Division 6 - Lincoln Laboratory
Massachusetts Institute of Technology
Cambridge 39, Massachusetts

SUBJECT: MEMORY TEST COMPUTER: PROGRAMMING MANUAL

To: N.H. Taylor and those listed

From: Philip R. Bagley

Date: November 25, 1953; Revised April 23, 1954

ABSTRACT: The Memory Test Computer is a high-speed electronic digital machine with over 8,000 registers of magnetic core and drum storage. 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 80,000 instructions per second, exclusive of input-output instructions. The terminal equipment consists of paper tape reader and punch, electric printers, oscilloscopes and camera.

In addition to a brief description of the computer, the machine functions, instruction code, and terminal equipment codes are presented 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.

TABLE OF CONTENTS

FOREWORD 4

SECTION I. GENERAL DESCRIPTION OF THE COMPUTER 5

Purpose of the Computer 5
Characteristics of MTC 5
Physical Layout 5
Bus System and Information Transfer 5
Simplified Block Diagram 6
Memory Element 6
Control Element 6
Arithmetic Element 7
Console 7
Power Supply 7
Terminal Equipment 7

SECTION II. GUIDE TO CODING 8

Computer Programs 8
Computer Components 8
Representation of Instructions 9
Representation of Numbers 9
Computer Procedure 10
Notation for Coding 12

SECTION III. OPERATION CODE 16

Notes on the Operation Code 16
Description of the Operation Code 17
Execution Times of Instructions 24

SECTION IV. TERMINAL EQUIPMENT 26

Camera 26
Printers 26
Punch 26
Readers 26
Scopes 27
Plotting Scope 27
Memory Address Scope 27
Charactron and Typotron Scopes 27

SECTION V. INPUT AND BASIC CONVERSION PROGRAMS 31

Brief Description of the Input Process 31
Punched Paper Tape and Tape Equipment 32
Basic Conversion Program 34

TABLE OF CONTENTS (continued)

Preparation of Standard Tape for Conversion 36
Operating the Basic Conversion Program 41
Structure of Converted (4-6-6) Tape 43
4-6-6 Input Program 46

APPENDIX 49

MTC Instruction Code (Brief Form) 49
"FL" Flexowriter Codes 50
IBM Printer Codes 52
Charactron and Typotron Codes: MIT Matrix Mod VI 53
Vocabulary, MTC Basic Conversion Program 54
Temporary Precaution to be Observed When Programming for The Magnetic Drum 55
Brief Operating Guide for MTC 57

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, Memorandum M-2527-1, 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

This manual properly applies to the MTC Computer System as it will exist on or about July 1, 1954. Before this date some of the following features may not be installed:

1. the instructions ch, et, rf, and sm.
2. the "short cycle" and "roundoff" variations of the cr and sr instructions.
3. The scope camera and 16" display scope unit.
4. The Charactron with the facilities described herein.
5. The addresses on the magnetic drum "interleaved" by a factor of 16.

Programs dependent on any of the above facilities should not be run before July 1 unless the programmer has checked with the MTC Section.

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

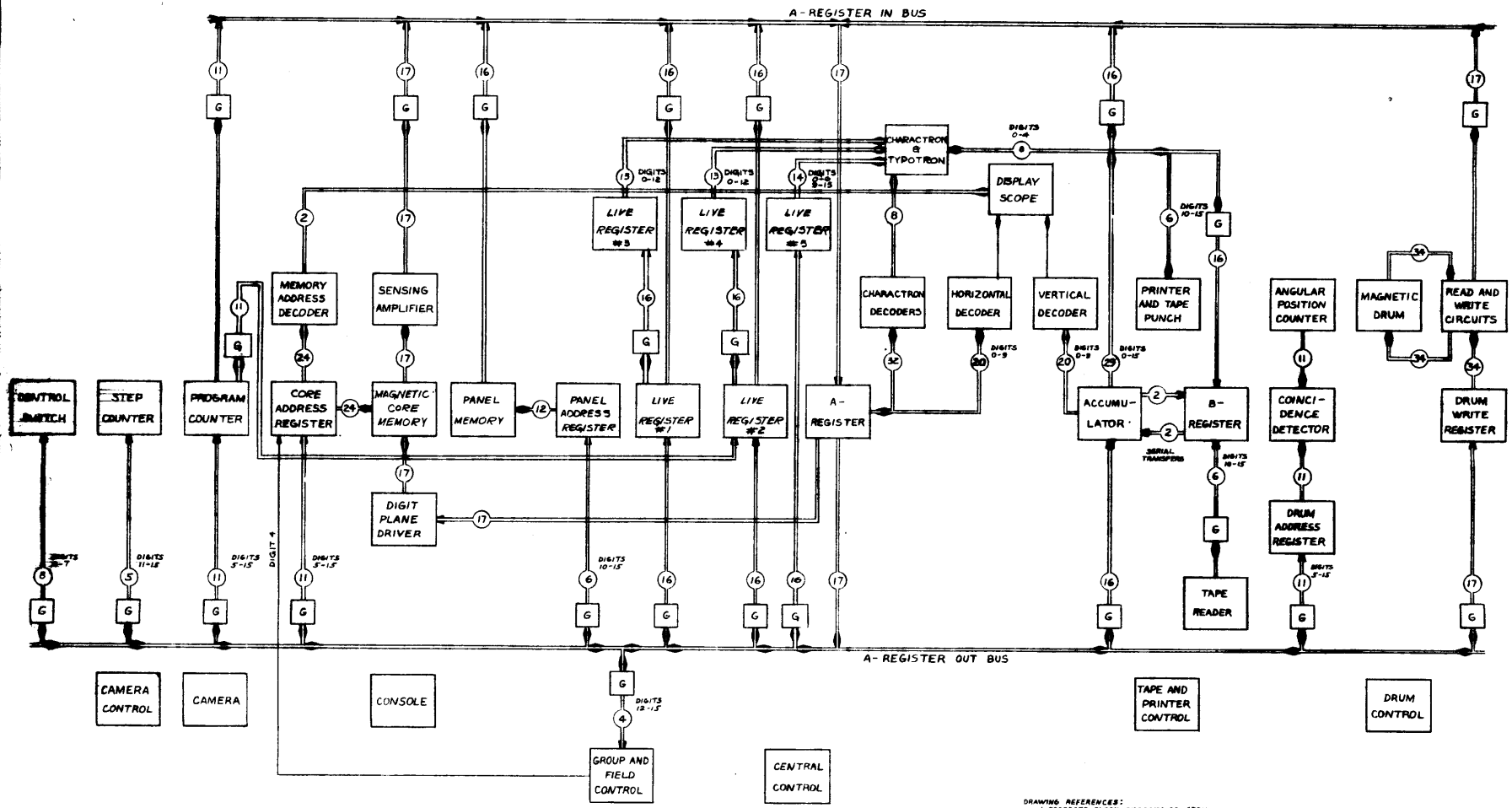
The writer owes debts of thanks to Professor C.W. Adams, upon whose work Section 2 of this memo is largely based, and to W.A. Hosier, who has patiently edited this memo and has contributed many valuable suggestions.

