SERIAL NO. E175

RW-300 DIGITAL CONTROL COMPUTER

PROGRAMMING MANUAL

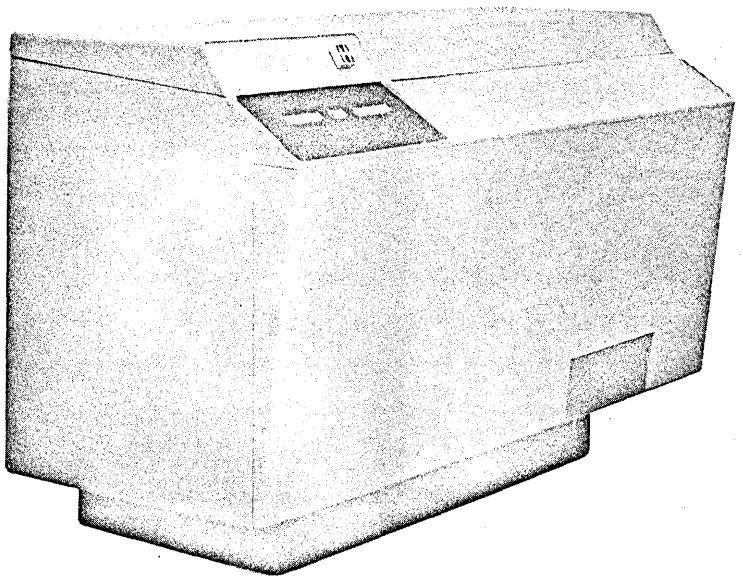
March, 1961

TRW Computers Company

a division of *Thompson Ramo Wooldridge Inc.*

8433 FALLBROOK AVENUE • CANOGA PARK, CALIFORNIA

The RW-300 Digital Control Computer



The RW-300 Digital Control Computer is used for closed-loop control of industrial processes, for automatic testing, for on-line data interpretation, and for simulations.

In real-time control and test applications, the RW-300 communicates directly with instruments which measure or sense operating variables. Under the direction of the program stored in memory, the RW-300 uses information from these instruments, performs calculations, reduces data, and generates the control signals required to fulfill process or test objectives. Fail-safe features incorporated in the system equipment and program, coupled with the inherent reliability of the RW-300, ensure dependable operation.

The computer operator is generally advised of operating conditions by typed records -- printed out either periodically or on demand. Further, the operator is able to supply the RW-300 with special data whenever the process or test must be operated under unusual conditions.

IN THIS PROGRAMMING MANUAL. . .

Section I . . . contains a general description of the RW-300 and peripheral equipment.

Section II . . . describes the operation codes used in preparing instructions for the computer.

Section III. . . tells how the instructions are assembled into programs.

Section IV. . . introduces a programming technique which makes maximum use of computer time.

Section V . . . contains detailed information relating to the digital input and output equipment used with the RW-300.

Section VI . . tells how programs are loaded into the computer under the direction of a program stored permanently in computer memory.

Section VII . . details the functions of RW-300 controls and indicators.

Section VIII . . contains reference material pertaining to number systems and scaling.

A Glossary of computer terminology is included at the end of the manual.

The last pages of the manual contain reference tables that are useful in preparing programs for the RW-300.

OTHER PUBLICATIONS ARE AVAILABLE. . .

A list of programming aids and mathematical subroutines contained in the RW-300 Program Library can be obtained by writing to TRW Computers Company.

A manual describing Optimum Programming Using Symbols (OPUS) includes instructions for preparing programs in symbolic form.

An interpretive routine is available for applying the RW-300 to general-purpose and scientific calculations.

TABLE OF CONTENTS

TABLE OF CONTENTS -- Continued

TABLE OF CONTENTS -- Continued

SECTION III - BASIC PROGRAMMING PAGE

TABLE OF CONTENTS -- Continued

SECTION IV -- OPTIMUM PROGRAMMING PAGE

TABLE OF CONTENTS -- Continued

SECTION V -- DIGITAL INPUT AND OUTPUT PAGE

TABLE OF CONTENTS -- Continued

SECTION VI -- PROGRAM LOADING PAGE

TABLE OF CONTENTS -- Continued

SECTION VII -- OPERATING CONTROLS AND INDICATORS PAGE

TABLE OF CONTENTS -- Continued

LIST OF ILLUSTRATIONS

SECTION I

GENERAL DESCRIPTION

CHARACTERISTICS

The RW-300 Digital Control Computer is a stored-program, serial computer employing a magnetic drum memory with a total capacity of 8,080 words. RW-300 computers with an expanded memory have a total memory capacity of 15,776 words. The RW-300 word is composed of 18 binary digits.

A word may represent either numerical data (17 bits, with a sign bit), or one half of a computer instruction. Two words form a complete instruction; one half of the instruction specifies the memory location of the operand, and the other half of the instruction specifies the memory location of the next instruction. The half-instruction containing the next-instruction address includes one of 21 basic operation codes, and the half-instruction containing the operand address includes an execution code. The basic instruction codes are modified by the execution code and the operand address to provide a flexible command structure.

Continuously variable voltages from measuring instruments and transducers are automatically converted to digital form and stored in the RW-300's memory--without programmed instructions. Up to 1,024 of these continuously variable "analog" input signals can be accommodated. Computational results representing control information are automatically converted from digital form to voltages which can be applied to conventional

controllers or other control devices. Up to 128 of these "analog" output signals can be provided.

Up to 511 on-off signals from different sources can be accepted by the computer, permitting input from digital clocks, switches, paper-tape readers, etc. A like number of digital output signals can be provided for the control of motors, indicators, alarms, logging typewriters, paper-tape punches, etc.

In addition to possessing the input-output capabilities necessary for real-time control, the RW-300 has been designed to provide the reliability required for continuous service.

Subroutines and interpretive routines are available to extend the application of the RW-300 to scientific and general-purpose computation.

EXTERNAL FEATURES

The console model of the RW-300 (see frontispiece) is desk size and weighs approximately 600 pounds. It operates from 120-volt, 60-cps power, and requires no special air conditioning. Power consumption is approximately 500 watts. Usually supplied in the console cabinet 36 inches high, 56 inches long, and 29 inches deep, the same basic computer can also be furnished in an upright, air-purged cabinet that is 84 inches high, 48 inches long, and 24 inches deep.

Operating controls for the console model are mounted on the sloping front edge of the cabinet and are accessible when the computer cover is closed. The operating controls are pushbuttons for turning the RW-300 on

and off, for starting, stopping, and resuming the program stored in memory, and for loading new programs into the memory. Beneath the cover, a test and maintenance panel contains controls, indicators, and displays which facilitate program check-out and computer maintenance. The operating controls and indicators are described in Section VII of this manual.

INSTRUCTION SYSTEM

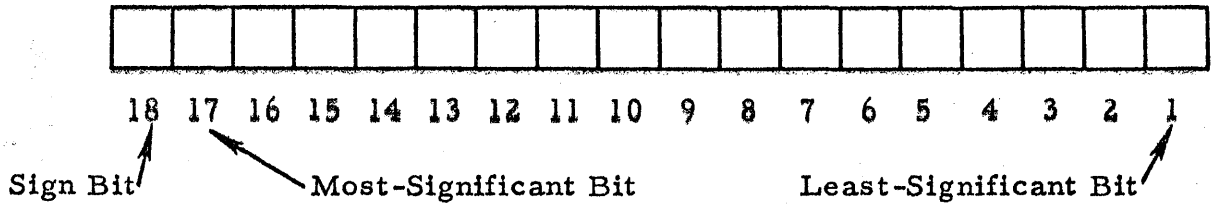
The RW-300 instruction system includes 21 basic operation codes, or commands, many of which can be modified to permit a large number of distinct operations. The operation codes control arithmetic operations, logical operations, and peripheral equipment. Section II contains a brief description of the instruction format, followed by a complete description of the operation codes.

The time required to perform arithmetic and logical operations depends upon the relative locations in memory of the instruction and the operand. Using the optimum programming techniques described in Section IV, the times (in milliseconds) required to complete typical instructions are as follows:

Add or Subtract	0.78 ms
Multiply, full length	2.99 ms
Divide, full length	3.12 ms
Transfer	0.65 ms
Load Register	0.65 ms
Store	0.78 ms

WORD LENGTH

In the RW-300, the basic unit of information is a word. A word consists of 18 binary digits, or bits: a sign bit and 17 bits of absolute magnitude. The sign bit is zero for positive and one for negative. (See Section VIII for an explanation of the binary number system.)



A word may represent: a numerical value, one half of an instruction, typewriter or punch symbols, or a bit pattern that can be used for control purposes. Section III describes the form of instruction words and data words.

MEMORY

The memory of the RW-300 is a magnetic drum, nine inches in diameter and nine inches long, which rotates at a speed of 3600 rpm. The magnetic drum may contain a basic 8,080 word memory or an expanded memory of 15,776 words.

Basic Memory

There are 7,936 words of general storage on 62 tracks of 128 words each, 32 words of fast-access storage in a circulating register, and 128 words in one track for a permanently stored load program. A word time (the time required for one word on the drum to pass a given point) is 0.13 milliseconds. The "average" access time (time required to find a word) in

general storage is 8.3 milliseconds. The 32-word circulating register, or "revolver", has an average access time of approximately 2 milliseconds.

Any of the 7,936 word locations can be written into during program loading by using control facilities on the RW-300 test and maintenance panel. In the basic computer, the 32-word revolver and 1,024 word locations in eight 128-word tracks can be written into under program control. Up to eight additional tracks of program-writable storage can be provided by adding a module to the basic computer. However, this additional program-writable storage is reduced by the number of tracks reserved for the storage of analog input data. The number of tracks reserved for analog input data does not affect the 32-word revolver nor the 1,024 words of program-writable memory available in the basic computer.

Expanded Memory

RW-300 computers with an expanded memory have 15,776 words of storage on 123 tracks. Normally, expanded memory machines have 1,536 words of program-writable storage, although up to 16 additional tracks (2,048 words) of program-writable storage can be provided by the addition of an extra module to the basic computer. The 16 additional tracks will be reduced by the number of tracks reserved for the storage of analog input data.

The method of identifying memory locations on the drum is described in connection with a description of basic programming in Section III. More detailed drum characteristics are presented in connection with the optimum programming techniques described in Section IV. Through the use of optimum

programming techniques, access time in general storage can be greatly reduced below the "average" access time of 8.3 milliseconds.

ARITHMETIC UNIT

The arithmetic unit is that part of the RW-300 which actually performs arithmetic and logical operations under control of the program stored in memory. The unit includes three circulating one-word registers (A, B, and C) on the drum. In addition, it contains a serial adder and flip-flops used for storage, time delay, and logical manipulations.

A Register

The A register, or accumulator, is located on the drum, and has a capacity of 17 bits plus sign (one word). It is the principal arithmetic register and holds the result of most operations. The A register can be loaded from memory, and the contents of the A register can be stored in the memory. It has the capability of shifting left or right one binary place per word time. In the operations of addition, subtraction, left shifting, and division, overflow from the A register is possible (i.e., the computer may attempt to put a one to the left of the 17th bit). When overflow occurs, the overflow indicator is turned on.

B Register

The B register, located on the drum, has a capacity of 17 bits plus sign. It holds, at various times, the multiplier, remainder, or least-significant half of the double-length product. As in the case of the A register, the B register can be loaded from memory, and the contents of the B register

can be stored in the memory. It has the ability to shift left one binary place per word time. When shifting left, it is coupled to the A register so that the bit in position 17 of the B register is shifted into position 1 of the A register.

C Register

The C register, located on the drum, also has a capacity of 17 bits plus sign. Its operation is not under the control of the computer's program, and therefore the programmer is seldom concerned with it. At various times, the C register holds the multiplicand, divisor, subtrahend, addend, or execution code.

Adder

The adder forms, in one digit time, the sum of one bit from the augend, one bit from the addend, and the carry bit from the previous addition. It outputs the sum bit and the carry bit. Since the sum is formed serially, bit by bit, the adder requires one word time to generate the sum of two 17-bit numbers. (This should not be confused with the length of time required to carry out an Add instruction.)

CONTROL UNIT

The control unit processes the instructions in the sequence dictated by the program stored in memory. In processing an instruction, the control unit performs the following functions:

a. Obtains instructions from memory.

b. Decodes and interprets the instructions.

c. Connects and activates other units by sending out individual commands to the other units in the proper sequence to perform the desired function.

d. Initiates the transfer of information between units.

e. Stops the execution of the program.

f. Keeps track of time so that the various parts of an instruction are executed in the appropriate sequence.

Some of the components of the control unit are described below.

Y Register

The Y register is a one-word circulating register on the drum which holds the operand address.

N Register

The N register is a one-word circulating register on the drum which holds the address of the next instruction.

Track Register

The track register is a six-bit flip-flop register which holds the track address when a program instruction refers to a memory location. The flip-flops in this register also serve other purposes; for some instructions, the track register supplements the instruction register. For digital input and output instructions, the track register addresses groups of input or output lines.

Instruction Register

The instruction register is a five-bit flip-flop register which holds the instruction code, but temporarily holds the execution code when an instruction is being read from memory.

DIGITAL INPUT AND OUTPUT

All digital input and output functions are accomplished by a single operation code which controls the transfer of information between the A register and external equipment. Digital outputs are in the form of relay-contact closures, and digital inputs are accomplished by sensing voltage changes on input lines.

The RW-300 digital input-output facilities are extremely flexible. The paragraphs which follow contain a description of the basic facilities and a description of options available to customers. A more complete description, along with programming instructions, is included in Section V.

The basic digital input-output unit provided with the RW-300 is a Flexowriter, which consists of an electric typewriter, a paper-tape and edge-punched card reader, and a paper-tape and edge-card punch. The Flexowriter can read, punch, or print at the rate of 8 characters per second. In the basic computer, seven digital input lines are reserved for accepting information from the Flexowriter reader (or from some other input device), and 18 digital input lines are available for accepting information from digital input switches or other devices dictated by the requirements of the installation.

Relays within the basic computer convert Flexowriter signal levels to levels compatible with the digital input circuits. A negative five volts applied to one or more of the other 18 digital input lines causes the computer to read a one on that line when a digital input instruction is executed, and an open or positive voltage ($\geq$2. 5 volts) causes a zero to be read.

Although a group of 18 digital-output control lines is available for system expansion, the basic computer contains only eight relays for providing digital outputs to the Flexowriter printing or punching circuits. Auxiliary control relays are provided within the basic computer for turning the Flexowriter motor on and off; for initiating printing when a logging typewriter is used; for initiating punching when a high-speed paper-tape punch is used; and for accepting "ready" signals from these output devices.

A variety of optional digital input-output equipment is available. Additional typewriters, including line printers, can be supplied to log out raw data and finished computations. A Ferranti high-speed paper-tape reader (60 characters/sec) and a Teletype paper-tape punch (60 characters/sec) can alternate with or be substituted for the Flexowriter reader and punch.

When used in control applications, the computer's basic digital input capabilities can be expanded to accept 29 additional groups of digital input lines (with 18 lines in each group) for a maximum of 540 digital input lines. In an expanded digital input system, a zero is read into the computer when an input line is grounded; a one is read when an input line is open.

Basic digital output capabilities can be expanded to provide 29 additional groups of output lines (with 18 lines in each group) for a maximum of 540 digital output lines.

The additional equipment required for expansion of digital input and output facilities is accommodated in an operator's desk-type console shown in figure 1-1. The operator is shown with the Flexowriter, and a logging typewriter is shown on top of the console control panel. Section V includes a functional description of console operating controls.

ANALOG INPUT AND OUTPUT

Conversion Capabilities

The analog input-output equipment contained in the basic computer converts voltages from measuring instruments into digital form, and converts digital information from computer memory into voltage or current at rates up to 3,840 conversions per second. Throughout this manual, the inputs from measuring instruments and transducers are called "analog inputs" and the outputs to controllers, recording instruments, etc., are called "analog outputs".

The basic computer accommodates the circuits required to accept up to 32 analog inputs and provide up to 36 analog outputs. By installing relays and associated circuits in an upright cabinet similar to that shown in figure 1-1, the number of analog inputs can be expanded to 1,024; installing additional analog output modules in an output cabinet enables the number of analog outputs to be expanded to 128.

In most control installations, the auxiliary analog input cabinet contains an oven for maintaining at a constant temperature the junctions between the thermocouple leads and the system wiring. The auxiliary analog

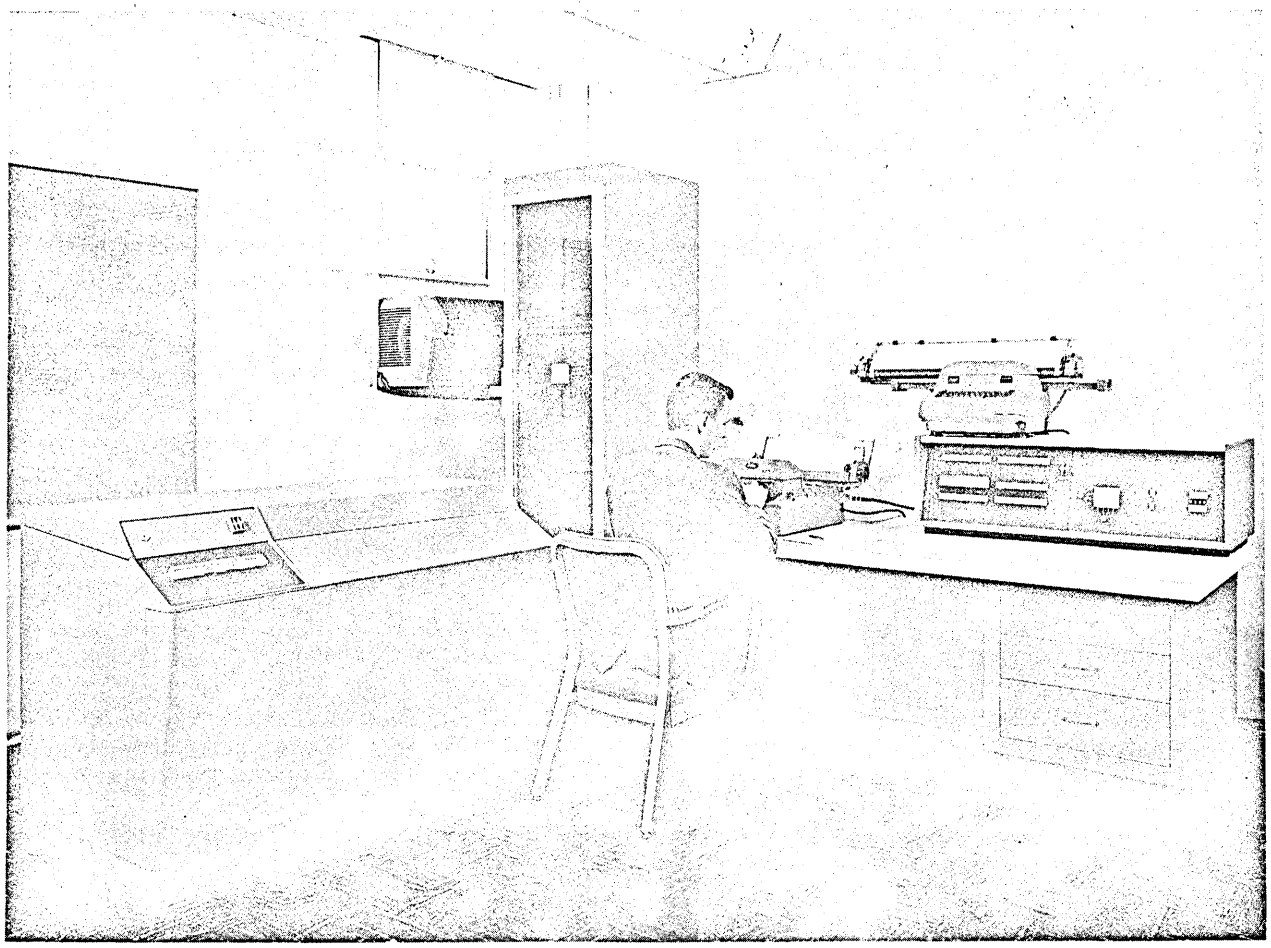


Figure 1-1

RW-300 and Peripheral Equipment

cabinet also accommodates amplifiers for raising the amplitude of low-level instrument signals (e.g., thermocouple voltages) to a level compatible with the computer's analog-to-digital conversion circuits. The cabinet also contains filters for removing hum and noise from incoming instrument lines.

Equipment in the computer (and in the auxiliary analog cabinet) operates as an independent subsystem, so that information from

measuring instruments is constantly converted to digital form, and the latest digital representation is stored in memory without programmed instructions. Similarly, when new operating parameters have been computed and stored in memory by the program, these latest values are converted to analog form and transmitted to control instrumentation. The digital-to-analog conversions are also performed automatically, and do not have to be programmed. The analog converter is time-shared among inputs and outputs.

Analog outputs are updated (adjusted to correspond to a number stored in memory under program control) automatically at least once every 1/30 second. The frequency at which analog inputs are updated (input voltages converted to digital form and stored in memory) depends upon the particular installation. For a basic system (32 voltage gates, 0 relays), the inputs can be updated every 1/30 second; for a system employing 1,024 inputs, the input information stored in memory can be updated every second, but under severe noise conditions, where filters are employed, the inputs may be updated every two seconds, four seconds, or eight seconds to allow a stabilization period for the noise filters. Actually, the analog input-output system is flexible and is furnished to meet the needs of each application. Analog input information can be converted and stored at the maximum rate of 3,840 samples per second. Longer delays between input samplings are also possible to provide a stabilization period for transducers.

Systems can be changed or expanded by means of field modifications. Thus, a system can be installed initially as a computing data logger and

later, by connecting the analog outputs of the computer to controllers, can be expanded to an automatic control system.

Analog signals other than d-c voltages, such as pressures, a-c voltages, etc., are converted to d-c voltages by the use of transducers or special converters. Amplifiers and filters are provided for low-level signals from thermocouples and strain gages. Analog-to-digital and digital-to-analog conversions are accurate to ± 0.05 percent of full scale.

Input Conversion Range and Number Representation

Two types of analog input capabilities are available: (1) a unipolar converter which converts to digital form voltages in the range from 0 to + 10.23 volts d-c, and (2) a bi-polar converter which converts to digital form voltages in the range from -10.23 volts to +10.23 volts.

When converted to digital form, the analog signals are represented by 10 binary digits. Because the least-significant digit represents a conversion resolution of 10 millivolts, the conversions are accurate to ± 5 millivolts, or ± 0.05 percent of full scale.

In a unipolar system, the digital equivalent of the analog output signal is contained in bit-positions 8 through 17 of a computer word, with the most-significant bit in position 17. Using a bi-polar system, the magnitude bits are in bit-positions 8 through 17, and the sign bit is in bit-position 18.

As an example, assume that one of the variables to be measured is temperature, and assume that the range of values for the temperature reading is 200 to 700 degrees Fahrenheit. A continuous analog representation

of this variable can be developed by a thermocouple and an amplifier to take the form of a voltage with a range of 0 to 10.23 volts. The analog subsystem will sample this voltage, develop a 10-digit binary representation of the voltage, and store it in a specified location on the magnetic drum. The most significant of these 10 bits will be in bit-position 17 of a computer word, the least significant in bit-position 8. In a unipolar system, the sign bit and bit positions 7 through 1 are set to zero for analog inputs and ignored for analog outputs. Since the exact memory location for each input which is converted to digital form will be known to the programmer, it will be necessary for the program to refer only to that location and interpret the binary number stored.

Assuming a linear relationship between degrees Fahrenheit and voltage, a table of values for these representations could be as follows:

Degrees Fahrenheit	Volts	Binary Representation
200.0	0.00	00 00 00 00 00
200.5	0.01	00 00 00 00 01
205.0	0.10	00 00 00 10 10
250.0	1.00	00 01 10 01 00
325.0	2.50	00 11 11 10 10
450.0	5.00	01 11 11 01 00
575.0	7.50	10 11 10 11 10
700.0	10.00	11 11 10 10 00

Input Storage Locations

Analog inputs are stored in tracks 08 through 15 of the drum. If 128 inputs are required, only track 08 would be used, but for 1,024 inputs, tracks 08 through 15 would be required. Not all sectors of a track need be used for inputs. The exact number of sectors and tracks used depends upon the number and type of inputs required for a particular system, and assignments are made to minimize equipment requirements, cost, and memory space for the particular application.

In a track reserved for analog input data, unassigned sectors cannot be used for general program storage.

Output Conversion Range and Number Representation

Each analog output is capable of controlling the voltage or current in its load to an accuracy of ± 0.05 percent, and is capable of supplying a current of 5 milliamperes to a transducer or controller.

A maximum analog output current of 20 milliamperes can be supplied at the customer's option.

A binary number to be converted to analog form is written into a specific memory location by the program. The binary number for analog outputs occupies bit-positions 8 through 17 of the computer word, with the most-significant digit in bit-position 17, and the least-significant digit in bit-position 8.

For "voltage" type analog outputs, the voltage applied to a controller or transducer may be from 0 to 10.23 volts. To apply 10.23 volts to the load, bit-positions 8 through 17 of the corresponding memory sector are filled with ones under program control. The relationship between a number in memory

and an analog output voltage is linear, so that the output voltage is equal to the decimal equivalent of the number in memory (taking bit-position 8 as the least-significant bit) times 0.01 volt. Three of the 1,023 possible values are tabulated below.

Binary Number in Analog Output Sector of Memory	Voltage Across Load (max. current = 5 ma)
0111111111110000000	10.23 volts d-c
0100000000000000000	5.12 volts d-c
0000000011000000000	0.06 volts d-c

In some applications the voltage applied to the load (controller) is not as important as the current through the load. For these applications a precision resistor is placed in series with the load, and the analog output is connected to control the voltage across the precision resistor, thereby controlling the current through the load. Although the relationship between the load current and the number in memory is linear, the number in memory must be scaled an amount determined by the ratio of the load/metering resistance. If the load and metering resistances are equal, one half of the analog output voltage appears across the load, and the other half appears across the metering resistance. In this case, maximum current through the load and metering resistances is obtained when the program writes:

```
18                  1
 0100000000000000000
```

in the appropriate analog output sector. Writing a larger number into the analog output sector could cause a nonlinear relationship between the output

and the number in memory, and in some extreme cases might damage that specific analog output circuit.

The exact configuration of the analog output circuits (voltage output or current output) is determined by the type of controller, transducer, instrument, etc., that forms the load for the analog output. During the planning stage of an installation, the programmer is advised of the limits, linearity, response time, and other factors that affect the program. The sector numbers that will control analog outputs are also determined during the planning stage.

Output Storage Locations

Analog outputs are taken from track 07 by the analog converter. Track 07 is program writable; any portion not used for writing analog outputs may be used by the program for other storage purposes.

MAGNETIC TAPE UNIT

Introduction

The RW-300 Magnetic Tape Unit provides practically unlimited memory capacity for data reduction and control applications requiring more storage than the basic drum memory provides. The computer and magnetic tape unit form a system that is used for on-line data acquisition and processing, for recording historical data, for table look-up, and for preparing tapes for analysis by an IBM 700-series data processing system. Data is then transferred from the drum to the tape for temporary or permanent storage. For interpretation, analysis, and presentation with the RW-300 computer, raw

data is transferred back to the computer. Auxiliary programs and subroutines are also transferred from tape to the drum. Transfer of digital data between the tape unit and the computer is accomplished through a magnetic core buffer.

Two models of the RW-300 Magnetic Tape Unit are available. If the "standard" unit is employed, data stored on the tapes must be transferred back to the RW-300 for reduction and presentation. Another model of the RW-300 Magnetic Tape Unit, called the "compatible" tape unit, prepares magnetic tapes in a format that is acceptable to some models of IBM magnetic tape units. Thus, data stored by a "compatible" RW-300 Magnetic Tape Unit can be processed either by an IBM system, or by the RW-300.

Description

The magnetic tape unit consists of one magnetic core buffer and from one to eight magnetic tape transports. Figure 1-2 shows the RW-300 and one element of a magnetic tape unit. In the figure, the six-foot rack contains the buffer, buffer controls, and one tape transport with associated writing and power supply circuits. Similar cabinets are used to accommodate additional tape transports and associated writing and power supply circuits.

The buffer has a capacity of 128 computer words of 18 bits each. This amount of information is commonly called a "block" in the standard tape unit and a "record" in a compatible tape unit.

