DATATRON
Electronic Data Processing Systems

HANDBOOK
central computer
This handbook supersedes and replaces previous editions of Bulletin 3010, Summary Command List, and Bulletin 3040A (Programming and Coding Manual). Symbols and nomenclature used to designate commands conform to the revised standard practice adopted in March, 1956.

First printing March, 1956
# TABLE OF CONTENTS

**DATATRON DIGITAL COMPUTER MODEL 204**
- General ........................................ 1
- DATATRON Electronic Data Processing Systems ..... 1
- Components of the DATATRON .................. 1

**OPERATING CHARACTERISTICS OF THE DATATRON**
- How Information is Represented in the DATATRON 2
- How Information is Stored in the DATATRON .... 2
- Location of Information on the Magnetic Drum 3
- Operation of Quick Access Storage Loops ....... 3
- Electronic Registers ............................ 4
- Arithmetic Registers ............................ 4
- Command Structure ............................. 4
- C Register .................................... 5
- Operation Sequence ................................ 5
- Operation Cycle .................................. 5
- B Register .................................... 6
- Decimal Point .................................. 6
- Overflow ....................................... 6
- Checking Facilities .............................. 7

**COMPUTER COMMANDS**
- Arithmetic .................................... 8
- Manipulation and Transfer of Information .... 12
- Decision Making and Branching ............... 16
- Using the B Register ........................... 20

**GENERAL PROGRAMMING PROCEDURES**
- Scaling ....................................... 22
- Command Modification and Cycling .......... 22
- Use of Quick Access Loops .................... 23
- General Rules for Use of the Quick Access Loops 23
- Data Editing ................................... 23
- Table Look-Up .................................. 25

**OPERATION AND CONTROLS**
- Description of the Operating Controls on the DATATRON 29
- Manipulation of the Contents of the Registers 29
- Operating Instructions .......................... 29
Figure 1 Power Control — Computer Cabinet — Motor-Generator
GENERAL

Electronic data processing systems have five components—input, storage (working and auxiliary), arithmetic, control, and output. This handbook describes the characteristics and explains the use of the DATATRON Digital Computer Model 204 (called the DATATRON). This unit, which consists of the Computer Cabinet, the Power Control, and the Motor-Generator (Figure 1), contains the working storage, arithmetic, and control components of a complete system. A Stabiline Voltage Regulator (not shown in Figure 1) is also included in the basic DATATRON.

DATATRON ELECTRONIC DATA PROCESSING SYSTEMS

The DATATRON is a general purpose, internally programmed, decimal, electronic computer with magnetic drum storage. It is the heart, or central controlling and processing unit, of an electronic data processing system which accomplishes the functions of:

1. Accepting data directly from punched cards, punched tape, magnetic tape, keyboard—employing input units singly or in multiple.
2. Selecting from magnetic tape files the historical or reference records necessary to process data.
3. Processing data—comparing, computing, analyzing, sorting, classifying as required—in obedience to a series of commands (instructions) which have previously been stored in the system (stored program).
4. Bringing up to date the historical or reference records maintained on magnetic tape, and returning the up-dated records to magnetic tape.
5. Transmitting required information directly into punched cards, punched tape, magnetic tape, printed documents, visual indications—employing output units singly or in multiple.

As a result of its ability to control data processing systems of wide scope, and because of its economical and reliable operation, the DATATRON has been applied effectively to a wide range of commercial, manufacturing, scientific and engineering problems.

In speed of computer operation, the DATATRON is classed below the very large-scale electronic data processors—and considerably above card-programmed computers, other externally programmed computers, and the small, stored program computers.

In capacity and data processing capability, the DATATRON (as the central unit in a system) approaches large-scale systems in power and ability to produce an effective and economical flow of work.

COMPONENTS OF THE DATATRON

The Computer Cabinet contains the arithmetic and control units (see Figure 1). The center section contains the magnetic drum working storage and the Control Panel. Switches, indicators, and displays required by the operator are mounted on this panel.

The Motor-Generator converts electric power as furnished to the installation into three stable levels of direct current voltage. This unit may be installed at some distance from the Computer Cabinet, or it may be installed, if properly soundproofed, in the immediate vicinity of the other components of the DATATRON.

The Power Control converts the output of the Motor-Generator into the eight highly stable levels of direct current voltage required by the DATATRON. It contains controls and meters for monitoring these voltages (a maintenance function), and the controls for starting and shutting down the DATATRON.

The Voltage Regulator refines alternating current voltage as supplied to the installation, furnishing a regulated power supply to the vacuum tube filaments. The output of the Voltage Regulator is routed through the Power Control on its way to the Computer Cabinet.
HOW INFORMATION IS REPRESENTED IN THE DATATRON

Information is represented in the DATATRON as fixed length numbers, each of which contains ten decimal digits. Each ten digit number is preceded by an additional digit (Figure 2) which
— represents the algebraic sign of the number, or
— is sometimes used to control machine operation, or
— is an arbitrary zero having no special significance.

Each of these eleven digit units of information, called a word, may represent numerical data, alphabetic data, alphanumeric data, or a command which the DATATRON is to obey. For example:

0 4259 64 4955 represents the number + 4 259 644 955
0 4259 64 4955 represents the noun B R U I N
0 4259 64 4955 represents the command “Clear the A Register. Add the contents of storage cell 4955.”
0 4259 86 4955 represents Part Number B R 6 I N

The position of the word 0 4259 64 4955 in storage, in relation to the commands (stored program) which the DATATRON is to obey, determines which of the three possible interpretations illustrated above will be applied to the word.

0 4 2 5 9 6 4 4 9 5 5

Figure 2

The eleven digit word is treated as a unit by the DATATRON. It is stored as a unit, and it is manipulated as a unit. However, if it is necessary to break up a word into smaller units of information, or to combine words into longer records, this can be done by placing the proper series of commands in the DATATRON.

HOW INFORMATION IS STORED IN THE DATATRON

Over 4000 words of information are stored in the DATATRON on the surface of a large-capacity magnetic drum which revolves at 3570 revolutions per minute. This unusual storage capacity makes possible
— adequate reference to data,
— adequate facility for classification of data,
— convenient use of long programs,
— convenient insertion of temporary programs for “spot” analysis,
— improved internal sorting techniques, and
— a reduction, in many cases, in the number of times the same data must be fed through the central data processor to secure the desired results.

Once placed on the drum, information is retained (whether or not the power is turned on) until it is “erased” by writing new information on the drum over the old information.

Only the digits zero and one are represented on the surface of the magnetic drum—and this representation is made by magnetizing a small spot on the drum for each digit. All zero spots are magnetized in the same direction of polarity, and all one spots are magnetized alike in the opposite direction. Four such spots (called bits of information or binary digits) are used to represent one decimal digit. In this scheme of representation (binary-coded decimal), one bit of information is assigned the value 1, the second bit is assigned the value 2, the third bit is assigned the value 4, and the fourth bit is assigned the value 8. Decimal digits are represented according to the following table:

<table>
<thead>
<tr>
<th>Bit</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 Bit</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4 Bit</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2 Bit</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1 Bit</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Write heads and read heads are mounted on the magnetic drum casing (Figure 3). As the drum cylinder revolves inside the casing, the surface of the drum passes these heads. The function of each write head is to place
information on the surface of the drum by magnetizing four spots at a time according to the code tabulated above. The function of each read head is to interpret the pattern of magnetic spots on the surface of the drum, four bits of information at a time, thus making the information available for use.

LOCATION OF INFORMATION ON THE MAGNETIC DRUM

A space on the drum large enough to write the contents of exactly one word is called a storage cell. Storage cells are arranged in bands which extend around the circumference of the magnetic drum. Each band consists of four tracks of magnetized spots (Figure 4), making possible the use of the binary-coded decimal scheme of representing digits. Four zeros, one in each of the tracks, separate each word from its adjoining words. Associated with each band is a read head and a write head, or a combination read-write head.

Figure 4 Information Stored on a Drum

Each storage cell on the magnetic drum has its own address, a four digit number which identifies the cell and specifies its location. The top twenty bands on the magnetic drum (Figure 5) each contain 200 words, a total of 4000 storage cells being located in the portion of the magnetic drum called main storage. The addresses of these cells are the numbers 0000 through 3999. The bottom four bands on the drum each contain exactly twenty different words. These are the four quick access loops which make up the DATATRON's high speed storage or loop storage. The addresses of the cells in the loops are the numbers 4000 through 7999. However, since each loop contains twenty words, cell 4020 contains the same word as cell 4000, cell 5569 contains the same word as cell 5009, cell 6738 contains the same word as cell 6018, etc. The larger address numbers are sometimes used to achieve desirable programming effects.

OPERATION OF QUICK ACCESS STORAGE LOOPS

Each main storage band has associated with it one combination read-write head (Figure 6). A word stored in a main storage cell passes the read-write head only once in every revolution of the magnetic drum. A word stored in a main storage cell is available for use, then, once in every revolution of the drum. The access time (or waiting time) for this word can vary from zero to 0.017 seconds (17 milliseconds). The average access time for the word is 8.5 milliseconds, the time for a half-revolution of the drum.

Each quick access loop has a separate read head and, twenty words distant from this head along the drum circumference, a separate write head (Figure 7). Since a complete band around the magnetic drum contains 200 words, these two heads are one-tenth of the drum circumference apart.

As each word passes under the read head, it is always immediately rewritten twenty words back along the drum circumference. If a block of twenty words is placed in a quick access loop, this continual process of reading and writing will duplicate the twenty words in ten locations around the drum—in the first revolution of the magnetic drum following the transfer of information into the loop.

A word stored in one of the cells of a quick access loop is available for use once in every one-tenth of a drum revolution, or ten times in every revolution. In effect, the quick access loops supply data and commands at the same rate as if the magnetic drum were revolving at 35,700 revolutions per minute. The access time for a word in a loop can vary from zero to 1.7 milliseconds. The average access time for a word stored in a loop is 0.85 milliseconds.
In most applications, DATATRON commands are transferred from main storage into the quick access loops before the execution of the commands. Similarly, data and intermediate results are normally stored in the quick access loops, or transferred from main storage into the quick access loops.

To accomplish the necessary manipulation of information, block transfer commands are used. These commands move twenty words at a time from main storage to loop, or from loop to main storage, at the rate of 1.7 milliseconds per block of twenty words. This is the amount of time required for twenty words to pass by a read head. The actual transfer of each digit is almost instantaneous.

Words transferred from main storage to a loop remain (in unaltered form) in main storage, facilitating the process of making memo entries in records. Words transferred from a loop to main storage remain (in unaltered form) in the loop.

DATATRON programs are written to maintain a continuous flow of data and commands through the loops. Thus, the DATATRON maintains the high rate of processing associated with optimum (or minimal access) programming, but retains the reliability inherent in a conservative speed of drum revolution.

**ELECTRONIC REGISTERS**

On the magnetic drum, each decimal digit is represented by a combination of four magnetized spots, each spot being an indicator of either zero or one. This method of representing information has proven to be extremely reliable.

An electronic circuit, called a flip-flop, can also represent zero or one by being in one of two possible states—either "low" or "high." Several registers, or storage cells with zero access time, use the flip-flop circuit to store information. In these registers, each decimal digit is represented by four flip-flops. Just as in the case of the magnetized spots on the magnetic drum, relative values are assigned to each flip-flop. The first flip-flop is assigned the value 1, the second flip-flop is assigned the value 2, the third flip-flop is assigned the value 4, and the fourth flip-flop is assigned the value 8. Decimal digits are represented in electronic registers according to the table of combinations used to represent decimal digits on the drum. (Refer to How Information is Stored in the DATATRON.)

**ARITHMETIC REGISTERS**

Three electronic registers are used to contain numbers involved in computation and data processing (Figure 8).