Changes in the computer which affect programming methods will be reported in periodic supplements to this memorandum. The Division 6 Document Room maintains a record of the recipients of this memo to insure that they will receive all future supplements.

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.

D-47039

MN



DRAWING REFERENCES:
1. PROPOSED BLOCK DIAGRAM: SD-47011

GRADED BY: DATE:

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
DIGITAL COMPUTER LABORATORY

SIMPLIFIED BLOCK DIAGRAM, MTC

D-47039

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 80,000 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 (currently 4096 registers, expandable to 24,576 registers). The terminal equipment consists of punched paper tape readers and punch, printers, and scopes (one scope fitted with a camera under computer control).

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 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 storage, 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 printers, 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. Three 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 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 two fields, each of 2048 registers. In the future more fields, up to a maximum of twelve, may be connected. 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 loud-speaker. 32 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 main input devices are two punched paper tape readers. The output devices are: two printers, which can print any format achieved by the keys of a typewriter; a paper tape punch, for punching tape which is later used in the reader; and various oscilloscopes, one which is equipped with a camera operating under control of the computer. The oscilloscopes may be used to generate any pattern, including alphanumerical characters and graphs.

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 varous 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 (at present) 8,256 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 0, 1, 2, 3, or 4, 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 and 4, having extended addresses 3-0000 through 3-2047 and 4-0000 through 4-2047, are on the magnetic drum. (As more magnetic drum registers are added, they will have extended addresses beginning with 5-0000.) 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 three 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 Storage (addresses 0-63). LR2 and LR5 may each be substituted for any one or more registers of Plugboard storage only (addresses 32-63). At the moment, however, LR5 is a special-purpose register which can be read into but not out of.

Input-Output. Information entering the computer is temporarily stored in the B-Register (BR). Information leaves the computer via the AC. 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 contol element takes its instructions one at a time from storage, where the instructions are stored as individual words.

Inter-connections. The four main elements of the computer (storage, 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, to and sr, can be used to detect the overflow without any alarm. There is also a switch by which the alarm may be suppressed entirely.

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 storage addresses in the computer. At least 2 additional digits are required in order to specify all 8,256 registers. Hence, it has been necessary to deal with storage 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.

Sample Program Illustrating the Use of Field-Switch Instructions (tro, tno, and sof). This program, a simple field-to-field "bootstrap", rewrites itself alternately in field 1 and field 2 each time it is performed. After doing this a predetermined number of times, it can be made to stop, to print something and resume, or whatever the operator intends. It will be recalled that the "start over" button always starts from register 0-0. The reader will need to refer to the operation code starting on p. 17.

Panel Storage

```
0-0  tr     2
0-1  counting decrement
0-2  ca     6   \
0-3  sof    1    } sets up counting register,
0-4  st    25    } then starts program proper.
0-5  tro    0   /
0-6  0.77777    (constant)
0-7  ha*    0
```

*If the program is not to stop after switching the required number of times, some other sequence of instructions can be substituted for ha 0.

Magnetic Core Storage

```
1-0000 ca   22  \ Sets counters to
1-0001 st    3   } begin transfer
1-0002 ra    5  / with register 0.
1-0003 ca   --  \
1-0004 sof   2   } Transfers
1-0005 st   --  / one word.
1-0006 sof   1  \
1-0007 ca    3   \
1-0008 ad   23    } Indexes for next word.
1-0009 st    3   /
1-0010 ra    5  /
1-0011 su   24  \ Checks for end of
1-0012 tn    3  / transferred block.
```

Magnetic Core Storage (cont'd)

```
1-0013 ca   25  \ Subtracts one
1-0014 sof   0   } decrement from
1-0015 su    1   } field-switch
1-0016 sof   1   } count.
1-0017 st   25  /
1-0018 sof   0  \ Transfers to 0-7
1-0019 tno   7   } if count is
1-0020 sof   1   } complete.
1-0021 tr   26  / Does rest of program.
1-0022 ||ca   0
1-0023 0.00001
1-0024 ||ad   0
1-0025 Stores switch count.
1-0026 \
  thru  } Registers 1-0026 through 1-2034 may contain any program not involving
1-2034 / field-switching.
1-2035 sof   2  \
1-2036 ca    4   \
1-2037 ra    6    \
1-2038 ra   16     \ Interchanges fields of all sof 1 and sof 2 instructions for next iteration of program.
1-2039 ra   20     /
1-2040 ra 2041    /
1-2041 sof   1   /
1-2042 ca    6  /
1-2043 sof   2  /
1-2044 ra    4  /
1-2045 ra 2035 /
1-2046 ra 2043 /
1-2047 tro   0  Starts program over in other field.
```

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 content is 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 (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 (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.

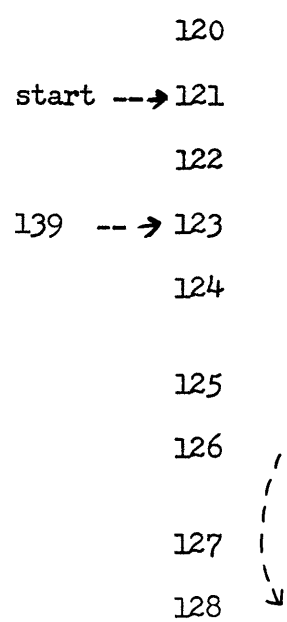
Procedure in the arithmetic element. The execution of an addition includes the process of adding in carries; this process treats all 16 digits as if they were numerical digits, a carry from AC 0 ("end-around carry") being added into AC 15. (This compensation is necessary because of the method of representing 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. 20f.)

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 two dashes, 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 on the following pages.



	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	(-)	addresses 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 ◀#, where ◀ is any lowercase 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.

