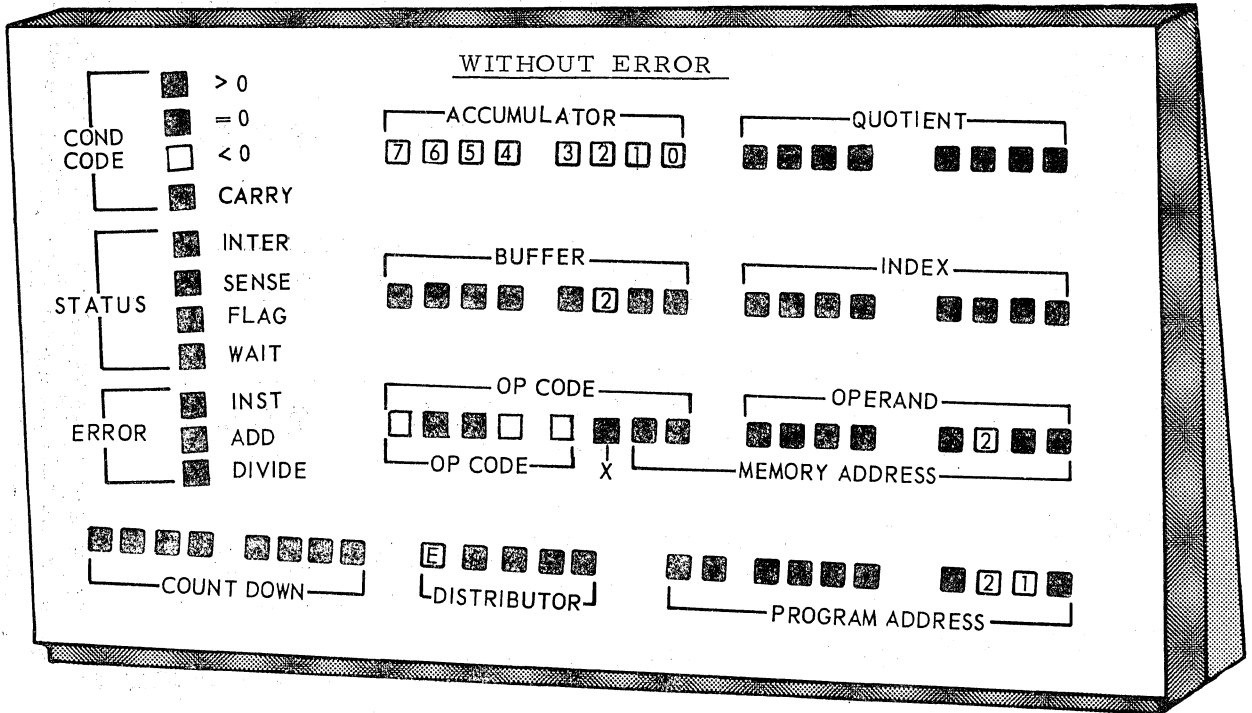




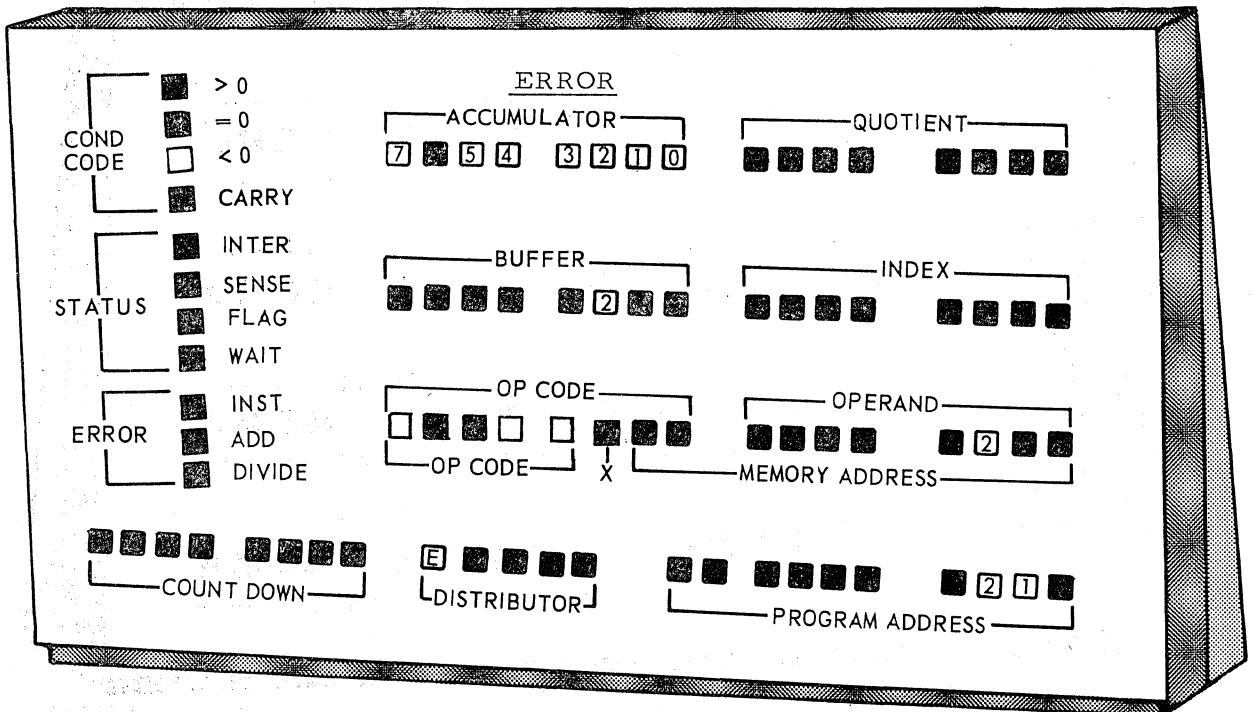


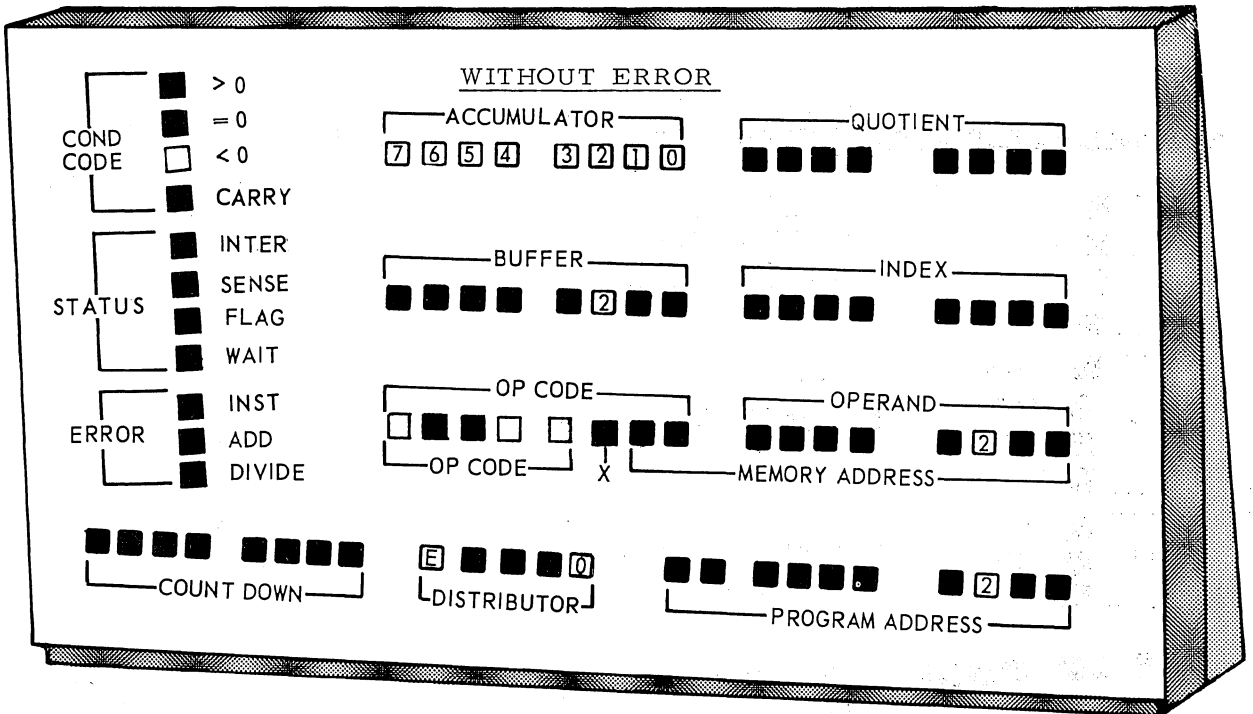
Since the wrong Op Code was executed the accumulator was not stored into location 101 in the machine with the error. Notice that the Memory Address and Program Address are both the same. Only the accumulator, buffer, and Op Code registers are different. We continue with acquisition of the BST instruction.



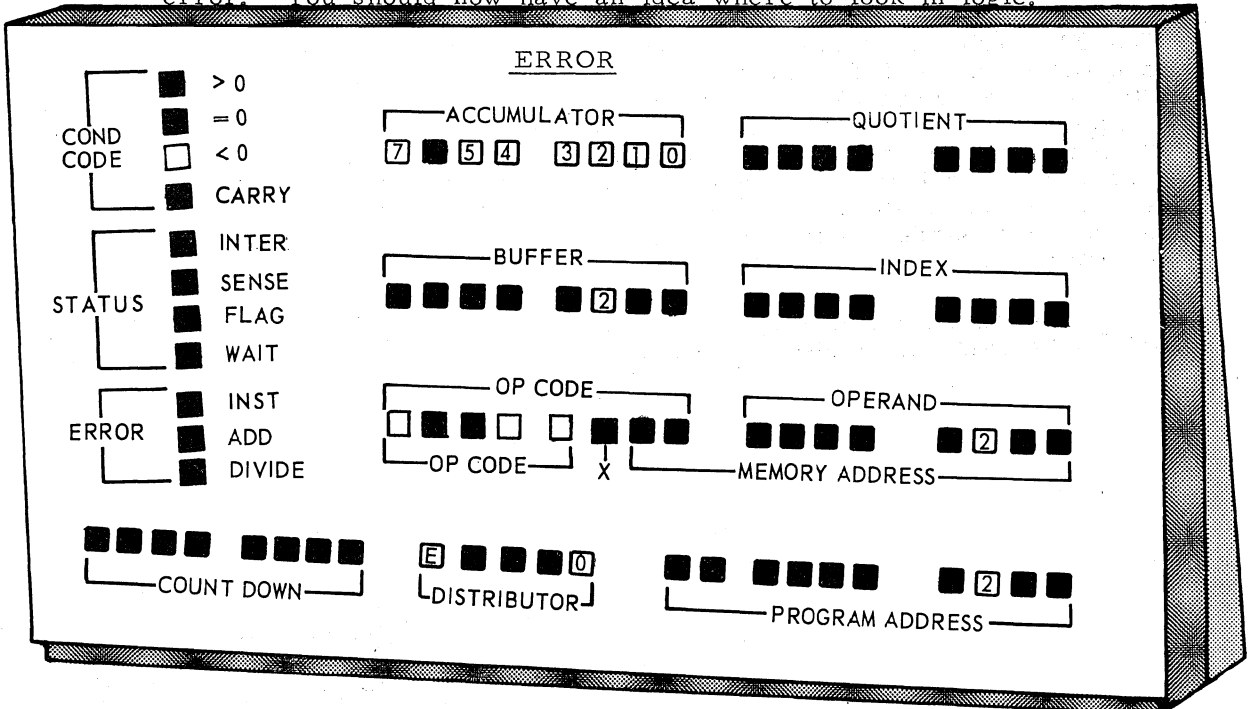


The only register that is in error after acquisition of the BST instruction is the accumulator which still contains BF. This is probably because the code for BST does have a one in position 6. We continue with the execution phase.