- **A Register** holds an eleven digit word. This register is an accumulator in which the results of all arithmetic operations appear.
- **R Register** holds ten decimal digits. This register is primarily an extension of the A Register. However, multiplication and division are the only arithmetic operations which affect the R Register.

**COMMAND STRUCTURE**

A DATATRON command is made up of three parts (Figure 9):

- (a) the four digit address—which designates the location of the storage cell referred to during execution of the command;
- (b) The two digit order—which designates the specific operation to be performed;
- (c) the five control digits—which designate variations in the execution of the command.
C REGISTER

C Register receives each command from the magnetic drum through the D Register (Figure 10). The function of this register is to start the operation of the control component of the DATATRON.

The C Register is composed of three sub-registers (reading from left to right):

- **Order Register** holds the two digits which designate the specific operation to be performed.
- **Address Register** holds the four digits which designate the location of the storage cell referred to during execution of the command. The contents of the Order Register and the Address Register, together, are the same as the six right hand digits of the command word as it appears in the D Register and on the magnetic drum.
- **Control Counter** holds the four digits which specify the address of the next command which will be executed—after the completion of the operation specified in the Order Register and the Address Register.

OPERATION SEQUENCE

In normal, continuous operation, commands are executed in the order in which they are stored on the magnetic drum. Thus, if commands are stored in storage cells 1000, 1001, and 1002, the command stored in cell 1001 will be executed after the command stored in cell 1000 and the command stored in cell 1002 will be executed after the command stored in cell 1001.

The Control Counter counts up 1 after each command comes into the C Register so that the next command will be read from the next cell. In the preceding example, when the command stored in cell 1000 is being executed, the Control Counter will read 1001. When the command stored in cell 1001 is being executed, the Control Counter will read 1002 (Figure 10).

To change this normal method of sequential operation, change of control commands are used. These commands may be used to alter the sequence of command execution arbitrarily—in which case they are unconditional changes of control. A similar series of commands may be used to alter the sequence of command execution only in response to the presence of a machine condition (see Overflow, below). These conditional changes of control are used for decision-making or branching.

Instead of allowing the Control Counter to count up 1, the change of control commands insert their address digits into the Control Counter, and thus specify the next command to be executed.

OPERATION CYCLE

As has been noted, the D Register sends command words along one path, and data words along another.

In the **fetch phase** of the operation cycle (Figure 11), the command word located in the storage cell specified in the Control Counter is brought from the magnetic drum, through the D Register, through the Adder, to the C Register.

In the **execute phase** of the operation cycle (Figure 12), the data word specified in the command just fetched is brought from the magnetic drum, through the D Register, through the Adder (where an arithmetic operation takes place) to the A Register.

During DATATRON operation, the fetch phase and the execute phase alternate as the operation cycle repeats.

---

Figure 10 Action of Control Counter
**B REGISTER**

The B Register holds any four decimal digits from 0000 to 9999. These digits can be added to the address digits of a command word as the command goes through the Adder to the C Register (Figure 13).

The addition of the contents of the B Register to a command (command modification) is signalled by the first control digit of the command word, when the word reaches the D Register. If the digit is 1, the contents of the B Register are added. If the control digit of the command word is 0, the contents of the B Register are not added (see Figure 11).

The contents of the B Register can be increased by one, or decreased by one, during the execution of a series of commands. When the series of commands is repeated many times, the B Register can serve, in this case, as a tallying device.

**DECIMAL POINT**

Inside the DATATRON, a decimal point is considered to be fixed at the left of each ten digit word stored on the magnetic drum or in the electronic registers.

The eleventh digit, at the left of the decimal point, represents the algebraic sign of numerical data (zero for plus and one for minus), or (in the case of a command word) is sometimes used to control machine operation, or (in the case of alphabetic or alphanumeric data) is an arbitrary zero having no special significance.

Outside the DATATRON, the decimal point may be located in its proper position (by programming) regardless of its internal position. For example:

<table>
<thead>
<tr>
<th>Internally</th>
<th>Externally</th>
<th>16.21</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1621 00 0000</td>
<td>0001 62 1000</td>
<td>may be represented as</td>
</tr>
<tr>
<td>0 0000 00 1621</td>
<td>0000 00 1621</td>
<td></td>
</tr>
</tbody>
</table>

When a number in the combined A Register and R Register is shifted left, the shifting command may tally the number of positions shifted and record this number in a register called the Special Counter. The contents of the Special Counter may later be added to or subtracted from the A Register.

**OVERFLOW**

Whenever the execution of a command produces a result which is too large to be inserted in the A Register, an overflow condition is set up in the DATATRON. This is the machine condition which can result in a conditional change of control (previously discussed in the handbook section, Operation Sequence). The presence of the overflow condition is determined as follows:

Indication to DATATRON — Overflow flip-flop is in a "high" state.

Indication to operator — Overflow light is ON.
EXAMPLE 1

Actual Arithmetic
0.9000 00 0000
+0.8000 00 0000
1.7000 00 0000*

*Carry produces overflow to the left.

DATATRON Arithmetic
0 9000 00 0000
0 8000 00 0000
0 7000 00 0000**

**Overflow indicates ON. Zero to the left of decimal point position represents plus sign.

EXAMPLE 2

Actual Arithmetic
3.0000 00 0000*
0.3000 00 0000/0.9000 00 0000

*Division of larger number produces overflow to the left.

DATATRON Arithmetic
0 0000 00 0000
0 3000 00 0000/0 9000 00 0000**

**Overflow indicates ON. Zero to the left of decimal point position represents plus sign.

The overflow condition may follow the arithmetic manipulation of the contents of the A Register. Overflow always follows the test for and detection of a difference between the algebraic sign of the A Register and the sign of a number brought from a storage cell for comparison.

When the possible appearance of an overflow is anticipated, a conditional change of control command is inserted in the program to allow the program to branch (take one of two possible alternate paths). When an unanticipated overflow occurs (a programming error) the DATATRON stops.

CHECKING FACILITIES

The DATATRON automatically stops upon the appearance of an unanticipated overflow (see Overflow).

An alarm light is turned on and computation is stopped by a forbidden combination (binary-coded decimal digits 10 through 15) in the A, B, D, and R Registers, the Address Register, the Control Counter, and the Shift Counter. Inspection of the register contents as indicated on the Control Panel indicates the failure location.

An alarm will stop machine operation if the storage cell counter does not contain 0 at the start of each drum revolution. This check prevents information from being recorded on or read from incorrect locations on the drum.

An audible alarm indicates excessive rise in exhaust air temperature in the computer cabinet and after a preset interval up to 15 minutes, dc voltages will be shut off if the temperature stays at or above a predetermined level.

A very extensive marginal checking system is available to maintenance personnel. This system makes it possible to vary voltages applied to each section of the DATATRON to induce errors caused by marginal components. The use of the marginal checking system greatly simplifies the operation of an effective preventive maintenance system.

Figure 14 Marginal Checking Panel
COMPUTER COMMANDS

This section defines the DATATRON commands available to the programmer and illustrates their use. Appendix I of this handbook contains a summary of these commands.

ARITHMETIC

Commands for Addition and Subtraction.

Addition and subtraction commands affect the A Register, but not the R Register.

The series of commands below illustrates the use of the add and subtract commands, and the effect that each command has on the A Register and the R Register.

Assume that:
1. Storage cell 1000 contains the number 0 2222 22 2222.
2. Storage cell 1001 contains the number 1 3333 33 3333.
3. The A Register contains the number 1 9874 53 1234.
4. The R Register contains the number 0000 560000.
5. Insert a 7 on Keyboard for Digit Add.

<table>
<thead>
<tr>
<th>Program</th>
<th>A Register</th>
<th>R Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAD 1000</td>
<td>0 2222 22 2222</td>
<td>0000 560000</td>
</tr>
<tr>
<td>AD 1001</td>
<td>1 1111 11 1111</td>
<td>0000 560000</td>
</tr>
<tr>
<td>ADA 1001*</td>
<td>0 2222 22 2222</td>
<td>0000 560000</td>
</tr>
<tr>
<td>CADA 1001*</td>
<td>0 3333 33 3333</td>
<td>0000 560000</td>
</tr>
<tr>
<td>CSU 1000</td>
<td>1 2222 22 2222</td>
<td>0000 560000</td>
</tr>
<tr>
<td>SU 1001</td>
<td>0 1111 11 1111</td>
<td>0000 560000</td>
</tr>
<tr>
<td>SUA 1001*</td>
<td>1 3333 33 3333</td>
<td>0000 560000</td>
</tr>
<tr>
<td>SUA 1001*</td>
<td>1 6666 66 6666</td>
<td>0000 560000</td>
</tr>
<tr>
<td>DAD 0000</td>
<td>1 6666 66 6659</td>
<td>0000 560000</td>
</tr>
</tbody>
</table>

*In addition and subtraction of absolute numbers, the number is treated as a positive number, regardless of its sign.

The condition of overflow in AD, ADA SU, SUA is possible and will appear as follows:

<table>
<thead>
<tr>
<th>Program</th>
<th>A Register</th>
<th>R Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>*SU 1000</td>
<td>1 2096 75 3456</td>
<td>0000 560000</td>
</tr>
<tr>
<td>SUA 1001</td>
<td>1 5430 08 6789</td>
<td>0000 560000</td>
</tr>
<tr>
<td>CADA 1001</td>
<td>0 3333 33 3333</td>
<td>0000 560000</td>
</tr>
<tr>
<td>ADA 1001</td>
<td>0 6666 66 6666</td>
<td>0000 560000</td>
</tr>
<tr>
<td>ADA 1001</td>
<td>0 9999 99 9999</td>
<td>0000 560000</td>
</tr>
<tr>
<td>*ADA 1001</td>
<td>0 3333 33 3332</td>
<td>0000 560000</td>
</tr>
</tbody>
</table>

*Overflow indicates ON.
Addition and Subtraction commands can be used in: Posting, Accumulating receipts, Debiting and Crediting accounts, and in general, Updating records.

**PROBLEM:** A store has four sections. Following each day's business the owner wants to know net receipts. Each section reports total receipts and amount of sales commissions.

**TO FIND:** Net Receipts

**SOLUTION:**

<table>
<thead>
<tr>
<th>Location</th>
<th>Loop</th>
<th>S</th>
<th>Control Digits</th>
<th>Operation</th>
<th>Operand Address</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0</td>
<td>0</td>
<td></td>
<td>CA0</td>
<td>1000</td>
<td>A = 0.0000 01 9432.</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
<td>1</td>
<td></td>
<td>SU1</td>
<td>1001</td>
<td>A = 0.0000 01 5546.</td>
</tr>
<tr>
<td>0002</td>
<td>2</td>
<td>2</td>
<td></td>
<td>AD2</td>
<td>1002</td>
<td>A = 0.0000 03 0749.</td>
</tr>
<tr>
<td>0003</td>
<td>3</td>
<td>3</td>
<td></td>
<td>SU3</td>
<td>1003</td>
<td>A = 0.0000 02 7709.</td>
</tr>
<tr>
<td>0004</td>
<td>4</td>
<td>4</td>
<td></td>
<td>AD4</td>
<td>1004</td>
<td>A = 0.0000 03 7074.</td>
</tr>
<tr>
<td>0005</td>
<td>5</td>
<td>5</td>
<td></td>
<td>SU5</td>
<td>1005</td>
<td>A = 0.0000 03 5203.</td>
</tr>
<tr>
<td>0006</td>
<td>6</td>
<td>6</td>
<td></td>
<td>AD6</td>
<td>1006</td>
<td>A = 0.0000 04 5355.</td>
</tr>
<tr>
<td>0007</td>
<td>7</td>
<td>7</td>
<td></td>
<td>SU7</td>
<td>1007</td>
<td>A = 0.0000 04 3325.</td>
</tr>
</tbody>
</table>

**ASSUME:** Information from sections located in memory cells:

- 1000 0 0000 01 9432 (Section 1 - Sales)
- 1001 0 0000 00 3886 (Section 1 - Commissions)
- 1002 0 0000 01 5203 (Section 2 - Sales)
- 1003 0 0000 00 3040 (Section 2 - Commissions)
- 1004 0 0000 00 9367 (Section 3 - Sales)
- 1005 0 0000 00 1873 (Section 3 - Commissions)
- 1006 0 0000 01 0152 (Section 4 - Sales)
- 1007 0 0000 00 2030 (Section 4 - Commissions)

**ANSWER:** Located in the A Register — $433.25
Commands for Multiplication and Division:

Multiplication and division commands affect both the A Register and the R Register.