In the "standard" magnetic tape unit, a tape calibrator is used to establish specific writable blocks (no defects) on the tape, and these blocks may be written into individually, without disturbing the contents of other

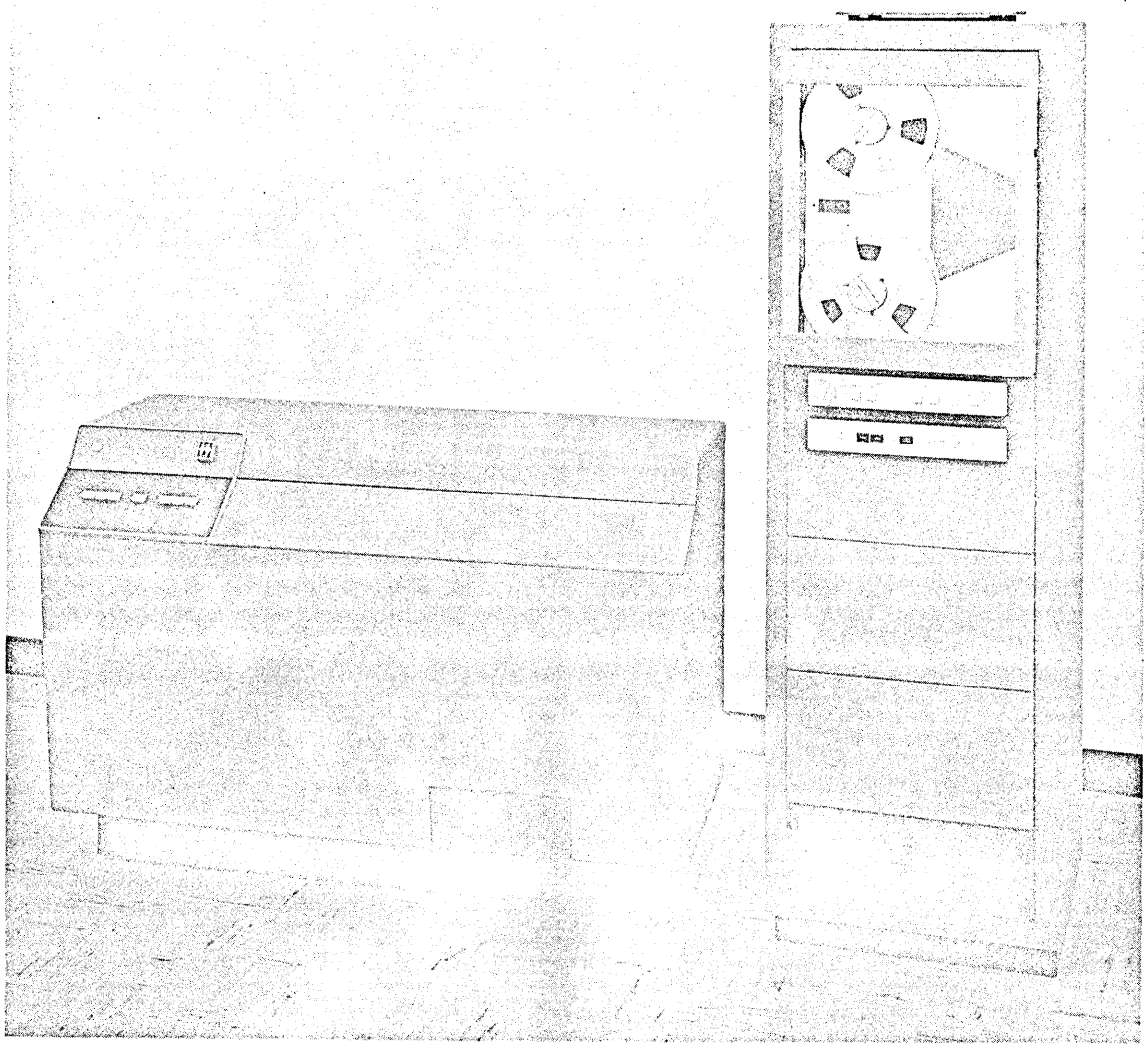


Figure 1-2

RW-300 and Magnetic Tape Unit

blocks on the tape. The tape calibrator automatically rejects any areas of tape that are of low quality.

Format characteristics of a "compatible" magnetic tape unit require that the entire length of the tape be used for sequential storage

of all records. Therefore, when a record is rewritten on a tape in the "compatible" system, all succeeding records within a "file" must also be rewritten.

Each transport holds one reel of tape with a capacity of approximately 730,000 computer words for the "standard" unit and 1,000,000 computer words for the "compatible" unit. The magnetic tape unit transfers data at the maximum usable rate of 1,536 words per second. An eight-transport system can record data automatically at the maximum rate for nearly 70 minutes. Since the tape reels are easily replaced, the data-recording period may be extended indefinitely.

The transports are modified Ampex FR-400 digital tape handlers operating at 75 inches per second in the write, search, and read modes and rewinding at 160 inches per second. Each transport accommodates 2400 feet of standard 1/2-inch magnetic tape wound on a 10 1/2-inch reel.

The buffer is a specially designed assembly, which in addition to the core storage includes control, parity generation and checking, error sensing, timing, and power circuits. Parity checking and error surveillance are provided to prevent loss and inaccuracy of data during transfers to and from magnetic tape.

Operation

The computer program controls the normal, automatic operation of the magnetic tape unit. The magnetic tape unit furnishes error-condition signals to aid this control, and has manual controls and indicators for non-automatic operations.

A single computer command with a number of variations determines the mode of operation of the magnetic tape unit. This command causes the unit to transfer data, search, rewind, or back space. Data is transferred as follows:

a. from computer to buffer

b. from buffer to tape

c. from tape to buffer

d. from buffer to computer

To locate data recorded on tape, the computer searches the tape while it is traveling either forward or in reverse until a specified record address (first word of the record) is found.

The tape unit provides the computer with indications of six conditions that interfere with the transfer of data. These conditions are:

a. buffer power supplies inoperative

b. buffer in use

c. tape error (parity violation)

d. transport inoperative

e. end of reel

f. write amplifiers disabled

During the last three conditions, the unit causes the computer to stop if it attempts to transfer data.

Computer commands related to the magnetic tape unit are described briefly in Section II, and more complete operating information is contained in a separate manual.

Specifications

Number of buffers: one.

Number of transports: one to eight.

Power: 120 volts, 60 cps.

Type of buffer: magnetic core.

Physical characteristics of buffer: mounted in one tape transport.

Capacity of buffer: 128 words of 18 bits each (one block), or 64 words of 36 bits each (one "record" in a "compatible" system).

Type of tape transport: modified Ampex FR-400 Digital Tape Handler.

Dimensions of transport: 72 in. high, 23 in. wide, and 24 in. deep.

Recording Density	"Standard" Tape Unit 150 lines per inch, 8 bits per line	"Compatible" Tape Unit 200 lines per inch, 7 bits per line
Number of data bits	6	6
Number of timing bits	1	0
Number of parity bits	1	1
Tape width	1/2 in.	1/2 in.
Tape length	2400 feet	2400 feet
Tape reel	10 1/2 in.	10 1/2 in.
Tape speed forward or reverse	75 in./sec., within 1%	75 in./sec., within 1%
Tape rewind speed	160 in./sec.	160 in./sec.
Maximum bit rate	11.25 kps	15 kps
Minimum bit time	89 μs.	66 μs.

Magnetic Tape Unit -- Continued

Recording Density	"Standard" Tape Unit 150 lines per inch, 8 bits per line	"Compatible" Tape Unit 200 lines per inch, 7 bits per line
Record length (128 words)	2. 58 in.	1. 92 in.
Inter-record spacing	2. 72 in.	0. 75 in.
Records per 2400-foot tape	5440	10, 800
Allowable record numbers	2^{18}	2^{18}
Record time	34. 4 ms.	25. 6 ms.
Maximum usable data transfer rate	1, 536 words/sec.	1, 536 words/sec.
Number of heads	8 read; 8 write	7 read; 7 write

SECTION II

INSTRUCTIONS

INTRODUCTION

An instruction in the RW-300 is composed of two computer words. The first word contains an execution code and an operand address. The second word contains an operation code and the address of the next instruction to be performed. An instruction is written by the programmer in the following form:

execution code	operand address	operation code	next-instruction address

The exact form taken by the instruction in the memory and the format for listing instructions are described in Section III.

There are 21 basic operation codes in the instruction repertoire of the RW-300. However, many more than 21 commands are available, because computer response to certain operation codes is modified by the execution code and the operand address.

In the paragraphs which follow, each of the operation codes is described in conjunction with the variations made possible by execution codes from 00 through 31 and by operand track addresses from 00 through 63. The sector number associated with an operand address does not modify the operation codes, nor are the operation codes modified by the track and sector numbers of the next-instruction address. In describing the operations, the following notation is used:

a. "A" is used to designate the A register.

b. "B" is used to designate the B register.

c. "M" is used to designate the memory location specified by the operand address.

d. Parentheses are used to designate "contents of"; (A) means the contents of the A register.

e. Arrows are used to designate "replaces"; (A)→(B) means the contents of the A register replace the contents of the B register.

Figure 2-1, located at the end of this section of the manual, contains examples which show how different operation codes affect the contents of arithmetic registers and memory.

LOAD A (M)→(A)

Decimal Code:	29
Mnemonic Code:	LA
Operand Address:	specifies the location of the number (M) which will replace the contents of the A register.
Execution Code:	does not affect operation--use ~~any number from 00 through 31~~ 00.

The contents of M replace the contents of the A register. The B register is unchanged.

LOAD B (M)→(B)

Decimal Code:	07
Mnemonic Code:	LB

Operand Address: specifies the location of the number (M) which will replace the contents of the B register.

Execution Code: does not affect operation--use ~~any number from 00 through 31~~ 00.

The contents of M replace the contents of the B register. The A register is unchanged.

LOAD A NEGATIVE $-(M) \longrightarrow (A)$

Decimal Code: 21

Mnemonic Code: LN

Operand Address: specifies the location of the number (M) which, with sign changed, will replace the contents of the A register.

Execution Code: does not affect operation--use ~~any number from 00 through 31~~ 00.

The contents of M, with sign changed, replace the contents of the A register. The B register is unchanged.

STORE A $(A) \longrightarrow (M)$

Decimal Code: 30

Mnemonic Code: SA

Operand Address: specifies the location M in which the number in the A register will be stored

Execution Code: does not affect operation--use ~~any~~ 00 ~~number from 00 through 31~~.

The contents of the A register replace the contents of M. The A register and the B register are unchanged.

STORE B (B) ⟶ (M)

Decimal Code: 20

Mnemonic Code: SB

Operand Address: specifies the location M in which the number in the B register will be stored.

Execution Code: does not affect operation--use ~~any~~ 00 ~~number from 00 through 31~~.

The contents of the B register replace the contents of M. The A register and the B register are unchanged.

ADD (A) + (M) ⟶ (A)

Decimal Code: 25

Mnemonic Code: A

Operand Address: specifies location of addend, (M).

Execution Code: does not affect operation--use ~~any~~ 00 ~~number from 00 through 31~~.

The contents of M are added algebraically to the contents of the A register; the signed sum replaces the previous contents of the A register. If the sum is zero, the sign of the A register is unchanged. The B register is unchanged.

If the number of significant A-register bits and/or the number of significant M bits is 17, an addition command can result in a carry bit that cannot be accommodated in the A register. The carry bit, representing the most-significant bit of the sum, "overflows" the A register and is lost. Overflow does not halt the computer, but turns on the overflow indicator. Overflow can be detected by use of the transfer command: Transfer on Overflow.

To obtain meaningful results from an Add instruction, it is necessary that the augend (A) and the addend (M) have the same scale factor. Scaling considerations are described in Section VIII.

SUBTRACT (A) - (M) → (A)

Decimal Code:	24
Mnemonic Code:	S
Operand Address:	specifies location of subtrahend, (M).
Execution Code:	does not affect operation--use 00 ~~any~~ ~~number from 00 through 31~~.

The contents of M are subtracted algebraically from the contents of the A register; the signed difference replaces the previous contents of the A register. If the difference is zero, the sign of the A register is unchanged. The B register is unchanged.

Overflow can occur under the same conditions as specified above for the Add command. The minuend and subtrahend should have the same scale factor, as discussed in Section VIII.

SHIFT

Decimal Code: 01

Mnemonic Code: SH

Operand Address: specifies the type of shift.

Execution Code: specifies number of places to be shifted.

The track number in the operand address specifies the type of shift as follows:

Operand Track Address	Type of Shift
Track 00 through 15	shift (A) right; (B) unchanged.
Track 16 through 31	shift (A) left; (B) unchanged.
Track 48 through 63	shift (A) left; (B) left into A.

The execution code specifies the number of places to be shifted. Any number from 00 through 31 can be used to obtain an instruction for shifting from 00 through 31 places. The sign bits of the A and B registers are unchanged by the Shift command.

The two types of shifts involving only the A register are open-ended, so that bits shifted off either the right or the left end of the A register are lost. When A is shifted right, the bit positions vacated at the left end of the A register are filled with zeros. When A is shifted left, the bit positions vacated at the right end of the A register are filled with zeros; any non-zero

bits shifted off the left end of the A register turn on the overflow indicator. (Overflow can be detected by using the Transfer on Overflow command.)

When both the A register and B register are shifted left, the two registers are coupled together so that the most-significant (17th) bit of the B register moves to the least-significant (1st) bit position of the A register. Bit positions vacated at the right end of the B register are filled with zeros. Bits shifted off the left end of the A register are lost, but non-zero bits turn on the overflow indicator.

The Shift command may also be used to multiply or divide by powers of two. Shifting register contents left one place is equivalent to multiplying by two, and shifting right one place divides by two. The use of the execution code to adjust the scale factor is described in conjunction with scaling in Section VIII.

TRANSFER ON NEGATIVE	Decimal Code:	09
	Mnemonic Code:	TN
	Operand Address:	becomes the next-instruction address if the A register holds a negative number.
	Execution Code:	does not affect operation--use ~~any number from 00 through~~ 00 31.

Transfer on Negative -- Continued

If the sign of the A register is negative, the operand address becomes the next-instruction address. Otherwise, the next-instruction address remains as specified in the program listing. The A register and the B register are unchanged. A zero with a negative sign causes transfer to occur.

TRANSFER ON ZERO	Decimal Code:	11
	Mnemonic Code:	TZ
	Operand Address:	becomes the next-instruction address if the number in the A register is zero.
	Execution Code:	does not affect operation--use 00 ~~any number from 00 through 31.~~

If the contents of the A register are zero (plus or minus), the operand address becomes the next-instruction address. Otherwise, the next-instruction address remains as specified in the program listing. The A register and the B register are unchanged.

TRANSFER ON OVERFLOW	Decimal Code:	10
	Mnemonic Code:	TF
	Operand Address:	becomes the next-instruction address if the overflow indicator is on.

	Execution Code:	does not affect operation--use ~~any~~ 00 ~~number from 00 through~~ .

If the overflow indicator has been turned on since the last Transfer on Overflow command, the operand address becomes the next-instruction address, and the overflow indicator is turned off. Otherwise, the next-instruction address remains as specified in the program listing. The A register and the B register are unchanged.

COMPARE MAGNITUDE	Decimal Code:	15
	Mnemonic Code:	CM
	Operand Address:	specifies the location of the number (M) to be compared with the number in the A register.
	Execution Code:	if the comparison of absolute values indicates that the number in the A register is smaller than the number in memory, the next-instruction address if formed by adding the execution code to the

$(A)_{initial} \longrightarrow (B)$

$\left|(A)\right|_{initial} - \left|(M)\right| \longrightarrow (A)_{final}$

next-instruction address specified in the program listing.

The magnitude of (A) is compared with the magnitude of (M) by subtracting the absolute value of the number in the specified memory location from the absolute value of the number in the A register. At the conclusion of the instruction, the B register holds the number originally in the A register, and the A register holds the (signed) difference between the absolute values of (A) and (M). Overflow cannot occur.

If $\left|(A)\right|$ is equal to or greater than $\left|(M)\right|$, the final contents of the A register will be positive, and the computer will read the next instruction from the location specified as the next-instruction address in the program listing.

If $\left|(A)\right|$ is less than $\left|(M)\right|$, the final contents of the A register will be negative, and the computer will read the next instruction from an address formed by adding the execution code to the next-instruction address specified in the program listing. The addition performed to form a new next-instruction address is modulo 128, meaning that the track number of the next-instruction address will be unchanged, and only the sector number will be modified. (Example: if an execution code of 20 is added to sector 120, the new sector number will be 20 + 120 - 128, or sector 12 of the track originally specified.)

EXTRACT $(A) \otimes (M) \longrightarrow (A)$

Decimal Code: 05

Mnemonic Code: EX

Operand Address: specifies the location of the number (M) to be used for logical multiplication of the contents of the A register.

Execution Code: does not affect operation--use ~~any~~ 00 ~~number from 00 through 31~~.

The logical product of the contents of the A register and M replaces the contents of the A register. All 18 bits are used. Each bit (including the sign bit) of A is matched with the corresponding bit of M. When the corresponding bits of both A and M are ones, a one remains in that position of A. When the corresponding bit of either A or M is zero, a zero replaces the contents of that position in A. (The Extract operation is described in Section V.) The B register is unchanged.

MERGE (A) $\oplus$ (M) $\longrightarrow$ (A)

Decimal Code: 31

Mnemonic Code: MG

Operand Address: specifies the location of the number (M) to be logically added to the contents of the A register.

Execution Code: does not affect operation--use ~~any~~ 00 ~~number from 00 through 31~~.

The logical sum of the contents of the A register and M replaces the contents of the A register. Logical addition is performed bit-by-bit, and all 18 bits are used. Each bit of A is matched with the corresponding bit of M.

When the corresponding bit of either A or M, or both, is a one, a one replaces the contents of that position in A. When the corresponding bits of both A and M are zeros, a zero replaces the contents of that position in A. The B register is unchanged.

SWITCH

Decimal Code: 02

Mnemonic Code: SW

Operand Address: specifies the type of switch

Execution Code: does not affect operation--use ~~any~~ 00 ~~number from 00 through 31~~.

Operand Track Address	Type of Switch
Track 00 through 15	(A) ⟶ (B); (A) unchanged
Track 16 through 31	(B) ⟶ (A); (B) unchanged
Track 32 through 47	(A) ⟷ (B)
Track 48 through 63	0 ⟶ (A), (B)

Four types of switches are possible, depending upon the track number specified in the operand address.

For any track address from 00 through 15, the contents of the A register replace the contents of the B register, and the A register remains unchanged.

For any track address from 16 through 31, the contents of the B register replace the contents of the A register, and the B register is unchanged.

For any track address from 32 through 47, the contents of the A and B registers are interchanged.

For any track address from 48 through 63, the contents of the A and B registers are cleared.

STOP	(A) unchanged, (B) unchanged	
	Decimal Code:	00
	Mnemonic Code:	SP
	Operand Address:	does not affect operation--use ~~any track number from 00 through 63 and any sector number from 00 through 127.~~ 00-00
	Execution Code:	does not affect operation--use ~~any number from 00 through 31.~~ 00

The Stop command halts the program. If the RESUME button on the operator's control panel is pressed, the computer reads the next instruction from the location specified as the next-instruction address in the program listing. The A and B registers are unchanged.

DIGITAL	Decimal Code:	06
	Mnemonic Code:	DG
	Operand Address:	specifies the input or output device.
	Execution Code:	for digital input, specifies number of bits taken in and their position

in the A register; for "one-bit" digital outputs, specifies on or off control.

Operand Track Address	Type of Digital Command
00 through 31	Digital output from the A register
32 through 63	Digital input to the A register

Inputs from digital devices ~~(switches, Flexowriter, tape readers, etc.)~~ replace the contents of the A register.

Eighteen input lines form an "input group" that is assigned a specific track address. An execution code of 18 results in all 18 lines being read into the A register, replacing the previous contents of the A register.

Outputs to digital devices ~~(lights, alarms, controls, Flexowriter, logging typewriter, etc.)~~ are controlled by the A register. The contents of the A register are unchanged by a Digital output command. Eighteen output lines form an "output group" that is assigned a specific track address, and each line in the addressed group has a corresponding bit position in the A register.

In the case of "multi-bit" outputs, all 18 lines in an addressed group are set on-or-off to correspond to the one-or-zero bit pattern in the A register.

In the case of "one-bit" outputs, one or more lines in an addressed group can be controlled without disturbing the other lines in the group. A one is placed in the A register bit position(s) of the line(s) to be affected by a

specific Digital output instruction. An odd-numbered execution code turns the affected line(s) on; an even-numbered execution code turns the affected line(s) off.

The use of the Digital command in conjunction with specific input and output equipment is described in Section V. That section also lists track addresses commonly assigned to specific equipment. Note that the "track" address assigned to specific equipment has no relationship to drum tracks in computer memory. The track number in the operand address of a Digital instruction does not refer to memory at all, and only has significance in the selection of the input or output device.

NO OPERATION (A) unchanged, (B) unchanged

Decimal Code:	03
Mnemonic Code:	NO
Operand Address:	does not affect operation--use 00-00
Execution Code:	does not affect operation--use 00

This command causes the computer to transfer unconditionally to the next-instruction address in the program listing. The contents of the A register and the B register are not affected. The No Operation code is useful in conjunction with exit instructions in arithmetic subroutines.

TAPE	Decimal Code:	22
	Mnemonic Code:	TA
	Operand Address:	specifies the track of the computer memory that is to be transferred to magnetic tape; specifies the mode of operation.
	Execution Code:	specifies whether information from the tape is to be placed in track 14 or track 15 of the computer memory; addresses a specific tape transport.

~~The characteristics and applications of the magnetic tape unit are described briefly in Section I.~~

Information is transferred between the computer drum and magnetic tape through the magnetic tape unit's buffer. Information is transferred in blocks of 128 words each. An identifying word, recorded as the first word in a block of information on the magnetic tape, permits the tapes to be searched for specific blocks of information.

~~Detailed programming and operating information for the RW-300 Magnetic Tape Unit is contained in a separate manual.~~

MULTIPLY	(A) X (M) ⟶ (A, B)	
	Decimal Code:	16
	Mnemonic Code:	M
	Operand Address:	specifies the location of the multiplier (M).
	Execution Code:	specifies the number of multiplier bits used to derive the product, and specifies the number of product bits in the B register.

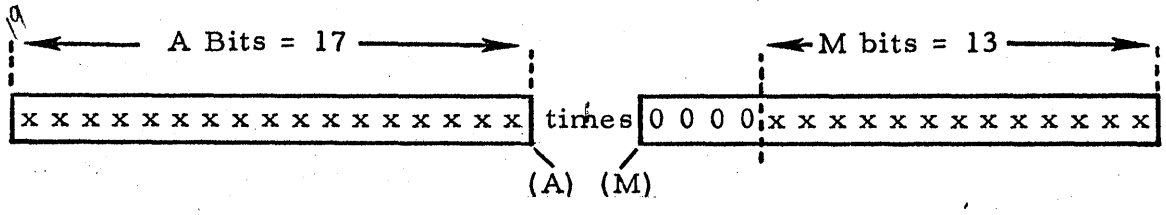
The contents of the A register are multiplied by the contents of M. The original contents of the A and B registers are replaced by the product. The signs of the A and B registers agree, and are the algebraic sign of the product. To obtain a meaningful product, the execution code "E" must be equal to, or greater than, the number of significant multiplier bits in M. Overflow cannot occur.

The number of product bits generated by a Multiply instruction is equal to the total number of significant bits in the multiplicand (A) and multiplier (M). (Depending upon the magnitude of the multiplier and multiplicand, the number of significant product bits may be one less than the sum of multiplicand and multiplier bits.) To accommodate all product bits, the B register serves as an extension of the A register. There will always be "E" low-order product bits in the B register; the high-order product bits will be in the A

register. Thus, the execution code can be used to control the apportionment of product bits between the A and B registers.

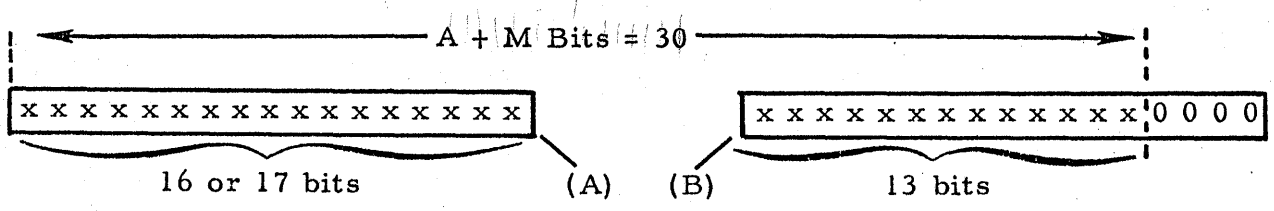
Section VIII of this manual tells how the execution code affects the scale factor of the product. The paragraphs below contain examples which show how different execution codes affect the results of a Multiply instruction. In the examples, "0000...." indicates leading zeros; "xxxx...." indicates significant bits, which may be some combination of ones and zeros. Sign bits are not shown.

In the following example, the number of multiplicand (A) bits is 17, and the number of multiplier (M) bits is 13.



To obtain a meaningful product, the execution code "E" must always be equal to or greater than the number of multiplier bits. In the example, there are a total of 30 bits in the multiplier and multiplicand, so the number of product bits will be either 29 or 30. With a "minimum" execution code of 13, the B register will contain 13 of the low-order product bits.

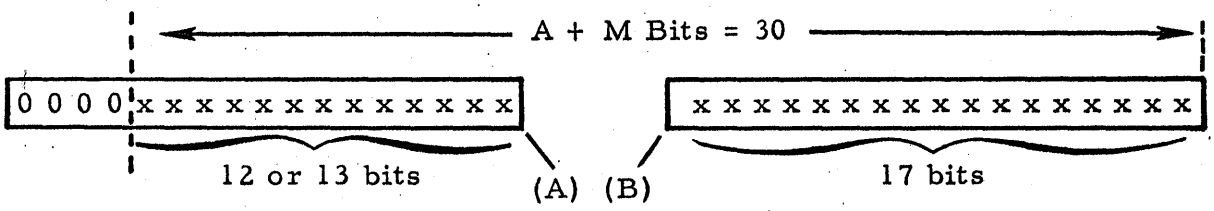
For E = 13, the product is:



Depending upon the magnitude of the multiplier and the multiplicand in the example, either 17 or 16 bits will be in the A register. For execution codes

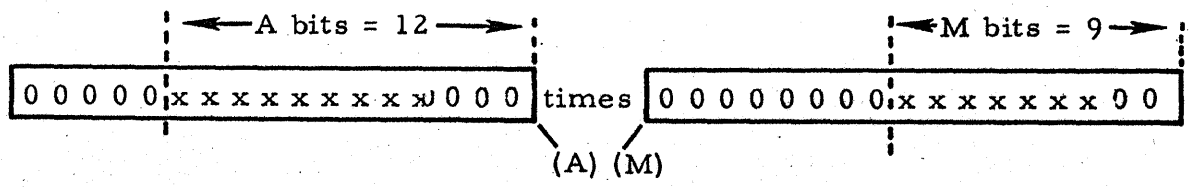
greater than minimum, the product bits are shifted right; fewer product bits are obtained in the A register, and more product bits are obtained in the B register. An execution code of 17 always results in complete multiplication using all 17 multiplier bits.

For E = 17, the product is:

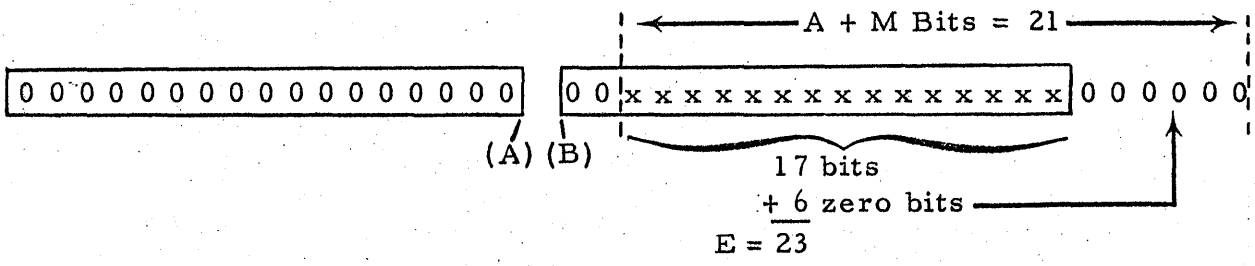


In the above example, it was shown that a minimum execution code (an execution code equal to the number of multiplier bits) results in a maximum number of product bits in the A register. In the example, there were 17 multiplicand bits in the A register before multiplication, and either 16 or 17 product bits in the A register after multiplication. The following generalization applies to any Multiply instruction: After multiplication, the number of significant bits in the A register remains the same, or is reduced by one, if the execution code does not exceed the number of multiplier bits. Also, "E" low-order product bits are always shifted into the B register.

