BOOKS ... V AND VI

Minivac 601

How Computers Work for Man

MINIVAC Games

MINIVAC 601

BOOKS V-VI

SCIENTIFIC DEVELOPMENT CORP.
WATERTOWN, MASS.

The Minivac Manual was prepared and edited by the staff of

Scientific Development Corporation

First Printing—August, 1961

CONTENTS

BOOK V: HOW COMPUTERS WORK FOR MAN

BOOK VI: MINIVAC GAMES

BOOK V

How Computers Work for Man

PREFACE

The previous books in this series have examined various functions performed by modern high-speed digital computers. In this book, MINIVAC 601 is used to demonstrate how these functions are combined in various situations to enable the digital computer to do a specific job. Examples of computer applications are examined in five major sections.

The first section—***Simple Computer-like Devices Encountered Every Day***—presents a number of familiar basic switching circuit devices. These applications involve relatively simple, special-purpose machines which are not "computers" in the general sense in which computers were defined in Book II, although they perform computer-like functions.

The second section—***Computer Applications in Business and Industry***—contains a description of some general applications which computer systems find in business and industry. Included in this section is a discussion of how computers "read" printed material in processing bank checks. Industrial systems used to provide automatic process control and checking of production are also included in this section.

Section three—***Computer Applications in Science and the Military***—contains examples of problem-solving and control applications in which large computers are now being used. In addition to illustration of the kinds of problems which computers are helping scientists to solve, this section contains experiments demonstrating how computers are utilized in radar tracking, automatic coding systems, and missile check-out procedures.

Section four—***Computer Applications in the Social Sciences***—examines the potentials of computers in such areas of election prediction, language translation, and simulation of human learning functions.

Although the first four sections of this book provide a summary of the many areas in which high-speed computers are finding applications, there is one computer application which cannot reasonably be considered under any of the four headings discussed above. Although this fifth area is probably the least important of all computer applications, it may well be the only application with which your friends are really familiar. Section five—***Computer Application in Science Fiction Films***—is included to insure that you will be able to use MINIVAC 601 to demonstrate how computers are used on the movie and TV screen.

Much work has recently been done in the development of "compiler programs" which permit programmers to communicate instructions to computers using a language similar to that used to communicate mathematical problems to men. The compiler program converts the symbols of mathematical and/or logical expressions into binary codes which can be processed by the computer. In the appendix to this book—***Computer Programming Languages***—commonly used programming languages are discussed.

1. Simple Computer-like Devices Encountered Every Day

This section contains examples of several familiar computer-like machines. These simple devices perform specialized functions using basic computer or "switching circuit" functions.

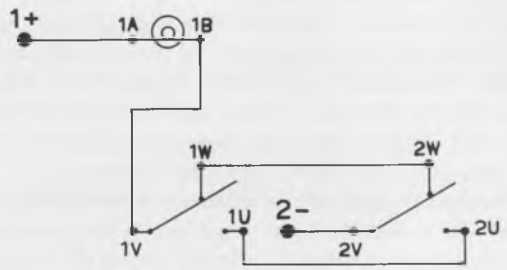
It is important to remember that the machines used to perform these functions do not have

the capacity or flexibility of the more generalized high-speed digital computers discussed in later sections of this book. These simple machines, however, provide excellent examples of specialized computer functions used to perform some basic jobs for man.

Experiment 1: The Lights on the Stairs

It seems appropriate to begin our examples of simple computer-like systems with the ordinary light switch. In many homes a light in a particular location can be controlled from 2 points. This is often true of the light in a stair well which can be turned on or off either at the top or the bottom of the stairs. The problem presented in this case (i.e., how to turn the light on or off from either the top or the bottom of the stairs) is solved using a basic computer function.

The function used in this circuit is a version of the AND circuit. The circuit diagram and program below summarize the situation. With this circuit in use, the light can be turned on or off from either of 2 locations. Assume that slide switch 1 is the switch at the top of the stairs and slide switch 2 is the switch at the bottom of the stairs. Regardless of which position the switch is in, the switch at the other location can always turn the light either on or off.



CIRCUIT DIAGRAM—LIGHT ON THE STAIRS

Program for dual-control light:

1A/1+
1B/1V
1U/2U
1W/2W
2V/2—

To use the program:

Light 1 can be turned either ON or OFF using slide switches 1 or 2.

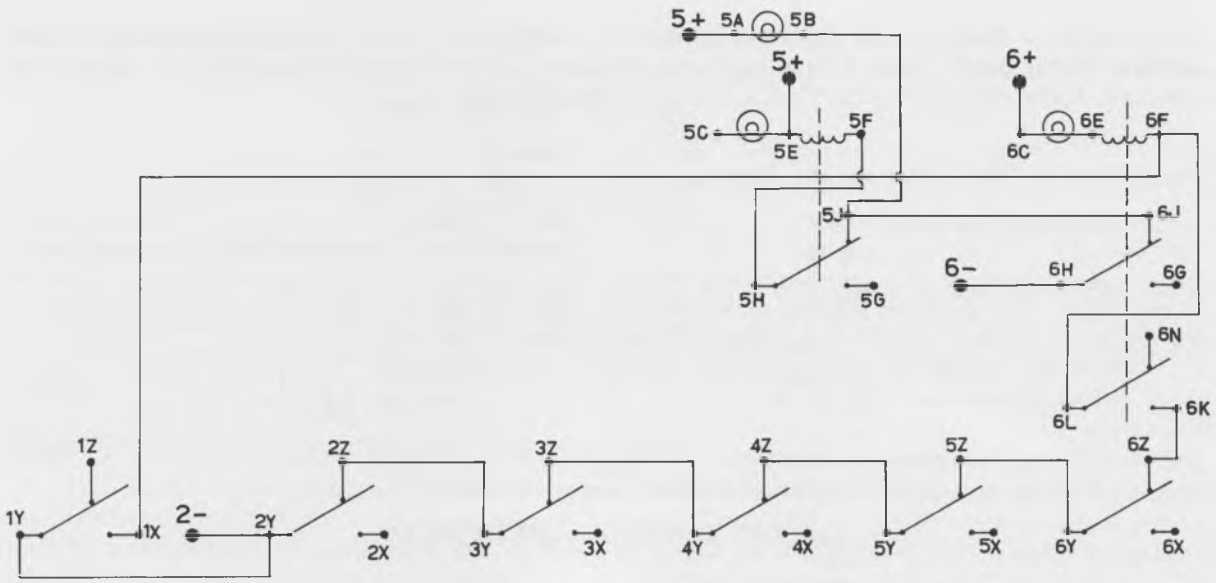
For example, moving slide switch 1 to the LEFT will turn light 1 ON if slide switch 2 is RIGHT. The light will go OFF if slide switch 2 is now moved to the LEFT.

Experiment 2: The Burglar Alarm

This next circuit provides an example of the basic NOT function in a device found in some homes, most office buildings, and all banks: the burglar alarm. The essential element of this application is that ***all*** of a set of contact points must be closed for the alarm ***not*** to sound. The contact points may be on doors, windows or vault openings.

If any set of contacts is opened—by a door or window being opened—the circuit is broken and the alarm sounds. Once the alarm sounds, the device can be re-set by closing the contacts and throwing a re-set switch. In actual practice, the re-set switch is not readily accessible; in some cases, in fact, the re-set switch may be at police headquarters.

The program which follows illustrates a simple alarm circuit. If any contact is broken by pushing any of pushbuttons 2 through 6, the alarm (light 5) comes on. The system is re-set by pushing pushbutton 1.



CIRCUIT DIAGRAM—BURGLAR ALARM

Program for Burglar Alarm:

1X/6F	3Z/4Y	5E/5+	6C/6+
1Y/2Y	4Z/5Y	5F/5H	6F/6L
2Y/2—	5A/5+	5J/6J	6H/6—
2Z/3Y	5B/5J	5Z/6Y	6K/6Z

To use the program:

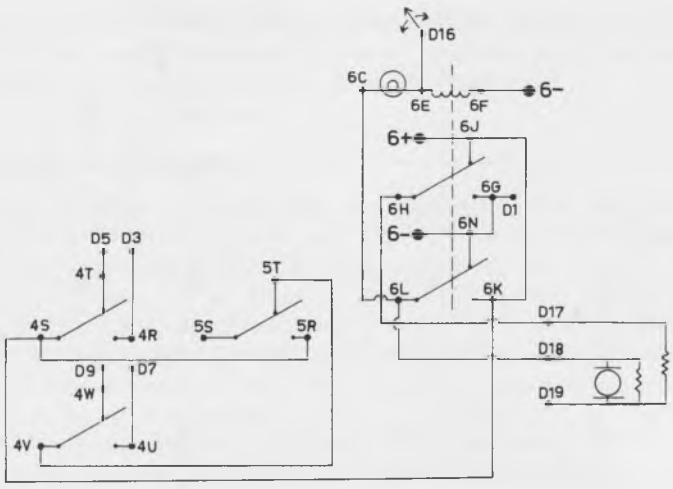
Turn power ON. The alarm (light 5) comes ON. Set the system by pushing the re-set button (pushbutton 1). The burglar alarm is now set. If any contact is broken by pushing any of pushbuttons 2 through 6, the alarm will come on. After an alarm has been given, the system can be re-set by pushing pushbutton 1.

Experiment 3: The Metronome

Computers frequently use timing circuits to perform a series of operations or to repeat an operation periodically. A familiar example of a timing device is the metronome, which is designed to generate a signal at periodic intervals.

The program which follows demonstrates a basic timing circuit using the turning of the rotary switch dial between two points to generate a periodic signal. Commercial metronomes can produce signals with a wide range of periods; with this circuit, however, only four periods can be produced using the positions of slide switches 4 and 5 to determine the period. Through an appropriate circuit, the MINIVAC is capable of generating 15 different periods.

The program and circuit diagram for a metronome are:



CIRCUIT DIAGRAM—METRONOME

Program for metronome:

4R/D3	4W/D9	6F/6—	6J/6K
4S/5R	5S/6K	6G/D1	6J/6+
4T/D5	6C/6L	6G/6N	6L/D18
4U/D7	6E/D16	6H/D17	6N/6—
4V/5T			

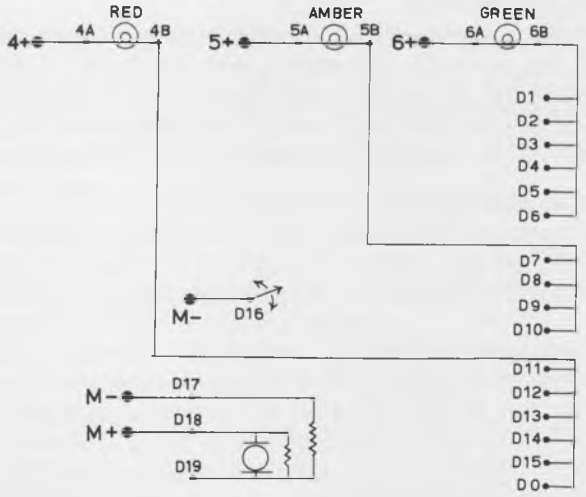
To use the program:

Turn power on. The rotary switch dial will begin to turn back and forth, generating a periodic signal. To change the period of the signal, move slide switches 4 and 5.

Experiment 4: The Traffic Light

Timing circuits are used in many devices. One of the most familiar is the traffic light which follows a pre-programmed cycle from red to green to yellow and back to red. A continuous cycling device is used in this case to produce the familiar sequence of light.

Using the rotary switch mechanism and the lights, MINIVAC can be programmed to simulate the three-light sequence of a traffic light.