The series of commands below illustrates the use of commands for multiplication and division, and the effect that each command has on the A Register and the R Register.

**MULTIPLICATION**

M
000p 60 xxx
Multiply the contents of xxx by the contents of the A Register. Insert the twenty digit product in the A Register and the R Register. The most significant digits are in the A Register.

MRO
000p 70 xxx
Multiply the contents of xxx by the contents of the A Register. Round the product to ten digits. Clear the R Register.

During the execution of the Multiply command, the R Register is cleared to permit the insertion of the least significant ten digits of the product.

The A Register will contain the proper algebraic sign of the product.

**DIVISION**

DIV
000p 61 xxx
Divide the twenty digit contents of the A Register and the R Register by the contents of xxx.

(a) If Overflow indicates ON, clear the A Register and the R Register.

(b) If Overflow indicates OFF, insert the quotient in the A Register, and insert the undivided remainder (if any) in the R Register.

In division, the divisor must be greater than the portion of the dividend in the A Register. If the dividend is greater than or equal to the divisor, the quotient will exceed the capacity of the A and R Register and an overflow will occur. If the dividend is contained in the A Register, then the R Register must be cleared before dividing.

Assume that:
1. Storage cell 1000 contains the number 0 2222 22 2222.
2. Storage cell 1001 contains the number 1 3333 33 3333.
3. The A Register contains the number 0 9999 99 9999.
4. The R Register contains the number 9999 999999.

<table>
<thead>
<tr>
<th>Program</th>
<th>A Register</th>
<th>R Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAD</td>
<td>1000 0 2222 22 2222</td>
<td>9999 999999</td>
</tr>
<tr>
<td>M</td>
<td>1001 1 0740 74 0740</td>
<td>5925 925926</td>
</tr>
<tr>
<td>CAD</td>
<td>1000 0 2222 22 2222</td>
<td>5925 925926</td>
</tr>
<tr>
<td>MRO</td>
<td>1001 1 0740 74 0741</td>
<td>0000 000000</td>
</tr>
<tr>
<td>CAD</td>
<td>1000 0 2222 22 2222</td>
<td>0000 000000</td>
</tr>
<tr>
<td>DIV</td>
<td>1001 1 6666 66 6666</td>
<td>2222 222222</td>
</tr>
</tbody>
</table>

Overflow is impossible in multiplication.

Example of overflow in division:

<table>
<thead>
<tr>
<th>Program</th>
<th>A Register</th>
<th>R Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAD</td>
<td>1001 1 3333 33 3333</td>
<td>0000 000000</td>
</tr>
<tr>
<td>*DIV</td>
<td>1000 0 0000 00 0000</td>
<td>0000 000000</td>
</tr>
</tbody>
</table>

*Overflow indicates ON.
Multiplication and Division commands can be used in: 
Determining rates, Payroll extension, Billing, Tax computation, and general engineering computations.

PROBLEM: A store owner wants to take advantage of a close-out sale to purchase 2,250 items at $10.00 each. There will be a shipping cost of $380.00. There is a $909.00 discount if the purchase is made on an 18 month contract at 6.5% interest. The store owner wants to know what his monthly interest payments will be.

_SOLUTION:_

<table>
<thead>
<tr>
<th>Location</th>
<th>Main</th>
<th>Loop</th>
<th>Control Digits</th>
<th>Operation</th>
<th>Operand Address</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>CD</td>
<td>1000</td>
<td>$A = 0.0002 25 0000.</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
<td>1</td>
<td></td>
<td>M</td>
<td>1001</td>
<td>$A = 0.0000 22 5000.</td>
</tr>
<tr>
<td>0002</td>
<td>2</td>
<td>2</td>
<td></td>
<td>AD</td>
<td>1002</td>
<td>$A = 0.0000 22 8800.</td>
</tr>
<tr>
<td>0003</td>
<td>3</td>
<td>3</td>
<td></td>
<td>SU</td>
<td>1003</td>
<td>$A = 0.0000 21 9710.</td>
</tr>
<tr>
<td>0004</td>
<td>4</td>
<td>4</td>
<td></td>
<td>MR</td>
<td>1004</td>
<td>$A = 0.0000 14 2812.</td>
</tr>
<tr>
<td>0005</td>
<td>5</td>
<td>5</td>
<td></td>
<td>DIV</td>
<td>1005</td>
<td>$A = 0.7934 00 0000.</td>
</tr>
</tbody>
</table>

ANSWER: Located in the A Register — $79.34
MANIPULATION AND TRANSFER
OF INFORMATION

SL
000p 14 00nn
Shift the contents of the A Register and the R Register nn places left. The nn digits shifted out of the left end of the A Register re-enter the right end of the R Register in the same order. The sign does not move.

SR
000p 13 00nn
Shift the contents of the A Register and the R Register nn places right. The nn digits shifted out of the right end of the R Register are lost, and nn zeros enter the left end of the A Register. The sign does not move. The maximum value for nn is 19.

Overflow can not occur on shifting commands.

CR
000p 33 0000
Clear the R Register.

RO
000p 23 0000
Round the twenty digit contents of the A Register and the R Register to ten digits. Clear the R Register.

STOP
000p 08 0000
Stop machine operation.

The operation of the DATATRON stops, but no information is lost. Operation is resumed at the next program step when the START button is pressed.
Manipulation commands are provided in the DATA-TRON to facilitate the effective use of arithmetic commands during operation.

**PROBLEM:** A store wants to take advantage of a closeout sale of purchase 2,250 items at $10.00 each. There will be a shipping cost of $380.00. There is a $909.00 discount if the purchase is made on an 18-month contract at 6.5% interest. The store owner wants to know what his monthly principal and interest payments will be.

**SOLUTION:**

<table>
<thead>
<tr>
<th>Location</th>
<th>Loop</th>
<th>Control</th>
<th>Operation</th>
<th>Operand Address</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0</td>
<td>C</td>
<td>R</td>
<td>0000 0000 000000</td>
<td></td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
<td>A</td>
<td>A=0 0000 00 2250</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0002</td>
<td>2</td>
<td>M</td>
<td>1001 A=ZERO, R=0002 250000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0003</td>
<td>3</td>
<td>S</td>
<td>0010 A=0 0002 25 0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0004</td>
<td>4</td>
<td>A</td>
<td>1002 A=0 0002 28 8000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0005</td>
<td>5</td>
<td>S</td>
<td>1003 A=0 0002 19 7100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0006</td>
<td>6</td>
<td>S</td>
<td>1006 A=0 0002 19 7100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0007</td>
<td>7</td>
<td>M</td>
<td>1004 A=ZERO R=0142 811500</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0008</td>
<td>8</td>
<td>S</td>
<td>0007 A=0 0000 14 2811 R=5000 000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0009</td>
<td>9</td>
<td>R</td>
<td>0000 A=0 0000 14 2812, R=ZERO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0010</td>
<td>0</td>
<td>S</td>
<td>0010 A=ZERO, R=0000 142812</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0011</td>
<td>1</td>
<td>D</td>
<td>1005 A=0 0000 00 7934, R=ZERO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0012</td>
<td>2</td>
<td>S</td>
<td>1007 A=ZERO STORING MONTHLY INTEREST</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0013</td>
<td>3</td>
<td>A</td>
<td>1006 A=0 0002 19 7100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0014</td>
<td>4</td>
<td>S</td>
<td>0010 A=ZERO, R=0002 197100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0015</td>
<td>5</td>
<td>D</td>
<td>1005 A=0 0000 12 2066, R=0000 000002</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0016</td>
<td>6</td>
<td>A</td>
<td>1007 A=0 0000 12 9995</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0017</td>
<td>7</td>
<td>S</td>
<td>1008 A=ZERO, 17 MONTHLY PAYMENTS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0018</td>
<td>8</td>
<td>S</td>
<td>0010 A=0 0000 00 0002</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0019</td>
<td>9</td>
<td>A</td>
<td>1008 A=0 0000 12 9997</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0020</td>
<td></td>
<td>S</td>
<td>1009 A=ZERO STORING LAST PAYMENT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TO FIND:** Monthly payments—principal and interest.

**ASSUME:** Information for purchase located in memory cells:

- 1000 0000 00 2250 (Quantity)
- 1001 0000 00 1000 (Unit Price)
- 1002 0000 03 8000 (Shipping Cost)
- 1003 0000 09 0900 (Discount)
- 1004 0000 00 0065 (Interest Rate)
- 1005 0000 00 0018 (Number of Payments)

**ANSWER:** Located in memory cells 1008 (17 monthly payments), and 1009 (last payment).
**BT4 BLOCK TO LOOP 4**

<table>
<thead>
<tr>
<th>Program</th>
<th>A Register</th>
<th>R Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>BT₄</td>
<td>1 6214 91 2721</td>
<td>2179 430198</td>
</tr>
<tr>
<td>BT₅</td>
<td>1 6214 91 2721</td>
<td>2179 430198</td>
</tr>
<tr>
<td>BF₄</td>
<td>1 6214 91 2721</td>
<td>2179 430198</td>
</tr>
<tr>
<td>BF₅</td>
<td>1 6214 91 2721</td>
<td>2179 430198</td>
</tr>
</tbody>
</table>

Block transfer the contents of twenty consecutive main storage cells, beginning with xxxx, to the 4000 quick access loop. Use BT₅ (35) for the 5000 loop, BT₆ (36) for the 6000 loop, and BT₇ (37) for the 7000 loop.

**BF4 BLOCK FROM LOOP 4**

<table>
<thead>
<tr>
<th>Program</th>
<th>A Register</th>
<th>R Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>BF₄</td>
<td>1 6214 91 2721</td>
<td>2179 430198</td>
</tr>
<tr>
<td>BF₅</td>
<td>1 6214 91 2721</td>
<td>2179 430198</td>
</tr>
</tbody>
</table>

Block transfer the contents of the 4000 quick access loop to twenty consecutive main storage cells, beginning with xxxx. Use BF₅ (25) for the 5000 loop, BF₆ (26) for the 6000 loop, and BF₇ (27) for the 7000 loop.

In blocking to a quick access loop, main storage is unchanged and the previous contents of that loop are completely erased.

In blocking from a quick access loop to main storage, 20 words in main storage are erased and 20 new words are written. The quick access loop remains unchanged.

**EX EXTRACT**

<table>
<thead>
<tr>
<th>Program</th>
<th>A Register</th>
<th>R Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>EX</td>
<td>1 6014 01 2701</td>
<td>2179 430198</td>
</tr>
</tbody>
</table>

Extract from the contents of the A Register by changing each digit in the A Register (including sign) to zero if the digit in the corresponding position in xxxx is zero. The digit in the A Register remains unchanged if the digit in the corresponding position in xxxx is one.

**CIRA CIRCULATE A**

<table>
<thead>
<tr>
<th>Program</th>
<th>A Register</th>
<th>R Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIRA</td>
<td>2 7211 62 1491</td>
<td>2179 430198</td>
</tr>
</tbody>
</table>

Shift the contents (including sign) of the A Register nn + 1 places left. The digits shifted out of the left end of the A Register re-enter the right end of the A Register in the same order.