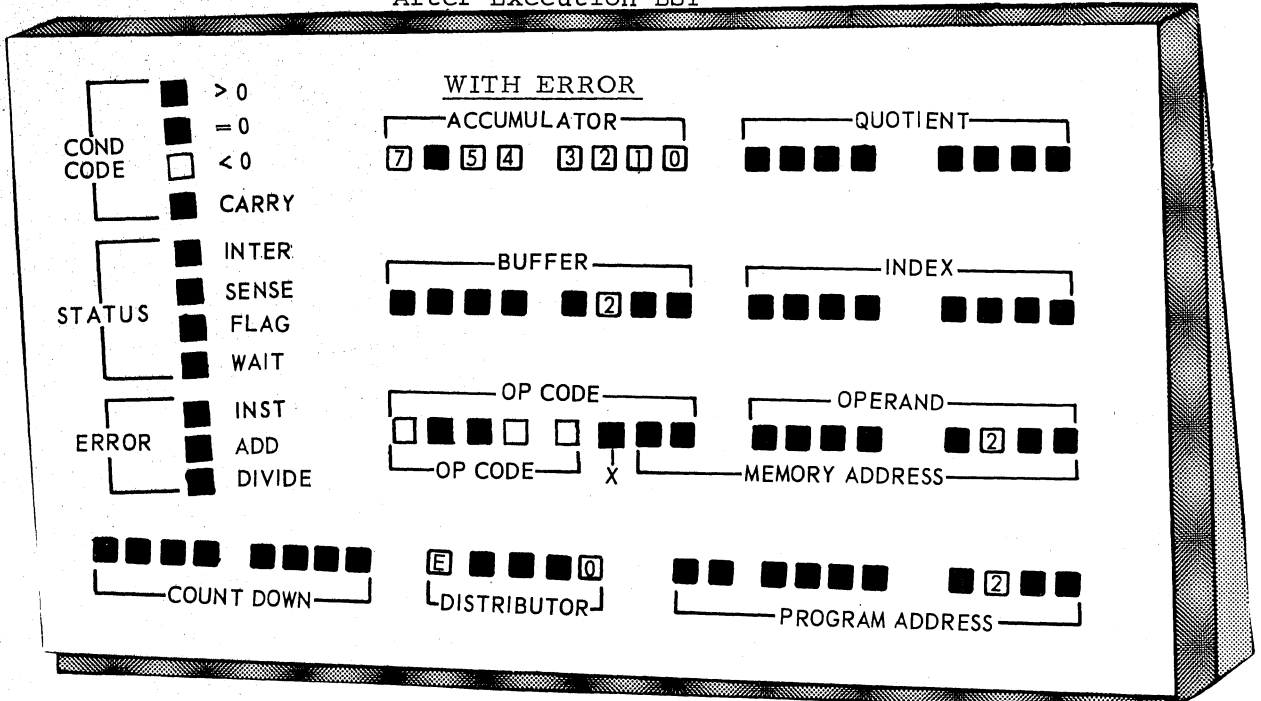




After execution of the BST the only error indication that exists is still the accumulator contents. Remember that incorrect information was loaded into the accumulator (BF instead of FF) and that the store was not performed in the machine with the error. You should now have an idea where to look in logic.



After Execution BST



RDA26-473

Since this program does a store in memory address 101, we need to see what value has been stored. We find 00 which indicates that nothing was stored. This is another error indication. Check memory location 100. Here we find BF. Our program doesn't store anything into memory location. We know there was a value FF in memory location 100 before the error was inserted. This indicates there may be a problem getting the correct value from memory. We find BF in both the Accumulator and Buffer, after the execution of the LDA instruction.

What does the fact that the value BF is in both the Buffer and the Accumulator mean?

The LDA instruction brings a value from memory into the Buffer, and then to the Accumulator. Since the Buffer and Accumulator are the same, the Buffer to Accumulator transfer was correct. We know that BF is what entered the Buffer from memory. This tends to point to the problem as being unable to read correctly from memory.

Do any other error indications point to a possible problem of getting the correct values from memory?

After the acquisition of the STA instruction, which has an OP Code of 49, we find 09 in the OP Code Register. The acquisition phase of any instruction brings that instruction out from memory and places it into the OP Code (S-Register) Register. We know the correct program was in memory when we started, and we have done nothing to change it. It appears that instead of reading 49 from memory location 02, we instead bring out 09.

What do the values 49 and FF have in common with 09 and BF, respectively?

This might be better understood if we compare the binary values of these numbers.

	Bit positions 7 6 5 4 3 2 1 0
49	0 1 0 0 1 0 0 1
09	0 0 0 0 1 0 0 1
FF	1 1 1 1 1 1 1 1
BF	1 0 1 1 1 1 1 1

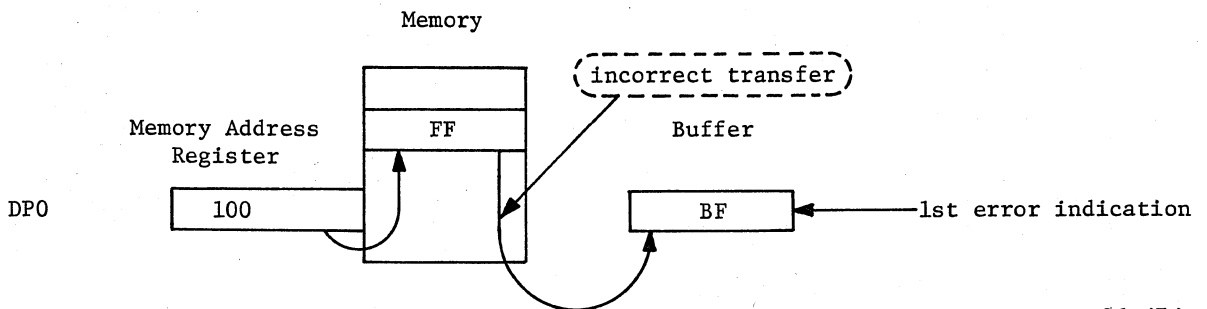
Bit position 6 is changed  
Error indication

The computer should bring out the value FF during the execution of the LDA instruction, but bit position 6 doesn't get set. This causes BF to be loaded instead of FF. During the acquisition of the STA instruction, 49 should be brought from memory to the OP Code Register. Bit position 6 does not set causing 09 to be loaded into the OP Code Register.

Is this problem one of losing bit 6 as we transfer the data from the Memory Module to the Buffer?

This is possibly the problem. Let us review our error indications. We know the program was stored correctly because we checked it after it was loaded and tested.

We know that location 100 contains the value FF, and that the LDA instruction should load the Accumulator with FF. The value BF is being loaded which is incorrect. During acquisition phase of the STA instruction, 09 is loaded into the S-register instead of 49. In both cases bit 6 is dropped. To determine if the error occurs when the data is transferred from memory, we set the computer to distribution mode and step through the execution phase of the LDA instruction.



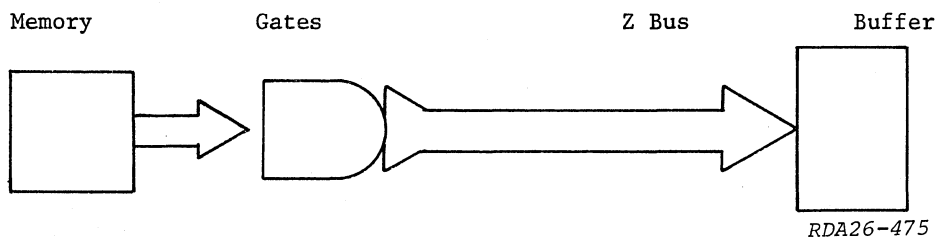
This is as far as we need to go. We can see that the error occurs somewhere between the Memory Module and the Buffer Register.

What can cause this loss of data bit 6 as the data is transferred out from Memory to the Buffer Register?

The ISB signal causes the data that has been addressed in the Memory Module to be placed on the Z bus. The data is then transferred into the Buffer Register. We could, therefore, lose data bit 6 in the following places:

1. The correct data was not stored in Memory.
2. Data bit 6 is not being correctly read from the Memory.
3. Data bit 6 is not being placed correctly on the Z bus.
4. The Z bus bit 6 is disconnected from the Buffer.
5. Data bit 6 isn't being clocked or transferred into the Buffer.

Another way to look at this list is that the data is gated out from the Memory when you address it. From the Z bus, the Buffer takes the data by parallel loading it into its internal flip-flops. This is our data flow pattern:



We can begin our actual signal tracing at any one of these places. We can start at Memory and work our way to the Buffer, or we could start at the Buffer and work back to the Memory. Finally, we could start our signal tracing somewhere in the middle, and trace in the direction of the error. This approach is usually the quickest.

In this example, we will use all three approaches. We will start at Memory for our first approach. Since we loaded the program correctly and checked it before the trouble was inserted, we will assume that the correct data is in the Memory. If we can find no error any place else along the data flow, then we will know that the Memory itself has problems. The fact that parts of the program work correctly also show that we can assume the Memory Module is working correctly insofar as it contains correct data.

Is data read out from the Memory Module correctly?

We will further check the Memory as we read data from the Memory. We read some location in Memory and check bit 6 to see if data is being read correctly. This can be measured on pin 8 6KC (sheet 8 coordinate 7C). Measuring the level at that pin after DPO of the LDA instruction (when FF is being read out) we find a high. Since we are checking a level, set the oscilloscope for internal sync, and adjust for a trace. Ground the probe, and move the trace to a crosshatch line. This is your zero voltage reference. Set the volts/cm setting to 2. Measuring the level at pin 8 after DPO we find +5 volts (the trace jumps up 2 1/2 cm). This is our expected value and means that data is being read out from the memory correctly.

Is data being gated onto the Z bus correctly?

Since the transfer we are interested in occurs only during CP1 of DPO of the LDA instruction, we need to place the computer in a loop on the LDA instruction. We do this by altering our program. We replace the STA instruction with a BUN back to the LDA instruction. We do this by storing 90 00 into memory locations 02 and 03. Our new program is now:

Memory Location	Hex Code	Mnemonics	Comment
00,01	21 00	LDA 100	
02,03	90 00	BUN 00	Branches back to 00

We place the oscilloscope sync lead on pin 5 of 21L (sheet 17 coordinate 6E), the instruction decoder, LDA signal. Each time the LDA instruction is decoded, the sweep will be triggered. We place channel A probe on pin 12 25HF (sheet 18 coordinate 6C). During DPO of the LDA and during DPO of the BUN, the trace will go low to our zero-volt reference. We are interested in the first time it goes low, because this is the DPO of the LDA instruction. We place channel B probe on pin 6 7KB (sheet 8 B7). This is the Z6 signal, or the output from the gate onto the Z bus. We clear and start the computer in program mode and watch the scope.

The sweep is triggered when the LDA instruction is decoded at DPA2. The channel A trace starts across the scope face at DPO of the LDA instruction it goes low. While channel A is low we look to see if channel B goes low. This would mean that a true bit is being put on the Z bus Z6 line. (The Z6 line goes low or true.) We find that this does occur, so the answer to the last question is yes. Correct data is being gated onto the Z bus.

Is correct data from the Z bus bit 6 reaching the Buffer?

We leave the oscilloscope sync lead connected to pin 5 21L (sheet 17 6E), the instruction decoder, LDA signal. We leave channel B probe where it is. This line goes low when data is put on line Z6. We place channel A probe on pin 6 7E (sheet 3 7E). We clear and start the program again. Channel B trace shows when the data is placed on the Z bus. Channel A should show when the data reaches the Buffer. Both channels should have identical traces, but they do not. The following shows the situation that exists.

We measure a +5 volts at pin 6 7E (sheet 3 7E). We have found what is causing the machine to fail. The problem is an open on the Z bus, bit position 6.

If we had started our search at the Buffer, we already know incorrect data is being shifted in. We would check the input line for bit 6 and we would still find incorrect data. We would then check the input to the Z bus and find correct data. Our error then is between the Z bus and the input to the Buffer.

Using the start-in-the center or divide-by-two approach, we would check our signal at the input to the Z bus first. Finding the correct data there, we would next check the input to the Buffer. Here our data is incorrect; therefore, we know the error is between the input to the Z bus and the input to the Buffer. Since this is only a wire, we know that an open exists between these two points.

This brings us to the close of our troubleshooting examples. In finding a computer error we must first determine the error indications. These indications help us determine the malfunction. By asking questions that analyze the error indications, we can limit the area that we need to check. Finally, we trace signals using one of the three methods discussed. These methods will lead us to the failure.

REVIEW QUESTIONS 3-3

000	LAN 002
002	ADD 004
004	STA 008
006	BST 008

1. With the above program, the computer stops with DP3 (1 0011) in the D-Register, ADD ERROR flip-flop set, A8 in the A-Register and 004 in both the P- and M-Registers. Memory location 008 has zero (00) stored into it. Which of the following may cause this error?

- a. 30GB Pin 5 open (sheet 23-D5)
- b. 28DB Pin 13 grounded (sheet 23-C5)
- c. 22ED Pin 9 open (sheet 22-9D)
- d. 22ED Pin 8 open (sheet 22-D9)

2. With the above program, both memory location 008 and the A-Register have E7 stored. Which of the following may cause this error?

- a. 30GB Pin 5 open (sheet 23-D5)
- b. 28DB Pin 13 grounded (sheet 23-C5)
- c. 22ED Pin 9 open (sheet 22-9D)
- d. 22ED Pin 8 open (sheet 22-9D)

3. With the above program when we check memory location 008 it has 08 stored in to it. The A-Register has E8 which is correct. Which of the following may cause this error?

- a. 30GB Pin 5 open (sheet 23-D5)
- b. 28DB Pin 13 grounded (sheet 23-C5)
- c. 22ED Pin 9 open (sheet 22-D9)
- d. 22ED Pin 8 open (sheet 22-D9)

000	LDA 101
002	ADD 102
004	STA 103
006	BST 103

C(101) = 05  
C(102) = 05  
C(103) = 00

4. With the above program when we check memory location 103 it is found that it has 08 stored into it instead of 0A, although the accumulator has 0A which is correct. Which of the following may cause this error?