CIRCUIT DIAGRAM—TRAFFIC LIGHT

Program for traffic light:

4A/4+	D1/D2	D8/D9	D15/D0
4B/D11	D2/D3	D9/D10	D16/M—
5A/5+	D3/D4	D11/D12	D17/M—
5B/D7	D4/D5	D12/D13	D18/M+
6A/6+	D5/D6	D13/D14	
6B/D1	D7/D8	D14/D15	

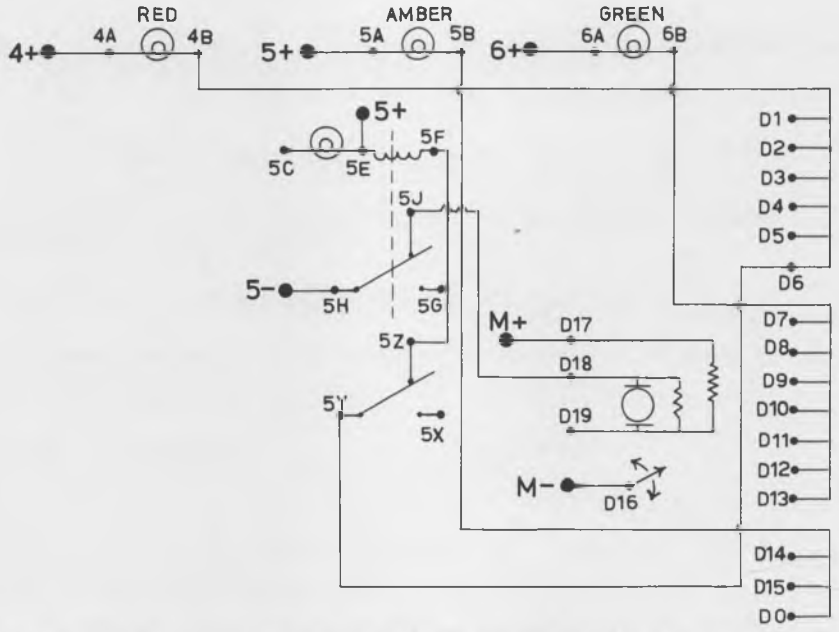
To use the traffic light program:

Turn power on. The rotary switch will turn, turning lights 4, 5 and 6 on and off in succession. Light 4 represents "red", light 5 represents "amber", and light 6 represents "green". The length of time any given light remains on may be varied by changing the connections on the rotary switch.

Experiment 5: The Traffic Light with Pedestrian Control

The traffic light provides a good opportunity to move from a simple timing circuit to a more complicated circuit which involves sequence control. The traffic light in Experiment 4 moves through a pre-determined sequence, and unless the device is re-programmed, it will continue to follow the sequence as long as power is fed to it. It is often desirable, however, to interrupt the sequence or control it in some manner.

Specifically in the case of the traffic light it is desirable to have an "over-ride" control which allows a pedestrian to modify the normal pattern of the light. The program and circuit which follow illustrate an "interrupt circuit" which permits an operator to temporarily change or start a pre-programmed sequence.



CIRCUIT DIAGRAM—TRAFFIC LIGHT WITH PEDESTRIAN CONTROL

Program for traffic light with pedestrian control:

4A/4+	5F/5Z	6B/D7	D5/D6	D12/D13
4B/D1	5H/5—	D1/D2	D7/D8	D14/D15
5A/5+	5J/D18	D2/D3	D8/D9	D15/D0
5B/D14	5Y/D6	D3/D4	D9/D10	D16/M—
5E/5+	6A/6+	D4/D5	D10/D11	D17/M+
			D11/D12	

To use the program:

Turn power on. The "red" light (light 4) comes ON. To cross the street: push pushbutton 5 until the "red" light (light 4) goes OFF and the "green light" (light 6) comes ON. After the light has turned green, it will go to amber (light 5) and return to red again.

In actual practice, this traffic light would be tied to a complementary light facing the flow of traffic. This traffic light, facing the pedestrian, would normally be red, the complementary light would normally be green. Since the two lights are connected, changing the pedestrian's light by pushing a button would automatically change the complementary light.

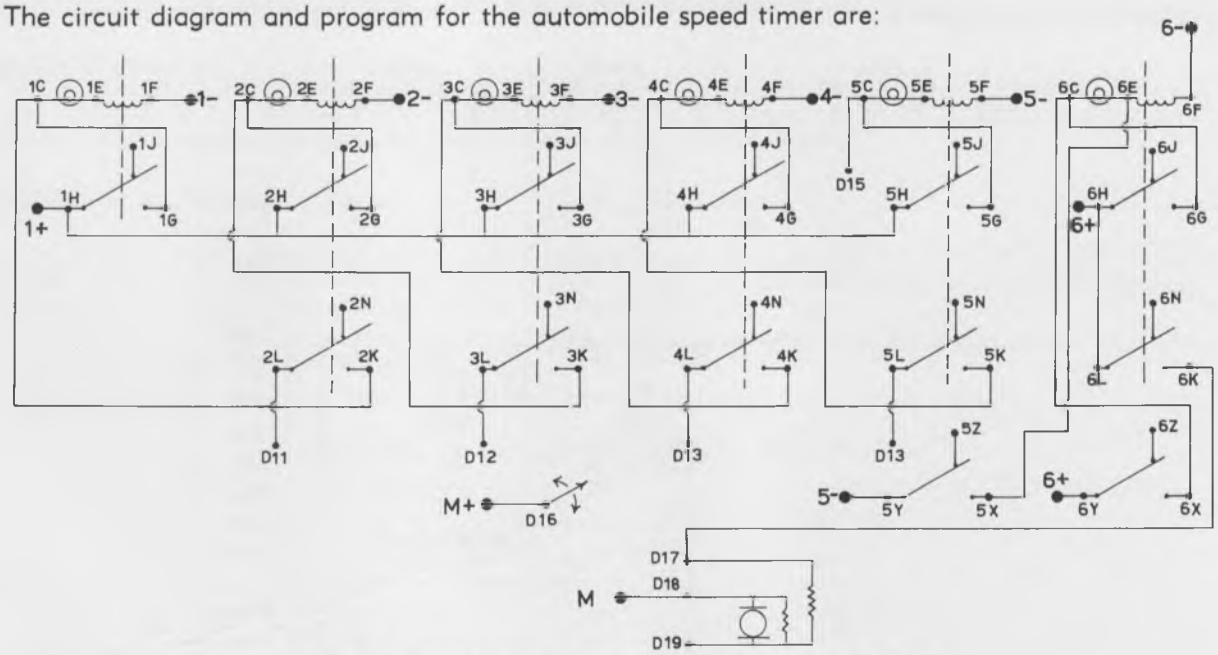
Experiment 6: The Automobile Speed Timer

As an example of a computer-like device encountered, we hope, ***not*** every day, consider the device used by police who post the warning: "Speed Electrically Timed." The mechanism used to determine the speed of passing cars depends essentially on a device which measures the time which lapses as a car passes between two given points.

The actual devices use a variety of mechanisms (radar beam, switches on the road, etc.) to indicate when a car has passed the two selected points. However, we can simulate the action of the timing mechanism by pushing pushbuttons to indicate when a car passes the check points.

The signal that a car is passing the first check point will be indicated by pushing pushbutton 6; the signal that the car is passing the second check point will be indicated by pushing pushbutton 5. Pushing pushbutton 6 starts the timing motor (in this case the rotary switch); pushing pushbutton 5 stops the motor. As the rotary switch turns, MINIVAC will count its revolutions. After the motor has stopped, we will be able to calculate the speed of the passing car.

The circuit diagram and program for the automobile speed timer are:



CIRCUIT DIAGRAM—AUTOMOBILE SPEED TIMER

Program for the automobile speed timer:

1C/1G	2L/D11	4H/5H	6C/6X
1C/2K	3C/3G	4L/D13	6F/6—
1F/1—	3C/4K	5C/5G	6H/6+
1H/1+	3F/3—	5C/D15	6H/6L
1H/2H	3H/4H	5F/5—	6K/D17
2C/2G	3L/D12	5L/D14	6Y/6+
2C/3K	4C/4G	5X/6E	D16/M+
2F/2—	4C/5K	5Y/5—	D18/M—
2H/3H	4F/4—	6C/6G	

The simplest way to calculate automobile speeds with the above program is to make up a dial plate to cover the regular input-output dial. To make such a dial plate, follow these steps:

1. Measure the distance between the two points to be used as check-points.
2. Correct this distance from whatever unit it was measured in to miles. For example, if the distance was measured in yards, divide this distance by 1760 (the number of yards in a mile) to get the distance in miles.
3. Count the number of complete revolutions which your rotary switch makes in one minute and note this number. Since the program above indicates a revolution of the rotary switch when the dial has completed 15/16 of a revolution, we must multiply the number of complete revolutions in a minute by 16/15 to obtain a "revolutions per minute" figure for the dial plate.
4. Multiply the "revolutions per minute" figure which you have obtained by 60 to get the number of revolutions per hour.
5. Multiply this last "revolutions per hour" figure by the distance between the two checkpoints expressed in miles. This will give you the speed in miles per hour which a car would be traveling if the rotary switch turned exactly one revolution while the car passed between the two checkpoints.

For example: assume that the checkpoints are 50 yards apart, and that the rotary switch turns 45 full revolutions per minute. Then:

$$50 \text{ yards} = \frac{50}{1760} = .029 \text{ miles}$$

$$45 \times \frac{16}{15} = 48 \text{ revolutions per minute}$$

$$\text{revolutions per hour} = 48 \times 60 = 2880$$

And if a car were to pass from the first checkpoint to the second checkpoint while the rotary switch dial moved from 0 to 15, its speed would be:

$$.029 \times 2880 = 83.52 \text{ miles per hour}$$

To find the car's speed if the dial turned through two revolutions—from 0 to 15 and then from 15 to 14—we will divide the last figure by 2:

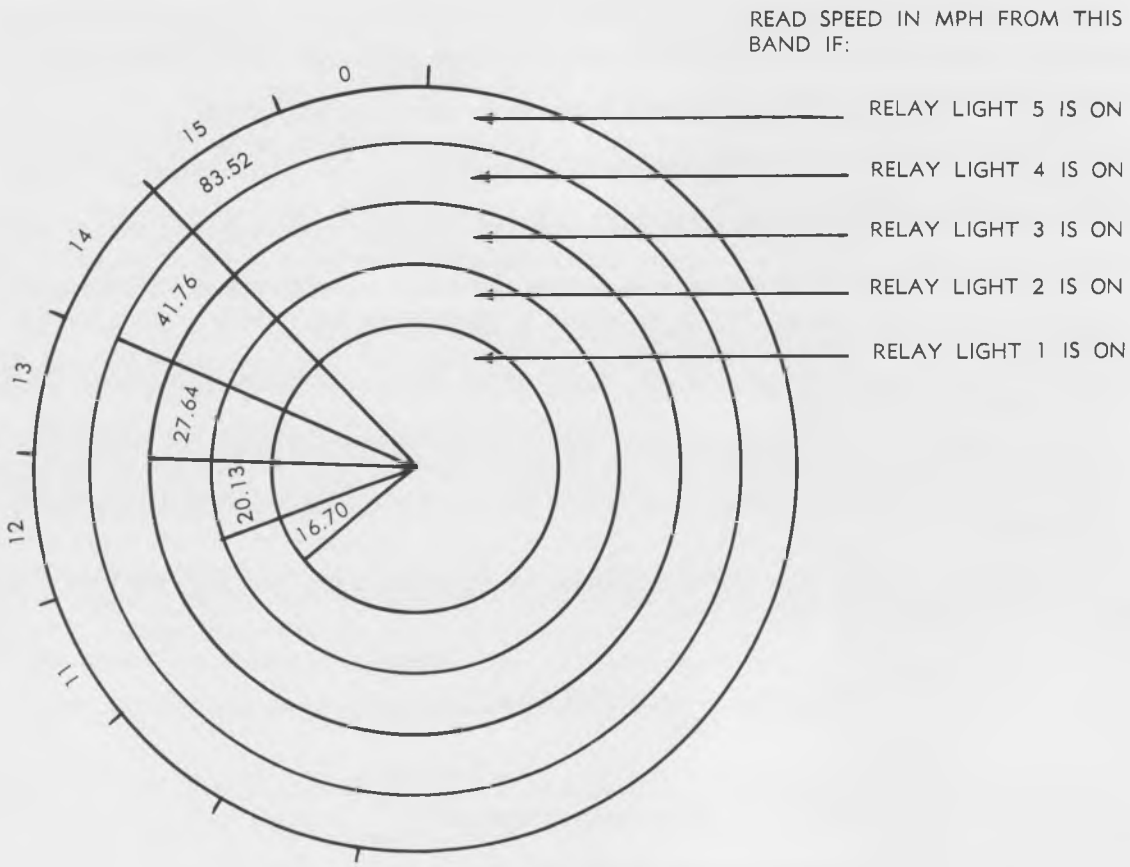
$$\frac{83.52}{2} = 41.76 \text{ miles per hour.}$$

Similarly, if the dial turns through three revolutions—from 0 to 15, then from 15 to 14, then from 14 to 13—the speed of the car is:

$$\frac{83.52}{3} = 27.64 \text{ miles per hour.}$$

The dial plate will thus have a sequence of speeds depending upon the number of revolutions the rotary switch makes as the car passes between the check points. Since the program is such that the relay indicator lights count the number of revolutions of the rotary switch, we will know which speed to read by observing which relay lights are on.