**UA UNIT ADJUST**

<table>
<thead>
<tr>
<th>Program</th>
<th>A Register</th>
<th>R Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>UA</td>
<td>1 7214 91 2721</td>
<td>2179 430198</td>
</tr>
</tbody>
</table>

Increase by one the most significant position of the A Register if the digit in this position is even. When the digit is odd, there is no change. The sign of the A Register is immaterial.
The Block transfer commands enable the programmer to place data in the quick access loops and thereby shorten operation time. The manipulation commands presented on the opposite page are excellent for editing and separating parts of a word.

PROBLEM: A warehouse maintains a file of supplies. The data is filed in the following code:

```
O uuvw wx xyyy
```

where: uu is the Supplier code
v is the Color code
ww is the Warehouse Classification code
xx is the Assembly code
yyy is the Detail code

TO FIND: Revised file. (Only one file will be revised in the example program below.)

ASSUME: Example file for revision is located in cell 1000. The program is located in loop 7.

```
It has been determined that the Warehouse Classification code should be revised to contain three digits and that the color code is not necessary. The current files shall be assigned the Warehouse Classification code of lw (100 plus the current code number).
```

SOLUTION:

<table>
<thead>
<tr>
<th>Location</th>
<th>Loop</th>
<th>Control Digits</th>
<th>Operation</th>
<th>Operand Address</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>3000</td>
<td>7000</td>
<td></td>
<td>BT4</td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td>3001</td>
<td>7001</td>
<td></td>
<td>CAD</td>
<td>4000</td>
<td>A = 0, uuvw wx xyyy.</td>
</tr>
<tr>
<td>3002</td>
<td>7002</td>
<td></td>
<td>EX</td>
<td>7009</td>
<td>A = 0, uuvw wx xyyy.</td>
</tr>
<tr>
<td>3003</td>
<td>7003</td>
<td></td>
<td>GIRA</td>
<td>0001</td>
<td>A = 0, uuvw wx xyy.</td>
</tr>
<tr>
<td>3004</td>
<td>7004</td>
<td></td>
<td>UA</td>
<td>0000</td>
<td>A = 0, uuvw wx xyy.</td>
</tr>
<tr>
<td>3005</td>
<td>7005</td>
<td></td>
<td>GIRA</td>
<td>0002</td>
<td>A = 0, uuvw wx xyy.</td>
</tr>
<tr>
<td>3006</td>
<td>7006</td>
<td></td>
<td>STC</td>
<td>4000</td>
<td></td>
</tr>
<tr>
<td>3007</td>
<td>7007</td>
<td></td>
<td>BF4</td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td>3008</td>
<td>7008</td>
<td></td>
<td>STOP</td>
<td>0000</td>
<td></td>
</tr>
<tr>
<td>3009</td>
<td>7009</td>
<td>L1 D1</td>
<td></td>
<td>1111</td>
<td>EXTRACT CONSTANT.</td>
</tr>
</tbody>
</table>
DECISION MAKING AND BRANCHING

CC  CHANGE CONDITIONALLY
000p 28 xxxx
Overflow indicates ON: Change control to xxxx. Reset
Overflow.
Overflow indicates OFF: Control continues in sequence.
Overflow indicates OFF: command is ignored and con-
rol continues in sequence.

CCB  CHANGE CONDITIONALLY, BLOCK
000p 28 xxxx
Overflow indicates ON: Block transfer the contents of
twenty consecutive main storage cells, beginning with
xxxx, to the 7000 loop. Change control to 70xx. Reset
Overflow.
Overflow indicates OFF: Control continues in sequence.
Overflow indicates OFF: command is ignored and con-
rol continues in sequence.

CCR  CHANGE CONDITIONALLY, RECORD
000p 29 xxxx
Overflow indicates ON: Clear the R Register. Store in the
four most significant positions of the R Register the ad-
ress (as contained in the Control Counter) of the com-
mand next in sequence. Change control to xxxx. Reset
Overflow.
Overflow indicates OFF: Control continues in sequence.
Overflow indicates OFF: command is ignored and con-
rol continues in sequence.

CCBR  CHANGE CONDITIONALLY, BLOCK, RECORD
000p 39 xxxx
Overflow indicates ON: Block transfer the contents of
twenty consecutive main storage cells, beginning with
xxxx, to the 7000 loop. Clear the R Register. Store in the
four most significant positions of the R Register the ad-
dress (as contained in the Control Counter) of the com-
mand next in sequence. Change control to 70xx. Reset
Overflow.
Overflow indicates OFF: Control continues in sequence.
Overflow indicates OFF: command is ignored and con-
rol continues in sequence.

CU  CHANGE UNCONDITIONALLY
000p 20 xxxx
Change control to xxxx.

CUB  CHANGE UNCONDITIONALLY, BLOCK
000p 30 xxxx
Block transfer the contents of twenty consecutive main
storage cells, beginning with xxxx, to the 7000 loop.
Change control to 70xx.

CUR  CHANGE UNCONDITIONALLY, RECORD
000p 21 xxxx
Clear the R Register. Store in the four most significant
positions of the R Register the address (as contained in
the Control Counter) of the command next in sequence.
Change control to xxxx.

CUBR  CHANGE UNCONDITIONALLY, BLOCK, RECORD
000p 31 xxxx
Block transfer the contents of twenty consecutive main
storage cells, beginning with xxxx, to the 7000 loop.
Clear the R Register. Store in the four most significant
positions of the R Register the address (as contained in
the Control Counter) of the command next in sequence.
Change control to 70xx.
Change of control commands enable the programmer to allow for possible exceptions during the execution of the program.

**PROBLEM:** A new car agency requests each salesman to report sales, in dollars, twice a month, and the number of demonstrations for customers made once a month. The sales for the first half of the month are located in memory cells 1000–1999. The sales for the last half of the month are located in memory cells 2000–2999. The number of demonstrations are located in memory cells 3000–3999. (All of the above are arranged according to salesman number; i.e., salesman no. 9, reported the information in 1009, 2009, 3009.)

**TO FIND:** Total dollar sales per salesman during the month, per customer demonstration.

**ASSUME:** For example purposes, use only salesman no. 10. (1010, 2010, 3010 cell nos.)

**SOLUTION:**

<table>
<thead>
<tr>
<th>Location</th>
<th>Main</th>
<th>Loop</th>
<th>S</th>
<th>Control Digits</th>
<th>Operation No</th>
<th>Alpha</th>
<th>Operand Address</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>0800</td>
<td>0800</td>
<td></td>
<td>o80</td>
<td>CUB</td>
<td>0801</td>
<td></td>
<td></td>
<td>PLACE PROGRAM IN LOOP 7.</td>
</tr>
<tr>
<td>0801</td>
<td>7001</td>
<td></td>
<td></td>
<td>BT4</td>
<td>1000</td>
<td></td>
<td></td>
<td>FIRST HALF MONTH SALES.</td>
</tr>
<tr>
<td>0802</td>
<td>7002</td>
<td></td>
<td></td>
<td>BT5</td>
<td>2000</td>
<td></td>
<td></td>
<td>SECOND HALF MONTH SALES.</td>
</tr>
<tr>
<td>0803</td>
<td>7003</td>
<td></td>
<td></td>
<td>BT6</td>
<td>3000</td>
<td></td>
<td></td>
<td>DEMONSTRATIONS.</td>
</tr>
<tr>
<td>0804</td>
<td>7004</td>
<td></td>
<td></td>
<td>CR</td>
<td>0000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0805</td>
<td>7005</td>
<td></td>
<td></td>
<td>CAD</td>
<td>4010</td>
<td></td>
<td>TOTAL SALES</td>
<td></td>
</tr>
<tr>
<td>0806</td>
<td>7006</td>
<td></td>
<td></td>
<td>AD</td>
<td>5010</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0807</td>
<td>7007</td>
<td></td>
<td></td>
<td>CC</td>
<td>7015</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0808</td>
<td>7008</td>
<td></td>
<td></td>
<td>ST</td>
<td>4010</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0809</td>
<td>7009</td>
<td></td>
<td></td>
<td>DIV</td>
<td>6010</td>
<td></td>
<td>SALES ÷ DEMONSTRATIONS</td>
<td></td>
</tr>
<tr>
<td>0810</td>
<td>7010</td>
<td></td>
<td></td>
<td>CCR</td>
<td>*</td>
<td></td>
<td>DIVIDEND ADJUSTMENT</td>
<td></td>
</tr>
<tr>
<td>0811</td>
<td>7011</td>
<td></td>
<td></td>
<td>STC</td>
<td>6010</td>
<td>*</td>
<td>SALESMAN/DEMONSTRATOR</td>
<td></td>
</tr>
<tr>
<td>0812</td>
<td>7012</td>
<td></td>
<td></td>
<td>BF4</td>
<td>1000</td>
<td></td>
<td>ANSWERS TO MAIN STORAGE</td>
<td></td>
</tr>
<tr>
<td>0813</td>
<td>7013</td>
<td></td>
<td></td>
<td>BF6</td>
<td>3000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0814</td>
<td>7014</td>
<td></td>
<td></td>
<td>STOP</td>
<td>0000</td>
<td>OR CONTINUE TO NEXT SALESMAN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0815</td>
<td>7015</td>
<td></td>
<td></td>
<td>SR</td>
<td>00 01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0816</td>
<td>7016</td>
<td></td>
<td></td>
<td>UA</td>
<td>0000</td>
<td>ADDITION ADJUSTMENT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0817</td>
<td>7017</td>
<td></td>
<td></td>
<td>CU</td>
<td>7008</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Additional Remarks:**

* AT THE END OF THE DIVIDEND ADJUSTMENT PROGRAM, THERE WILL BE A CUB 0811 COMMAND TO RE-ENTER ABOVE PROGRAM. **
NOR  NORMALIZE (CHANGE ON ZERO)
000p 15 xxxx
(a) If the content of the A Register is not zero, shift the
twenty digits in the A Register and the R Register left until
the most significant position in the A Register is not zero.
The sign does not move. Record the number of shifts in the
Special Counter.
(b) If the content of the A Register is zero, shift the con-
tents of the R Register left into the A Register, clear the
R Register, and change control to xxxx. The sign does not move.

CNZ  CHANGE ON NON-ZERO  000p 04 xxxx
Test the contents of the A Register (not the sign) for zero.
(a) If the A Register setting is zero, set the sign of the A
Register to zero and continue control in sequence.
(b) If the A Register setting is not zero, change control
to xxxx.

A non-zero number in the A Register is unaltered.

ADSC  ADD SPECIAL COUNTER
000p 16 0000
Add the contents of the Special Counter to the least
significant position of the A Register.

The condition of Overflow is possible with this command
if the A Register is at least 99999 99 9991. If the Special
Counter is 10, the machine will stop on “forbidden com-
bination” without performing the addition.

SUSC  SUBTRACT SPECIAL COUNTER
000p 17 0000
Subtract the contents of the Special Counter from the
least significant position of the A Register.

(See Add Special Counter.)

OSGD  OVERFLOW ON SIGN DIFFERENCE
000p 73 xxxx
If the sign of the A Register differs from the sign of xxxx,
Overflow indicates ON.

The contents of the A Register is not affected. This command
must always be followed by a Conditional Change command.

Program  A Register  R Register
NOR  3172  1 0006 21 4912  2179 430198
NOR  3172  1 6214 91 2217  9430 198000
[Special Counter is 3. Control continues sequentially.]
Program  A Register  R Register
NOR  3172  1 0000 00 0000  2179 430198
NOR  3172  1 2179 43 0198  0000 000000
[Special Counter is 10 (forbidden combination). Control transfers to cell 3172.]
Program  A Register  R Register
CNZ  3172  1 6214 91 2721  2179 430198
CNZ  3172  1 6214 91 2721  2179 430198
[Control transfers to cell 3172.]
Program  A Register  R Register
ADSC  0000  0 0000 00 0005  2179 430198
SUSC  0000  0 0000 00 0052  2179 430198
[SUSC  0000  0 0000 00 0050  2179 430198]
[Assume Special Counter is 2.
Assume cell 3710 is positive. Assume cell 3720 is negative.

