Division 6 - Lincoln Laboratory
Massachusetts Institute of Technology
Lexington 73, Massachusetts

SUBJECT: MEMORY TEST COMPUTER PROGRAMMING REFERENCE MANUAL

To: N. H. Taylor and MTC Section

From: Philip R. Bagley

Date: 9 May 1955

Approved: W. A. Hosier
W. A. Hosier

Abstract: The Memory Test Computer is a high-speed electronic digital machine with over 28,000 registers of magnetic core and drum memory. It operates in the binary number system with a word length of 16 binary digits for both numbers and instructions, and it can perform 26 different kinds of instructions. For programs confined to core memory the average speed of operation is 75,000 instructions per second, exclusive of input-output instructions. The terminal equipment consists of paper tape readers and punch, IBM card reader and punch, printer, display oscilloscopes, camera, digital data transmitting and receiving equipment, and an output buffer system associated with the magnetic drum memory.

In addition to a brief description of the computer, the principles of coding, the machine functions, instruction code, and terminal equipment codes are described in sufficient detail for the purposes of programming. The input process, carried out through the medium of the Input and Basic Conversion Programs, is fully described. A brief operating guide for the computer is given.

The research reported in this document was supported jointly by the Department of the Army, the Department of the Navy, and the Department of the Air Force under Air Force Contract No. AF 19(122)-458.

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

FOREWORD

The purpose of this memorandum is to provide in convenient form all the information necessary to the preparation of programs for the Memory Test Computer (MTC). This revision, 6M-2527-2, supersedes the following memoranda:

M-1881, Memory Test Computer: Guide to Coding and MTC Instruction Code

M-2527, Memory Test Computer: Programming Manual

M-2527, Supplement #1, Input and Basic Conversion Programs for the Memory Test Computer

M-2527, Supplement #2, Corrigenda for M-2527

M-2527, Supplement #3, Second Set of Corrigenda for M-2527

M-2527-1, Memory Test Computer: Programming Manual

M-2527-1, Correction #1, Memory Test Computer: Programming Manual Correction

6M-2527-1, Correction #2, Memory Test Computer: Programming Manual Correction

Changes in the computer which affect programming methods will be reported in periodic supplements to this memorandum. It will be greatly appreciated if errors or obscure passages in this memo are brought to the attention of the writer or of the MTC Section Leader so that appropriate changes can be incorporated in future supplements and revisions.

For those readers who are already familiar with the preceding revision of this memo (M-2527-1) and who would like to know the essential differences between it and this revision, a summary of changes and additions covered by this revision is given below:

1. A unit containing an IBM card reader and punch has been incorporated in MTC. New operation codes op 1000+n and op 3000+n (octal) control the card reader and card punch respectively. Conventions corresponding to those for punched tapes have been established for punched cards, but no punched-card conversion program has yet been written.

2. Live Registers #3 and #4, heretofore not accessible to the programmer, now serve as input and output links to electronic equipment external to the computer, currently a Digital Data Transmitter and Receiver, and a Data Link Converter (AN/GKA-1). The instruction st LR 4 performs the special function of reading a word from the external equipment to Live Register #4.

3. On read in, the Input Program now accumulates the check sum in Live Register #5 at address 0-50 (octal) instead of in core memory register 1-3777 (octal). This change necessitated modification of the Input Program and the Basic Conversion Program, and of the structure of binary tapes. To distinguish the new system from the old, the Input and Basic Conversion Programs and the 4-6-6 binary tape structure described in the previous revision (#1) of this memo will be designated "A" and the new counterparts will be "B". For the convenience of programmers, a utility program is available which will convert existing Type A binary tapes to Type B. This conversion is not a necessity, however, since Input Program A and Basic Conversion Program A will be retained.

4. In addition to the change mentioned above, Input Program B has provisions to avoid the necessity of suppressing the overflow alarm when reading in.

5. Four fields of the magnetic drum memory have been wired as display buffers which simulate one TD field and one RD field, or two DD fields, for the purpose of testing the XD-1 display generation frames.

6. A machine idiosyncrasy has manifested itself in connection with overflow. Under certain circumstances an overflow will stop the computer in an alarm condition, but the overflow alarm light will not be on. A more complete explanation is given in Section III in the paragraphs discussing overflow.

7. The execution times of most instructions are different from those published in revision 1.

8. The character selection codes for the Charactron and Typotron have been entirely changed from those previously published. Additional conventions have been established for controlling limited category and feature selections for the Charactron display.

9. The programming manual text has been expanded to include a discussion of scale factors employed in programming.

10. The ri instruction has been slightly modified, so that it is no longer necessary to clear the B-Register prior to executing an ri instruction.

The reader is presumed to be familiar with the binary and octal number systems. If he is not, he should read MIT Digital Computer Laboratory Report R-90-1, The Binary System of Numbers.

Several other laboratory publications will be of interest to MTC programmers:

M-2787, MTC U-68 Utility and Analysis Program

6M-3497, MTC Subroutine Library

6M-3509, MTC Technicians Training Manual, Chapter III

It is expected that frequent reference will be made to the pages in this memo which describe the instructions in detail. To facilitate the location of these pages, a black marker has been placed on the right edge of each of these pages. The markers will be readily visible when the binding edge of the memo is held in the left hand, and the pages are fanned back with the right thumb on the right-hand edge of the front page.

The writer owes debts of thanks to Professor C. W. Adams, upon whose work Section II of this memo is largely based, to F. R. Durgin for his contribution on the IBM card reader and punch, and to W. A. Hosier, who has patiently edited this memo and has contributed many valuable suggestions.

SECTION I. BRIEF DESCRIPTION OF THE MEMORY TEST COMPUTER

Purpose of the Memory Test Computer. The original purpose of the construction of the Memory Test Computer was, as its name implies, to provide realistic tests of the practicability of the newly-developed coincident-current magnetic core memory. After several months of extensive memory tests on the computer, the original magnetic-memory system was transferred to WWI, abruptly terminating this testing program. There existed at this time, however, a growing need for a "proving ground" for devices and techniques proposed for use in future computers, as well as for additional computing facilities. The activity of the computer has been redirected toward filling these needs.

Characteristics of MTC. MTC has the general characteristics of WWI, but speed has been sacrificed occasionally where it seemed to demand excessive complexity or power. It is a parallel, digital machine, operating in the binary number system, having a word length of 16 bits, (for both instruction and numbers), and using one-tenth microsecond pulses for information transfer. The machine has the facilities to perform 26 different instructions, and it can execute instructions at an average rate of 75,000 per second. MTC has three kinds of internal memory: "test memory" or "panel memory" (64 registers), magnetic-core memory (4096 registers), and magnetic drum memory (24,576 registers). The terminal equipment consists of punched paper tape readers and punch, an IBM card reader and punch, printer, several display oscilloscopes (one scope fitted with a camera under computer control), Digital Data Transmitter and Receiver, an AN/GKA-1 Data Link Converter, and an output buffer system associated with the magnetic drum memory.

Physical layout. The entire computer except for power supplies is housed in a room approximately 27 x 50 feet. An air-conditioning system supplies filtered air with controlled temperature and humidity. Most of the computer is in racks of special design so that the digit positions of a register are in a vertical line. All the units serving the nth digit position are then in a horizontal line across the whole computer frame. The control functions are generated by standard pulse-control equipment mounted in 19-inch racks behind the main computer frame. The operating console is a long desk in the center of the room, housing the start and stop controls, panel memory, marginal checking controls, flip-flop register indicator lights, alarm controls and lights, and a monitor scope. Adjacent to the console are the paper tape readers and punch, the card reader and punch, the printer, and a large display scope. The magnetic memory array is mounted in an open frame with the general dimensions of a "shower stall."

Bus system and information transfer. Nearly all transfers of words between registers are made in two steps via a buffer register designated the "A-Register." First, a word is read from the originating register via the A-Register In Bus to the A-Register. Second, the word is read from the A-Register via the A-Register Out Bus to the terminating register. While this technique requires two steps instead of one to effect a computer transfer, it avoids some electronic problems present in other bus systems.

Simplified block diagram of MTC system. The block diagram on the following page shows all the units in the computer system and their interconnections for transferring numerical information. All of the command circuits have been omitted for the sake of clarity. The number of wires in each connection is indicated by a number N, thus:

Memory element. The internal memory is composed of three types of memory units: Panel Memory, composed of toggle switches, an IBM plugboard, and five flip-flop storage registers; Magnetic Core Memory; and Magnetic Drum Memory.

Panel Memory contains 64 numbered registers of 16 bits each, of which 32 are toggle switch registers and 32 are connections to an IBM plugboard. Five flip-flop registers may be substituted, with certain restrictions, for one or more of these 64 registers. The principal functions of Panel Memory are: testing various parts of the computer by "programming" its own internal commands, insertion of limited amounts of data, manual intervention in computer operations, and serving as input and output connections to various units of terminal equipment.

Magnetic Core Memory consists of two "fields," each of 2048 registers of "high-speed random-access" memory. Each register holds a 16-bit word plus a 17th bit for checking purposes.

Magnetic Drum Memory consists of twelve fields, each of 2048 registers of "medium-speed sequential-access" memory. Each drum register holds a 16-bit word plus a 17th bit for checking purposes.

Control element. The control sections of MTC may be roughly subdivided as follows: Central Control (which includes Group and Field Control, Core Memory Control, and Drum Control), Alarm Control, and In-Out Control (which includes Camera Control, Scope Control, Tape Reader Control, and Printer and Punch Control). Central Control is constructed principally of standard test equipment units, interconnected by video cables to provide the necessary control functions. Altering the logic or timing of Central Control usually involves nothing more than recabling and perhaps exchanging of units. Central Control is a delay-line type of control in which pulses are routed by gate tubes to selected chains of delay lines, which in turn distribute command pulses in appropriate sequences. Timing is "asynchronous" in that no master oscillator is used; instead, the last of a sequence of command pulses is used to start the next sequence.

Arithmetic element. The three registers of the arithmetic element are the Partial Sum Register, the Carry Register, and the B-Register. The Partial Sum Register plus the Carry Register constitute the Accumulator, but the term "Accumulator" is often colloquially applied

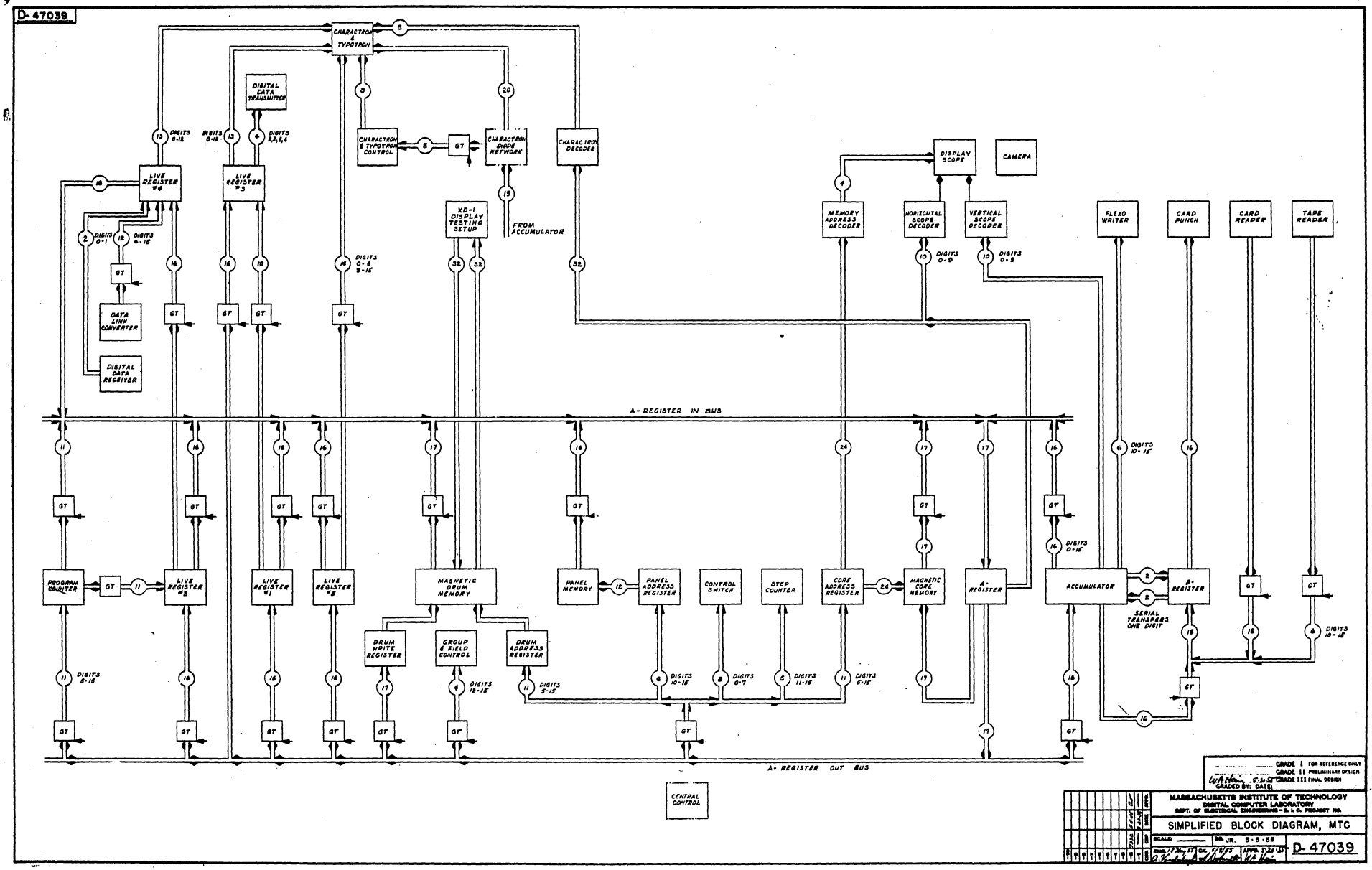
to the Partial Sum Register alone. The fundamental arithmetic operations which can be performed in the Partial Sum Register are simply addition and shifting right. Subtraction is accomplished by adding the subtrahend to the complement of the minuend, and recomplementing the result. Multiplication is by successive addition. The sole function of the Carry Register, as its name denotes, is to store momentarily the carries generated in addition. The B-Register is in effect a 16-digit extension of the Partial Sum Register. It can only perform shifting right. It holds the multiplier during multiplication, and holds the right-hand 15 bits of the product at the end of the multiplication. In addition to their arithmetic functions, the Partial Sum and B-Registers serve as connections to various input-output units.

Console. Most of the operating controls are available to an operator seated at the console desk. Switches are provided for turning the power on and off, starting and stopping the computer, and for marginal checking. Computer operations can be monitored visually by means of an oscilloscope and neon indicator lights and aurally by means of a loudspeaker. Thirty-two toggle-switch storage registers are available for the manual insertion of instructions and data. At the back of the console desk is a plugboard cabinet, into which prewired panels containing instructions and data may be plugged.

Power supply. The power supplies which provide filtered, regulated, and monitored power, are almost entirely under automatic control. The various supplies are sequenced on or off in order to protect circuits and to lengthen the life of circuit components. In the event of circuit failures or other emergencies, power to the machine is disconnected, and suitable alarms are given.

Terminal equipment. The regular input devices are two punched paper tape readers and a punched card reader. The regular output devices are: a printer, which can print any format achieved by the keys of a typewriter; a paper tape punch, for punching tape which is later read by the tape reader; a card punch; and various display oscilloscopes, one of which is equipped with a camera operated under control of the computer. The oscilloscopes may be used to generate any pattern, including alphanumerical characters and graphs.

In the category of special terminal equipment, which has a very limited application, are: a Digital Data Transmitter for transmitting digital information serially over a telephone circuit, and a Digital Data Receiver for receiving this information and decoding it into its component signals; an AN/GKA-1 Data Link Converter for decoding digital messages transmitted via a radio channel; and an output buffer arrangement associated with magnetic drum memory, providing a means by which MTC can simulate the display drums of the XD-1 computer.


D-47039
CHARACTRON & TYPOTRON
DIGITAL DATA TRANSMITTER
CHARACTRON & TYPOTRON CONTROL
GT
CHARACTRON DIODE NETWORK
CHARACTRON DECODER
DISPLAY SCOPE
CAMERA
LIVE REGISTER #6
LIVE REGISTER #3
XD-1 DISPLAY TESTING SETUP
FROM ACCUMULATOR
MEMORY ADDRESS DECODER
HORIZONTAL SCOPE DECODER
VERTICAL SCOPE DECODER
FLEXO WRITER
CARD PUNCH
CARD READER
TAPE READER
DIGITS 0-12
DIGITS 0-12
DIGITS 2,3,4,6
DIGITS 0-1
DIGITS 4-15
DIGITS 0-6, 9-15
DIGITS 0-9
DIGITS 0-9
DATA LINK CONVERTER
DIGITAL DATA RECEIVER
A-REGISTER IN BUS
DIGITS 10-15
DIGITS 0-15
PROGRAM COUNTER
LIVE REGISTER #2
LIVE REGISTER #1
LIVE REGISTER #5
MAGNETIC DRUM MEMORY
PANEL MEMORY
PANEL ADDRESS REGISTER
CONTROL SWITCH
STEP COUNTER
CORE ADDRESS REGISTER
MAGNETIC CORE MEMORY
A-REGISTER
ACCUMULATOR
B-REGISTER
SERIAL TRANSFERS ONE DIGIT
DRUM WRITE REGISTER
GROUP & FIELD CONTROL
DRUM ADDRESS REGISTER
DIGITS 8-15
DIGITS 12-15
DIGITS 5-15
DIGITS 10-15
DIGITS 0-7
DIGITS 11-15
DIGITS 5-15
DIGITS 10-15
A-REGISTER OUT BUS
CENTRAL CONTROL
GRADE I FOR REFERENCE ONLY
GRADE II PRELIMINARY DESIGN
GRADE III FINAL DESIGN
GRADED BY: DATE:
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
DIGITAL COMPUTER LABORATORY
DEPT. OF ELECTRICAL ENGINEERING - D. I. C. PROJECT NO.
SIMPLIFIED BLOCK DIAGRAM, MTC
SCALE
DR. J.R. 5-5-55
D-47039

SECTION II. GUIDE TO CODING

COMPUTER PROGRAMS

Program. A program is a sequence of actions by which a computer handles a problem. The process of determining the sequence of actions is known as programming.

Flow diagram. A flow diagram is a series of statements of what the computer has to do at various stages in a program. Lines of flow indicate how the computer passes from one stage of the program to another.

Coded program. Programs and flow diagrams are somewhat independent of computer characteristics, but instructions for a computer must be expressed in terms of a code. A set of instructions that will enable a computer to execute a program is called a coded program, and the process of preparing a coded program is known as coding. Individual coded instructions call for specific operations such as add, shift, etc.

COMPUTER COMPONENTS

Registers and words. A register has 16 digit positions, each able to store a one or a zero. A word is a set of 16 digits that may be stored in a register. A word can represent an instruction or a number.

Arithmetic element. Arithmetic operations take place in the arithmetic element, whose main components are three flip-flop registers, the A-Register, the Accumulator, and the B-Register (AR, AC, and BR). The 16 digit positions of AR starting from the left are denoted by AR 0, AR 1 . . . AR 15. The digit positions of AC and BR are denoted in a similar fashion. AR is a buffer register, through which all numbers and instructions are transmitted from and to the various registers and memory units. AC is a register which can perform addition, subtraction, and shifting, and can transmit to storage all or part of a 16-digit word. BR is an extension of AC to the right, but BR can only shift.

Memory. Memory consists of 28,736 storage locations termed "registers," each of which is identified by field number and an address. The field number is a 4-digit binary number equal to one of the decimal numbers 0 through 14, and the address is an 11-digit binary number equal to one of the decimal numbers from 0000 through 2047. The 15-digit binary number composed of the field number followed by the address may be referred to as the "extended address." Fields 1 and 2, having the extended (decimal) addresses 1-0000 through 1-2047 and 2-0000 through 2-2047, are in magnetic core memory. Fields 3 through 14, having extended addresses 3-0000 through 3-2047, 4-0000 through 4-2047, etc., up to 14-0000 through 14-2047, are on the magnetic drum.

Panel Memory consists of 64 registers, designated field 0. It is composed of 32 toggle-switch registers with addresses 0 through 31, and a plugboard of 32 registers with addresses 32 through 63. (It may

be noted that since Panel Memory is field 0, the address and extended address designations are identical.) There are five flip-flop storage registers, which are termed "Live Registers", and which are abbreviated "LR1", "LR2", etc. LR1 may be substituted for any one or more registers of Panel Memory (addresses 0-63). LR2, LR3, LR4, and LR5 may each be substituted for any one or more registers of plugboard storage only (addresses 32-63). At the moment LR3 and LR4 are used as buffer registers between the computer and some special terminal equipment.

Input-output. Information entering the computer is temporarily stored in BR. Information leaves the computer via the AC or BR. The computer regulates the flow of information between the internal storage and AC or BR, and also calls for any necessary manipulation of external units.

Control element. The control element controls the sequence of computer operations and their execution. The control element takes its instructions one at a time from memory, where the instructions are stored as individual words.

Interconnections. The four main elements of the computer (memory, control, arithmetic, and input-output) are connected to the input and output of the A-Register by a parallel communications system, known as "A-Register In" and "A-Register Out" busses.

REPRESENTATION OF INSTRUCTIONS

Operation section. When a word is used to represent an instruction the first (left-hand) 5 digits, or operation section, specify a particular operation in accordance with the operation code.

Address section. The remaining 11 digits or address section, are interpreted as a number with the binary point at the right-hand end. For the majority of instructions this number is the address of the register whose contents will be used in the operation. In the instructions cr, sr, pr, and ri, the number specifies the extent of a shift. In these and certain other instructions, various digit positions of the address part are used to specify "variations" of the operation given by the operation part.