In the general case, the maximum execution code is 17. However, higher execution codes can be used to shift insignificant product bits (zeros) off the right end of the B register. This is illustrated by the following example:

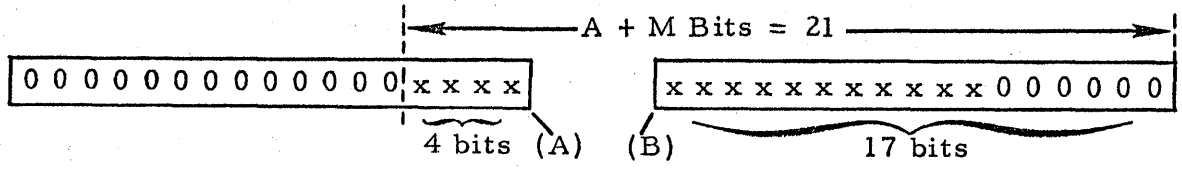


There are a total of 21 bits in the multiplier and multiplicand, so the number of product bits will be either 20 or 21. Also, there are a total of 6 zero bits in the least-significant bit positions of the multiplier and multiplicand; there will be at least 6 zeros in the least-significant bit positions of the product. The 6 zero bits can be shifted off the right end of the B register by using an execution code E = 17 + 6 = 23. For E = 23, the product is:

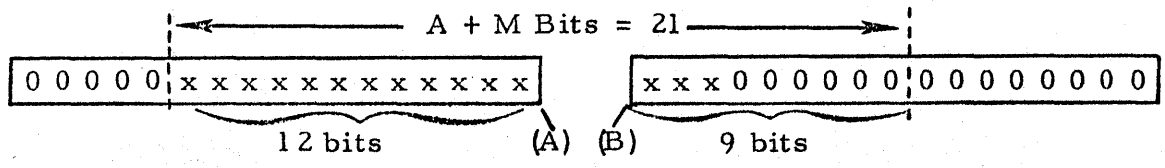


Caution must be used in assigning an execution code greater than 17. If any significant product bits (ones) are shifted off the right end of the B register, the product is not simply truncated -- the bits that remain in the A and B registers will be meaningless.

An execution code of 17 always results in a meaningful product. For E = 17, the product is:



An execution code equal to the number of significant bits in the multiplier always results in a meaningful product. For E = 9, the product is:



DIVIDE (A) ÷ (M) → (A); remainder → (B)

Decimal Code: 26

Mnemonic Code: D

Operand Address: specifies the location of the divisor (M).

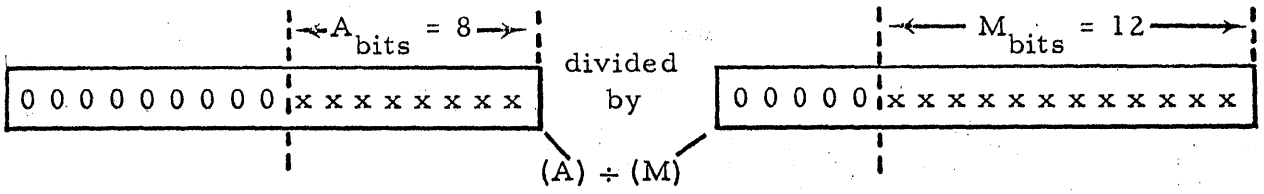
Execution Code: specifies the number of quotient bits.

The contents of the A register are divided by the contents of M. The original contents of the A register are replaced by the quotient, and the original contents of the B register are replaced by the remainder. The sign of the A register is the algebraic sign of the quotient; the B register takes the sign of the dividend. If the ratio of dividend to divisor is one or greater, the overflow indicator is turned on.

Basically, a Divide command yields one integer quotient bit, followed by a series of fractional quotient bits. Thus, the maximum binary quotient is 1.11111....., and the ratio of the dividend (A) to divisor (M) must be less than two. If (A) ÷ (M) ≥ 2, the quotient will not be meaningful.

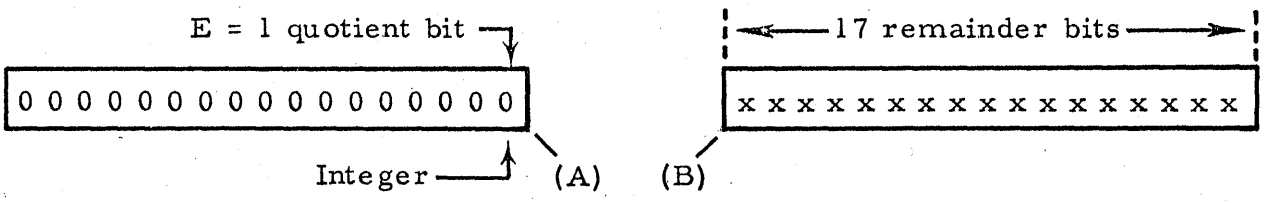
The number of quotient bits generated is always equal to the execution code "E". The execution code can be chosen to obtain a quotient with a specific scale factor, as discussed in Section VIII. The effect of different execution codes on the quotient is illustrated by the examples which follow. In the examples, "00000...." is used to designate leading zero bits, and "xxxxx...." is used to designate some combination of ones and zeros that represent significant bits. Sign bits are not shown.

Consider dividend (A) with 8 significant bits, and divisor (M) with 12 significant bits.



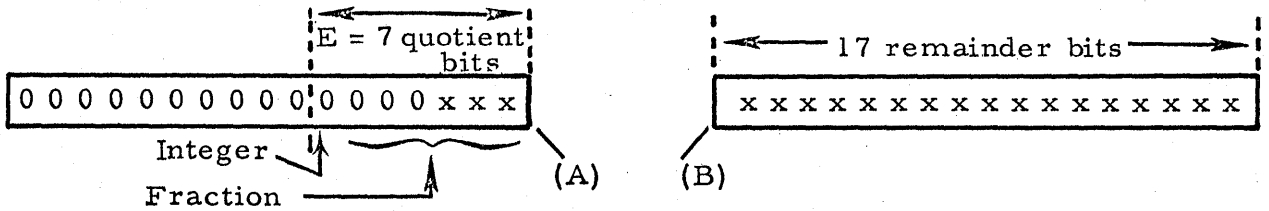
Using an execution code of 1 causes only one quotient bit to be generated. The first quotient bit generated is the integer bit. In this example, the divisor is much larger than the dividend, and the integer bit is zero.

For E = 1, the quotient and remainder are:



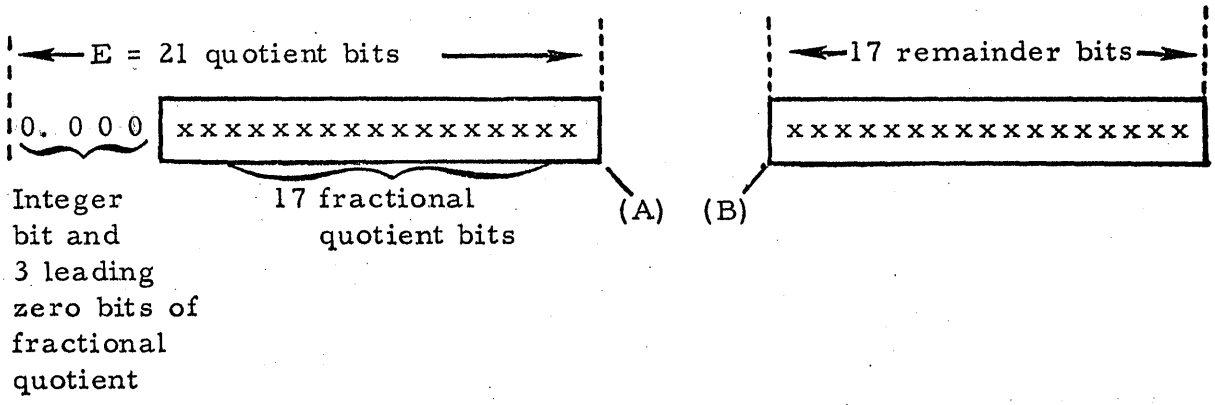
Using an execution code of 7 causes seven quotient bits to be generated: an integer quotient bit, and six fractional quotient bits.

For E = 7, the quotient is:



In the above example, the first four quotient bits are zeros because the divisor has four more significant bits than the dividend. In general, the minimum number of leading quotient bits that will be zero can be predicted by subtracting the number of dividend (A) bits from the number of divisor (M) bits. The execution code can then be chosen to eliminate M - A quotient zeros by adding M - A to 17 to form the execution code.

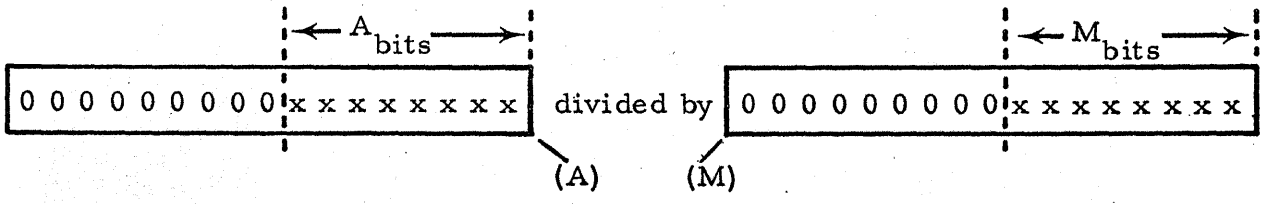
In the example: E = 17 + M - A = 21. For E = 21, the quotient is:



When an execution code greater than 17 is used, quotient bits are lost off the left end of the A register. In the example, E - 17, or 4 quotient bits were shifted off the left end of the A register. An execution code as high as 31 can be used, with the result that 31 - 17, or 14 quotient bits will be lost. For execution codes greater than 17, E - 17 quotient bits are always lost off the left end of the A register.

The overflow indicator is never turned on when zero or non-zero quotient bits are shifted off the left end of the A register during a divide instruction. The overflow indicator is only turned on when the quotient is one or greater. A quotient greater than one always turns on the overflow indicator, even if quotient bits are not shifted off the left end of the A register.

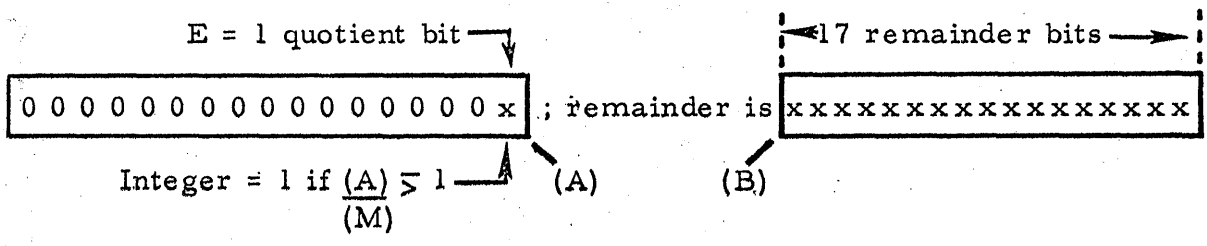
In the following example, the quotient will be less than two, but may be ≥ 1 because there are as many bits in the dividend (A) as in the divisor (M).



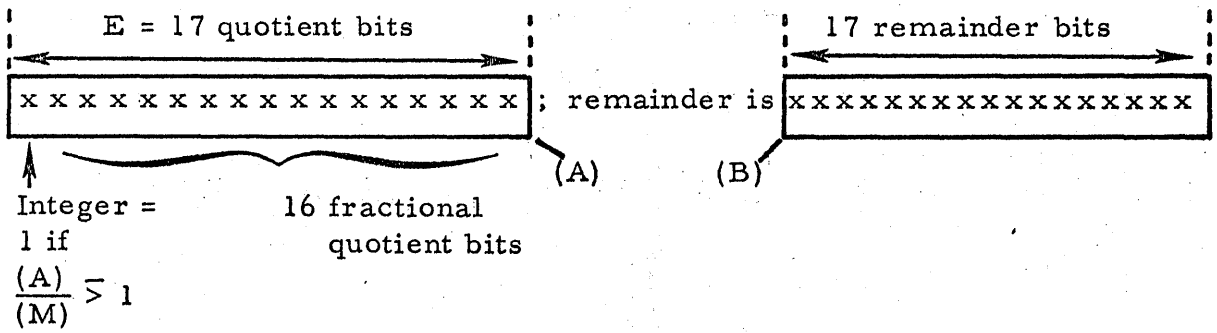
Examples of quotients obtained using execution codes 1, 17, and 18 are tabulated below.

If the quotient is equal to, or greater than one, the overflow indicator is turned on.

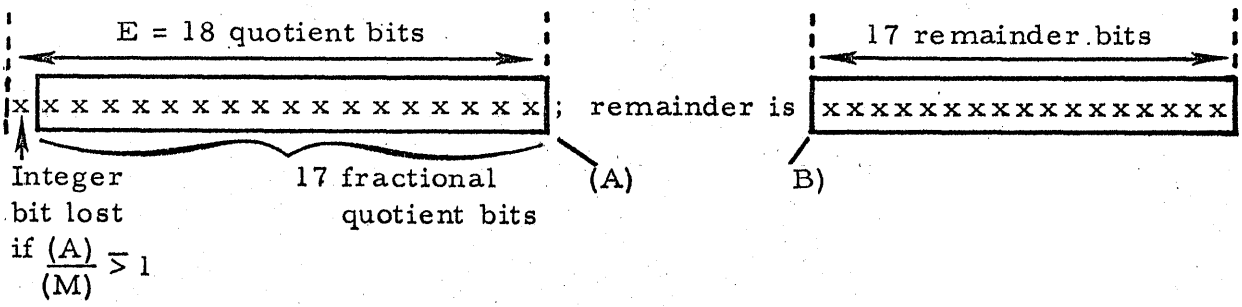
For E = 1, the quotient is:



For E = 17, the quotient is:



For E = 18, the quotient is:



If there are more bits in the dividend than the divisor, the quotient may be greater than two. If (A) ÷ (M) < 2, the quotient will always be meaningful; if (A) ÷ (M) ≥ 2, the quotient will be meaningless.

Special Cases:

If the dividend and the divisor are both zero, the "E" quotient bits will be all ones (with the appropriate sign), and the remainder will be all zeros.

If the dividend is non-zero and the divisor is zero, the quotient (with E = 18) will be all ones (with the appropriate sign) minus the dividend. The remainder will be all zeros.

A dividend and divisor having a quotient equal to or greater than two can be used to obtain a valid remainder in the B register if the execution code "E" is limited to 18 - A, where "A" is the number of significant bits in the dividend. Under these conditions, there will be "E" quotient bits in the A register; the bits will all be ones.

EFFECTS ON REGISTERS

The table in figure 2-1 shows how the different operation codes affect the contents of memory, the A register, and the B register. Because it is cumbersome to write 18 binary digits to show the contents of M, A, and B, the numbers are expressed in octal form. (Each octal digit represents 3 binary digits, as discussed in Section VIII.) Note that these octal numbers include the 18th or sign bit. Therefore, any octal number of 400000 or greater has a one (-) in the sign bit. For example:

S	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0

will be written as 360000 and

S	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
1	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0

will be written as 624000.

The first column in the table indicates the operation that is to be performed. The next four columns list the execution code (EX), the operand address (OPRND), the operation code (OP), and the next-instruction address (NI). Although these codes and addresses are shown in the sequence that they would appear in a program listing, the series of instructions does not represent a program.

The remaining columns of the table indicate the contents of memory, the A register, and the B register. The first line of each instruction represents the contents before the instruction is executed; the second line represents the contents after the instruction is completed. The following symbols are used:

a. XX means that the contents of the register or the memory location are destroyed in the process of executing the instruction.

b. SS means any sector number.

c. -- means that the register contents, operand address, and/or execution code is of no importance in the instruction.

The table lists only operations that affect memory, registers, or next-instruction addresses. Not included are: Stop, Digital, No Operation, and Tape.

Figure 2-1. Commands and Registers

Operation	EX	OPRND	OP	NI	(M)	(A)	(B)	NOTES
Load A	--	-----	29	-----	042500 042500	XX 042500	------	(M)→(A)
Load B	--	-----	07	-----	042500 042500	------ ------	XX 042500	(M)→(B)
Load A Neg	--	-----	21	-----	042500 042500	XX 442500	------	-(M)→(A)
Store A	--	-----	30	-----	XX 042500	042500 042500	------	(A)→(M)
Store B	--	-----	20	-----	XX 042500	------	042500 042500	(B)→(M)
Add	--	-----	25	-----	230000 230000	534000 074000	------	(M)+(A)→(A); note that original (A) is negative.
Add	--	-----	25	-----	230000 230000	270000 120000	------	(M)+(A)→(A);overflow occurs; turns on overflow indicator.
Subtract	--	-----	24	-----	270000 270000	230000 440000	------	(A)-(M)→(A); since (M)>(A), answer is negative.
Subtract	--	-----	24	-----	670000 670000	230000 120000	------	(A)-(M)→(A); since M is -, overflow occurs; turns on overflow indicator.
Shift →(A)	05	15-SS	01	-----	------	042500 001052	------	Track no. specifies right shift of A; EX specifies 5 places.
Shift (A)←	07	31-SS	01	-----	------	042500 120000	------	Track no. specifies left shift of A; EX specifies 7 places; turns on overflow indicator.
Shift (A),(B) ←	17	63-SS	01	-----	------	042500 006700	406700 400000	Track no. specifies left shift of A,B; EX specifies 17 places; turns on overflow. indicator.
Transfer on Negative	--	-----	09	45-75 45-75	-----	042500 042500	------	Sign of A is +; NI remains as programmed.
Transfer on Negative	--	46-73	09	45-75 46-73	------	442500 442500	------	Sign of A is -; OPRND becomes NI.
Transfer on Zero	--	-----	11	45-75	------	042500 042500	------	(A) not zero; NI stays as programmed.

Figure 2-1. Commands and Registers (continued)

Operation	EX	OPRND	OP	NI	(M)	(A)	(B)	NOTES
Transfer on Zero	--	46-73	11	45-75 46-73	------	400000 400000	------	(A) are (-) zero; OPRND becomes NI.
Transfer on Overflow	--	-----	10	45-75 45-75	------	------	------	Overflow indicator off; NI remains as programmed.
Transfer on Overflow	--	46-73	10	45-75 46-73	------	------	------	Overflow indicator on; OPRND becomes NI.
Compare Magnitude	29	-----	15	TT-75 TT-75	034200 034200	042500 006300	XX 042500	(A)→(B); since (M) < (A), NI is unchanged.
Compare Magnitude	29	-----	15	TT-103 TT-04	442500 442500	034200 406300	XX 034200	(A)→(B); since (M) > (A) sector of NI becomes 103 + 29 -128 or 4.
Extract	--	-----	05	-----	021415 021415	234277 020015	------	(M) ⊗ (A)→(A)
Merge	--	-----	31	-----	021415 021415	234277 235677	------	(M) ⊕ (A)→(A)
Switch	--	15-SS	02	-----	------	042500 042500	XX 042500	Oprnd. track nos. 00-15 specify (A) →(B).
Switch	--	31-SS	02	-----	------	XX 400277	400277 400277	Oprnd. track nos. 16-31 specify (B) →(A).
Switch	--	47-SS	02	-----	------	042500 400277	400277 042500	Oprnd. track nos. 32-47 specify (B) ⇄(A).
Switch	--	63-SS	02	-----	------	042500 000000	400277 000000	Oprnd. track nos. 48-63 specify 0→A, B.
Multiply	17	-----	16	-----	000012 000012	000002 000000	XX 000024	(M) X (A)→(A), (B) $10_{10} \times 2_{10} = 20_{10}$
Multiply	14	-----	16	-----	400012 400012	000002 400000	XX 400240	$-10_{10} \times 2_{10} = -20_{10}$ Shifted left 17 - E = 3 places.
Multiply	17	-----	16	-----	012000 012000	000200 000005	XX 000000	(M) X (A)→(A), (B) $10_{10} \times 2_{10} = 20_{10}$
Multiply	15	-----	16	-----	012000 012000	000200 000024	XX 000000	$10_{10} \times 2_{10} = 20_{10}$ Shifted left 17 - E = 2 places.

Figure 2-1. Commands and Registers (continued)

Operation	EX	OPRND	OP	NI	(M)	(A)	(B)	NOTES
Multiply	12	-----	16	-----	012000 012000	000200 000040	XX 000001	10_{10} X 2_{10} = ? Execution code less than number sig. bits in M -- invalid product.
Divide	18	-----	26	-----	300000 300000	230000 312525	XX 200000	(A) ÷ (M) → A; remainder → (B)
Divide	18	-----	26	-----	200000 200000	300000 200000	XX 000000	(A)÷(M)> 1 turns on overflow indicator; overflow occurs, but fractional quotient in A register is valid.
Divide	17	-----	26	-----	200000 200000	300000 300000	XX 000000	(A)÷(M)> 1 turns on overflow indicator; however integer and fractional quotient in A register is valid.
Divide	10	-----	26	-----	200000 200000	002500 000012	XX 100000	(A) ÷ (M) → A; remainder → (B); quotient shifted right 18 - E = 8 places.
Divide	26	-----	26	-----	200000 200000	002500 100000	XX 000000	(A) ÷ (M) → A; remainder → (B); quotient shifted left E - 18 = 7 places; significant bits are lost from A register; does not turn on overflow indicator.

SECTION III

BASIC PROGRAMMING

INTRODUCTION

This section tells how numbers and instructions are represented in the RW-300 computer. The operations defined in the preceding section are used to illustrate the instruction format, and simple program listings are presented.

On the magnetic drum which serves as the computer's internal memory, words are recorded on tracks as variations in magnetic flux. There are 64 tracks of interest to the programmer, and these tracks are numbered 00 through 63. Each track accommodates 128 words in sectors that are numbered 00 through 127. A particular location in memory is specified by the track number and sector number. For example, 17-07 specifies sector 07 of track 17.

RW-300 computers with the expanded memory have 123 tracks, or 15, 776 words, available to the programmer. Refer to Section IV for programming information for computers having this optional feature.

Data words (numbers) and instruction words are represented in the computer as binary numbers. A word is 18 binary digits in length. Although there are two space bits separating words on the drum, the space bits are of no concern to the programmer.

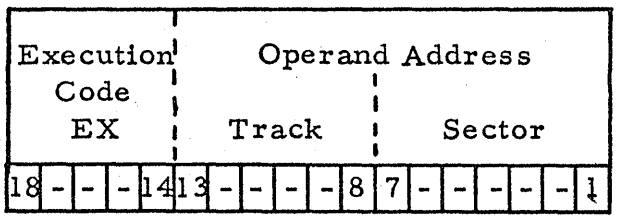
A word may represent numerical information, half of an instruction, or any pattern of 18 bits desired by the programmer (e.g., for program modification). Since the computer is a binary machine, the programmer

must have a working knowledge of the binary number system. (See Section VIII.)

INSTRUCTION WORDS

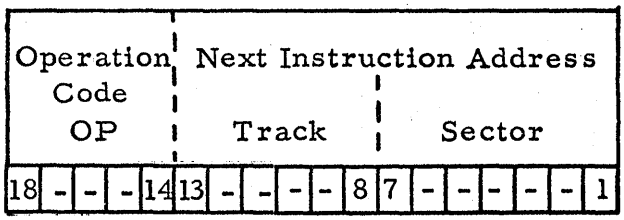
An instruction is two computer words, or 36 bits, in length. Two addresses are included in each instruction: the address of the operand and the address of the next instruction. This arrangement provides what is commonly called a "one-plus-one" instruction form, which permits optimum (i.e., minimum-time) programming. The operation to be performed, OP, is one of 21 different arithmetic or logical commands. The execution code, EX, determines the number of bits in a product or quotient, the number of places shifted, or other special functions as described in Section II.

WORD ONE



(Bit Positions)

WORD TWO



(Bit Positions)

FORMAT FOR LISTING INSTRUCTIONS

The series of instructions comprising a program are usually listed on a form similar to the one shown below.

CI	EX	OPERAND	OP	NI	REMARKS
A	I		I		
A	I		I		
A	I		I		

The columns of the listing have the following significance:

A is an "indicator" which, when read by the computer during loading, announces that the number which follows is an address.

CI is the location where the instruction is to be stored. CI stands for "current instruction".

I is an indicator which announces that the numbers which follow are to be stored in the location previously specified.

EX is the execution code contained in the first word of the instruction.

OPERAND is the address of the operand that will be used in the instruction. (The operand address may specify a number in memory that will be used in a computation, or may modify the instruction -- the exact significance of the operand address is explained in relation to each of the operation codes.)

I is an indicator which announces that the numbers which follow are to be stored in a sector whose number is one greater than that specified in the CI column. Thus, the entire instruction appears in two successive word (sector) locations on the drum, and the first word is located in the sector specified in the CI column.

OP is the operation code contained in the second word of the instruction.

NI is the address of the next instruction, i.e., the address of the instruction which will be read by the computer after the current instruction has been completed. (In some cases, the next instruction will be read from the address specified as the operand address, or will be formed by adding the execution code to the address specified in the NI column.)

The REMARKS column may be used by the programmer to make notes that can be referred to when the program is being checked out or modified.

The program is punched on paper tape using an off-line Flexowriter. The typing (punching) format must be compatible with the RW-300 load program contained in track 63 of memory. After the tape is prepared, it is threaded through the Flexowriter tape reader. The operator presses the LOAD button on the RW-300 control panel to initiate loading. The load program, tape-punching format, and related operating procedures are described in Section VI.

Sector 00 of track 00 is called the "origin" because the RW-300 reads the instruction in sectors 00 and 01 whenever the START button is pressed. If the following listing were punched on tape, loaded into the RW-300, and the START button pressed, the first word of the instruction (execution code and operand address) would be read from sector 00 of track 00, and the second word of the instruction (operation code and next-instruction address) would be read from sector 01 of track 00.

	CI	EX	OPERAND	OP	NI	REMARKS
A	00-00	I 00	52-17	I	29 17-06	Load A with 460_{10}

The instruction commands the computer to load the A register with the operand located in sector 17 of track 52, and proceed to the next instruction in sector 06 of track 17. As noted in the remarks column, this instruction loads the A register with the binary equivalent of the decimal number 460.

DATA WORDS, OR CONSTANTS

A data word, having numerical significance, is composed of 17 magnitude bits, plus a sign bit. The sign bit is zero for positive numbers and one for negative numbers.

Consider the decimal number +0.9375, which is equivalent to +0.1111 in binary. If +0.9375 were stored in the computer, it would appear in the following form:

	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit → Positions	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

If the number were negative, the binary digit in bit-position 18 would be a one.

The bit pattern shown above might also represent the decimal number 3.75 if the programmer chose to think of the binary point as being between bits 15 and 16 (3.75 in decimal = 11.11 in binary). Location of the binary point is a scaling consideration which is discussed in Section VIII.

FORMAT FOR LISTING CONSTANTS

Constants must be placed in computer memory during the loading operation. Constants are usually listed on a form similar to the one shown below.

	ADDR.	CONST.	REMARKS
A		C	
A		C	
A		C	

Format for Listing Constants -- Continued

The columns of the listing form have the following significance:

A is an indicator which, when read by the computer during loading, announces that the number which follows is an address.

ADDR. is the location where the number is to be stored.

CONST. is the number, or constant.

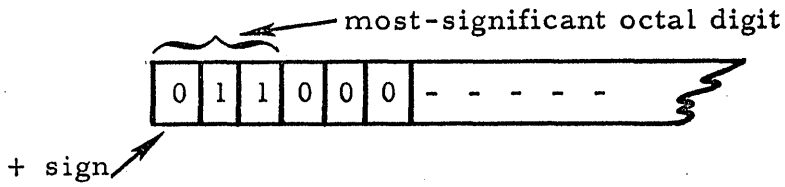
The REMARKS column may be used by the programmer to make notes that can be referred to when the program is being checked out or modified. Constants are punched on paper tape along with the instructions of the program, and the punching format must be compatible with the load program described in Section VI.

If the decimal constant 460 were to be listed for storage in memory, the number would first be converted to its six-digit octal equivalent, 000714. See Section VIII for information on decimal-to-octal conversion. For storage in sector 17 of track 52, the listing of the constant would take the following form:

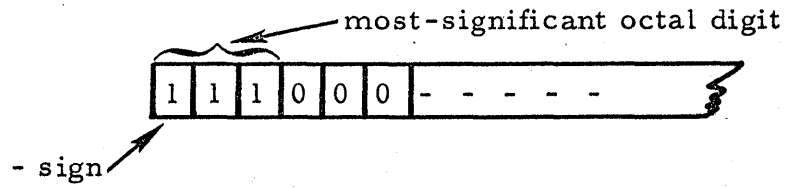
	ADDR.	CONST.	REMARKS
A	52-17	C000714	460_{10}

The six-digit octal number representing a constant includes 17 magnitude bits and one sign bit. In the listing of constants, the sign bit must be included in the most-significant octal digit. Therefore, the most-significant octal digit of a constant contains two magnitude bits and one sign bit.

The octal equivalent of +98304 is 300000, which would appear in computer memory as follows:



The octal equivalent of -98304 is -300000. Since the most-significant octal digit must include a one to form the negative sign bit, the number would have to be specified on the program listing as 700000. The number would appear in computer memory as follows:



USE OF MEMORY

General

In listing the instructions and constants, the programmer must choose memory locations that are accessible to the computer when the program is being loaded onto the drum and when the program is running. The choice of these memory locations is governed by the characteristics of memory, the characteristics of the load program, the configuration of the system, and by conventions that have been established on the basis of operating experience.

The paragraphs which follow describe the constraints imposed upon reading and storing information in memory, and describe how listings should be organized so that punched tapes can be loaded easily. Suggestions for keeping track of storage locations are also presented.

The choice of memory locations determines the time required for the computer to complete calculations. These timing considerations relate to "optimum programming" described in Section IV.

Reading Information from Memory

Any sector in tracks 00 through 63 can be read by the computer under program control.

Tracks 08 through 15 may or may not contain the digital equivalent of an analog input, depending upon the number of analog inputs accommodated by the system. Also, tracks 14 and 15 may contain data being transferred from the magnetic tape system.