Cell  Program  A Register  R Register
0000  OSGD3710  0 6214 91 2721  2179 430198
0001  CC  0004  0 6214 91 2721  2179 430198
0002  OSGD3720  0 6214 91 2721  2179 430198
0003  CC  0005  0 6214 91 2721  2179 430198
0004  STOP 0000  0 6214 91 2721  2179 430198
0005  If +, control transfers to cell 0005.
If −, control continues to cell 0004.
Decision making commands are used to determine the nature of a number. As a result of this determination the next steps are performed, either in normal sequence or by branching to another part of the program.

**PROBLEM:** An inventory count of various parts is located in memory cells 1000-1999. If a count is 100 or less, that part should be re-ordered.

**SOLUTION 1:**

<table>
<thead>
<tr>
<th>Location</th>
<th>Control</th>
<th>Operation</th>
<th>Operand</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>0640</td>
<td>7000</td>
<td>$\text{BT}4$</td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td>0641</td>
<td>7001</td>
<td>$\text{BT}4$</td>
<td>---</td>
<td>RE-ORDER PROGRAM</td>
</tr>
<tr>
<td>0642</td>
<td>7002</td>
<td>$\text{CA}D$</td>
<td>4000</td>
<td>FIRST INVENTORY COUNT</td>
</tr>
<tr>
<td>0643</td>
<td>7003</td>
<td>$\text{SU}$</td>
<td>7009</td>
<td>BALANCE TO MAINTAIN</td>
</tr>
<tr>
<td>0644</td>
<td>7004</td>
<td>$\text{CNZ}$</td>
<td>7006</td>
<td></td>
</tr>
<tr>
<td>0645</td>
<td>7005</td>
<td>$\text{SU}$</td>
<td>6000</td>
<td>RE-ORDER PROGRAM</td>
</tr>
<tr>
<td>0646</td>
<td>7006</td>
<td>$\text{DS}D$</td>
<td>7004</td>
<td></td>
</tr>
<tr>
<td>0647</td>
<td>7007</td>
<td>$\text{CC}$</td>
<td>6000</td>
<td>RE-ORDER PROGRAM, COUNT 0</td>
</tr>
<tr>
<td>0648</td>
<td>7008</td>
<td>$\text{CA}D$</td>
<td>4000</td>
<td>FIRST INVENTORY COUNT</td>
</tr>
<tr>
<td>0649</td>
<td>7009</td>
<td>$\text{SU}$</td>
<td>7013</td>
<td>BALANCE TO MAINTAIN</td>
</tr>
<tr>
<td>0650</td>
<td>7010</td>
<td>$\text{NOR}$</td>
<td>6000</td>
<td>RE-ORDER PROGRAM, COUNT 0</td>
</tr>
<tr>
<td>0651</td>
<td>7011</td>
<td>$\text{STOP}$</td>
<td>6000</td>
<td>OR CONTINUE TO NEXT COUNT</td>
</tr>
<tr>
<td>0652</td>
<td>7012</td>
<td>0</td>
<td>0000</td>
<td>0008 CONSTANT</td>
</tr>
<tr>
<td>0653</td>
<td>7013</td>
<td>0</td>
<td>0000</td>
<td>0100 CONSTANT</td>
</tr>
</tbody>
</table>

**SOLUTION 2:**

<table>
<thead>
<tr>
<th>Location</th>
<th>Control</th>
<th>Operation</th>
<th>Operand</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>0640</td>
<td>7000</td>
<td>$\text{BT}4$</td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td>0641</td>
<td>7001</td>
<td>$\text{BT}4$</td>
<td>---</td>
<td>RE-ORDER PROGRAM</td>
</tr>
<tr>
<td>0642</td>
<td>7002</td>
<td>$\text{CA}D$</td>
<td>4000</td>
<td>FIRST INVENTORY COUNT</td>
</tr>
<tr>
<td>0643</td>
<td>7003</td>
<td>$\text{NOR}$</td>
<td>6000</td>
<td>RE-ORDER PROGRAM, COUNT 0</td>
</tr>
<tr>
<td>0644</td>
<td>7004</td>
<td>$\text{CA}D$</td>
<td>7012</td>
<td></td>
</tr>
<tr>
<td>0645</td>
<td>7005</td>
<td>$\text{SU}$</td>
<td>6000</td>
<td>RE-ORDER PROGRAM, COUNT 0</td>
</tr>
<tr>
<td>0646</td>
<td>7006</td>
<td>$\text{O}S\text{D}$</td>
<td>7006</td>
<td>CHECK IF COUNT UNDER 100</td>
</tr>
<tr>
<td>0647</td>
<td>7007</td>
<td>$\text{CC}$</td>
<td>6000</td>
<td>RE-ORDER PROGRAM, COUNT UNDER 100</td>
</tr>
<tr>
<td>0648</td>
<td>7008</td>
<td>$\text{CA}D$</td>
<td>4000</td>
<td>FIRST INVENTORY COUNT</td>
</tr>
<tr>
<td>0649</td>
<td>7009</td>
<td>$\text{SU}$</td>
<td>7013</td>
<td>BALANCE TO MAINTAIN</td>
</tr>
<tr>
<td>0650</td>
<td>7010</td>
<td>$\text{NOR}$</td>
<td>6000</td>
<td>RE-ORDER PROGRAM, COUNT 0</td>
</tr>
<tr>
<td>0651</td>
<td>7011</td>
<td>$\text{STOP}$</td>
<td>6000</td>
<td>OR CONTINUE TO NEXT COUNT</td>
</tr>
<tr>
<td>0652</td>
<td>7012</td>
<td>0</td>
<td>0000</td>
<td>00008 CONSTANT</td>
</tr>
<tr>
<td>0653</td>
<td>7013</td>
<td>0</td>
<td>0000</td>
<td>0100 CONSTANT</td>
</tr>
</tbody>
</table>

**TO FIND:** The nature of the inventory count for each part, and determine whether or not it is necessary to re-order.

**NOTE:** Two methods of solution are shown.

**ASSUME:** The inventory count located in cell 1000 shall be used as an example.
USING THE B REGISTER

Assume cell 3170 contains 0 1279 42 0019.

<table>
<thead>
<tr>
<th>Program</th>
<th>A Register</th>
<th>B Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>1 6214 91 2721</td>
<td>2174</td>
</tr>
<tr>
<td>0000</td>
<td>1 6214 91 2721</td>
<td>0019</td>
</tr>
<tr>
<td>0000</td>
<td>1 6214 91 2721</td>
<td>0020</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Program</th>
<th>A Register</th>
<th>B Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>1 6214 91 2721</td>
<td>0019</td>
</tr>
<tr>
<td>0000</td>
<td>1 6214 91 2721</td>
<td>0020</td>
</tr>
<tr>
<td>0000</td>
<td>0 0000 00 0020</td>
<td>0020</td>
</tr>
</tbody>
</table>

**SET B**

SB
000p 72 xxxx
Set the B Register to the value of the four least significant positions of xxxx.

**INCREASE B**

IB
000p 32 0000
Add one to the contents of the B Register.

**B TO A**

BA
000p 11 0000
Clear the A Register. Add the contents of the B Register.

**DECREASE B**

DB
000p 22 xxxx
Subtract one from the contents of the B Register.
(a) If the new B Register setting is 9999 (0000 - 1), control continues in sequence.
(b) If the new B Register setting is not 9999, change control to xxxx.

Control will transfer to cell 1004.

Control continues in sequence to cell 1001.
The B Register is one of the most valuable tools available to the programmer. Use of the commands for the B Register is as varied as the problems in the business world. The use of the commands is shown below. For a more refined use of the B Register, refer to the discussions of various techniques on the following pages.

**PROBLEM:** A manufacturing concern has an hourly payroll of 1000 employees. A daily computation of hours and earnings is required by employee number. Daily computations are accumulated for the weekly payroll.

**TO FIND:** Accumulate hours and amount by employee number. Current daily totals are readily available if required. Totals are accumulated for weekly payroll processing.

**ASSUME:** Employee numbers range from 0000 through 0999. In the cell corresponding to the employee number is the current day's record for that employee. That is, in cell 0000 will be found the following information for employee 0000: his daily hours worked, and his wage rate. This information is stored in the following manner: xxx (hours), yyy (rate), 0000.

The accumulated hours and pay for each employee will be stored in cells 1000 through 1999, with the total for employee 0000 located in cell 1000, that of employee 0001 in cell 1001, etc. This information is stored in the following manner: xxxx (hours), yyyyyy(amount of pay).

<table>
<thead>
<tr>
<th>Location</th>
<th>Control Digits</th>
<th>Operation</th>
<th>Operand Address</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 600</td>
<td></td>
<td>SB</td>
<td>6017</td>
<td></td>
</tr>
<tr>
<td>1 601</td>
<td></td>
<td>CAD</td>
<td>(0000)</td>
<td>CURRENT HOURS AND RATE</td>
</tr>
<tr>
<td>2 602</td>
<td></td>
<td>EX</td>
<td>6018</td>
<td>EXTRACT AND STORE</td>
</tr>
<tr>
<td>3 603</td>
<td></td>
<td>STC</td>
<td>2000</td>
<td>RATE</td>
</tr>
<tr>
<td>4 604</td>
<td></td>
<td>CAD</td>
<td>(0000)</td>
<td>CURRENT HOURS AND RATE</td>
</tr>
<tr>
<td>5 605</td>
<td></td>
<td>EX</td>
<td>6019</td>
<td>EXTRACT AND STORE</td>
</tr>
<tr>
<td>6 606</td>
<td></td>
<td>ST</td>
<td>2001</td>
<td>HOURS</td>
</tr>
<tr>
<td>7 607</td>
<td></td>
<td>SR</td>
<td>0002</td>
<td>SCALE</td>
</tr>
<tr>
<td>8 608</td>
<td></td>
<td>MRO</td>
<td>2000</td>
<td>CURRENT AMOUNT</td>
</tr>
<tr>
<td>9 609</td>
<td></td>
<td>STC</td>
<td>2000</td>
<td>→ STORE</td>
</tr>
<tr>
<td>0 601</td>
<td></td>
<td>CAD</td>
<td>2001</td>
<td>CURRENT HOURS</td>
</tr>
<tr>
<td>1 601</td>
<td></td>
<td>SR</td>
<td>0001</td>
<td>Scale to Add Hours</td>
</tr>
<tr>
<td>2 601</td>
<td></td>
<td>AD</td>
<td>2000</td>
<td>Add Current Amount</td>
</tr>
<tr>
<td>3 601</td>
<td></td>
<td>ST</td>
<td>(0000)</td>
<td>Store Current Hours and Amount</td>
</tr>
<tr>
<td>4 601</td>
<td></td>
<td>AD</td>
<td>(1000)</td>
<td>Accumulated Hours and Amount</td>
</tr>
<tr>
<td>5 601</td>
<td></td>
<td>STC</td>
<td>(1000)</td>
<td>→ STORE</td>
</tr>
<tr>
<td>6 601</td>
<td></td>
<td>DB</td>
<td>6001</td>
<td>TALLY</td>
</tr>
<tr>
<td>7 601</td>
<td></td>
<td>STOP</td>
<td>0999</td>
<td>Modification Constant</td>
</tr>
<tr>
<td>8 601</td>
<td>0001</td>
<td></td>
<td></td>
<td>Constant For Extracting Rate</td>
</tr>
<tr>
<td>9 601</td>
<td>1100</td>
<td></td>
<td></td>
<td>Constant For Extracting HOURS</td>
</tr>
</tbody>
</table>
SCALING

Scaling is an important part of a program. It is a problem for the programmer only, not the computer. The computer will always handle the data as ten digit numbers. The programmer, then, has the problem of keeping track of the movement of the decimal point.