- a. 5G Pin 15 open (sheet 1-E4)
- b. 7GC Pin 9 open (sheet 1-C3)



- c. 7GC Pin 8 open (sheet 1-C3)
  - d. 5G Pin 18 open (sheet 1-D4)
5. With the above program both memory location 103 and the A-Register contain 1A instead of 0A. Which of the following may cause this error?
- a. 5G Pin 15 open (sheet 1-E4)
  - b. 7GC Pin 9 open (sheet 1-C3)
  - c. 7GC Pin 8 open (sheet 1-C3)
  - d. 5G Pin 18 open (sheet 1-D4)
6. With the above program when we check memory location 103 it contains 0E instead of 0A. Which of the following may cause this error?
- a. 5G Pin 15 open (sheet 1-E4)
  - b. 7GC Pin 9 open (sheet 1-C3)
  - c. 7GC Pin 8 open (sheet 1-C3)
  - d. 5G Pin 18 open (sheet 1-D4)

```

000      LDA 101
002      SUB 102
004      STA 103
006      BST 103

```

```

C(101) = 1A
C(102) = 15
C(103) = 00

```

7. With the above program the computer stops with condition code equal zero and memory location 103 equal zero. Which of the following may cause this error?
- a. 31JA Pin 3 grounded (sheet 23-B8)
  - b. 23DA Pin 8 grounded (sheet 23-C8)
  - c. 30GC Pin 8 open (sheet 23-C4)
  - d. 30GB Pin 5 open (sheet 23-D5)
8. With the above program when memory location 103 is checked it is found to have 03 stored into it, although the accumulator has 05 which is correct. Which of the following may cause this error?
- a. 31JA Pin 3 grounded (sheet 23-B8)
  - b. 23DA Pin 8 grounded (sheet 23-C8)
  - c. 30GC Pin 8 open (sheet 23-C4)
  - d. 30GB Pin 5 open (sheet 23-D5)

9. With the previously mentioned program when memory location 103 and the A-Register are checked, both contain 03. Which of the following may cause this error?

- a. 31JA Pin 3 grounded (sheet 23-B8)
- b. 23DA Pin 8 grounded (sheet 23-C8)
- c. 30GC Pin 8 open (sheet 23-C4)
- d. 30GB Pin 5 open (sheet 23-D5)

000	LAI 15
002	SRL 03
004	STA 08
006	BST 07

10. With the above program we find that 15 has been stored in both memory location 008 and the accumulator. Which of the following may cause this error?

- a. 24KB Pin 8 grounded (sheet 18-D9)
- b. 30GD Pin 13 grounded (sheet 23-D4)
- c. 27CB Pin 4 open (sheet 24-D5)
- d. 30GB Pin 5 open (sheet 23-D5)

11. With the above program we find that 0A has been stored in both memory location 008 and the accumulator. Which of the following may cause this error?

- a. 24KB Pin 8 grounded (sheet 18-D9)
- b. 30GD Pin 13 grounded (sheet 23-D4)
- c. 27CB Pin 4 open (sheet 24-D5)
- d. 30GB Pin 5 open (sheet 23-D5)

12. With the above program we find that 08 has been stored in memory location 008 and the A-Register has 02 which is correct. Which of the following may cause this error?

- a. 24KB Pin 8 grounded (sheet 18-D9)
- b. 30GD Pin 13 grounded (sheet 23-D4)
- c. 27CB Pin 5 open (sheet 24-D5)
- d. 30GB Pin 5 open (sheet 23-D5)

#### CHAPTER REVIEW

- 1. WHAT IS THE PURPOSE OF A DIAGNOSTIC PROGRAM?
- 2. WHAT IS AN OPERATOR ERROR?

3. WHAT IS THE DIFFERENCE BETWEEN AN OPERATOR ERROR AND A MACHINE ERROR?
4. DRAW AND LABEL A FLOW CHART TO CHECK A SIMPLE INSTRUCTION IN A DIAGNOSTIC!
5. WHAT IS THE DIFFERENCE BETWEEN THE OLD AND THE NEW DIAGNOSTIC PROGRAMS?
6. WHAT ARE THE 3 MAIN REQUIREMENTS OF A VALID DIAGNOSTIC PROGRAM?
7. SHEET 11 WHAT MALFUNCTION ON WILL PREVENT THE LOADING OF....
  - A) GO BIT
  - B) Y0 AND Z0 BITS
  - C) Y0 BIT
  - D) Z0 BIT
  - E) THE CLEARING OF FLIP-FLOP 12FB SO THAT THE LIGHT FOR M0 BIT IS OFF.
8. WHAT IS THE STATIC CONDITION OF NAND-GATE 13CA D3 (SHEET 14)?
  - A) PIN 1
  - B) PIN 3
  - C) PIN 2
  - D) WHAT AFFECT DOES THE STATIC CONDITION OF PIN 3 HAVE?
9. TURN TO SHEET 6.
  - A) LIST ALL INSTRUCTIONS THAT WOULD BE AFFECTED BY AN OPEN ON PIN 1 U198 COORDINATES E6
  - B) WHAT AFFECT WOULD THIS HAVE ON LOADING THE Q-REGISTER? COULD YOU STILL PARALLEL LOAD THE Q-REGISTER? LEFT SHIFT? RIGHT SHIFT?
  - C) LIST ALL INSTRUCTIONS THAT WOULD BE AFFECTED BY A GROUND ON PIN 22 3E U198 Cord. E6
  - D) HOW WOULD THIS AFFECT THE SHIFT INSTRUCTIONS?

10. TURN TO SHEET 4.

- A) PIN 11 FLIP-FLOP 3MB C2 IS OPEN. WHEN WILL FLIP-FLOP LOAD IN DATA AT PIN 12?
- B) WHAT AFFECT WILL THIS HAVE ON THE OPERATION OF THE C-REGISTER? WHY?

11. TURN TO SHEET 9. Pin 1 of the lamp driver 9AA is open.

- A) WHEN WILL THE LIGHT L48 BE ON?
- B) IF PIN 2 IS HIGH, WHAT LEVEL WILL BE FOUND AT PIN 3?
- C) WHAT LEVEL IS NEEDED TO TURN THE LIGHT ON?
- D) IF PIN 1 IS GROUNDED - WHAT WILL BE THE CONDITION OF THE LAMP L48?

12. LIST THE 3 METHODS OF SIGNAL TRACING.. WHICH IS NORMALLY THE FASTEST METHOD? WHY?

## SIGNAL NAME GLOSSARY

The Glossary lists all mnemonics used by the COM-TRAN TEN. They are listed in alphabetical order. The page numbers refer to the schematic sheets where the signal is generated. The signal name, page reference, and signal description are provided, reading from left to right.

### A

AO - A7	1	Accumulator register bit 0 (LSB) through bit 7 (MSB)
ADD	17	Instruction; ADD
ADEB	15	Add Error Bypass; Enabled by a control panel switch.
AEM	15	Acquisiton-Execution Mode. Enabled by a control panel switch.
AND	17	Instruction; AND
APH	29	Enables Alpha mode decoding in the Interface.
APHM	29	Alpha Mode signal; Informs the control panel that the Interface is in Alpha Mode.
ARG	18	Arithmetic Group; composed of ADD and SUB Instructions
AXM	23	Add the contents of the Index register to the contents of the Memory register. The end carry is disregarded.
AZ	1	The Accumulator register contains zeros in all bit positions.

### B

BO - B7	3	Buffer register bit 0 (LSB) through bit 7 (MSB)
BNC	17	Instruction; Branch on No Carry.
BNG	17	Instruction; Branch on Negative.
BPS	17	Instruction; Branch on Positive.
BSB	17	Instruction; Branch to Subroutine.
BST	17	Instruction; Branch and Stop.
BUN	17	Instruction; Branch Unconditional.
BXZ	17	Instruction; Branch on Index Zero.
BZ	3	The Buffer register contains zeros in all bit positions.
BZE	17	Instruction; Branch on Zero

### C

CO - C7	4	Countdown register bit 0 (LSB) through bit 7 (MSB)
CARY	19	Carry signal which is enabled if an arithmetic carry is detected from the Arithmetic Logic Unit.

CCEQ	19	Condition Code Equals Zero. Reflects the result of an Arithmetic Instruction.
CCGT	19	Condition Code Greater Than Zero. Reflects the result of an Arithmetic Instruction.
CCLT	19	Condition Code Less Than Zero. Reflects the result of an Arithmetic Instruction.
CKE	18	Clock the Execution flip-flop. This determines whether the computer is in Acquisition or Execution Mode.
CL	15	Clear all register contents and flip-flops.
$\overline{\text{CL}}$	26	Clear all register contents and flip-flops.
CLA	23	Reset all bits of the Accumulator to zeros.
COL	29	Colon decode signal used in the Interface.
COMP	20	Selects the two's complement of either the Accumulator or Quotient register.
COMP A	20	Executes a two's complement on the contents of the Accumulator.
COMP Q	20	Executes a two's complement on the contents of the Quotient register.
CP1	7	Clock pulse one.
CP2	7	Clock pulse two.
CP3	7	Clock pulse three.
CR	29	Carriage return signal used in the Interface.
CZ	4	The Countdown register contains zeros in all bit positions.

D

DO - D3	5	Distributor bit 0 (LSB) through bit 3 (MSB)
DEC	24	Decrement the Countdown register contents by one.
DECA	21	Decrement the Accumulator contents by one.
DIV	17	Instruction; Divide.
DPO - DP15	18	Distributor pulse 0 (Execution mode) through pulse 15 (Execution mode).
DPA0 - DPA15	18	Distributor pulse 0 (Acquisition mode) through pulse 15 (Acquisition Mode)
DSLA	24	Input serial data to the Accumulator. Data present in the register will be shifted to the left one place as each bit of serial data is inputed.
DSLQ	20	(Same as above only substitute Quotient register for Accumulator.)

DSM	15	Distributor Mode. Enabled by a control panel switch.
DSRA	15	Input serial data to the Accumulator. Data present in the register will be shifted to the right one place as each bit of serial data is inputed.
DVEB	15	Division Error Bypass. Enabled by a switch on the control panel.
<u>DVEB</u>	26	Division Error Bypass. Enabled by a switch on the control panel.
DBG V	18	Divide Group signal which is enabled during DP8-11 of the Divide Instruction.

E

E	5	Execution Mode.
EC	2	End Carry generated by the Arithmetic Logic Unit.
ENABLE	7	Clock signal which enables the Distributor.
ERA	19	Arithmetic Error.
ERD	19	Division Error.
ERI	19	Instruction Error.

F

FO - F7	2	Arithmetic Logic Unit bit 0 (LSB) through bit 7 (MSB)
FLC	17	Instruction; Clear Flag
FLG	19	Flag flip-flop output enabled by FLS Instruction.
FLS	17	Instruction; Set Flag.

G

GO - G9	14	Index adder bit 0 (LSB) through bit 9 (MSB)
---------	----	---

H

H1	25	Hex 1 Switch signal from the control panel matrix.
H2	25	" 2
H3	25	" 3
H4	25	" 4
H5	25	" 5
H6	25	" 6
H7	25	" 7
H8	25	" 8
H9	25	" 9
HA	25	" A
HB	25	" B
HC	25	" C
HD	25	" D
HE	25	" E
HF	25	" F

HEX	29	Enables Hex decoding in the Interface.
HEXM	29	Hex Mode signal; informs the control panel that the Interface is in Hex mode.
HX2	29	Low order hex word in the Interface.
HXS	25	Signal from the control panel which indicates that a hex switch has been depressed.

I

I0 - I9	10	Input register bit 0 (LSB) through bit 9 (MSB)
IAND	22	Initiate AND signal to the Arithmetic Logic Unit.
IBS	24	Initiate transfer of Buffer register contents to memory storage.
ICLK	15	Input register clock which enables transfer of data from Input switches to Input register.
IEX	22	Initiate Exclusive-OR signal to the Arithmetic Logic Unit.
INA	22	Initiate Add signal to the Arithmetic Logic Unit.
INCA	21	Increment the Accumulator contents by one.
INCM	24	Increment the Memory register by one.
INEB	15	Instruction Error Bypass generated by the control panel.
INM	15	Instruction Mode, enabled by control panel switch.
INOC	20	Initiate Output Command. Computer is ready to output data to an external device.
INR	20	Initiate Read. Initiates Read circuitry in the computer and the Interface.
$\overline{\text{INR}}$	29	Initiates Read. Initiates Read circuitry in the computer and the Interface.
INRD	20	Initiate Read Data Block. Enables read circuitry in the computer and the Interface.
INS	22	Initiate Subtract signal to the Arithmetic Logic Unit.
INSS	20	Initiate Sense Status. Enables the previously addressed external device to send a status word to the computer.
INT	19	Interrupt signal which is generated from an external device. It halts computer operations.
INW	20	Initiates write circuitry within the computer and the external device.
INWD	20	Enables write circuitry within the computer and the external device.