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 sometimes symbolized $\lfloor \ldots$

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

This is sometimes 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. OPERATION CODE

NOTES ON THE OPERATION CODE

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

Abbreviations. The abbreviations used are the following:

AC = Accumulator

AR = A-Register

BR= B-Register

x = address of a storage register

a,k,n = positive integers

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

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 alram, 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.

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

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.

DESCRIPTION OF THE OPERATION CODE

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. 16.)

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.

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.

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

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. 16).

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. 27 .

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 is left in BR, and the previous contents of BR is 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 ha y+1 in LR2, the previous contents of LR2 being lost. If this instruction is stored in panel memory, the contents of LR2 is undisturbed.

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 ha y+1 in LR2, the previous contents of LR2 being lost. If this instruction is stored in panel memory, the contents of LR2 is undisturbed.

tn x	transfer on negative	#18	10010	tn

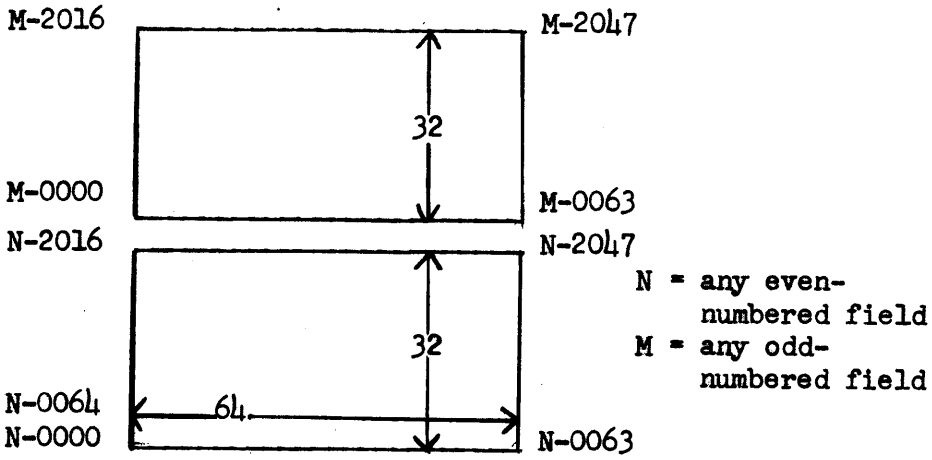
If the number in AC is negative (i.e., has a "1" in digit position 0), perform the same function as tr above. If the number in AC is positive, this instruction has no effect.

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.

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:



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 program alarm. (The program 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," p. 16.) If no overflow condition exists, this instruction has no effect.

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 21.

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)

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. 16.)

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!

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 1-15 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.

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). Since this instruction reads in only ones and does not clear any digit positions in BR, BR should, if necessary, be cleared by a previous instruction.

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.

EXECUTION TIMES OF INSTRUCTIONS

Instruction times for programs confined to core memory. Given in the table below are the times for the execution of instructions which are stored in core memory and which refer only to addresses in core memory.

Instruction	Execution Time
mh	44 μsec (average)
ds	100 μsec
id ca ad cs su et st	10 μsec
ra rf	13.6 μsec
ch (Charactron)	50 μsec
ch (Typotron)	100 μsec
sm	11.5 μsec
tr tro tn tno to	8.4 μsec
sof	8.0 μsec
md	50 μsec
cr sr	8 + .75n μsec where n = no. of shifts
pr (print)	130 millisec
pr (punch)	83 millisec
ri (Ferranti)	4.8 millisec
ri (Flexo)	110 millisec
op (index camera)	200 millisec
op (erase Typotron)	50 millisec

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.

Instructions in	Data in	Average instruction time (μsec)
Panel Memory	Panel Memory	12
Panel Memory	Core Memory	12
Panel Memory	Drum Memory	10,300
Core Memory	Panel Memory	12
Core Memory	Core Memory	12
Core Memory	Drum Memory	10,300
Drum Memory	Panel Memory	700
Drum Memory	Core Memory	700
Drum Memory	Drum Memory	20,600

SECTION IV. TERMINAL EQUIPMENT

Camera. The Fairchild Scope Recording Camera is normally fitted to a 16" display scope. The camera shutter is normally open (when the camera control circuits have been manually turned on). The execution of the instruction op 0 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 200 milliseconds. 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.

Printers. There are two output printers available, only one of which may be connected at any given time. Either printer is actuated by the pr instruction, in accordance with the contents of AC 10-15.

The Flexowriter accommodates a maximum of 95 characters per line, and prints a maximum of 8 characters per second. (See Flexowriter Printer Code, p. 50).

The IBM Printer normally accommodates a maximum of 90 characters per line and prints a present maximum of 7 characters per second. 157 characters per line are possible when using special wide paper. (See IBM Printer Code, p. 52).

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.

Readers. The 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.

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 16" 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 100 μsec. The contents of AC at the time the instruction ds x is executed determines the vertical position Y, while the contents of register x determines 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 $(+1 - 2^{-9})$.

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 20 for a 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."

Charactron and Typotron. The Charactron is a special 19" scope which is equipped to display alphanumerical characters, selected symbols, and vectors, as well as points. The Typotron is a second special 5" scope which can display alphanumerical characters, symbols, and points. The feature present in the Typotron which is absent from the Charactron is the ability to retain a display for an indefinite period. Any pattern

once displayed on the Typotron phosphor mosaic is held without flicker by a holding beam until the whole surface is erased by the instruction op 2000 (octal).

Displays are achieved on the Charactron and Typotron by a technique somewhat more involved than the display of points on the plotting scope:

A. The Charactron has a dual deflection system --- slow (electromagnetic, about 50 microseconds) and fast (electrostatic, about 5 microseconds). The electrostatic system is of limited deflection (5 bits each x & y, corresponding to a 31 x 31 square array of contiguous characters) and in the charactron is intended to be superimposed on a central deflection set up by the electromagnetic circuits, which give full deflection over the tube face. It also has facilities for vector generation and dimming of characters. (The character matrix is depicted in drawing A-58258-1, MIT Matrix Mod. VI.)

1. The magnetic position of the Charactron beam is fixed by the contents of two "live registers": #1 and #2, normally assigned the octal addresses 0-61 and 0-62, with 13 bits each of x- and y-deflection respectively inserted as signed binary numbers in digits 0-12. The extra precision built into this system is to allow an 8-diameter magnification under manual control at the charactron display console.