In addition and subtraction, scaling must locate the digits in a number so that the decimal points are in line. For example, 52.9 plus 5.32 is written

\[
\begin{array}{c}
52.9 \\
+ 5.32 \\
\hline
58.22
\end{array}
\]

and not

\[
\begin{array}{c}
52.9 \\
+ 53.2 \\
\hline
58.22
\end{array}
\]

No further scaling is necessary for addition or subtraction.

However, multiplication and division change the position of the decimal point. In multiplication, when two 10 digit numbers are multiplied, the product is a 20 digit number in the A Register and the R Register. In division, the dividend is a 20 digit number, the divisor is a 10 digit number, and the quotient is a 10 digit number appearing in the A Register.

The following rule will be helpful to the programmer:

1. In multiplication, the number of places to the right of the decimal point in the product is the sum of the number of decimal places to the right of the decimal point in the multiplicand and the multiplier.

2. In division, the number of places to the right of the decimal point in the quotient is the number of places to the right of the decimal point in the dividend (A and R) minus the number of decimal places to the right of the decimal point in the divisor.

Examples:

1. 0 0002 25 0000 multiplied by 0 1000 00 0000 is 0 0000 22 5000 0000 000000.

   The number represented by 0 0002 25 0000 is actually 250.

   The number represented by 0 1000 00 0000 is actually 10.

   Therefore, the number of decimal places to the right of the decimal point is:

   \[
   \text{Multiplcand } 3 \\
   \text{Multiplier } 8 \\
   \text{Product (A and R) } 11
   \]

   The product, therefore, represented by
   \[
   0 0000 22 5000 0000 000000 
   \]

   is actually 22,500.

2. 0 0000 14 2812 0000 000000 divided by 0 0000 18 0000 is 0 7934 00 0000.

   The number represented by 0 0000 14 2812 0000 000000 is actually 1428.12.

   The number represented by 0 0000 18 0000 is actually 18.

   Therefore, the number of decimal places to the right of the decimal point is:

   \[
   \begin{array}{c}
   \text{Dividend } 12 \\
   \text{Divisor } 4 \\
   \text{Quotient (A) } 8
   \end{array}
   \]

   The quotient, therefore, represented by
   \[
   0 7934 00 0000 
   \]

   is actually 79.34.

COMMAND MODIFICATION
AND CYCLING

In using a stored program machine such as DATATRON, it is often necessary to provide a means of modifying commands in order that a small program may be used repeatedly, but with different data each time the calculation is performed. To accomplish this effect, it is required that the instruction addresses which refer to the data be changed on each cycle.

The DATATRON provides two means for the programmer to accomplish command modification; i.e., B modification and programmed modification.

B modification of commands:

Any command with a 1 in the first control digit (numerical minus sign) will have the contents of the B Register added to the instruction address of the command before it is executed. Example: Sum 500 sales stored in cells 0000-0499.

<table>
<thead>
<tr>
<th>Location</th>
<th>Control</th>
<th>Operation</th>
<th>Operand</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main Loop</td>
<td>S</td>
<td>Alpha</td>
<td>Address</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>700 0</td>
<td>0</td>
<td>0000</td>
<td>SET BREGISTER TO 0498</td>
</tr>
<tr>
<td>1</td>
<td>700 1</td>
<td>0</td>
<td>0000</td>
<td>CLEAR AREGISTER AND ADD LAST SALE</td>
</tr>
<tr>
<td>2</td>
<td>700 2</td>
<td>1</td>
<td>0000</td>
<td>SUM SALES</td>
</tr>
<tr>
<td>3</td>
<td>700 3</td>
<td>0</td>
<td>0000</td>
<td>TALLY</td>
</tr>
<tr>
<td>4</td>
<td>700 4</td>
<td>0</td>
<td>0000</td>
<td>STOP BREGISTER HAS COUNTED DOWN 499 TIMES</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In this case, the B Register serves two functions; i.e., automatic address modifier and tallying device, for which DB performs both functions.

**Programmed modification:**

Any command may be inserted into the A Register. It appears to the DATATRON just as a number, and any number may be added or subtracted, thereby modifying the command. Usually, only the instruction address is modified. Example: 500 sales are stored in 0000-0499 which are to be summed.

The command in cell 7001 was modified on each cycle to serve as a tally device and refer to the next data item.

<table>
<thead>
<tr>
<th>Location</th>
<th>Loop</th>
<th>Control Digits</th>
<th>Operation</th>
<th>Operand Address</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>700</td>
<td>0</td>
<td>CAD</td>
<td>0000</td>
<td>FIRST SALE</td>
</tr>
<tr>
<td>1</td>
<td>700</td>
<td>0</td>
<td>AD</td>
<td>[0001]</td>
<td>ADD NEXT SALE</td>
</tr>
<tr>
<td>2</td>
<td>700</td>
<td>0</td>
<td>STC</td>
<td>7010</td>
<td>STORE PARTIAL SUM TEMPORARILY</td>
</tr>
<tr>
<td>3</td>
<td>700</td>
<td>0</td>
<td>CAD</td>
<td>7001</td>
<td>MODIFY ADD COMMAND</td>
</tr>
<tr>
<td>4</td>
<td>700</td>
<td>0</td>
<td>AD</td>
<td>7009</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>700</td>
<td>0</td>
<td>CC</td>
<td>7011</td>
<td>TEST FOR COMPLETION OF PROBLEM</td>
</tr>
<tr>
<td>6</td>
<td>700</td>
<td>0</td>
<td>STC</td>
<td>7001</td>
<td>MODIFIED ADD COMMAND</td>
</tr>
<tr>
<td>7</td>
<td>700</td>
<td>0</td>
<td>CAD</td>
<td>7010</td>
<td>PARTIAL SUM</td>
</tr>
<tr>
<td>8</td>
<td>700</td>
<td>0</td>
<td>CU</td>
<td>7001</td>
<td>REPEAT CYCLE</td>
</tr>
<tr>
<td>9</td>
<td>700</td>
<td>0</td>
<td></td>
<td>OT01</td>
<td>TALLY AND MODIFYING CONSTANT</td>
</tr>
<tr>
<td>0</td>
<td>701</td>
<td>0</td>
<td></td>
<td>OT00</td>
<td>TEMPORARY-REFERENCE CELL</td>
</tr>
<tr>
<td>1</td>
<td>701</td>
<td>0</td>
<td>STOP</td>
<td>0000</td>
<td>PROBLEM COMPLETED</td>
</tr>
</tbody>
</table>

**USE OF THE QUICK ACCESS LOOPS**

**General:**

The sole purpose of the quick access loops is to increase the speed of computer operation. Normally, all commands are executed from one of these loops, and, if possible, the data groups used in a section of the program are stored in any of the loops. Only under difficult situations should the program sequence be shifted to main storage where the speed of operation is ten times as lengthy. The blocking commands are provided to facilitate the use of the quick access loops.

**GENERAL RULES FOR USE OF THE QUICK ACCESS LOOPS**

**PROGRAMMING:** Experience shows that the 7000 loop receives greatest use in programming because of the CUB command's dual function of blocking from main storage to the 7000 loop and then transferring control to the first command of the twenty words blocked. Of course, the program will operate at the same speed when using any of the three remaining loops. Normally programs are broken into segments of twenty commands or less such that discrete small groups of program steps are used in the loops as a unit. This does not mean that every twentieth step must be a CUB command but that a CUB or CU or any other transfer command must appear at least once in every twenty commands as used in one loop.

**DATA STORAGE:** For considerations of speed, it is desirable to store data and constants in the quick access loops along with the program. Generally, the program which has the least number of main memory instruction addresses will be the most rapid.

**ADDRESSING:** The accepted practice for designating quick access loop instruction addresses is to keep them within the ranges 4000-4019, 5000-5019, 6000-6019, 7000-7019.

**DATA EDITING**

Usually in business problems, arithmetic computation is not complex. The general case is the making of logical decisions and the preparation of data in proper format for these decisions. The DATATRON provides five very convenient commands for this purpose, viz., CIRA, EX, SL, SR, UA.

CIRA. This command provides a means of "opening" a word any number of spaces and inserting additional digits. CIRA is really a shift left command that applies to the A Register only.

Example: Expand a six digit account number to eight digits, inserting 00 between the second and third digits.
EXTRACT. This command is of great value in breaking down coded numbers (part numbers, account numbers, policy numbers, etc.) such that certain logical decisions requiring a particular type of processing may be made easily.

Example: Check a nine digit part number to determine whether the sixth digit (warehouse class code) is a 7. If so, increase unit cost by 1%.

<table>
<thead>
<tr>
<th>Location</th>
<th>Control Digits</th>
<th>Operation</th>
<th>Operand Address</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 700 0 0 0000 CAD 7014 A = 0.0XXX XX XXXY. PART.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 700 1 0 0000 EX 7011 A = 0.000X 00 0000. CLS.CPE.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 702 0 0000 SU 7010</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 703 0 0000 CNZ 7009 CHECK FOR ‘7’</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 704 0 0000 CAD 7012 A = 0.0000 00 XXXX. UNIT COST</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 705 0 0000 SR 0002 A = 0.0000 00 00XX. 1%.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 706 0 0000 DO 0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 707 0 0000 AD 7012 A = 0.0000 00 XXXX. UNIT COST</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8 708 0 0000 STC 7013</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9 709 0 0000 STOP 0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 7010 0 0007 00 0000 CONSTANT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 7011 0 0010 00 0000 CONSTANT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 7012 0 0000 00 XXXX OLD UNIT COST</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 7013 0 0000 00 XXXX NEW UNIT COST</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 7014 0 0XXX X XXXX PART NO.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
TABLE LOOK-UP

In general, table look-up with the DATATRON is a simple and rapid machine process. Basically, the problem is this: a file exists in memory, and a particular item from the file is called for by each input item. If the file is lengthy, the search for the desired item may be time-consuming unless it is efficiently programmed.

There are several methods of performing the search:

1. Comparing item by item, starting at the head of the table. This is the slowest but simplest method.
2. Comparing by starting at the middle of the table and continuously eliminating half the remaining possibilities. This is Binary Search and is usually fairly rapid. This technique should be used when the third method is not applicable.
3. The most used method, a particularly efficient one with the DATATRON, is this: construct the file in such a way that the location of each item is correlated to the input number itself. If this can be done, the DATATRON can use the input number (part number, employee number, etc.) to specify the address of the item in the table. Then the table entry can be called up directly, and no search for it need be performed.

In the simplest case, the input number can directly identify the memory location in the table. An example of this is the following.

PROBLEM: A bank has 3000 checking accounts. It keeps the current balance for each account in a file in the DATATRON memory. Transactions are to be entered in groups as they are processed. Each account is assigned a number from 0000 to 2999, and its current balance is kept in the corresponding memory location. Thus account number 1504 has its current balance recorded in location 1504.

100 transactions have been read into memory locations 3500-3599. Each is recorded in the following form:

\[ \pm \text{xxxxx} yyyy \]

where \( \pm \text{xxxx} \) = transaction

\( yyyy \) = account number

The table look-up in this example was done without any searching, because the account number could be used to directly indicate the proper location of the proper current balance.

In most practical cases, the identifying number (account, employee, or part number) does not directly correspond to a memory address. The memory location corresponding to an input item must be computed or looked up as a separate operation before the table look-up can be carried out. In some cases, an arithmetic computation can convert the item into the appropriate address. In others, it is necessary to use a "dictionary" to find the memory location corresponding to an input item. In either case, the program operating time is somewhat increased. However, this method will still be much faster than an item-by-item search of the main file or a binary search.