INWD	29	Enables write circuitry within the computer and the external device.
INX	17	Instruction; Increase Index.
IOR	17	Instruction; Inclusive-OR.
IOR1	22	Initiate Inclusive-OR signal at the Arithmetic Logic Unit.
IRB	15	Initiates Read Until Interrupt instruction.
IRD	15	Initiates Manual Output instructions.
IS0	25	Input register switch 1 signal from the control panel.
IS1	25	" 2 "
IS2	25	" 3 "
IS3	25	" 4 "
IS4	25	" 5 "
IS5	25	" 6 "
IS6	25	" 7 "
IS7	25	" 8 "
IS8	25	" 9 "
IS9	25	" 10 "
ISB	24	Initiate the transfer of the contents of the addressed memory location to the Buffer register.
IWB	15	Initiate Write Block instruction.
IWT	15	Initiate Manual Input instruction.
L		
LAI	17	Instruction; Load Accumulator Immediate.
LAMP TEST	26	Illuminates all lights driven by logic signals.
LAN	17	Instruction; Load Accumulator Negative.
LAPH	10	Interface signal which illuminates the Alpha indicator on the control panel.
LCC	17	Instruction; Load Consecutive.
LCG	18	Logic group signal enabled by AND, XOR, and IOR.
LCI	18	Instruction; Load C Immediate.
LDA	17	Instruction; Load A
LDQ	17	Instruction, Load Quotient Register.
LF	29	Line feed signal used in the Interface.
LHEX	10	Interface signal which illuminates the HEX indicator on the control panel.
LX	22	Load the Index register with either the contents of the Memory register or the Y bus.

LXI		Instruction; Load Index Register Immediate.
M		
MO - M9	11	Memory register bit 0 (LSB) through bit 9 (MSB)
MNI	17	Instruction; Manual Input
MNO	17	Instruction; Manual Output
MPY	17	Instruction; Multiply
MLD	7	Manual load signal which is generated by the control panel switches.
O		
OCD	17	Instruction; Output Command.
P		
PO - P9	12	Program register bit 0 (LSB) through bit 9 (MSB)
Q		
QO - Q7	6	Quotient register bit 0 (LSB) through bit 7 (MSB)
QOQ6	6	Quotient register bit positions 0-6 are reset to zero.
QZ	6	All Quotient register bit positions are reset to zero.
R		
R	27	Read mode signal used in the Interface.
RO - R7	27	Interface read data bit 0 (LSB) through bit 7 (MSB)
RAO	17	Instruction; Replace Add One.
RDB	17	Instruction; Read Data Block.
RDG	20	Read group signal composed of RDI, MNI, and RDB instructions.
RDI	17	Instruction; Read Until Interrupt.
RES	27	Resume pulse used in the Interface.
RESM	28	Resume pulse generated by the Interface and used by the computer.
REXMT	26	Retransmit signal generated by a control panel switch.
RPG	18	Replace group signal composed of RAO and RSO instructions.
RPT	15	Repeat mode which is set by a control panel switch.
RSO	17	Instruction; Replace Subtract One.
RST	26	Reset signal generated by a control panel switch. This signal clears the Input register to zeros.

## S

SO - S7	13	Operation Code register bit 0 (LSB) through bit 7 (MSB)
SA	26	Accumulator input switch signal.
SAEM	26	Enable Acquisition-Execution Mode. This signal is generated by a control panel switch.
SAL	23	Shift the content of the Accumulator left.
SAM	28	Set Alpha Mode. This signal is generated by the external device and used in the Interface.
SAOV	1	Set Add Overflow. Generation of this signal results in an Add Error.
SAPH	15	Set Alpha Mode signal generated by a control panel switch. This enables Interface Alpha Decoding.
<u>SAPH</u>	26	Set Alpha Mode signal generated by a control panel switch. This enables Interface Alpha Decoding.
SAQL	22	Shift the contents of the Accumulator and Quotient register left. The contents of bit position of the Quotient register will be shifted into bit position 0 of the Accumulator
SAQR	22	Shift the contents of the Accumulator and Quotient register right. The contents of bit position 0 of the Accumulator will be shifted into bit position 7 of the Quotient register.
SAR	23	Shift the contents of the Accumulator right.
SB	26	Buffer register input switch signal.
SC	26	Countdown register input switch signal.
SC8	22	Set bit position 3 of the Countdown register to a logical 1. Thus the Countdown register will contain a hex count of 08.
SD	26	Distributor input switch signal.
SDP1	24	Set the Distributor to a hex count of 1.
SDP4	21	" " 4.
SDP6	19	" " 6.
SDP9	21	" " 9.
SDP11	21	" " B.
SDP15	21	" " F.
SDSM	26	Enable Distributor Mode. This signal is generated by a control panel switch.
SE	5	Set Execution Mode. This is accomplished by clocking the E flip-flop.
SEL	29	Select signal which selects the external device which the computer will interface with.

SEN	15	Output of the Sense flip-flop which is set by the Sense switch or the SST instruction.
SERI	17	Set Instruction Error. Halts the computer when an invalid instruction is encountered.
SERD	24	Set Divide Error. Enabled when an illegal division is attempted.
SFG	18	Shift group composed of Shift instructions.
SHEX	15 26	Set Hex Mode. This signal is generated by a control panel switch and enables the interface to decode in a hex format.
SHM	28	Set Hex Mode in the Interface. This signal is generated by the external device.
SINM	26	Set Instruction Mode. This signal is generated by a control panel switch.
SINT	28	Set Interrupt signal which comes from the interface.
SKF	17	Instruction; Skip on Flag.
SKG	18	Skip Group made up of Skip Instructions.
SKI	17	Instruction; Skip on Interrupt
SKS	17	Instruction; Skip on Sense
SLA	17	Instruction; Shift Left Arithmetic.
§LL	17	Instruction; Shift Left Logical.
§M	26	Memory register input switch.
SNF	23	Sign flip-flop. This detects the sign of the result of an Arithmetic operation.
SP	26	Program Register input switch.
SPCK	18	Stop Clock signal. This will halt the computer clock.
SPSW	26	Stop switch signal.
SQ	26	Quotient register input switch.
SQL	22	Shift the contents of the Quotient register left.
SQR	22	Shift the contents of the Quotient register right.
SRA	17	Instruction; Shift Right Arithmetic.
§RB	26	Set Read Until Interrupt signal from the control panel.
SRD	26	Set Manual Output signal from the control panel.
SRL	17	Instruction; Shift Right Logical.

SRPT	26	Set Repeat Mode signal from the control panel.
SS	26	Operation Code register input switch.
SSEN	62	Set sense signal from control panel.
SST	17	Instruction; Sense Status.
STA	17	Instruction; Store Accumulator.
START	15	Start Computer Clock signal.
STG	18	Store group composed of Store instructions.
STOP	20	Stop computer clock signal.
STQ	17	Instruction; Store Quotient
STSW	26	Start signal generated at control panel.
STX	17	Instruction; Store Index
SUB	17	Instruction; Subtract
SW	26	Signal generated whenever a switch is pushed.
SWB	26	Set Write Block signal from control panel.
SWT	26	Set Write signal from control panel.
SX	26	Index register input switch.

T

T9	27	Timing pulse 9 used in the Interface.
TAB	23	Transfer the contents of the Accumulator to the Buffer register.
TAZ	23	Transfer the contents of the Accumulator to the Z bus.
TBM	20	Transfer the contents of the Buffer register to the Memory
TBS	20	register. <i>TRANS. BUFFER TO S REGISTER</i>
TEB	20	Transfer external data to the Buffer register.
TGM	20	Transfer the contents of the Index adder to the Memory register.
TIA	15	Transfer the contents of the Input register to the Accumulator.
TIB	24	Transfer the contents of the Input register to the Buffer register.
TIBI	15	Enable signal which accomplishes TIB subcommand.
TIC	15	Transfer the contents of the Input register to the Countdown register.
TID	15	Transfer the contents of the Input register to the Distributor.
		<i>TBA — TRAN. Buffer to Accu.</i>

TIM	15	Transfer the contents of the Input register to the Memory register.
TIP	15	Transfer the contents of the Input register to the Program register.
TIQ	15	Transfer the contents of the Input register to the Quotient register.
TIS	15	Transfer the contents of the Input register to the Operation Code register.
TIX	15	Transfer the contents of the Input register to the Index register.
TIY	22	Transfer the contents of the Input register to the Y bus.
TMP	21	Transfer the contents of the Memory register to the Program register.
TMX	22	Transfer the contents of the Memory register to the Index register.
TPHB	20	Transfer the two high-order bits of the Program register to the Buffer register, and generate the BUN for the BSB.
TPLB	20	Transfer the eight low-order bits of the Program register to the Buffer register.
TPM	24	Transfer the contents of the Program Register to the Memory register.
TQB	22	Transfer the contents of the Quotient Register to the Buffer register.
TQZ	22	Transfer the contents of the Quotient register to the Z bus.
TTY In		Data from the external device.
TTY Out	27	Data transferred to the external device.
TWOS	22	Signal which enables a two's complement of a given number.
TXB	23	Transfer the contents of the Index register to the Buffer register.
TYC	22	Transfer the contents of the Y bus to the Countdown register.
TYI	23	Transfer the contents of the Y bus to the Input register.
TZB	19	Transfer the contents of the Z bus to the Buffer register.
W		
W	27	Write enable signal in the Interface.
WO - W7	29	Interface write data bit 0 (LSB) through bit 7 (MSB)
WAIT	23	The computer is waiting for an input/output signal from an external device.
WDB	17	Instruction; Write Data Block.