Example. The instruction ca x has the effect of clearing AC (making all the digits zero) and then copying into AC the word that is in the register whose address is x. If q is a quantity in some register, the operation needed to copy q in AC is not ca q but ca x, where x is the address of the register that contains q.

REPRESENTATION OF NUMBERS

Single-word representations. When a word is used to represent a number the first digit indicates the sign and the remaining 15 are numerical digits. For a positive number the sign digit is zero, and the

15 numerical digits with a binary point at their left specify the magnitude of the number. The negative, -y, of a positive number y is represented by complementing all the digits, including the sign digit, that would represent y. (The complement is formed by replacing every zero by a one and every one by a zero.) In this way a word can represent any multiple of 2^{-15} from $-1 + 2^{-15}$ to $1 - 2^{-15}$. Neither +1 nor -1 can be represented by a single word. Zero has two representations, either 16 zeros or 16 ones, which are called +0 and -0 respectively.

Overflow; increase of range and precision. With a single-word representation numbers are limited to the range $(-1 + 2^{-15})$ through $(1 - 2^{-15})$. Programs must be so planned that arithmetic operations will not cause an overflow beyond this range. The range may be extended by assuming a scale factor or by using 2 registers (30 digits, not counting sign) to represent a single number. An instruction which results in an overflow will normally cause an "overflow alarm" during the succeeding instruction, and stop the computer. Two instructions, "transfer on overflow" (to) and "shift right" (sr), can be used to detect the overflow without any alarm. There is also a switch by which the alarm may be suppressed entirely.

Scale factors. In order that computations can be performed involving numbers outside the limited range of $-1 + 2^{-15}$ to $+1 - 2^{-15}$ all numbers with which the computer must deal are made to fall in this range by the application of scale factors. A scale factor is a factor, or multiplier, remembered by the programmer, by which a number is multiplied so that it falls within the computer's range. For example, in order to store the number 36 in an MTC register it must be multiplied by some number, say 2^{-7}, to make the product smaller than $+1 - 2^{-15}$. Then 2^{-7} becomes the scale factor.

Scale factors customarily are chosen to be some power of 2, principally as a matter of convenience. During the course of a program it occasionally becomes necessary to change the scale factor of a number. Since MTC operates in the binary number system, the simple process of shifting a number to the left or right is equivalent to multiplying or dividing by a power of 2. If the stored number's scale factor initially is a power of 2, then shifting the number will change the scale factor to some other power of 2.

Algebraic principles demand that 2 numbers can be added together only if their multipliers, or scale factors, are the same. We cannot, therefore, add directly the quantities 5×2^{-6} and 2×2^{-4}. We must first make the scale factors equal by, for example, making the scale factor of second quantity equal to that of the first through multiplying it by 2^{-2}. This is accomplished physically in the computer by shifting the

quantity 2×2^{-4} two places to the right, making it 2×2^{-6}. Then we can add the two quantities thus: $(5 \times 2^{-6}) + (2 \times 2^{-6}) = 7 \times 2^{-6}$. One reason for remembering scale factors should now be obvious.

In the case of multiplying two numbers, the rules of algebra do not require that their scale factors be adjusted. The result of a multiplication is simply the product of the 2 original numbers accompanied by a scale factor which is the product of the 2 original scale factors. Hence we can multiply directly $(5 \times 2^{-6}) \times (2 \times 2^{-4})$ to get 10×2^{-10}. Here again we must keep in the mind the scale factors of the multiplier and the multiplicand so that we can determine the scale factor of the product.

Some care must be exercised in the choice of a scale factor. A number in MTC is represented by fifteen binary bits (or 5 octal digits) to the right of the point. This is very roughly equivalent to 3 decimal digits. Where possible we want these 3 digit positions to contain significant digits (but not initial zeros, significant or not). We can make this so by choosing the proper scale factor. For example, suppose we wish to store the decimal fraction .003512. If we did not apply a scale factor, we could store only the first 3 digits to the right of the point of which two digits are 0, so the stored number would be precise only to 1 digit. If, however, we multiply by a scale factor of 2^8, our number becomes .899072. Now we are rid of the initial zeros, and we can store .899, which gives us a precision of 3 decimal digits.

COMPUTER PROCEDURE

Sequence of operations. After the execution of an instruction, the Program Counter in the control element holds the address of the register from which the next instruction is to be taken. Control calls for this instruction and carries out the specified operation. If the operation is not a transfer of control, the address in the Program Counter then increases by one so that the next instruction will be taken from the next consecutive register. The "transfer of control" instructions permit resetting the Program Counter to a particular number so the next instruction can be taken from a particular register.

Memory address selection scheme. The 11 binary digits of the address section are sufficient to uniquely specify one out of only 2048 memory addresses in the computer. At least 4 additional digits are required in order to specify all 28,736 registers. Hence, it has been necessary to deal with memory as numbered "fields" of 2048 registers. Since an instruction obviously cannot give the field number in addition to the address, the field number for each address must somehow be given in advance of executing the instruction. The scheme for doing this is as follows:

The instruction sof, select operation field, is used to choose the field of registers to which the address parts of all succeeding instructions refer. (The number of the field currently selected is held in the "Operation Field Register," which is reset each time an sof instruction is executed.)

The field of registers from which instructions are currently being taken is specified in the Program Field Register. By a "transfer out" (tro) instruction, this register may be reset to the field number held in the Operation Field Register. Thus if the Operation Field Register contains n, and a tro x instruction is executed, succeeding instructions will be taken from field n, starting with register x.

An illustration of the use of these instructions will be found on p. 28.

Copy and store instructions. The copying of a word, or part of a word, from one location to another affects only the latter location, whose previous contents are lost.

Zero. The number zero, if it results from a sequence of operations the last of which is addition, will be represented as negative zero (binary 1.111 111 111 111 111), except that plus zero added to plus zero gives plus zero. If the last operation is a subtraction (including "subtract magnitude") the representation will be positive zero (binary 0.000 000 000 000 000), except that plus zero subtracted from minus zero gives minus zero. The sign of a zero resulting from shifting is the same as the sign of the number before shifting. (An exception may occur with the use of the "cycle right" instruction.)

Manipulation of instructions. Words representing instructions may be handled in the arithmetic element as numbers.

Procedure in the arithmetic element. The execution of an addition includes the process of adding in carries; this process treats all 16 digits as if they were numerical digits, a carry from AC 0 ("end-around carry") being added into AC 15. (This compensation is necessary because of the method of representing negative numbers.) A subtraction is executed by complementing the AC, adding the subtrahend, and complementing the AC again. Roundoff may be performed by the instructions cr 1000+n and sr 1000+n (octal notation), where n is the amount of shift preceding roundoff. (see cr, sr instructions, p. 24f.)

NOTATION FOR CODING

Addresses. A coded program requires certain registers to be used for specified purposes. The addresses of these registers must be chosen before the program can be run on the computer, but for study purposes this final choice is unnecessary, and the addresses can be indicated by a system of symbols or index numbers.

Writing a coded program. Registers from which control obtains instructions may be called action registers, and should be listed separately from registers containing other information, which may be called data registers. A coded program is written out in two columns: the first contains the index number of each action or data register, and the second column indicates the word that is initially stored in that register. In many cases part or all of a word may be immaterial because the contents of the register in question will be changed during the course of the program. This state of affairs is indicated by a dash, for example, ca—.

Conventional notation. In order to make a program more readily understandable to others and more easily remembered by the author himself, it is desirable to write short descriptions of the functions served by certain key instructions and groups of instructions. It is also desirable to indicate breaks and confluences in the "flow" of the program and to indicate instructions which are altered or otherwise abnormally used during the program. Some generally accepted symbols for this purpose are exemplified and described below.

	120	ra 124	
start -->	121	ca 161	initial entry (i.e., start of program)
	122	ra 132	
139 --->	123	ca 181	re-entry point, showing origin of re-entry
	124	su (182)	address altered by program, initial value shown
	125	sr 16	
	126	tn 128	conditional short break in consecutivity (note other form below)
	127	ad 140	
	128	ad 133	
	129	st	address indicated by arrow (e.g. address = 130 in this case), used primarily at early stages of writing
	130	(ca217/cs217)	word altered by program, alternative values shown
	131	tr 78	no break in consecutivity, despite tr operation, where a closed subroutine is called in

(122,167)	132	st (-)	address altered by program, initial value immaterial, locations of altering instructions shown, alternative values not shown
	133	ca 217	semi-pseudo instruction, serves both as instruction and number
	134	tr 136	short break in consecutivity, used especially where a closed subroutine with program parameters is called for
	135	su 115	
	136	ad 114	
	137	tn 141	conditional break in consecutivity (note short form above)
	138	st 114	
	139	tr 123	break in consecutivity (note short form above)
	140	ra 0	pseudo-instruction, serves only as a number, not as instruction
137, 171->141		st 171	entry point, showing origins of entry

"Floating address" notation. A "floating address" system of notation enables a programmer to write his instructions so that they refer to the words of his program and not to the location of those words in memory.

For example, consider the following set of instructions with fixed addresses:

```
32  ca 41
33  ad 100
34  st 41
35  ca 42
36  ad 100
37  st 42
38  ca 43
39  ad 100
40  st 43
41  ca 101
42  mh 102
43  st 103
44  tn 32
```

Seven of these instructions refer to the locations of other instructions within the group. If any instructions (or words) were to be added to or deleted from this set, a considerable amount of renumbering would be necessary, in general. A floating address system removes the need for this, by labelling each word to which reference is made by a floating address label. The floating address is of the form $\alpha\#$, where α is any lower-case letter of the alphabet except o and l, and where # is any integer of the form 1, 2, 3,..., with initial zeros suppressed. This floating address is then used as the address section of any instruction which is to refer to the word so labelled. Thus the above program might become:

```
f3,  ca m9
     ad 100
     st m9
     ca h5
     ad 100
     st h5
     ca b2
     ad 100
     st b2
m9,  ca 101
h5,  mh 102
b2,  ts 103
     tn f3
```

Note that floating addresses may be used in any sequence and that words referring to a floating address may occur either earlier or later than the word labelled by the corresponding floating address. Thus insertions into or deletions from such a program may be made without any renumbering or any alterations to the existing words.

Through the medium of a "floating-address conversion program," a computer can accept a program written in floating address notation. Because MTC does not at the present time, however, possess this type of conversion program, programs coded for MTC must be translated into "actual address form" before they can be introduced into the computer.

The abbreviations RC, CR. Abbreviations used in referring to the register that contains a certain word or the word in a certain register are

RC . . . = (Address of) register containing....

This is sometime symbolized $\lfloor$ ...

CR . . . = Contents of register (whose address is) ...

This is sometime symbolized (...)

The symbol ha x. When an address forms part of an instruction it is represented by the last 11 digits of a word whose first 5 digits specify an operation. An address that is not part of an instruction is represented by the last 11 digits of a word whose first 5 digits are zero, which is equivalent to specifying the operation ha. Thus the word for an unattached address x may be written ha x. It may also be written as +x or as $+x \times 2^{-15}$.

SECTION III. INSTRUCTION CODE

DESCRIPTION OF THE INSTRUCTION CODE

Introduction. Included under each instruction are the function, the contents (if altered) of AC, BR, and register x after the instruction and possible alarms.

Abbreviations. The abbreviations used are the following:

AC	=	Accumulator	x	=	address of a memory register
AR	=	A-Register	a,k,n	=	positive integers
BR	=	B-Register			

Contents of various registers. The contents of AC, BR, AR, and the register whose address is x are undisturbed unless the contrary is stated.

Reduction modulo q. A number p reduced with respect to the modulus q, "modulo q," is defined as the numerator of the fractional remainder when p is divided by q.

Example 1: 60 mod 32: $\frac{60}{32} = 1 + \frac{28}{32}$, hence 60 mod 32 = 28

Example 2: 1.37 mod 1: $\frac{1.37}{1} = 1 + \frac{.37}{1}$, hence 1.37 mod 1 = .37

Individual instruction. The following pages treat in detail the actions of each individual instruction.

Abbreviation	Name	Number	Binary	
ha -	halt	#0	00000	ha

Stop the computer. The address section is of no significance.

mh x	multiply	#1	00001	mh

Multiply the contents of AC and the contents of register x, leaving a 30-digit product of proper sign in AC and in BR 0-14, and leaving a zero in BR 15. The previous contents of BR are lost. Cancel any "overflow condition" which may exist. (See "Overflow," p. 27.)

ds x	display	#2	00010	ds

Display on the plotting scope a point whose horizontal deflection is specified by the contents of digit positions 0-9 of register x, and whose vertical deflection is specified by the contents of digit positions 0-9 of AC.

Abbreviation	Name	Number	Binary	
id x	identity check	#3	00011	id

Compare the contents of AC with the contents of register x. If it is not identical, give an "identity check" alarm and prepare to skip the immediately succeeding instruction (by adding 1 to the Program Counter). The identity check alarm feature may be suppressed by a switch on the Alarm Suppression Panel. After the instruction has been executed, AC will contain 1's in each digit position where the original digit did not agree with the corresponding digit position of register x. All other digit positions in AC will be zero.

st x	store	#4	00100	st

Store in register x a copy of the contents of AC. The previous contents of register x are destroyed. When x is the address location of LR4, st x performs the special function of reading in to LR4 from the AN/GKA-1 Data Link Converter. For further details refer to Data Link Converter, p. 33.

ra x	replace address	#6	00110	ra

Replace digits 5-15 of register x with a copy of digits 5-15 of AC. Because digits 5-15 of an instruction are termed the "address section," the instruction ra in effect replaces the address section of the contents of register x with a copy of the address section of the instruction contained in AC.

rf x	return from	#7	00111	rf

Replace digits 5-15 of register x with a copy of digits 5-15 of LR2. LR2 will normally contain an address y+1, where y is the address of the instruction in core or drum memory which effected the most recent transfer of control. (See tr instruction below.)

ca x	clear and add	#8	01000	ca

Clear AC (but not BR), and add in a copy of the contents of register x.

ad x	add	#9	01001	ad

Add to the contents of AC a copy of the contents of register x. If an overflow occurs, the succeeding instruction must be to or sr b (where $b \neq 0$) or an overflow alarm will be given and the computer will stop. If the result in AC after the execution of this instruction is zero, it will be a negative zero; exception: $(+0) + (+0) = +0$.

Abbreviation	Name	Number	Binary	
cs x	clear and subtract	#10	01010	cs

Clear AC (but not BR), and put in AC the negative of the contents of register x, except that cs RC (+0) will leave +0 in AC.

su x	subtract	#11	01011	su

Add to contents of AC the negative of the contents of register x. If an overflow occurs, the succeeding instruction must be to or sr b (where b $\neq$ 0) or an overflow alarm will be given and the computer will stop. If the result in AC after the execution of this instruction is zero, it will be positive zero; exception: (-0) - (+0) = -0.

et x	extract	#12	01100	et

In each digit position of AC which contains a "1" substitute a copy of the contents of the corresponding digit position of register x. Alternatively stated, put in each digit position of AC the logical product of the present contents of that digit position and the corresponding digit position of register x. Cancel any overflow condition which may exist (see "Overflow," p. 27).

ch x	charactron display	#14	01110	ch

Display a point, vector, or character on the Charactron, or display a character on the Typotron, depending on which unit is selected. The selection of a scope unit, and the nature and position of the display are determined by the contents of AC, LR1, LR2, LR5, and register x. Refer to detailed discussion on Charactron and Typotron, p. 37.

sm x	subtract magnitudes	#15	01111	sm

From the positive magnitude of the contents of AC subtract the positive magnitude of the contents of register x. The previous contents of AC are left in BR, and the previous contents of BR are lost. If the result in AC after the execution of this instruction is zero, it will be positive zero.

tr x	transfer	#16	10000	tr

Prepare to take the next instruction from (that is, transfer control to) register x of the same field where this instruction was stored. If this instruction is stored at address y in core or drum memory, put the instruction op y+1 in LR2, the previous contents of LR2 being lost. If this instruction is stored in panel memory, the contents of LR2 are undisturbed. Set LR3 and LR4 to negative zero.

Abbreviation	Name	Number	Binary	
tro x	transfer out	#17	10001	tro

Prepare to take the next instruction from register x of field a, where a is the field specified by the last preceding "select operation field" instruction (see sof instruction below). If this instruction is stored at address y in core or drum memory, put the instruction op y+1 in LR2, the previous contents of LR2 being lost. If this instruction is stored in panel memory, the contents of LR2 are undisturbed. Set LR3 and LR4 to negative zero.

tn x	transfer on negative	#18	10010	tn

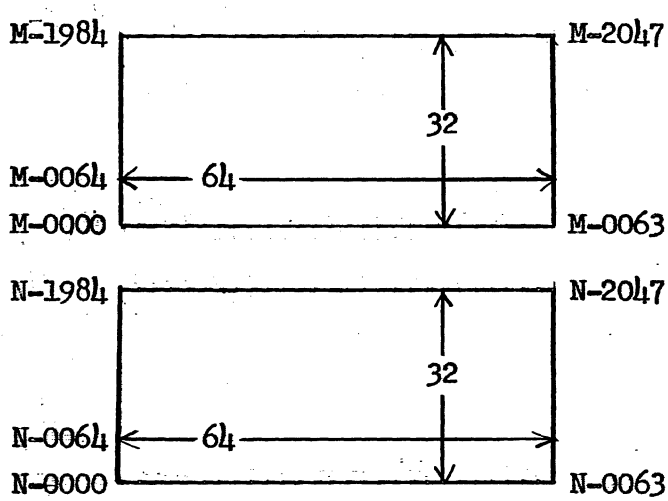
If the number in AC is negative (i.e., has a "1" in digit position 0), perform the same function as tr above. If the number in AC is positive, this instruction has no effect other than to set LR3 and LR4 to negative zero.

tno x	transfer on negative out	#19	10011	tno

If the number in AC is negative, perform the same function as tro above. If the number in AC is positive, this instruction has no effect other than to set LR3 and LR4 to negative zero.

md x	memory address display	#20	10100	md

Display a point on the memory address scope, the position of the point corresponding to the address x. The correspondence is as follows:



M = any even-numbered field*

N = any even-numbered field*

* The field number specified by the most recent sof instruction.

Abbreviation	Name	Number	Binary	
sof a	select operation field	#21	10101	sof

Until the next sof instruction is given, treat address parts of instructions which refer to memory as referring to field a. If digit 5 of the instruction contains a "1," generate a programmed alarm. (The programmed alarm feature may be suppressed by a switch on the alarm suppression panel.)

to x	transfer on overflow	#22	10110	to

If an overflow condition exists, cancel the overflow condition and perform the same function as tr above. (See "Overflow," page 27.) If no overflow condition exists, this instruction has no effect other than to set LR3 and LR4 to negative zero.

cr n	cycle right	#24	11000	cr

If digit 7 of the instruction contains a "0," perform "regular cycle": shift the entire contents of AC and BR to the right n places (n is treated as reduced modulo 32), the digits shifted out of BR15 being introduced into AC0 so that no digits are lost.

If digit 7 of the instruction contains a "1," perform "short cycle": shift the contents of AC and BR, with the exception of AC0, to the right n places (n is treated as reduced modulo 32), the digits shifted out of BR15 being introduced into AC1 so that no digits are lost. In "short cycle" the contents of AC0 remains undisturbed.

If digit 6 of the instruction contains a "1," roundoff the magnitude of the number in AC and BR to 15 numerical digits in AC. If roundoff is specified, it occurs after any cycling which may have been specified. The contents of BR are not disturbed by the roundoff process.

If digit 5 of the instruction contains a "1," clear BR after the cycle (if any) and roundoff (if any) have been executed.

Read the cautionary footnote on page 25.

For convenience, the variations of the cr instruction are given in the table below with their corresponding octal and decimal addresses: (n is less than 32 decimal).

Abbreviation	Name	Number	Binary
octal	decimal	operations	
cr n	cr n	regular cycle.	
cr 400+n	cr 256+n	short cycle.	
cr 1000+n	cr 512+n	regular cycle, roundoff.*	
cr 1400+n	cr 768+n	short cycle, roundoff.	
cr 2000+n	cr 1024+n	regular cycle,	clear BR.
cr 2400+n	cr 1280+n	short cycle,	clear BR.
cr 3000+n	cr 1536+n	regular cycle, roundoff,	clear BR.*
cr 3400+n	cr 1792+n	short cycle, roundoff,	clear BR.
sr n	shift right	#25	11001 sr

Shift the contents of AC and BR, excepting the sign digit, to the right n places (n is treated as reduced modulo 32). The digits shifted right out of BR15 are lost. The sign digit will be introduced into every digit position left vacant by the shift. If an overflow condition exists, and if n modulo 32 is not equal to zero, the overflow will be recovered on the first shift so that the result in AC and BR will be the original sum divided by 2^n, and the overflow condition will be cleared. (See "Overflow," p. 27.)

If digit 6 of the instruction contains a "1," roundoff the magnitude of the number in AC and BR to 15 numerical digits in AC. If roundoff is specified, it occurs after any shifting which may have been specified. The contents of BR are not disturbed by the roundoff process.

If digit 5 of the instruction contains a "1," clear BR after the shift (if any) and roundoff (if any) have been executed.

For convenience, the variations of the sr instruction are given in the table below with their corresponding octal and decimal addresses: (n is less than 32 decimal).

*Since it was anticipated that the combined feature "regular cycle, plus roundoff" would not normally be useful, equipment was saved by allowing this combination of operations to produce a result which is not always the obvious one. Programmers are warned to avoid this combination unless they can deduce the appropriate result!

Abbreviation	Name	Number	Binary
octal	decimal	operations	
sr n	sr n	shift right.	
sr 1000+n	sr 512+n	shift right, roundoff.	
sr 2000+n	sr 1024+n	shift right,	clear BR
sr 3000+n	sr 1536+n	shift right, roundoff,	clear BR.
pr n	print/punch	#28	11100 pr

If digit 7 of the instruction is a "1," perform "regular cycle" (see cr above), then actuate a key on the printer corresponding to the contents of digits 10-15 of AC.