Track 63 contains the load program which starts when the LOAD button is pressed.

Storing Information in Memory

By mating the "track group selection plug" with an appropriate "track group" jack, information can be stored in any sector of tracks 00 through 61. The plug and jack, located on the test and maintenance panel, are described in Section VII. Each track group (except track groups 56 through 61) includes eight tracks, as follows:

00 through 07

08 through 15

16 through 23

24 through 31

32 through 39

40 through 47

48 through 55

56 through 61

When the program is running, the track group selection plug on the RW-300 test and maintenance panel is usually connected to the jack marked 0-7, because these tracks can be written into under program control. This track group is sometimes referred to as "scratch pad" memory, because it is most frequently used to hold the intermediate results of calculations. If the program instructs the computer to store information in some track group other than 00 through 07, the computer halts and turns on the ERROR light. The program will resume if the track group selection plug is moved to the correct jack and the RESUME button is pressed. However, subsequent attempts of the program to store in some other track group will be foiled by another ERROR indication. Therefore, instructions and constants are usually loaded into track groups 08 through 61, and tracks 00 through 07 are reserved for intermediate results of calculations.

Specific sectors of track 07 are reserved for analog outputs, and only control information relating to specific output lines should be written into these sectors.

Specific tracks in track group 08-15 are reserved for analog inputs. Instructions or constants cannot be loaded into any sectors of these reserved tracks.

Except during program loading, information can always be stored in the 32-word recirculating register of track 62. The 32-word register is

described in conjunction with optimum programming (Section IV). Information can be stored in track 62 without reference to the track group selection plug. However, the information must be written under program control, and not during loading, because certain sectors of track 62 are written into by the load program, and any attempt to store in these sectors during loading will be fruitless because the load program will write other information there.

Information cannot be stored in track 63; any attempt to store in track 63 causes the ERROR light to glow and halts the program.

Organization of Listings

The lists of instructions and constants should be organized so that instructions to be stored in specific track groups appear together.

When a program of instructions and constants is being loaded, the computer stores instructions and constants in the tracks specified in the CI column of the instruction listing (and in the ADDR. column of the constant listing). If the listing calls for storage in different track groups, the computer signals ERROR, and the track group selection plug must be moved to the appropriate track group before the instruction or constant can be stored. Therefore, to speed up the loading process, the programmer should organize his listings so that all of the instructions to be loaded into a track group appear together on the paper tape punched from the listings.

Record Keeping

Only one word can be stored in any sector of memory. When a punched tape of a program listing is being loaded into the computer, new information will be stored in the locations specified, thus destroying anything stored

earlier. Therefore, if two separate instructions are assigned the same memory location, the instruction read last will be the instruction stored in the memory location.

To avoid assigning the same memory location to separate information, the programmer must keep a record of sectors in which instructions and constants are to be stored. The form shown below is convenient for record-keeping purposes.

Channel _____

0		32		64		96	
2		34		66		98	
4		36		68		100	
6		38		70		102	
8		40		72		104	
10		42		74		106	
12		44		76		108	
14		46		78		110	
16		48		80		112	
18		50		82		114	
20		52		84		116	
22		54		86		118	
24		56		88		120	
26		58		90		122	
28		60		92		124	
30		62		94		126	

Note that only even-numbered sectors are listed on the form. Since an instruction occupies two sectors, it is common practice when programming the RW-300 to specify an even-numbered sector location for instructions; in

this way, the first word of the instruction is stored in the even-numbered sector, and the second word is stored in the following odd-numbered sector.

There is no restriction to prevent the programmer from assigning an odd-numbered sector to the first word of an instruction--in which case the second word will be stored in the following even-numbered sector. If sector-location 127 is assigned to the first word of an instruction, the program listing must specify that the second word of that instruction is to be stored in sector 00 of the same track. In this special case, the complete instruction is listed as two "half-instructions":

	CI		EX	OPERAND		OP	NI	REMARKS
A	01-127	I	00	52-17	I			LOAD A (Ex and Operand)
A	01-00	I	00	17-06	I			(Op and NI)

SAMPLE PROGRAMS

The following examples show how to construct computer programs using the listing formats described in the preceding paragraphs, and using the commands described in Section II.

Example I

The following is a program to add two numbers, $\underline{x}$ and $\underline{y}$, and store the result in the memory location 01-02. Let $\underline{x} = +1273_8$ and $\underline{y} = -243_8$.

	CI		EX	OPERAND		OP	NI	REMARKS
A	00-00	I	00	01-00	I	29	00-02	x → A Register
A	00-02	I	00	01-01	I	25	00-04	x + y → A Register
A	00-04	I	00	01-02	I	30	00-06	Store result in 01-02
A	00-06	I	00	00-00	I	00	38-124	Stop

	ADDR.	CONST.	REMARKS
A	01-00	C001273	x
A	01-01	C400243	y

To prepare this program for loading into the computer, a paper tape is punched by typing the above listing on an off-line Flexowriter, using the typing format described in Section VI.

To load the program, the programmer first threads the punched tape through an on-line Flexowriter tape reader, and then presses the LOAD button on the computer's control panel. The tape is read under control of the load program in track 63 of the computer, and under control of the load program, the instructions and constants are stored in the locations specified in the CI and ADDR. columns of the program listing.

After the last symbol of the punched tape has been read, the computer will continue to read the blank trailing tape unless the last symbol punched on the tape is an "s". If an "s" is punched at the end of the tape, the computer will stop. If there is no "s" punched at the end of the tape, the loading operation may be halted by pressing the STOP button.

Pushing the START button causes the computer to begin the program by reading the first instruction contained in sectors 00 and 01 of track 00.

When the computer completes the sample program, it will halt. The results of the addition (001030_8) will be in the A register, as well as in memory location 01-02. The contents of the A register may be observed on the oscilloscope of the test and maintenance panel, as described in Section VII.

When the computer halts in response to a Stop instruction (operation code 00), it may be directed to proceed to the next-instruction address by pressing the RESUME button. In the example, pressing the RESUME button (after the computer halts) causes the computer to proceed to the instruction located in sector 124 of track 38.

The functions of the RW-300 control buttons are summarized in Section VII.

Example II

This program calculates the average of the squares of n small numbers: $\sum \frac{1}{n} x_i^2$, i = 1, ..., n. In this example, n = 4, and n is stored in memory location 01-50. The numbers x_1, ..., x_4 are stored in locations 01-00 through 01-03, respectively.

When the Stop instruction is executed, the average of the squares will be in the A register and in memory location 02-00. (See page 3-15.)

Note 1: Multiplication results in a double-length product. Since this example involves small numbers, the significant digits of the product are assumed to be in the B register, in which case the A register will contain zeros. This is not a general assumption, and it is the responsibility of the programmer to predetermine how significant product bits will be distributed between the A and B registers by adjusting the "scale factor" and execution code. These considerations are discussed in conjunction with number systems and scaling in Section VIII.

Note 2: The significant product bits in the B register are switched into the A register and then stored. The two instructions that accomplish

	CI			EX	OPERAND		OP	NI	REMARKS	
	A	00-00	I	00	01-00	I	29	00-02	Load A	$X_1 \rightarrow (A)$
Note 1	A	00-02	I	17	01-00	I	16	00-04	Multiply	$X_1 \rightarrow (A), (B)$
Note 2	A	00-04	I	00	32-00	I	02	00-06	Switch	$(A) \longleftrightarrow (B)$
	A	00-06	I	00	02-00	I	30	00-08	Store A	$X_1^2 \rightarrow$ Location 02-00
	A	00-08	I	00	01-01	I	29	00-10	Load A	$X_2 \rightarrow (A)$
	A	00-10	I	17	01-01	I	16	00-12	Multiply	$X_2^2 \rightarrow (A), (B)$
	A	00-12	I	00	32-00	I	02	00-14	Switch	$(A) \longleftrightarrow (B)$
	A	00-14	I	00	02-00	I	25	00-16	Add	$X_1^2 + X_2^2 \rightarrow (A)$
	A	00-16	I	00	02-00	I	30	00-18	Store A	$X_1^2 + X_2^2 \rightarrow$ location 02-00
	A	00-18	I	00	01-02	I	29	00-20	Load A	$X_3 \rightarrow (A)$
	A	00-20	I	17	01-02	I	16	00-22	Multiply	$X_3^2 \rightarrow (A), (B)$
	A	00-22	I	00	32-00	I	02	00-24	Switch	$(A) \longleftrightarrow (B)$
	A	00-24	I	00	02-00	I	25	00-26	Add	$X_1^2 + X_2^2 + X_3^2 \rightarrow (A)$
	A	00-26	I	00	02-00	I	30	00-28	Store A	$X_1^2+X_2^2+X_3^2 \rightarrow$ location 02-00
	A	00-28	I	00	01-03	I	29	00-30	Load A	$X_4 \rightarrow (A)$
	A	00-30	I	17	01-03	I	16	00-32	Multiply	$X_4^2 \rightarrow (A), (B)$
	A	00-32	I	00	32-00	I	02	00-34	Switch	$(A) \longleftrightarrow (B)$
	A	00-34	I	00	02-00	I	25	00-36	Add	$X_1^2+X_2^2+X_3^2+X_4^2 \rightarrow (A)$
Note 3	A	00-36	I	18	01-50	I	26	00-38	Divide	$(X_1^2+X_2^2+X_3^2+X_4^2) \div 4 \rightarrow (A)$
	A	00-38	I	00	02-00	I	30	00-40	Store A	Average $\rightarrow$ location 02-00
	A	00-40	I	00	-- --	I	00	-- --	Stop	

this (00-04 and 00-06) could be replaced by a Store B instruction. However, subsequent instructions use the switch-and-store sequence to form a running sum. In this example, all instructions are listed in the same pattern; in the next example, the pattern is made a part of a repetitive routine.

Note 3: Division by 4, to obtain the average of the sum of the squares, could be accomplished in less time by using a Shift-right instruction (shift right two places), which is equivalent to dividing by 4.

Example III

The following program solves the same problem as the program in Example II. However, it illustrates a "loop", a repetitive routine that reduces the number of instructions. The instructions for taking an x_i, squaring it, adding this to a running sum, and storing the result, are written out only once, as opposed to four times in the example above. However, the program returns to this sequence of steps as many times as necessary. A loop can handle virtually any number of x_i's without being lengthened in proportion.

Locations 01-00 through 01-03 again contain $x_1 \ldots x_4$. Locations 01-50 through 01-52 contain the constants 000004, 000003, and 000001, respectively. Location 01-54 is a "counter" which keeps track of the number of times the computer has gone through the loop. Location 02-00 contains the running sum and, at the end, the final answer.

The flow chart in figure 3-1 shows how the program is to operate.

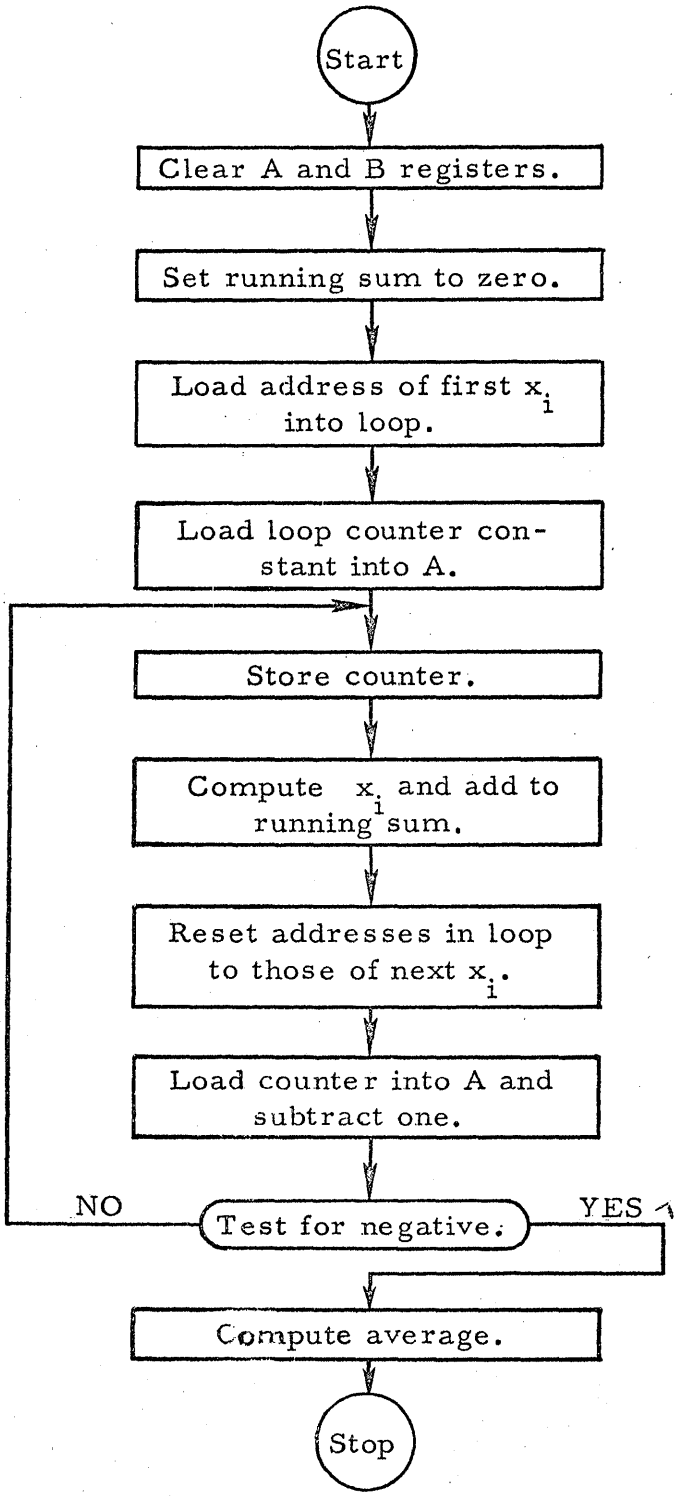


Figure 3-1. Flow Chart of Program in Example III

	CI			EX	OPERAND		OP	NI	REMARKS	
	A	00-00	I	00	50-00	I	02	00-02	Switch	0 → (A), (B)
Note 1	A	00-02	I	00	02-00	I	30	00-04	Store A	0 → 02-00
Note 2	A	00-04	I	00	01-53	I	29	00-06	Load A	17 01-00 → (A)
	A	00-06	I	00	00-14	I	30	00-08	Store A	17 01-00 → 00-14
	A	00-08	I	00	00-16	I	30	00-10	Store A	17 01-00 → 00-16
	A	00-10	I	00	01-51	I	29	00-12	Load A	000003 → (A)
Note 3	A	00-12	I	00	01-54	I	30	00-14	Store A	000003 → 01-54
Note 4	A	00-14	I	XX	XX-XX	I	29	00-16	Load A	X_i → (A)
	A	00-16	I	XX	XX-XX	I	16	00-18	Multiply	X_i^2 → (A), (B)
	A	00-18	I	00	32-00	I	02	00-20	Switch	(A) ↔ (B)
	A	00-20	I	00	02-00	I	25	00-22	Add	X_i^2 + (02-00) → (A)
	A	00-22	I	00	02-00	I	30	00-24	Store A	X_i^2 + (02-00) → (02-00)
Note 5	A	00-24	I	00	00-14	I	29	00-26	Load A	(00-14) → (A)
	A	00-26	I	00	01-52	I	24	00-28	Subtract	(00-14) + 1 → (A)
	A	00-28	I	00	00-14	I	30	00-30	Store A	(00-14) + 1 → (00-14)
	A	00-30	I	00	00-16	I	30	00-32	Store A	(00-14) + 1 → (00-16)
Note 6	A	00-32	I	00	01-54	I	29	00-34	Load A	counter → (A)
	A	00-34	I	24	01-52	I	15	00-12	Compare Magnitude	counter - 1 → (A) if -, go to 00-36
	A	00-36	I	00	02-00	I	29	00-38	Load A	Σx_i^2 → (A)
	A	00-38	I	18	01-50	I	26	00-40	Divide	$\Sigma x_i^2 \div 4$ → (A)
	A	00-40	I	00	02-00	I	30	00-42	Store A	$\Sigma x_i^2 \div 4$ → 02-00
	A	00-42	I	--	-----	I	00	-----	Stop	
	A	01-53	I	17	01-00				Half-instruction	00-14 and 00-16

	ADDR.	CONST.	REMARKS
A	01-50	C000004	n = 4
A	01-51	C000003	counter
A	01-52	C000001	decrement

Note 1: The Switch command clears the A and B registers (fills the registers with zeros), and the Store A command sets the running sum to zero. Clearing the A and B registers and storing the result is equivalent to loading the A register with zeros and storing the zeros. However, the clear-and-store operations save drum space and computation time.

Setting the running sum to zero is an "initializing" step which is necessary to prepare the computer for entering the loop. Other initializing steps include the storing of the first x_i address in the loop (covered by Note 2) and setting the loop counter to 3 (covered by Note 3).

The initializing steps are necessary if the program is to be repeated for different values of x. Without the initializing steps, the program could be run only once; the punched paper tape of the program would have to be reloaded before the calculation could be repeated.

Note 2: This and the next two steps set the operand addresses of the instructions in 00-14 and 00-16 to the address of the first x_i. The half-instruction "17 01-00" (execution code and the operand address) are stored in both 00-14 and 00-16. Actually, what is required is "00 01-00" in 00-14 and "17 01-00" in 00-16. However, instruction 00-14 is a Load A instruction, and the execution code is ignored. To load different half-instructions into locations 00-14 and 00-16 would require several additional memory sectors.

Note 3: This initializing step establishes the counter in sector 01-54. Each time through the loop, one will be subtracted from the counter, and a test will be performed. When the counter becomes negative, the program will have been through the loop four times and the squaring-and-summing calculations will be complete.

Note 4: XX XX-XX represents the execution code and operand address of x_1 . . x_4, which will be reset each time through the loop by adding one to the operand address contained in 00-14 and 00-16. See Note 5.

Note 5: In this and the next three steps, the operand address of instructions 00-14 and 00-16 are increased by one, to form the operand address of the next x_i. The Subtract command is used because the half-instruction "17 01-00" appears in memory as 420200_8, a negative number. By subtracting the constant 000001_8, the half-instruction becomes "17 01-01" the second time through the loop, "17 01-02" the third time through the loop, etc.

Note 6: The number in the counter is brought into the A register for comparison with the stored constant 000001. The flow chart indicates that one is subtracted from the counter, and if the result is negative, computation is complete; if the result is not negative, the new number is stored in the counter location. Although this operation could be performed by two operations (Subtract and Transfer on Negative), the subtraction-and-testing operation is performed in one operation by the Compare Magnitude instruction.

The Compare Magnitude instruction subtracts one from the counter, and the difference replaces the contents of the A register. The first three times through the loop, the contents of the A register after the Compare

Magnitude instruction are positive or zero, and the computer reads the next instruction from location 00-12. The instruction at location 00-12 stores the new counter in preparation for the next pass through the loop. After the loop has been repeated four times, the contents of the A register are negative after the Compare Magnitude instruction, and the computer reads the next instruction from the address formed by adding the execution code (24) to the next-instruction address (00-12). Thus, after passing through the loop the fourth time, the program proceeds to divide the sum of the squares by four (00-36).

THE RW-300 HAS HAD THE FOLLOWING TRACK ASSIGNMENTS MADE:

11 digital input tracks:

33, 36, 37, 39, 40, 41,

42, 43, 44, 45, 46.

9 digital output tracks:

0, 2, 4, 6, 7,

8, 9, 10, 11.

Flexowriter, high speed teletype punch, tracks:

0, 2.

Ferranti reader track:

32.

Magnetic tape units track:

39.

When reading the programming manual and the operations manual, the various commands apparently have different numbers. This apparent inconsistency results from showing the commands in the programming manual in the digital form and the operating manual in the octal and binary forms.

SECTION IV

OPTIMUM PROGRAMMING

INTRODUCTION

The programming examples in Section III employ "sequential program" listings; i.e., successive instructions are assigned consecutive locations in memory. A sequential program causes the computer to spend an excessive amount of time searching for instructions and operands. "Optimum programming" is a technique for selecting storage locations so that a minimum amount of computer time is lost in waiting.

To understand how optimum programming saves machine time, it is necessary to consider the physical characteristics of the computer memory. The magnetic drum used as memory is shown schematically in figure 4-1. Zeros and ones, representing instruction and data words, are recorded on the drum as variations in magnetic flux.

The program can be loaded into tracks 00 through 61, and each of these tracks has one head for reading information stored on the drum. The track address determines which of the 61 heads is used when information is taken from these "general storage" tracks.

Information is read from the drum serially (bit by bit) so that it requires "one word time" to read one computer word from memory. One word time is the time required for one word, or sector, to pass under a read head. Since there are 128 sectors in each track, and the drum makes a

complete revolution in one sixtieth of a second, one word time is approximately 130 microseconds.

In performing a series of instructions, the computer reads the first two-word instruction in two word times, but additional time is required to carry out the instruction. For example, to load the A register requires five word times, and if the Load A instruction is in sector 00 of some track, the computer will not be ready to read the next instruction until sector 05 is passing under the read heads. If, using sequential programming, the next instruction is located in sector 02, the drum must complete its revolution before sector 02 again passes under the read heads. Thus, sequential programming causes a delay of nearly one drum revolution (1/60 second) between the reading of each instruction. With optimum programming, more than 20 instructions can be accomplished in one drum revolution.

The above example describes the penalty paid for listing instructions in consecutive memory locations. The same type of time loss is incurred if the operand is not in an optimum location. In the case of instructions involving operands, about two drum revolutions could be required to complete a single instruction. But if operands and next instructions are assigned optimum locations, more than 40 instructions can be carried out in those two drum revolutions.

The paragraphs which follow describe the organization of the RW-300 internal memory and techniques for optimum programming.

MEMORY ORGANIZATION

For general storage needs, the 7,936 words of tracks 00 through 61 are available. Each of these tracks is equipped with a read head, thus permitting data to be read from any sector of general storage. These read heads are all aligned with respect to timing; i.e., at any given time the read heads are all at the same sector of their respective tracks. Therefore, the time at which information is read from a sector is independent of the track number.

Tracks 00 through 07

Tracks 00 through 07 provide program-writable memory for temporary storage of data. Each of these tracks has a single head which functions as both a read and a write head. Track 07 is fitted with an additional head which reads data for conversion to analog output. If more than 64 analog outputs are required, track 07 is fitted with two additional heads. However, the program read/write head of this track performs the same function as other program read/write heads.

Tracks 08 through 15

Tracks 08 through 15 have read/write heads similar to those on tracks 00 through 07. These tracks may also be fitted with an extra analog write head which is not under computer program control. However, the computer can be modified to make all or part of these tracks writable under program control.

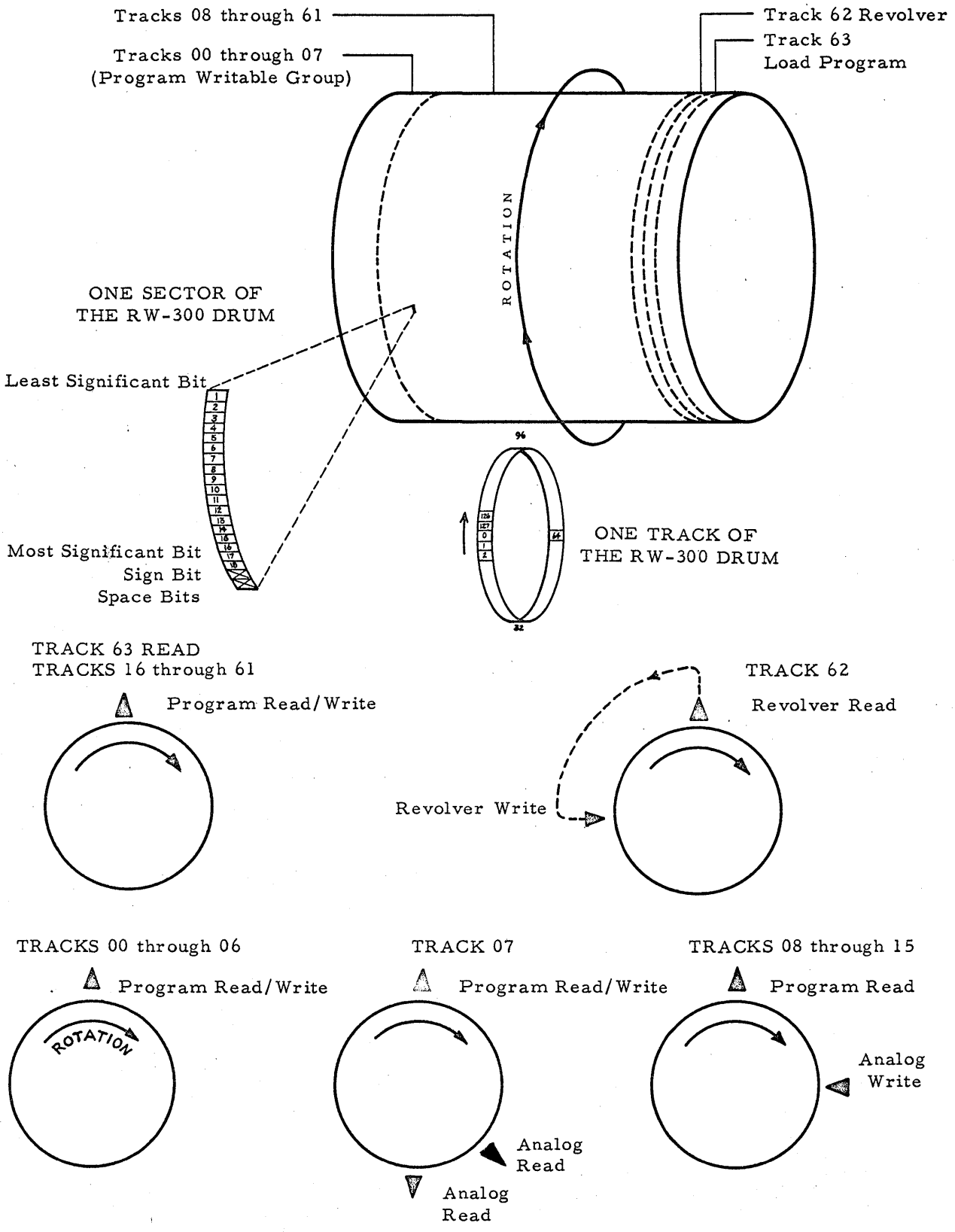


Figure 4-1. RW-300 Memory

When analog input facilities are employed, the digital equivalents of voltages from instruments are written into one or more of these tracks, starting with track 08. The number of tracks used to accommodate the analog input data is determined by the number and type of analog inputs.

A maximum of 128 analog inputs can be accommodated by each track, so that with 128 analog inputs, track 08 would not be available for general storage; with 256 analog inputs, tracks 08 and 09 would be unavailable for general storage, and so on. In some cases, to minimize the amount of equipment required, fewer than 128 inputs are accommodated on each track, but the analog inputs never occupy tracks other than tracks 08 through 15.

The programmer must not attempt to store part of the program in tracks 08 through 15, which are reserved for analog inputs. Although the program can be written into the reserved tracks during loading, the write head that records analog input information will write converted analog input information (or zeros) into the sectors, destroying the information written during the loading operation. The tracks reserved for analog inputs are specified during the planning stage of an installation, and any tracks not reserved for analog inputs can be used for general program storage.

Tracks 08 through 61

For loading information into the computer, it is necessary to be able to write on tracks 08 through 61. Also, under special circumstances, it may be desirable to use some of these tracks for writable space during program operation. This can be accomplished by manually connecting the track selector

plug on the test and maintenance panel to the jack representing the appropriate track group (00-07, 08-15, 16-23, 24-31, 32-39, 40-47, 48-55, 56-61).

Any attempt to write on a track which has not been selected as writable will cause the record ERROR light to turn on and the program to stop.

Track 62

Track 62 is the circulating register, or "revolver", which provides 32 words of fast-access storage. The revolver has a read head which is aligned with the read heads of the other tracks. Track 62 also has a write head and can always be written on by the program. The revolver write head precedes the read head by 32 sectors; i.e., a word on the revolver passes under the read head 32 word times after passing under the write head. (See figure 4-1.)

The revolver read and write heads also differ from the read and write heads of general writable storage in that they are continually reading and writing. As each sector of the revolver passes under the read head, its contents are read and immediately written 32 sectors later. For example, if a word is written into 62-04, then 32 word times later this sector will pass under the read head and its contents will be read and written into 62-36; in another 32 word times 62-36 will pass under the read head and its contents will be written into 62-68, etc.

Because each word is written in four sectors around the track (at 32-word intervals) within one drum revolution, the average access time for information on the revolver is one-fourth that of general memory. The

circulation of data on the revolver is interrupted only by writing new data onto this track.

Track 62 is a convenient track to use for storing instructions, half-instructions, or information used repeatedly in any part of the program. The use of track 62 is described near the end of this section.

Track 63

Track 63 has a read head which is aligned with the read heads of other tracks, but has no write head. This track is permanently reserved for the load program described in Section VI. Any attempt to write on track 63 will cause the ERROR light to turn on and will halt the program.