2. Having positioned the magnetic "center" of a display, if one first desires to display a vector, one loads a third register (LR #5, normally addressed 0-65 octal) with signed vector components: a sign and six bits of x in digits 0-6, sign and six bits of y in digits 9-15. If a character rather than a vector is desired, this step is omitted.

3. One then proceeds to load a word containing certain control information into the Accumulator:

digit 0: "1" for vector, "0" for no vector

digit 1: "1" for focus, "0" for defocus (focus is used for points or vectors, defocus with characters)

digit 2: "1" for dim, "0" for bright

digit 3: "1" for intensify charactron, "0" for not.

digit 4: "1" for intensify typotron, "0" for not.

4. Having done this, one may then give the special charactron-typotron display instruction ch, binary 01110; at the address indicated by the address part of this instruction one must have stored a word which gives in digits 10-15 the matrix position desired (see Appendix: Charactron and Typotron Codes, MIT Matrix Mod. VI, p.53), and digits 0-9 the "format" position to determine electrostatic subdeflection. The positive binary number in digits 10-12 selects the row (y-position) of

the matrix (see drawing referred to above); that in digits 13-15 selects the column (x-position) in this matrix. The signed binary number in digits 0-4 similarly selects y-position in the 31x 31 electrostatic "format" square; that in digits 5-9 the analogous x-position.

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.

B. The Typotron technique is like that for the charactron but simpler by certain omissions:

1. No magnetic deflection is used;

2. No vectors can be displayed;

3. No control information in AC is used EXCEPT "intensify typotron" ("1" in digit 4).

Character selection and electrostatic deflection are done in one word as described in paragraph 4 above.

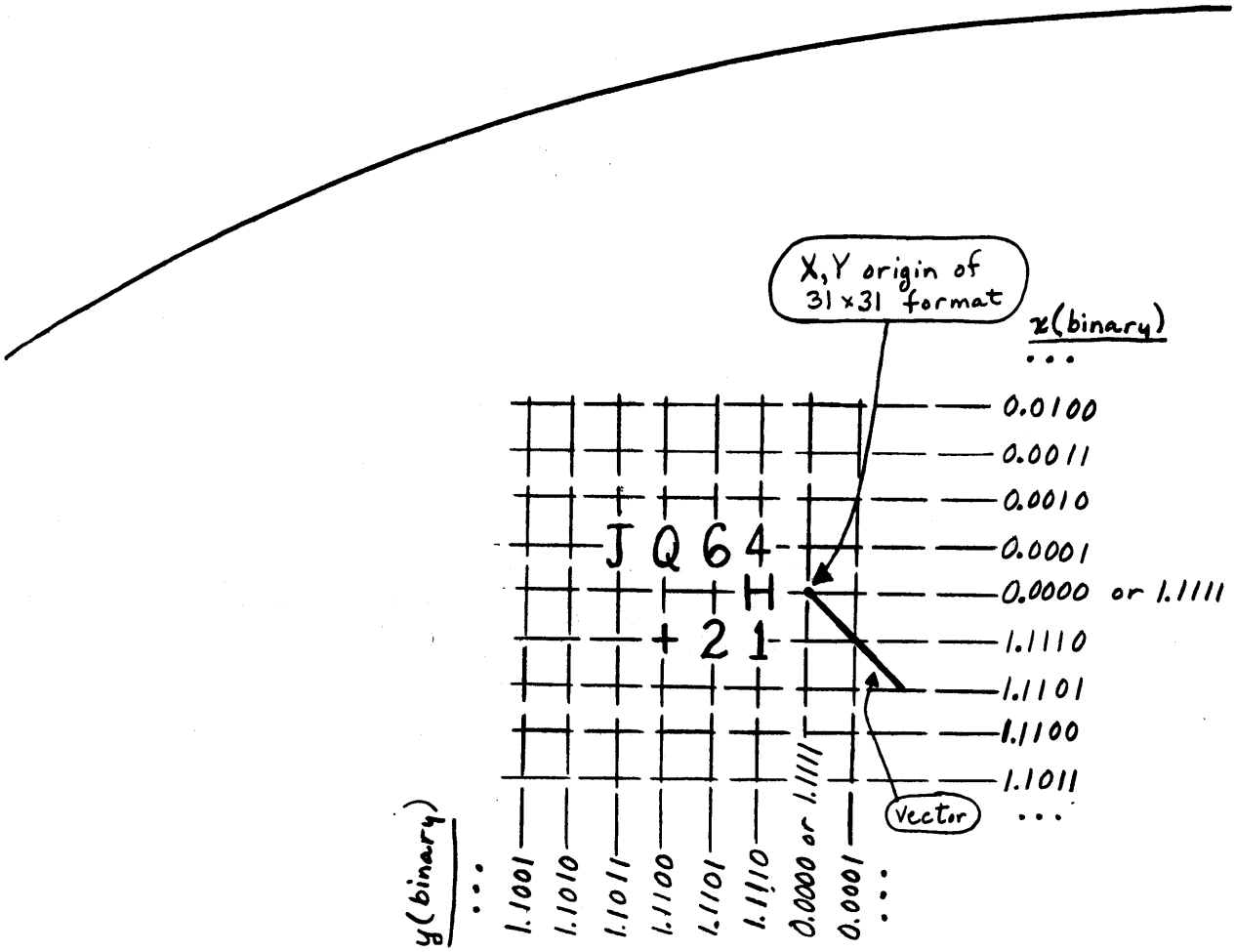
For ready reference, the possible "control words" to be put in AC are listed below:

Function		Binary	Octal
Charactron:	bright vector	1.1010.....	1.50000
	dim vector	1.1110.....	1.70000
	bright point	0.1010.....	0.50000
	dim point	0.1110.....	0.70000
	bright character	0.0010.....	0.10000
	dim character	0.0110.....	0.30000
Typotron:	character	0.0001.....	0.04000

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

```
ca    RC X
sof   0
st    61   (LR1)          } Set up X deflection
sof   1
ca    RC Y
sof   0
st    62   (LR2)          } Set up Y deflection
sof   1
ca    RC   0.001000 00 1110011   } Set up vector
                x        y
sof   0
st    65   (LR5)
sof   1
ca    RC    1.1010 ....          } Obtain control word (vector-
                                   focus-bright-Charactron)
ch    RC    0.0000 00000 001100  } Display vector
                x     y    vector
```

```
ca  RC  0.0010  .....                                  } Obtain control word
                                                       (character-defocus-bright-
                                                        charactron)
ch  RC  0.0001  11010  111 100  (J)                    }
            x       y  character                       }
ch  RC  0.0001  11011  111 000  (Q)                    }
ch  RC  0.0001  11100  101 101  (6)                    }
ch  RC  0.0001  11101  101 100  (4)                    }
ch  RC  0.0000  11101  011 010  (H)                    } Display 8 characters at
ch  RC  1.1110  11011  010 000  (+)                    }   specified locations
ch  RC  1.1110  11100  101 011  (2)                    }
ch  RC  1.1110  11101  100 010  (1)                    }
```



SECTION V. INPUT AND BASIC CONVERSION PROGRAMS

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 storage, 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.