The dial plate for the example above would look like this:



DIAL PLATE—AUTOMOBILE SPEED TIMER

To time an automobile's speed:

Push pushbutton 6 as a car passes the first check point.
Push pushbutton 5 as the car passes the second check point.
Read the speed of the passing car from the dial plate: the speed is read from the band corresponding to the lowest numbered relay indicator light which is on.

Experiment 7: The Train Gate Control

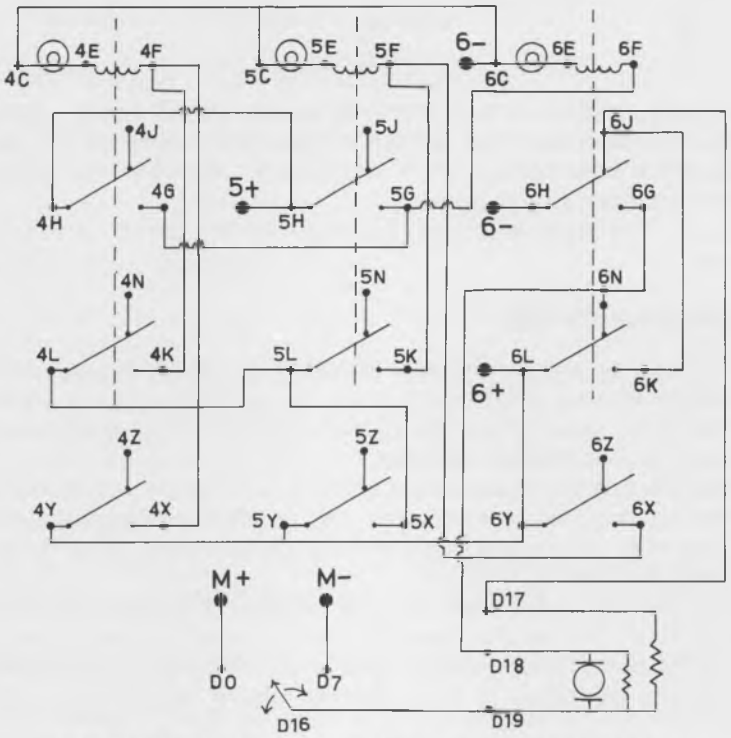
The gate control at a train crossing provides a simple example of a function known as "environmental sensing." A device with environmental sensing is one which is capable of receiving information about its environment and then acting on the basis of the information. In a large-scale computer, the actual "sensing" device would ordinarily be a specially-designed input unit which would automatically feed environmental information into a computer which was pre-programmed to handle it.

An extremely complex version of environmental sensing is the control system used in an automatically controlled oil refinery process. Input in the form of fluid flow, temperature, pressure, etc., is feed into a computer which processes this information and feeds its results into specialized output units which adjust valve settings, re-route fluid flow, or, in case of emergency, turn on various safety devices.

In the case of the simple train gate control, the control device must be capable of sensing the presence of a train and, when it does, must act to close the gate. The actual "input unit" for this type of system is generally a set of pressure switches under the rails. These switches remain on as long as a train's weight is on them. In actual practice there are many sets of switches located up and down the track, spaced less than a locomotive's length apart. When any of these switches is activated, the gate closes and remains closed until none of the switches is on.

For demonstration purposes, we can assume that there are only three switch locations: one each on either side of the crossing, and one in the middle of the crossing. The switch in the middle of the crossing must be so designed that cars passing over it will not close the gate.

The circuit diagram and program for a train gate control system are:



CIRCUIT DIAGRAM—TRAIN GATE CONTROL

Program:

4C/5C	5F/5K	6J/D17
4F/4X	5G/6F	6L/6+
4F/4K	5H/5+	6L/6Y
4G/5G	5L/5X	6N/D18
4H/5H	5Y/6Y	D0/M+
4L/5L	6C/6—	D7/M—
4Y/5Y	6G/6N	D16/D19
5C/6C	6H/6—	
5F/6X	6J/6K	

To use the program:

Turn power on and set the pointer knob of the rotary switch at 0.

The rotary switch at 0 represents the gate up. To indicate that a train approaches from the left and continues through the crossing, push pushbutton 4, 5, and 6 as follows:

1. Push pushbutton 4.
2. While holding pushbutton 4 down, push pushbutton 5.
3. Hold pushbutton 5 down and release pushbutton 4.
4. While holding pushbutton 5 down, push pushbutton 6.
5. Hold pushbutton 6 down and release pushbutton 5.
6. Release pushbutton 6.

The pattern for a train moving from left to right through the crossing may be expressed as the following sequence of operations:

Pushbutton 4	Pushbutton 5	Pushbutton 6
0	0	0
1	0	0
1	1	0
0	1	0
0	1	1
0	0	1
0	0	0

Notice that the gate closes (the pointer knob moves to the "gate closed" position—7) as soon as the first switch is activated and remains closed until none of the switches is activated The same will be true for a train moving from left through the crossing, or for a train which enters the crossing, stops and backs out again.

Push pushbutton 5. This represents a car passing over the center of the crossing. Notice that the gate does not close.

Experiment 8: The Two-Floor Elevator

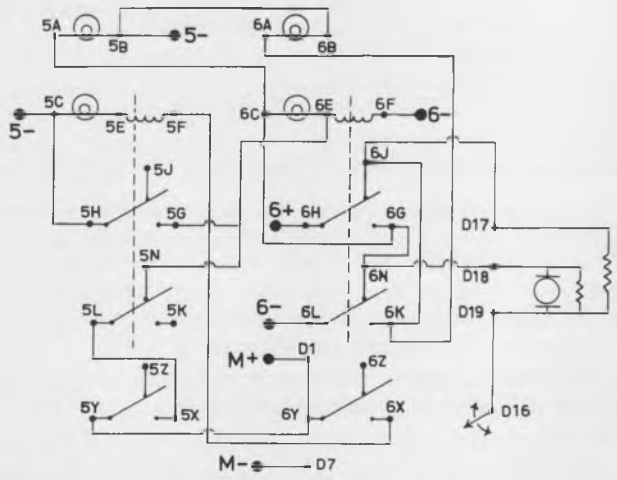
A number of familiar devices exhibit an important characteristic of a computer - memory. One of the most common devices which has a memory built into it is the automatic elevator. The complexity of the controlling mechanism for an automatic elevator depends primarily on its size —that is, the number of floors which it services.

The simplest possible automatic elevator control, of course, would be for a two-floor elevator. The system would then be required to remember only at which of ***two*** floors it was stopped, and it would not need to remember commands from various other floors. The program which follows simulates this basic elevator control circuit.

The rotary switch dial at 1 indicates that the elevator is at the first floor; the dial at 7 indicates that the elevator is at the second floor. Similarly, binary output light 5 on indicates that the elevator is at the first floor; binary output light 6 on indicates that the elevator is at the second floor. These lights are analogous to the indicator above an elevator door which tells you where the elevator is.

Pushbuttons 5 and 6 serve a dual purpose; they represent the "call" buttons on the floors, as well as the control buttons inside the elevator. That is, pushbutton 5 is used to direct the elevator to the first floor—as would be done ***either*** by pushing a "call" button on the first floor ***or*** by pushing the "first floor" button inside the elevator. Similarly, pushbutton 6 is used to direct the elevator to the second floor.

The circuit diagram and program for the two-floor elevator control system are:



CIRCUIT DIAGRAM—TWO-FLOOR ELEVATOR CONTROL SYSTEM

The program is:

5A/6C	5G/6E	6F/6—	6N/D18
5B/5—	5L/5X	6G/6N	6Y/D1
5B/6B	5N/6E	6H/6+	D1/M+
5C/5—	5Y/6Y	6J/6K	D7/M—
5C/5H	6A/6K	6L/6—	D16/D19
5F/6X	6C/6G	6J/D17	

To use the program:

Set the "elevator" at the first floor by setting the pointer knob of the rotary switch at 1. To call the "elevator" to the second floor, push pushbutton 6. The "elevator" will immediately move to the second floor (the pointer knob will point to 7). To return the "elevator" to the first floor, push pushbutton 5.

Experiment 9: The Three-Floor Elevator

The requirements for the previous elevator control system were very simple. Since there were only two possible locations for the elevator, the control system's memory had only to remember whether the elevator was up or down. If we add only one more floor to the system, however, the necessary circuitry becomes much more complicated. The system must be able to sense where it is, remember its destination, and then must be able to move to its destination.

As before, the pushbuttons will be used to represent both the call buttons on the various floors and the control buttons inside the elevator. Whenever a pushbutton is pushed, the system will go through the following steps:

1. The system will determine on which floor the elevator is presently located.
2. It will compare its present location with its destination (as indicated by which pushbutton was pushed).
3. If the destination differs from present location, the system will store the destination and proceed with step 4.
4. The system will select the direction in which it is to move (up or down).
5. The "elevator" will begin to move in the selected direction.
6. As the "elevator" begins to move, the pushbuttons will interlock so that the system will ignore further calls.
7. The "elevator" will continue to move to its stored destination.
8. When it reaches the stored destination, the "elevator" will stop and clear the destination from its memory.
9. As the "elevator" stops, the pushbutton interlock will clear and the system is ready for a new command.

The program for the three-floor elevator control system is:

1A/1E	2B/2—	3F/3G	5F/6C
1B/1—	2C/3C	3G/3X	5G/5N
1C/2C	2E/D7	3J/4E	5J/5K
1E/D1	2F/2G	3Y/4J	5J/D17
1F/1G	2G/2X	4C/5F	5L/5—
1G/1X	2J/3H	4F/4+	5N/D18
1H/1+	2X/6H	4G/4H	6C/6—
1H/1L	2Y/3Y	4H/4+	6F/D12
1J/2H	3A/3E	4L/D19	D0/D1
1K/5E	3B/3—	4N/D18	D6/D7
1Y/2Y	3C/4C	5C/5G	D12/D13
2A/2E	3E/D13	5E/6G	D16/M—

	call button	indicator light	elevator location points
Floor 1	pushbutton 1	output light 1	D0-D1
Floor 2	pushbutton 2	output light 2	D6-D7
Floor 3	pushbutton 3	output light 3	D12-D13

To use the program:

Set the "elevator" at the first floor by setting the pointer knob at D0. Turn power on. Output light 1 will come on indicating the location of the "elevator". To call the "elevator" to the third floor, push pushbutton 3. To direct the "elevator" to the second floor, push pushbutton 2.

Notice that, even with the relatively complex circuit above, the system is still not able to accept more than one command at once. Much more capacity is required for a control system which will remember successive commands and select the appropriate next move.

Experiment 10: The Automatic Toll Collector

Most of the specialized circuits discussed so far have been designed to perform various control functions. This next circuit, however, is designed to function as a decision-maker. The automatic toll collector has the responsibility of deciding whether or not a combination of coins given to it is sufficient to pay the toll. Ignoring the very specialized problem of recognizing various coins, let us consider the decision-making problem which the automatic toll collector must solve.