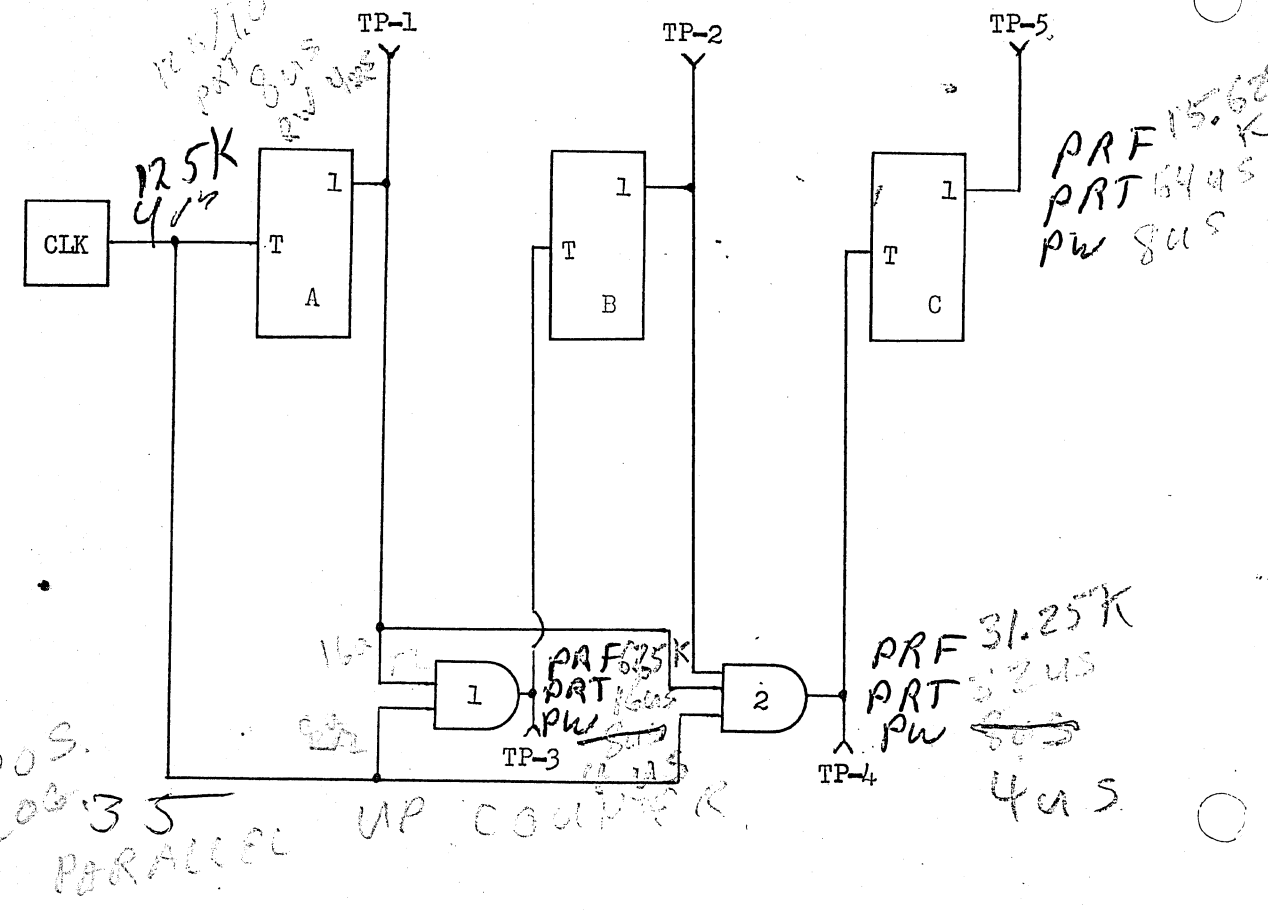
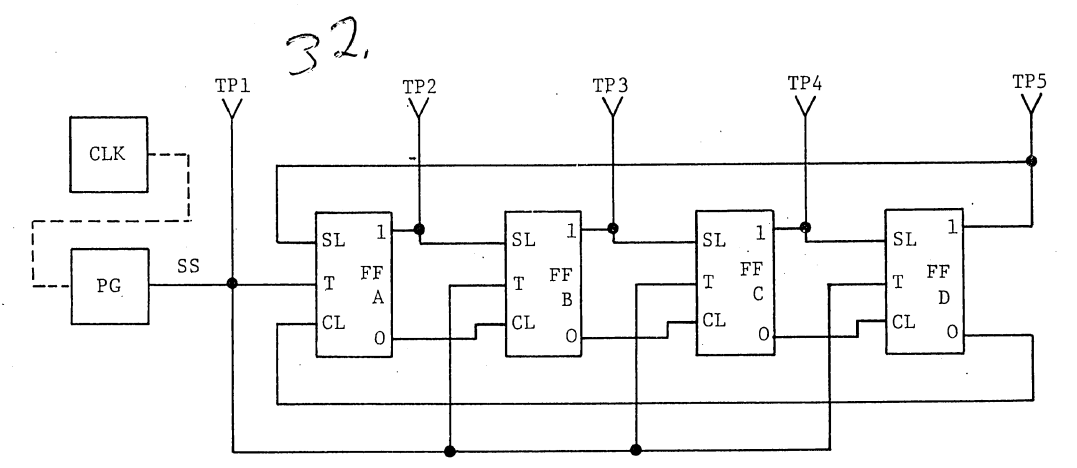
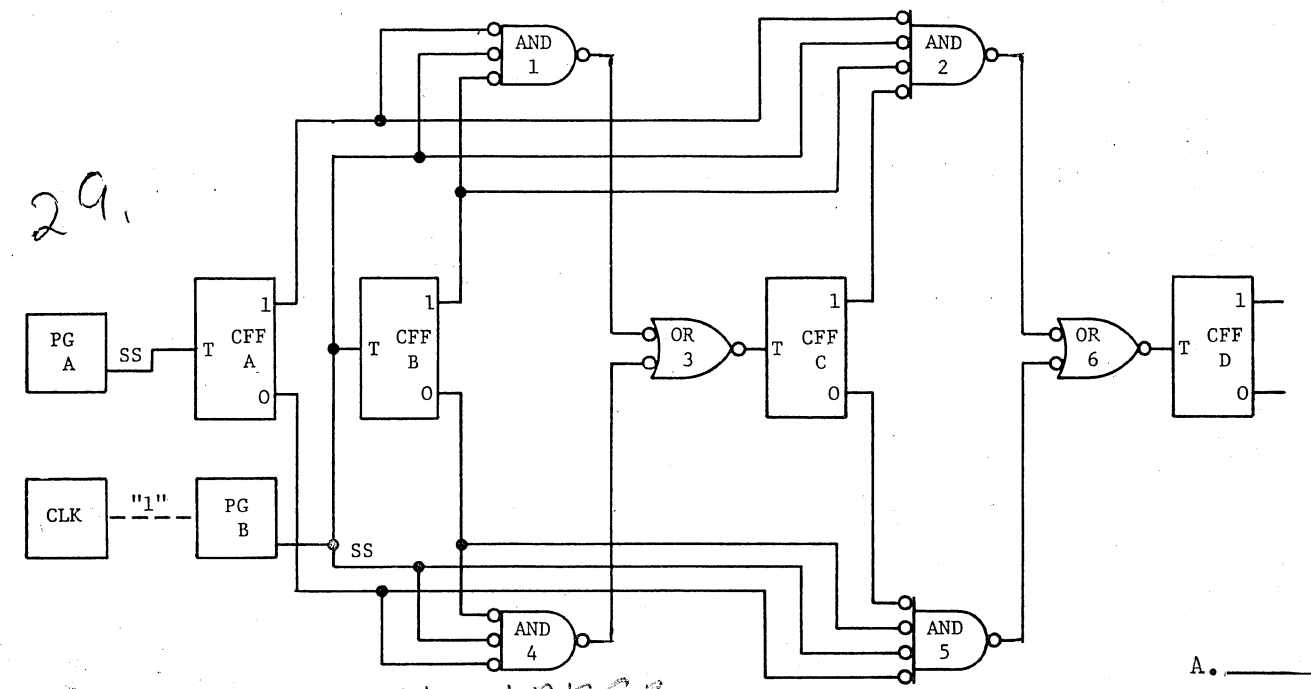
WG	18	Write group composed of MNO and WDB instructions.
X		
XO - X7	14	Index register bit 0 (LSB) through bit 7 (MSB)
XON	29	Reader on signal generated by the interface.
XOR	17	Instruction; Exclusive-OR
XZ	14	Index register contains zeros in all bit positions.
Y		
YO - Y7	3	Y bus bit 0 (LSB) through bit 7 (MSB)
Z		
ZO - Z7	*	Z bus bit 0 (LSB) through bit 7 (MSB)

\* Z bus generated on pages: 1, 6, 8, 10, 12, 14, & 28.

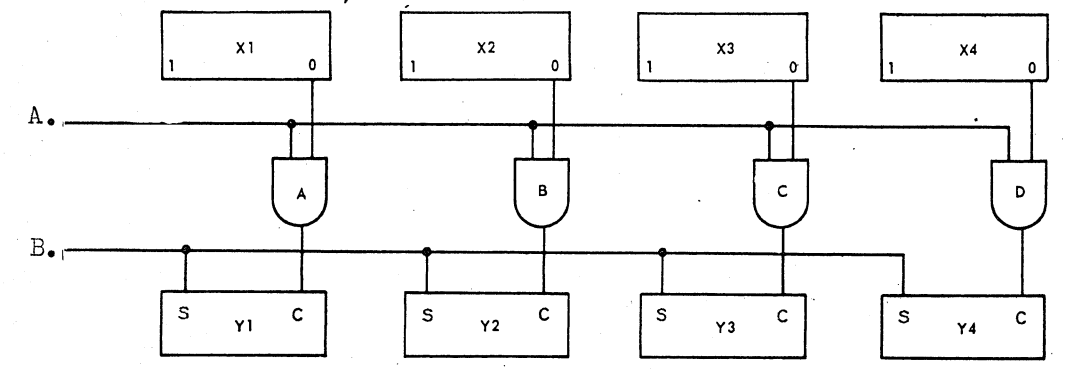
## NOTES



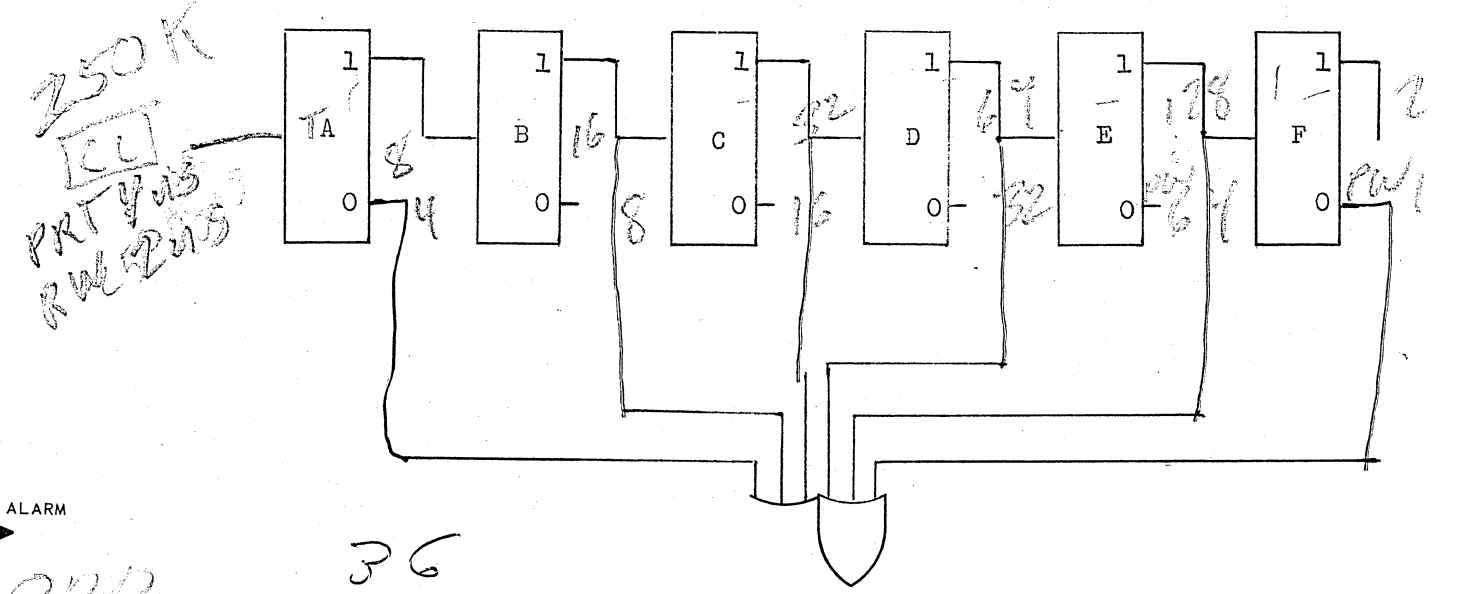
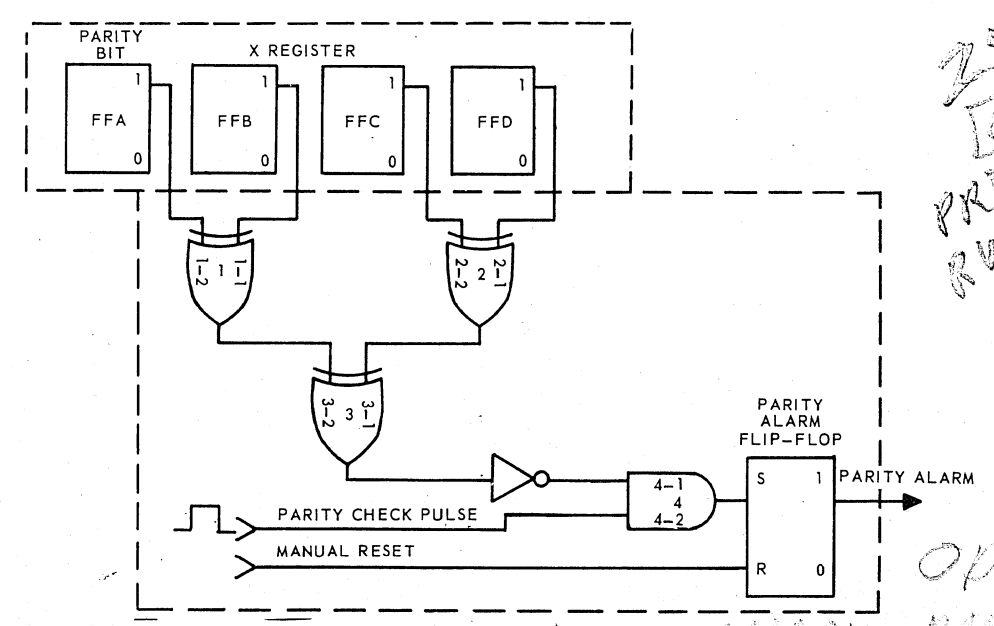
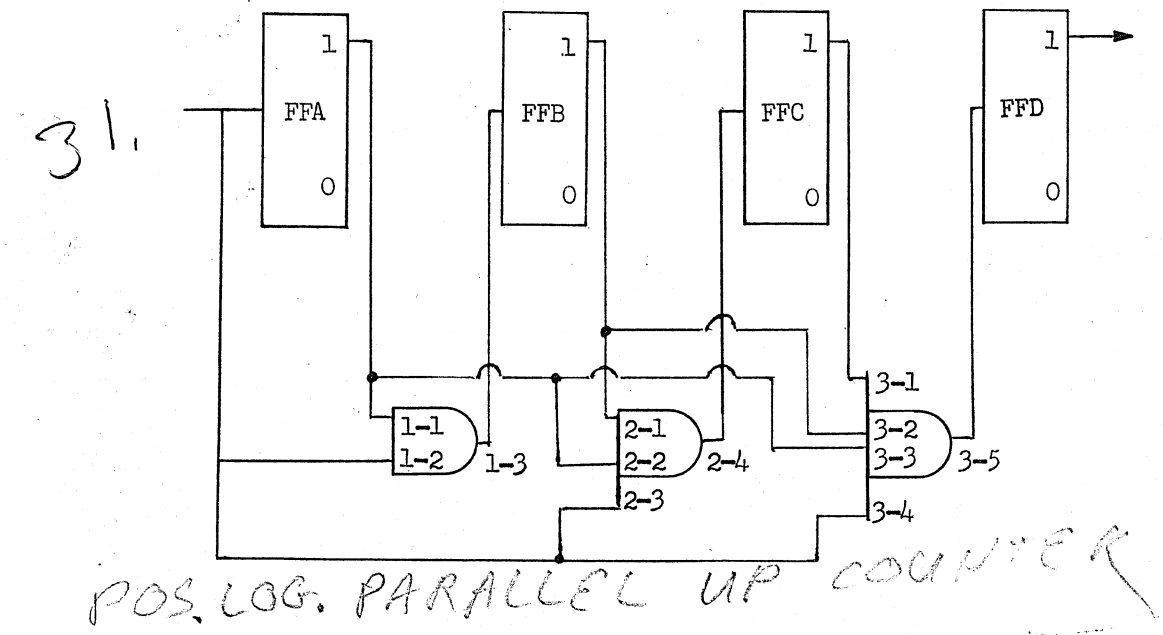
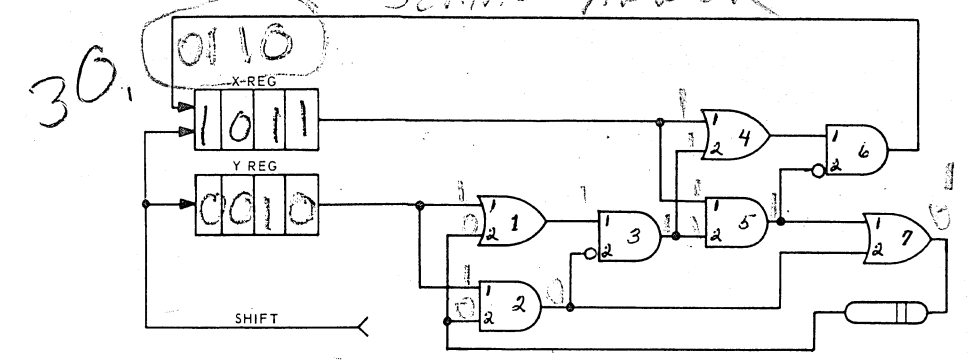
PARALLEL UP DOWN COUNTER