EXPANDED MEMORY

RW-300 computers with the expanded memory drum have 123 tracks of interest to the programmer. For identification purposes these tracks are divided into two sets: track set A, with tracks numbered from 00A through 63A, and track set B, with tracks numbered from 00B to 63B.

Five tracks of each set are common to each other. These common tracks are 00, 05, 06, 07, and the revolver, 62. Information written into track 05A, for example, is also written into track 05B. These five common tracks are regarded as single tracks.

Track set A is identical to the tracks found on the 8,000 word drum, i.e., track 63A contains the load program, and tracks 08A through 15A are reserved for analog input data.

Tracks 08B through 15B are available to the programmer without regard to the corresponding tracks in track set A which might be used for analog input data.

Track 63B is not a writable track. However, it may contain a service routine which is available to the programmer.

Switching computer read control from one track set to the other is accomplished by a digital output instruction using a track address of 03 in the operand address. An odd number in the execution code will transfer control to the B set of tracks. An even number in the execution code will transfer control to the A set of tracks.

Depressing the LOAD or START button always causes the computer to read from the A set of tracks.

A three-position track set transfer switch, designated "A-B-REMOTE" and located on the test and maintenance panel, provides track set A or track set B writing options. When the switch is placed in position A, a store instruction will cause the information to be written into the corresponding track and sector of track set A. When the switch is placed in position B, a store instruction will cause the information to be written into the corresponding track and sector of track set B. Common tracks are not affected by the drum transfer switch. The A and B positions of the switch would normally be used only when storing programs into the computer during load operations.

No Record Error provisions exist if the operator should inadvertently leave the track set transfer switch in the wrong position during loading

operations. Under these circumstances the data will be entered into the corresponding tracks of the incorrect track set.

The REMOTE position of the track set transfer switch allows either track set A or track set B to be selected for writing under program control. A one-bit digital output instruction will transfer writing control from one track set to the other. One-bit outputs are described in Section V. The track address and the corresponding digit in the A register to affect the track set relay are specified for each RW-300 computer individually and may vary from one machine to the other. A delay of 16.6 ms, or approximately one drum revolution, should be allowed between the one-bit digital output instruction and the next store instruction.

SELECTING OPTIMUM MEMORY LOCATIONS

To prepare a program that can be performed in a minimum amount of time, the programmer must select memory locations so that the next instruction to be executed is passing under the appropriate read head immediately after the previous instruction has been executed. Similarly, when an operand is being read from memory, the operand should begin passing under the appropriate read head the moment the computer's arithmetic circuits are ready to receive that operand.

A reference table of R-W 300 instructions, the last page of this programming manual, contains a column labeled "Execution Time". This

column tells how long it takes the computer to get ready for the operand and how long before the computer is ready to read the next instruction. The time is specified in word times, which is directly equivalent to numbers of sectors.

Note the Add command in the table of RW-300 instructions. From the table, "CI⟶Oprnd Add." is 3 word times; "Total CI⟶NI" is either 6 or 7 word times. This means that the computer requires 3 word times to prepare for receiving the operand, and either 6 or 7 word times to complete the addition and be ready for the next instruction. Thus, if the first word of an Add instruction is assigned the memory location 00-00 (CI column in the program listing), the operand should be stored in 00-00 + 3, or 00-03. The first word of the next instruction should be assigned the location 00-00 + 6 or 00-00 + 7.

In the case of the Add and Subtract instructions, the next-instruction address location depends upon the signs and magnitudes of the two numbers to be added or subtracted. If the sign of the A register is unchanged by the operation, only 6 word times are required. If the sign of A changes, 7 word times are required.

When determining optimum sector numbers, the number of word times and current-instruction address are added "modulo 128." This means that if the sum exceeds 128, then 128 is subtracted from the sum to obtain the optimum sector number: 114 + 17 - 128 = 03.

In the case of the three transfer commands, the computer will be ready to read the next instruction in either 4 or 5 word times (table of RW-300 instructions), depending upon whether or not transfer conditions

are met. If transfer conditions are not satisfied (not negative, no overflow, not zero), the first word of the next instruction should be assigned a location that is 4 word times greater than the current-instruction address. If transfer conditions are satisfied, the first word of the next instruction should be assigned a location that is 5 word times greater than the current-instruction address. Note that when transfer conditions are satisfied, the operand address becomes the next-instruction address (Section II).

In the case of a transfer command, it would not be possible to assign absolutely optimum addresses to both operand and next-instruction addresses within the same track. Absolutely optimum addresses would have consecutive sector numbers, and this is not possible, because each address represents a two-word instruction. However, optimum addresses can be assigned by locating the two branches of the transfer instruction in different tracks. No computer time is lost in switching from track to track. A Transfer on Negative instruction with optimum operand and next-instruction addresses might be:

	CI		EX	OPERAND		OP	NI
A	43-00	I	00	14-05	I	09	37-04

In the case of the Compare Magnitude command, the computer is ready for the next instruction in 5 word times if $|(M)| \leq |(A)|$, but if $|(M)| > |(A)|$, 7 word times must elapse before the computer is ready to read the first word of the next instruction. Thus, an execution code of 02 should be used to form an absolutely optimum next-instruction address. An

execution code of 01 or 00 would result in the loss of one drum revolution, or 1/60 second of computing time.

The table of RW-300 instructions indicates that optimum next-instruction addresses for some operations depend upon the execution code ("nn" in the table). For these operations, the execution code is added (along with the required number of word times) to the sector number of the current-instruction address to form the optimum address of the next instruction. A Multiply instruction with an optimum operand and next-instruction address might be:

CI		EX	OPERAND		OP	NI
A	43-00	I	10	43-03	I	16	43-16

If the number of words in the program approaches the number of words available in general storage, it will not be possible for the programmer to assign absolutely optimum memory locations to every instruction, because the desired memory location may have been previously assigned. In this event, more than the specified number of word times should be added to the current-instruction address.

Because it is impossible to optimize completely a program which uses all sectors in a given area of memory, priority should be given to program segments that are repeated many times. Frequently repeated program loops should be more highly optimized than program segments that do not contain loops. Therefore, frequently used loops should be programmed first so that optimum storage locations can be chosen from a relatively empty storage area. As the programming work progresses, the storage area will begin to fill, and absolutely optimum memory locations will not be available.

Less computing time is lost if the less-than-optimum locations are assigned to noniterative operations.

The general requirements for keeping a record of assigned memory locations are discussed in conjunction with memory usage in Section III. The procedure of assigning only even-numbered addresses to instructions will aid in the record-keeping task. When only even-numbered addresses are used for the first word of instructions, the optimum sector number is determined by adding more than the specified number of word times to the current-instruction address. For example, the Switch instruction requires 5 word times; with the first word of the Switch instruction in 00-00, the address of the first word of the next instruction should be 00-06.

The preceding paragraphs describe the general procedures and philosophy for selecting optimum memory locations. In the paragraphs which follow, similar commands are grouped together, and specific requirements for each command are presented.

Load, Merge, and Extract

The load instructions (LA, LN, and LB) and the Merge and Extract instructions all have identical timing requirements; only the LA instruction will be discussed.

Example 1: Load (M) into A.

	CI		EX	OPERAND		OP	NI
A	38-00	I	00	05-03	I	29	38-05

Since the computer takes three word times to go from the current-instruction address to the first accessible storage cell, the optimum sector location of M is the sector number of the current-instruction address plus three. The computer requires two word times to read the operand and load it into the A register; therefore, the optimum sector location for the next-instruction address is the sector number of the operand plus two.

Add and Subtract

The Add and Subtract instructions have identical timing requirements; only the Add instruction will be discussed.

Example 2: Add (M) to (A).

	CI		EX	OPERAND		OP	NI
A	38-05	I	00	05-08	I	25	38-12

The optimum sector location for the operand is determined by adding three to the current-instruction address sector number. The addition operation requires three word times if the sign of A does not change as a result of the operation, four word times if the sign of A changes. Therefore, unless the programmer knows that the sign of A will not change, he should allow four words between the operand address and the next-instruction address. The penalty for not doing this could be the loss of a complete drum revolution.

Multiply and Divide

The Multiply and Divide instructions have identical timing requirements; only the Multiply instruction will be discussed.

Example 3: Multiply (A) by (M).

	CI		EX	OPERAND		OP	NI
A	38-12	I	17	05-15	I	16	38-35

The optimum sector location for the operand is determined by adding three to the current-instruction sector number. The optimum sector location of the next-instruction address is determined by adding "E" (execution code) plus three to the sector number of the operand address.

Compare Magnitude

Example 4: Compare $|(M)|$ with $|(A)|$.

	CI		EX	OPERAND		OP	NI
A	38-35	I	02	05-38	I	15	38-40

The optimum sector location for the operand is determined by adding three to the current-instruction sector number. The optimum next-instruction address is determined by adding two to the operand address. However, if the results of the comparison are negative, $(M) > (A)$, the actual next-instruction address will be the listed next-instruction address plus the execution code "E". Since seven word times are required before the computer is ready to read from the modified next-instruction address, the minimum execution code should be two.

Transfer

The transfer instructions (TN, TZ, and TF) all have identical timing requirements; only the TF instruction will be discussed.

Example 5: Take the operand address as the next-instruction address if the overflow indicator is on; if the overflow indicator is off, read the next instruction from the address specified in the NI column of the listing.

CI			EX	OPERAND		OP	NI
A	38-40	I	00	05-45	I	10	38-44

Since the computer takes five word times to test the overflow indicator and substitute the operand address for the next-instruction address, the optimum sector location for the first word of the next instruction is the sector number of the current-instruction address plus five. The computer requires only four word times to be ready for the next-instruction address if the overflow indicator is off; therefore, the optimum sector location of the next-instruction address is the sector number of the current-instruction address plus four.

Switch

Example 6: Switch the contents of the A register into the B register.

CI			EX	OPERAND		OP	NI
A	38-44	I	00	15-00	I	02	38-49

The Switch command uses the track number of the operand address to specify the type of switch, and the sector number has no significance. Therefore, any sector number can be assigned as the sector number of the operand address. Since the computer requires five word times to complete the switch instruction, the optimum sector location for the next-instruction address is the sector number of the current-instruction address plus five.

Shift

Example 7: Shift the contents of the B and A registers left 3 places.

	CI		EX	OPERAND		OP	NI
A	38-49	I	03	48-00	I	01	38-56

The Shift command uses the track number of the operand address to specify the type of shift, and the sector number has no significance. Therefore, any sector number can be assigned as the sector number of the operand address. Since the execution code "E" specifies the number of places shifted, and since one word time is required for each place shifted, the optimum sector location for the next-instruction address is determined by adding "E" plus four to the sector number of the current-instruction address.

No Operation

Example 8: Transfer unconditionally to the next-instruction address.

	CI		EX	OPERAND		OP	NI
A	38-56	I	00	42-116	I	03	38-60

Since the operand address has no significance in the No Operation command, any track and sector number can be assigned. The optimum sector location of the next-instruction address is determined by adding 4 to the sector number of the current-instruction address.

Store

The store instructions (SA and SB) have identical timing requirements; only the SA instruction will be discussed:

Example 9: Store (A) in (M).

	CI		EX	OPERAND		OP	NI
A	38-60	I	00	05-32	I	30	38-66

The optimum operand address (storage sector M) for store instructions is determined by adding four to the sector number of the current-instruction address. The optimum location of the next instruction is determined by adding six to the sector number of the current-instruction address.

Digital

The timing requirements of digital commands depend upon whether the command is a digital input or a digital output, and upon the type of input or output device addressed by the command. Digital inputs and outputs associated with the Flexowriter (track address 00 or 32) require 110 milliseconds. All other digital outputs require at least 10 milliseconds. Digital inputs from switches require 6 word times.

Example 10: Input to A register from toggle switches.

	CI		EX	OPERAND		OP	NI
A	38-66	I	18	36-00	I	06	38-72

The Digital command uses the track number of the operand address to specify a particular device, and the sector number has no significance. Therefore, any sector number can be assigned as the sector number of the operand address. For clearing the A register, for digital inputs from the toggle switches, and for some other digital input instructions, the optimum sector location for the next-instruction address is the sector number of the current-instruction address plus six. Section V describes timing considerations for digital output commands.

COMPARISON OF OPTIMUM AND SEQUENTIAL PROGRAMMING

The next-instruction address in each of the preceding examples specifies the current-instruction address of the next example. If the ten examples were assembled to form a program, the commands would be: Load A, Add, Multiply, Compare Magnitude, Transfer on Overflow, Switch, Shift, No Operation, Store A, and Digital Input. This optimum program would be completed in approximately one-half of a drum revolution (1/120 second). However, if the instructions and constants used in the program were assigned sequential locations in memory, approximately 15 drum revolutions (1/4 second) would be required to complete the same program.

Although the optimum program formed by the ten examples could be completed by the computer in about 1/30 the time required for a sequential program, no general statement can be made concerning the amount of computer time conserved by optimum programming. It is difficult to program a problem so that all memory locations are absolutely optimum because instructions and constants in optimum locations for one segment of a program may interfere with the selection of optimum locations for other segments of the same program.

REVOLVER

Track 62, the "revolver", provides fast-access storage for 32 computer words. Any word written into the revolver passes under the revolver read head every 32 word times, thus making that word accessible four times during each drum revolution.

A number used repeatedly in a calculation can be stored in the revolver so that it will be readily available. Also, frequently repeated instructions can be placed in the revolver. When instructions are placed in the revolver, the two words that make up the instruction are stored in sequential revolver locations so that the two half-instructions will be read as a single instruction.

Each sector of the revolver has a separate address. Thus, if a number is stored in 62-00, it can be read next from sector 62-32, then 62-64, etc. It is convenient to think of revolver locations as R0, R1, ... R30, R31 and use the following table.

Table of Equivalent Revolver Locations

R0	0	32	64	96	R16	16	48	80	112
R1	1	33	65	97	R17	17	49	81	113
R2	2	34	66	98	R18	18	50	82	114
R3	3	35	67	99	R19	19	51	83	115
R4	4	36	68	100	R20	20	52	84	116
R5	5	37	69	101	R21	21	53	85	117
R6	6	38	70	102	R22	22	54	86	118
R7	7	39	71	103	R23	23	55	87	119
R8	8	40	72	104	R24	24	56	88	120
R9	9	41	73	105	R25	25	57	89	121
R10	10	42	74	106	R26	26	58	90	122
R11	11	43	75	107	R27	27	59	91	123
R12	12	44	76	108	R28	28	60	92	124
R13	13	45	77	109	R29	29	61	93	125
R14	14	46	78	110	R30	30	62	94	126
R15	15	47	79	111	R31	31	63	95	127

If the optimum sector number for a Store A instruction were 53, the programmer would use 62-53 as the operand address, and would note in the REMARKS column of the program listing "(A)→R21". When the stored number is to be read from the revolver, the sector number of the operand address can be any sector number on the "R21" line in the table of revolver locations.

EXPANDED MEMORY

The rules for optimum memory locations apply to those RW 300 computers having the optional expanded memory. However, special considerations dictate that optimum programming is also a function of the over-all program arrangement. Frequent and unnecessary switching back and forth from one track set to the other can be time-consuming and wasteful of program space.

Once the basic requirements of a specific process or problem have been established, the programmer should plan track and track set locations in such a manner as to keep track set switching at a minimum.

OPTIMUM PROGRAMMING USING SYMBOLS

The RW 300 Program Library includes an assembly routine which accepts programs coded in symbolic form. The routine is called "OPUS," Optimum Programming Using Symbols. Copies of the OPUS instruction manual are available from TRW Computers Company.

OPUS reduces the work of the programmer by requiring that only mnemonic operation codes and some other information be listed. In most cases, next instruction addresses need not be specified; OPUS assigns an optimum next instruction address. When it is necessary to refer to an address several times within the program, the programmer assigns letter or letter-number combinations as symbolic addresses. These symbolic addresses, chosen arbitrarily by the programmer, serve as mnemonic

devices for the programmer -- OPUS automatically assigns an optimum address to each symbolically identified operand or next-instruction address. Mnemonic operation codes are used: LA for Load A, A for Add, SW for Switch, etc.

The OPUS package includes an instruction manual and a punched paper tape containing the program. After the OPUS Routine has been loaded into the RW-300, the computer reads the tape containing the symbolically coded program, optimizes the program, and punches a new tape which is a line-for-line, machine-language translation of the symbolically coded program.

In addition to optimizing the symbolically coded program, the OPUS Routine provides the programmer with two printouts: an "Availability Table Map" of storage sectors occupied by the optimized program, and a "Symbol Table Map" of memory sectors assigned by OPUS to each of the symbolic addresses.

Although OPUS simplifies program listing, optimizing, and record keeping, the programmer must have a complete knowledge of machine-language instructions in order to correct or modify any program he prepares in symbolic form.

SECTION V

DIGITAL INPUT AND OUTPUT

INTRODUCTION

The extremely flexible digital input and output capabilities of the RW-300 are described briefly in Section I. In the paragraphs which follow, the input and output facilities of the RW-300 are described in detail, and sample program listings are presented.

The basic RW-300 is provided with a Flexowriter, which consists of three devices combined: an electric typewriter, a paper-tape punch, and a paper-tape reader. The Flexowriter is used to prepare punched tapes of programs, and the tape reader on the Flexowriter is used to load the program into computer memory. The Flexowriter is also used to obtain information printouts or punched tape under program control. Flexowriter codes and characteristics are described in detail at the end of this section.

Six switches on the test and maintenance panel of the basic computer can be used for "break-point" control of an operating program, or for inserting information under program control.

Ferranti paper-tape readers (60 characters per second) and Teletype paper-tape punches (60 characters per second) are optional equipment used with the RW-300 to obtain higher input-output speeds. Other paper-tape equipment and punched-card readers and card punches can also be used with the RW-300.

The flexibility and expandability of the RW-300 digital input and output system permit a wide variety of digital input and output devices to be specified. In addition to the high-speed readers and punches, special input switches and indicators are available to aid the communication between the operator and the computer. This section of the manual contains a description of some of the input-output equipment, and includes information needed by the programmer who will be using the equipment.

DIGITAL COMMAND

The operation code for communication with digital input-output equipment is 06. The operand track address specifies whether the Digital command is an input or an output:

Operand Track Number	Type of Digital Command
00 through 31	Output from computer's A register
32 through 63	Input to the computer's A register

In the case of a Digital output instruction, the execution code is only significant for the "one-bit" outputs described in conjuction with expanded input and output capabilities.

In the case of a Digital input instruction, the execution code determines where the input bits will be in the A register. The number of lines read into the A register is equal to the execution code. Execution codes greater than 18 should not be used with the Digital instruction. The significance of the execution code for Digital input instructions is illustrated in examples which follow.

BASIC INPUT-OUTPUT CAPABILITIES

Inputs

Digital inputs permit the computer to accept on-off signals from two groups of input lines. One of these groups (line L_{12}-L_{18}) is reserved for the Flexowriter or other seven-bit input. The other group of 18 inputs (lines L_{21} - L_{38}) is used to accept inputs from other external devices, or from toggle switches on the test and maintenance panel.

If lines L21 through L38 are all set to one (connected to -5 volts), and if the DIGITAL INPUT selector switch on the test and maintenance panel is turned to EXTERNAL, the following instruction will cause 18 ones to be read into the A register.

CI		EX	OPERAND	OP	NI	REMARKS
A	TT-SS	I 18	40-SS	I 06	TT-SS	

The track and sector numbers of the current instruction (CI column) and the next instruction (NI column) depend upon the relationship of the Digital instruction to the rest of the program. The track number of the operand address is shown as 40, but any track address from 36 to 63 is satisfactory. The sector number of the operand address does not affect a Digital instruction.

The execution code of 18 causes the 18 ones read into the A register to appear in bit-positions 1 through 18 of the A register. An execution code of 17 causes the input from L38 to appear in bit-position 17, and all succeeding bits to be shifted one place to the right so that the input from L21 is lost.

The effect of different execution codes is shown in the following table.

Execution Code	Bit Position in A Register																			
	18	17	16	.	.	.	.	.	.	.	.	.	4	3	2	1
18	L38	L37	L36	.	.	.	.	.	.	.	.	.	L24	L23	L22	L21
17	0	L38	L37	.	.	.	.	.	.	.	.	.	L25	L24	L23	L22
16	0	0	L38	.	.	.	.	.	.	.	.	.	L26	L25	L24	L23
--	--	--	--					--					--	--	--	--
4	0	0	0	.	.	.	.	.	.	.	.	.	L38	L37	L36	L35
3	0	0	0	.	.	.	.	.	.	.	.	.	0	L38	L37	L36
2	0	0	0	.	.	.	.	.	.	.	.	.	0	0	L38	L37
1	0	0	0	.	.	.	.	.	.	.	.	.	0	0	0	L38
0	0	0	0	.	.	.	.	.	.	.	.	.	0	0	0	0

A useful application of the Digital input instruction is to clear the A register. If the execution code specified is zero, no lines are read, and the A register will contain all zeros.

Inputs from Toggle Switches

The six toggle switches on the test and maintenance panel can be sampled on input lines L21 through L26 by turning the DIGITAL INPUT selector switch on the panel to INTERNAL and using a Digital input instruction similar to the one shown above. Using an execution code of 18 will fill the least-significant bit positions of the A register with the bit pattern represented by the settings of the DIGITAL INPUT toggle switches. In addition to the bit pattern obtained from the toggle switches, the A register will contain any inputs that are applied to Lines L27 through L38.

The Extract operation (operation code 05) can be used to remove unwanted bits obtained during a Digital input instruction. In the following example, it is desired to preserve only the six bits read into the A register from toggle switches L21 through L26. This is accomplished by an Extract

instruction using the octal constant 000077. Because bit-positions 1 through 6 of the constant contain ones, the information read into those A-register bit positions will be preserved. However, bit-positions 7 through 18 of the constant contain zeros, and the Extract operation will cause zeros to be placed in bit-positions 7 through 18 of the A register.

CI		EX	OPERAND		OP	NI	REMARKS
A	TT-SS	I	18	40-SS	I	06	17-08	DG in from L21 through L38
A	17-08	I	00	42-30	I	05	TT-SS	Extract L21 through L26

At the conclusion of the second instruction, the A register will contain one bits only for DIGITAL INPUT toggle switches set to ONE; all other bits in the A register will be zeros.

Inputs from Flexowriter

To transmit information from the paper tape in the Flexowriter reader to the A register, a Digital instruction is used with an operand track address of 32. An execution code of 07 is normally used so that the Flexowriter signals applied to lines L12 through L18 appear in bit-positions 1 through 7 of the A register. A Digital input instruction with an operand track address of 32 causes the Flexowriter to read one frame of paper tape and then advance the tape one frame in preparation for a subsequent Digital input instruction. Flexowriter characteristics, Flexowriter codes, and the format of the punched paper tape are described at the end of this section of the manual.

Output to Flexowriter

To print or punch information on the Flexowriter, the information must first be loaded or shifted into the A register. The code in the A register may then be transmitted to the Flexowriter by a Digital instruction with an operand track address of 00; the execution code is of no consequence when the Flexowriter is addressed in a Digital output command. The character printed and/or punched by the Flexowriter depends upon the contents of bit-positions 1 through 8 of the A register during a digital output to the Flexowriter.

The Flexowriter's response to digital outputs from the RW-300 depends upon the selected mode of Flexowriter operation, and depends upon the pattern of bits in the A register. The mode of Flexowriter operation can be controlled either by switches above the Flexowriter keyboard, or by digital output signals sent to the Flexowriter from the RW-300.

Bit patterns in the A register which control the Flexowriter's mode of operation are called "control codes". Specific bit patterns, or codes in the A register activate a typewriter key or cause some other Flexowriter response (carriage return, space, etc.). Codes which neither control the Flexowriter nor cause some form of typewriter response, are considered "illegal" codes. "Legal" Flexowriter codes are tabulated in figure 5-2.

Modes of Flexowriter operation include:

a. Print

b. Punch

c. Print-and-Punch

The characteristics of these operating modes are explained in conjunction with the detailed Flexowriter characteristics described at the end of this section.

To effect printout of symbols from the A register, the Flexowriter must be in the Print mode of operation.

Sample Printout Listing

The following example illustrates the use of Digital input and output commands to obtain a signed, two-digit, octal printout of a specific memory location. The octal number in this example is -35, which is stored in memory as 750000, but the printout listing is valid for the sign and the first two octal digits of any number.

To print and/or punch a symbol on the Flexowriter, the appropriate Flexowriter code for that symbol must be placed in bit-positions 1 through 8 of the A register, and the computer must execute a Digital output instruction addressed (track 00) to the Flexowriter.

In the sample listing, the number in memory is loaded into the A register, and the sign is examined by means of a Transfer on Negative instruction. If the sign of the number is negative, the next instruction loads the A register with the Flexowriter code for a negative sign, and the following Digital output instruction causes the Flexowriter to print "-". If the sign of the number is positive, the next instruction loads the A register with the Flexowriter code that causes the Flexowriter to print "+".

After the sign has been printed, the A register is cleared using a Digital input instruction with an execution code of zero; the two

most-significant bits of the number to be printed are then shifted from the B register into the A register. If the first two bits are both zero, the Flexowriter will execute a space in response to the Digital output command which follows. If the first two bits are not zero, the Flexowriter will print the appropriate octal digit.

The A register is cleared in preparation for printing the second digit, this time by a right shift. The right shift removes the two bits from the A register, but does not shift the bits into the B register. The second octal digit in the B register is brought into the A register by a left shift, and is then printed.

	CI		EX	OPERAND		OP	NI	REMARKS
A	00-00	I	00	01-04	I	29	00-06	M → A
A	00-06	I	00	00-00	I	02	00-12	A → A, B
A	00-12	I	00	00-18	I	09	00-16	Test for negative sign
A	00-16	I	00	01-20	I	29	00-24	+ → A
A	00-18	I	00	01-22	I	29	00-24	- → A
A	00-24	I	00	00-00	I	06	00-34	print sign
A	00-34	I	00	32-00	I	06	00-40	0 → A
A	00-40	I	02	48-00	I	01	00-48	A $\overset{2}{\leftarrow}$ B
A	00-48	I	00	00-00	I	06	00-58	print first number
A	00-58	I	02	00-00	I	01	00-66	0 → A
A	00-66	I	03	48-00	I	01	00-74	A $\overset{3}{\leftarrow}$ B
A	00-74	I	00	00-00	I	06	TT-SS	print second number

	ADDR.	CONST.	REMARKS
A	01-04	C750000	number to be printed
A	01-20	C000040	Flex code for positive sign
A	01-22	C000021	Flex code for negative sign

The above listing does not print zeros, but does print octal digits 1 through 7. The non-parity* Flexowriter codes for symbols 1 through 7 correspond to those bits as they appear in the A register: 0000001 in the A register causes a 1 to be printed on the Flexowriter; 0000010 in the A register causes a 2 to be printed, etc. However, zero (0000000 in the A register) is the Flexowriter code for a space.

To print zeros, the above printout listing would have to be modified so that each octal digit would be tested to determine whether it is zero. If not zero, the next instruction would be a digital output to the Flexowriter. If the A register contents were zero, the next instruction would load the A register with the Flexowriter code for zero before the Digital output command to the Flexowriter. As in the case of the sign printout, the Flexowriter code for zero would have to be stored in some predetermined location of memory.

The preceding example is not an example of minimum-time programming, but is presented to show a maximum number of command variations. The time required to execute a Flexowriter output is six word times plus any waiting time that is characteristic of the Flexowriter or other output device.

---

*"Parity" and "Non-Parity" Flexowriter codes are described in conjunction with Flexowriter characteristics at the end of this section.

The RW-300 Program Library includes decimal as well as octal printout routines.

EXPANDED INPUT-OUTPUT CAPABILITIES

Inputs

The digital input capabilities of the RW-300 can be expanded to a maximum of 28 additional groups of 18 lines each, or 504 additional on-off signals, with each group selected by a different track address (36 through 63) of the DG command. The track addresses are assigned to specific input functions or devices. If a Ferranti high-speed reader is used in addition to the Flexowriter, 33 is the operand address track number

When any one of the input groups is addressed (operand track address 36 through 63), that group will be connected to lines L21 through L38, but the input will not be complete unless the DIGITAL INPUT selector switch on the test and maintenance panel is turned to EXTERNAL. If the DIGITAL INPUT selector switch is set to INTERNAL, any operand track address from 36 through 63 will cause the toggle switches to be read on lines L21 through L26.

In an expanded digital input system, as contrasted to the basic digital input system, grounded lines are read in as zeros, and open lines are read in as ones. In all other respects, including the effect of the execution code, digital input characteristics are the same as those described for the basic computer.

Some of the digital input equipment which has been used in RW-300 applications is described in the paragraph titled "Input-Output-Equipment."

Outputs

The digital outputs provide a means of transmitting on-off signals to external devices. The output signals may be used for turning the Flexowriter

on and off, activating indicator lights or alarm devices, etc. Up to 28 groups of 18 relay-controlled output lines, or 504 outputs, are available as optional equipment with the basic computer. On special order, the output system may be expanded to 30 groups if no high-speed punch is included.