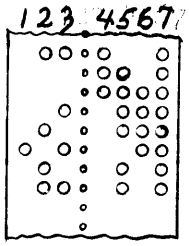
PUNCHED PAPER TAPE AND TAPE READERS

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.

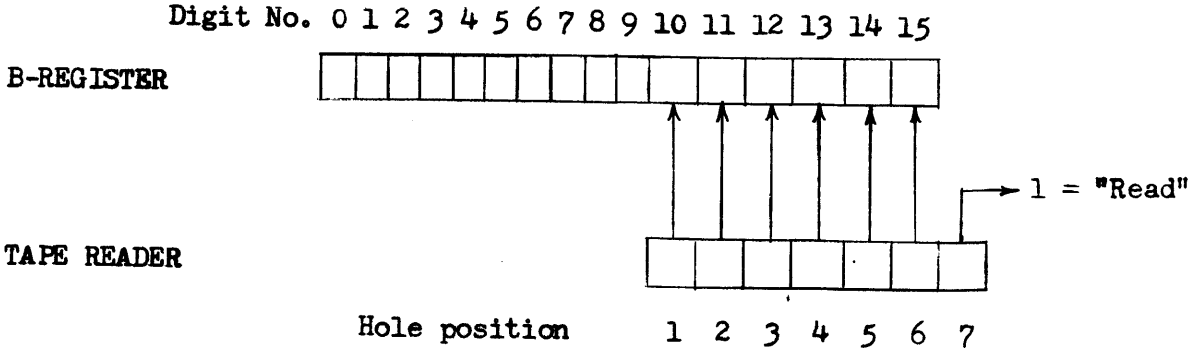
A portion of paper tape looks like this:

hole number



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 alighnment with the feed holes.

The paper tape in current use is of a medium weight blue paper to provide both durability required for repeated handling, and opacity required by the photoelectric reader. 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".

BASIC CONVERSION PROGRAM

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 address 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 succeeding 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 in a succeeding section.

Direct Read-In Conversion. The reader may have discerned that it is in theory not necessary to convert every program which is to be put into the computer. The conversion program could be arranged to put the converted program directly in storage, 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.

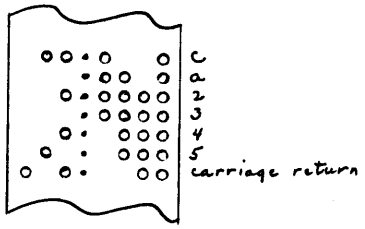
2. Check to see that the rotary switch behind the carriage is set to "Normal Print", that the left and right margins are at 4 and 79, and that tabular stops are positioned at 14, 27, 40, 53, and 66. (Margin and Tab stop settings do not apply to Barta Tape Room.)

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.

Address 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 storage 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 address 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 lowercase 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 2-2306

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. If for any reason the starting address indicator is entirely omitted, no undesirable consequences will ensue. No starting address will be punched on the tape, however, and the conversion program will continue trying to read and convert more standard tape.

Terminal Characters. Each heading, indicator, instruction, or number must be terminated by at least one carriage return or tab. (For an address 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 1 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.

```
TAPE 452, Smith
OCTAL

2-3440|  cr2000    ri20      ad3461    ra3444
         st3500    cr2035    su3542    su3511
         tn3440    su3542    tn3540    ad3501
         tr3441
2-3500|  0.00160   0.00260
2-3512|                      1.00000

START AT 2-3440


TAPE 463, Jones
DECIMAL

1-0|     ca31      ad37      st58      su122     mh122
         mh123     st59      ca31      su33      st33
         tr0
1-30|    0.17965   +.0003    -.9722    1.77777   0.01010
1-37|                        +0

START AT 1-6
```

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: if without manual moving of the carriage the tape prints an acceptable copy, the tape is valid; that is, there are no mistakes possible on tape that do not show on the typewritten copy when printed from tape. There is one unfortunate exception to the rule: the omission of a punch in the 7th hole position will not show on the print, but will cause the accompanying character to be ignored by the Conversion Program!

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 BASIC CONVERSION PROGRAM

Preparations. 1. Read in the Conversion Program via the 4-6-6 Input Program (refer to p.46 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 instructions 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 "Program Alarm".

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

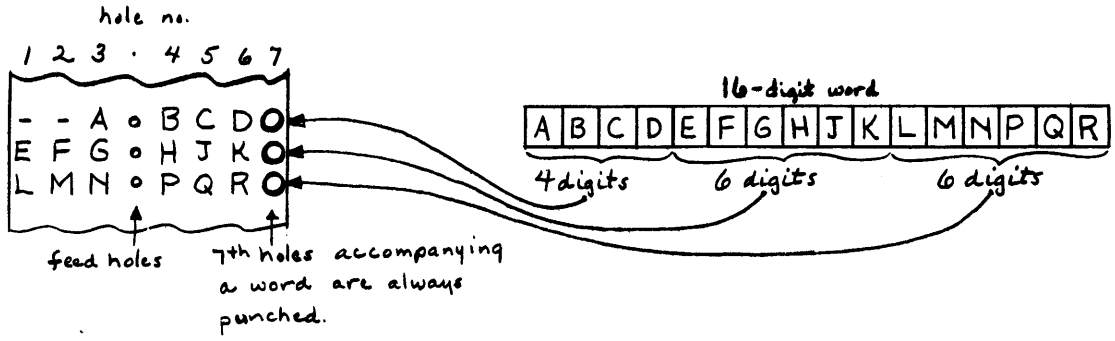
1.00001	Indeterminate error.
1.00002	Improper use of point.
0.00003	Improper character in decimal integer or fraction.
0.00004	Excessive magnitude of decimal integer.
0.00005	Improper number of digits in decimal fraction.
0.00006	No such instruction or indicator word.
0.00007	Improper character in address 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	Address assignment has become excessive.
1.00014	Improper character in start-over address.
1.00015	Decimal digits in octal field number of address assignment.
1.00016	" " " " " " " start-over address.
1.00017	Excessive magnitude of field number of address assignment.
1.00020	" " " " " " start-over address.
1.00021	Decimal digits in address part of octal address assignment.
1.00022	" " " " " " " start-over address.
0.00023	" " " " " " " instruction.
1.00024	Excessive magnitude of address part of address assignment.
1.00025	" " " " " " start-over address.
0.00026	" " " " " " 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

Tape produced by the Basic Conversion Program 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 most recent key word.