If digit 7 of the instruction is a "0," perform "regular cycle" (see cr above), then punch a line of paper tape, hole positions 1 to 6 on the punched tape corresponding to the contents of digits 10-15 of AC. If digit 6 of the instruction pr n is also a "0," punch the 7th hole position.

If digit 5 of the instruction is a "1," clear BR after the cycling (if any) is completed.

The variations of the pr instruction are given in the table below with their corresponding octal and decimal addresses: (n is less than 32 decimal)

octal	decimal	operations
pr n	pr n	regular cycle n, punch (incl. 7th hole).
pr 400+n	pr 256+n	regular cycle n, print.
pr 1000+n	pr 512+n	regular cycle n, punch (no 7th hole).
pr 2000+n	pr 1024+n	regular cycle n, punch (incl. 7th hole), clear BR.
pr 2400+n	pr 1280+n	regular cycle n, print, clear BR.
pr 3000+n	pr 1536+n	regular cycle n, punch (no 7th hole), clear BR.

Abbreviation	Name	Number	Binary	
ri n	read in	#30	11110	ri

Read the next line of punched paper tape which is accompanied by a punch in the 7th hole position. Put in BR 10-15 the 6-digit binary combination corresponding to the pattern of punches. After the read has been executed, perform "regular cycle" (see cr above).

If digit 5 of the instruction contains a "1," clear BR after the cycling (if any) is completed.

op k	operate	#31	11111	op

Perform a function which is determined by the address k.

If k = 0, index the camera; that is, advance the camera film one frame, momentarily closing the shutter while the film is in transit.

If k = 2000 (octal) or 1024 (decimal), erase the stored display on the Typotron.

If k = 1000+n (octal), read a word from the card reader into BR. The integer n, reduced modulo 4, selects from which one of the four columns of 16-bit word positions the word will be read.

If k = 3000+n (octal), punch in a card the word contained in BR. The integer n, reduced modulo 4, selects into which one of the four columns of 16-bit word positions the word will be punched.

For a further explanation of card reading and punching, refer to the discussion of the IBM card punch and reader (p. 32f).

NOTES ON THE INSTRUCTION CODE

Overflow. Every instruction except to (transfer on overflow) and sr (shift right) performs an overflow check to see whether an overflow has occurred (due to the execution of a preceding ad or su instruction). If an overflow condition exists, an overflow alarm is generated and the computer stops. The alarm may be anticipated and nullified by inserting to or sr b (where b $\neq$ 0) into the program immediately after the overflow-generating ad or su. Note that sr 0 can neither detect the overflow condition nor generate an alarm, hence any overflow condition must still be treated by a succeeding instruction. Alternatively, the overflow alarm may be manually suppressed, in which case overflows may be ignored or detected at the programmer's discretion. With the overflow alarm suppressed, however, if an overflow condition exists when a second overflow occurs, the overflow condition will automatically be canceled. The programmer must then remember that if he wishes to detect a specific overflow, he must insure that no overflow condition already exists at the time the overflow which he wishes to detect occurs. This can be accomplished by the execution of any instruction which removes an overflow condition, namely: et, to, sr b where b $\neq$ 0, and mh.

Because of the circuit logic chosen to handle overflows, a slightly misleading situation can arise in connection with overflow alarms. If the overflow alarm is not suppressed, and if the following succession of events occur:

1. An ad or su instruction generates an overflow;

2. The immediately succeeding instruction is either:

   a. an ad or su instruction which generates a second overflow,

   b. an mh or et instruction which will clear any overflow,

the computer will stop with the master alarm indicator ON, but the overflow alarm light will not be on.

The "field-switching" instructions sof and tro. The following portion of a program illustrates the use of the instruction sof and tro.

Location	Instruction	Effect
0-0	sof 1	Prepare to take next instruction
0-1	tro 106	from register 106 of field 1
1-106	ca 120	Add the contents of 1-120 to those
1-107	sof 2	of 2-35 and store the sum in 1-121.
1-108	ad 35	
1-109	sof 1	
1-110	st 121	
1-111	tr 200	Go to register 200 of field 1 for the next instruction.

Shifting left. An instruction which shifts a number in AC and BR to the left has been omitted from MTC because of the relatively large amount of equipment which it would require. By making use of the "short cycle" feature of the cr instruction, the theoretical instruction "shift left n" can be simply programmed by the following triple of instructions:

octal	decimal
cr 437-n	cr 287-n
sr n	sr n
cr 437-n	cr 287-n

Under certain circumstances, "shift left n" may be approximated by the first cr instruction alone.

EXECUTION TIMES OF INSTRUCTIONS

Individual instruction execution times. Given in the chart on the next page are the times for the execution of instructions which are stored in panel or core memory and which refer to operands in either panel or core memory. The times in the table, calculated from information shown on block schematic drawings of the control element, may be in error by an estimated ±5 per cent. Since individual instruction execution times are not constant for instructions stored on, or referring to, the magnetic drum, they are not included in the chart.

Average instruction times. Estimates of average instruction times for MTC programs are given in the table below. The assumptions on which this table is based are that programs consist of no in-out instructions (ds, ch, md, pr, ri, and op), 5% mh instructions, and 5% transfer instructions (tr, tro, tn, tno, and to). Data is assumed to be at random on the drum; the more ordered the data, the shorter will be the average instruction time for instructions stored in panel or core memories which refer to data on the drum.

		Instructions in:		
		Panel Memory	Core Memory	Drum Memory
Instructions refer to:	Panel Memory	13.8	14.3	1182
	Core Memory	12.6	13.1	1182
	Drum Memory	10,300	10,300	20,600

AVERAGE MTC INSTRUCTION EXECUTION TIMES (μsec)

TIMING RESTRICTIONS DUE TO DRUM MEMORY

Because writing on the magnetic drum creates transient conditions in the drum circuits which persist for about 40 μsec, any attempt to read from the drum within 40 μsec after writing may result in an erroneous word being read. There is at present no circuitry which will automatically guard against this occurrence, hence it is the programmer's responsibility to do so.

There is actually a very small probability, however, that this situation will arise in normal programs. In view of this fact, it may well be appropriate for the programmer not to consider the problem unless he encounters otherwise unexplainable drum parity alarms during the operation of his program.

For programmers who do wish to take the problem into account, the following discussion is presented.

Rules for determining disallowed combinations of instructions can be devised from the following facts:

1. The physical sequence of addresses on the drum surface is

0,	128,	256,	. . .	128n,	. . .	1792	1920,
1,	129,	257,	. . .	128n+1,	. . .	1793,	1921,
. . .							
. . .							
k,	128+k,	256+k,	. . .	128n+k,	. . .	1792+k,	1920+k,
. . .							
. . .							
126,	254,	382,	. . .	128n+126,	. . .	1918,	2046,
127,	255,	383,	. . .	128n+127,	. . .	1919,	2047.

(where $0 \leq n \leq 15$; $0 \leq k \leq 127$.)

2. Adjacent register positions on the drum arrive under the drum read-write heads at intervals of 10 μsec.

3. If an instruction is being taken from drum address x, the "program timing cycle" of the instruction must have begun more than 13 μsec before the drum register x arrives under the heads.

4. If a word is being read from the drum by an instruction, the instruction must have begun more than 18 μsec before the selected register arrives under the heads.

From the above facts a precise rule can be derived: For a given reference to the drum which involves reading a word from the drum, no part of an instruction which writes on the drum (i.e., st, ra, or rf)

EXECUTION TIMES* FOR INSTRUCTIONS WHICH REFER TO MEMORY				
INSTRUCTION	Instruction in PANEL Memory		Instruction in CORE Memory	
	Address Part Refers to PANEL Memory	Address Part Refers to CORE Memory	Address Part Refers to PANEL Memory	Address Part Refers to CORE Memory
ha (note 1)	10.8 μs	11.3 μs	11.3 μs	11.8 μs
mh (note 2)	51.7 + 0.5q μs	51.7 + 0.5q μs	52.2 + 0.5q μs	52.2 + 0.5q μs
ds	19 μs	19 μs	19 μs	19 μs
id	11.7 μs	10.4 μs	12.2 μs	10.9 μs
st	11.3 μs	11.3 μs	11.8 μs	11.8 μs
ra	14.4 μs	16.6 μs	14.9 μs	17.1 μs
rf	14.4 μs	16.6 μs	14.9 μs	17.1 μs
ca	11.7 μs	10.4 μs	12.2 μs	10.9 μs
ad	11.7 μs	10.4 μs	12.2 μs	10.9 μs
cs	11.7 μs	10.4 μs	12.2 μs	10.9 μs
su	11.7 μs	10.4 μs	12.2 μs	10.9 μs
et	11.7 μs	10.4 μs	12.2 μs	10.9 μs
ch:				
Charactron Vector	250 μs	250 μs	250 μs	250 μs
Charactron Character	225 μs	225 μs	225 μs	225 μs
Typotron Character	290 μs	290 μs	290 μs	290 μs
sm	12.0 μs	12.0 μs	12.5 μs	12.5 μs

EXECUTION TIMES* FOR INSTRUCTIONS WHICH DO NOT REFER TO MEMORY		
INSTRUCTION	Instruction in PANEL Memory	Instruction in CORE Memory
tr	9.5 μs	10.0 μs
tro	9.5 μs	10.0 μs
tn	9.5 μs	10.0 μs
tno	9.5 μs	10.0 μs
md	9.3 μs	9.3 μs
sof	9.1 μs	9.6 μs
to	9.5 μs	10.0 μs
cr (note 3)	9.3 + .73n μs	9.8 + .73n μs
sr (note 3)	9.3 + 73n μs	9.8 + .73n μs
pr (punch)	83 ms	83 ms
pr (print):		
Carriage return or tabulation	200-500 ms	200-500 ms
Characters and other machine functions	125 ms	125 ms
ri (Ferranti)	4.8 ms	4.8 ms
ri (Flexo)	110 ms	110 ms
op (index camera)	1.4 sec	1.4 sec
op (erase Typotron)	40 μs	40 μs
op (card reader)	(Note 4)	(Note 4)
op (card punch)	(Note 4)	(Note 4)

* Estimated accuracy of times is ± 5 percent.

Note 1. Although ha does not make use of an operand from memory, the address part of the instruction nevertheless refers to one.

Note 2. Where q = the number of binary ONES in the positive magnitude of the number of AC before execution.

Note 3. Where n = the number of steps in the shifting or cycling operation.

Note 4. The execution times of the op instructions associated with the card reader and punch are totally dependent on the mechanical positions of the reader and punch. After either has come up to speed, the rate of reading or punching is 48 words (= 1 card) in 0.6 seconds, or an average of 80 words per second.

INDIVIDUAL MTC INSTRUCTION EXECUTION TIMES*
AS OF 21 MARCH 1955

may have been performed in the preceding 27 μsec in the case of #3 above, or in the preceding 22 μsec in the case of #4 above.

When maximum program speed is not essential, the rule can be simplified: If an instruction is stored at drum register x or has an address part specifying drum register x, the 25 μsec interval preceding the execution of such an instruction must not contain any part of an instruction which causes writing in the three drum registers immediately preceding register x.

SECTION IV. TERMINAL EQUIPMENT

CARD EQUIPMENT

Card punch. The card punch is part of an IBM 513 Reproducing Punch, suitably modified to operate under computer control. This unit can punch IBM cards at a maximum rate of 100 cards per minute, with 48 16-bit binary words to a card. Punching follows the usual convention: a hole stands for a binary "1," and the absence of a punch indicates a binary "0." The binary words are normally punched in card columns 17-80, in 12 rows of 4 words each, starting with the "nines," or bottom, row. Other punching formats are possible by providing different plugboard connections. (Refer to "Card Machine Controls," p. 69f.)

The card punch is actuated by the instruction op 3000+n (octal). This instruction punches the word contained in BR into word position 0, 1, 2, or 3 (according to the value of n taken modulo 4) of the next row not already completed. The card punch circuits require that exactly 48 instructions op 3000+n be given for each card to be punched. In these 48 instructions the value of n modulo 4 should be successively 0, 1, 2, 3, 0, 1, 2, 3, 0, etc. A convenient way to program these values of n is to index the instruction op 3000+n by adding 1×2^{-15} to the instruction itself for each succeeding word to be punched. The resulting sequence of instructions to punch 48 words would thus be:

op 3000, op 3001, op 3002 . . . . . op 3057 (octal).

Since once a card is started through the punching mechanism it proceeds at a constant rate, there are restrictions on the allowable intervals between the punching instructions:

1. Between the first and last of the four instructions which punch words in the same row on a card not more than 4 milliseconds may elapse.

2. Between the instruction which punches the last word in one row and the instruction which punches the first word in the next row (on the same card) not more than 36 milliseconds may elapse.

If either of the above restrictions is violated, the punch will issue a card machine alarm, and the computer will stop.

If it is desired to keep the punch operating at its maximum rate of 100 cards per minute, an additional timing restriction must be observed: between the instruction which punches the last (48th) word on a card and the instruction which punches the first word on the succeeding card not more than 24 milliseconds may elapse.

Card reader. The card reader is part of an IBM 513 Reproducing Punch, modified to provide proper signals to the computer. This unit can read IBM cards at a maximum rate of 100 cards per minute. The cards are normally read as though they contained 16-bit binary words in columns 17-80, in 12 rows of 4 words each, starting with the "nines," or bottom, row. By providing different plugboard connections, however, any 64 of the total of 80 card columns can be read in any configuration. (Refer to "Card Machine Controls," p.69f.)

The card reader is actuated by the instruction op 1000+n (octal). This instruction reads into BR the word contained in word position 0, 1, 2, or 3 (according to the value of n taken modulo 4) of the next row not already finished. The card reading circuits require that exactly 48 instructions op 1000+n be given for each card to be read. In these 48 instructions the value of n modulo 4 should be successively 0, 1, 2, 3, 0, 1, 2, 3, 0, etc. A convenient way to program these values of n is to index the instruction op 1000+n by adding 1×2^{-15} to the instruction itself for each succeeding word to be read. The resulting sequence of instructions to read 48 words would thus be:

op 1000, op 1001, op 1002 . . . . . op 1057 (octal).

Since once a card is started through the reading mechanism it proceeds at a constant rate, there are restrictions on the allowable intervals between the reading instructions:

1. Between the first and last of the four instructions which read words in the same row on a card not more than 12 milliseconds may elapse.

2. Between the instruction which reads the last word in one row and the instruction which reads the first word in the next row (on the same card) not more than 28 milliseconds may elapse.

If either of the above restrictions is violated, the reader will issue a card machine alarm, and the computer will stop.

If it is desired to keep the reader operating at its maximum rate of 100 cards per minute, an additional timing restriction must be observed: between the instruction which reads the last (48th) word on a card and the instruction which reads the first word on the succeeding card not more than 24 milliseconds may elapse.

DATA LINK EQUIPMENT

Data Link Converter (AN/GKA-1). The AN/GKA-1 Data Link Converter is part of the AN/GKA-1 Data Link. The function of the converter is to receive and store in a relay register the 12-bit digital messages arriving on a manually-designated subcarrier frequency. The execution of

the special instruction st LR4 will put the complement of the latest 12-bit message into digit positions 4 to 15 of LR4 (normally at address 0-77 octal); positions 0-3 will contain ONEs. At the time the program requests a message by st LR4, however, the relays in the converter may be in the process of transition to a new message. Hence it is necessary for the programmer to request the message several times at intervals comparable to a relay operate-time (about 3 milliseconds). This must be repeated until the same message has been received in response to two successive requests.

All transfer instructions (trn, tn, tro, tno, and to) reset LR4 to negative zero. This resetting function is needed in connection with Charactron and Typotron displays, but it must be disabled when LR4 is required in a program using the Data Link Converter. This is done by unplugging the single coax cable at C308 J3-6.

Digital Data Transmitter and Receiver. A Digital Data Transmitter (DDT) and a Digital Data Receiver (DDR) together provide a facility for transmitting digital messages over a telephone circuit. MTC is equipped with both a DDT and a DDR so that it can transmit messages to, or receive messages from, similar equipment at a remote point.

A message consists of a train of pulses representing binary ZEROs, binary ONEs, and synchronizing (SYNC) signals. The rate at which these pulses are transmitted is termed the "data rate" and is either 1300 or 1600 pps (pulses per second). The data rate and the structure, or format, of a message depends on the exact use to which the message will be put.

Instructions for programming the transmission of a message are given as rules and explanations in the following numbered paragraphs:

1. In order to transmit a message, the computer program must specify to the DDT at each pulse period* which one of the three types of pulses should be transmitted. Each of the types of pulses is designated by a 2-bit binary code:

Pulse	Pulse Code
ZERO	00
ONE	01
SYNC	10

*The pulse period is the interval between the end of one pulse and the end of the succeeding pulse. Numerically this is equal to the reciprocal of the data rate.

To instruct the DDT to transmit a particular pulse, the program places the binary code designation for the desired pulse into digit positions 2 and 3 of LR3 (normally at address 0-51 octal).

2. The start of a message is indicated by a SYNC pulse. The circuit design of the DDR requires that this SYNC pulse be immediately preceded by at least one binary ZERO, and followed by either a binary ZERO or another SYNC pulse.

3. In addition to specifying the type of pulse to be transmitted each pulse period, the program must generate timing signals for the DDT. Timing signals are supplied ordinarily by complementing the contents of digit position 6 of LR3 at the start of each pulse period. The circuit design of the DDR requires, however, that every individual SYNC pulse (or the first of a pair of adjacent SYNC pulses) be accompanied by a "1" in digit 6, regardless of its contents during the preceding pulse period. Restated, each message must start with a timing digit of "1," and this digit must be alternately 1, 0, 1, 0, etc., for the entire message (and until the start of the next message).

4. The program must also supply a carrier frequency subharmonic. For a data rate of 1300 pps, this is accomplished by complementing the contents of digit position 5 of LR3 once at the start of each pulse period. For 1600 pps, complement digit 5 at alternate pulse periods.

5. The program must place the pulse codes and the timing and carrier generation digits in LR3 at precisely the data rate. In order to do this, the program is synchronized with an external timing pulse generator (TPG) which restarts the program at a frequency equal to the data rate. For this technique to be effective, however, the program must come to a halt before the TPG generates the next RESTART pulse. Hence the total execution time of the program must be less than one pulse period. (If the data rate is 1300 pps, the pulse period is 769 μsec; for 1600 pps, the period is 667 μsec.)

6. The first action taken by the program after it is restarted is to store new pulse, timing, and carrier information in LR3. This information must not be disturbed until the beginning of the next pulse period.

7. A sample outline of a program for 1600 pps transmission and an <u>even number</u> of pulses per message (including SYNC) is as follows:

Instruction	Function
A ad B ad C ra B sof 0	determine timing control code for next pulse period
ha -	wait for next timed RESTART pulse
st 51 (LR3) sof ( )	store in LR3 the pulse and timing control codes to be transmitted
Start - - ->	obtain code of next pulse to be transmitted; retain it in AC2 and AC3
tr A	(AC4 must contain a zero at this point)
B ha ( )	temporary storage for current timing control code (in digit positions 5 and 6)
C 0.01000	octal constant for generating next timing control code

For programming the reception of a message via a DDR, the rules are quite simple: Each time a pulse is received from the telephone line, the DDR places the complement of the corresponding 2-bit pulse code into digit positions 0 and 1 of LR4 (positions 2 through 15 will contain ONEs) and at the same time issues a RESTART pulse to the computer. RESTART pulses will occur at a frequency equal to the data rate of the incoming message. The program must come to a halt before the DDR generates the next RESTART pulse, hence the total execution time of the program must be less than 1 pulse period (refer to paragraph 4 above). (Note: LR4 is normally addressed 0-77 octal.)

A suggested outline of a program to receive messages via the DDR is as follows:

Instruction		
ha -	wait for next timed RESTART pulse	
sof 0 cs 77 (LR4) sof 1	obtain code of pulse just received	
tn A	neg if SYNC pulse	analyze code of received pulse
su B		
tn C	neg if ZERO pulse, pos if ONE pulse	
B 0.40000	octal constant for testing whether code is for ONE or ZERO pulse	

All transfer instructions (tr, tn, tro, tno, and to) reset LR3 and LR4 to negative zero. This resetting function is needed in connection with Charactron and Typotron displays, but it must be disabled when LR3 or LR4 are required in a program using the DDT or DDR. This is done by unplugging the single coax cable at C308 J3-6.

DISPLAY EQUIPMENT

Camera. The Fairchild Scope Recording Camera is normally fitted to a 12" display scope. The camera shutter is normally open. The execution of the instruction op 0 ("index camera") advances the film to the next unexposed frame, momentarily closing the camera shutter while the film is in transit. The op 0 instruction requires about 1.4 seconds. A "frame number" for each exposure is visible on the outside of the camera unit, and is photographically recorded on each frame of exposed film.

Everything displayed in the interval between two index operations will be recorded on a single frame of film. To photograph a series of programmed displays, then an initial indexing is required to bring into position an unexposed frame of film. After each display to be recorded on a single frame is completed, the camera must be indexed to a new frame. The camera must be indexed at least once after the last recorded display in a program in order to prevent displays by succeeding programs from being recorded on the same frame.

Charactron and Typotron. The Charactron is a special 19" scope which is equipped to display alphanumerical characters, selected other symbols, points, and vectors. It has facilities for dimming of characters, for manually selecting for display only information accompanied by certain identification codes, and for expansion (under manual control) of any part of the display. The Typotron is a second special 5" scope which can display alphanumerical characters and selected other symbols. The Typotron has the unique ability to retain a display for an indefinite period: any pattern once displayed on the Typotron phosphor mosaic is held without flicker until the whole surface is erased by the instruction op 2000 (octal). For illustrations of the characters possible on the Charactron, refer to the MIT Matrix Mod X in the appendix. For Typotron characters, see MIT Matrix Mod XI in the appendix. A detailed description of the controls and mounting console for the Charactron and Typotron can be found in Memorandum 6DR-127, AN/FSQ-7 Display Console Specifications.