We will assume that a 10¢ toll is to be collected. This toll can be paid by depositing:

one dime

two nickels

one nickel and five pennies

ten pennies

The automatic toll collector must examine the input (the coins) and decide if it is adequate. If the payment is sufficient, the system must indicate to the driver that he may proceed. However, if a car passes through the toll gate without paying sufficient toll, the system must sound an alarm so that the guards will be notified immediately.

The program for an automatic toll collector is:

1A/3K	2H/2+	4A/4E	5J/5X
1A/2E	3A/3E	4B/5B	5J/6H
1B/1E	3B/4B	4C/5C	5L/6G
1B/2B	3C/4C	4E/4J	5X/D5
1C/1J	3E/4G	4H/5N	5Y/6Y
1E/2C	3F/3H	4K/5N	6A/6E
1F/1H	3F/4F	4Y/5Y	6C/6—
1J/2K	3G/3L	5A/5E	6E/6J
2A/3N	3G/3+	5B/6B	6X/D16
2B/3B	3J/4H	5C/6C	6X/D18
2C/3C	3J/4X	5E/6G	D0/D19
2F/2G	3L/3Y	5F/5H	D5/D10
2F/4N	3N/6Y	5F/6F	D17/M—
2H/2L	3X/4L	5G/5+	

Output light 1 ON represents "Thank You—Go Ahead"
Output light 2 ON represents "Stop—Pay Toll 10¢"

Relay 1 and relay indicator light 1 ON represents "Alarm"

Pushing pushbutton 3 represents "Drive Away"
Pushing pushbutton 4 represents depositing one dime
Pushing pushbutton 5 represents depositing one nickel
Pushing pushbutton 6 represents depositing pennies (number of pennies indicated on rotary switch)

To use the program:

Turn power on. Output light 2 comes on indicating "Stop—Pay Toll 10¢."

Deposit 10¢ in toll:
to deposit one dime, push pushbutton 4
to deposit two nickels, push pushbutton 5 twice
to deposit one nickel and five pennies:
push pushbutton 5 once; then set the rotary switch dial to 5 and push pushbutton 6
to deposit ten pennies:
set the rotary switch dial to 10 and push pushbutton 6.
When 10¢ has been deposited, output light 1 will come on indicating "Thank You—Go Ahead."
Push pushbutton 3 to "Drive Away."
The toll collector will re-set.

If you try to drive away without depositing enough money, the alarm will sound. To turn off the alarm, deposit 10¢. If you deposit more than 10¢, the automatic toll collector will accept the extra money and do nothing about it.

Experiment 11: The Telephone Dialing System

The telephone dialing system is an excellent example of a highly specialized computer system capable of handling a particular type of complex problem. The entire dialing system is far too complicated for this discussion. However, we can examine one small segment of the problem: the conversion of a dialed number into a type of binary code which can be understood by the circuits of the telephone exchange.

The telephone dialing system uses a special binary code known as the ***Gray Code.*** This code is binary in nature; that is, it uses only the digits 0 and 1. However, it does not follow the usual rules for the development of a number system. The system is so designed that when moving from any number to the next highest number there is a change of only ***one*** bit.

For this experiment, we will use a three-bit Gray Code which will allow us to count from 1 through 7. We will use the Gray Code equivalent of zero for 8 so that we can represent the numbers 1 through 8:

Decimal Number	Three-Bit Gray Code
1	001
2	011
3	010
4	110
5	111
6	101
7	100
8	000

The program which follows is in essence a conversion program from decimal to Gray Code. The decimal number "dialed" will be displayed in Gray Code on output lights 4, 5, and 6. This represents only the first step in the handling of dialed numbers in a telephone system: the conversion of the dialed number to a form which can be processed at the telephone exchange.

The program is:

```
1C/1—   2N/6G   4E/5E   6L/6W
1C/1H   3A/6E   4F/5F   6R/6U
1F/D0   3B/3—   4H/6N   6S/D1
1G/4B   3C/4C   5A/5K   6U/D18
2C/3C   3E/4J   5B/6B   6V/D2
2E/3E   3F/4G   5C/6C   D1/D3
2F/3F   3G/4A   5F/5K   D2/D4
2F/3G   3H/3+   5H/6T   D3/D5
2G/2N   3K/4E   5L/6H   D4/D6
2H/5G   3L/6K   6A/6F   D5/D7
2J/2K   3N/4F   6C/6—   D6/D8
2J/6E   4B/5B   6F/6G   D16/M+
2L/5J   4C/5C   6H/6+   D17/M—
```

To use the program:

With power off, set slide switch 6 RIGHT. Set the rotary switch dial at 0. Turn power on.

Slowly turn the rotary switch dial to the number you wish to "dial". (Be sure that the pointer knob is pointing directly at the number.)

Set slide switch 6 LEFT.

The rotary switch will return to 0 and the number "dialed" will appear in Gray Code on output lights 4, 5, and 6.

(Note: output light 3 should be ignored.)

To "dial" another number, set slide switch 6 RIGHT and re-set the system by turning power off and on again.

Special-Purpose vs. General Computers

In the preceding experiments, various devices which exhibit certain computer-like characteristics have been examined. These devices share one common characteristic: they are designed to perform a specialized and routine function. None is capable of performing all of the major computer functions discussed in Book II. Moreover, none is capable of performing generalized problem-solving functions.

In each case, though, a large-scale generalized computer could easily be programmed to solve the specific problem. It is just not practical to have a full computer system to, for example, control the lights in a home. Therefore, simple computer-like devices are designed to perform these specific tasks. Technically, any of these simple, highly specialized devices could be called a computer since it can receive input, can process and/or store the input, and can yield output. More accurately though, these devices should be called "computer-like" because of their extremely limited capabilities.

The computer systems discussed in Book II were ***general*** computers. That is, their input and output capabilities allow them to handle a wide range of communication, and their processing and storage units are such that various types of problems can be handled. These ***general*** computers are used primarily in situations where there is likely to be a great deal of variety in the type of problem to be solved. A research organization, for example, would require the versatility of the general computer if it were using the computer for "one-shot" problems (problems which would only be programmed onto the computer once).

For routine problem-solving—for example, making out a weekly payroll—an organization may find a ***special-purpose*** computer most practical. A special-purpose computer is just what its name implies: a computer designed to handle a limited number of problems in a particular fashion.

Special-purpose computers are often made to order, in which case computer "building-blocks" are frequently used. Computer building-blocks are simply generalized logical units which perform particular tasks. A building-block might be a half-adder, or one bit of an accumulator, or a single flip-flop. Any number of building-blocks can be combined to produce a special-purpose computer to handle a specific problem.

When a special-purpose computer is in the design stage, the phrase "black box" is often used. This is just a convenient, if amusing, method of expressing the particular problem at hand. When dealing with a black box, the designers are not concerned with ***how*** the system will actually go about solving the particular problem; they are concerned rather with ***what*** the system must do. The black box is essentially the processing section of a particular part of the problem-solving unit. The inputs and outputs are specified; later in the design stage, building-blocks will be used to carry out the requirements of the black box.

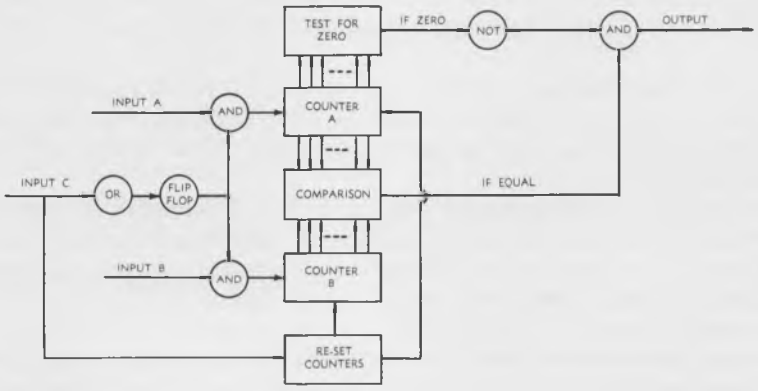
We might define a black box as follows:

a black box is a device with specified inputs and outputs; the outputs are expressed as functions of the inputs. The inner workings of the black box are unspecified.

To illustrate both ***black boxes*** and ***building blocks*** let us consider the problem of designing a special-purpose computer which will do the following.

The system will have three inputs (A, B and C) and one output. Each of two inputs (A and B) is a series of electrical pulses of different rates. The third input (C) is a re-set input. Inside the computer are two counters: one counter counts the number of pulses from input A; the second counter counts the number of pulses from input B. If at any time the contents of counter A equal the contents of counter B, but do not equal zero, both counters will stop and a pulse will appear as output. Input C will then set counters A and B to zero and the system will begin counting inputs A and B again.

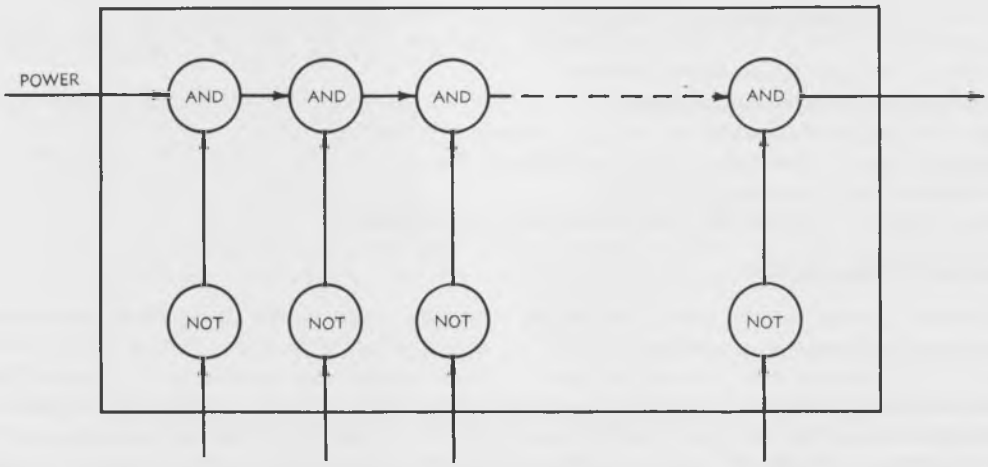
The first step in the design of this special-purpose computer will be to "block diagram" the problem; that is, we will represent the problem graphically so that we can see just what will be required. The block diagram of our problem looks like this:



In this diagram, the black boxes are represented by the rectangles; the building blocks are represented by the circles. The building blocks are simply basic logical units which perform a definite function. The black boxes, however, are specified only as far as input and output; the method for arriving at the output as a function of the input is not defined. When we use a black box to represent some part of our computer we are saying only "insert some circuitry which will do the necessary job".

At a later stage in the design of our computer, we can take each black box apart and select the appropriate building blocks which are required. In actual practice, this job would probably be done by the design department of the firm from which we were buying our special-purpose computer.

As an example of taking a black box to the building block stage, the "Test for Zero" black box might look like this:



"TEST FOR ZERO" BLACK BOX

When the actual computer is built, the appropriate components will be wired together exactly as indicated in our sketches. Notice that when completed, our special-purpose computer is actually only a particular combination of the basic logical units which we examined in Book III. A general purpose computer, on the other hand, is a large number of basic logical units combined in such a way that particular combinations of them can be automatically effected by internal programming.

2. COMPUTER APPLICATIONS IN BUSINESS AND INDUSTRY

The Computer System in the Business World

General purpose computers and electronic data processing machines have found extensive application in the business world, performing a wide variety of tasks with great speed and accuracy. In this section, we will discuss the kinds of functions which computers can effectively handle in a business operation.

A basic answer to the question, "What can computers do in business?" might very well be "any repetitious task". In general, the computer is best equipped to handle any situation in which the same basic operations are to be performed many times using different data, but following identical, prescribed procedures.

Similarly, a valid answer to the question, "What is it unreasonable to expect computers to do in a business situation?" would be, "to solve a unique or non-recurring problem." These answers are not necessarily valid in a research or scientific problem-solving situation where the complexity of a given problem may well justify the use of a high speed computer in order to obtain only one answer at a particular time.