<table>
<thead>
<tr>
<th>Location</th>
<th>Control</th>
<th>Operation</th>
<th>Operand</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main</td>
<td>Loop</td>
<td>S Digits</td>
<td>No Alpha</td>
<td>Address</td>
</tr>
<tr>
<td>0 7000</td>
<td>9000</td>
<td>CAD 3500</td>
<td>A = xxxxx0 yyyy</td>
<td></td>
</tr>
<tr>
<td>1 7001</td>
<td>ST 7013</td>
<td>B = yyyy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 7002</td>
<td>SB 7013</td>
<td>A = 00000 xxxxx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 7003</td>
<td>SR 0005</td>
<td>Add balance from file</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 7004</td>
<td>AD 0000</td>
<td>Store new balance in file</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 7005</td>
<td>ST 0000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 7006</td>
<td>CAD 7000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 7007</td>
<td>AD 7012</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8 7008</td>
<td>CC '7011</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9 7009</td>
<td>STC 7000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 7010</td>
<td>CU 7000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 7011</td>
<td>STOP 0000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 7012</td>
<td>00010</td>
<td>00001 Constant To modify 7000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PROBLEM: A bank keeps current balances for checking accounts in a file in DATATRON memory. The account numbers vary from 00000 to 99999, but only 1500 accounts exist. As before, the problem is to post transactions in the current balance file.

In this case, there is no correspondence between account number and memory location. For example, the current balance for account number 14708 may be in memory location 1695, while 1696 contains the current balance for account number 35614. The file is maintained in account number sequence however. A dictionary must be provided which will relate the account number to the memory location where the current balance is stored.

The solution to this problem, as given below, is fairly complex. However, it shows the DATATRON performing a practical operation at high efficiency, and demonstrates the power of this table look-up technique.

SOLUTION: Table #2 is composed of 1500 current balances stored in memory locations 1600-3099, and Table #1 is composed of 1500 account numbers stored in locations 0100-1599. With each account number is stored the address of its balance (Table #2) so that the entry has the following form.

\[
\begin{align*}
\text{Table} & \quad 0 \text{dddd} \ \text{xxxx} \\
\text{Entry} & \quad \text{dddd} = \text{address of current balance in Table #2.} \\
& \quad \text{xxxx} = \text{account number}
\end{align*}
\]

The table of account numbers, Table #1, is arranged in ascending sequence. A third table, Table #3, is constructed, which consists of 100 BT4 commands referring to Table #1. The first two digits of the account number will correspond to a memory cell in Table #3. The instruction address of the BT4 command in that cell will refer to Table #1. This table is stored in 0000-0099, and makes possible a very rapid look-up of the account number, in the following way.

1. An input account number is in 5000, and the amount to be posted in 5001.
2. The first two digits of the account number call for the appropriate BT4 order from the Table #3 in 0000-0099.
3. A BT4 command is executed, bringing a block of 20 account numbers from Table #1 into Loop 4.
4. The 20 account numbers are searched by sequential comparison, and the desired account number is found. This is the only actual search necessary.
5. The address accompanying the matched account number is used to refer to the proper cell in the Table #2.
6. The posting is performed, and the operation is complete.

<table>
<thead>
<tr>
<th>Table #3</th>
<th>Table #1</th>
<th>Table #2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cells</td>
<td>Cells</td>
<td>Cells</td>
</tr>
<tr>
<td>0000-0099</td>
<td>0100-1599</td>
<td>1600-3099</td>
</tr>
</tbody>
</table>

Table 1 BT4 addresses. Account numbers and Table 2 addresses. Current balances.
<table>
<thead>
<tr>
<th>Location</th>
<th>Control Digits</th>
<th>Operation</th>
<th>Operand</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main 0</td>
<td>698</td>
<td>CAD</td>
<td>5000</td>
<td>GIVEN ACCOUNT NUMBER</td>
</tr>
<tr>
<td>Loop 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>698</td>
<td>SR</td>
<td>0003</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>698</td>
<td>AD</td>
<td>7001</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>698</td>
<td>STC</td>
<td>6984</td>
<td>COMMAND MADE UP BY</td>
</tr>
<tr>
<td>4</td>
<td>698</td>
<td>[CAD</td>
<td>00XX]</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>698</td>
<td>STC</td>
<td>6986</td>
<td>(6986)=0.0000BT4 (0100-1599)</td>
</tr>
<tr>
<td>6</td>
<td>698</td>
<td>[BT4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>698</td>
<td>SB</td>
<td>7000</td>
<td>SET (Q) to 19</td>
</tr>
<tr>
<td>8</td>
<td>698</td>
<td>CAD</td>
<td>4000</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>698</td>
<td>CR</td>
<td>0000</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>699</td>
<td>SL</td>
<td>0005</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>699</td>
<td>cira</td>
<td>0005</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>699</td>
<td>su</td>
<td>5000</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>699</td>
<td>CnZ</td>
<td>6999</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>699</td>
<td>SL</td>
<td>0010</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>699</td>
<td>AD</td>
<td>7001</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>699</td>
<td>st</td>
<td>7004</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>699</td>
<td>cu</td>
<td>7002</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>699</td>
<td>0 0000</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>699</td>
<td>db</td>
<td>6988</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>700</td>
<td>Stop</td>
<td>0019</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>700</td>
<td>0 0000</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>700</td>
<td>su</td>
<td>6998</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>700</td>
<td>STC</td>
<td>7006</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>700</td>
<td>[CAD</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>700</td>
<td>AD</td>
<td>5001</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>700</td>
<td>[STC</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>700</td>
<td>Stop</td>
<td>0000</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Figure 15 Power Control Panel

Figure 16 Supervisory Control Panel
OPERATION AND CONTROLS

DESCRIPTION OF THE OPERATING CONTROLS ON THE DATATRON

1. CLEAR. When the CLEAR button is depressed, the A, R, B, C Registers, and all alarm indicators are set to zero and the DATATRON is placed in the execute state.

2. EXECUTE. When the DATATRON is in the execute phase, the EXECUTE light is on. When the START button is depressed, the command in the C Register will be executed. The DATATRON may be placed in the execute phase without depressing the CLEAR button by pressing the button directly beneath the EXECUTE light.

3. FETCH. When the DATATRON is in the fetch phase, the FETCH light is on. When the START button is depressed, the command whose storage address is specified by the contents of the Control Counter is inserted into the Order and Address Registers. The DATATRON may be placed in the fetch phase by pressing the button beneath the FETCH light.

4. CONTINUOUS-STEP.
   a. CONTINUOUS. The DATATRON will automatically progress at high speed in sequence according to the program.
   b. STEP. The DATATRON will proceed by single program steps in sequence according to the program. The START button must be pressed for each execute or fetch operation.

5. START. When the START button is depressed, the DATATRON will change from execute to fetch or vice versa. If the CONTINUOUS-STEP switch is set to STEP, program by program operation is performed; if set to CONTINUOUS, the DATATRON will proceed at high speed.

6. LOCK-NORMAL. During normal operations, this switch is in the NORMAL position. When it is in the LOCK position, the DATATRON is prevented from changing the phase, fetch, or execute of the cycle.

7. OVERFLOW. During the operation of the program in STEP or CONTINUOUS, if an overflow condition occurs and the command causing the overflow is not immediately followed by a Conditional Change command, the DATATRON will stop and the OVERFLOW indicator will be set. The indicator may be reset by pressing the RESET button beneath it.

8. SECTOR. When an internal timing error occurs, the DATATRON will stop, setting the SECTOR alarm. This indicator may be reset by pressing the RESET button beneath it.

9. F.C. When a forbidden combination (a binary-coded decimal digit greater than 9) is sensed, the DATATRON will stop. The sensing occurs on both fetch and execute cycles, and a forbidden combination may be detected in the A, R, B, C or D Registers.

10. CONTROL ALARM. When the command STOP (08) or an overflow not followed by a Conditional Change occurs, the CONTROL alarm is set. The indicator may be reset by pressing the Reset button.

11. IDLE LIGHT. Whenever the DATATRON ceases high speed operation, the IDLE indicator is set.

12. ZERO. By pressing the ZERO button and simultaneously pressing any of the red buttons under the registers, a particular flip-flop of that register is reset.

13. All other buttons or switches on the Supervisory Control Panel are of no particular use to the operator; these are used primarily by the Customer Service Engineer.

MANIPULATION OF THE CONTENTS OF THE REGISTERS

For each digit of any register, there are four flip-flops. Directly beneath each digit there are four red buttons. When any red button is pressed, the corresponding flip-flop is ON. This provides a direct means of changing quantities in any register. When the DATATRON is being manually controlled, this is the only way of inserting desired quantities into any of the registers.

OPERATING INSTRUCTIONS

All controls for operating the DATATRON are located on the Power Control Unit and the Supervisory Control Panel on the center of the DATATRON.

1. To turn the DATATRON on:

   a. Turn on the MASTER CONTROL 115 V switch on the panel inside the left-hand door of the Power Control Cabinet.
   b. Press the BLOWER ON button.
   c. Press the FILAMENT ON button.
   d. Press DRUM ON button.
   e. Wait five minutes for tube temperatures to stabilize.
   f. Press the MOTOR GEN(ERATOR) ON button.
   g. Press the DC FAIL RESET button.
   h. Press the DC ON button.
   i. This sequence must be followed.

2. To insert information (manual input): Storage of data, commands, or constants in the DATATRON is accomplished as follows:
   a. Set CONTINUOUS-STEP switch to STEP and press the CLEAR button.
   b. Insert the data item or command in the A Register by depressing the corresponding red buttons to form the proper binary-coded decimal representation of the digits desired.
c. Insert into the Order Register the STC (02) command.
d. Insert into the Address Register the address of the storage location into which the number in the A Register is to be stored.
e. Press the START button. The STC command will be executed and the DATATRON will immediately stop.

3. To transfer control to a stored command:
   a. Press the CLEAR button.
   b. Set the CONTINUOUS-STEP switch to whichever phase is desired.
   c. Insert the CU (20) command into the Order Register. Insert into the Address Register the location of the command to which control is to be transferred.
   d. Press START button.

4. To operate the DATATRON continuously: Assuming that a command is in the C Register:
   a. Set the CONTINUOUS-STEP switch to CONTINUOUS.
   b. Press START button.

5. To operate the DATATRON step-by-step: Assuming that a command is in the C Register:
   a. Set the CONTINUOUS-STEP switch to STEP.
   b. Press START button. The command in the C Register will be executed.
   c. Press START button. The command in the location specified by the Control Counter will be fetched into the Order and Instruction Address Registers.
   d. Press the START button. The command now in the Order and Instruction Address Registers will be executed. The START button must be depressed once for each fetch phase and once for each execute phase. As each execute phase is completed, the progress of the problem may be observed in the A, R, B, C and D Registers.

6. To resume DATATRON operation at any selected point in the program: Assuming that the DATATRON has stopped (Overflow, the command STOP, forbidden combination, sector alarm, or a program imperfection):
   a. Following the command STOP:
      Press START button.
   b. Following any other type of stop:
      Execute Number 3.

7. To inspect a stored program in sequence: Occasionally it is necessary to inspect a series of data, commands, or constants stored on the drum.
   a. Press the CLEAR button.
   b. Insert into the Order Register CAD (64).
   c. Place LOCK-NORMAL switch at LOCK.
   d. Insert in the Address Register (C Register) the address of the location whose contents are to be displayed.
   e. Press START button.
   f. Return to Step d. until all necessary items have been examined.

8. To turn the DATATRON off: All controls to perform shut-down are located on the Power Control Unit.
   a. Press DC OFF button.
   b. Press MOTOR GEN(ERATOR) OFF button.
   c. Press DRUM OFF button.
   d. Press FILAMENT OFF button.
   e. Press BLOWERS OFF button.
   f. Turn off the MASTER CONTROL 115 V switch on the inside panel of the Power Control Unit.
   g. This sequence must be followed.
STANDARD SYMBOLS

xxxx— four digit address of storage cell on magnetic drum, or four digit address of block of information on magnetic tape.

p — control digit inserted in command word to act as breakpoint instruction.

f — control digit inserted in command word to act as format instruction to output device.

n — control digit inserted in command word to designate quantity.

0 — digit normally used to complete word.

u — control digit inserted in command word to designate unit number of input or output component.

h — control digit inserted in command word to designate head number for magnetic tape search operation.

t — control digit inserted in command word to set punch (or printer) relays.