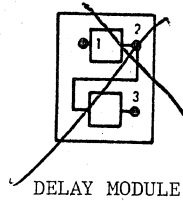
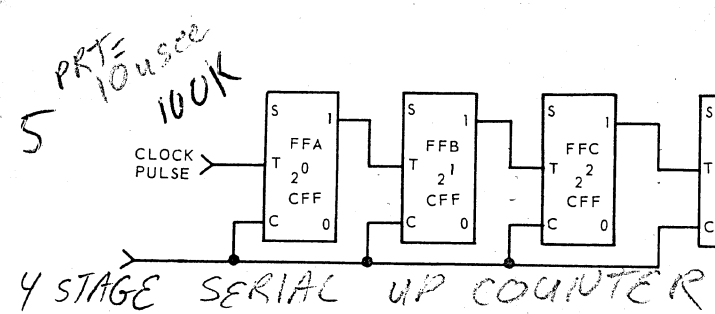
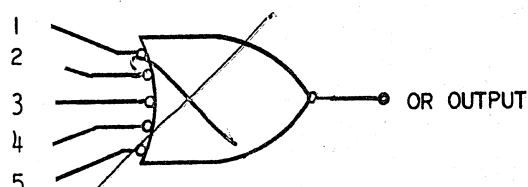
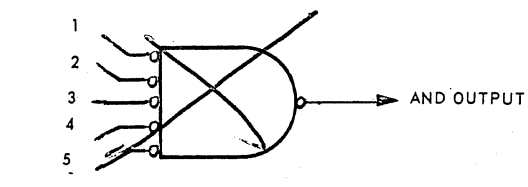
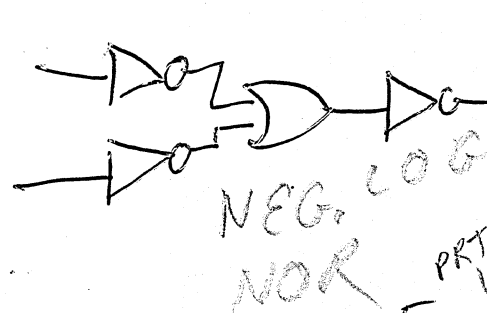
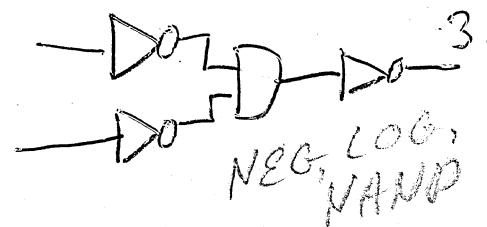
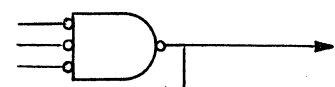
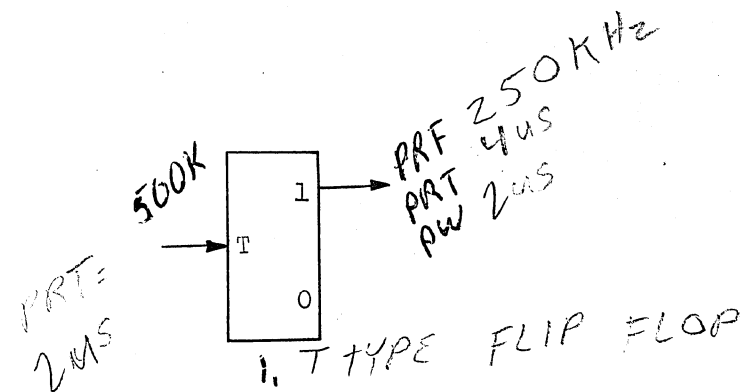