The Charactron has a dual deflection system---slow (electromagnetic, with a setup time of about 40 μsec) and fast (electrostatic, with a setup time of about 20 μsec). The elctrostatic system provides limited deflection (5 bits each of y and x, corresponding to a 31 by 31 square array of contiguous symbols) and in the Charactron is intended to be superimposed on a central deflection set up by the electromagnetic circuits, which give full deflection within a 12" inscribed square.

The main (electromagnetic) deflection of the Charactron beam is fixed by the contents of LR1 and LR2 (normally assigned the octal address 0-61 and 0-62). Thirteen bits each of Y- and X-deflection values are inserted respectively as signed binary numbers in digit positions 0-12. To display a vector, one loads LR5 (normally at address 0-50 octal) with signed vector components: a sign and 6 numerical bits of y in digits 0-6, a sign and 6 numerical bits of x in digits 9-15. If a character rather than a vector is desired, however, vector components are superfluous. The Accumulator must hold a word containing certain control information:

digit 0:	"0" for no vector, "1" for vector
digit 1:	"0" for defocus (used with characters), "1" for focus (used with points and vectors)
digit 2:	"0" for bright, "1" for dim
digit 3:	"0" for Charactron, ("1" for Typotron)
digit 4:	"0" for normal operation, "1" to remove compensation for purposes of testing
digits 5-6:	feature selection code (explained below)
digits 7-15:	category selection code (explained below)

Digit positions 5-15 of the control word contain two identifying codes. A display of any symbol will not be visible unless the manual selection switches on the console corresponding to both these codes have been turned ON. Two independent sets of identifying codes and selection switches have been provided, called "feature selection" and "category selection." The feature is identified by a 2-bit code in AC 5 and 6, and the category by a 9-bit code in AC 7-15, according to the following scheme:

Feature	Feature code (binary)
A	00
B	01
C	10
D	11

Category	Category code (octal)
1	001
2	002
3	004
4	010
5	020
6	040
7	100
8	200
9	401
10	402
11	404
12	410
13	420
14	440
15	500
16	600

In normal usage, wherein a display "message" is composed of several rows of characters, all characters in a given message are given the same category, and different groups of characters are given different feature codes. By manipulating the switches on the display console, the operator can select the categories of messages he wishes to see, and which features or sets of characters he desires to see within each message.

It is possible to assign a particular display to as many as eight categories simultaneously, although it can have only one feature. The console switching is rather flexible: some features or categories may be forced (i.e., cannot be suppressed), and some can be displayed dim by manual option if switching is suitably wired. For the exact possibilities of category and feature switching and the associated programming details, the programmer should contact the Group 62 Display Section.

The instruction which gives the command to display is the Charactron-Typotron display instruction ch q. At the address q must be stored a word which gives in digits 10-15 the code for the character to be displayed, and in digits 0-9 the "format" position of the character. The 6-bit character code in digits 10-15 actually specifies the position which the desired character occupies in an 8 by 8 character matrix (illustrated in the drawing MIT Matrix Mod X in the appendix). The positive binary number in digits 10-12 selects the row (y-position) in the matrix; that in digits 13-15 selects the column (x-position). The signed binary number in digits 0-4 selects the y-position of the character in the 31 by 31 format array whose center was previously determined by the main Y- and X-deflection values in LR1 and LR2. Similarly the number in digits 5-9 selects the analogous x-position. Adjacent symbols in the format array are normally positioned at locations separated by 1/8 (binary 0.0010). Note that binary 0.0000 is equivalent to 1.1111 when giving the y and x coordinates in the format.

To summarize, at the time the ch instruction is given, the contents of certain registers is as follows:

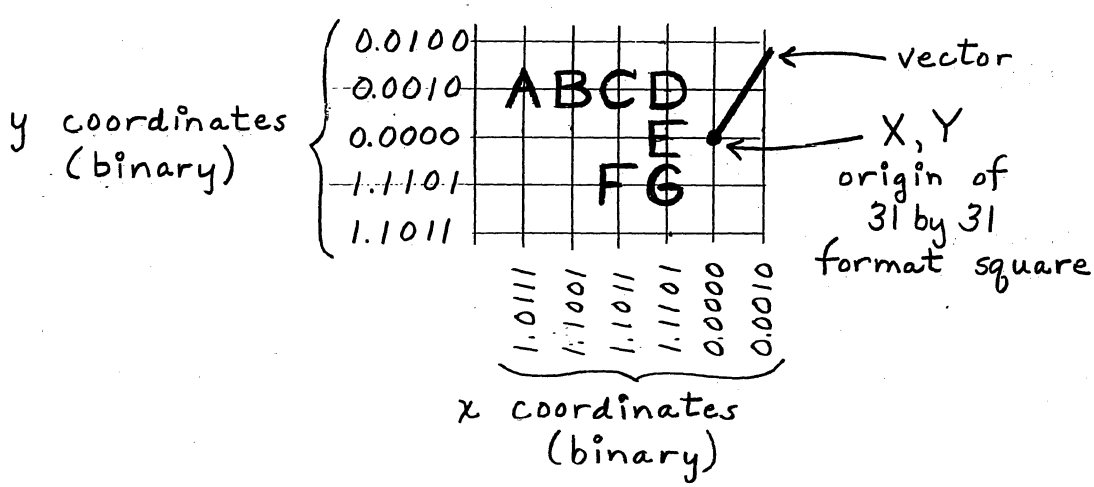
LR1	±Y (13 bits)			position of center of 31 by 31 format array
LR2	±X (13 bits)			
LR5	±y (7 bits)		±x (7 bits)	vector components (if necessary)
AC	(16 bits)			control word
q	±y (5 bits)	±x (5 bits)	C (6 bits)	format deflection; character code.

If the control information in the Accumulator is not to be changed (as is usually the case), the remaining characters of this "format" may be displayed by repeated ch instructions.

The execution of any transfer instruction (namely tr, tn, tro, tno, or to) clears the circuits which provide the magnetic deflection of the beam. The first ch instruction following such a transfer will activate the magnetic deflection circuits and set them up in accordance with the contents of LR1 and LR2. Only the first ch after any transfer instruction is involved in setting up the magnetic deflection, and the contents of LR1 and LR2 are used only during the execution of this first ch. It is therefore necessary to precede the first of a series of ch instructions (all of which utilize the same magnetic deflection of the beam) by a loading of LR1 and LR2 with the appropriate deflection values, and by a transfer instruction. The loading must follow the transfer instruction.

Here is an example of a Charactron display instruction cycle which will generate the display shown in the accompanying illustration:

```
ca   RC Y                                  }
sof  0                                     } Set up Y deflection
st   61  (LR1)                             }

sof  1                                     }
ca   RC X                                  } Set up X deflection
sof  0                                     }
st   62  (LR2)                             }

sof  1                                     }
ca   RC  0.001000  00  0110011             }
         \_____/       \_____/             }
            y             x                } Set up vector
sof  0                                     }
st   50  (LR5)                             }

sof  1                                     } Obtain control word (vector,
ca   RC  1.100 010 000 000 010             } focus, bright, Charactron,
                                           } feature B, category 2)

ch   RC  0.0000 00000 011011               } Display vector
         \____/ \____/ \_____/             }
           y      x    vector

ca   RC  0.000 011 000 000 010             } Obtain control word
                                           } (Character, defocus, bright,
                                           } Charactron, feature C,
                                           } category 2)

ch   RC  0.0010 10111 110 101 (A)          }
         \____/ \____/ \_____/             }
           y      x    character           }
ch   RC  0.0010 11001 011 110 (B)          }
ch   RC  0.0010 11011 101 110 (C)          } Display 8 characters at
ch   RC  0.0010 11101 110 001 (D)          } specified locations in
ch   RC  0.0000 11101 110 110 (E)          } 31 by 31 format array
ch   RC  1.1101 11011 010 110 (F)          }
ch   RC  1.1101 11101 101 000 (G)          }
```



The Typotron technique is like that for the Charactron but considerably simpler by certain omissions:

1. No magnetic deflection is used

2. No vectors can be displayed

3. The only control information required in AC is in digits 3 and 4:

   digit 3: "1" for Typotron display

   digit 4: "0" for normal display, "1" to remove compensation for the purpose of testing.

The characters possible on the Typotron are illustrated in MIT Matrix Mod XI, and the character codes are given in the chart Typotron Codes, MIT Matrix Mod XI, both in the appendix. It may prove useful to note that the character codes for capital letters and for numerical digits are identical for both Typotron and Charactron.

Memory address scope. The "memory address scope" or "memory scope" is a display which is limited to points in a 64 x 64 array, each position in the array corresponding to 1 of 4096 possible memory addresses. The purpose of the memory scope is to provide a visual means of observing some internal computer functions associated with memory. The memory scope does not exist as a physical unit, but can be observed by switching the 12 1/2" monitor scope unit to "memory display."

When operated in the "programmed mode," the memory scope is set up and intensified by the instruction md x. The point in the 64 x 64 array which is intensified is determined by the address part x. Refer to page 23 for diagram of the correspondence between points and selected addresses.

A second mode of operation of the memory scope is possible. This mode is called "Automatic Memory Display," and is achieved by running the computer with the "Suppress Operation Timing" switch ON. The computer will then refer to all core memory addresses in sequence. The memory scope automatically displays a pattern which corresponds to the binary contents of the memory digit plane selected by a 17-position "digit selector switch."

Plotting scope. The "plotting scope" is not a specific scope unit, but rather an oscilloscope output display which may be visible on several scope units. The purpose of the plotting scope is to enable the computer to present output information plotted on a point-by-point basis.

The plotting scope system now embraces two scope units, which are deflected in parallel and intensified simultaneously:

1) A 12 1/2" monitor mounted on the operating console,

2) A 12" scope semipermanently fitted with a Fairchild Scope Recording Camera which is operated under computer control (see Camera above).

A single point is displayed on the plotting scope by the ds instruction, each point requiring about 19 μsec. The contents of AC at the time the instruction ds x is executed determine the vertical position Y, while the contents of register x determine the horizontal position X. The Cartesian coordinates X and Y determine the position of the intensified point with respect to the origin at the center of the scope face. Positive values of X and Y represent distances to the right and upward respectively. The limits of the deflection correspond to the numbers $(-1 + 2^{-9})$ and $+1 - 2^{-9})$.

XD-1 Display Test Setup. Four fields of the magnetic drum memory have been wired as an output buffer. This arrangement makes it possible for MTC to simulate in a limited manner the XD-1 display drums. Either one TD and one RD field, or two DD fields, can be simulated. Fields 7, 8, 9, and 10 of drum memory have each been provided with an extra set of read-write heads, wired such that the four 16-bit fields appear as two 32-bit fields. The left and right halves of the 32-bit field designated "A" are made up of the drum memory fields 9 and 10 respectively. The left and right halves of field "B" are drum memory fields 7 and 8 respectively. Either field A or field B will always be selected and will transmit every 32-bit word which passes under the associated heads, entirely independently of any references made to drum memory by the computer. At the XD-1 display generation frames each 32-bit word will be accepted or rejected according to the plan described below.

When TD-RD simulation is desired, field A represents one field of the TD drum and field B represents one field of the RD drum. Field A is then considered by the XD-1 display equipment to consist of 256 8-word slots (groups of 8 physically adjacent registers), and field B of 2048 1-word slots. When a Start-TD pulse is generated in the XD-1 SDG frame, field A is selected, one drum revolution elapses, then, starting with drum register 761 decimal (1371 octal), every 13th slot is accepted. After 13 drum revolutions all 256 slots of field A have been accepted. Next a Start-RD pulse is generated to select field B, one drum revolution elapses, then again starting with drum register 1273 decimal (2371 octal), every 7th slot (1-word slot) is accepted. After 7 drum revolutions all 2048 slots of field B have been accepted. A Start-TD pulse is again generated and the entire display cycle repeats.

When DD simulation is desired, fields A and B alternately represent the DD drum. Both fields are considered by the XD-1 display equipment to consist of 2048 1-word slots each. When a Start-DD pulse is generated in the XD-1 DDG, the nonselected field becomes the selected one, 64 drum revolutions elapse, and then, starting with drum register 761 decimal (1371 octal), every 63rd slot is accepted. After 63 drum revolutions all 2048 slots of one field have been accepted. Another Start-DD pulse will be generated, and the entire cycle repeats.

The registers on the drum which are physically adjacent are not consecutively numbered, but are numbered as outlined on p. 30. It is up to the programmer to calculate the addresses of the drum registers which comprise any slot.

The meanings of the bits in TD, RD, and DD slots will be found in Memorandum 6M-2877, Specifications for the AN/FSQ-7 (XD-1) Display System.

PAPER TAPE EQUIPMENT

Tape punch. The paper-tape punch punches standard 7-hole Flexowriter tape at a maximum rate of 12 lines per second. It is actuated by the pr instruction, in accordance with the contents of AC 10-15. The 7th hole position is normally punched with each line of information, but if a "1" is inserted in digit 6 of the instruction pr n, the 7th hole will be omitted.

Tape readers. One input device for MTC is a high-speed paper tape reader (Ferranti Mark II), which reads standard 7-hole Flexowriter tape at a maximum rate of 200 lines per second. One line of tape is read in response to each ri instruction. (There are no restrictions on the minimum or maximum frequency of ri instructions in the program.) The reader will ignore and automatically skip over blank tape and over any punched information not accompanied by a punch in the 7th hole position. Whenever the high-speed reader is not in service, a slow-speed paper tape reader (Flexowriter FL) will be connected. The slow-speed reader performs in the same manner as the high-speed reader except that it operates at approximately 9 lines per second.

PRINTERS

Flexowriter. The Flexowriter printer is actuated by the pr instruction. The character printed or the machine function (such as carriage return) executed is specified by the contents of AC 10-15. (See "FL" Flexowriter Codes in Appendix.) The printer accommodates a maximum of 95 characters per line, and prints a maximum of 8 characters per second.

IBM Type 407 Accounting Machine. This machine is not a piece of MTC terminal equipment, but it can effectively be used as a delayed printer, printing from cards punched by the computer. For a description

and operating instructions for the 407, refer to Principles of Operation, Type 407 Accounting Machine. Both the machine and a copy of this manual are kept in the Card Preparation Room, C-168.

SECTION V. INPUT AND BASIC CONVERSION PROGRAMS

INTRODUCTION

This section deals specifically with punched paper tape as the input medium for MTC. The principles could apply equally well, however, to punched cards. The use of punched cards as input media for MTC is now in its early stages of development, but cards will eventually be utilized in a manner similar to punched tape. Section VI describes the plans already made for punched card input.

Certain of the programs and the paper tape structure treated in this section are designated "B" or "Type B" to indicate a distinction between these items and their counterparts described in earlier editions of this memorandum.

BRIEF DESCRIPTION OF THE INPUT PROCESS

The process of introducing into computer memory the binary words comprising a program is based on reading punched paper tape by an auxiliary program already in the computer. This latter program is termed an Input Program. The Input Program is set up manually in its binary form and is located in Panel Memory. Because of the relatively small number of registers contained in Panel Memory, however, the ability of the Input Program is limited to reading tape which carries the exact binary form of the words to be put into memory.

The translation of a program from a manuscript in coded notation to a binary tape suitable for reading by the simple Input Program is a complex problem. For all but very short tapes, the task is too complicated and too tedious for a human being to satisfactorily perform. But by means of a second auxiliary program termed a Conversion Program, the computer itself can be made to carry out the required translation!

The first step of the input process is the preparation by an operator of a form of paper tape known as "standard tape." Standard tape is produced on a Flexowriter unit similar in appearance to an ordinary typewriter. By means of the Flexowriter keyboard, the operator copies the programmer's manuscript of a program, producing simultaneously a typescript and a standard tape. The distinguishing feature of standard tape is that it is punched in an arbitrary binary code which has no direct relation to the binary words required by the Input Program.

The second step of the input process is the reading in to the computer of the Conversion Program. The Conversion Program must be on a tape in its exact binary form so that it can be read in by the Input Program.

The operation of the Conversion Program is the third step. The Conversion Program reads a standard tape, translates the information to the exact binary form of the words to be put in memory, and punches the words on a second tape.

The fourth step is the reading in to the computer of this translated or "converted" tape via the Input Program. Once a program has been produced in the form of a converted tape, it can be read into the computer by the Input Program alone, without further need for the Conversion Program.

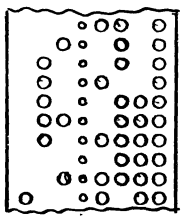
PUNCHED PAPER TAPE AND TAPE READERS

Inserting information into the computer is accomplished by means of electro-mechanical readers which are able to sense the patterns punched on tape. A binary number is transmitted to the computer by the tape reader for each row or "line" of digits on tape, the presence of a hole representing a binary "1" and the absence of a hole representing a binary "0".

Two kinds of punched paper tape have been mentioned: "standard" tape, prepared on a Flexo unit by an operator, and "converted" tape, produced on the computer's Flexo punch by the Conversion Program. While the two kinds of tape differ in the meanings of the punched configurations, they are otherwise physically similar. Hence a reader is able to read both kinds of tape.

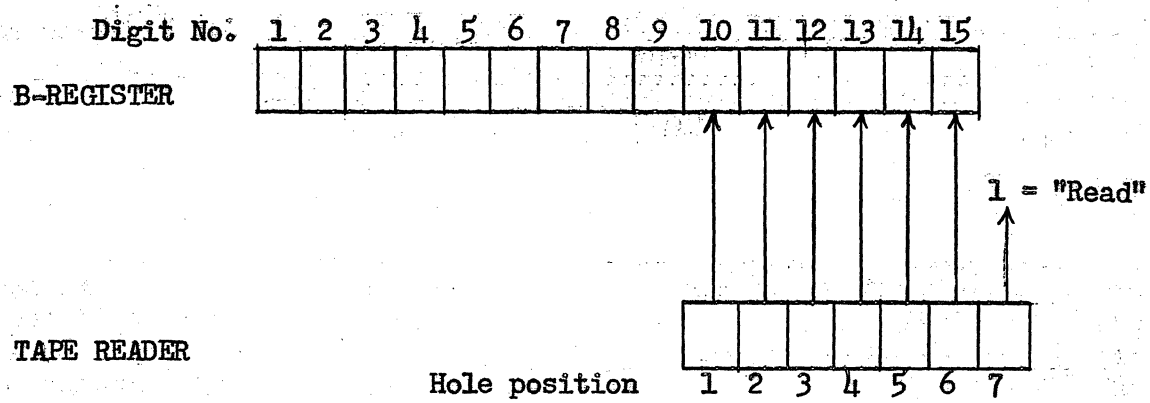
A portion of paper tape looks like this:

hole number 1 2 3 4 5 6 7



There are seven hole-positions to each line, and each position may or may not be punched. Hole positions 1 through 6 are used for punching information to be read into the computer. Hole position 7 is used as an indicator. To distinguish a line containing all zeros in hole positions 1 through 6 from blank tape (which is necessary for mechanical reasons), hole position 7 will always be punched if there is any information in the line. The readers are arranged to read only those lines which have the 7th hole position punched and hence to ignore blank tape.

The six-digit binary number corresponding to the punches in hole positions 1 through 6 is sent by the tape reader into the last six digit positions of the B-Register. This process is represented schematically thus:



Note that the 7th hole position is not used to send a digit to the B-Register, but is used to indicate to the reader whether the information contained in hole positions 1 through 6 should be sent.

The set of smaller holes between hole positions 3 and 4 is known as "feed holes." They are engaged by a drive sprocket in the punching equipment and in the mechanical reader for the purpose of advancing and positioning the tape. The feed holes are present on the entire length of the tape, and all other holes are punched in alignment with the feed holes.

The paper tape in current use is of a medium weight pink paper. Because the equipment for preparing paper tapes is designated by the trade name "Flexowriter" (abbreviated to "Flexo"), the paper tape is often referred to as "Flexo tape."

TAPE BASIC CONVERSION PROGRAM B

Function of a conversion program. The main function of a conversion program is to translate, or convert, a program from the programmer's coded notation into the 16-digit binary words with which the computer operates. In addition to punching these words on a "converted" tape, it must also provide the proper directions for the Input Program. Subject only to the limitations of time and memory, a conversion program can be written which will recognize any consistent and unambiguous set of conventions of coded notation. It is thus a completely flexible device which converts whatever notation has been chosen as convenient for programmers into whatever form is necessary for the computer. In particular, a comprehensive conversion program makes it possible for a programmer to select previously written and tested subroutines which are automatically called in from a file and assembled with a program during

the conversion process.

The aggregate of words, numbers, and symbols which the conversion program can handle is termed the "vocabulary." A single vocabulary expression may signify as little as part or all of a binary word, or as much as an entire subroutine, or even a whole program. Not all vocabulary expressions are converted and punched: some are merely directions for the conversion program.

Basic Conversion Program. The Basic Conversion Program is one with the minimum convenient vocabulary. A program (or data) to be converted by the Basic Conversion Program must be limited to the following conventions:

1. Instructions with "absolute" addresses (that is, addresses as octal or decimal numbers, not in some symbolic form).

2. Numbers expressed in any of these forms (numbers need not be confined to only one form):

   a. Octal constants.

   b. Decimal integers with factor of 2^{-15} implied.

   c. Decimal fractions.

Other necessary vocabulary terms include:

1. Memory location assignments (addresses designating where various parts of the program belong in memory), in "absolute" form.

2. A word defining whether the succeeding notations for addresses are in the octal or decimal number system.

3. The program serial number or tape number.

4. The address at which the program is to start.

A more detailed discussion of the Conversion Program vocabulary, together with examples, appears in the next part of this section.