A 10-millisecond delay cricuit allows time for the relays to change their output state (one-to-zero, or zero-to-one), and this 10-millisecond period must elapse before a subsequent Digital command can be executed. Once a relay has been set to one or zero, it remains in that state until changed by another Digital command affecting that particular relay.

Each group of output lines is selected by an operand address track number (04 through 31).

If a logging typewriter is used in addition to the Flexowriter, track 01 is the operand track number for this typewriter. If a high-speed punch is used, 02 is the track number; track 03, in this latter case, may not be used as a digital output address, and tracks 34 and 35 may not be used as digital input addresses. RW-300 computers with the expanded memory and a high-speed punch may use track 03 output address for selecting track set writing control.

The number of groups of output lines and the characteristics of the output-line groups depend upon the needs of the installation. Each group of 18 output lines may be connected in either one of two modes: "multi-bit" output or "one-bit" output. The output modes are selected when the equipment is fabricated for a particular installation. Characteristics of operation are described in the paragraphs which follow.

One-Bit Outputs

"One-bit" outputs affect only those relays (in the addressed group) which correspond to A-register bit positions containing a one. If the execution code is an even number, the affected relays are set to zero. If the execution code is an odd number, the affected relays are set to one. Thus, any or all of the 18 relays in a particular one-bit output group can be addressed by a Digital output instruction. One-bit outputs are often employed to control peripheral equipment as tabulated below:

Execution Code	Operand Address	Operation Code	Contents of A Register*	Function
00	04-00	06	$A_1 = 1$	Turn Flexowriter Off
01	04-00	06	$A_1 = 1$	Turn Flexowriter On
00	04-00	06	$A_2 = 1$	Turn Logging Typewriter Off
01	04-00	06	$A_2 = 1$	Turn Logging Typewriter On
00	04-00	06	$A_9 = 1$	Turn Punch Off
01	04-00	06	$A_9 = 1$	Turn Punch On

* $A_n = 1$ means: bit position n of the A register contains a one.

Although most applications require that a relay which has been set to one or zero retain that state until changed by another digital output addressed to that relay, specific one-bit output lines may be connected to provide a momentary signal. This latter configuration is used where "set" or "reset" signals are required. When the momentary feature is provided, an even execution code in the Digital instruction will cause a 10-millisecond contact closure in the output line specified by a one in the A register. An odd execution time will cause no output signal.

Multi-Bit Outputs

A "multi-bit" output is defined as one in which all relays in the addressed 18-bit group are set to one or zero, according to the contents of each corresponding bit in the A register. In the case of multi-bit outputs, the execution code is of no consequence. An installation usually employs both multi-bit and one-bit outputs, but these are separate output groups, activated by separate operand track addresses.

Outputs to a logging typewriter are typical of multi-bit outputs, in which all output lines provide voltages corresponding to the contents of the A register. In this application only 5 of the output lines would feed the logging typewriter, and the application is considered a "five-bit" output.

INPUT-OUTPUT EQUIPMENT

The types of input-output equipment included in an RW-300 installation depend upon the needs of the user. Computer flexibility permits a wide choice of display, attention-seeking, printing, punching, and data-insertion devices. Several commonly used devices are described in the paragraphs which follow.

Digital Indicators

To display all or part of a computer word, a bank of 18 indicator lights may be used to represent the binary word in a specific register. A light which is ON represents a one, and a light which is OFF represents a zero.

Twenty-Four-Hour Clock

Mounted in the upper right-hand corner of the control console shown in figure 1-1, the 24-hour clock provides a visual indication of time and provides the RW-300 with absolute time to the nearest minute or 1/10 minute. The clock is used as a time reference by the program for periodic control calculations, data-logging cycles, instrument calibration checks, catalyst checks, etc.

Driven by a synchronous motor, the clock provides fourteen one-bit signals from a system of stepper switches. Thirteen bits represent real time in a complemented, binary-coded-decimal format. The fourteenth bit is a "ready" signal to the computer, used to avoid incorrect time information during changes from one time state to the next. When the ready signal is present, the time can be read into the A register in response to a Digital input command. Memory assignments for clock information are as follows:

Operand Track Address = 37

Bit Position	Assignment
1	Minutes
2	Minutes
3	Minutes
4	Minutes
5	Tens of Minutes
6	Tens of Minutes
7	Tens of Minutes
8	Hours
9	Hours
10	Hours

Bit Position	Assignment
11	Hours
12	Tens of Hours
13	Tens of Hours
14	Unassigned
15	Unassigned
16	Unassigned
17	Unassigned
18	Clock "Ready" Signal

Visual indication of the time is given by a direct reading indicator in peripheral equipment, with midnight as 0000 and the end of the day as 2359. Each digit may be up-dated by one of four pushbutton switches to allow adjustment of the clock to local time.

Manual Inputs

Manual inputs are provided to permit the operator to enter instructions or data into the control program. Instructions are sometimes used to print out selected sectors of memory under program control. Data entries are sometimes used to provide the control program with process operating information that is not fed in automatically through the analog input system. This type of data may represent a process variable that changes slowly and is costly to instrument in analog form; or the information may represent a unique operating mode.

A group of toggle switches is sometimes provided so that the information or instruction can be entered in binary form by setting the toggle switches to the desired binary pattern.

In figure 1-1, the white square in the center of the operating panel (below and to the right of the logging typewriter) is a matrix indicator. The matrix indicator is composed of four rows and four columns. Two selector switches, below and to the left of the matrix indicator, are used to select any one of the sixteen possible inputs designated in the matrix squares. The matrix switch and indicator enable the operator to select a specific input function.

The information to be inserted into the program is set on Digitran switches which are located on the control panel to the right of the matrix. The binary-coded equivalent of the decimal or octal number set into the Digitran switches is inserted into the program only after the operator presses an EXECUTE MATRIX button on the control panel.

Programming considerations cause the information to be read in only at a time acceptable to the program, so that control calculations are not interrupted. If the operator desires that the information be read into the program immediately, he can accomplish this by pressing the START button on the RW-300 control panel. The degree of priority assigned to manual inputs is a programming consideration which depends upon the application. In some applications the computer accepts the new data on a tentative basis, performs predictive calculations to determine how the new data will affect the process, and prints out the result so that the operator can judge whether or not the new data should be inserted.

Watchdog Timer

A watchdog timer is a fail-safe device which is incorporated to provide periodic checks of computer operation. The timer is of the rundown type and must be reset periodically. The rundown time is adjustable from approximately 1 second to 30 minutes.

The computer periodically sends a one-bit output to the timer for resetting purposes. If the computer fails to send this one-bit output within the required time, the timer will run down and generate an output which may be used to halt the computer, send it back to start, sound alarms, etc.

In a typical application of the watchdog timer, the program may include instructions to perform periodically a series of operations (add, subtract, shift, etc.) using a converted known analog voltage. The result of these operations is compared with a stored constant which represents the correct solution. If the comparison proves the computations to be correct, a one-bit output is executed to reset the timer. An error in analog input conversion or in any of the arithmetic or logical operations would result in failure to reset the timer. The timer would run down, signal the operator, and send the computer to halt. The last correctly calculated control signals would be maintained.

Ferranti Reader

When required, a Ferranti seven-level paper-tape reader provides high-speed tape input. The input from this unit may be permanently substituted for the paper-tape reader of the Flexowriter (i.e., it can be made the only means for reading paper tape). However, it is also possible to retain

both the Flexowriter and Ferranti facilities, with selection being made by the operand track address or by a one-bit digital output.

Use of the Ferranti tape reader requires additional circuits to make the Ferranti signal levels compatible with those of the RW-300. The Ferranti is modified to provide an interlock feature ("ready" signal). The maximum speed of the Ferranti used with the RW-300 is 60 frames per second; a Digital input command must not be addressed to the Ferranti more frequently than once per RW-300 drum revolution.

Teletype Punch

A Teletype seven-level paper-tape punch may be attached to the RW-300 for high-speed tape output. This punch is controlled by the program which may choose between this unit and the Flexowriter by the use of the operand track address, or by a one-bit digital output. The punch is modified to provide an appropriate "ready" signal to the RW-300.

The Teletype high-speed tape punch uses a 3600-rpm, 60-cps, synchronous motor and has a maximum punching rate of 60 characters per second. This maximum rate can be achieved only when doing a series of outputs with the following characteristics: the number of word times between the Load A instructions and the corresponding Digital output instruction must be equal to, or less than, 17. If this time is exceeded, the punching rate will be 30 characters per second. To be certain of achieving the required minimum timing, there must be no Digital input instructions between successive Digital output instructions. If this latter requirement is not met, the average punching rate will be from 30 to 60 characters per second.

FLEXOWRITER

Modes of Flexowriter Operation

The Flexowriter operates under the control of the RW-300 when paper-tape information is being read into the computer, and when information from the computer is being recorded in printed form and/or punched-tape form on the Flexowriter. In addition, the Flexowriter is operated independently of the computer when program listings are being typed and/or punched, and when punched tapes are being duplicated or printed.

The modes of Flexowriter operation include:

a. Print

b. Punch

c. Print-and-Punch

These modes of Flexowriter operation can be controlled either by the computer, or by switches located above the Flexowriter keyboard.

Each of the switches above the Flexowriter keyboard, as well as the keys on the keyboard, represent some Flexowriter function. Each Flexowriter function has a corresponding Flexowriter code which can be represented by a pattern of punched holes in the Flexowriter paper tape and by a bit pattern in the computer's A register. Any hole pattern or bit pattern which does not represent a Flexowriter function is called an "illegal" code. Among the "legal" codes recognized by the Flexowriter are four control codes:

a. Punch On

b. Punch Off

c. Non-Print

d. Print Restore

These control codes affect the mode of Flexowriter operation in the same way as the corresponding switches located above the Flexowriter keyboard. When the Flexowriter is operating under computer control, a digital output instruction addressed to the Flexowriter is used to send the control codes from the computer's A register to the control circuits of the Flexowriter.

A digital output to the Flexowriter from the A register with the control code 224 (Print Restore) will cause the Flexowriter to operate in the Print mode. In the Print mode, the Flexowriter does not respond to illegal codes, nor does it respond to the two legal codes Tape Feed and Stop Code when these codes are sent to the Flexowriter from the computer. All other legal Flexowriter codes are executed, regardless of the contents of the parity* bit. Flexowriter codes are tabulated in figure 5-2.

The Flexowriter printing capability can be disabled manually by the Non-Print switch on the Flexowriter, or by a digital output to the Flexowriter from the computer with the control code 230 (Non-Print).

A digital output to the Flexowriter from the computer's A register with the control code 250 (Punch On) will put the Flexowriter in the Punch mode. Subsequent digital outputs to the Flexowriter will punch out on tape

*The parity bit is used as an error-checking feature which can be incorporated in the standard Flexowriter. Use of the error-checking feature is described along with other Flexowriter characteristics near the end of this section.

any bit pattern except control codes contained in A-register bit-positions 1 through 8 -- provided the Flexowriter is not in the Print-and-Punch mode. This feature is necessary to obtain a "binary dump", or copy of the bit patterns in computer memory.

When in the Punch mode, a control code (such as Punch Off or Print Restore) sent to the Flexowriter by the computer causes the Flexowriter to execute that code (change the mode of Flexowriter operation) and also punch the pseudo code 377 on paper tape. A digital output from the A register with the control code 244 (Punch Off) will cause the Flexowriter to punch the pseudo code, and then stop punching. If the punch/non-punch function is controlled by a one-bit digital output, Flexowriter response to the punch/non-punch codes can be eliminated, and bit patterns representing the control codes may be punched without changing the mode of operation.

The Print-and-Punch mode of operation is achieved under computer control by sending Punch On and Print Restore control signals to the Flexowriter. When in this mode of operation, the Flexowriter will respond only to legal codes. Any character code, or functional code such as carriage return, will be executed by the typewriter and will be punched on tape. Illegal codes are neither punched nor printed. The Tape Feed code is punched, but there is no typewriter response. The Stop Code is punched, but the Flexowriter does not stop. The four Flexowriter control codes will be executed,

and the 377 code will be punched.

The Flexowriter can be operated in any of the three modes independently of the computer. The Flexowriter's Print Restore switch puts the Flexowriter into the Print mode. In this mode, the contents of a paper tape threaded through the Flexowriter tape reader will be printed out when the Flexowriter's Start Read switch is pressed. The Flexowriter stops reading when it comes to a Stop Code punched on the paper tape, or when the Flexowriter's Stop Read switch is pressed. There is no typewriter response to the Tape Feed Code, or to illegal codes.

In the Punch mode, a paper tape threaded through the Flexowriter tape reader can be duplicated by the Flexowriter paper-tape punch. This independent operation is achieved by pressing the Flexowriter Punch On switch and Start Read switch. The punch will duplicate all codes punched in the tape, will respond to the four control codes, but will not respond to the Stop Code -- provided the Flexowriter is not in the Print-and-Punch mode.

When operating "off-line", independently of the computer, the Flexowriter can be placed in the Print-and-Punch mode by pressing the Print Restore and Punch On switches on the Flexowriter. In this independent mode, a paper tape threaded through the Flexowriter can be duplicated and its contents printed by pressing the Flexowriter's Start Read button. Only legal codes will be typed and punched. Illegal codes will be ignored. The Flexowriter will not respond to the four control codes, but will stop reading the tape when it comes to the Stop Code.

When punched tapes are prepared on the Flexowriter, blank tape following the last punched frame may be obtained by proceeding as follows:

a. Press Punch On switch above the Flexowriter keyboard.

b. Hold down Card Feed microswitch near Flexowriter punching mechanism.

c. Press, then release Tape Feed switch above the Flexowriter keyboard.

Blank tape (with tape-feed holes) will be produced until the Card Feed microswitch is released. The Punch Off switch above the Flexowriter keyboard may then be pressed if the next Flexowriter operation (e.g., program loading) does not require the punch mode of operation. However, if the Punch Off switch is pressed before releasing the Card Feed microswitch, the read lines to the computer will be temporarily disabled, and the next digital input from the Flexowriter will be read as all zeros, regardless of the tape code under the Flexowriter tape reader. Thus, the Card Feed microswitch should always be released before pressing the Flexowriter's Punch Off switch.

General Flexowriter Characteristics

The Flexowriter normally supplied with the RW-300 has the following features:

a. 16-inch carriage

b. Reader and punch which handle paper tape and edge-punched cards

c. Separate "Red" and "Black" codes for color shift

d. Elite Gothic type

e. Large capital letters and numerals in lower case; small capital letters and various symbols in upper case

f. Separate codes for Non-Print, Print Restore, Punch On, and Punch Off modes of Flexowriter operation

g. Alphabet, numerals 0-9, and punctuation identical with IBM 705 standard code

h. Special keys carrying the numerals 10, 11, and 12

i. Provisions to receive folded or roll paper up to 15 inches wide, typing a line up to 13 1/2 inches long.

j. No special code-delete feature. Paper-tape frames can be deleted using the Tape Feed switch above the Flexowriter keyboard Information cannot be readily deleted from edge-punched cards.

k. Ability to duplicate tapes, regardless of code legality

The Flexowriter inhibits reading into the computer if the Flexowriter is in the Print mode. Therefore, reading into the computer and printing cannot be performed simultaneously.

The Flexowriter Punch mode is selected through the use of Flexowriter codes or digital outputs under program control, depending on the Flexowriter wiring. The Flexowriter Print mode is selected through the use of Flexowriter codes only.

In a basic (unexpanded) RW-300 system, a Digital command with an operand track address of either 01 or 33 will turn the Flexowriter motor on or off, depending upon whether the least-significant bit of the A register is

a one or a zero, respectively. In an expanded system, the Flexowriter motor is normally controlled by a one-bit output.

At the customer's option, the Flexowriter can be provided with the following features:

a. A No. 2-pin feed platen can be used with the Flexowriter, giving a maximum usable paper width of 13-1/8".

b. By addition of a selector bar tab from the Flexowriter, the parity bit can be punched in the Print-and-Punch mode.

c. Removal of Input Relay No. 7 prevents parity bits from entering the computer.

d. Removal of Output Relay No. 7 prevents parity bits from being sent out by the computer.

e. A separate one-bit digital output to control the Flexowriter punch can be supplied to avoid any undesired codes appearing on the tape when turning the punch off.

f. Insertion of a parity channel permutation bar into the Flexowriter will place the Flexowriter under complete parity control in the Print and Print-and-Punch modes.

The paragraphs which follow describe the form of the punched paper tape used with the Flexowriter, the "parity" option, Flexowriter codes, and timing considerations.

Paper Tape

The paper tape used by the Flexowriter, and by most input-output units is a standard eight-level paper tape, one inch wide. In the tape shown in

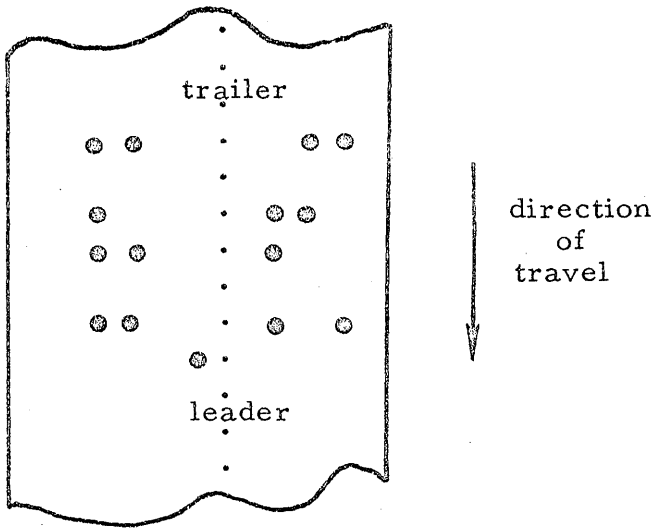


Figure 5-1 Segment of Punched Tape

Figure 5-1, a hole punched in the tape is used to represent a one, and a blank (no punch) to represent a zero. Thus, a row across the tape (a "frame") may be used to represent a binary number. The least-significant bit appears along the right-hand edge of the tape shown in figure 5-1. The small holes which lie between level 3 and level 4 are sprocket holes which are used to time and guide the movement of the tape in all tape units. Blank tape ahead of the first punched frame is called "leader", and blank tape behind the last punched frame is called "trailer".

A frame of punched holes may be represented by a three-digit octal number. In this presentation, the sprocket holes separate the two least-significant octal digits. The five non-zero frames shown in figure 5-1 represent, from top to bottom, the octal numbers 143, 106, 144, 145, and 10. Various combinations of punched holes are used to represent characters and to control the electric typewriter. From the list of Flexowriter codes in

figure 5-2 it may be seen that the five punched frames in figure 5-1 represent the letters "C", "O", "D", "E", and the decimal number "8".

Parity Checking

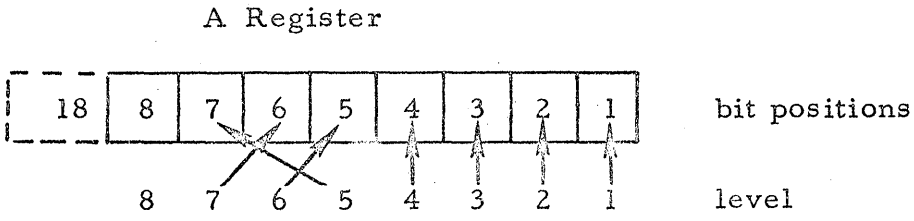
The Flexowriter incorporates a seven-channel code for digital inputs. Six of the channels contain the desired input character, and the seventh channel (level 5 on the tape) may be used for "parity" checking.

In general, parity check makes use of a code employing binary digits in which the total number of ones (or zeros) in each permissible code is always odd or always even. In the Flexowriter parity check, the total number of ones must always be odd. Thus, if the character punched on the tape contains an odd number of ones, the parity channel will contain a zero. Conversely, if the character punched on the tape contains an even number of ones, the parity channel will contain a one.

Any correctly punched tape employing parity checking will always contain an odd number of ones for each character. This feature enables the computer to be programmed to detect errors that could occur during tape punching or reading. However, parity checking is a programming option that is not normally employed simply to check the loading of a program punched on tape. Program loading can be checked more easily using the check-sum capability of the load program (Section VI). However, a parity-checking capability is desirable when information of uncertain accuracy is being read into the computer. For example, information transmitted over land lines to a paper-tape punch can be verified by parity checking.

System specifications usually indicate whether or not the parity-checking option is to be included in the features of the Flexowriter used with the system. If parity checking is to be employed, the Flexowriter includes provisions for handling a parity bit in level 5 of the punched tape. The punched tape shown in figure 5-1 is without a parity bit; the fifth column (level) from the right edge of the tape does not contain any punched holes, even though some of the characters are represented by an even number of bits.

The bits read from the seven levels of punched Flexowriter tape change positions on entering the computer as shown below.



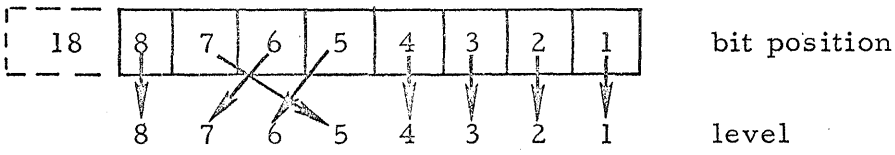
This locates the parity bit (level 5 on the tape) in the highest position of the A register (normally bit-position 7) where it can be readily extracted by the program.

When parity checking is desired, the load program must include instructions to determine whether the sum of the seven bits is odd or even. The load program must also include instructions specifying the desired action in the event of a parity error. Furthermore, after the parity check has been made, some action must be taken to ensure that the parity bit is zero before entering the input information into the program.

If the parity-checking feature is not included in the Flexowriter, the parity bit is never present, and no special programming is required to elimi nate that bit before the input character is interpreted by the load program. However, if the parity feature is included in the Flexowriter, the parity bit must be suppressed--whether or not the parity bit was employed to perform a parity check. The Extract instruction provides a convenient technique for suppressing the parity bit. The technique is outlined in conjunction with a description of digital inputs from the toggle switches.

When a tape is punched with the Flexowriter under program control, any seven-bit pattern in the A register of the computer will be punched on the tape in response to a digital output command addressed to the Flexowriter. Upon leaving the computer, bits 5, 6, and 7 in the A register change position as shown below.

A Register



Tape Punch

The eighth line, or channel to the Flexowriter (corresponding to the eighth level on the paper tape) is not used for character codes, but is used in the four Flexowriter control codes: Non-Print, Print Restore, Punch Off, and Punch On. Therefore, the programmer must be aware of the contents of bit-position 8 in the A register when programming digital output commands addressed to the Flexowriter.

In this section of the manual, a sample printout listing was used to illustrate the method of printing out non-zero octal numbers on the Flexowriter. The technique described is valid if the Flexowriter is not equipped for parity checking. However, if the parity-checking feature is incorporated in the Flexowriter, parity conditions must be satisfied, and the A register must contain an odd number of bits if the number is to be printed on the Flexowriter. The sample printout listing would have to be modified to test for an even number of bits in the A register prior to the digital output command; if the number of bits were even, a one bit would have to be placed in bit-positon 7 of the A register to form a legal Flexowriter code. A one may be placed in bit-position 7 of the A register by merging (operation code 31) the octal constant 000100 with the contents of the A register.

Flexowriter Codes

The octal numbers representing Flexowriter characters are tabulated below. The octal numbers used by the programmer in conjunction with inputs and outputs depend upon whether the system's Flexowriter uses the parity-checking feature described above.

Flexowriter Timing Considerations

Typing, reading, punching, or punching-and-typing all occur at about eight characters per second on the Flexowriter.

When a digital input command is sent to the Flexowriter, a frame of paper tape in the Flexowriter tape reader is read into the computer's A register. A series of digital input commands addressed to the Flexowriter are executed at the nine-character-per-second rate established by the Flexowriter.

Figure 5-2. Table of Flexowriter Codes

FLEXOWRITER CHARACTER		OCTAL CODES ON TAPE		OCTAL CODES IN A REGISTER	
UPPER CASE	LOWER CASE	WITH PARITY	WITHOUT PARITY	WITH PARITY	WITHOUT PARITY
A	A	141	141	61	61
B	B	142	142	62	62
C	C	163	143	163	63
D	D	144	144	64	64
E	E	165	145	165	65
F	F	166	146	166	66
G	G	147	147	67	67
H	H	150	150	70	70
I	I	171	151	171	71
J	J	121	101	141	41
K	K	122	102	142	42
L	L	103	103	43	43
M	M	124	104	144	44
N	N	105	105	45	45
O	O	106	106	46	46
P	P	127	107	147	47
Q	Q	130	110	150	50
R	R	111	111	51	51
S	S	62	42	122	22
T	T	43	43	23	23
U	U	64	44	124	24
V	V	45	45	25	25
W	W	46	46	26	26
X	X	67	47	127	27
Y	Y	70	50	130	30
Z	Z	51	51	31	31
)	0	40	40	20	20
'	1	1	1	1	1
"	2	2	2	2	2

Figure 5-2. Table of Flexowriter Codes -- Continued

FLEXOWRITER CHARACTER		OCTAL CODES ON TAPE		OCTAL CODES IN A REGISTER	
UPPER CASE	LOWER CASE	WITH PARITY	WITHOUT PARITY	WITH PARITY	WITHOUT PARITY
#	3	23	3	103	3
$	4	4	4	4	4
%	5	25	5	105	5
¢	6	26	6	106	6
&	7	7	7	7	7
*	8	10	10	10	10
(	9	31	11	111	11
°	10	112	112	52	52
?	11	133	113	153	53
:	12	54	54	34	34
—	,	73	53	133	33
.	.	153	153	73	73
/	-	61	41	121	21
=	+	100	100	40	40
	LOWER CASE	172	152	172	72
	UPPER CASE	174	154	174	74
	TAB	76	56	136	36
	SPACE	20	00	100	00
	BLACK	32	12	112	12
	RED	114	114	54	54
	CARRIAGE RETURN	136	116	156	56
	PUNCH ON	-	-	250	250
	PUNCH OFF	-	-	244	244
	NON PRINT	-	-	230	230
	PRINT RESTORE	-	-	224	224
	TAPE FEED	177	157	177	77
	STOP CODE	13	13	13	13

When a digital output command is sent to the Flexowriter, a 110-millisecond timing circuit is activated. No more information can then be sent to the Flexowriter or obtained from it until the 110 milliseconds have passed. The Flexowriter delay circuit constitutes an interlock which prevents the computer from transmitting information to the Flexowriter before that information can be accepted by the Flexowriter for printing and/or punching. However, other instructions can be executed during this time, and it remains the programmer's option to make maximum use of computer time by inserting other instructions in the program between digital output commands.

A digital input command sent to the Flexowriter must not be followed immediately by digital output commands to the Flexowriter. The program must include a waiting time of at least seven drum revolutions between digital input and digital output instructions involving the Flexowriter. If this waiting time is not included in the program, the first digital output instruction will not be executed. There is no programming restriction on placing digital input instructions immediately after digital output instructions--a delay circuit within the computer prevents information loss.

On digital output to the Flexowriter, a longer delay than 110 milliseconds may be required for a long tab or a carriage return. This longer delay can be programmed by inserting a trivial Flexowriter command, such as lower-case shift. Because of the 16-inch carriage, the programmer must output a number of dummy characters after each carriage return to permit a full return of the carriage before further printing. The same statement holds

true for execution of a tab. As a guide, the following will permit safe carriage returns:

Number of dummy characters	Point from which returning the carriage
1	2 inches
2	4 inches
3	7 inches
4	9 inches
5	10 inches
6	12 inches

These may vary slightly from system to system.

SECTION VI

PROGRAM LOADING

INTRODUCTION

Procedures for preparing and organizing program listings are described in Sections III, IV, and V. This section of the manual deals with loading programs into the computer and typing the program listings on the Flexowriter. Certain typing, or format restrictions are necessary to enable the computer's load program to interpret the symbols punched on the tape.

The paragraphs which follow contain a brief description of the load program, followed by a summary of typing format requirements and operating procedures.

LOAD PROGRAM

The load program, permanently stored on track 63 of computer memory, is started whenever the LOAD button on the computer's control panel is pressed. The load program controls digital inputs from the Flexowriter tape reader (or some alternate tape- or card-reading device).

Each time the load program performs a Digital input instruction, a frame of tape (one symbol) is read into the A register, and the tape advances so that the next frame can be read in response to the next Digital input instruction.

The load program examines each symbol read into the computer from the tape reader, and, upon receipt of certain "indicator" symbols, proceeds to assemble the information for storage in specific memory locations.

The listings described in Sections III, IV, and V contain indicators having the following significance:

Indicator	Significance	
"A"	Address.	The location in which an instruction or constant is to be stored.
"C"	Constant.	An octal number to be stored in the location specified.
"I"	Instruction.	An execution code and operand address, or operation code and next-instruction address, to be stored in the location specified.