A block, or group, of words to be stored (a "store block") begins with a pair of key words: sof a and st x, which designate that the first word of the block is to be stored in register x of field a, and that the block is 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, sof a and tro x, which will direct the computer to take the next instruction from register x of field a, the address of the first instruction of the main program.

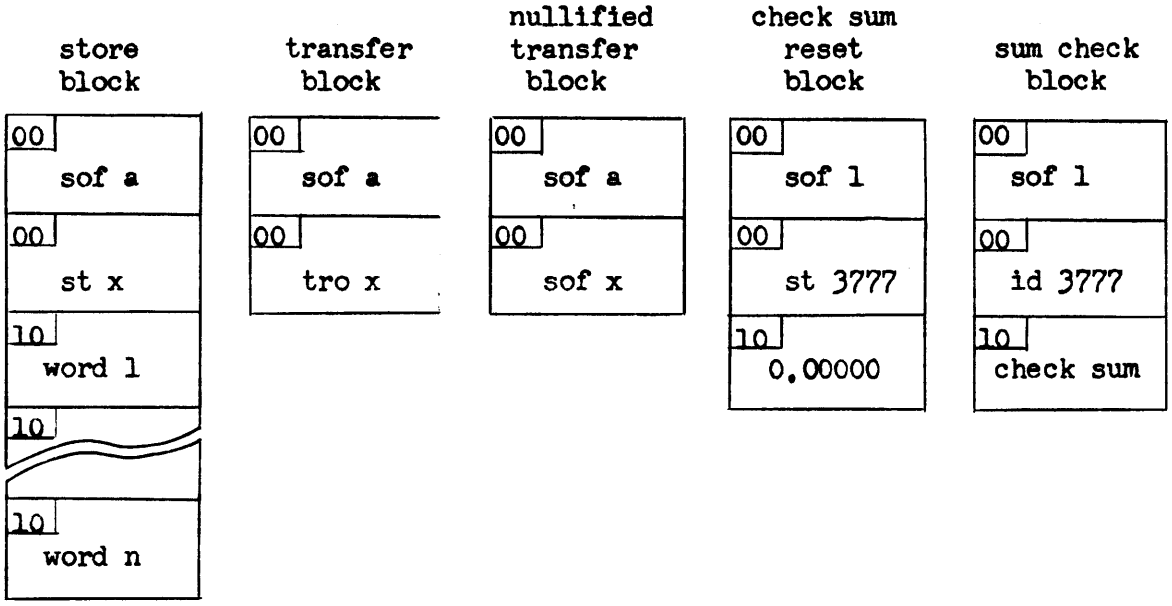
For a reason to be discussed later, it is sometimes desirable to "nullify" a transfer block: that is, to make it have no effect on the Input Program. This can be done by manually punching a single hole which will change the second key word tro x to sof x. The result is a "nullified transfer block".

A sum check block consists of two key words followed by a check sum. The key words are sof 1 and id 3777 (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. Because of lack of additional flip-flop memory registers in Panel Storage, register 1-3777 in core memory is used as temporary storage for the check sum accumulated by the Input Program.

Also 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

The input program 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 except "Overflow" 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
Revised 12-18-53

Notes:

Read-in errors will result in Identity Check alarm. Defective tapes may be read in by suppressing the Identity Check alarm. "Transfer block" on tape can be nullified by adding a hole which changes tro to sof. Tapes can be read in to any memory field. Program fits entirely on plugboard; no switches need be set up in toggle-switch storage. Unconditional stop after read-in; restart button will transfer control to main program. Directive 00 indicates key word; 10 (value=2^{-15}) indicates word to be acted upon in accordance with previous key word.

	Reg.	Contents	
Start →	40	cr 2020	leave AC & BR clear
	41	ri 32	read in 3 lines of tape
	42	ri 32	
	43	ri 22	
	44	tn 60	neg. if word to be stored, or check sum
	45	cr 2036	store key word temporarily
	46	st 62	
	47	tn 51	neg. if key word is sof or tro
	50	tr 72	
	51	cr 2036	examine negative key word; neg. again if sof
	52	tn 55	
	53	ha 1	key word is tro. Stop. "Restart" will transfer control to main program.
	54	tr 61	
	55	ca 62	
	56	st 61	key word is sof. Store it in final location.
	57	tr 72	
	60	cr 2036	execute pair of key words with check sum or word-to-be-stored held in AC
	61	(sofa)	
	62	(st x, id 3777, tro x)	temp. storage
	63	cr 20	add one to st instruction in preparation for the next word to be stored (holding in BR the word just treated)
	64	sof 0	
	65	ca 53	
	66	ad 62	
	67	st 62	
	70	cr 2020	
	71	ad 53	add word and directive to check sum (directive "10" has value = 2^{-15})
	72	sof 1	
	73	ad 3777	
	74	to 75	
	75	st 3777	
	76	sof 0	
	77	tr 40	

This page available separately as drawing SA-55786-2

Signed: Philip R. Bagley
Philip R. Bagley

Approved: W. Ogden Jr.
Warner Ogden, Jr.

PRB/rb

Drawing: D-47039-1(A-Reduction),
following p. 5.

APPENDIX

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 x	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

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

A. If digit 5 is a "1", give a program 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 AC0 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.
G. If digit 5 is a "0", index camera; if a "1", erase Typotron.