However, in most business situations the problems to be solved are such that computers can be profitably employed in performing repetitive calculations which must be made in the course of daily operations. In such situations it is usually well worth the time spent in programming a computer to solve a particular kind of problem since the computer will solve the same problem many times in the future using whatever data is applicable at the time. On the other hand, the extensive programming time required to prepare a computer to solve a particular problem may make it economically unwise to program the computer to solve a unique problem. The computer can be a valuable tool for the businessman who knows how to use it and understands what it can and cannot do. It can, though, be a very expensive and highly inefficient window-dressing in the office of a businessman who does not fully understand both its potentialities and its limitations.

What Computers Can Do

Books III and IV examined specific arithmetic and logical operations which computers can perform. When these operations alone or in combination are used to solve a problem requiring large amounts of data at once or using different data over a long period of time, the right computer can be an invaluable asset.

In general, an electronic data processing machine can do the job more effectively than a human clerk if the job to be done involves:

Repetitive arithmetic calculations.
Repetitive decisions based on explicit decision rules.
Categorizing or cataloging large numbers of items.
Comparing large numbers of items.
Tabulating or summarizing quantitative information.

What Computers Cannot Do

In general, an electronic data processing machine is not worth using when small amounts of data are to be analyzed in a unique manner at a single point in time. A few people with adding machines or calculators can process several statistical tests on hundreds of pieces of data more rapidly than a programmer can prepare a high speed computer to process that data and card punch operators can prepare the data for input into the electronic data processing system.

The section in Book III dealing with sufficient information is particularly applicable in business situations. In order to solve a problem a computer needs both data and instructions. The

importance of this simple fact is often overlooked, sometimes with disastrous results. The basic requirement of sufficient information leads to a list of jobs which a computer should not be called upon to perform:

Non-repetitive calculations on small amounts of data.
Problem-solving where inadequate data is available.
Problem-solving where the exact procedure to be followed can not be specified.
Decision-making in situations where qualitative factors must be considered.

Truth vs. Output

It is an unfortunate fact that some people consider information to be accurate simply because it appears on a printed page. Since most businessmen are exposed to computer output only in the form of printing prepared by an output device attached to the computer, there is a danger of accepting the results of the computer operation without first questioning (1) the data which the computer was given and (2) the methods, procedures, or techniques which the programmer instructed the computer to use in order to arrive at its conclusion.

The answer which a computer produces is only as good as the data which it was given and the procedures which it was instructed to follow. The computer is a completely faithful servant. It does everything it is told to do and, providing that the instructions which it is given are not internally inconsistent, it will not question the legitimacy of either the data or procedures.

Will Computers Make Management Obsolete?

Some businessmen view computers as antagonists ready to usurp power and reduce the management function to punching computer input cards. These businessmen, who seem to fear that their conference room will be replaced by a panel of blinking lights, cannot appreciate the legitimate potentialities of electronic data processing equipment because they fail to understand the basic and important limitations of the potential.

Rather than detracting from the importance of management personnel, the computer, if properly used, can add immeasurably to their ability to handle a wider range of problems more effectively, to explore greater numbers of alternatives and to make decisions based on more information, more completely analyzed. The computer simply removes the tedium of repetitious considerations and permits management personnel to focus their attention on policy-making and effective utilization of more and better information in more competent problem-solving.

Examples of Computer Handling of Business Problems.

The Payroll Problem

Computer handling of a large payroll processing job provides a good example of an operation which computers are well-qualified to perfom in a business situation. Assume that the computer is given the following information for each employee on punched cards or that it already has this information in storage or on magnetic tape:

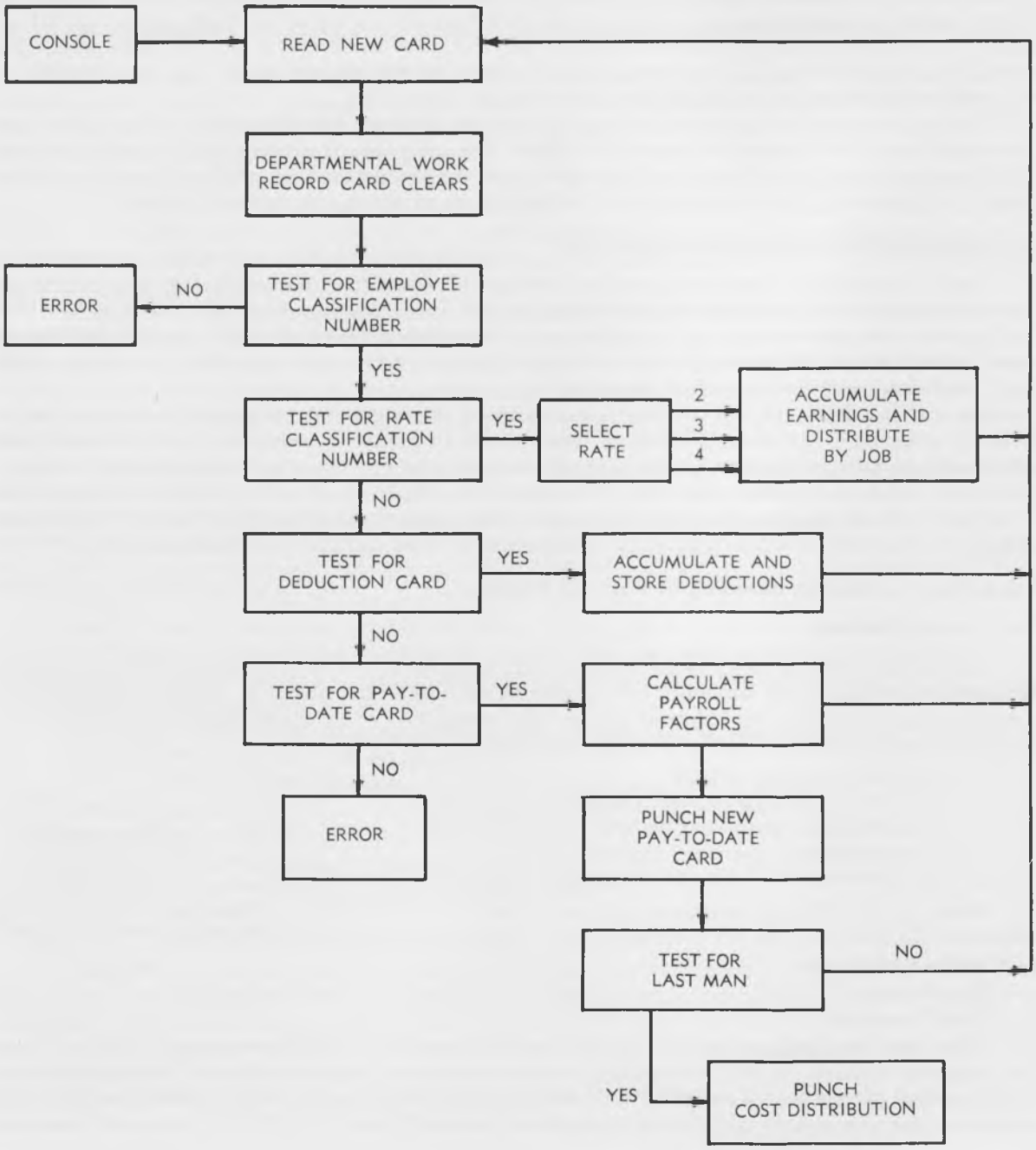
1. An identification number
2. The employee's guaranteed pay rate
3. The employee's regular pay rate
4. The employee's overtime pay rate
5. An itemization of all authorized deductions for that employee.

Assume also that the computer has been programmed so that storage locations are available in which the following information may be accumulated for each employee:

Attendance hours
Regular pay
Overtime pay.

With this information stored in the computer, work records from various departments are fed into the machine. As this information is received by the machine, identification numbers on departmental job cards are matched with employee identification numbers stored in the computer and the appropriate calculations are undertaken as follows:

1) A comparison instruction is used to determine whether the man receives his guaranteed rate or a job rate.
2) The appropriate rate is multiplied by the number of hours in regular and overtime activity and the appropriate amounts are stored in the reserved location.
3) A record of total hours on each job is accumulated and this information is stored in the appropriate register.
4) After total earnings have been determined, deductions are processed and final pay records and labor costs records are supplied as output. The following flow chart summarizes the operation of this program:



This flow chart illustrates a relatively comprehensive job done by a computer using only the simplest of operations. The only logical operation used to perform the job outlined above is a "comparison" operation which is simply a logical AND. In performing the comparison the computer compares each bit of the register containing the identification number for a particular man with each bit of the register containing the identification number on the job record. If each bit of the man's identification number register ***AND*** each bit of the register containing identification number from the job record are the same, the comparison is satisfied and the computer proceeds with the program.

The flow chart shows graphically one of the advantages of a computerized payroll system—the ability to perform more than one job at one time. Notice that, in the same operation, the payroll is processed and the cost distribution is calculated. This is a valuable trait, particularly to a large firm which has many operations, each of which must be correctly accounted for.

The Inventory Problem

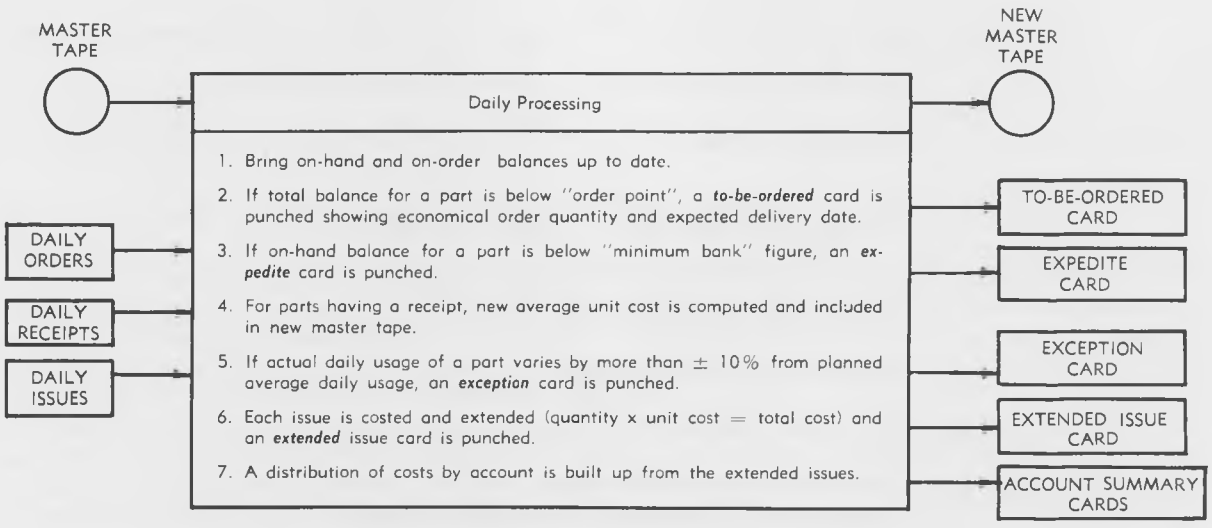
Maintaining an appropriate inventory for several thousand parts used in an active production line is a very real problem to businessmen who must minimize time delays, storage costs, and purchase costs in lot sizes, to mention only a few factors. An appropriate computer system can handle the routine data flow and standardized decision-making ***after*** the correct program has been written.

It cannot be over-emphasized that a computer system is only as good as its input. This is definitely true of an inventory control system in which the computer yields purchase decisions as output. If the computer can be given precise decision rules for purchase, as well as accurate information regarding the inventory, it can easily produce routine purchase decisions.

A particular advantage of a computerized inventory control system is the speed with which information can be processed, making possible daily control of large inventories which would otherwise be practically impossible.

Various inventory control systems use punched cards, paper tape or magnetic tape—or combinations of these. As an example of a functioning system, let us consider one which uses magnetic tape plus punched cards.