A complete instruction requires two sequential sector locations. Only the first sector is specified in the listing (CI column) and is punched on the tape; the load program stores the first word of the instruction in the specified sector, adds one to that sector number, and stores the second word of the instruction in the next sector. For sequential listings the storage location need be specified only once, and all subsequent instructions (or constants) will be stored in sequential sector numbers by the load program. After an instruction is stored in sector 127, the next instruction is stored in sector 00 of the next higher track. Note that if the next-instruction address specified in a program listing is sector 127, the program will read the first word

of the instruction from sector 127, but will read the second word of the instruction from sector 00 of the same track. Therefore, in sequential listings, it is necessary to assign specific addresses to instructions that will be stored in sectors 127 or 00 in order to define the track number. In the case of an optimum program, the location of the first word of each instruction must be specified.

When a punched tape is threaded through the Flexowriter tape reader and the LOAD button pressed, the load program reads the blank tape leader until a meaningful indicator is received. Upon receipt of the indicator, the load program reads, interprets, and stores the instructions punched on the paper tape. At the end of the punched instruction information, the load program continues to read the tape trailer until the stop indicator "S" is read. Upon receipt of the "S" indicator the computer halts.

The load program can be halted while it is accepting information from the Flexowriter by pressing the STOP button; by pressing the RESUME button, loading can be resumed.

The load program also recognizes indicators "L", "M", and "J". The "M" indicator is used in conjunction with a memory check sum, a feature of the load program which permits loading accuracy to be verified. The "L" indicator is used to specify that tape symbols are in binary format. The "J" indicator causes the computer to leave the load program and begin another program at a specified address. The use of these indicators is described following the discussion of the standard punched tape format.

As an equipment option, the RW-300 load program can be prepared to recognize the indicator "D". Recognition of this indicator temporarily transfers program-loading control to track 61. In track 61, a "Decimal Input Routine" interprets constants that have been listed in a decimal format. The Decimal Input Routine is part of the RW-300 Program Library.

STANDARD PUNCHED TAPE FORMAT

Punched tapes prepared for loading under the direction of the load program are usually typed and punched using spaces, tabs, and carriage returns for maximum readability. The basic format is as follows:

Normal Instruction Format

```
  s          s              s              c
  p      t   p    t     t   p    t         a
                                           r.
A a TT-SS a I a XX a TT-SS a I a XX a TT-SS
  c      b   c    b     b   c    b         r
  e          e              e              e
                                           t.
```

Normal Octal Constants Format

```
  s               c
  p        t      a
                  r.
A a TT-SS  a CXXXXXX
  c        b      r
  e               e
                  t.
```

Thus, the listings

	CI		EX	OPERAND		OP	NI
A	42-126	I	08	00-110	I	16	20-106

	ADDR.	CONST.
A	08-19	C437152

would be typed

A 42-126 I 08 00-110 I 16 20-106

A 08-19 C437152

Note that special Flexowriter keys are used so that only two characters are required to obtain the numbers 106, 110 and 126.

Format requirements demand that no extra characters or spaces appear between characters shown as closely spaced groups in the above illustrations. For example, if 08-18 were typed 08-1 8, the address would not be interpreted correctly by the load program; similarly, C437152 must not be typed C 437152.

The spaces, tabs, and dashes suggested for the normal instruction format can be replaced by any convenient character. However, one character or space must appear in these positions on the punched tape. For example, if 08-18 were typed as 08Q18, the address would not be misinterpreted by the load program.

Care must be exercised in correcting typing errors when preparing a punched tape. A tape feed (octal 157) must not be used to delete an incorrect character in the six frames following an "A", nor in the nine frames following an "I", nor in the six frames following a "C". If an error is made following an indicator, the entire word, including the indicator, must be deleted.

The programmer should not enter data from the punched tape directly into the revolver track during the load operation. Since the load program itself uses the revolver extensively, any data entered from the punched tape may be destroyed by the load program.

DECIMAL PUNCHED TAPE FORMAT

The load program, when modified and used in conjunction with the decimal input routine, will accept numbers in either integral, fractional, or mixed form. The decimal numbers will be converted to their binary equivalents, correctly scaled, and stored in the indicated track and sector.

The decimal numbers to be entered must be within the range 0.00001 N 131,071.

Two examples of the tape format used to enter decimal constants are shown.

```
 s       s          c
 p       p          a
AaTT-SSaD-103.025   r.
 c       c          r
 e       e          e t.
```

```
 s       s  s       c
 p       p  p       a
AaTT-SSaDa13.0275   r.
 c       c  c       r
 e       e  e       e t.
```

The D indicator is recognized by the load program as meaning that the signed information to follow is in decimal form, and must be converted to the binary equivalent.

The sign of the decimal number immediately follows the D indicator. A space is accepted as a positive sign, and a hyphen is accepted as a negative sign.

When the format above is used, the binary scaling of the converted number is a function of the toggle switches, S1 - S6, on the computer test and maintenance panel. If the operator had previously set the switches to the configuration, 000110, the decimal numbers entered would be scaled 2^{-6} after being converted to binary. (For scaling considerations, refer to Section VIII.)

When several decimal constants are to be loaded, each having different scale factors, the scale factor of each may be entered following the decimal number in the following manner:

A TT-SS D - 20.55-05 C/R

A TT-SS D 113.092-12 C/R

The first number will be scaled 2^{-5} after conversion, and the second number will be scaled 2^{-12} after conversion.

The configuration, A TT-SS D 0. C/R will load zero into the specified track and sector.

OPERATING CONDITIONS

When loading a program, the punched paper tape is threaded through the Flexowriter tape reader (or other tape-reading device). When using the

Flexowriter, power must be applied, and the Flexowriter must be in the non-print and non-punch modes.

The LOAD button is pressed on the RW-300 control panel to begin loading. To stop loading, the STOP button is pressed. To resume after a stop, the RESUME button is pressed.

When the LOAD button is pressed, the ERROR light on the test and maintenance panel (Section VII) will glow if the track selection plug is not mated with the appropriate jack on the track selection panel. Before the loading operation can be resumed, the track selection plug must be moved to the appropriate track group. Example: if some portion of the program is written for track 33, the track selection plug must be connected to the jack marked 32-39. Following a record-error indication, the track selection plug is moved to the appropriate position, and the RESUME button is pressed to continue loading. A record-error indication obtained by attempting to load into track 63 cannot be cleared. The program must be corrected to eliminate any instructions assigned to track 63.

If the loading operation halts before the program has been loaded, and if there is no record-error indication, and if the tape does not contain the indicator "S", the halt is probably due to a format error. The loading operation can be restarted by pressing the LOAD and/or RESUME buttons. Although the faulty word that caused the halt will not be loaded correctly, subsequent instructions will be loaded satisfactorily.

MEMORY SUMS

The load program keeps a running sum of all bits read from the punched tape during the loading process. When a punched tape is loaded for the first time, the memory sum can be noted for comparison with sums obtained during subsequent loadings of the same tape.

When the LOAD button is pressed, the memory sum is set to zero. Each address, constant, and instruction that is read in by the load program is algebraically added to form a running check sum.

To make use of the sum, the indicator "M" is punched on the tape, followed by six spaces (blank tape). The load program halts when the "M" indicator is read, and at this time the memory sum appears in the A register. The octal contents of the A register are read by the programmer using the oscilloscope on the test and maintenance panel (Section VII). The octal contents of the A register are then punched into the six blank spaces on the paper tape, immediately following the "M" indicator.

Subsequent loading operations from the same tape will cause the load program to halt if the running check sum does not agree with the memory sum punched at the end of the tape. The running check sum appears in the A register.

The memory sum can be placed anywhere along the tape. If the running check sum agrees with the memory sum read from the tape, the load program sets the running check sum to zero and resumes the loading operation. If the sums disagree, the load program halts. If the RESUME button

is pressed, loading will resume and the running check sum will be set to zero in preparation for comparison with the next memory sum.

Note that the running check sum and memory sum are neither true sums of all the bits read from the tape nor of all the bits stored in memory, but are numbers generated by adding addresses, constants, indicators, etc.

BINARY LOADING

The "L" indicator tells the load program that the hole patterns in the following three tape frames are to be interpreted as the binary representation of a computer word. Each frame following the indicator "L" is read as six binary digits, with holes representing ones.

```
  s
Ap TT-SSLBBBLBBB.....etc.
  a
  c
  e
```

The address (TT-SS) is the memory location assigned to the first binary word to be loaded. Following this location address are, in consecutive order, the words, 3 binary frames per word, each preceded by an "L". No other binary format is acceptable to the load program.

Although it is not convenient for the programmer to translate constants into the binary format, the translation is accomplished simply and effectively under program control.

Tapes can be prepared in the binary format by a programming aid called the "Utility Package". Among the subroutines contained in the Utility Package is a "binary dump" routine which can be used to punch a paper tape

representing the contents of specific memory tracks. The paper tape obtained under the control of the Utility Package is called a "standard binary dump". The load program in track 63 of the RW-300 reads tapes punched in the standard binary dump format, and loads the information into the computer in one-half to one-third the time required to load a tape prepared using the standard punched tape format described previously.

A tape prepared using the standard punched tape format is called a "listable" tape because it can be read by the Flexowriter (independent of the computer) to obtain a printout of each instruction. Any printout of a standard binary tape is meaningless.

The Utility Package can also be used to prepare a tape representing memory contents in the form of a "fast binary dump", but tapes in this format can only be loaded using the Utility Package.

JUMP INSTRUCTION

If the first instruction of a program is located in 00-00 (the origin) of memory, the first instruction will be read by the computer when the START button is pressed.

If the first instruction of a program is not located at the origin, the computer can be sent to the first instruction either by placing an unconditional transfer instruction in the origin, or by loading a jump instruction.

For unconditional transfer to a program beginning in memory location 56-96, the origin may be loaded with an instruction similar to the following:

	CI		EX	OPERAND		OP	NI
A	00-00	I	00	56-96	I	10	56-96

The above instruction uses the Transfer on Overflow operation code, but any transfer operation code or the No Operation code could be used. The Transfer on Overflow operation code is advantageous because it turns off the overflow indicator.

The jump instruction is read by the load program. By preparing a tape: A 56-96J and pressing the LOAD button, the programmer can cause the computer to read the first instruction from memory location 56-96 (or any other memory location designated).

The jump instruction can be punched at the end of a program being loaded so that the computer will begin executing the program as soon as the loading operation is finished.

Because tapes must be loaded into the computer with the Flexowriter in the non-print and non-punch mode, it is sometimes desirable to halt the computer just prior to the jump into the program. The halt permits the programmer to set the Flexowriter to printing or punching conditions required to obtain output information from the program. The halt feature can be incorporated by using the form: A 56-96SJ. The computer halts after reading the "S" indicator. When the programmer presses the RESUME button, the load program reads the "J" indicator and jumps to the designated memory location.

SECTION VII

OPERATING CONTROLS AND INDICATORS

OPERATOR'S PANEL

Only the operator's panel is accessible when the cover is down (See frontispiece.) Controls include:

a. POWER ON

b. POWER OFF

c. STANDBY

d. LOAD

e. START

f. STOP

g. RESUME

Each control is a pushbutton switch containing an indicator which glows to indicate the operating mode. The seven buttons may be divided into two categories: power controls and operating controls.

Power Controls

Pressing the POWER ON button connects the line voltage supply to the RW-300 and turns on the STANDBY light. At the end of a 2.5-minute waiting period, the POWER ON button must be pressed again to place the computer in operation.

Pressing the POWER OFF button disconnects the line voltage supply from the RW-300.

Pressing the STANDBY button turns on the STANDBY light, turns off the POWER ON light, and removes operating voltages from the computer. When the STANDBY button is pressed, line voltage is not disconnected from the computer, and operation can be restored immediately (no 2.5-minute delay) by pressing the POWER ON button.

The programmer is not normally concerned with the power controls, but only with the operating controls.

Operating Controls

Use of the operating controls, described in conjunction with program loading in Section VI, are summarized in the paragraphs which follow.

To load a program into the drum from the paper-tape reader, the track 63 load program must be placed in operation. By pressing the LOAD button, the next-instruction (N) register is set to all ones, the computer is placed in automatic operation, and the first instruction of the load program (Section VI) is read from memory location 63-127.

When the START button is pressed, the next-instruction (N) register is set to zero, the computer is placed in automatic operation, and the instruction located at the origin (address 00-00) is read. Thus, the first instruction in the program must be located in 00-00 if the START button is used.

Pressing the STOP button causes the computer to cease execution of the program after completion of the current instruction; the computer idles, and the STOP light is lit. The STOP light is also turned on when a STOP instruction is executed by the computer or when a record error occurs

(Section VI). The computer may be returned to continuous operation by pressing the RESUME button.

If the computer has stopped as a result of a manual stop, program stop, or record-error stop, it may be returned to continuous operation by pressing the RESUME button (if the record-error has been cleared). This action will cause the computer to proceed to the location indicated by the next-instruction address of the last instruction performed. When the program resumes, the STOP and ERROR lights will be turned off.

TEST AND MAINTENANCE PANEL

The test and maintenance panel shown in Figure 7-1 is located under the hinged lid of the RW-300 console cabinet. The panel contains controls and indicators used during program loading, program check-out, and maintenance.

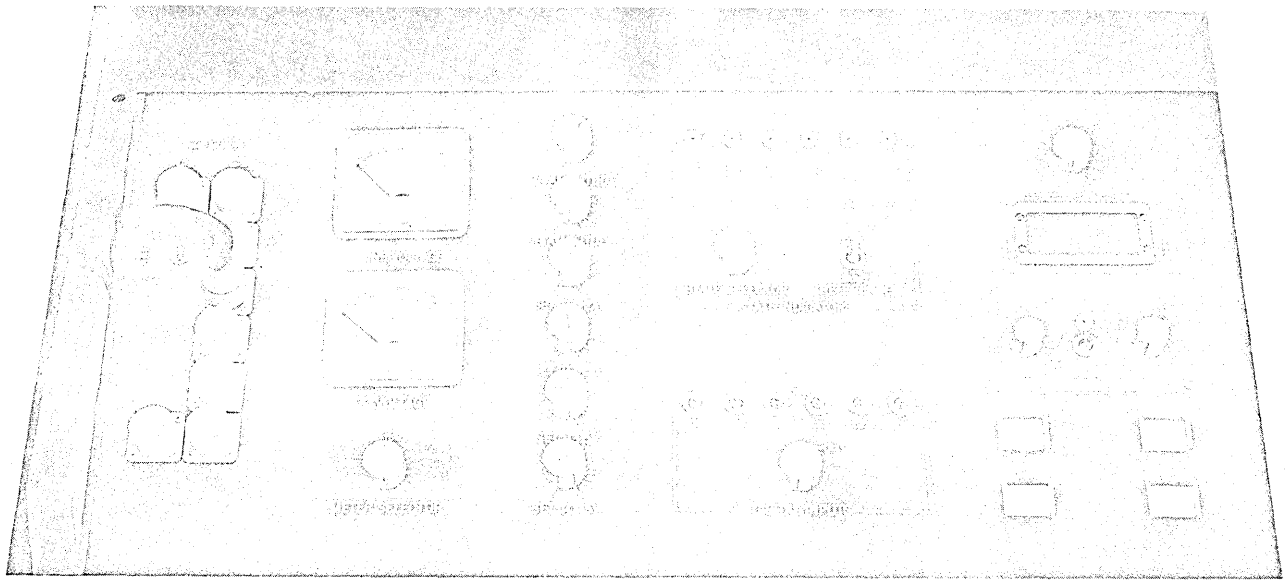


Figure 7-1. Test and Maintenance Panel

Note: Computers with an expanded memory have a three-way track set toggle switch above the Track 0-7 socket on the test and maintenance panel.

Program Loading

The controls and indicators used during program loading include:

a. Track selection plug and jacks

b. ERROR indicator light

c. Track set selection switch (expanded memory only)

The use of these in conjunction with program loading is described in Section VI. In addition, Section VI refers to obtaining track memory sums from the oscilloscope on the test and maintenance panel. The interpretation of the oscilloscope display is described in the paragraphs of this section.

Maintenance

The controls and indicators provided for maintenance purposes include:

a. LINE VOLTAGE meter

b. VOLTAGE meter

c. METER SELECTOR switch

d. Six controls for adjusting voltages and clock amplitudes

e. PHASE MARGIN

These controls are used to adjust operating voltages and to perform tests. They are not normally adjusted by programming or operating personnel.

Operation

The controls of interest to the programmer and operator under normal operating conditions include:

a. TRACK 0 & 7 WRITE, ON/OFF toggle switch

b. DIGITAL INPUT selector switch

c. L26, L25, L24, L23, L22, L21 toggle switches

d. ERROR indicator light

The TRACK 0 & 7 WRITE switch, located near the track selection plug, provides control over information written into tracks 00 and 07. Track 07 is the analog output track (Sections I and III).

Momentary power dips can cause the write heads of tracks 00 through 07 to store spurious information in these tracks. If the power dips should destroy the contents of the origin (sector 00 of track 00), the program could not be restarted; or if spurious information should be introduced into the analog output track, faulty control signals would be generated. Therefore, in process control applications, the TRACK 0 & 7 WRITE switch is normally turned OFF. To write in tracks 00 or 07 when the TRACK 0 & 7 WRITE switch is in the OFF position, any store instruction involving these tracks must be preceded by a one-bit digital output instruction which bypasses the protection circuit. The store instruction must then be followed by another one-bit digital output instruction to restore the protection circuit.

Although power dips might destroy information in tracks 01 through 06, the loss of this information does not affect the control system because the control calculation is restarted whenever a serious power-line transient occurs.

The DIGITAL INPUT selector switch is normally set to EXT (external) when the computer is controlling a process. When set to INT (internal), information from the six toggle switches (L26 through L21) is read under program control. The programming requirements for obtaining inputs from the toggle switches are given in Section V.

The toggle switches are used in conjunction with certain programming aids. The "Utility Package" referred to in Section VI is composed of several subroutines which can be selected by means of the toggle switches on the test and maintenance panel.

The ERROR light is turned on when a program or the load program attempts to write (carry out a store instruction) into an address in track 63 or in a track not currently connected through the track selector plug and jack. Program writing or loading into track 63 is impossible, regardless of which group of tracks is connected. Normal operation may be resumed either by changing the programmed storage location, or by connecting the track selector plug to the appropriate jack.

PROGRAM CHECK-OUT

The controls and indicators of the test and maintenance panel that are used during program check-out include:

a. FETCH button

b. EXECUTE button

c. RUN button

d. STATE indicators

e. Oscilloscope

Fetch and Execute Buttons

These controls are used to suspend instruction processing for the purpose of inspecting the contents of registers. Depressing the FETCH button suspends operations at the moment the next operand is selected, and the computer idles in this state. Depressing the EXECUTE button suspends operations at the moment the next instruction is selected, and the computer idles in this state. Hence, the effect of pressing either button during continuous operation is to stop program execution, and the effect when operating in the FETCH-EXECUTE mode is to advance the program by "half" steps. In either case, the appropriate light is turned on.

Run Button

Depressing the RUN button causes the computer to resume automatic high-speed execution after the computer has been in the FETCH or EXECUTE modes.

State Indicators

This portion of the test and maintenance panel may be used both in code checking and trouble-shooting. The STATE INDICATORS consists of a bank of six neon lights, a three-position rotary switch called the STATE SELECTOR and a REFERENCE toggle switch. The neon lights display the status of specific flip-flops, as determined by the setting of the STATE SELECTOR switch:

M Flip-flops M1 to M6 inclusive (track address)

PE Flip-flops P1 to P5 inclusive (operation code);
E1 (equality flip-flop)

SZ Flip-flops S1, S2, S3 (state counter); Z2 (overflow); Z4 (carry); A3 (input selection)

When the REFERENCE toggle is set to ONE, only flip-flops having a current status of one will light the neons. When set to ZERO the converse is true. Failure of a neon to light on either setting of the Reference Switch indicates that the corresponding flip-flop or neon indicator circuit is faulty.

The information displayed on the neon indicators is dependent upon the setting of the STATE SELECTOR switch and is dependent upon whether the computer is in the FETCH or EXECUTE mode. The interpretation of displays is covered in the last paragraph of this section.

The six jacks located above the neon lights are used for trouble-shooting in connection with the jack located under the oscilloscope.

Oscilloscope

A small oscilloscope is provided to aid in code checking and maintaining the computer. In code checking, its function is to display the contents of various registers when operating in the FETCH-EXECUTE mode. When register contents are displayed, the word is divided into two levels on the oscilloscope face, 10 digits to a level, as shown in figure 7-2.

A dot in a low position is a one, and a dot in a high position is a zero. The sweep is triggered so that the least-significant bit appears at the bottom right of the display. The bits in position 19 and 20 (upper left) are always zero.

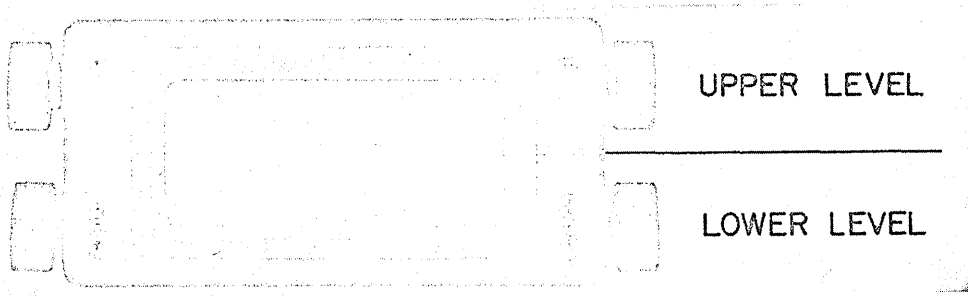


Figure 7-2. Oscilloscope Display

The octal number 123456 is displayed in figure 7-2. Interpretation is as follows:

Bit Position:	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Binary Representation:	0	0	1	0	1	0	0	1	1	1	0	0	1	0	1	1	1	0
Octal Equivalent:		1			2			3			4			5			6	

When the oscilloscope is used to observe instruction words, partitioning lines between 13 and 14 (on the oscilloscope mask) serve as an aid in recognizing the operation code or execution code normally contained in bit-positions 14 through 18. Partitioning lines between 7 and 8 aid in recognizing track address (bit-positions 8 through 13) and sector number (bit-positions 1 through 7).

Near the face of the oscilloscope are operating controls for horizontal positioning (H), vertical positioning (V), FOCUS, and brilliance control (BEAM).

Below the face of the oscilloscope are INPUT and WORD selector switches which can be used for displaying the contents of specific registers on the oscilloscope. The information displayed on the oscilloscope is

dependent upon the setting of the INPUT and WORD selector switches, and is dependent upon whether the computer is in the FETCH or EXECUTE mode. The interpretation of displays is covered in the last paragraph of this section.

Also below the face of the oscilloscope is an input JACK which can be used by maintenance personnel to feed signals from the state-indicator jacks to the oscilloscope. The CA INT. MOD. toggle switch provides the beam intensification necessary to produce the dot pattern when a register is viewed on the oscilloscope. The toggle switch is normally left in the ON position.

Tables for Interpreting Indicators

When using the FETCH and EXECUTE buttons to observe the step-by-step execution of a program, it is usually necessary to refer to the original program listing. For the purpose of correlating the indicator displays with the program listing, assume the following: The current instruction (CI column of the listing) has been acquired by the computer whenever the computer is in the FETCH mode. The tabulations which follow describe the significance of the indicator displays when the computer is in the FETCH and EXECUTE modes.

FETCH MODE

SELECTOR POSITIONS			Oscilloscope or Neon Display
Oscilloscope INPUT Selector	Oscilloscope WORD Selector	STATE SELECTOR	
A	Every		Same as previous EXECUTE (Result of previous instructions)
B	Every		Same as previous EXECUTE (Result of previous instructions)
C	Every		Execution code of current instruction in bits 1-5
N	Every		Next-instruction address in bits 1-13.
Y	Every		Current operand address in bits 1-13
R1	1st Word		Current operand
		M	In M_{1-6}, track number of current operand
		PE	In P_{1-5}, current operation code
		SZ	In Z2, overflow status of previous operations

EXECUTE MODE

SELECTOR POSITIONS			Oscilloscope or Neon Display
Oscilloscope INPUT Selector	Oscilloscope WORD Selector	STATE SELECTOR	
A	Every		(A) after completion of current instruction
B	Every		(B) after completion of current instruction
C	--		Zero
N	Every		Next-instruction address
Y	--		Zero
R1	1st Word		First word of next instruction
R1	2nd Word		Second word of next instruction
		M	In M_{1-6}, track number of next instruction
		PE	In P_{1-5}, operation code of executed instruction
		SZ	In Z2, current overflow status

SECTION VIII

NUMBER SYSTEMS AND S LING

READING COMPUTER NUMBERS

A familiarity with binary and octal number systems will simplify communication with the RW-300 for the programmer, because the computer uses the binary number system in its internal operations. The binary number system, which permits only the symbols 0 and 1, is particularly compatible with the on-off type of circuits used in digital computers.

Since each computer word consists of 17 binary digits (bits), plus sign, writing or working arithmetic with these numbers is cumbersome. Therefore, program instructions are listed as decimal numbers; numerical quantities, or constants, are entered in octal form.

If a computer instruction is

EX	OPRND	OP	NI
00	62-110	29	00-112

then the two instruction words will appear in the computer in binary as

	EX	OPERAND	
		track	sector
first word	00000	111110	1101110

	OP	NI	
		track	sector
second word	11101	000000	1110000

To use the oscilloscope (figure 7-2) or indicator lights to verify an instruction in the computer, the programmer must be able to convert binary numbers to their decimal equivalents.

A computer word may be a numerical quantity, rather than an operation or execution code and track address. These quantities must be converted from decimal to octal form for listing in the program; the octal number is stored in the computer in binary form. For example, C000325 would appear in the computer as the binary number 00000000011010101. To check this, the programmer must be able to convert binary numbers to octal.

So that a maximum number of significant digits will be carried through programmed calculations, the programmer will sometimes find it necessary to perform preliminary calculations to determine what shifting instructions should be included in the program. Although these calculations can be performed in decimal, and the results converted to octal, programming time is saved if the arithmetic operations can be performed using the octal numbers that will be used by the computer. In the paragraphs which follow, binary and octal arithmetic are described, along with numbering systems and methods of converting from one numbering system to another.

NUMBER SYSTEMS

In the decimal number 213.75, the symbols 2, 1, 3, 7, and 5 represent the sum of

$$2(10^2) + 1(10^1) + 3(10^0) + 7(10^{-1}) + 5(10^{-2}) \text{ or}$$

$$200 + 10 + 3 + 7/10 + 5/100$$

Thus, in the decimal system each position in a number has the value of some power of 10, and each digit is the coefficient of the power of ten represented by that position. The place immediately to the left of the decimal point is the power of 10^0, the next is 10^1, etc. The place immediately to the right of the decimal point is the powe of 10^{-1}, etc. Consequently, every time the decimal point within a number is moved a place to the left, the number is divided by 10; if the point is moved to the right it is multiplied by 10. Because there are 10 symbols (0 through 9) permitted in the system, it operates with powers of 10, and the base or radix of the system is 10.

The decimal number system is a "place" or positional notation system; a similar system can be devised using any base or radix. Both the binary and octal systems are parallel to the decimal system, but binary uses only 2 symbols (0 and 1) and has a base of 2, while the octal system uses 8 symbols (0 through 7) and has a base of 8. The base of a number is indicated by a subscript, as 11010101_2 (binary) or 325_8 (octal) or 213_{10} (decimal).

Each place in a binary number represents a power of two. The binary number 11010101.11 is the sum of

$$1(2^7) + 1(2^6) + 0(2^5) + 1(2^4) + 0(2^3) + 1(2^2) + 0(2^1) + 1(2^0) + 1(2^{-1}) + 1(2^{-2})$$

or 128 + 64 + 0 + 16 + 0 + 4 + 0 + 2 + 1/2 + 1/4

or 213 3/4.

Each place in an octal number is a power of 8. The octal number 325.6 is the sum of

$$3(8^2) + 2(8^1) + 5(8^0) + 6(8^{-1}) \quad \text{or}$$

$$192 + 16 + 5 + 6/8 \quad \text{or} \quad 213\ 3/4.$$

Note that the zero power of any number is always 1 and the first power of any number is the number itself. As in the decimal system, every time the point is shifted to the right, the number is multiplied by the base for every place moved. If the point is shifted to the left, the number is divided by the base for every place moved. The programmer uses this principle in scaling binary numbers.

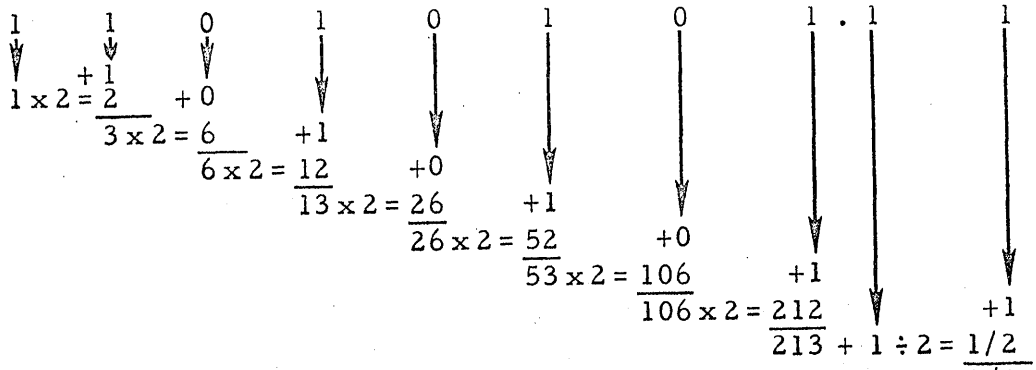
CONVERSIONS

Binary to Decimal

In the above example, the binary number 11010101.11 was converted to its decimal equivalent by adding the products of each digit times the power of 2 associated with its position. This method can be employed using the table "powers of 2" at the end of this manual. The conversion can be made more quickly by using the fact that every shift of the binary point one place to the right multiplies by two.