This page available separately as drawing SA-56946-3. 4-15-54

"FL" FLEXOWRITER CODE
Alphanumerical Sequence

Lower Case	Upper Case	Character 123456	Octal Value	Decimal Value
a	A	000110	6	(6)
b	B	110010	62	(50)
c	C	011100	34	(28)
d	D	010010	22	(18)
e	E	000010	2	(2)
f	F	011010	32	(26)
g	G	110100	64	(52)
h	H	101000	50	(40)
i	I	001100	14	(12)
j	J	010110	26	(22)
k	K	011110	36	(30)
l	L	100100	44	(36)
m	M	111000	70	(56)
n	N	011000	30	(24)
o	O	110000	60	(48)
p	P	101100	54	(44)
q	Q	101110	56	(46)
r	R	010100	24	(20)
s	S	001010	12	(10)
t	T	100000	40	(32)
u	U	001110	16	(14)
v	V	11100	74	(60)
w	W	100110	46	(38)
x	X	111010	72	(58)
y	Y	101010	52	(42)
z	Z	100010	42	(34)

Lower Case	Upper Case	Character 123456	Octal Value	Decimal Value
0	°	111110	76	(62)
1	¹	010101	25	(21)
2	²	001111	17	(15)
3	³	000111	7	(7)
4	⁴	001011	13	(11)
5	⁵	010011	23	(19)
6	⁶	011011	33	(27)
7	⁷	010111	27	(23)
8	⁸	000011	3	(3)
9	⁹	110110	66	(54)
|	_	000101	5	(5)
space bar		001000	10	(8)
=	•	001001	11	(9)
+	/	001101	15	(13)
color change		010000	20	(16)
.	)	010001	21	(17)
,	(	011001	31	(25)
-	-	011101	35	(29)
back space		100011	43	(35)
tabulation		100101	45	(37)
carr. return		101001	51	(41)
stop		110001	61	(49)
upper case		111001	71	(57)
lower case		111101	75	(61)
nullify		111111	77	(63)

This page available separately as drawing SA-56942

11-10-53

"FL" FLEXOWRITER CODE
Binary Numerical Sequence

Octal Value	Decimal Value	Character 123456	Lower Case	Upper Case
0	(0)	000000	not used	
1	(1)	000001	not used	
2	(2)	000010	e	E
3	(3)	000011	8	8
4	(4)	000100	not used	
5	(5)	000101	|	_
6	(6)	000110	a	A
7	(7)	000111	3	3
10	(8)	001000	space bar	
11	(9)	001001	=	.
12	(10)	001010	s	S
13	(11)	001011	4	4
14	(12)	001100	i	I
15	(13)	001101	+	/
16	(14)	001110	u	U
17	(15)	001111	2	2
20	(16)	010000	color change	
21	(17)	010001	.	)
22	(18)	010010	d	D
23	(19)	010011	5	5
24	(20)	010100	r	R
25	(21)	010101	1	1
26	(22)	010110	j	J
27	(23)	010111	7	7
30	(24)	011000	n	N
31	(25)	011001	,	(
32	(26)	011010	f	F
33	(27)	011011	6	6
34	(28)	011100	c	C
35	(29)	011101	-	"
36	(30)	011110	k	K
37	(31)	011111	not used	

This page available separately as drawing SA-56943-1

Octal Value	Decimal Value	Character 123456	Lower Case	Upper Case
40	(32)	100000	t	T
41	(33)	100001	not used	
42	(34)	100010	z	Z
43	(35)	100011	back space	
44	(36)	100100	l	L
45	(37)	100101	tabulation	
46	(38)	100110	w	W
47	(39)	100111	not used	
50	(40)	101000	h	H
51	(41)	101001	carr. return	
52	(42)	101010	y	Y
53	(43)	101011	not used	
54	(44)	101100	p	P
55	(45)	101101	not used	
56	(46)	101110	q	Q
57	(47)	101111	not used	
60	(48)	110000	o	O
61	(49)	110001	stop	
62	(50)	110010	b	B
63	(51)	110011	not used	
64	(52)	110100	g	G
65	(53)	110101	not used	
66	(54)	110110	9	9
67	(55)	110111	not used	
70	(56)	111000	m	M
71	(57)	111001	upper case	
72	(58)	111010	x	X
73	(59)	111011	not used	
74	(60)	111100	v	V
75	(61)	111101	lower case	
76	(62)	111110	0	0
77	(63)	111111	nullify	

11-10-53

PRINTER CODES FOR IBM PRINTER, MTC

Character	Octal Code	Decimal Value
ø	00	(0)
1	01	(1)
2	02	(2)
3	03	(3)
4	04	(4)
5	05	(5)
6	06	(6)
7	07	(7)
8	10	(8)
9	11	(9)
A	12	(10)
B	13	(11)
C	14	(12)
D	15	(13)
E	16	(14)
F	17	(15)
G	20	(16)
H	21	(17)
I	22	(18)
J	23	(19)
K	24	(20)
L	25	(21)
M	26	(22)
N	27	(23)

Character	Octal Code	Decimal Value
O	30	(24)
P	31	(25)
Q	32	(26)
R	33	(27)
S	34	(28)
T	35	(29)
U	36	(30)
V	37	(31)
W	40	(32)
X	41	(33)
Y	42	(34)
Z	43	(35)
period	44	(36)
+	45	(37)
-	46	(38)
=	47	(39)
/	50	(40)
comma	51	(41)
$	52	(42)
apostrophe	53	(43)
space	54	(44)
tab.	55	(45)
car. return	56	(46)

FONT:

ø 1 2 3 4 5 6 7 8 9
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 . + - =
/ , $ '

This page available separately as drawing SA-56944

11-10-53

CHARACTRON and TYPOTRON CODES
MIT MATRIX MOD. VI
Alphanumerical Sequence

Character	Octal Code	Decimal Value	Character	Octal Code	Decimal Value
A	36	(30)	n	37	(31)
B	23	(19)	r	47	(39)
C	31	(25)	t	57	(47)
D	51	(41)	u	67	(55)
E	24	(20)	0	52	(42)
F	41	(33)	1	42	(34)
G	64	(52)	2	53	(43)
H	32	(26)	3	43	(35)
I	33	(27)	4	54	(44)
J	74	(60)	5	44	(36)
K	22	(18)	6	55	(45)
L	62	(50)	7	45	(37)
M	75	(61)	8	56	(46)
N	63	(51)	9	46	(38)
O	52	(42)	.	02	(2)
P	34	(28)	+	20	(16)
Q	70	(56)	✚	03	(3)
R	61	(49)	↑	10	(8)
S	65	(53)	✱	01	(1)
T	66	(54)	∠	11	(9)
U	35	(29)	↓	12	(10)
V	73	(59)	↳	13	(11)
W	76	(62)	●	14	(12)
X	71	(57)	▲	04	(4)
Y	72	(58)	□	25	(21)
Z	77	(63)	⊡	26	(22)
a	60	(48)	■	27	(23)
b	50	(40)	○	15	(13)
d	40	(32)	⊙	16	(14)
f	30	(24)	●	17	(15)
h	21	(17)	◇	05	(5)
			◈	06	(6)
VECTOR	14	(12)	◆	07	(7)
POINT	14	(12)	blank	00	(0)

VOCABULARY

MTC BASIC CONVERSION PROGRAM

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 Address Assignments (samples):

40 1-40 3-2021

Instructions (samples):

mh72 tro 2-2036 ca 1-0

Octal constants (samples):

0.12345 1.65432

Decimal fractions (samples):

+.9987 -.1234

Decimal integers (samples):

+1234 -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 storage 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 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.