A complete record of the inventory is kept on a master tape which is up-dated daily. Additions to and issues from inventory are recorded first on punched cards and then fed into the master tape. Daily output from the processing is in the form of punched cards which give order instructions, notice of abnormal usage and account summaries, as well as a revised master tape. The following chart shows the input, processing steps and output for such a system:



Notice that of the five output card types two are "warning" cards. Both the ***expedite*** and ***exception*** cards alert management to the fact that some part of the system requires attention. As in the payroll problem, cost distributions are built up as the inventory processing is done.

Check Processing

The use of computers to facilitate the physical handling of printed materials is effectively illustrated in the computerized processing of bank checks. The actual job to be done is a bookkeeping procedure and, as such, the instructions and procedures are not unlike those of the payroll problem.

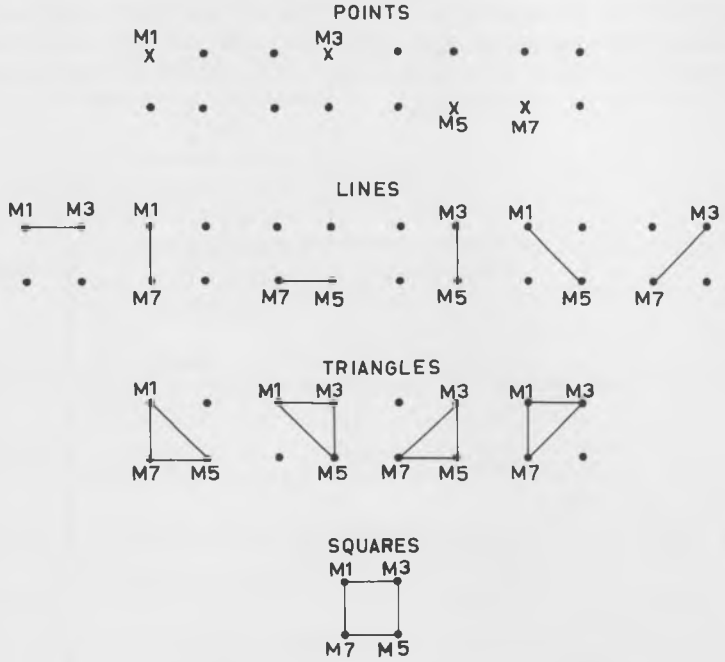
The distinguishing aspect of the bank check problem is that the "input" is a printed paper (a check) rather than a punched card. In order to handle the problem then, the computer must be able to "read" the printing. (This ability of a computer to "read" printing also arises in the processing of credit billings.)

Previously, we have discussed how a computer "understands" information fed to it by recognizing the presence or absence of holes in a card, magnetic marks, etc. at particular locations. The problem in "reading" printed characters is basically the same: the computer must be programmed to recognize and interpret a "code" which identifies the input symbols. The "code" in this case is the standard alphabet or standard arabic numbers.

The first step in communicating information to a computer as input in printed form is to position the printing so that the computer can observe it. This is directly analogous to placing punched cards in a reader, or loading magnetic tape on a magnetic tape reader. A variety of "reading" systems for printed material are used, and most require placing the printed material in a device which either holds it or moves it to a light source or magnetic sensing device.

Once the material is positioned in front of the sensing equipment, these specialized input devices communicate to the computer the presence or absence of a magnetic or ink spot at a particular location in a matrix of light or of magnetic sensing elements. The process by which the computer recognizes the symbols positioned against the matrix can be demonstrated on MINIVAC's game matrix.

For simplicity, consider a matrix composed of four sensing points: M1, M3, M5 and M7. Using this matrix, MINIVAC can be programmed to recognize simple geometric shapes. The possible geometric shapes which MINIVAC can recognize are:



GEOMETRIC SHAPES—VARIOUS FORMS

These various forms would be communicated to a commercial computer as information regarding the presence or absence of spots of ink or magnetic material at the "sensing points" of the matrix. The "sensing points" of MINIVAC's matrix respond to electricity rather than light or magnetic ink. Therefore we will supply current to the various sensing points of the matrix to simulate the effect produced by the sensing elements in a printed character reader.

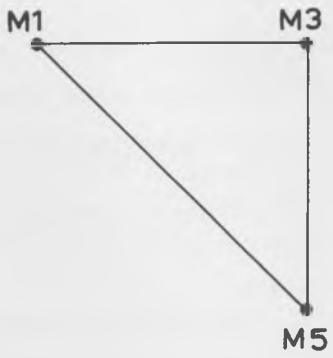
Light 1 on will indicate that MINIVAC senses a point
Light 2 on will indicate that MINIVAC senses a line
Light 3 on will indicate that MINIVAC senses a triangle
Light 4 on will indicate that MINIVAC senses a square

The program for the geometric shape recognition circuit is:

1A/2K	2A/2G	3H/5G	5H/6G
1B/1—	2B/2—	3J/3K	5J/5K
1C/2C	2C/3C	3L/5K	5L/6J
1F/2F	2F/M7t	3N/4G	6C/6—
1G/4A	2H/3N	4B/4—	6F/M1t
1H/3G	2J/2K	4C/5C	6H/6+
1J/1K	2L/4J	4F/M5t	M10/M+
1K/3A	3B/3—	4H/5N	
1L/3K	3C/4C	5C/6C	
1N/2G	3F/4F	5F/M3t	

To use the program:

Indicate a geometric shape to MINIVAC by making connections from M10 to the top terminals of the various points on the matrix. For example, to indicate the triangle:



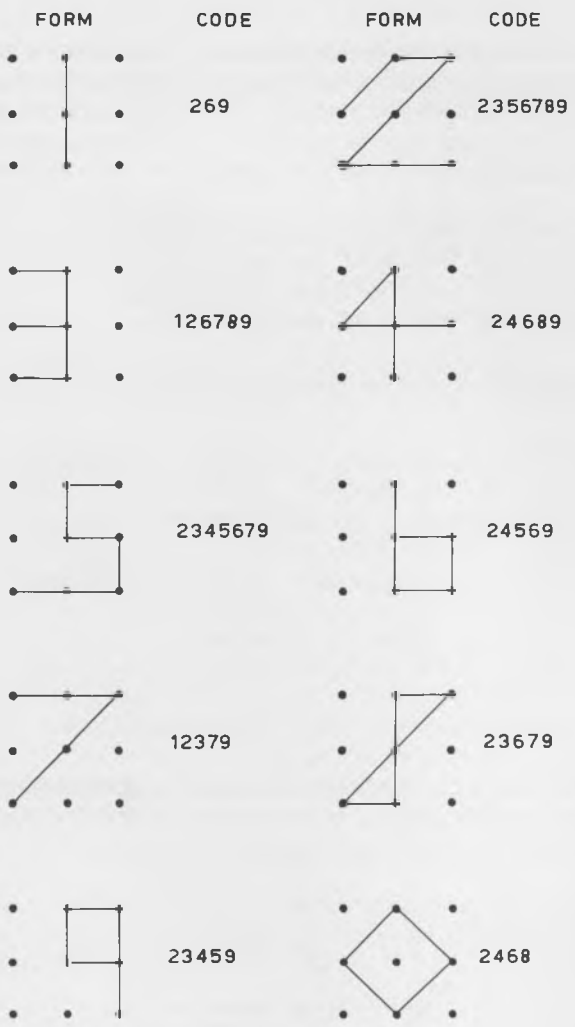
Make the connections:

M10/M1t
M10/M3t
M10/M5t

Light 3 will come on indicating that MINIVAC senses a triangle.

This program for the recognition of geometric shapes is similar in function to the Quantity Recognition circuit discussed in Book III. A large computer would determine how many points in its matrix of sensing elements were being activated by the presence of ink or magnetic material. With the limited size of the matrix used in the above example, a properly programmed computer can recognize a particular geometric form.

To recognize printed characters requires an expanded matrix—one large enough so that each possible character can be uniquely defined. We can uniquely define the numbers 0 through 9 using the full game matrix on the MINIVAC. The result will be ten forms, each with a unique recognition code.



CHARACTER FORMS AND CODES

Notice that not only the form, but also the position of the characters is important. For example, the character 1 must be centered in the block of sensing elements.

The experiment which follows uses a more complex variation of the quantity recognition circuit to recognize arabic numerals which are "drawn" on the game matrix.

Experiment 12: Arabic Numeral Recognition

In this experiment we will program MINIVAC to recognize arabic numerals which are "drawn" on the matrix. The nine terminal locations of the matrix will provide the sensing elements for the computer. The program will allow MINIVAC to recognize numerals according to the code represented above.

The program for the recognition of arabic numerals is:

1C/2C	3C/4C	4H/5J	6H/6—
1F/M9t	3F/M6t	4L/5K	6X/M10
1G/6com	3G/D6	5C/6C	6X/D17
1H/4K	3H/4N	5F/M4t	6Y/6+
1J/D0	3J/D9	5G/D5	6com/D4
2C/3C	3K/D8	5H/6G	M10/M11
2F/M5t	3L/4J	5L/6J	D16/D19
2G/D2	3N/D7	5N/D1	D18/M—
2H/4G	4C/5C	6C/6—	
2J/D3	4F/M8t	6F/M7t	

Now connect four short programming wires to M10, leaving one end of each free. Also, connect four short programming wires to M11, leaving one end of each free.

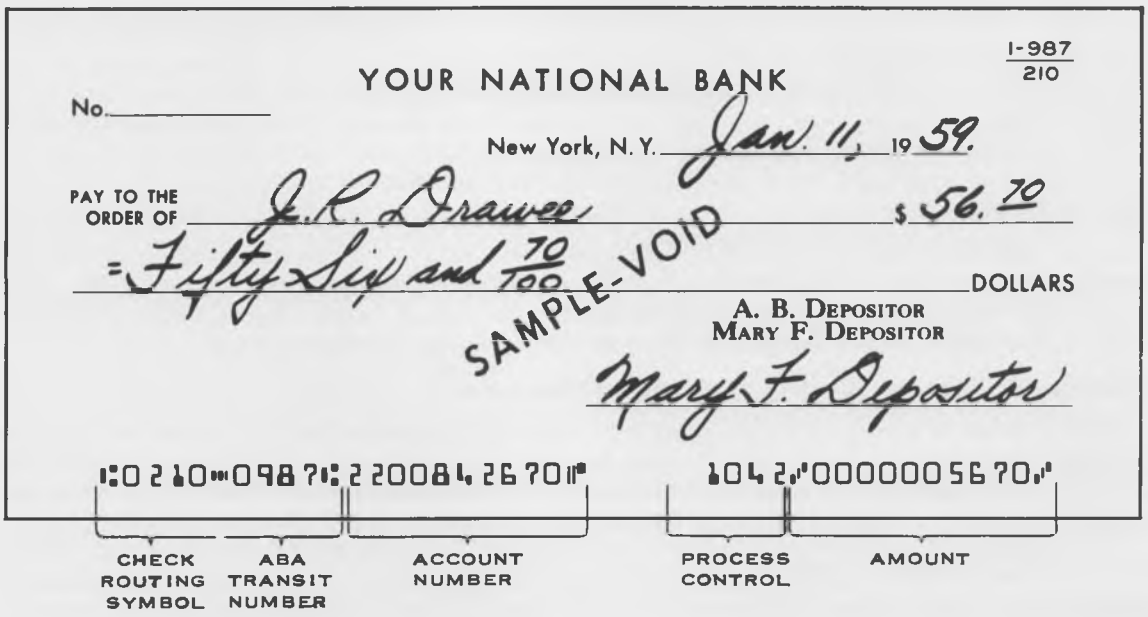
To use the program:

Turn power on. Select a number from 0 to 10. "Draw" this number on the top terminals of the matrix with the free wires from M10 and M11, using the arabic numeral code previously presented. Push pushbutton 6. The pointer knob of the rotary switch will turn to the number "drawn" on the matrix.

To try another number, be sure that all eight wires to M10 and M11 are free. Then "draw" the new number on the top terminals of the matrix and proceed as above.

You have undoubtedly noticed that some of the numbers used in the above experiment are peculiarly shaped. For instance, the "8" is a bit lop-sided. However, for the computer to recognize the numerals, each character ***must*** be presented according to the code which the machine is programmed to understand. This is true of any device designed to "read" printing: the device will read only the particular type face which it is designed to understand.