33. Type of Xfer

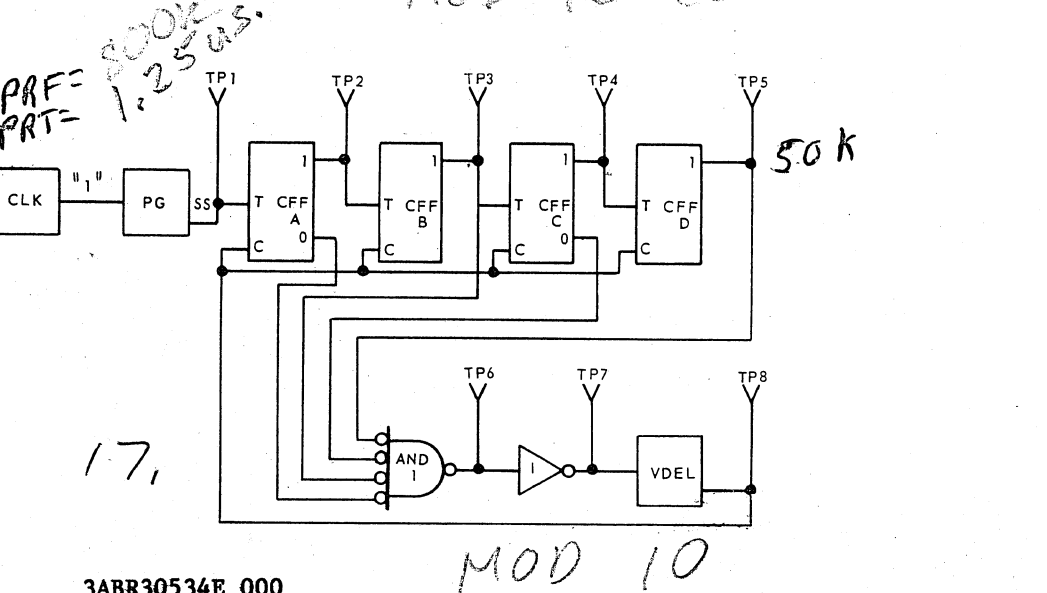
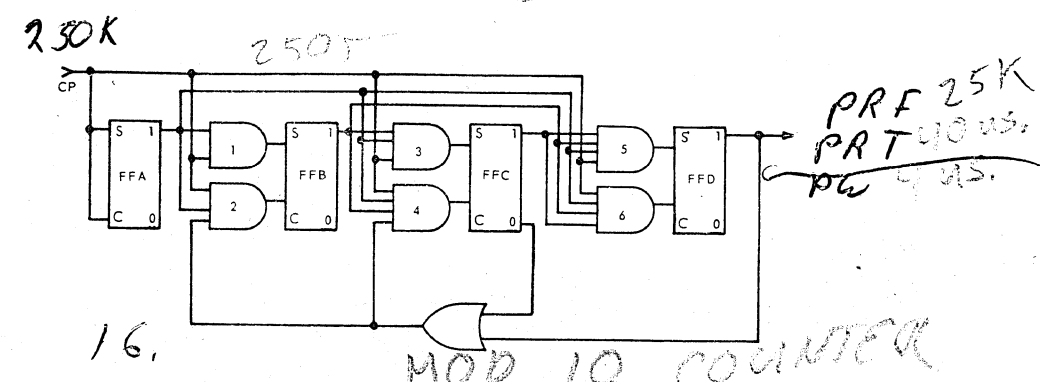
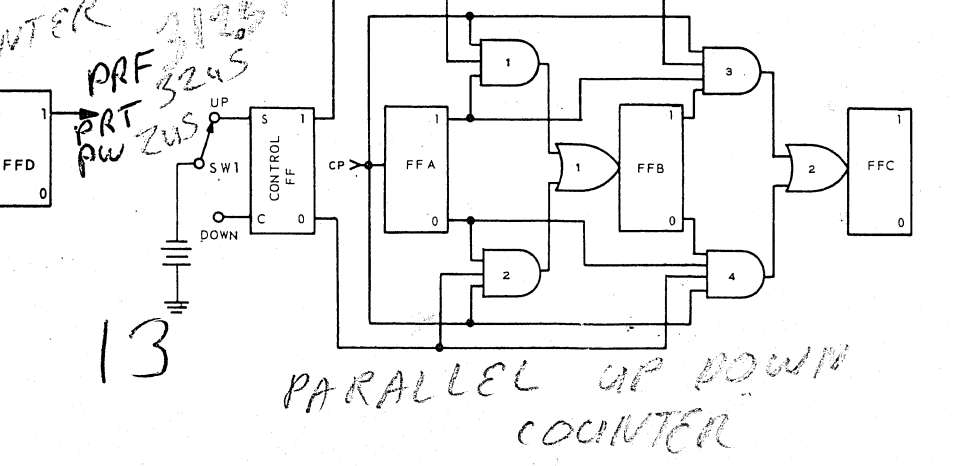
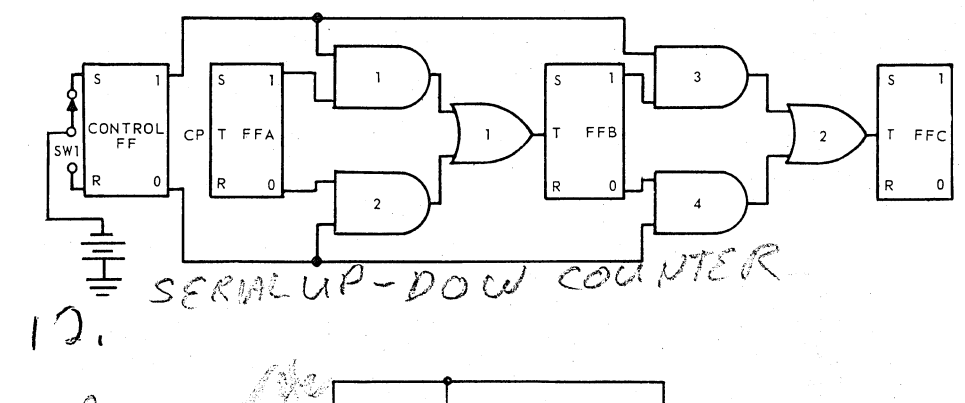
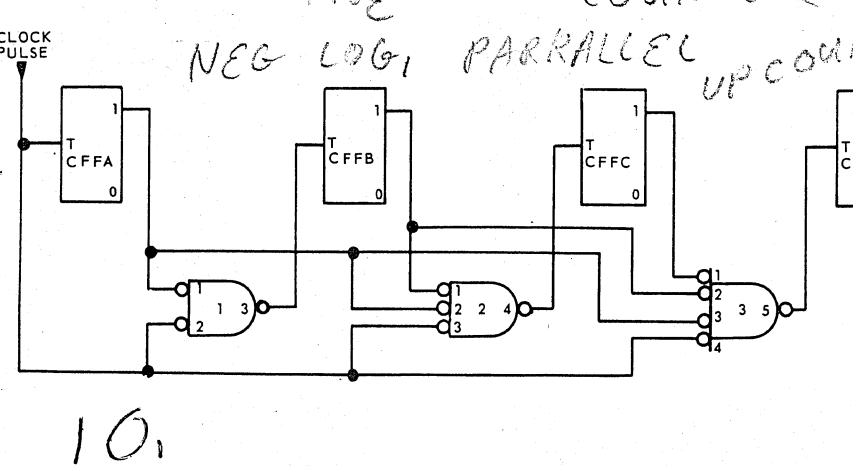
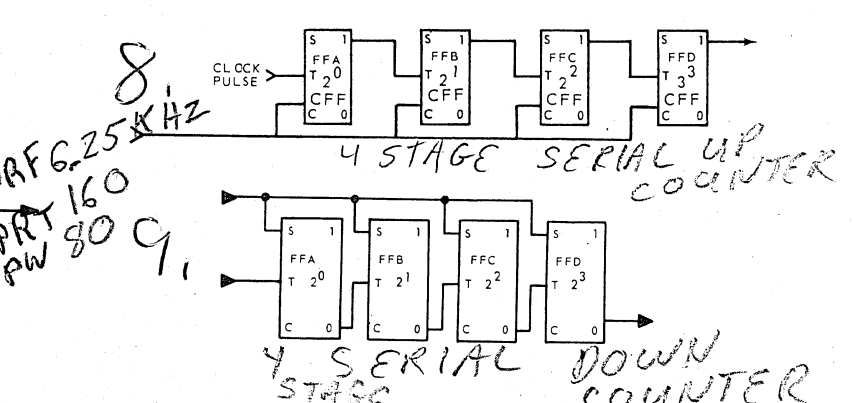
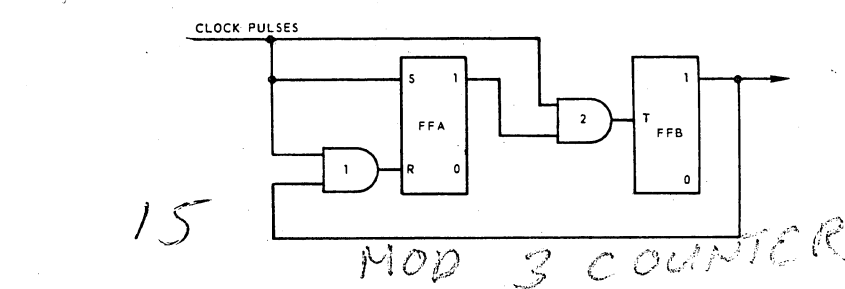
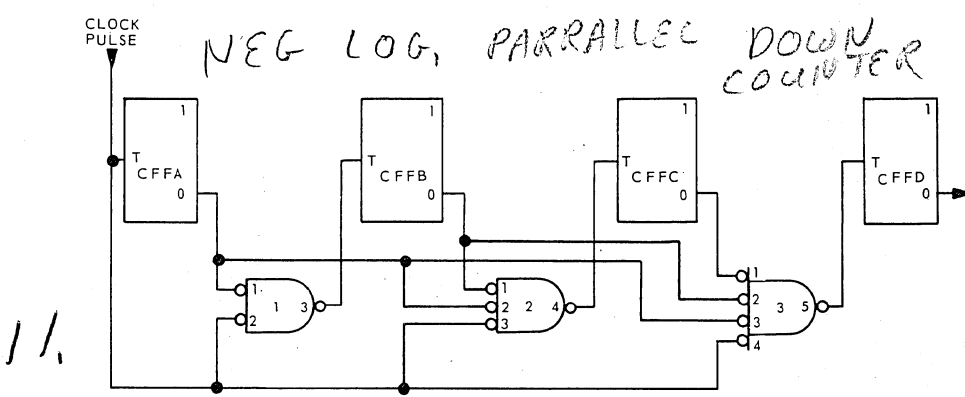
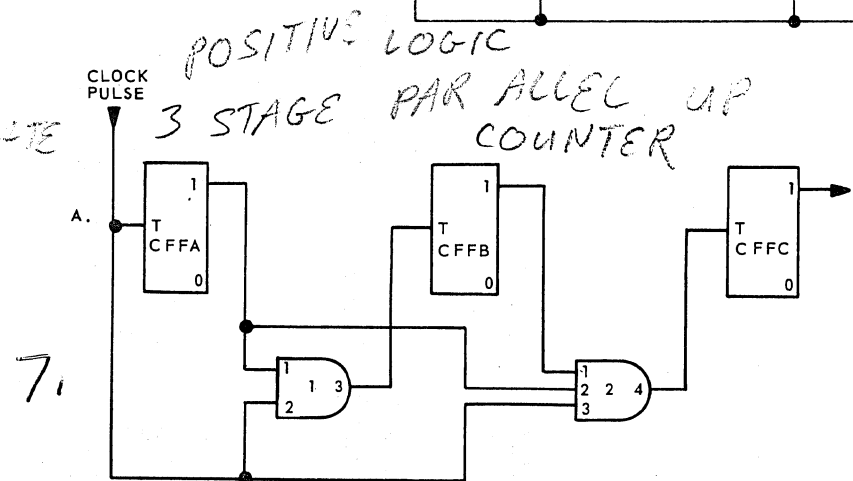
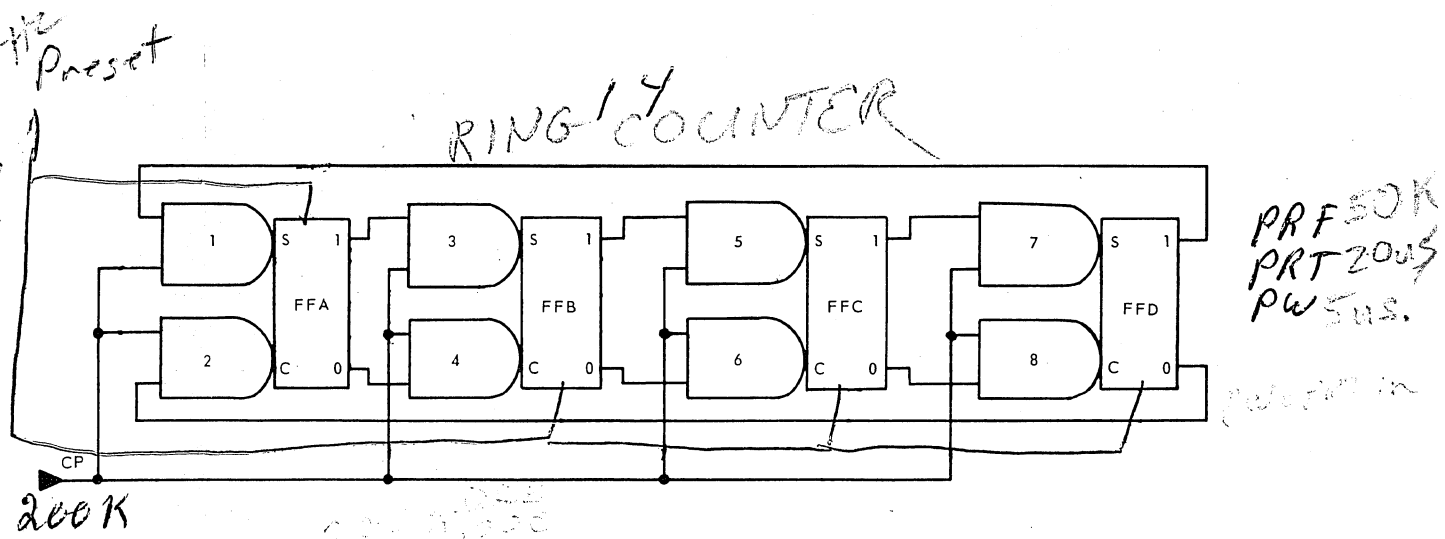
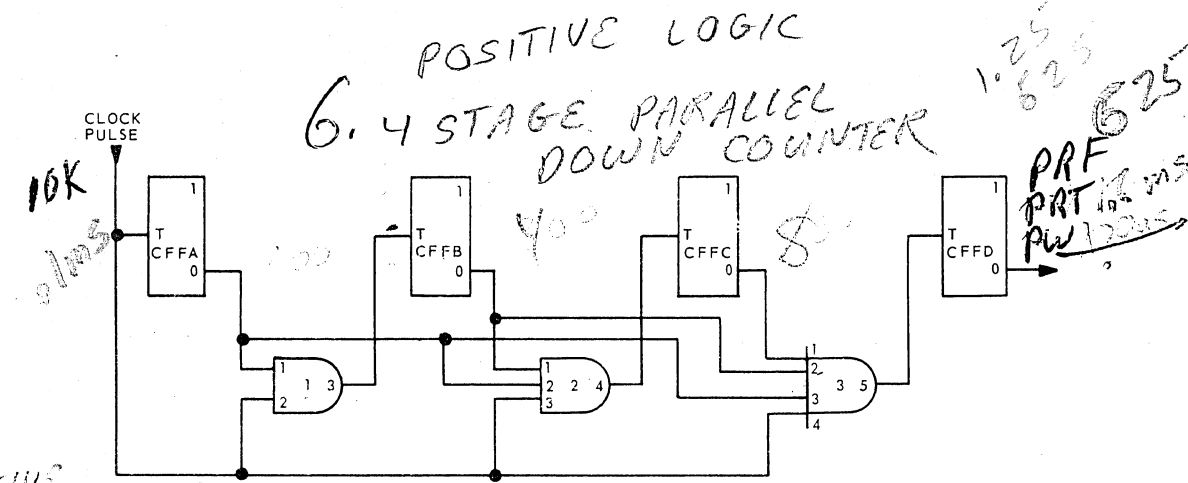
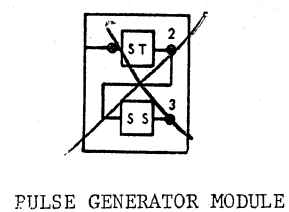


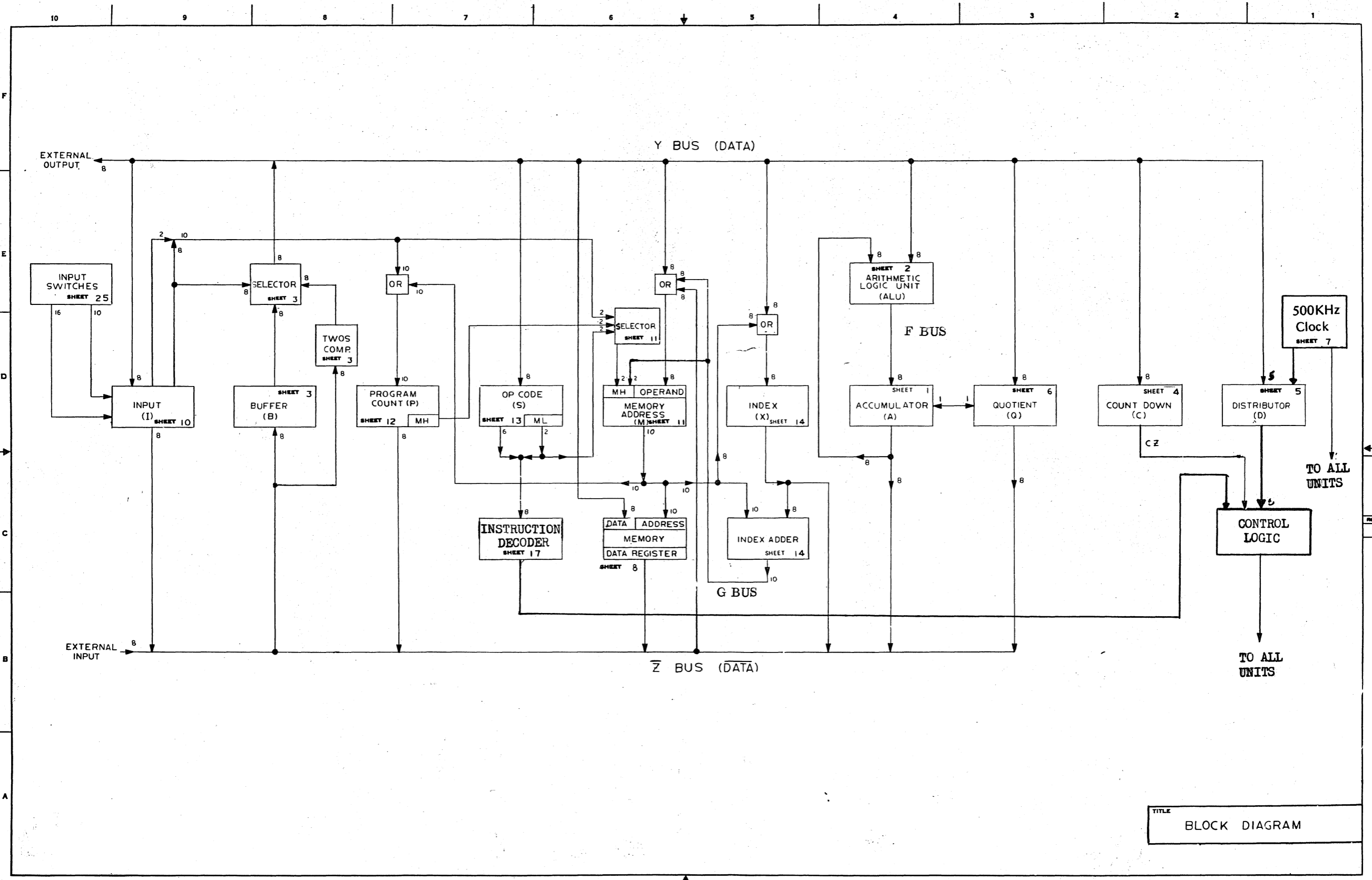
SERIAL ADDER



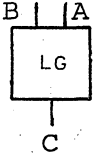
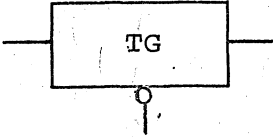
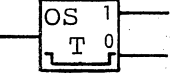
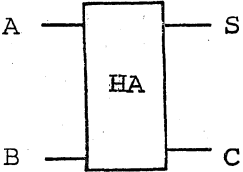
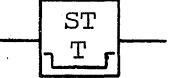
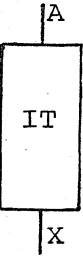
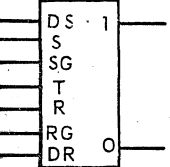
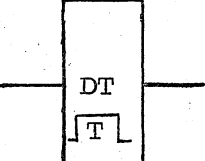
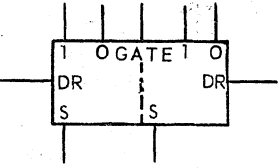
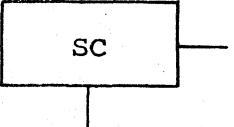
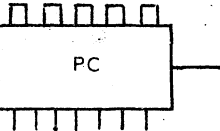