The Basic Conversion Program punches on the tape the converted form of the above vocabulary expressions. In addition, it automatically provides appropriate blank tape, spaces the punched information to simplify visual checking, and provides information which makes possible a check on the accuracy of the punching, and later of the reading, of the tape. The resulting tape is termed "4-6-6" because of the physical arrangement of the punches. The structure of the converted, or 4-6-6, tape is presented in more detail starting on page 57.

Direct read-in conversion. The reader may have discerned that it is in theory not necessary to convert to "4-6-6" form every program which is to be put into the computer. The conversion program could be arranged to put the converted program directly in memory, instead of punching it out in the form of a converted tape. This process is called "direct read-in conversion." Direct read-in conversion is possible only with programs which do not occupy any of the same registers occupied by the conversion program.

A direct read-in conversion program does not exist for MTC, and probably will not exist in the foreseeable future. There are several reasons for converting programs to 4-6-6 form instead:

1. It is more convenient to treat all programs alike - that is, convert them - than it is to convert some and not others.

2. It is more economical of time and effort to read in a converted program via the Input Program, than to perform a direct read-in of a standard tape via a conversion program, since the latter alternative usually requires reading in of the conversion program also. Furthermore, a standard tape is almost invariably more than twice as long as its converted counterpart, hence is twice as bulky and requires twice the time to read it in.

3. Once a converted tape has been checked and found correct, further errors due to conversion are not possible.

The first conversion program. The Basic Conversion Program is punched on a tape in converted form in order that it may be read in by the Input Program. Since it is obvious that the conversion program could not have converted itself, how was the converted form prepared? The conversion program was written out with every instruction and number expressed as an octal constant. This was converted by a simple program which could convert only octal constants. The simple program in turn had been laboriously punched by hand.

PREPARATION OF STANDARD TAPE FOR CONVERSION

Preliminary preparations. A coded program is prepared for conversion by typing the program on an FL Flexowriter Unit which produces simultaneously a typewritten copy and a "standard" paper tape. Before commencing to type a program the following preparations are necessary:

1. Turn on the power switch at the right end of the keyboard.

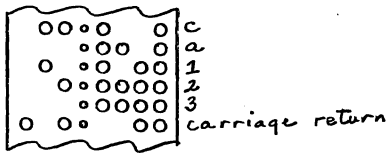
2. Check to see that the rotary switch behind the carriage is set to "Normal Print," that the left margin is at 14, and that tabular stops are positioned at 24, 34, 44, ... 164, 174.

3. Check that "Punch On" and "7th Hole" keys are depressed.

4. By depressing the "Feedout" key, feed out approximately six inches of blank tape.

5. Additional preparations, required for equipment at Barta Tape Room, are covered in Memorandum M-2461

General remarks concerning standard tape. Each key on the Flexo keyboard causes a unique 6-digit binary number (termed a "Flexo character" or "code") to be punched in one line on tape. (Six binary digits will permit 64 different combinations, of which the Flexo utilizes only 51. The binary code assigned to each key is arbitrarily set by the manufacturer.) An example of a single instruction punched in standard form is given below:



Each Flexo character to be interpreted by the Conversion Program must be accompanied by a punch in the 7th hole position. Blank tape, with or without 7th hole, may occur anywhere in any quantity.

Typographical errors. If the typist makes a mistake while punching a tape and detects the error immediately (before any more characters are punched on the tape), then the tape can be corrected by manually backing the tape one line in the punch. This places the incorrect character under the punching pins. If the typist then presses the "Code Delete" key, all seven holes will be punched across that line of tape (this is called a "nullify" character). This character will be ignored by the Conversion Program. Similarly, if several characters have been punched after the erroneous one, all of these characters could be punched over with the "nullify" character, starting with the first incorrect one. The typing and punching can then be resumed with the correct characters. If an error is undetected for a large number of lines, it is usually necessary to duplicate the tape up to the error, then punch the correct character, skip the error on the original tape, and continue to duplicate it.

Splices are usually unsuitable for the tape reader, and are generally inconvenient to make. Occasionally ingenious ways of correcting small mistakes can be found, but there are no standard and recommended ways available.

Heading. The heading must commence with the word TAPE. Either a tape number or the programmer's name or both may be included on the same line. However, they must be typed in that sequence and may not have any intervening tabs or carriage returns. The heading must be followed by at least one carriage return. Sample headings for main, modification, and parameter tapes are given below.

TAPE 452, P. Bagley

TAPE 452m2, P. Bagley

TAPE 452p6, P. Bagley

Address number base indicator. Before any addresses are typed, an address number base indicator, either OCTAL or DECIMAL, must be given, thus: OCTAL

Each section of a program which requires a base different from that of a preceding section must be preceded by the appropriate base indicator.

Since octal and decimal numbers are expressed in unique forms (see below), they may be distributed throughout a program without regard to the base indicators.

Location assignments. An address followed by a vertical bar (thus: 2-40|) specifies the address at which the next word is to be stored. Succeeding words will be stored in consecutive registers if no new location assignment is given. The bar may be followed by any combination of carriage and/or tabs.

A field number need be prefixed only for the first location assignment and for each assignment in a field different from the immediately preceding assignment.

Initial zeros may be omitted from field numbers and addresses.

Instructions. An instruction is written as two or three lower-case letters followed by an address with initial zeros suppressed.

The address part of any instruction may be prefixed by a field number for the programmer's convenience, but field numbers thus expressed are ignored by the conversion program. Initial zeros may be omitted from field numbers and addresses.

Numbers. Decimal fractions (magnitude less than 1) are written as +. or -. followed by exactly 4 decimal digits.

Decimal integers, less than 32768 ($=2^{+15}$), with an implicit factor of 2^{-15} are written as + or -, followed by as many digits as necessary, and no decimal point.

Octal numbers (magnitude less than 1) are written as 0. or 1. followed by exactly 5 octal digits. "1." indicates the start of a negative number, the remaining digits being the sevens complement digits of the desired number.

End of the program. The end of the program is indicated by the words "Start At" followed by the address of the first instruction to be executed.

Samples:

START AT 1-100 START AT 2306

If a starting address is not appropriate, a "dummy" starting address is ordinarily given in order to signify the end of the tape and stop the conversion process.

Terminal characters. Each heading, indicator, instruction, or number must be terminated by at least one carriage return or tab. (For a location assignment, the bar serves as the terminal character.) Additional carriage returns or tabs, however, will be ignored.

Disregarded characters. For the convenience of the typist, blank (000000), nullify (111111), space, back space, color change, comma, stop, and upper and lower case shifts are ignored by the conversion program.

Synonyms. Carriage return and tabs are treated as synonyms. The Flexowriter numeral and the lower case l are not treated as synonyms.

Page layout. Suggested page layouts for octal and decimal programs are given on the next page.

Detecting errors in standard tapes. It is well to note that there is no automatic check (corresponding to a "sum-check") on errors made in standard tapes. The only checks on the accuracy of standard tapes are:

1. Visually proofreading the "print," a typewritten copy of the information punched on the tape. The print is made by running the punched tape through the reader section of the Flexo unit. A simple rule applies to checking the print: provided that each line punched on tape

TAPE 45, Smith
OCTAL

2-3440|	cr2000	r120	ad	ra3444
	st3500	cr2035	su3542	su3511
	tn3440	su3542	tn3540	ad3501
	tr3441	ca3501	mh3566	st3511
2-3460|	tr3677	cs3724	su3725	ra3442
	ca3762	st3445	st3447	ca3662
	sm3667	st3400	mh3715	cr431
	st0	tr3440	mh3576	sr3000
2-3500|	tr3440			
2-3602|			0.00160	1.77757
	+25	+.9722	-.8764	+1
2-3660|	0.12224	0.14532	0.06352	0.66673
	1.25546	1.23562	1.77741	

START AT 2-3461

Tape 4031m2 Jones
decimal

1-0|	ca31	ad37	st	su122	mh122
	mh123	st59	ca31	su33	st33
	sof2	tro1776	0.17744	1.21233	+21
	-3442				
31|		+0	+0	+0	+1
	+6	-2			

start at 6

SUGGESTED LAYOUTS FOR PROGRAMS
PREPARED ON THE FLEXOWRITER

has the 7th hole position punched, and provided that the expression "START AT X" is followed by a carriage return, there are no mistakes possible which do not show on the "print."

2. Comparison on the "tape comparer" with a copy of the tape known to be correct.

Modifications and parameters. It is on occasion desirable to make changes or corrections in a tape which is already prepared. A tape containing the changes alone can be made and read into the computer after the main program since the latest word to be read into a given storage register supersedes the previous contents of that register.

A tape containing changes which correct mistakes in a program or which improve a program in some way is usually termed a "modification," or "mod." Ordinarily the converted form of a modification is appended to the converted form of the main tape by reproducing both the main and modification tapes on a single new tape. As many modifications as necessary may be used.

A tape containing sets of data or changes which lend variations to a program is a "parameter" tape. It is generally desirable to be able to select for read-in any one or more of a number of parameters, hence parameters are usually kept as separate tapes instead of being appended to main tapes.

The standard and converted forms of both modification and parameter tapes are prepared in the same manner as main tapes. They customarily bear the same identifying number as the main tapes with which they are associated, but are distinguished by an additional modification or parameter number (examples: "m6," "p1").

OPERATING THE TAPE BASIC CONVERSION PROGRAM B

Preparations.

1. Read in the Conversion Program via the 4-6-6 Input Program (refer to p. 60 for instructions on operating the Input Program).

2. Turn on the Flexo equipment if it is not already on.

3. See that there is sufficient tape in the punch unit. If necessary, feed out about 6 inches of blank tape by pressing the FEEDOUT button mounted on the Flexowriter table.

4. Put the tape to be converted in the reader and close the gate which holds the tape in place. When using the high speed tape reader, the 7th hole position on the tape must be toward the operator.

5. In toggle switch registers 0 and 1 put the binary values of the instruction sof 2 and tro 3000 (octal), respectively.

6. Press the START OVER button. If the conversion is completed correctly, 6 inches of blank tape will be automatically fed out of the punch and the program will stop.

Errors in conversion. If any of the following situations occur, the Conversion Program will stop with a "Programmed Alarm."

The meanings of the octal numbers appearing in AC to identify the error are as follows:

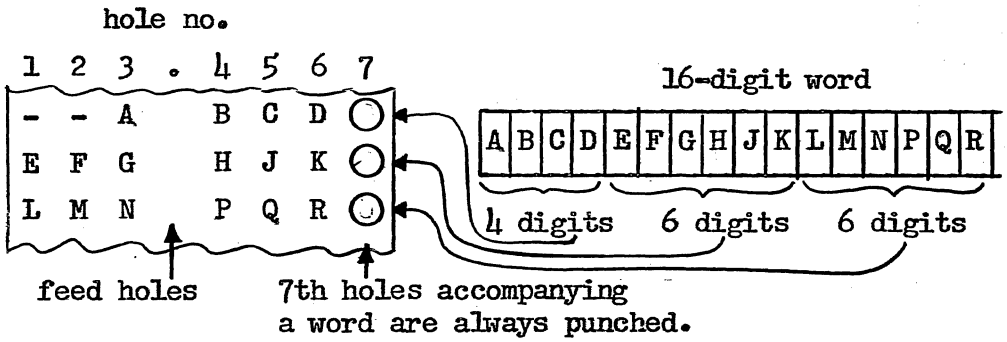
1.00001 Indeterminate error.
1.00002 Improper use of point.
0.00003 Improper character in decimal integer or fraction.
0.00004 Excessive magnitude of decimal integer.
0.00005 Improper number of digits in decimal fraction.
0.00006 No such instruction or indicator word.
0.00007 Improper character in address part of instruction.
0.00010 Improper number of digits in octal constant.
0.00011 Improper character in octal constant.
1.00012 Memory field not specified.
1.00013 Location assignment has become excessive.
1.00014 Improper character in start-over address.
1.00015 Decimal digits in octal field number of location assignment.
1.00016 Decimal digits in octal field number of start-over address.
1.00017 Excessive magnitude of field number of location assignment.
1.00020 Excessive magnitude of field number of start-over address.
1.00021 Decimal digits in address part of octal location assignment.
1.00022 Decimal digits in address part of octal start-over address.
0.00023 Decimal digits in address part of octal instruction.
0.00024 Excessive magnitude of address part of location assignment.
0.00025 Excessive magnitude of address part of start-over address.
0.00026 Excessive magnitude of address part of instruction.
1.00027 Heading improper or omitted.
1.00030 Number base indicator not given.
1.00031 Start-over address lacks a field number.

When an error code has a zero in the sign digit position, it indicates that the standard tape contains an error in a word that is later to be put into memory (that is, a word which is not a key word or other indication to the Conversion Program). The extended address to which this word is assigned is displayed in BR. When this type of error occurs, the extended address can simply be noted, and the Conversion Program started over at register 2-3001. Conversion will then continue, but all erroneous words thus passed over must be corrected by modifications at the end of the converted tape.

Conversely, a "1" in the sign digit position of an error code indicates a type of error which cannot be corrected at the end of the converted tape.

STRUCTURE OF CONVERTED (4-6-6) TAPE, TYPE B

Tape produced by the Tape Basic Conversion Program B is called "4-6-6." It carries the exact binary values of "words." Since a line of tape can store only six digits, three successive lines of tape are required to store an entire word. A word is punched on tape in the following fashion:



The name "4-6-6 tape" can be seen to originate from the fact that a 16-digit word is broken up into successive groups of 4, 6, and 6 digits. The tape reader reads the groups of digits in this order; namely, 4, 6, then 6.

The reading of 4-6-6 tape, reassembling the 16-digit words, and storing them in their proper locations in the computer, is accomplished by the "4-6-6 input program." This program is stored semi-permanently in the computer in registers 0-32 through 0-63, which group of registers is designated "Plugboard Storage." Pushing the button labeled "Start over at 40" (octal) on the computer control panel instructs the computer to start performing the 4-6-6 input program. The tape reader is under the control of the 4-6-6 input program, reading one line of tape for each ri instruction that the computer performs.

Normally the 16-digit words are read from the tape, assembled one at a time, and stored in consecutive memory registers in the computer. There are three circumstances, however, which require that the 4-6-6 input program be able to perform other functions:

1) At the beginning of the tape (and occasionally elsewhere) it is necessary to specify an address at which the input program is to start storing words.

2) At the end of a tape (and occasionally elsewhere) it is necessary to direct the computer to leave the 4-6-6 input program and to start taking instructions from a particular address in the main program. (This is called "changing control to the main program.")

3) A checking procedure known as "sum-check" is used to check the reliability of the tape punching and reading equipment. This is done by accumulating the arithmetic sum of all words read in since the previous check sum. Each word read in is added to the cumulative sum, any "overflow" (any part of the sum which is greater than unity) being neglected. This sum accumulated by the 4-6-6 input program is compared with a supposedly identical sum which is punched on the tape. If the sums do not agree, a mistake has been made and the Input Program stops with an "identity check" alarm.

Each of these three special situations is controlled by "key words" on the 4-6-6 tape.

The first of each group of three lines on tape has two digit positions which are not required by the 16-digit binary word. The second of these spare positions is not used, but the first position is occupied by a single digit called the "directive." If the directive is "0," the accompanying word is a key word--in reality an instruction which is to be performed by the Input Program. If the directive is a "1," the accompanying word is to be handled by the Input Program in accordance with the <u>most recent pair of key words</u>.

A block, or group of words to be stored (a "store block") begins with a pair of key words: <u>sof a</u> and <u>st x</u>, which designate that the first word of the block is to be stored in register <u>x</u> of field <u>a</u>, and that the block <u>is</u> a store block. In the absence of other key words, the Input Program will automatically store the succeeding words in successive registers.

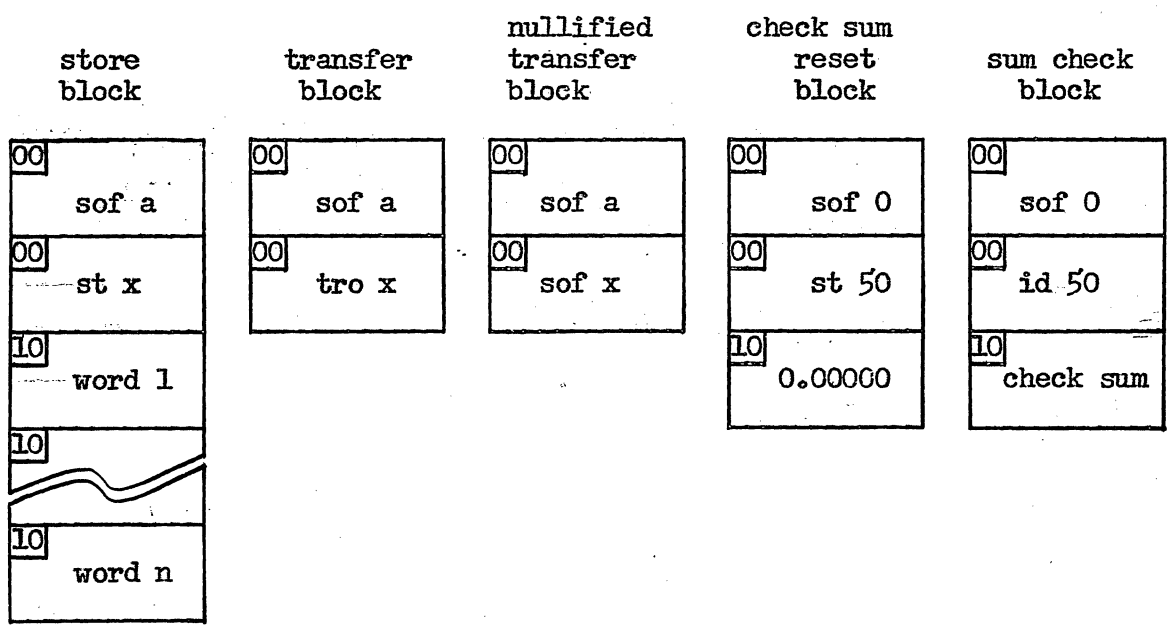
A group of words which stops the Input Program and starts the main program ("changes control to the main program") is a "transfer block." It consists of a pair of key words alone, <u>sof a</u> and <u>tro x</u>, which will direct the computer to take the next instruction from register <u>x</u> of field <u>a</u>, the address of the first instruction of the main program.

For a reason to be discussed later, it is sometimes desirable to "nullify" a transfer block: that is, to make it have no effect on the Input Program. This can be done by manually punching a single hole which will change the second key word tro x to sof x. The result is a "nullified transfer block."

A sum check block consists of two key words followed by a check sum. The key words are sof 0 and id 50 (octal). These words direct the Input Program to compare the check sum (#1) it finds on tape with the check sum (#2) the Input Program has been accumulating. Register 0-50 in panel memory is used as temporary storage for the check sum accumulated by the Input Program.

Due to space limitations, the Input Program is unable to set the check sum to zero, hence the need for a "check sum reset block." The reset block is in reality a store block which stores the quantity zero in the register assigned for the check sum. Following the reading in of a reset block, the Input Program will accumulate a sum of all words and their directives which are read from tape. After an appropriate number of words on tape, a sum check block will be found. When the sum check block is encountered by the Input Program, the sum accumulated thus far will be checked. If there is more data on the tape, a reset block should occur so that a new sum may be begun.

A graphic representation of the various kinds of blocks is given below:



The sequence of punched information on a 4-6-6 tape will be:

A. Tape number (without 7th hole)

B. Check sum reset block

C. One or more store blocks

D. Sum check block (customarily after every group of 128 words)

E. Transfer block

The sequence B, C, D, may be repeated as many times as desired. On tapes which have been "modified" by having an additional tape appended, it is customary to nullify all transfer blocks except the last, in order that control will be transferred to the main program only after all the tape has been read in.

A blank line of tape is left between blocks to facilitate visual checking. Ten blank lines are left after each check sum, these being points at which the tape may be positioned in the reader and the Input Program started over.

THE 4-6-6 INPUT PROGRAM B

The input program (shown on the next page) exists for the purpose of reading tape and transferring its contents to internal memory. The program fits entirely in 32 plugboard registers so that no toggle switches need be set up in order to operate the program.

The steps in operating the Input Program are:

1. Install plugboard containing Input Program.

2. See that no alarms are suppressed.

3. Place paper tape in reader, (if Hi-Speed Reader, 7th hole toward the front of the reader).

4. Press "Start over at 40."

5. If no alarms occur, program will stop when tape is read in. Main program can be started by pressing "Restart."

6. If alarm occurs due to a failure of the reader, tape may be manually drawn back until the 1" blank space which was most recently read is again directly under the head. Then the Input Program can be started over.

MTC INPUT PROGRAM B

Start→	40	ri 32	read in 3 lines of tape (= 1 word)
	41	ri 32	
	42	ri 22	
	43	tn 60	negative if this is a check sum or a word to be stored
	44	cr 2036	
	45	st 62	store key word temporarily
	46	tn 55	negative if key word is sof a or tro x
	47	tr 73 — 0	
LR5	50	( )	current sum of words read in
LR3	51	( )	not used by this program
	52	cr 2002	store key word sof a in final location
	53	st 61	
	54	tr 73	
	55	cr 36	negative if key word is sof a
	56	tn 52 — 11	
	57	ha 1	key word is tro x; stop
	60	cr 2036	execute pair of key words with check sum or word to be stored held in AC
LR1	61	(sof a)	
LR2	62	(st x, id 50, tro x)	
	63	sof 0	
	64	cr 20	retain in BR the word just treated
	65	ca 62	add one to st x in preparation for next word to be stored
	66	ad 57	
	67	st 62	
	70	cr 2020	add word and directive to check sum (directive "10" has value = 2^{-15})
	71	ad 57	
	72	to 73	
	73	ad 50	
	74	to 75	
	75	st 50	
	76	tr 40	
LR4	77	( )	not used by this program

NOTES

1. Read-in errors will result in identity check alarm. Defective tapes may be read in, however, by suppressing the alarm.
2. "Transfer block" on tape can be nullified by adding a hole which changes tro to sof.
3. Tapes can be read into any memory field.
4. Program fits entirely on plugboard; no switches need be set up in toggle-switch storage.
5. Unconditional stop occurs after read in; pushing RESTART will start the program which has just been read in.
6. A directive 00 indicates a key word; a directive 10 (value = 2^{-15}) indicates a word to be acted upon in accordance with most recent pair of key words.