Convert binary numbers to decimal by multiplying the most-significant binary digit by two and adding the next binary digit. Multiply this sum by two, add the next, and continue to the least-significant digit. A similar procedure follows for digits to the right of the point except that they are divided by two instead of multiplied.

Example:



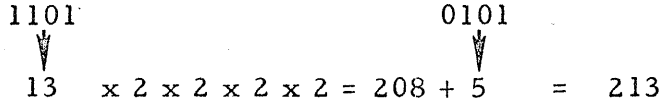
$111010101.11_2 = 213\ 3/4_{10}$

For an even faster conversion method, memorize the binary numbers from 1 through 15. They are

1 = 1	3 = 11	5 = 101	7 = 111
2 = 10	6 = 110	10 = 1010	14 = 1110
4 = 100	12 = 1100	11 = 1011	15 = 1111
8 = 1000	13 = 1101		
9 = 1001			

When converting a binary number to decimal, begin with the decimal equivalent of the first three or four digits and multiply it by two for every succeeding place to the point. Then add this number to the decimal equivalent of the rest of the binary digits.

Example: Convert 11010101 to decimal.



Decimal to Octal

The quickest way to convert a decimal number to octal is to divide the decimal number successively by 8 and note the remainders. The remainders, in reverse order, form the octal equivalent. If all or part of the number is a fraction, repeatedly multiply the fraction by 8, noting the integers resulting from each multiplication. The process is continued until the fractional product is zero, or until the desired accuracy has been obtained. The integers, in the order obtained, form the octal equivalent.

Example: Convert 213.75 to an octal number.

$213 \div 8 = 26 + 5$ $\qquad$ $.75 \times 8 = 6.00$

$26 \div 8 = 3 + 2$

$3 \div 8 = 0 + 3$

$213.75_{10} = 325.6_{8}$

Binary to Octal to Binary

The octal number system is used for notation and for performing arithmetic because it is easier to convert between binary and octal than it is to convert between binary and decimal. Since $8 = 2^3$, each octal digit is the equivalent of three binary digits. To convert an octal number to binary, write the binary equivalent of each octal digit.

Example: 3 2 5 . 6 in octal =

011 010 101 .110 or 11010101.11 in binary

To convert a binary number to octal, begin at the binary point and divide the number into triads (groups of three digits each). Then write the octal equivalent of each group.

Example: 1 1 0 1 0 1 0 1. 1 1 in binary =

3 2 5 6 in octal

Decimal to Binary

Decimal to binary conversion is like decimal to octal conversion except that the decimal number is successively divided by 2, rather than 8. The remainders from the divisions, in reverse order, form the binary number. Fractions are multiplied by 2 until a fractional product of zero is reached, or until the required accuracy has been obtained. The integers resulting from the multiplication, in the order obtained, form the binary equivalent.

Example: 213.75 in decimal = 213 ÷ 2 = 106 + 1 .75 x 2 = 1.50

106 ÷ 2 = 53 + 0 .50 x 2 = 1.00

53 ÷ 2 = 26 + 1

26 ÷ 2 = 13 + 0

13 ÷ 2 = 6 + 1

6 ÷ 2 = 3 + 0

3 ÷ 2 = 1 + 1

1 ÷ 2 = 0 + 1 = 11010101.11 in binary

Octal to Decimal

Octal to decimal conversion is most conveniently performed by adding the products of each octal digit times the power of 8 associated with its position. For example: 325.6 in octal =

$$3(8^2) = 192$$

$$2(8^1) = 16$$

$$5(8^0) = 5$$

$$6(8^{-1}) = 6/8 \text{ or } 3/4$$

$$= 213\ 3/4 \text{ in decimal}$$

Powers of 8 are listed in a table at the end of this manual.

BINARY ARITHMETIC

The rules for binary arithmetic are:

Addition	Subtraction	Multiplication	Division
0 + 0 = 0	0 - 0 = 0	0 x 0 = 0	0 ÷ 0 = undefined
0 + 1 = 1	1 - 1 = 0	0 x 1 = 0	0 ÷ 1 = 0
1 + 1 = 0 (with 1 to carry)	1 - 0 = 1	1 x 1 = 1	1 ÷ 1 = 1
	0 - 1 = 1 (with 1 borrowed)		1 ÷ 0 = undefined

Examples:

Addition		Subtraction	
augend:	1 1 0 0 1	borrow:	0 1 1 10
addend:	1 1 0 1	minuend:	1 0 0 0
carry:	11 1	subtrahend:	1
sum:	10 0 1 1 0	difference:	1 1 1

Examples:

Multiplication

```
Multiplicand:      1 0 0 1 1
Multiplier:          1 1 0 1
                   1 0 0 1 1
                 0 0 0 0 0
               1 0 0 1 1
           1 0 0 1 1
Carry:       1 1
Product: 1 1 1 1 0 1 1 1
```

Division

```
        1.1
101 ) 111.1
      101
       10 1
       10 1
```

OCTAL ARITHMETIC

Octal arithmetic is like decimal arithmetic. To perform the arithmetic operations quickly, octal addition and multiplication tables can be used, or memorized. An alternative to this is to perform the operations mentally in decimal and convert the sum or product to octal before writing it. Below are tables showing octal multiplication and addition.

OCTAL ADDITION TABLE

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	10
2	2	3	4	5	6	7	10	11
3	3	4	5	6	7	10	11	12
4	4	5	6	7	10	11	12	13
5	5	6	7	10	11	12	13	14
6	6	7	10	11	12	13	14	15
7	7	10	11	12	13	14	15	16

OCTAL MULTIPLICATION TABLE

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	10	12	14	16
3	0	3	6	11	14	17	22	25
4	0	4	10	14	20	24	30	34
5	0	5	12	17	24	31	36	43
6	0	6	14	22	30	36	44	52
7	0	7	16	25	34	43	52	61

In decimal addition, 7 + 7 = 14. The octal equivalent of 14 is 16 ($1 \times 8^1 + 6 \times 8^0$); thus, in the octal addition table above 7 + 7 = 16. In working an addition problem, the 6 is written and the 1 is carried to the next column.

GLOSSARY

Specific examples used in these definitions usually refer to the RW-300 only.

Absolute Value - the magnitude of a number without regard to the algebraic sign of the number.

Access Time - the time interval between the instant at which information is: (a) called for from storage and the instant at which delivery is completed, i.e., the read time; or (b) ready for storage and the instant at which storage is completed, i.e., the write time.

Accumulator (A Register) - the register in the arithmetic unit in which sums and other arithmetic and logical results are formed.

Adder - a device capable of forming the sum of two quantities plus a carry digit from a previous addition.

Address - a label (usually a set of numbers) which identifies a register or location in which information is stored.

Analog - representing numerical quantities by means of continuous, physical variables, e.g., translation, rotation, voltage, resistance; contrasted with "digital".

Analog Conversion - the operation of changing analog information to its digital (numerical) equivalent, or vice-versa.

Analog Input - the acceptance of analog voltages from transducers and the conversion of this data to equivalent digital form for processing by the computer.

Analog Output - the conversion of digital information generated by the computer into equivalent voltages or currents to operate controls or indicators.

Arithmetic Unit - that portion of an automatic digital computer in which arithmetic and logical operations are performed.

Base (Radix) - the fundamental number of a system of numbers. Thus, 10 is the base of the decimal number system, 2 of the binary, 8 of the octal.

Binary - involving the integer 2, as in a binary number system (base 2), a binary choice (between two alternatives), or a binary operation (combining two quantities).

Binary Number - a numerical value expressed in binary notation.

Bi-Polar - in the RW-300, having to do with both negative and positive analog input voltages; the bi-polar analog-digital converter.

Bit - a binary digit.

Bit Time - the length of time required for one bit to pass a given point on the magnetic drum (approximately 6.5 microseconds in the RW-300).

Block - a group of information recorded on magnetic tape corresponding to one track of information recorded on the magnetic drum.

Branch - see Transfer.

Break-Point - a point in a routine at which the computer samples a manually set switch to determine the subsequent course of the program.

Carry - (1) the digit to be added to the next higher column when the sum of the digits in one column equals or exceeds the number base; (2) the process of forwarding the carry digit.

Check - a means of verifying information during or after an operation.

Marginal Checking - a system or method of determining computer circuit weaknesses by varying the operating conditions of the circuits.

Circulating Register - see Register.

Code (noun) - a system of symbols and rules for use in computer operations.

Execution Code - a binary code used to modify certain operations in the RW-300 such as shifts, multiplication, division, etc. More specifically, the five bits contained in bit-positions 14 through 18 of the first word of a two-word instruction.

Instruction Code - the symbols, names, and definitions of instructions which are directly intelligible to a given computer.

Mnemonic Code - a code, usually alphabetic, chosen so that it can be remembered easily. Example: MG for Merge, LA for Load A, etc.

Code (verb) - to prepare problems in computer code for a specific computer.

Command - often used as a synonym for Instruction or Operation. See Instruction Code.

Computer - any device capable of accepting information, performing sequences of arithmetic and logical operations, and supplying the results of these operations.

Control Unit - that portion of an automatic digital computer which directs the sequence of operations, interprets coded instructions, and initiates the proper commands to the computer circuits to execute the instructions.

Convert - (1) change numerical information from one number base to another (e.g., decimal to binary) and/or from some form of fixed-point to some form of floating-point representation, or vice-versa; (2) change analog information (e.g., distances, rotations, voltages, etc.) into digital information (numerical) or vice versa.

Core - a toroid of ferromagnetic material capable of being magnetized in either of two directions, and therefore a binary device which can store one bit for indefinite periods.

Core Storage - an array or matrix of cores capable of storing large numbers of bits.

Core Buffer - a core storage and associated equipment to permit communication between the RW-300 and the magnetic tape transports.

Data - any information (usually numbers) taken in, operated on, or obtained from a computer.

Raw Data - unappraised information entered into the computer.

Data Word - a word containing or reserved for numerical information, as opposed to an instruction word.

Digit - one of the n symbols of integral values ranging from 0 to n - 1 inclusive in a scale of numbering of base n, especially one of the ten decimal digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Digit Time - See Bit Time.

Digital - using discrete numbers in a given scale of notation to represent all the quantities that occur in a problem or a calculation, as opposed to analog.

Digital Input - the transfer of information from a digital source (paper-tape reader, digital switches, digital clock, etc.) to the computer.

Digital Output - the transfer of information in digital form from the computer to some external device such as a typewriter, paper-tape punch, relays, etc.

Double-Length Product - see Product.

Drum - cylinder with coating of ferromagnetic material used as a storage device in digital computers; elementary lengths of a drum track are magnetized to represent ones or zeros.

Execution Code - see Code.

Extract - an operation whereby some portion of the A register is retained while the remaining portions are cleared by a logical multiplication of the contents of a specified storage location.

Fixed Point - see Point.

Flip-Flop - an electronic circuit having two stable states. A flip-flop can store one binary digit of information.

Flow Chart (Flow Diagram) - a graphical representation of a sequence of operations which is usually drawn up before a program is written to show how the program is to operate.

Frame - on a paper tape, one row of 6 or 8 binary digits.

General Storage - see Storage.

Indicator - (1) on a panel, a light, oscilloscope, flag, or other device which displays information visually. (2) in the format requirements of a punched paper tape, a symbol which tells the RW-300 load program what kind of information is to follow.

Input - information accepted by the computer from cards, punched tape, magnetic tape, instruments, etc.

Input Group - digital input lines, normally 18, addressed by a specific track number.

Input Line - a wire capable of assuming two discrete voltage levels for representation of either a one or a zero.

Instruction - see Code.

Leader - that portion of a punched paper tape that precedes the input data. It is usually left blank.

Load (verb) - (1) to enter data into the computer, as "to load a tape". (2) to cause data to enter a register from a memory location, as "Load A".

Load Program - a permanently stored program on track 63 which controls the loading of programs and data into the memory.

Logging Typewriter - an output device for keeping a running log of process conditions.

Logical Operation - see Operation.

Logical Addition - by definition, $0 + 0 = 0$, $1 + 0 = 1$, $0 + 1 = 1$, $1 + 1 = 1$. Logical addition is performed bit by bit. Thus, if corresponding bit positions of two registers are both zero, the result is zero in that position. Otherwise, the result is a one in that position.

Logical Product - by definition, $0 \cdot 0 = 0$, $0 \cdot 1 = 0$, $1 \cdot 0 = 0$, $1 \cdot 1 = 1$. The logical product is formed bit by bit and has ones only in those bit positions which had ones in the corresponding positions of both registers.

Loop - a portion of a program that is repeated until some predetermined condition has been satisfied.

Magnetic Core - see Core.

Magnetic Tape - flexible tape coated with ferromagnetic material. In digital computer applications, elementary lengths of the tape are magnetized to represent ones or zeros.

Magnetic Tape Handler (Transport) - device for moving tape so that information can be magnetically recorded or read.

Magnetic Tape Buffer - see Core Buffer.

Memory - any device into which units of information can be stored and from which the information can be obtained at a later time; in the RW-300, the magnetic drum is the internal memory.

Memory Sum - a word consisting of the algebraic sum (modulo 2^{17}) of all binary digits in a specified portion of a paper tape. The memory sum is used to determine whether or not all the information on a tape has been stored in the computer correctly.

Merge - an operation whereby any portion of the contents of a specified storage location may be inserted into the corresponding portion of the A register by logical addition.

Microsecond - a millionth of a second.

Millisecond - a thousandth of a second.

Module - in the RW-300, individual circuits are assembled and wired on insert cards. Interconnections between circuits and physical mounting for cards are provided by modules.

Modulo - an operator which denotes division by a base number; e.g., if a = b modulo p, a is the remainder obtained after b is divided by the base number p. 132 modulo 128 is 4.

Multi-Bit Output - a digital output circuit arrangement which causes a group of output lines to be set to a one-and-zero pattern corresponding to the contents of the A register when the Digital output command is executed.

Octal - involving the integer 8, as in the octal number system (base 8).

One-Bit Output - a digital output circuit arrangement which permits one or more output lines to be controlled without disturbing other lines in the group.

Operand - a number used in an operation.

Operation - (1) a defined action; (2) the action specified by a single complete instruction.

Arithmetic Operations - operations in which numerical quantities form the elements of the calculation (e.g., addition, subtraction, multiplication, division).

Logical Operations - the operations of comparing, selecting, matching, sorting, merging, etc.

Optimum Programming - see Program.

OPUS - Optimum Programming Using Symbols. An assembly routine for the RW-300 which reads a symbolically coded program and assigns optimum memory locations and numerical operation and execution codes.

Origin - RW-300 memory location from which the first program instruction is read when the computer's START button is pressed, location 00-00

Output - information sent from the computer to typewriters, punches, magnetic tape, indicators, controllers, etc.

Output Group - digital output lines, normally 18, addressed by a specific track number.

Output Line - a wire, or pairs of wires capable of assuming two discrete voltage levels for representing either a one or a zero.

Parallel - handled simultaneously, as opposed to serial.

Parity - in the RW-300, the condition of a binary code in which the total number of ones is always odd.

Parity Check - a test for data validity by examining the binary code to determine whether or not the total number of ones is odd.

Peripheral Equipment - accessory and auxiliary equipment used with a computer to form a complete system.

Point - in positional notation, the location or symbol separating the integral part of a number from its fractional part. In decimal notation the point is called the decimal point. In binary notation it is called the binary point.

Fixed-Point Representation - a notation or system of arithmetic in which all numerical quantities are expressed by a predetermined number of digits with the point implicitly located at some predetermined position.

Product, Double-Length - the result of a multiplication in which twice as many digits are retained as the computer normally holds in one register; e.g., a computer whose basic word consists of 17 binary digits will have, as the result of a multiplication, a 34-digit product.

Program (noun) - a list of instructions for the solution of a problem. See Routine.

Program (verb) - to plan a computation or process from the initial problem to the delivery of the results, including the integration of the operation into an existing system. Thus, programming consists of analyzing the problem, drawing a flow chart, and coding the problem. Also, it may include numerical analysis,

Program (verb) continued - systems analysis, specification of print formats and any other functions necessary to the use of a computer in a system.

Optimum Programming - arrangement of data and instructions in such a way that minimum waiting time is required to obtain information from the memory.

Program-Writable - in the basic RW-300, those areas of memory that can be written into under program control without moving the track group selector plug from the jack marked 0-7.

Radix - see Base.

Read - to copy, usually from memory, or from one form of memory to another, particularly from external or secondary storage (paper tape or magnetic tape) to internal storage.

Read Head - an electronic device which is capable of sensing and transmitting information recorded on a magnetic drum or on magnetic tape.

Register - device for storing one or more computer words, or parts thereof in the arithmetic and control units. In the RW-300 the A register and the B register are circulating arithmetic registers. The C, Y, and N registers are circulating control registers.

Register (continued) - The T, S, P, and M registers are non-circulating control registers.

Circulating Register - a register whose contents are continually read and re-written on the magnetic drum surface.

Revolver - a register providing 32 words of fast-access storage; track 62 of the RW-300.

Routine - a set of coded instructions arranged in proper sequence to direct the computer to perform a desired operation or series of operations. See Program.

Scale - to change the units in which a variable is expressed (e.g., moving the decimal point, or its binary equivalent) so as to bring it within the capacity of the machine or routine at hand.

Scale Factor - a magnitude indicating the number of places the true point is to the left or right of an arbitrary fixed position in a data word.

Sector - 1/128 of a track, providing storage for one computer word.

Sector Number - an integer ranging from 00 through 127, any one of which denotes a particular word on a track.

Serial - handled on by one in time, as opposed to parallel.

Shift - to move the contents of a register to the right or left.

Sign Bit (Sign Digit) - a one or a zero used to designate the algebraic sign of a quantity; a zero represents a plus and a one represents a minus in the RW-300.

State - a name applied to each of several timing and control sequences the computer must enter while performing instructions. There are eight possible states in the RW-300.

Storage - information storage facilities that are controlled by the computer. Also see Memory.

General Storage - RW-300 drum tracks 00 through 61.

Store - to transfer information to a storage location from which the information can be obtained at a later time.

Track - in the RW-300, a band around the magnetic drum, capable of storing 128 words.

Track Address - number designation specifying the track number of a storage location.

Trailer - blank tape behind the last punched frame on paper tape. See leader.

Transfer - (1) to move data from one location to another; to copy, exchange, read, record, store, transmit, or write data; (2) to transfer control; to jump from one part of a program to another.

Utility Package - collection of routines which aid the programmer in loading and checking programs.

Unipolar - in the RW-300, having to do with positive analog input voltages.

Word - a set of characters which occupies one storage location. In the RW-300, a word consists of 18 bits. The control unit treats two words or 36 bits as an instruction and the arithmetic unit treats one word, or 17 bits plus sign, as a quantity.

Word Time - the length of time required for a sector on the drum to pass a given point. Approximately 0.13 milliseconds in the RW-300.

Write - to record information on any internal or external storage medium.

Write Head - an electronic device which records information on a magnetic drum or on magnetic tape.

INDEX

TABLE OF POWERS OF 2

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25

Table of Non-Parity Flexowriter Codes

CHARACTER		OCTAL CODE	
UPPER CASE	LOWER CASE	IN COMPUTER	ON TAPE
A	A	61	141
B	B	62	142
C	C	63	143
D	D	64	144
E	E	65	145
F	F	66	146
G	G	67	147
H	H	70	150
I	I	71	151
J	J	41	101
K	K	42	102
L	L	43	103
M	M	44	104
N	N	45	105
O	O	46	106
P	P	47	107
Q	Q	50	110
R	R	51	111
S	S	22	42
T	T	23	43
U	U	24	44
V	V	25	45
W	W	26	46
X	X	27	47
Y	Y	30	50
Z	Z	31	51
)	0	20	40
'	1	1	1
"	2	2	2
#	3	3	3
$	4	4	4
%	5	5	5
¢	6	6	6
&	7	7	7
*	8	10	10
(	9	11	11
°	10	52	112
?	11	53	113
:	12	34	54
—	,	33	53
.	.	73	153
/	-	21	41
=	+	40	100
	LOWER CASE	72	152
	UPPER CASE	74	154
	TAB	36	56
	SPACE	00	00
	BLACK	12	12
	RED	54	114
	C/RETURN	56	116
	PUNCH ON	250	-
	PUNCH OFF	244	-
	NON PRINT	230	-
	PRINT RESTORE	224	-
	STOP CODE	13	13
	TAPE FEED	77	157

Table of Powers of 8

8^n	n	8^{-n}
1	0	1.0
8	1	0.125
64	2	0.015 625
512	3	0.001 953 125
4 096	4	0.000 244 140 625
32 768	5	0.000 030 517 578 125

Table of Equivalent Revolver Locations

R0	0	32	64	96	R16	16	48	80	112
R1	1	33	65	97	R17	17	49	81	113
R2	2	34	66	98	R18	18	50	82	114
R3	3	35	67	99	R19	19	51	83	115
R4	4	36	68	100	R20	20	52	84	116
R5	5	37	69	101	R21	21	53	85	117
R6	6	38	70	102	R22	22	54	86	118
R7	7	39	71	103	R23	23	55	87	119
R8	8	40	72	104	R24	24	56	88	120
R9	9	41	73	105	R25	25	57	89	121
R10	10	42	74	106	R26	26	58	90	122
R11	11	43	75	107	R27	27	59	91	123
R12	12	44	76	108	R28	28	60	92	124
R13	13	45	77	109	R29	29	61	93	125
R14	14	46	78	110	R30	30	62	94	126
R15	15	47	79	111	R31	31	63	95	127

TABLE OF RW-300 INSTRUCTIONS

Description of Operation	Operation Code		Instruction Time		Standard Inst. Format	
			CI → Oprnd	Total Time CI → NI	1st word	2nd word
	Alpha	Dec.	(word times)	(word times)	EX OPA	OP NIA
ADD: (A)+(M) → A	A	25	3	6 or 7	i 00 TT-SS	i 25 TT-SS
SUBTRACT: (A) − (M) → A	S	24	3	6 or 7	i 00 TT-SS	i 24 TT-SS
MULTIPLY: (A) × (M) → (A,B)	M	16	3	6 + nn	i nn TT-SS	i 16 TT-SS
DIVIDE (A) / (M) → (A), remainder B	D	26	3	6 + nn	i nn TT-SS	i 26 TT-SS
LOAD A: (M) → A	LA	29	3	5	i 00 TT-SS	i 29 TT-SS
LOAD B: (M) → B	LB	07	3	5	i 00 TT-SS	i 07 TT-SS
LOAD A NEGATIVE: − (M) → A	LN	21	3	5	i 00 TT-SS	i 21 TT-SS
STORE A: (A) → (M)	SA	30	4	6	i 00 TT-SS	i 30 TT-SS
STORE B: (B) → (M)	SB	20	4	6	i 00 TT-SS	i 20 TT-SS
TRANSFER NEG: If (A) <O, Oprnd Add. → NI	TN	09	-	4 or 5	i 00 TT-SS	i 09 TT-SS
TRANSFER OVERFLOW: If O.F., Oprnd Add. → NI	TF	10	-	4 or 5	i 00 TT-SS	i 10 TT-SS
TRANSFER ZERO: If (A) = O, Oprnd Add. → NI	TZ	11	-	4 or 5	i 00 TT-SS	i 11 TT-SS
COMPARE MAGNITUDE: If |(A)| − |(M)| < O; Add nn to NI Sector Address	CM	15	3	5 or 7	i nn TT-SS	i 15 TT-SS
EXTRACT: (A) ⊗ (M) → A	EX	05	3	5	i 00 TT-SS	i 05 TT-SS
MERGE: (A) ⊕ (M) → A	MG	31	3	5	i 00 TT-SS	i 31 TT-SS
STOP: Stop; CI → NI on RESUME	SP	00	-	4	i 00 00-00	i 00 TT-SS
NO OPERATION: CI → NI	NO	03	-	4	i 00 TT-SS	i 03 TT-SS
SHIFT: (A) → , (A) ← , (A,B) ←	SH	01				
Operand Track Address — Variations						
00-15 (A) right nn places			-	4 + nn	i nn 00-00	i 01 TT-SS
16-31 (A) left nn places			-	4 + nn	i nn 16-00	i 01 TT-SS
48-63 (A, B) left nn places			-	4 + nn	i nn 48-00	i 01 TT-SS
SWITCH:	SW	02				
Operand Track Address — Variations						
00-15 (A) → (B)			-	5	i 00 00-00	i 02 TT-SS
16-31 (B) → (A)			-	5	i 00 16-00	i 02 TT-SS
32-47 (A) ↔ (B)			-	5	i 00 32-00	i 02 TT-SS
48-63 O → (A,B)			-	5	i 00 48-00	i 02 TT-SS
DIGITAL	DG	06				
Operand Track Address — Variations						
00 Output to Flex			-	5	i 00 00-00	i 06 TT-SS
04-31 1 bit Outputs			-	5	i 00 04-00	i 06 TT-SS
32 Input from Flex			-	6	i 07 32-00	i 06 TT-SS
36-63 1 bit Inputs			-	6	i 18 36-00	i 06 TT-SS
MAGNETIC TAPE ADDRESS *	TA *	22				
Operand Address — Variations						
TT-10 Read Computer: (TK TT) → Buffer			-	134 min. 262 max.	i tu TT-10	i 22 TT-SS
TT-04 Write Computer: (Buffer) → TK t; t = O selects TK 14, t = 1 selects TK 15			-	5	i tu TT-04	i 22 TT-SS
TT-07 Read Tape: (1 tape Block) → Buffer			-	5	i tu TT-07	i 22 TT-SS
TT-05 Write Tape: (Buffer) → (1 tape Block)			-	5	i tu TT-05	i 22 TT-SS
TT-01 Rewind: Rewind tape unit u to start			-	5	i tu TT-01	i 22 TT-SS
TT-06 Backspace: Backspace tape unit u 1 block			-	5	i tu TT-06	i 22 TT-SS
TT-03 Search Forward: Search forward for key block			-	5	i tu TT-03	i 22 TT-SS
TT-02 Search Reverse: Search reverse for key block			-	5	i tu TT-02	i 22 TT-SS
* NOTE: Mag tape instruction times assume tape unit is ready						

LEGEND:

CI Current Instruction	NI Next Instruction	Add. Address		Oprnd. Operand	
OP Operation	EX Execution Code	A A-register		( ) Contents of	
Rn Revolver location n	u Magnetic Tape Unit u	B B-register		OPA Operand Address	
				NIA Next Inst. Address	

10.4 States 1, 2, 3, and 4 — All Instructions

STATE	INSTR.	DIGIT TIMES	DESCRIPTION
1	All	1 - 7	Search for equality between Nr and Sector No. If N ≠ Sect. No. $E_1 = 0$. If N = Sect. No. $E_1 = 1$
		8 - 13	Shift N(TRK) ⟶ M register M_1-M_6. N_{13} goes to M_1 and N_8 to M_6
		19	If $E_1 = 0$ Stay in State 1 If $E_1 = 1$ Go to State 2
		1 - 18	A, and B registers recirculating (20 bits) Y, C do not recirculate.
		19	Set $E_1 = 1$, Set $Z_4 = 1$
2	All	1 - 13	Put operand address (GSR) into Y register
		14 - 18	Shift EX. Time ⟶ P flip-flops (P_1 - P_5) to (E_{18} - E_{14})
		1 - 18	A and B registers recirculating (20 bits) N and C do not recirculate
		19	Go to St. 3
3	All	1 - 13	GSR ⟶ N_w (Next Instr.)
		1 - 5	Shift P Register ⟶ C_w (Ex. Time ⟶ C)
		14 - 18	GSR (Instr. Code) ⟶ P Reg. (P_1 - P_5)
		1 - 18	A and B 20 bits recirculate
		1 - 18	Y Recirculate
		6 - 18	C Recirculate
	All	19	Go to St 4 if FT = 1
	1 - 10 14 - 15	19	Go to St. 4
	11 - 13 17 - 19	19	Go to St. 5, if FT = 0
		9,10 19	0 ⟶ E_1
4	1 - 8,15,16	1 - 7	Search for equality between Y and Sect. No. (R66)
	9,10	1 - 7	Search for equality between Y and Store Sect. No. (J9)
	1 - 8,15,16	19	If $E_1 = 0$ Stay in St. 4 If $E_1 = 1$ Go to St. 5
	9,10	19	If E_1 FT' (Record Error)' = 1 Go to St. 5 If RE = 1 Go to St. 8
	11 - 13 17 - 19	19	Go to St. 5
	All	1 - 18	A,B,Y,C, and N Recirculate.
	14 (DG)	d17	0 ⟶ E_1 if all delays expired