ATC Keesler 1-2462





TITLE  
BLOCK DIAGRAM

SYMBOL	NAME	LOGIC EQUATION	DESCRIPTION	SYMBOL	NAME	LOGIC EQUATION	DESCRIPTION
	LINE RELAY GATE	$A \cdot \bar{B} = C$	PROVIDES LOGIC 1 OUTPUT IF AND ONLY IF A INPUT IS LOGIC 1 AND B INPUT IS A LOGIC 0.		TIMING GENERATOR		GENERATES CLOCK PULSES FOR SYNCHRONIZATION AND/OR TIMING.
	DELAY OR MONOSTABLE MULTIVIBRATOR OR ONE SHOT		CHANGES STATE (LOGIC 0 TO LOGIC 1) UPON ACTIVATION BY A TRANSITION TO LOGIC 1; MAINTAINS CHANGED STATE FOR A TIME T, THEN RETURNS TO ORIGINAL STATE.		HALF ADDER	$S = \bar{A} \cdot B + A \cdot \bar{B}$ $C = A \cdot B$	ACCEPTS TWO INPUTS, A AND B, AND PERFORMS BINARY ADDITION, PROVIDING TWO OUTPUTS: A SUM (S) AND A CARRY (C).
	SCHMITT TRIGGER		PROVIDES A SPECIFIC VOLTAGE OR LOGIC LEVEL OUTPUT FOR THE DURATION OF THE APPLIED INPUT WAVESHAVE. (USED PRIMARILY AS A WAVESHAPING CIRCUIT)		ISOLATION TRANSFORMER	$X = A$	ISOLATES INPUT LINE FROM INTERNAL CIRCUITS. OUTPUT LOGIC LEVEL EQUALS INPUT LOGIC LEVEL.
	BISTABLE MULTIVIBRATOR		<p>SET STATE OF A FLIPFLOP EQUALS "1" OUT OF "1" SIDE THE FOLLOWING INPUTS WILL SET A FLIP FLOP;</p> <p>A. A "1" OR <math>\nabla</math> APPLIED TO DS</p> <p>B. A "0" OR <math>\nabla</math> APPLIED TO S INPUT WITH A "0" APPLIED TO SG.</p> <p>RESET STATE OF A FLIP FLOP EQUALS "1" OUT OF "0" SIDE</p> <p>THE FOLLOWING INPUTS WILL RESET A FLIP FLOP;</p> <p>A. A "1" OR <math>\nabla</math> APPLIED TO DR</p> <p>B. A "0" OR <math>\nabla</math> APPLIED TO R INPUT WITH A "0" APPLIED TO RG.</p> <p>AN <math>\nabla</math> APPLIED TO THE T INPUT WILL CAUSE THE FLIP FLOP TO CHANGE STATES.</p> <p>NOTE: A "1" ON RG OR SG WILL INHIBIT THE FLIP FLOP FROM CHANGE OF STATE EVEN IF A <math>\nabla</math> IS APPLIED TO THE R OR S INPUTS. DS, DR, AND T ARE NOT EFFECTED BY THE GATE INPUTS AND WILL FUNCTION NORMALLY.</p>		DELAY TIMER		ACTIVATED BY A $\nabla$ ; OUTPUT EQUALS LOGIC 0 FOR TIME T, THEN BECOMES LOGIC 1; RESET BY A $\nabla$ .
	REGISTER		<p>SET STATE OF A REGISTER EQUALS "1" OUT OF "1" SIDE. A "1" OR <math>\nabla</math> APPLIED TO S WITH A "1" APPLIED TO THE GATE WILL SET REGISTER.</p> <p>RESET STATE OF A REGISTER EQUALS "1" OUT OF "0" SIDE. A "1" OR <math>\nabla</math> APPLIED TO DR WILL RESET REGISTER.</p>		SIGNAL CONVERTER		28V INTO SC EQUALS 0V AT OUTPUT; 0V INTO SC EQUALS -6V AT OUTPUT.
	PARITY CHECKER		PROVIDES "0" OUT WITH EVEN # OF 1'S IN. PROVIDES "1" OUT WITH ODD # OF 1'S IN.				

CODE			DESCRIPTION	DP0	DP1	DP2	DP3	DP4	DP5	DP6	DP7	DP8	DP9	DP10	DP11	DP12	DP13	DP14	DP15
SYM	HEX	BINARY		TPM, TPLB CL ERI	ISB	TBS	SERI, STOP IF INST ERR BYPASS	INCM	ISB	INCM	IF NOT RPT (AE+ INST) TMP	TBM S0, S1-M8 M9	IF S2=1 AQM S2M	SDP 15					

PG. 174-96

CLASS

20-27

40-49

50-59

70-77

80-87

LOGICAL

BRANCH

I/O

CODE			DESCRIPTION	MICRO-STEP	DP0	DP1	DP2	DP3	DP4	DP5	DP6	DP7	DP8	DP9	DP10	DP11	DP12	DP13	DP14	DP15			
SYM	HEX	BINARY			TBC	ISB	TBA	SDP 15															STCC CKE
LCI	01	0000 0001	LOAD C IMMEDIATE	K-C			SDP 15															178	
LDA	20	0010 0000	LOAD A	C(M)+A	24	23	SDP 15															179	
LAI	02	0000 0010	LOAD A IMMEDIATE	K+A			SDP 15															180	
LXI	12	0001 0010	LOAD INDEX IMMEDIATE	K-X			SDP 15															181	
LCC	30	0011 0000	LOAD CONSECUTIVE	C(M)-(M+1)			SDP 15															182	
LAN	38	0011 1000	LOAD A NEGATIVE	C(M)+1-A			SDP 15															183	
LQ	40	0100 0000	LOAD Q	C(M)+Q			SDP 15															184	
STA	48	0100 1000	STORE A	A-C(M)			SDP 15															185	
STX	50	0101 0000	STORE INDEX	X+C(M)			SDP 15															186	
STQ	58	0101 1000	STORE Q	Q+C(M)			SDP 15															187	
ADD	60	0110 0000	ADD	A+C(M)+A	CLCR ISB	INA IF OVFL SAOV, IF EC SCARY	SDP 15															188	
SUB	68	0110 1000	SUBTRACT	A-C(M)+A	CLCR ISB	INS IF OVFL SAOV, IF EC SCARY	SDP 15															189	
MPY	70	0111 0000	MULTIPLY	AxC(M)+AQ	ISB SCB	TBQ IF B7=0 SDP 4	2's COMPA	2's COMP Q	TAB	CLA	IF Q0=0 SDP 9	INA	SSNF=B7	SAQR SNF=A7 AQ=Q7	DEC	IF SDP						DVG	190
DIV	78	0111 1000	DIVIDE	AQ/C(M)+Q, R=A	ISB SCB	IF B=0 SERD STOP IF A7#B7 SSMF	IF A7=0 SDP 6	IF Q=0 2's COMPA IF Q0 1's COMPA	2's COMPQ	IF A7=1 SERD STOP	IF C=0 SDP 11	DEC	SAL	IF B7=1 INA, IF B7=0 INS	IF F7=0 TFA SDP 6	IF Q7(SNF + Q000) SERD, STOP	IF B7=1 2's COMPA	IF SNF=1 2's COMPQ					191
RAO	80	1000 0000	REPLACE ADD ONE	C(M)+A+1-C(M)	CLCR ISB	INCA IF EC SCARY			TAB	ISB												192	
RSO	88	1000 1000	REPLACE SUBTRACT ONE	C(M)+A-1-C(M)	CLCR ISB	DECA IF EC SCARY			TAB	ISB												193	
INX	03	0000 0011	INCREASE INDEX	X+K-X		TBM	AXM	TRX														194	
SLA	0B	0000 1011	SHIFT LEFT ARITHMETIC	AQ ← H ← K → AQ	TBC	IF C=0 SDP 15	SAQL	DEC SDP 1														195	
SRA	10	0001 0000	SHIFT RIGHT ARITHMETIC	AQ ← H ← K → AQ	TBC	IF C=0 SDP 15	SAOR	DEC SDP 1														196	
SLL	13	0001 0011	SHIFT LEFT LOGICAL	A ← H ← K → A	TBC	IF C=0 SDP 15	SAL	DEC SDP 1														197	
SRL	18	0001 1000	SHIFT RIGHT LOGICAL	A ← H ← K → A	TBC	IF C=0 SDP 15	SAR	DEC SDP 1														198	
AND	10	0001 1001	AND	A & K → A																		199	
IOR	1A	0001 1010	INCLUSIVE OR	A + K → A																		200	
XOR	1B	0001 1011	EXCLUSIVE OR	A ⊕ K → A																		201	
BUN	90	1001 0000	BRANCH UNCONDITIONAL	M → P	TMP		SDP 15															202	
BST	98	1001 1000	BRANCH AND STOP	M → P, STOP	TMP		SDP 15															203	
BSB	A0	1010 0000	BRANCH TO SUBROUTINE	1 90+P8, P9 → C(M) 2 P0-P7 → C(M+1) 3 M → P	(90+P8) P9 → B TPHB	ISB	INCM	TPLB	ISB	INCM	TMP	SDP 15										204	
BPS	A8	1010 1000	BRANCH ON POSITIVE	IF (>0)=1, M → P	IF CC=0 TMP		SDP 15															205	
BZE	B0	1011 0000	BRANCH ON ZERO	IF (=0)=1, M → P	IF CC=0 TMP		SDP 15															206	
BNG	B8	1011 1000	BRANCH ON NEGATIVE	IF (<0)=1, M → P	IF CC=0 TMP		SDP 15															207	
BNC	C0	1100 0000	BRANCH ON NO CARRY	IF CR=0, M → P	IF CARRY=0, TMP		SDP 15															208	
BXZ	C8	1100 1000	BRANCH ON INDEX=ZERO	IF X=0, M → P	IF X=0 TMP		SDP 15															209	
SKI	08	0000 1000	SKIP ON INTERRUPT	IF INT=1, P+2K → P	IF INT=0 SDP 15	CLINT	TBC	TPM	IF C=0 SDP 15	INCM	DEC	INCM	TMP	SDP 4								210	
SKS	09	0000 1001	SKIP ON SENSE SWITCH	IF SENSE=1, P+2K → P	IF SENSE=0 SDP 15		TBC	TPM	IF C=0 SDP 15	INCM	DEC	INCM	TMP	SDP 4								211	
SKF	0A	0000 1010	SKIP ON FLAG	IF FLAG=1, P+2K → P	IF FLAG=0 SDP 15		TBC	TPM	IF C=0 SDP 15	INCM	DEC	INCM	TMP	SDP 4								212	
WDB	D0	1101 0000	WRITE DATA BLOCK	C(M)-OUTPUT, UNTIL C=0	INW	ISB		INWD WAIT	IF C=0 SDP 15			INCM DEC SDP 1										213	
MNO	D8	1101 1000	MANUAL OUTPUT	C(M)+I, UNTIL C=0		ISB		TBI WAIT	IF C=0 SDP 15			INCM DEC SDP 1										214	
RDB	E0	1110 0000	READ DATA BLOCK	INPUT+C(M), UNTIL C=0	INR WAIT	INRD WAIT	TEB	ISB	IF C=0 SDP 15			INCM DEC SDP 1										215	
RDI	E8	1110 1000	READ UNTIL INTERRUPT	INPUT+C(M), UNTIL INT=1	INR WAIT	INRD WAIT	TEB IF INT=1 SDP 15	ISB				INCM SDP 1										216	
MNI	F0	1111 0000	MANUAL INPUT	I+C(M), UNTIL C=0			WAIT	TIB	ISB	IF C=0 SDP 15		INCM DEC SDP 1										217	
OCM	11	0001 0001	OUTPUT COMMAND	SETS UP I/O INTERFACE	INOC				SDP 15													218	
SST	00	0000 0000	SENSE STATUS	STATUS WORD → A	INSS	TEB	TBA	SDP 15														219	
FLC	28	0010 1000	CLEAR FLAG	FLAG=0	CLF			SDP 15														220	
FLS	F8	1111 1000	SET FLAG	FLAG=1	STF			SDP 15														221	

CONDITION CODES

CARRY - Carry or borrow  
(>0) - Greater than zero  
(=0) - Equal to zero  
(<0) - Less than Zero

\* DPI Signal Clears Condition Codes

NOT CARRY

REGISTERS

A - Accumulator  
C - Countdown  
M - Memory Address  
P - Program Address  
Q - Quotient  
AQ - Combined accumulator and quotient  
I - Input  
X - Index  
K - Constant

SPECIAL SYMBOLS

+ Goes to  
c() the contents of  
#(K) shift right K places  
←H(K) shift left K places  
+ add  
- subtract  
X multiply  
÷ divide  
⊕ and  
⊖ exclusive or  
— NOT or complement of

HO 3AQR30524  
KDA 3035

222  
223  
224  
225