4-9-55

SA-55786-4

SECTION VI. INPUT AND OUTPUT VIA PUNCHED CARDS

Punched cards can be utilized as input-output media in much the same manner as punched paper tape described in Section V. Most of the conventions and programs associated with the use of punched cards are, however, still in the formative stages. This section describes the card equipment, and those conventions and programs which already have been worked out.

Punched cards and card equipment. The MTC punched card equipment is designed to operate with IBM cards. An IBM card measures approximately 3.3 by 7.4 inches, and contains 80 columns and 12 rows of punching positions. The same physical card (with different identifying color stripes) is used for two or more fundamentally different coding conventions -- the "standard card" and the "binary card," analogous to "standard tape" and binary tape." The standard card is one in which the configuration of punches in a column represent one character in accordance with the IBM card code shown in the appendix. It may have variations, such as one form for symbolic addresses, another for absolute, etc. The binary card contains numbers or instructions in pure binary form, arranged in horizontal rows.

The machine on which standard cards are usually prepared is the 026 Printing Card Punch. A unique configuration of punches (in general, not more than two) are put in a column for each key depressed on the 026 keyboard. (Further details on operation of the punch are given in the IBM manual, Principles of Operation: Printing Card Punch, Type 026.) In contrast to the 026 punch, the card reader and punch connected to the computer are arranged to read and punch cards row by row. By proper computer programming, however, the computer reader and punch can deal with either standard or binary cards.

The computer card reader and punch have been arranged to deal with only 64 columns on a card, normally columns 17 through 80. Columns 1 through 16 can be used for identification information such as program and modification numbers, card sequence numbers, date, etc. This identification information is generally entered by the 026 punch on both standard and binary cards. A more complete description of the MTC card reader and punch is given under Card Equipment, p. 32f.

Preparation of standard cards for conversion. The normal procedure for putting a program into the computer via the medium of punched cards consists of manually preparing standard cards on the 026 Card Punch, conversion of these standard cards to binary cards with a card conversion program, and reading the binary cards into memory with the Card Input Program.

The conventions for standard cards have not all been decided upon, and the card conversion program is not yet written. The expressions for octal and decimal numbers and instructions will be the same as those

for standard tapes as given in Section V. The card conversion program will probably provide for the use of floating (symbolic) address notation similar to that already adopted for XD-1.

Unlike the procedure for nullifying errors on tapes, if an error is made in preparing a standard card, a new card must be punched. This is primarily because the column or position of holes on a card is as important as their grouping within a line or column.

Until a card conversion program is written, binary cards can be prepared by the following devious procedure: prepare the program on standard tape, convert it to binary tape, read in the binary tape, and punch out the program on binary cards with an existing utility program (Card Punchout Program) which punches on binary cards the contents of any selected region of memory.

Structure of binary cards. The binary card form is represented by the figure below.

	identi-fication	word position 0	word position 1	word position 2	word position 3	
12		word 40	word 41	word 42	word 43	DATA WORDS
11		word 36				
0		word 32				
1		word 28				
2		word 24				
ROW 3		word 20				
4		word 16				
5		word 12				
6		word 8				
7		word 4				
8		word 0	word 1	word 2	word 3	
9		key word 1	key word 2	word count	check sum	CONTROL WORDS
	columns 1-16	columns 17-32	columns 33-48	columns 49-64	columns 65-80	

The binary card can accommodate a maximum of 48 16-bit words, of which 4 are control words for the Card Input Program, and part or all of the remaining 44 are words to be stored in memory (data words, consisting of numbers and instructions).

The binary card illustrated above is termed a "data card." For a data card the control words are as follows:

1. Key word #1 is sof a, where a designates the field in which the data is to be stored;

2. Key word #2 is st x, where x is the address at which the first data word ("word 0") is to be stored (the remainder of the data words on the card will be stored at successive memory locations);

3. The word count is the number of data words punched on the card (which can be any number from 0 through 44, decimal);

4. The check sum is the complement of the sum of the first three control words and all the data words on the card.

A program in the form of a deck of binary cards will consist of any number of data cards followed by one "start card." The function of the start card is to instruct the Card Input Program to stop reading cards and prepare to start the program which has just been read in. The start card is a binary card containing only the 4 control words:

1. Key word #1 is sof a where a is the field containing the first instruction to be executed in the program;

2. Key word #2 is tro x where x is the memory address of the first instruction to be executed in the program;

3. The word count is 0 since there are no data words on the card;

4. The check sum is the complement of the sum of the other three control words.

Card input program. The main function of the Card Input Program is to read the 16-bit words from binary cards and store them in their proper locations in memory.

For each data card to be read in, the Card Input Program utilizes the 2 key words to determine the memory location for the first data word on the card. It reads the number of data words specified by the word count and stores them in successive memory locations. Finally it checks to see that the sum of the 4 control words and all the data

words is exactly 0. (The check sum value was calculated by the card conversion program so that it is the complement of the sum of all the other words on the card, which is the same as saying that it has been adjusted to make the sum of itself and all the other words be equal to 0.)

When the start card is read, the Card Input Program checks that the sum of the control words is 0, and then it stops. The two key words on the start card, which indicate the starting address of the program stored on the cards, are stored in the input program so that when the RESTART button is pushed, the program just read in will be started.

Achieving printed output via punched cards. The printing rate of the Flexowriter is approximately 8 characters per second. For programs requiring a large amount of printed output, it is desirable to conserve time by resorting to a faster method of output. This technique, termed "delayed printing," consists of having the program punch the output data on IBM cards in standard IBM card code, and later running the deck through an IBM Tabulator or Accounting Machine (not a part of MTC). Cards can be punched by the computer at a rate equivalent to about 100 characters per second, providing an effective reduction of printing time required by the Flexowriter by a factor of about 12.5.

SECTION VII. BRIEF OPERATING GUIDE FOR MTC

Operating controls. Nearly all the computer controls are on the console desk. The controls important to the operation of normal (as distinguished from computer test) programs are the following:

1. Alarm indicator lights(Parity, Inactivity, Identity Check, Programmed, Overflow, Card Machine, Drum Timing, and Fuse). Whenever the computer stops, at least one of these indicators will be on, provided they are not suppressed, to indicate the reason for the stop. The Inactivity Alarm simply means that the computer has stopped, so that in the absence of other indications the computer has stopped "normally" --- because of a "halt" instruction or a manual stop.

2. Alarm suppression switches are associated with each of the alarm indicators above. When an alarm is "suppressed" the corresponding condition which ordinarily initiates an alarm and stops the computer is ignored.

3. SUPPRESS OPERATION TIMING Switch is primarily for memory testing and should be OFF when a program is being run.

4. The pushbuttons START OVER (at 0) and START OVER at 40 (READ IN) cause control and arithmetic element to be cleared, and direct the computer to take its next instruction from register 0 or 40 (octal) of Panel Memory. (A duplicate of the START OVER at 40 button is mounted on the Ferranti Reader for the convenience of the operator.)

5. The RESTART pushbutton causes the computer to continue in normal fashion from the point at which it was most recently stopped. If the computer has stopped in an alarm, the "Clear Alarms" button must be pushed before the computer can be restarted.

6. The STOP pushbutton will cause the computer to stop at the end of the next "half-instruction," that is, at the end of the next "program timing" or "operation timing" cycle.

7. When the HALF INSTRUCTION switch is ON, the computer will stop at the end of every half-instruction. For each push of the RESTART button when the half-instruction switch is ON, the computer will perform a half instruction and then stop.

8. AUTOMATIC START OVER Switches. This switch is normally OFF. When this switch is ON, the computer is automatically started over at register 0-0 at a rate determined by the timing-pulse generator in rack location C814.

9. AUTOMATIC RESTART. When this switch is ON, it performs the same function as RESTART above at a rate determined by the frequency of a timing pulse generator. This feature is provided for the operation of special programs which must be synchronized to a timing source external to the computer. Care must be taken to see that the computer is not running at the instant each automatic restart pulse is introduced.

Flip-flop indicator lights. The contents of all flip-flop circuits in the computer are displayed on the Flip-Flop Indicator Panel, where red lamps represent "1's" and the white lamps represent "0's". Of chief interest to operators are the indicators for the following registers:

Partial Sum (Accumulator)
B-Register
A-Register
Control Switch
Flip-Flop "FX"
Program Field Register
Program Counter
Operation Field Register
Group and Field Switch
Core Address Register

Because the Program Counter is indexed immediately after it has been used to select the address of an instruction to be read from memory, the contents of the Program Counter at the end of any half-instruction will appear to be one greater than the address of the instruction which is being executed.

The contents of the Program Field Register and the Program Counter (less one) together indicate the "extended address" of the instruction which is currently being performed.

The contents of the Operation Field Register indicates the memory field which is currently selected ---that is, the field of memory to which the address parts of instructions will refer.

The contents of the Group and Field Switch and the Core Address Register indicate the extended address of the register currently selected (regardless of whether it is in panel, core, or drum memories, and whether it is a program or operation timing cycle).

At the end of any program timing cycle (FX holding a "0"), the A-Register contains the instruction which is about to be executed. At the end of any operation timing cycle (FX holding a "1"), the Control Switch and the Core Address Register together hold the instruction which was just completed.

Operating a program. After a correctly prepared program is read in, pushing the RESTART button will start the computer performing the program. (Refer to instructions on operation the 4-6-6 Input Program, p. 60.)

To start a program at any time other than immediately after read in, set up the following pair of instructions in registers 0 and 1 of toggle-switch storage, then push START OVER (at 0):

0-0 sof a
0-1 tro x

a and x are the field number and address, respectively, of the first instruction to be performed in the program.

All of the terminal equipment (scopes, punch, etc.) which is used by a program must be turned on prior to starting the program.

Manual intervention. On various occasions, such as when debugging a program, or when trouble is encountered with either the program or the computer, it is necessary for the operator to exercise some measure of control over what the computer does. Such action on the part of an operator goes by the general term of "manual intervention". Descriptions of the most common manual intervention actions are given below.

1. To inspect the contents of a single register, execute the following program:

```
0-0  sof a
0-1  ca x
0-2  ha -
```

The contents of register a-x will appear in AC.

2. To inspect the contents of a group of consecutive registers, turn on the HALF INSTRUCTION switch ON and then execute the following instructions. (push START OVER once and RESTART three times):

```
0-0  sof a
0-1  tro x
```

Now turn the SUPPRESS OPERATION TIMING switch ON. The contents of register a-x is now in AR. Each time RESTART is pushed, the contents of the next successive register will appear in AR. At any time, the contents of the Program Field Register and contents of the Program Counter less one will be the address of the register whose contents appears in AR.

3. To start the program at an arbitrary point, execute the following instructions:

```
0-0  sof a
0-1  tro x
```

4. To restart the program after an alarm, it is necessary to press CLEAR ALARMS, and then RESTART. Note that if the program stopped in an identity check alarm, the program will start with the second instruction after the id instruction which generated the alarm.

5. To change the contents of a register, execute the following program:

```
0-0  ca  4
0-1  sof a
0-2  st  x
0-3  ha
0-4  (word to be stored)
```

Setting up a program in toggle-switch storage. A short program (no more than 32 registers may be set up in its binary form in toggle-switch storage. Successive 16-bit words are set up in successive rows of toggle switches starting with row (or register) 0, each switch which is ON (up) corresponding to a binary "1" and each OFF switch to a binary "0." The rightmost (17th) switch in each register of switches is normally OFF. If this switch is on, Live Register #1 (LR#1) is substituted for the accompanying group of 16 switches. In this way LR1 can be substituted for any one or more toggle-switch registers.

It should be obvious that the program cannot alter the contents of a toggle-switch register.

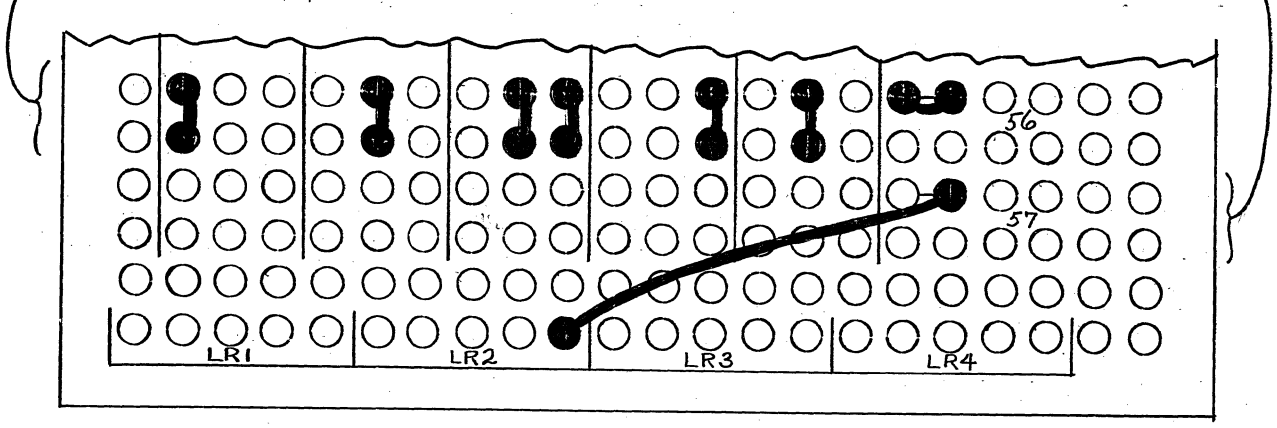
Setting up a program in plugboard storage. A program of no more than 32 registers can be set up on an IBM plugboard in a manner similar to putting a program in toggle-switch storage. On the plugboard, short wire jumpers take the place of turning switches on. Each address location used in the plugboard must contain a jumper which specified that the associated contains a word wired by jumpers or which of five Live Registers is to be substituted for that plugboard register.

It is possible to connect both a Live Register and a plugboard register to the same memory address. This requires a special jumper with three plugs.

A sample pair of properly wired plugboard registers are shown in the illustration below:

Register 56 (octal) contains the binary form of ca 2312 (octal).

LR2 has been substituted for the plugboard register 57 (octal).



Card machine controls. To make the card machine, containing both the card reader and punch, ready for operation:

1. Turn on the main power switch at the right end of the unit.

2. Set the six toggle switches on the front panel as follows:

Label	No. of switches	Setting
Reproduce	3	ON
Sel. Rep. and G. P. Comp.	1	OFF
Card "X" Punched Detail or Master	2	MASTER

3. Install a properly wired plugboard in the front of the unit. A permanently wired board is available for normal use. The programmer may, however, wire a board for any special application, in accordance with the instructions below.

4. Put blank cards in the punch (right-hand) hopper, face down and nine edge to the right.

5. Put a deck of prepared cards to be read or blank cards in the reader (left-hand) hopper, face down and nine edge to the right.

6. Hold down the start key (the right-hand lever of a pair of black levers) until the red "ready light" is turned on. The following conditions must all be satisfied before the ready light can be turned on:

   a. Cards are in both feed hoppers.
   b. A card is in position to be read.
   c. A card is in position to be punched.
   d. Neither card stacker is full.

   The computer cannot control the card machine unless the ready light is on.

Whenever card reading or punching stops, the last two cards read or punched will not have been delivered to the stackers. To run out these remaining cards, hold down the start key for at least two card cycles (roughly 2 seconds).

The card machine can be stopped manually at any time by pressing the stop key. The computer cannot regain control of the machine, however, until the ready light has been turned on again.

The figure on the next page shows the front view of the card machine control panel. The rules governing the wiring of the panel for use of the card machine under computer control are:

1. Nine jumpers must be provided to connect the control switch hubs to their associated computer lines. That is, in the section marked "Control Switches", each point on the first row must be connected to the point immediately below it in the second row.

2. Up to 64 "Read Brushes" (numbered to correspond with the columns on a card) can be connected to the 4 sets of "Read Gates" (numbered to correspond with the digit positions of a 16-bit computer word).

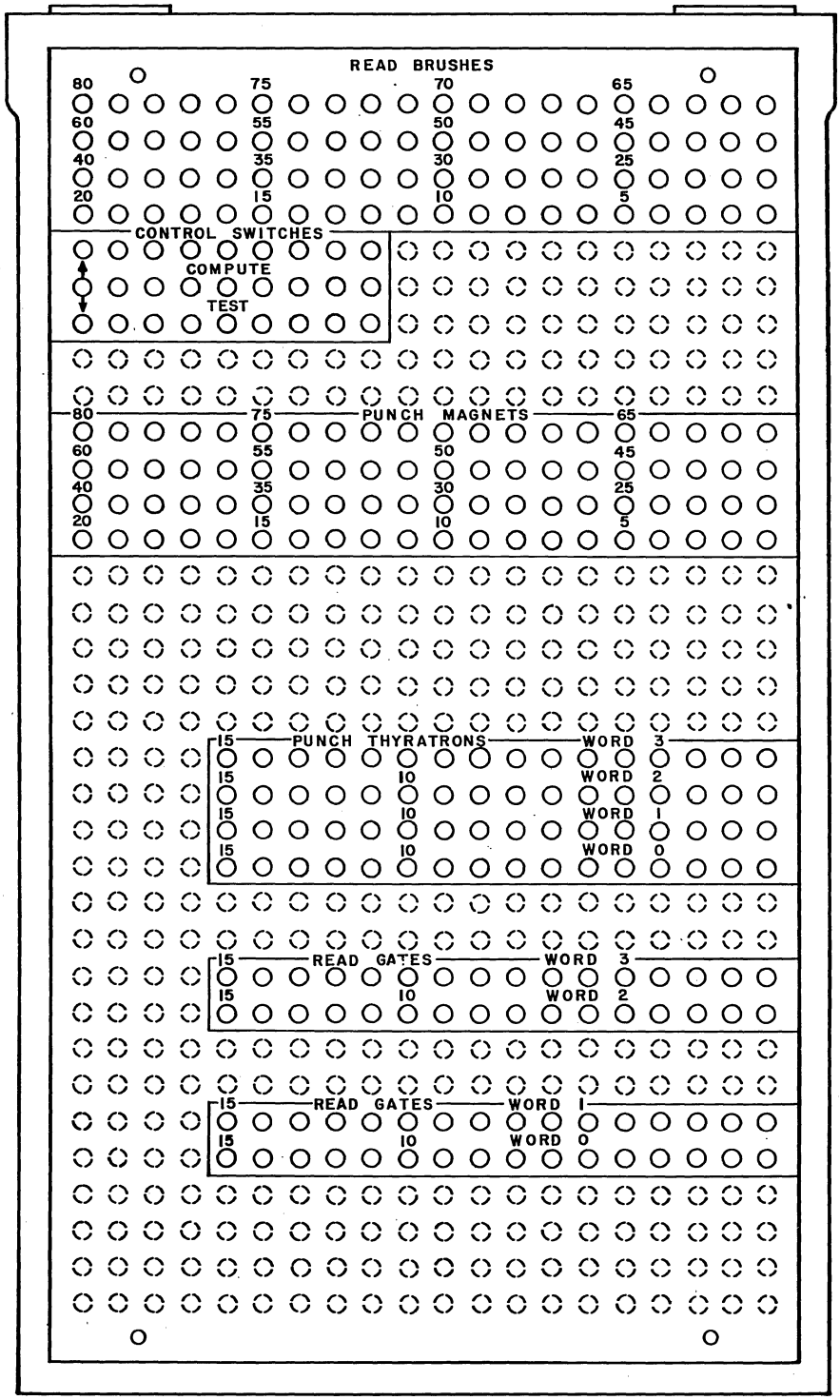
3. The 4 sets of "Punch Thyratrons" (numbered to correspond to the digit positions of a 16-bit computer word) can be connected to up to 64 "Punch Magnets" (numbered to correspond with the columns of a card).

Signed: Philip R. Bagley
Philip R. Bagley

PRB/rb

Drawings:

D-47039-2 (A-reduction), following page 9
B-62366 (A-reduction), following page 29
SA-55786-4, following page 60
B-62355 (A-reduction), following page 69
SA-56946-4, appendix
SA-56942-2, appendix
SA-56943-3, appendix
SA-62509, appendix
SA-62368, appendix
A-60383, appendix
SA-62367, appendix
A-60579, appendix
SA-57300-1, appendix



B-62355

FRONT VIEW

MTC CARD MACHINE
CONTROL PANEL

MTC INSTRUCTION CODE

Abbr		Name	Decimal	Binary	Notes
ha	-	halt	0	00000	
mh	x	multiply	1	00001	
ds	x	display	2	00010	
id	x	identity check	3	00011	
st	x	store	4	00100	
ra	x	replace address	6	00110	
rf	x	return from	7	00111	
ca	x	clear and add	8	01000	
ad	x	add	9	01001	
cs	x	clear and subtract	10	01010	
su	x	subtract	11	01011	
et	x	extract	12	01100	
ch	x	charactron display	14	01110	
sm	x	subtract magnitudes	15	01111	
tr	x	transfer	16	10000	
tro	x	transfer out	17	10001	
tn	x	transfer on negative	18	10010	
tno	x	transfer on negative out	19	10011	
md	x	memory address display	20	10100	
sof	a	select operation field	21	10101	A
to	x	transfer on overflow	22	10110	
cr	n	cycle right	24	11000	B C D
sr	n	shift right	25	11001	B C
pr	n	print/punch	28	11100	B E F
ri	n	read in	30	11110	B
op	k	operate	31	11111	G H

Notes: Variations of certain instructions are provided by digits 5, 6 and 7:

A. If digit 5 is a "1", give a programmed alarm.
B. If digit 5 is a "1", clear BR after shifting or cycling.
C. If digit 6 is a "1", roundoff AC & BR to 15 numerical digits in AC.
D. If digit 7 is a "0", perform "regular cycle" with all 32 bits in AC & BR; if digit 7 is a "1", perform "short cycle", leaving ACO undisturbed.
E. If digit 6 is a "1", suppress punching of the 7th hole position.
F. If digit 7 is a "0", actuate the punch; if a "1", actuate the printer.
GH. If the digit combination GH is: "00", index the camera; if "01", read a word from a card; if "10", erase Typotron; if "11", punch a word on a card.