Since each character must be unique, the type faces which are designed to be machine-read do not look exactly like the numerals with which we are generally familiar. An example of a specially-designed type face for use in a print-reader is that now used by many banks. In the illustration below, the characters are printed with a special magnetic ink for sensing by the reader:



BANK CHECK SHOWING SPECIAL TYPE FACE FOR MACHINE READING

Machine recognition of printed characters is presently limited to specially designed type faces. Once the printed information is recognized by the machine, modern computers can rapidly process this information. Banks now use computerized check processing systems, and the large credit card companies have found computerized handling of accounts to be a valuable tool. Research is presently being carried out on equipment designed to read any type face, and it is hoped that in a short time computers will be able to understand printed characters regardless of the type face.

Examples of Computer Handling of Industrial Problems

Although it is artificial to separate business from industry, we are establishing this distinction because of the types of problems which are handled by computer systems. The "business" applications discussed have involved primarily computer handling of data-processing problems.

The "industrial" applications which follow deal primarily with computerized control systems for production problems.

Experiment 13: Multiple Point Control

An expansion of the simple "Light on the Stairs" circuit (see Experiment 1) is used in industrial situations which require control from a number of points. An example of such a situation would be the problem of handling radioactive materials without approaching them. Intricate systems have been designed for this particular process; these systems consist of a great number of complex mechanical devices which allow an operator to handle these dangerous materials from behind safety shields.

Vital to such a remote control operation are, of course, a great number of safety devices—for instance, emergency cut-off switches as well as emergency "on" switches for certain safety mechanisms. For added safety, duplicate emergency switches are generally placed at various locations—at the actual equipment location, in the safety engineer's office, in the central power plant, etc. Thus, in case of an emergency, safety devices can be turned on or dangerous equipment can be turned off from the most accessible location and with the least possible danger to the employees.

This experiment demonstrates a basic control circuit which permits a single light (which could represent any safety device) to be turned on or off from twelve different "locations".

The program for a multiple point control system is:

1V/1Y	4A/6K	5E/6G	6A/6E
1W/2V	4B/4—	5F/6F	6B/6—
1Z/2Y	4W/5V	5F/5H	6C/6—
2W/3V	4Z/5Y	5G/5+	6E/6J
2Z/3Y	5A/5E	5J/6H	6H/6W
3W/4V	5B/5—	5W/6V	6L/6Z
3Z/4Y	5C/6C	5Z/6Y	6Z/6+

To use the program:

Set all slide switches to the RIGHT. Turn power on. Light 4 may be turned on or off by pushing **any** pushbutton or by moving **any** slide switch to the LEFT and then back to the RIGHT. In actual practice, these twelve switches would be located at various control points, rather than being grouped in one place. (Note: lights 5 and 6 are wired into the control circuit.)

Experiment 14: Sequence Control With Manual Operation

Another form of a control system used in an industrial process is one which insures that a predetermined sequence of operations is executed in the proper order. This type of system finds application, for instance, in a situation which calls for the combination of chemical compounds in a specified order—particularly when an error in the sequence would prove dangerous.

The program below provides constant control as would be required in the situation just outlined. The control circuit provides that the first step in the process be taken before the second can be begun, and so on through twelve consecutive steps. This particular circuit will not permit an error in sequence to be made. The system will not recognize a step taken out of order. Only the correct steps given in the correct sequence will be accepted as input. When all twelve steps in the operation have been completed, light 1 will come on indicating the end of the operation.

The program for a control system with manual operation is:

1A/1G	2F/2G	3H/4G	4com/5L	5S/5X
1B/1—	2F/3K	3N/4N	4R/3com	5T/6T
1C/2C	2H/3G	3com/4L	4S/4X	5Y/6Y
1F/1G	2N/3N	3R/4com	4T/5T	6A/6J
1F/2K	2com/3L	3S/3X	4Y/5Y	6B/6—
1H/2G	2R/5com	3T/4T	5C/6C	6C/6—
1com/2L	2S/2X	3Y/4Y	5F/5G	6F/6G
1R/6com	2T/3T	4C/5C	5F/6K	6F/6com
1S/1X	2Y/3Y	4F/5G	5H/6G	6H/6+
1T/2T	3C/4C	4F/5K	5N/6N	6R/1com
1Y/2Y	3F/3G	4H/5G	5com/6L	6S/6X
2C/3C	3F/4K	4N/5N	5R/2com	6Y/6+

To use this program:

Set all six slide switches to the RIGHT and turn power on.
Carry out the twelve steps in the operation by setting the slide switches and pushing the pushbuttons in this order:

Set slide switch 1 LEFT
Push pushbutton 1
Set slide switch 2 LEFT
Push pushbutton 2
Set slide switch 3 LEFT
Push pushbutton 3
and so on, through pushbutton 6.

When you have completed all twelve steps, light 1 will come on indicating "operation complete."

If you attempt to carry out a step in the wrong order, nothing will happen. The system accepts as input only the proper steps.
To re-set the system, set all slide switches to the RIGHT and turn power off, then on again.

Experiment 15: Automatic Sequence Control

The previous program provides basic control over an operation by requiring that a human operator perform the required steps in the appropriate order. However, it leaves execution of each of the steps to the operator.

In some complex industrial processes not only the sequence of operations, but also the timing of each step, is crucial. Under such circumstances it is often desirable to use a completely automated system to control the entire process. With the appropriate circuitry, each step in the operation will be performed in the specified sequence, with the specified delay between steps.

In the following program, pushing pushbutton 6 will start the automatic sequence. Lights 1 through 6 will come on in order, indicating that each step in the sequence has been completed. When the last light—light 6—comes on, pushing pushbutton 5 will turn the system off.

The program for automatic sequence control is:

1A/6G	2F/2—	4B/4—	5Y/M—
1B/1—	2G/5A	4C/4G	6B/6—
1C/1G	2H/3H	4C/5K	6C/6G
1C/2K	2L/D11	4F/4—	6C/6X
1F/1—	3A/4G	4H/5H	6F/6—
1G/6A	3B/3—	4L/D13	6H/6+
1H/1+	3C/3G	5B/5—	6H/6L
1H/2H	3C/4K	5C/5G	6K/D17
2A/5G	3F/3—	5C/D15	6Y/6+
2B/2—	3G/4A	5F/5—	D16/M+
2C/2G	3H/4H	5L/D14	D18/M—
2C/3K	3L/D12	5X/6E	

To use this program:

Turn power on. Start the system by pushing pushbutton 6. Lights 1 through 6 will come on in order, representing the completion of six successive steps. When light 6 has come on, push pushbutton 5 to stop the system.

Notice that the sequence can be stopped at any time by pushing pushbutton 5. This is analogous to having an emergency shut-off switch wired into the control circuit.

Control With Feedback

The three previous control circuits dealt with situations in which the sequence of operations was completely specified. Only the order of execution and the timing of each operation was controlled by computer programming. Many industrial processes, however, require execution of certain operations which must be determined by conditions occurring during the actual processing.

This would be the case if, for example, a specific ingredient were to be added to a chemical compound when the compound had reached a certain temperature.

Control in which information from the process determines execution of successive steps is known as "control with feedback." Information from the process is "fed back" to the control mechanism to modify successive operations, or to activate successive operations.

As in the sequence control systems, control systems with feedback can be completely automated or can be manually operated. We can demonstrate a manual control system with feedback using the "two-floor elevator control" circuit from Experiment 8. We will assume that the process to be controlled is dependent upon the continued oscillation of the rotary switch between 1 and 7.

To start the system, we will set the pointer knob at 1 and push pushbutton 6. This moves the pointer knob to 7, and light 6 will come on to indicate that the system is now at 7. Light 6 is, in this case, a "feedback indicator." As soon as light 6 comes on we must push pushbutton 5 to return the pointer knob to 1. When the pointer knob is at 1, light 5—another "feedback indicator"—comes on, telling us that pushbutton 6 must be pushed to turn the pointer knob to 7 again.

In an industrial situation, lights 5 and 6—the feedback indicators—would have special meaning:

> Light 6 on would indicate that the first of two steps had been executed and pushbutton 5 must be pushed to continue the process.
>
> Light 5 on would indicate that the second of two steps had been executed and pushbutton 6 must be pushed to continue the process.

Notice that in this manual control system with feedback, the system ***indicates*** to the operator that an operation has been executed. The operator must then carry out the next operation on the basis of this information which has been "fed back" to him.

Industrial processes which require split-second reaction to a particular signal are often designed so that the signal will automatically set off the appropriate next operation. In the illustration above, this can be done by a program which will automatically sense the position of the pointer knob and immediately act to change its position. An example of such a program is the metronome in experiment 3. The metronome circuit provides constant oscillation between two points on the rotary switch dial. The operator can change the period of oscillation by moving the slide switches, but the system automatically reacts to its own operation.

What is the "Best" Computer?

As more businesses turn to computerized operations, the question of the "best" computer arises more frequently. There can, of course, be no general answer to the question, since each firm must decide what problems it can best solve with a computer's help and then select the best system for its particular needs.

There are, however, three basic decisions to be made when the final selection of a computer system is to be made:

1. How large a computer system is required?
2. What speed of operation is required?
3. What kind of input the computer must be equipped to handle.

With these factors specified, the choice of the best system will be made simpler.

The first factor—size—can be broken down into two considerations: the size of the computer's memory and the size of numbers which the computer will be required to handle. In terms of memory size, the basic question to be answered is, "How much information will the system have to handle at any one time?" A firm with 1000 employees may want a system capable of processing the total payroll in a single operation; but if management is willing to process the payroll in two sections, a smaller system— one with less memory—could be used.

The size of numbers which the system will be required to handle raises the question of accuracy. If the system is to be used in applications where ten significant figures are sufficient, it

would be unnecessary to pay for the additional size of a machine capable of producing figures accurate to twenty significant digits.

In general, the larger the capacity of a computer system, the more it will cost. The capacity requirements must be accurately determined if the system is to be a practical one. And it should always be kept in mind that ***either*** in sufficient or excess capacity will be expensive.

The second factor—speed—is one which must be considered from two aspects: actual processing speed and input-output speed. Computer systems can be designed with high speed input-output units connected to high-speed processing units. For a firm which plans to enter large quantities of input, perform a large number of calculations on its data input, and which will require output immediately, a high-speed system would probably do the job most economically.

However, in an application where a great deal of calculation is to be done on a limited amount of input, with a limited amount of output, the high-speed input-output units would not be practical. To determine the required speed of the system—and hence the required speeds of the individual units of the system—the following should be accurately determined:

1. How much work must be performed in a given amount of time?
2. How quickly must output be received?
3. What is the ratio of calculations to communications?
4. Is the limiting factor in the operation calculating speed or communication speed?

When these questions have been answered, the speeds of the system's functional parts will be known and a complete system can be determined. With an efficient computer system, there should not be delays while one segment of the system catches up with another segment—such delays cost money.

The appropriate input-output devices can often determine the ultimate practicality of a computer system. For a great number of operations, punched cards are completely adequate. However, operations which require feeding large amounts of information into a master file of some type may find the added expense of magnetic tape well worth the cost. Certain specialized cases —where information is to be obtained directly from an operating process, or communicated over telephone lines from distant points—will require specially designed input-output devices which, although expensive, will be worth their cost through time saved.

The form of the output is particularly important since it must present the results of the computer's work to those who will make use of it. The permanency, accessibility and comprehensiveness of the output must be determined before final selection of the correct unit can be made.

Deciding upon the best system for a given operation—or set of operations—is just as important as the decision to computerize the operation in the first place. Whether the best system is to be simply a few small units or a complex combination of many high-speed, specialized units can only be determined on the basis of the job to be accomplished. When correctly selected, a computer system can be a valuable asset to a business organization. When incorrectly selected, a computer system can be expensive and time-wasting.