This page available
separately as drawing SA-57300

TEMPORARY PRECAUTION TO BE OBSERVED WHEN PROGRAMMING FOR THE MAGNETIC DRUM.

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 wish to take the problem into account, the following discussion is appended.

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, except that the interval between registers 2047 and 0 is 20 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) can 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 x (namely: x - 128, x - 256, and x - 384).

BRIEF OPERATING GUIDE FOR MTC

Operating Controls. The controls important to the operation of normal (as distinguished from computer test) programs are the following:

1. Alarm indicator light (Parity, Inactivity, Identity Check, Program, Overflow, 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 Indicator 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, 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. This is 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 the arithmetic element to be cleared, and the computer directed to take its next instruction from register 0 or 40 (octal) of Panel Memory.

5. The "RESTART" pushbutton causes the computer to continue in normal fashion from the point at which it was most recently stopped.

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.

FF Indicator Lights. The contents of all flip-flop circuits in the computer are displayed on the FF 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 operating the 4-6-6- Input Program, p. 46.)

To start a program at any time, 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.

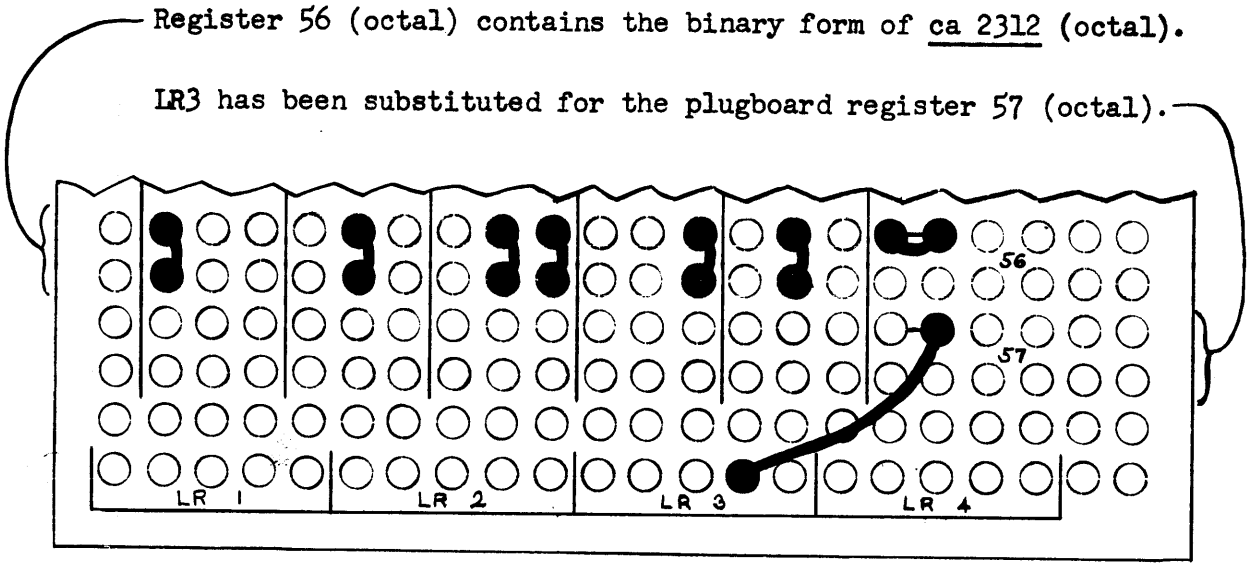
All of the terminal equipment (scopes, camera, punch, printer) which is used by a program must be turned on prior to starting the program.

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 no program will operate properly if it tries to 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 specifies that the associated contains a word wired by jumpers or which of five Live Registers is to be substituted for that plugboard register. (Reminder: LR3, LR4, and LR5 can be read into but not out of.)

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



Debugging a Program. In a complicated program, programming errors are almost unavoidable. The art of locating programming errors in a program while it is in the computer depends principally on the ingenuity of the operator. An auxiliary program has been written to assist the operator in gaining information about his program. The details of this program are covered in Memorandum M-2787, "MTC U-68 Utility and Analysis Program."

Distribution list:

J. N. Ackley
C. W. Adams
H. E. Anderson
J. A. Arnow
D. N. Arden
W. S. Attridge
P. R. Bagley
H. D. Benington
S. Best
H. W. Boyd
W. Canty
L. Chiodi
W. A. Clark
D. Combelic
O. T. Conant
J. D. Crane
J. W. Delmege
M. S. Demurjian
H. H. Denman
R. S. diNolfo
M. A. Epstein
R. S. Fallows
B. G. Farley
J. M. Frankovich
E. K. Gates
C. H. Gaudette
R. H. Gerhardt
C. C. Grandy
A. J. Grennell
G. Harris (Group 22)
I. Hazel
F. E. Heart
F. C. Helwig
H. B. Henegar
W. A. Hosier
H. D. Houser
A. D. Hughes
R. A. Hughes
R. C. Jeffrey
S. Knapp
B. J. Kollet
E. S. Kopley
A. P. Kromer
J. A. Levenson
W. Lone, Jr.
M. C. Mackey
A. A. Mathiasen
R. P. Mayer
P. J. Messenheimer
H. D. Neumann
W. Ogden, Jr.
R. Pacl
H. C. Peterson
J. Piro
J. D. Porter
R. Porter
P. G. Quinn
G. A. Rawling
D. Ross (Bldg. 32)
J. W. Salvato
J. Schallerer
H. H. Seward
A. Siegal
B. Stahl
J. L. Sullivan
L. L. Sutro
A. Vanderburgh, Jr.
R. F. vonBuelow
F. A. Webster
J. I. Woolf
C. A. Zraket

Library (50)

Tape Room, Barta (5)