SA-50940-4

3-22-55

SA-56946-4

"FL" FLEXOWRITER CODE
Alphanumerical Sequence

Lower Case	Upper Case	Character 123456	Octal Value	Decimal Value	Lower Case	Upper Case	Character 123456	Octal Value	Decimal Value
a	A	000110	6	(6)	0	°	111110	76	(62)
b	B	110010	62	(50)	1	¹	010101	25	(21)
c	C	011100	34	(28)	2	²	001111	17	(15)
d	D	010010	22	(18)	3	³	000111	7	(7)
e	E	000010	2	(2)	4	⁴	001011	13	(11)
f	F	011010	32	(26)	5	⁵	010011	23	(19)
g	G	110100	64	(52)	6	⁶	011011	33	(27)
h	H	101000	50	(40)	7	⁷	010111	27	(23)
i	I	001100	14	(12)	8	⁸	000011	3	(3)
j	J	010110	26	(22)	9	⁹	110110	66	(54)
k	K	011110	36	(30)	|	_	000101	5	(5)
l	L	100100	44	(36)	space bar		001000	10	(8)
m	M	111000	70	(56)	=	:	001001	11	(9)
n	N	011000	30	(24)	+	/	001101	15	(13)
o	O	110000	60	(48)	color change		010000	20	(16)
p	P	101100	54	(44)	.	)	010001	21	(17)
q	Q	101110	56	(46)	,	(	011001	31	(25)
r	R	010100	24	(20)	-	~	011101	35	(29)
s	S	001010	12	(10)	back space		100011	43	(35)
t	T	100000	40	(32)	tabulation		100101	45	(37)
u	U	001110	16	(14)	carr. return		101001	51	(41)
v	V	111100	74	(60)	stop		110001	61	(49)
w	W	100110	46	(38)	upper case		111001	71	(57)
x	X	111010	72	(58)	lower case		111101	75	(61)
y	Y	101010	52	(42)	nullify		111111	77	(63)
z	Z	100010	42	(34)					

SA-56942-2

3-22-55

SA-56942-2

"FL" FLEXOWRITER CODE
Binary Numerical Sequence

Octal Value	Decimal Value	Character 123456	Lower Case	Upper Case	Octal Value	Decimal Value	Character 123456	Lower Case	Upper Case
0	(0)	000000	not used		40	(32)	100000	t	T
1	(1)	000001	not used		41	(33)	100001	not used	
2	(2)	000010	e	E	42	(34)	100010	z	Z
3	(3)	000011	8	8	43	(35)	100011	back space	
4	(4)	000100	not used		44	(36)	100100	l	L
5	(5)	000101	|	_	45	(37)	100101	tabulation	
6	(6)	000110	a	A	46	(38)	100110	w	W
7	(7)	000111	3	3	47	(39)	100111	not used	
10	(8)	001000	space bar		50	(40)	101000	h	H
11	(9)	001001	=	:	51	(41)	101001	carr. return	
12	(10)	001010	s	S	52	(42)	101010	y	Y
13	(11)	001011	4	4	53	(43)	101011	not used	
14	(12)	001100	i	I	54	(44)	101100	p	P
15	(13)	001101	+	/	55	(45)	101101	not used	
16	(14)	001110	u	U	56	(46)	101110	q	Q
17	(15)	001111	2	2	57	(47)	101111	not used	
20	(16)	010000	color change		60	(48)	110000	o	O
21	(17)	010001	.	)	61	(49)	110001	stop	
22	(18)	010010	d	D	62	(50)	110010	b	B
23	(19)	010011	5	5	63	(51)	110011	not used	
24	(20)	010100	r	R	64	(52)	110100	g	G
25	(21)	010101	1	1	65	(53)	110101	not used	
26	(22)	010110	j	J	66	(54)	110110	9	9
27	(23)	010111	7	7	67	(55)	110111	not used	
30	(24)	011000	n	N	70	(56)	111000	m	M
31	(25)	011001	,	(	71	(57)	111001	upper case	
32	(26)	011010	f	F	72	(58)	111010	x	X
33	(27)	011011	6	6	73	(59)	111011	not used	
34	(28)	011100	c	C	74	(60)	111100	v	V
35	(29)	011101	-	-	75	(61)	111101	lower case	
36	(30)	011110	k	K	76	(62)	111110	0	0
37	(31)	011111	not used		77	(63)	111111	nullify	

3-22-55

SA-56943-3

SA-56943-3

SA-62509

CHARACTERS

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	+	-	0	1	2	3	4	5	6	7	8	9	/	+	-	.	¤	$	*	,	%
12	12	12	12	12	12	12	12	12	12																		12															12	12				
11										11	11	11	11	11	11	11	11	11										11																11	11			11
0																			0	0	0	0	0	0	0	0			0													0					0	0	
1	1									1																				1												1							
2		2									2								2												2													3	3		3		3	
3			3									3								3												3												4		4		4		4
4				4									4								4												4																
5					5									5								5												5															
6						6									6								6												6														
7							7									7								7												7													7
8								8									8								8												8					8	8	8	8	8	8	8	8
9									9									9								9												9											
(026 CARD PUNCH)																											&																@						P
(552 INTERPRETER)																																											3	4	C	D	L	M	T	U

The 718 Line Printer and the 407 Accounting Machine will print all the characters shown on the top line of this chart.
The 026 Card Punch currently makes the substitutions shown on the line labeled (026 CARD PUNCH).
The 552 Interpreter makes the substitutions shown on the line labeled (552 INTERPRETER).

IBM CARD PUNCHING CODE

(HOLLERITH CODE)

4-23-55

SA-62509

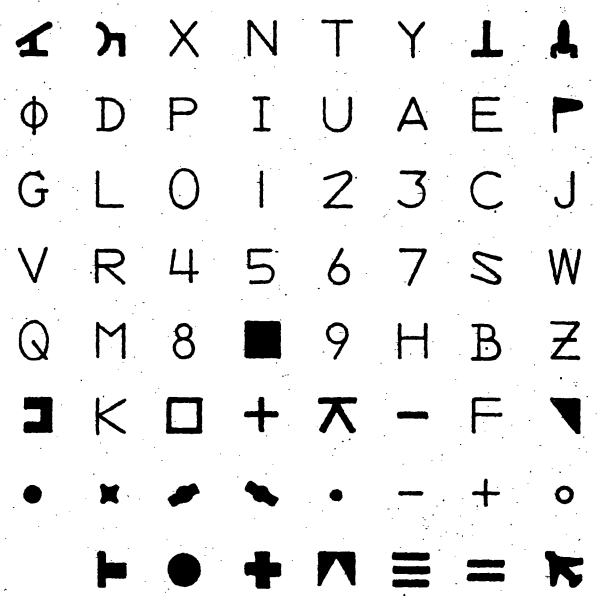
CHARACTRON CHARACTER CODES
MIT MATRIX MOD. X
Alphanumerical Sequence

Character	Octal Code	Decimal Value	Character	Octal Code	Decimal Value	Character	Octal Code	Decimal Value
Blank	00	(0)	T	74	(60)	⊥	76	(62)
Point	33	(27)	U	64	(52)	⅄	77	(63)
Vector	33	(27)	V	40	(32)	▶	67	(55)
A	65	(53)	W	47	(39)	⅄	24	(20)
B	36	(30)	X	72	(58)	⅄	07	(7)
C	56	(46)	Y	75	(61)	✚	03	(3)
D	61	(49)	Z	37	(31)	+	23	(19)
E	66	(54)	ϕ	60	(48)	−	25	(21)
F	26	(22)	∘	14	(12)	=	06	(6)
G	50	(40)	+	16	(14)	≡	05	(5)
H	35	(29)	−	15	(13)	⊐	20	(16)
I	63	(51)	0	52	(42)	⊢	01	(1)
J	57	(47)	1	53	(43)	◥	27	(23)
K	21	(17)	2	54	(44)	⋀	04	(4)
L	51	(41)	3	55	(45)	□	22	(18)
M	31	(25)	4	42	(34)	■	33	(27)
N	73	(59)	5	43	(35)	●	02	(2)
O	52	(42)	6	44	(36)	∘	17	(15)
P	62	(50)	7	45	(37)	•	10	(8)
Q	30	(24)	8	32	(26)	⊠	11	(9)
R	41	(33)	9	34	(28)	⊘	12	(10)
S	46	(38)	⅄	70	(56)	⊘	13	(11)
			⅄	71	(57)			

SA-62368

4-8-55
SA-62368

A-60383



NOTE A —

RATIO OF ℄ ℄ CHARACTER SPACING TO UPPER CASE CHARACTER HEIGHT IS 2.5 TO 1.

NOTE B —

ASPECT RATIO OF UPPER CASE "X" IS '4 TO 3.

M.I.T. MATRIX MOD X

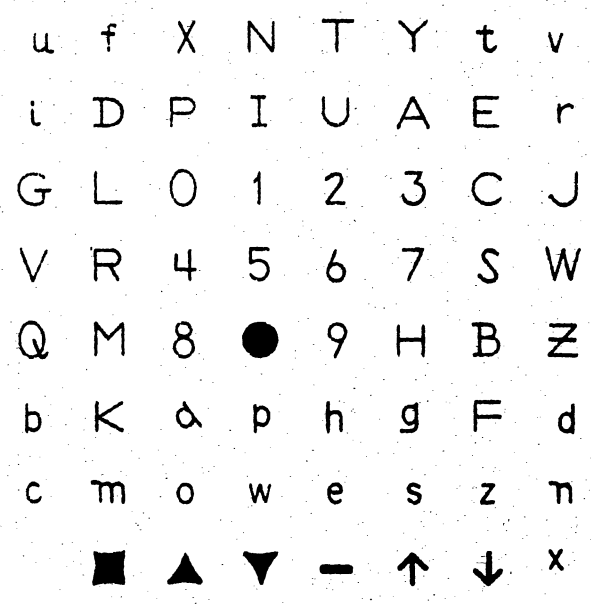
TYPOTRON CHARACTER CODES
MIT MATRIX MOD. XI
Alphanumerical Sequence

Character	Octal Code	Decimal Value	Character	Octal Code	Decimal Value	Character	Octal Code	Decimal Value
Blank	00	(0)	V	40	(32)	v	77	(63)
A	65	(53)	W	47	(39)	w	13	(11)
B	36	(30)	X	72	(58)	x	07	(7)
C	56	(46)	Y	75	(61)	z	16	(14)
D	61	(49)	Z	37	(31)	0	52	(42)
E	66	(54)	a	22	(18)	1	53	(43)
F	26	(22)	b	20	(16)	2	54	(44)
G	50	(40)	c	10	(8)	3	55	(45)
H	35	(29)	d	27	(23)	4	42	(34)
I	63	(51)	e	14	(12)	5	43	(35)
J	57	(47)	f	71	(57)	6	44	(36)
K	21	(17)	g	25	(21)	7	45	(37)
L	51	(41)	h	24	(20)	8	32	(26)
M	31	(25)	i	60	(48)	9	34	(28)
N	73	(59)	m	11	(9)	●	33	(27)
O	52	(42)	n	17	(15)	✖	01	(1)
P	62	(50)	o	12	(10)	▲	02	(2)
Q	30	(24)	p	23	(19)	▼	03	(3)
R	41	(33)	r	67	(55)	–	04	(4)
S	46	(38)	s	15	(13)	↑	05	(5)
T	74	(60)	t	76	(62)	↓	06	(6)
U	64	(52)	u	70	(56)			

4-8-55

SA-62367

A-60579



NOTE A —

RATIO OF ℄/℄ CHARACTER SPACING TO UPPER-CASE CHARACTER HEIGHT IS 2.5 TO 1.

NOTE B —

ASPECT RATIO OF UPPER CASE "X" IS 4 TO 3.

M.I.T. MATRIX MOD XI

VOCABULARY

MTC TAPE BASIC CONVERSION PROGRAM B

Tape numbers (samples for main, modification, and parameter tapes):

TAPE 452 TAPE 106m2 TAPE 93p6

Address number base indicators (a base indicator applies to all succeeding addresses until next base indicator is encountered):

OCTAL DECIMAL

Memory Location Assignments (samples):

40| 1-40| 3-2021|

Instructions (samples):

mh72 tro 2-2036 ca

Octal constants (samples):

0.12345 1.65432

Decimal fractions (samples):

+.9987 -.1234

Decimal integers (samples):

+12 -32767

End of program indicator (sample):

START AT 2-100

Notes:

1. An address x may or may not be preceded by a field number, thus: 2-x. A field number must be included in at least the first memory location assignment.
2. Tape numbers, base and end-of-program indicators, and words to be stored must each be followed by at least one tab or carriage return.
3. Initial zeros in addresses may always be omitted.
4. The numeral 1 and lower case letter l are not synonymous.
5. Characters ignored: comma, blank (000000), nullify (111111), space, back space, color change, upper and lower case shifts.

3-22-55

SA-57300-1

S. OLSEN
B-155

Division 6 – Lincoln Laboratory
Massachusetts Institute of Technology
Lexington 73, Massachusetts

SUBJECT: MTC CHANGES AND ADDITIONS AFFECTING PROGRAMMERS

To: Users of MTC

From: A. D. Hughes

Date: 24 February 1956

Approved: W. A. Hosier
W. A. Hosier

References:

1. Memorandum 6M-3974, Proposed MTC In-Out Block Transfer Instruction, B. G. Farley, 4 November 1955.

2. Memorandum 6M-2527-2 Memory Test Computer Programming Reference Manual, P. R. Bagley, 9 May 1955.

3. IBM, Pre-Preliminary Manual PM 8-13, Tapes and Miscellaneous 10 Units, Programmer's Reference Manual, 17 June 1955.

Abstract: Addition of tape units to MTC and a change in MTC's control logic have brought about some changes and additions affecting MTC programmers. The following instructions are altered:

1. id x (identity check): For normal operation, no alarm is given when words do not check.

2. sof a (select): In addition to "select operation field", sof a provides proper tape or drum selection for cb x. (Instruction renamed "select").

3. md x (memory address display): Delete this instruction.

4. cb x (copy block): New instruction used with sof a to implement block transfers of words for drum and tape.

5. pf x (perform): New instruction with variations which implements certain operations and senses certain conditions.

The drum (fields 3-14) is now provided only as a storage medium using cb x instructions. Registers from which control obtains instructions cannot be located on the drum.

The research reported in this document was supported jointly by the Department of the Army, the Department of the Navy, and the Department of the Air Force under Air Force Contract No. AF 19(122)-458.

INTRODUCTION

Three IBM 728 magnetic tape units have been installed on MTC. Also, the control logic of MTC has been changed. The resulting modifications with regard to the programmer include some changes to the previous MTC instruction code, some additions to the MTC instruction code, and minor changes to the MTC console.

This M-note gives the proper computer interpretations of those instructions which are affected by the changes to MTC. These explanations supersede the ones given in 6M-2527-2. The new instructions added to MTC are explained in the same manner. For all other instructions, 6M-2527-2 still provides the proper explanations.

Before using the drum and tape instructions given in this M-note the programmer is expected to be familiar with the MTC drum system and the XD-1 usage of the tape units. (See References) Additional information on the use of tape drive units is given in Appendix A.

1.0 CHANGES IN EXISTING MTC USAGE

1.1 Identity Check*

id x	identity check	#3	00011	id

Compare the contents of AC with the contents of register x. If it is not identical, prepare to skip the immediately succeeding instruction (by adding 1 to the Program Counter). If it is identical prepare to perform (not skip) the immediately following instruction. After the instruction has been executed, AC will contain 1's in each digit position where the original digit did not agree with the corresponding digit position of register x.

In conjunction with the identity check instruction there is an identity check alarm feature. Located on the MTC console is an identity check alarm indicator and identity check toggle switch. With the switch off or normal, the instruction behaves as stated above. With the switch on, the instruction behaves as stated above, and in addition, if the contents of AC do not agree with the contents of register x, the computer will stop and the identity check alarm indicator will come on.

Also, in conjunction with the identity check instructions are two perform instructions to be explained in Section 2.2.3.

* Supersedes id x, see reference 2, page 21.

1.2 Select

The sof instruction is now used as a general in-out and storage unit selection instruction and is renamed "Select" instead of "Select Operation Field". The former select operation field usage of the instruction insofar as fields 0, 1 and 2 are concerned (along with the programmed alarm feature) remain the same. The entire structure of the sof instruction is explained in Section 2.2.1.

1.3 Drum

Instructions appearing on the drum (Fields 3-16, octal) no longer may be performed by the computer directly, i.e. /sof n/ /tro x/ is not effective for any value of n except 0, 1 or 2, (panel or core storage). The drum is available as a storage medium using the cb (copy block) instruction, to be explained in Section 2.2.2.

1.4 Memory Address Display

The md (memory address display) instruction has been deleted. See reference 2, page 23.

1.5 Changes Not Affecting Instruction Code

Certain changes to MTC not affecting the instruction code have been made as an adjunct to MTC's new control logic. A speed-up in the memory cycle is anticipated, which will increase the overall operating speed of the computer. The amount of this speed-up has not been determined at this time.

On the MTC console, the physical location of some switches and indicators has been changed, and some additions have been made. All switches and indicators which were used in the past are included in the new layout, except that a few indicators have been changed to conform with the new control logic.

2.0 ADDITIONS TO THE MTC INSTRUCTION CODE

2.1 Introduction

The additions to the MTC instruction code are the new instructions cb (copy block), pf (perform) and the additional functions of the instruction sof (select). The use of tape units and drum fields* is accomplished by a series of instructions which transfer words between internal memory and the tape units or drum fields. This series of instructions is briefly as follows:

---

* At some time in the future all in-out equipment will be handled in much the same manner as drum and tape equipment.

1. ca: RC drum starting address, (used with drum only)

   (RC = register containing)

2. sof: code for mode, medium, unit

   a. select mode (read, write, dummy)

   b. select medium (core memory, tape, drum)

   c. select unit or field

   This instruction also provides sensing to insure in-out medium is physically ready.

3. ca: RC word count (positive number of 16-bit words to be copied)

4. cb: internal memory starting address

   Provide sensing to insure in-out medium is in proper condition to cb, then perform transfer of words.

This sequence of instructions is included here only to help in understanding the material to follow. Section 2.2 elaborates on the use of these instructions.

The sof instruction is used in its original form (select operation field) as well as with cb. Besides using sof and cb to effect use of tape units and drum fields, they can be used in modes to clear (read 0's into) blocks of words in core memory, and to write blocks of identical words (which could be 0's) on tape or drum from Panel Memory.

Section 2.2 explains the function of each instruction and its variations, if any. Section 2.3 gives a table of normal sequences of instructions to aid programmers in making use of tape units, drum fields and dummy modes.

2.2 Explanation of Individual Instructions and Variations

2.2.1 Select (formerly Select Operation Field)*

For convenience sof 0000 + n, sof 2000 + n, and sof 0400 + n are called "internal select" instructions; sof 1200 + n, sof 3200 + n, sof 1300 + n, sof 3300 + n, and sof 2300 + n are called "in-out select" instructions.

sof a	select	#21	10101	sof

* Supersedes sof a, reference 2, page 24.
ATTENTION: It has been proposed to substitute se or set for sof in view of the added functions of this instruction. We would appreciate hearing your reaction to such a substitution.

sof 0000 + n (CHANGE FIELDS)

Until the next internal select instruction is given, treat address parts of instructions which refer to memory as referring to field n; n shall be restricted to 0, 1 or 2 (panel and core memory).

sof 2000 + n (CHANGE FIELDS: ALARM)

Treat instruction just as sof 0000+n, and in addition generate a programmed alarm. (The programmed alarm feature may be suppressed by a switch on the MTC console.)

sof 0400 + n (CHANGE FIELDS: DUMMY READ MODE)

Treat instruction just as sof 0000 + n, and in addition select a dummy read mode of operation such that cb reads a block of zeros into core memory. Three properties of this instruction are as follows:

1. The internal memory field selected by this instruction remains effective until the next internal select instruction.

2. The dummy read mode selection remains effective until the end of the first following cb instruction. (If no cb instruction has followed this sof 0400 + n instruction, when an in-out selection instruction is given, the dummy mode selection is cancelled.)

3. If this instruction is inserted between an in-out select instruction and a cb instruction, the cb instruction will perform the dummy mode block transfer. However, succeeding cb instructions will revert back to the selection made by the in-out select instruction.

sof 1200 + n (SELECT DRUM FIELD: READ)

Sense the in-out interlock and wait until clear. Deselect any previously selected in-out media, modes and units. Select Drum Medium. Select Drum Field n, (3-16, octal). Select Read Mode. Sense to see if drum motor is on; if not give an alarm and stop. Copy drum starting address (AC 5-15) into Drum Address Register. The selections made by this instruction remain in effect until the next in-out select instruction is executed.

sof 3200 + n (SELECT DRUM FIELD: WRITE)

Treat instruction just as sof 1200 + n, except select the Write Mode instead of the Read Mode.

sof 1300 + n (SELECT TAPE UNIT: READ)

Sense the in-out interlock and wait until clear. Deselect any previously selected in-out media, modes and units. Select Tape Medium. Select Read Mode. Sense to see if the selected tape unit is Ready, i.e. physically in the proper readiness to manipulate the tape; if not, give an alarm and stop. The selections made by this instruction remain in effect until the next in-out select instruction is executed.

sof 3300 + n (SELECT TAPE UNIT: WRITE)

Treat instruction just as sof 1300 + n, except select the Write Mode instead of the Read Mode.

sof 2300 + n (SELECT TAPE UNIT: VERIFY)

Treat instruction just as sof 1300 + n, except select the Verify Mode instead of the Read Mode.