ARITHMETIC

\[ 000p \ 66 \ xxxx \]
Clear the A Register. Add the contents of xxxx.

\[ 000p \ 76 \ xxxx \]
Add the absolute value of the contents of xxxx to the contents of the A Register.

\[ 000p \ 80 \ xxxx \]
Add the floating point number in xxxx to the floating point number in the A Register.

\[ 0000 \ 10 \ 0000 \]
Stop machine operation. Add the next digit read (from manual keyboard or paper tape reader) to the least significant position of the A Register.

\[ 000p \ 65 \ xxxx \]
Clear the A Register. Subtract the contents of xxxx.

\[ 000p \ 67 \ xxxx \]
Clear the A Register. Subtract the absolute value of the contents of xxxx.

\[ 000p \ 75 \ xxxx \]
Subtract the contents of xxxx from the contents of the A Register.

\[ 000p \ 77 \ xxxx \]
Subtract the absolute value of the contents of xxxx from the contents of the A Register.

\[ 000p \ 81 \ xxxx \]
Subtract the floating point number in xxxx from the floating point number in the A Register.
**MULTIPLICATION**

**M**  
000p 60 xxxx  
Multiply the contents of xxxx by the contents of the A Register. Insert the twenty digit product in the A Register and the R Register. The most significant digits are in the A Register.

**MRO**  
000p 70 xxxx  
Multiply the contents of xxxx by the contents of the A Register. Round the product to ten digits. Clear the R Register.

**FM**  
000p 82 xxxx  
Multiply the floating point number in xxxx by the floating point number in the A Register. Insert the eighteen digit floating point product in the A Register and the R Register. The most significant digits are in the A Register.

**DIVISION**

**DIV**  
000p 61 xxxx  
Divide the twenty digit contents of the A Register and the R Register by the contents of xxxx.  
(a) If Overflow indicates ON, clear the A Register and the R Register.  
(b) If Overflow indicates OFF, insert the quotient in the A Register, and insert the undivided remainder (if any) in the R Register.

**FDIV**  
000p 83 xxxx  
Divide the eighteen digit floating point number in the A Register and the R Register by the floating point number in xxxx. Insert the ten digit floating point quotient in the A Register. Insert the undivided remainder (if any) in the least significant positions of the R Register.

**USING THE B REGISTER**

**SB**  
000p 72 xxxx  
Set the B Register to the value of the four least significant positions of xxxx.

**BA**  
000p 11 0000  
Clear the A Register. Add the contents of the B Register.

**IB**  
000p 32 0000  
Add one to the contents of the B Register.

**DB**  
000p 22 xxxx  
Subtract one from the contents of the B Register.  
(a) If the new B Register setting is 9999 (0000 – 1), control continues in sequence  
(b) If the new B Register setting is not 9999, change control to xxxx.

**MANIPULATION AND TRANSFER OF INFORMATION**

**ST**  
000p 12 xxxx  
Store the contents of the A Register in xxxx.

**STC**  
000p 02 xxxx  
Store the contents of the A Register in xxxx. Clear the A Register.

**SL**  
000p 14 00nn  
Shift the contents of the A Register and the R Register nn places left. The nn digits shifted out of the left end of the A Register re-enter the right end of the R Register in the same order. The sign does not move.

**SR**  
000p 13 00nn  
Shift the contents of the A Register and the R Register nn places right. The nn digits shifted out of the right end of the R Register are lost, and nn zeros enter the left end of the A Register. The sign does not move. The maximum value for nn is 19.

**NOR**  
000p 15 xxxx  
Normalizing (Change on Zero)  
See definition under "Decision Making and Branching" commands.

**CIRA**  
000p 01 00nn  
Shift the contents (including sign) of the A Register nn + 1 places left. The digits shifted out of the left end of the A Register re-enter the right end of the A Register in the same order.

**EX**  
000p 63 xxxx  
Extract from the contents of the A Register by changing each digit in the A Register (including sign) to zero if the digit in the corresponding position in xxxx is zero. The digit in the A Register remains unchanged if the digit in the corresponding position in xxxx is one.

**CR**  
000p 33 0000  
Clear the R Register.

**RO**  
000p 23 0000  
Round the twenty digit contents of the A Register and the R Register to ten digits. Clear the R Register.

**BT4**  
000p 34 xxxx  
Block transfer the contents of twenty consecutive main storage cells, beginning with xxxx, to the 4000 quick access loop. Use BT5 (35) for the 5000 loop, BT6 (36) for the 6000 loop, and BT7 (37) for the 7000 loop.
BF4
000p 24 xxxx
Block transfer the contents of the 4000 quick access loop to twenty consecutive main storage cells, beginning with xxxx. Use BF5 (25) for the 5000 loop, BF6 (26) for the 6000 loop, and BF7 (27) for the 7000 loop.

UA
000p 06 0000
Increase by one the most significant position of the A Register if the digit in this position is even.

ADSC
000p 16 0000
Add the contents of the Special Counter to the least significant position of the A Register.

SUSC
000p 17 0000
Subtract the contents of the Special Counter from the least significant position of the A Register.

DECISION MAKING AND BRANCHING

STOP
000p 08 0000
Stop machine operation.

OSGD
000p 73 xxxx
If the sign of the A Register differs from the sign of xxxx, Overflow indicates ON.

CNZ
000p 04 xxxx
Test the contents of the A Register (not the sign) for zero. (a) If the A Register setting is zero, set the sign of the A Register to zero and continue control in sequence. (b) If the A Register setting is not zero, change control to xxxx.

NOR
000p 15 xxxx
(a) If the content of the A Register is not zero, shift the twenty digits in the A Register and the R Register left until the most significant position in the A Register is not zero. The sign does not move. Record the number of shifts in the Special Counter. (b) If the content of the A Register is zero, shift the contents of the R Register left into the A Register, clear the R Register, and change control to xxxx. The sign does not move.

CC
000p 28 xxxx
Overflow indicates ON: Change control to xxxx. Reset Overflow. Overflow indicates OFF: Control continues in sequence.

CCR
000p 29 xxxx
Overflow indicates ON: Clear the R Register. Store in the four most significant positions of the R Register the address (as contained in the Control Counter) of the command next in sequence. Change control to xxxx. Reset Overflow. Overflow indicates OFF: Control continues in sequence.

CCBR
000p 39 xxxx
Overflow indicates ON: Block transfer the contents of twenty consecutive main storage cells, beginning with xxxx, to the 7000 loop. Clear the R Register. Store in the four most significant positions of the R Register the address (as contained in the Control Counter) of the command next in sequence. Change control to 70xx. Reset Overflow. Overflow indicates OFF: Control continues in sequence.

CU
000p 20 xxxx
Change control to xxxx.

CUB
000p 30 xxxx
Block transfer the contents of twenty consecutive main storage cells, beginning with xxxx, to the 7000 loop. Change control to 70xx.

CUR
000p 21 xxxx
Clear the R Register. Store in the four most significant positions of the R Register the address (as contained in the Control Counter) of the command next in sequence. Change control to xxxx.

CUBR
000p 31 xxxx
Block transfer the contents of twenty consecutive main storage cells, beginning with xxxx, to the 7000 loop. Clear the R Register. Store in the four most significant positions of the R Register the address (as contained in the Control Counter) of the command next in sequence. Change control to 70xx.

INPUT-OUTPUT

▼ TYPEWRITER

PTW
000p 03 fffn
Write on typewriter, transferring the sign and nn digits from the A Register. Digits ff act as an instruction to the typewriter. Shift the contents (including sign) of the A Register nn + 1 places left. The digits shifted out of the left end of the A Register re-enter the right end of the A Register in the same order.

PTWF
000p 07 0000
Actuate typewriter as directed by digit f.
**DATATRON COMMAND LIST**

**PAPER TAPE**

**PTR**

**READ**

0000 00 xxxx

Read from paper tape, transferring words to consecutive storage cells on the drum starting with xxxx. Stop input and start computation after reading a CU, CUB, CUR, or CUBR command (with a 6 or 7 in the sign position).

**PTW**

**WRITE**

000p 03 ffnn

Punch on paper tape, transferring the sign and nn digits from the A Register. Punch digits ff on tape to act as an instruction to a typewriter. Shift the contents (including sign) of the A Register nn + 1 places left. The digits shifted out of the left end of the A Register re-enter the right end of the A Register in the same order.

**PTWF**

**WRITE FORMAT**

000p 07 0f00

Punch the digit f on paper tape to act as an instruction to a typewriter.

---

**MAGNETIC TAPE**

**MTR**

**REWIND**

00up 52 0000

Rewind DataReader u.

**MTS**

**SEARCH**

00up 42 xxxx

Search for block xxxx under head h on DataReader u. Overflow indicates ON if a previous MTS command has not been completed.

**MTR**

**READ**

nnup 40 xxxx

Read nn consecutive blocks of twenty words each from DataReader u, transferring words to consecutive storage cells on the drum starting with xxxx. Overflow indicates ON if a previous MTS command has not been completed.

**MTW**

**WRITE**

nnup 50 xxxx

Write nn consecutive blocks of twenty words each on DataReader u, transferring words from consecutive storage cells on the drum starting with xxxx. Overflow indicates ON if a previous MTS command has not been completed.

---

**CARD FEED, CARD PUNCH AND TABULATOR WITH CARDATRON**

**CDR**

**READ**

nnnp 44 xxxx

Read 1000 - nn cards continuously, transferring words to consecutive storage cells on the drum starting with xxxx.

**CDW**

**WRITE**

nnnp 54 xxxx

Punch 1000 - nn cards (or print 1000 - nn lines) continuously, transferring words from consecutive storage cells on the drum starting with xxxx.

**EXC**

**EXTERNAL CONTROL**

000p 71 xxxx

Insert the contents of xxxx in the D Register. For each of the eight most significant digits in the D Register there is an electronic switch. A “3” changes the state of the corresponding switch, a “2” closes the corresponding switch, a “1” opens the corresponding switch, and a “0” does not alter the state of the corresponding switch.

---

**CARD FEED, CARD PUNCH AND TABULATOR WITH CARDATRON**

**CDRF**

**READ FORMAT**

0fup 48 xxxx

Load format band f on input u, transferring words from consecutive storage cells on the drum starting with xxxx.

**CDWF**

**WRITE FORMAT**

0fup 58 xxxx

Load format band f on output u, transferring words from consecutive storage cells on the drum starting with xxxx.

**CDR**

**READ**

00up 44 xxxx

Read the contents of one card from input u, transferring words to consecutive storage cells on the drum starting with xxxx. Reload input u with the contents of the next card.

**CDW**

**WRITE**

tfup 54 xxxx

Punch one card (or print one line) at output u, transferring words from consecutive storage cells on the drum starting with xxxx. Edit the information as directed by format band f. Control the punch (or printer) as directed by digit t.

**CDRI**

**READ INTERROGATE**

00up 45 xxxx

Interrogate input u. If input u is ready to read, clear the R Register. Store in the four most significant positions of the R Register the address (as contained in the Control Counter) of the command next in sequence. Change control to xxxx. If input u is not ready to read, control continues in sequence.

**CDWI**

**WRITE INTERROGATE**

00up 55 xxxx

Interrogate output u. If output u is ready to write, clear the R Register. Store in the four most significant positions of the R Register the address (as contained in the Control Counter) of the command next in sequence. Change control to xxxx. If output u is not ready to write, control continues in sequence.

---

**ElectroData**

DIVISION OF BURROUGHS

460 SIERRA MADRE VILLA, PASADENA, CALIFORNIA

Copyright 1956

By ElectroData DIVISION OF BURROUGHS

PRINTED IN U.S.A.
ELECTRONIC DATA PROCESSING SYSTEMS

ElectroData
DIVISION OF BURROUGHS
400 SIERRA MADRE VILLA, PASADENA, CALIFORNIA