2.2.2 Copy Block

Reference to instructions 1-4 in Section 2.1 shows the following sequence of the instructions to be used with cb. The abbreviations shown here are used in the discussion of the cb instruction below:

1. ca RC a (a = drum starting address) (Used for drum only)

2. sof s (s = proper mode, medium and unit designations)

3. ca RC wc (wc = word count)

4. cb x (x = internal memory starting address)

Note that no instruction which changes AC should be used between 1. and 2. or 3. and 4. However, any type of instruction may be used between 2. and 3. Note, also, (from Section 2.2.1, sof) that once 2. is given for tape or drum selections, it remains in effect for any number of 3. and 4. instructions to follow. For cb using tapes, the two instructions immediately following 4. must be reserved for conditional transfers, (see below).

Some care must be exercised to insure proper field selection of internal memory when this sequence of instructions is used. Each of the instructions 1., 3., and 4. refer to internal memory and must be preceded with proper field selection when required.

When the cb instruction is being performed, the AC is used as the word counter, initially holding wc. Copying into or out of internal memory begins with register x and proceeds in order through memory until the number of words initially specified by the AC has been

copied. It will be the programmer's responsibility to make sure that blocks to be copied fit into the space allotted. If the end of a core memory or drum field is reached before zero in the word counter, the address counters will start over at zero and continue, and no alarm will be given. Note that MTC can copy maximum blocks of 2048 (decimal) 16-bit words.

cb x	copy block	#5	00101	cb

cb x (s = 0400 + n: DUMMY READ MODE)* (CLEAR BLOCK IN CORE MEMORY)

Read into a block of core memory the number of zero words specified as wc, starting with register x. When the cb instruction is completed, clear dummy mode selection and revert to previous in-out select status, if any.

cb x (s = 1200 + n: DRUM READ)

Read a block of words from Drum Field n, starting with register a. Words are transferred into core memory, starting with register x; the number of words being given by wc in AC.

cb x (s = 3200 + n: DRUM WRITE)

Write a block of words onto Drum Field n, starting with register a. Words are transferred from internal memory, starting with register x; the number of words being given by wc in AC.

cb x (s = 1300 + n: TAPE READ)

1. Sense the in-out interlock and wait until clear. Sense to see if tape unit n is rewinding and wait until rewind is over. Sense to see if tape unit n is In File Area. If Not In File Area, immediately proceed to the next instruction without further cb operations. If tape unit n is In File Area continue with the cb instruction.

2. Start reading a record from the tape (unit n) (in-out interlock becomes set) storing each 32 bit tape word into 2 registers of core memory, starting with register x. Tape bits 1-16 appear in the first of the two core memory registers (corresponding to the left half word of XD-1) and tape bits 17-32 appear in the second register (corresponding to the right half word of XD-1).

---

* Note that the Dummy Read Mode selection temporarily overrides all other selection until the cb is executed. (See sof 0400 + n.)

3. The number of (16-bit) words transferred is determined by the tape record length or wc whichever is smaller. When End of Record or AC = 0 is reached, the block of words has been transferred into core memory. If the End of Record is reached before AC = 0, AC will contain the difference between wc and the record length. Note that wc should always be an even number. If AC = 0 is reached before the End of Record, the tape unit is allowed to continue moving tape as if reading, until the End of Record gap is reached.

4. Perform a parity check of each pair of words, using the tape parity bit (33). If a parity error occurs, immediately skip the instruction following the cb, and perform the next instruction without further cb operations. The tape unit is allowed to continue moving tape as if reading, until the End of Record gap is reached. However, the Tape Word Register will contain the pair of words which contain the parity error just read from the tape, and AC contains wc minus the number of words read. When the parity error is detected, a tape parity indicator will come on and can be observed if the second instruction following the cb is a halt. The word register and tape parity indicator will remain unchanged until a new cb or in-out select instruction is performed. A perform instruction is provided which suppresses the parity check feature of a single cb instruction. (See Section 2.2.3) (A toggle switch on the MTC console provides manual suppression of the tape parity.)

5. Upon completion of the transfer of words, the tape unit is again sensed for In File Area. If the (normal) condition exists, immediately skip the next two instructions following the cb, and perform the third instruction without further cb operations. If the tape unit is Not In File Area, immediately proceed to the next instruction without further cb operations. Regardless of when the cb instruction is terminated, the in-out interlock remains set until the tape reaches End of Record.

cb x (s = 3300 + n: TAPE WRITE)

1. Treat just as step 1. of TAPE READ.

2. Start writing a record onto the tape (unit n) (in-out interlock becomes set) storing 2 registers of core memory into a 32 bit tape word (plus parity). The bit arrangement is given in step 2. of TAPE READ.

3. The number of (16-bit) words transferred is wc. When wc words have been transferred an End of Record gap appears on the tape such that the record contains wc (16-bit) words. Note that wc must always be an even number.

4. Treat just as step 5. of TAPE READ.

cb x (s = 2300 + n: TAPE VERIFY)

1. Treat just as step 1. of TAPE READ.

2. Treat just as step 2. of TAPE READ, except do not store the words into core memory. Instead, perform an id on each word, i.e. compare the words being read from the tape with corresponding words stored in core memory, starting with register x. If the words do not agree treat the situation exactly as if a parity error had occurred, step 4. below.

3. Treat just as step 3. of TAPE READ, except the number of words is the number compared, not the number transferred.

4. Treat just as step 4. of TAPE READ.

5. Treat just as step 5. of TAPE READ.

cb x (PROGRAMMED DUMMY WRITE MODES)

The programmer may write a block of any repeated 16-bit word (such as all 0's) on tape or drum. This can be accomplished by preceding the cb (DRUM WRITE or TAPE WRITE having been selected) by a selection of field zero, (panel memory). Since the address register for Panel Memory does not advance like that for Core Memory during execution of cb, the result will be a repeated transfer to tape or drum of Panel Memory register x, the number of identical words being determined by wc.

2.2.3 Perform

The perform (pf) instruction has been added to provide various special operations and sense certain conditions. Most of the pf instructions are provided to aid in operating the tape units properly. The pf instructions which refer to one tape unit (marked with an asterisk, *) must have been preceded by an sof (SELECT TAPE UNIT) instruction in order to provide selection of the desired tape unit. Which of the three SELECT TAPE UNIT instructions used depends on the program; all three produce the same result with regard to the pf instructions.

pf a	perform	#23	10111	pf

pf 0000* (REWIND)

Move the tape back to the Load Point. This instruction does not sense the in-out interlock and as such can override reading, writing, or backspacing of the tape, i.e. while the tape is performing a cb or BACKSPACE instruction, the execution of a REWIND instruction will immediately terminate the existing tape instruction and move the tape back to the Load Point. While the tape is rewinding, the tape unit is in REWIND status.

The tape unit remains in either IN FILE AREA or NOT IN FILE AREA status during and upon completion of REWIND, the status of the unit when rewind begins. The in-out interlock is set upon execution of the rewind instruction, but remains set only for the first 40 milliseconds of REWIND.

When the tape has completed rewinding the tape unit is in the AT LOAD POINT status, indicating the tape is at the beginning of a reel. (See SENSE AT LOAD POINT, below)

pf 0001* (SET IN FILE AREA)

Set the selected tape unit to the IN FILE AREA status.

If a cb instruction had brought the tape past an End of Tape mark or the last instruction was WRITE END OF FILE, (see below) the tape unit becomes in the NOT IN FILE AREA status. It will remain in this status until either a BACKSPACE instruction (see below) or SET IN FILE AREA instruction is given. Note that after REWIND, a SET IN FILE AREA instruction must be given before any cb instructions will execute a TAPE READ or WRITE operation.

pf 0002* (BACKSPACE)

Move the tape back to the first previous End of Record gap. This instruction does not sense the in-out interlock and as such can override reading or writing of the tape, i.e. while the tape is performing a cb instruction, the execution of a BACKSPACE instruction will immediately terminate the existing tape instruction and move the tape back to the End of Record gap. While the tape is backspacing the in-out interlock is set, if not already, and in any case upon completion of BACKSPACE the in-out interlock is cleared.

If the tape unit is in the NOT IN FILE AREA status when BACKSPACE is given, the BACKSPACE instruction sets the unit to the IN FILE AREA status.

pf 0003* (WRITE END OF FILE)

Sense the in-out interlock and wait until clear. Write a one word record on the tape which writes the three end of file bits in the record.

While writing the end of file record, set the in-out interlock and upon completion of the operation, clear the in-out interlock. Writing of the end of file record puts the tape unit in the NOT IN FILE AREA status.

pf 0004 (GIVE ALARM ON ID CHECK, SET)

Set up the condition whereby upon execution of an id x instruction (see Section 1.1), if the contents of AC do not agree with the contents of register x, the computer will stop and the identity check alarm indicator will come on. This condition remains in effect until a GIVE ALARM ON ID CHECK, CLEAR instruction is given, below. Note that this instruction has the same effect as the on position of the identity check toggle switch. (Start at 40 gives a pf 0004 automatically.)

pf 0005 (GIVE ALARM ON ID CHECK, CLEAR)

Clear the condition set up by the GIVE ALARM ON ID CHECK, SET instruction, thus returning to normal the id instruction. The identity check toggle switch on position overrides the pf 0004 and pf 0005 instructions, but when the switch is turned off the condition according to the last GIVE ALARM ON ID CHECK instruction recurs. (Start over CLEARS.)

pf 0006 (SUPPRESS TAPE PARITY)

Do not perform step 4. of the next cb TAPE READ or VERIFY instructions, i.e. if a parity (or no identity check for TAPE VERIFY) occurs, ignore the condition as if it had not appeared. The suppression is cleared after the cb TAPE READ or VERIFY instruction and therefore must be given for each cb instruction for which the suppression is to take place.

pf 2000* (SENSE REWIND)

Sense to see if the selected tape unit is rewinding, i.e. in the REWIND status. If it is rewinding prepare to perform the immediately following instruction. If it is not rewinding, prepare to skip the immediately following instruction and perform the second instruction.

pf 2001* (SENSE NOT IN FILE AREA)

Sense to see if the selected tape unit is in the NOT IN FILE AREA status. If the condition exists, i.e. if it is NOT IN FILE AREA, prepare to perform the immediately following instruction. If the condition does not exist, i.e. if it is IN FILE AREA, prepare to skip the immediately following instruction and perform the second instruction.

pf 2002 (SENSE IN-OUT INTERLOCK)

Sense to see if the in-out interlock is set. If it is set, prepare to perform the immediately following instruction. If it is not set, prepare to skip the immediately following instruction and perform the second instruction.

pf 2003* (SENSE AT LOAD POINT)

Sense to see if the selected tape unit is in the AT LOAD POINT status. If it is not AT LOAD POINT, prepare to perform the immediately following instruction. If it is AT LOAD POINT, prepare to skip the immediately following instruction and perform the second instruction.

SIGNED: A. D. Hughes
A. D. Hughes

ADH:jg

Attachments: APPENDIX A
Distribution List

Operation of 728 Magnetic Drives

Before tape may be loaded the machine must be in an UNLOAD status as shown in picture. In this status the upper head section is raised, the capstans are retracted so the tape has an uninterrupted path between the outside idler. The machine is normally in this status following a run. The full reel of tape (file reel) is placed on the left reel and the tape threaded past the idlers and Read-Write Head Assembly and over to the empty reel (machine reel) on the right. The reel brakes may be released by depressing the buttons below the left reel. Tape is then wound manually onto the machine reel until several turns have been completed and the load point is well beyond the head assembly. The load point is a reflective spot of Aluminum placed on the near edge of the tape about 10' from the end to mark the electrical beginning of the tape.

The door may now be closed and the LOAD/REWIND button depressed. This will cause the Head Assembly to be lowered into position and tape to be sucked into the columns.

As soon as the upper head assembly is down in position the capstans are extended and begin to turn. The machine automatically goes into a Backward status and tape begins moving in a reverse direction. It will move in this direction until the Load Point is sensed. At this point it will stop and the machine is ready for use. If the load point has not been wound well into the machine reel, the machine will search off the end of the tape for the load point.

Control from the external source is exercised first by selecting the tape unit with the sof instruction. This will result in the SELECT light on the front panel coming on.

NOTE: RESET - Manual Before any manual operation may be undertaken, the machine must be put under manual control by depressing the RESET button.

Unload

The tape unit may be unloaded at any point in an operation, providing the tape is stopped and the unit is under manual control. This may be accomplished by depressing the UNLOAD button. This causes the tape to be retracted from the columns, the upper head assembly raised and the capstans retracted.

High Speed Rewind

When a tape has been completely written or read, the bulk of the tape is contained on the machine reel. Since time is a prime factor in the operation of this unit it is desired to accomplish a rewind to the file reel as fast as possible. This may be accomplished automatically by the computer pf instruction, or manually by depression of the LOAD/REWIND button. The machine will perform initially as in an UNLOAD operation but as soon as tape is out of the columns and the upper head assembly is up the file reel begins to wind tape from the machine reel.

Machine Reel Sensing Arm

The load point at the beginning of the tape is the only indication which can be used to stop the tape. However, this cannot be sensed at the terrifically high speed of rewind and cause the tape to stop immediately. Therefore, the high speed portion of rewind is suspended before the beginning of the tape is reached and the remaining few feet moved at normal speed. This is accomplished by using a Machine Reel Sensing Arm which closes a contact when only 1/2 inch of tape remains on the machine reel. The rewind stops at this point and the tape is automatically loaded into the vacuum columns. The tape is then moved in reverse (at normal speed) as during a normal loading operation until Load Point is reached. The machine then stops and is ready for further use.

Panel Buttons and Lights

Located on the door of the tape unit are the operating buttons and lights. These have effect only if the door is closed.

Select Switch

This switch is a rotary switch located in the center of the group and is used to set the tape unit to any one of the three addresses necessary. If the switch is set on address 1 the unit will be selected when address 1 is selected by the computer and so forth. If two tapes are set to the same address writing can take place on both. Reading will not be possible since they cannot be synchronized.

Select Light

The select light comes on when the tape unit is selected and remains on until the selection is removed.

Start Button

Depressing the START key causes the machine to be in a READY condition if (a) tape has been previously loaded into the column, (b) the door is closed, and (c) the tape is not moving.

Not Ready Light

This light is off if the machine is in a READY status as described above, and indicates that the machine is under manual control.

File Protection Light

This light is on when a protected reel has been loaded onto the unit. It provides a means to protect a reel of tape from erasure.

Load/Rewind Button

If the door is closed, depression of this key will cause loading of the tape into the columns and searching for the load point. If the tape has been manually unloaded in the fast rewind section of the tape, depressing this key will execute a high-speed rewind before the above operation takes place. This button is inoperative unless the tape unit is under manual control.

Unload Button

Operation of this button will remove the tape from the columns and raise the upper head assembly regardless of the distribution of tape on the two reels. If the tape is not at load point when the operator wishes to change it, a load point search should be first initiated by depressing the LOAD/REWIND button. The tape indicator (see below) is reset by depressing this button. This button is inoperative unless the tape unit is under manual control.

Not In File Area

This light comes on whenever the NIFA Flip-Flop is turned on by external control. This happens upon sensing the end-of-tape reflective spot or an end-of-file mark written on tape. Writing is prevented while the Flip-Flop is on.

Reset Button

This button resets all controls, (except the Tape Indicator) and, in general will stop any tape operation which has been initiated. As mentioned before, manual control exists after it has been pushed.

Tape Handling

Since the recorded information on tape comes within 0.020" or the edge of the tape the tape reels must be handled with extreme caution. Avoid any pinching of the rims of the reels or any contact with the edge of the tape. Any pushing or pulling necessary to put the reel on the machine or remove it should be done at the root diameter of the reel and not at or near the outside.

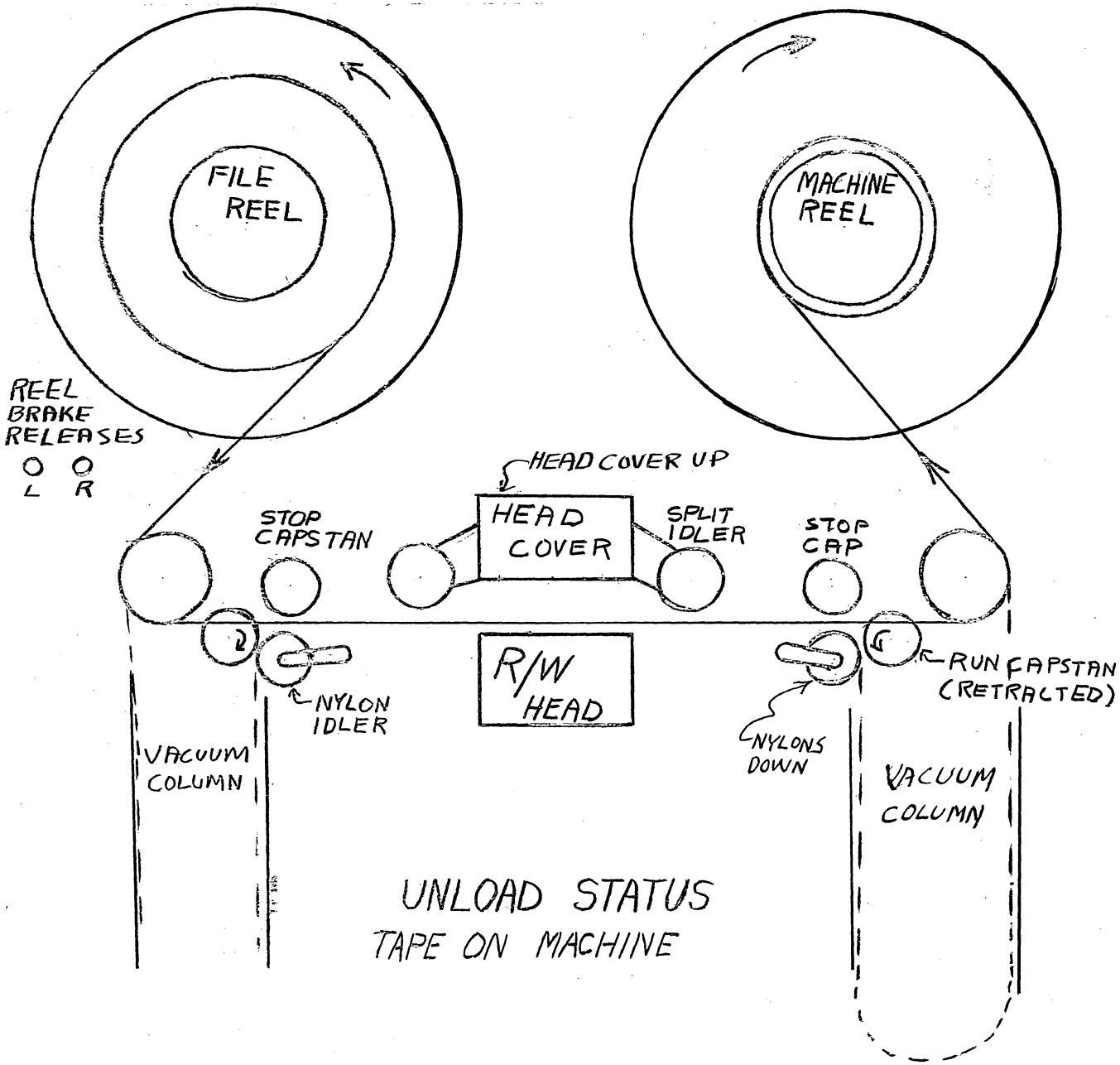
When shipping, the tape should be protected in a dust proof container and the reel be suitably supported in this container with additional protection afforded by a stiff cardbbard shipping box. The file protect ring must be in the groove around the reel but in order that writing may be done on that tape.

Electrical Logic Circuits

Select and Ready

A tape unit will be ready for selection when the tape has been loaded into the columns, the reel door interlock is closed, and the tape unit is not in the process of finding the load point (Rewind or Load Operation.). If these conditions are met they will be indicated by the plus status of two of the three inputs to AND circuit A12 (75.01).

The third input will come up when the external select line corresponding to the setting of the Select Switch comes up. The output of AND circuit A12 becomes the SELECT AND READY line.



UNLOAD STATUS
TAPE ON MACHINE

6M-2527-2 Cor. # 2

Distribution List

GROUP 21

*J. C. Watson

GROUP 22

*R. K. Bennett
*C. M. Festa
*G. B. Harris
*W. F. Heart
*R. C. Holland
*W. F. Holst
*A. Mathiasen
H. Meinholtz
*B. Stahl
*C. Uskavitch
*C. Wakstein

GROUP 24

P. Conrad
G. Dineen
J. Dumanian
*B. Jensen

GROUP 34

*M. Kannel
*A. Tritter

GROUP 61

H. E. Anderson
J. Bockhorst
M. Clark
V. A. Fuller
E. L. Lafferty
J. Levenson
B. Persell
R. F. Russo
C. R. Shocit
S. Towers (Rand)

GROUP 62

I. Aronson
W. Canty
F. R. Durgin
*R. H. Gerhardt
*S. Ginsburg
*R. Mayer
P. Messenheimer
*H. Rundquist
*A. Werlin

GROUP 63

*W. Clark
*C. Corderman
*S. DiFazio

GROUP 64

*R. D. Buzzard
*B. Farley
*E. Gates
*E. Glover
*A. Habeeb
*W. Hosier
*A. Hughes
*N. Ockene
*D. Parrott
*E. Robinson
*E. Rich
*J. Salvato
*T. Stockebrand
*A. Vanderburgh

GROUP 67

*P. Bagley
*H. Benington
*R. Carmichael
*A. Rupp

GROUP 71

E. Sonier

Norm Taylor
*G. E. Valley D. O.
*R. L. Dougherty BTL
*D. Meng BTL