3. COMPUTER APPLICATIONS IN SCIENCE AND THE MILITARY

Introduction

As in business and industry, computers find wide applications in basic problem-solving operations for science and the military. And, the characteristics of computer problem-solving which influence the decision to use or not to use a computer in the business situation are equally applicable in the military or scientific situation. That is, computer problem-solving is normally most effective when it can be applied to a repetitive process to obtain solutions of a consistent type using relatively large amounts of data.

However, in the areas of science and the military computers find particularly valuable application in the solution of unique problems. The simulation technique discussed in Book III is frequently used, for instance, in the solution of single problems which would otherwise require many months—even years—of tedious calculation by mathematicians and scientists. Because of the specialized nature of the problems to be solved, computers are often used by scientists and

members of the military for projects which would be uneconomical in the business world. A problem involving a defense project, for example, cannot be evaluated in terms of cost alone; programming a unique military problem for computer solution may be well worth the expense if the results will affect the progress of the national defense effort.

A characteristic of computers which is particularly useful in scientific and military work is their ability to perform extremely rapid calculations with absolute accuracy. This means that a properly programmed computer can, in a matter of minutes, examine and analyze hundreds of alternative solutions to a given problem. For situations in research where the only reasonable approach is a "trial-and error" method, the computer's capabilities are making possible the solution of previously "impossible" problems.

Real-Time Problem Solving

In experiment 7 in this book we briefly examined "environmental sensing." An expansion of this function is often applied with great success in the handling of scientific and military problems. In general, the situations which we have examined have involved feeding data into a computer system—data in the form of mathematical and/or logical relationships, quantitative results from operating processes, etc. ***Real-time*** problem solving differs from these situations in that the data is supplied directly to the computer from its environmental and the computer is instructed to act upon this information immediately and provide recommendations for action or analyses of the environmental situation.

In real-time problem solving, the computer is given the capacity to directly sense its environment without requiring a human intermediary to supply the environmental information. For example, a computer can be used in conjunction with a radar tracking system so that as soon as the tracking system identifies an object in its path the computer will immediately give an alarm and begin plotting the location and movement of the object. A coordinated system such as this provides constant watch on the "environment" (the scope of the radar tracking equipment), yet does not require constant scrutiny by human operators.

Computer systems which are used for real-time problem solving in conjunction with specialized input equipment can greatly increase the value and utility of that equipment.

Computer Handling of Scientific Problems

Computers are rapidly becoming as familiar—and valuable—tools to the scientist as the vacuum pumps, power supplies, intricate glass tubing and other equipment vital to the experimental stage of a research project. It would be an impossible task to even list the scientific problems and projects which are presently being programmed for computer solution.

This does not mean, of course, that computers have all the answers to the problems. The computer is simply a valuable tool which permits scientists to examine more information more accurately, and with far greater speed than has ever before been possible.

Scientists use computers and computer techniques at every level of sophistication—from simple tabulation and correlation of data to extremely complex simulation programs for complicated analyses. For example, research teams have simulated the atom on computer systems in attempts to predict the locations and actions of atomic particles. In other scientific fields, vast amounts of data taken from thousands of observations—of the earth's structure, of the flow of ocean currents, of the paths of molecular particles—are being analyzed in attempts to find the basic concepts and relationships which govern their actions.

The field of engineering—which is becoming increasingly indistinguishable from what was previously known as "pure science"—has put computers to work on its particular types of problems with great success. Engineers find the simulation technique described in Book III especially useful: they are now able to test designs safely, and often without the cost of building a test model.

Technically speaking, the actual design of computer systems falls under electrical engineering, while the design and development of the various computer techniques falls between electrical engineering and pure mathematics. However, as is often the case in scientific and engineering work, there is a considerable amount of traffic among the various specific fields. A phys-

icist may find it necessary to develop a special technique to handle a problem; and this technique may well be one which can be effectively used by a chemist or a metallurgist. On the other hand, an electrical engineer may develop a technique which proves valuable to a biologist.

As a basic scientific instrument, computers have only begun to prove their value. As scientists and engineers use this new tool, they will make it even more valuable by expanding its capabilities and finding new uses for it.

Examples of Computer Handling of Military Problems

Computers are, of course, used by the military to handle operating problems, just as they are used in the business world. Basic research projects carried out under military contract require the use of computer systems as indicated above. However, there are many computer applications which are unique to the military—particularly applications which concern national defense. The experiments and discussion which follow illustrate only a few of the applications presently in use.

Signal Systems:

We have mentioned the use of a computer to signal a human operator when an operation has been completed, or to notify the operator of an unusual processing situation. In these cases, though, the signal is only an indication that something has occurred: the signal is important only in that it alerts the operators to the occurrence. In a military situation, however, the signal itself becomes important. Systems are often required which will, upon receipt of a specific input, produce a particular continuous signal.

The complexity of a computerized signal system is a function, basically, of the required inputs and outputs. The actual input unit may be as simple as a button to be pushed when the system is to be started—as, for example, an automatic SOS transmitter. On the other hand, the input unit can be a sophisticated radar system which produces a particular input when an object passes into its scope. In any case, the actual processing unit of a signal system is designed to:

1. Recognize a particular input as a "start transmission" command.
2. Search its memory for the appropriate signal.
3. Feed the signal into a designated transmitting device.

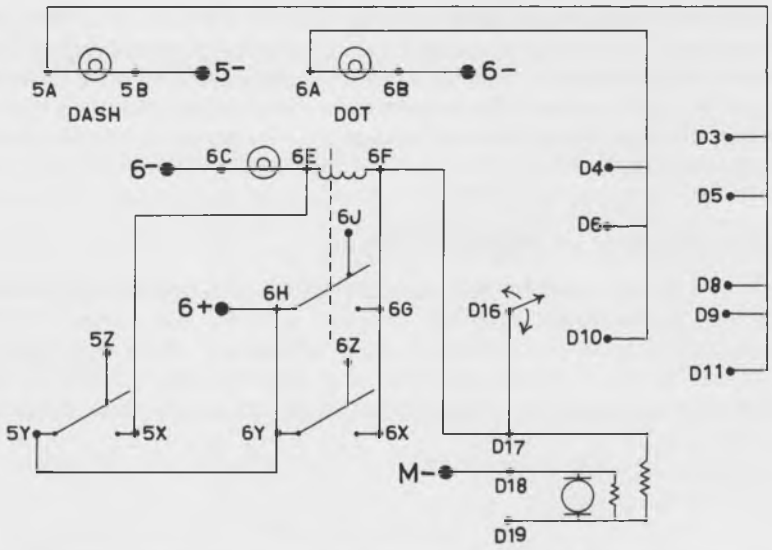
Once the signal system has begun its cycle, it will continue to transmit the signal until it reaches "end transmission" command—which can be pre-programmed into the system or can be given by a human operator. The output from an automatic signal system can vary in complexity from a simple blinking light to a coded message directed to distant points.

The three experiments which follow provide examples of the programming necessary to processing units used in automatic signal systems.

Experiment 16: Automatic Message Transmission

This circuit uses the turning of the rotary switch to generate a sequence of dots and dashes representing the amateur radio call "CQ" in Morse Code. Once turned on, this system will continue to "transmit" the CQ call until it is turned off.

The program and circuit diagram for the automatic CQ transmitter are:



CIRCUIT DIAGRAM—AUTOMATIC CQ TRANSMITTER

Program:

5A/D3	6B/6—	6H/6+	D6/D10
5B/5—	6C/6—	6H/6Y	D8/D9
5X/6E	6F/6G	D3/D5	D9/D11
5Y/6Y	6F/D17	D4/D6	D16/D17
6A/D4	6G/6X	D5/D8	D18/M—

The Morse Code representation for CQ is:

C: — · — ·
Q: — — · —

Light 5 on represents a dash.
Light 6 on represents a dot.

To use the program:

Set the rotary switch at 0 and turn power on. Push pushbutton 6. This provides the input command "start transmission." As the rotary switch turns, lights 5 and 6 will flash the CQ signal.

To stop transmission, push pushbutton 5.

Experiment 17: Automatic Name Transmission

This program presents a slightly more sophisticated version of the transmission technique used in the previous experiment. The name to be transmitted is "JOHN", and it will be sent in two parts: first "JO", then "HN".

To send the entire name requires a "selective sequencing" circuit. The computer must determine whether it is in the first or second path of the 2-path cycle, and produce the output appropriate to each path. That is, on its first revolution the rotary switch will instruct the computer to transmit "JO"; on its second revolution it will instruct the computer to transmit "HN."

As in the previous experiment, will be transmitting in Morse Code. The Morse Code representation for JOHN is:

J: · — — —
O: — — —
H: · · · ·
N: — ·

The program for this automatic transmitter is:

3C/4C	4F/5N	5L/D0	6H/6+
3F/4G	4H/5H	5X/6E	6H/6Y
3G/3K	4K/5F	5Y/6Y	6com/D2
3G/5com	4L/D15	5com/D11	D3/D4
3H/D12	4N/5E	6A/6com	D4/D5
3J/6com	5A/5com	6B/6—	D5/D12
3L/D13	5B/6B	6C/6—	D16/D17
4C/5C	5C/6C	6F/6G	D18/M—
4E/5K	5F/5G	6F/6X	
4F/4G	5H/5+	6G/D17	

Light 5 on represents a dash.
Light 6 on represents a dot.

To use the program:

Set the rotary switch at 0 and turn power on. Push pushbutton 6. The system will immediately begin transmitting "JOHN" in Morse Code. Notice that as the rotary switch turns the computer alternately transmits "JO" and "HN." The selective sequencing circuit causes the computer to select the correct segment of the "message" for each part of the transmission.

To stop transmission, push pushbutton 5.

Different code words can be transmitted using this basic circuit by re-programming the connections to the rotary switch.

Experiment 18: Automatic Transmission with Differential Spacing

The preceding experiments used different lights to indicate dots and dashes. It is possible, however, to produce "differential spacing" so that dots and dashes may be distinguished by the length of time a single light is on.

The program below illustrates a circuit with differential spacing by automatically transmitting the "SOS" distress signal in Morse Code:

3E/3H	5E/D19	6G/D4	D7/D8
3F/3—	5F/6K	6H/D16	D8/D11
3J/6A	6A/D1	6L/D18	D11/D12
4C/5C	6B/6—	D1/D2	D12/D13
4C/4F	6C/6—	D2/D3	D16/D17
4E/5E	6E/D9	D3/D6	D17/M+
5C/5F	6F/6G	D6/D7	D18/M—

The Morse Code representation of SOS is:

S: · · ·
O: — — —
S: · · ·

To use the prgoram:

Turn power on. The system will immediately begin transmitting "SOS". Light 6 and relay 3 will both indicate dots and dashes: they will remain on longer for a dash than for a dot. The computer will continue to transmit "SOS" until the power is turned off.

Computerized Code Systems:

A computer application of particular value to the military is the processing of messages transmitted in code. This application has three distinct aspects:

1. The routine translation of messages transmitted in "known" codes.
2. The more difficult translation of messages received in "unknown" codes.
3. The generation of new codes.

Routine translations are handled in much the same fashion as was done in the binary-decimal converters. A computer is simply programmed for direct conversion. If the system is handl-

ing more than one code, a technique known as "table-lookup" is often used. With "table-look-up", the computer holds the various codes in storage until directed to use a particular one. To translate, it "looks up" the appropriate substitution for each letter. The advantage of a computerized system for routine code translation is, of course, the speed with which the translations are carried out.

Translation of "unknown" codes—that is, "breaking" codes—can often be effectively handled by a computer system. If the "unknown" code was generated by a computer, another computer system will be ***required*** to break that code.

If the "unknown" code was originally man-made, the computer attempting to break the code will use a trial-and-error method based on the frequencies of appearance of the letters in the coded message. One type of code which a computer ***cannot*** break is one which does not use substitution of letters, but depends instead on phrases with special meaning only to the receiver for whom they were intended. Breaking codes of this type requires "inside information" which a computer cannot provide.

An example of a man-made code which a computer could easily break would be a substitution code based on a simple three-letter shift. That is, A is D, B is E, C is F, etc. An example of a man-made code which a computer could not break would be the following set of messages which would be known only to the transmitter and the receiver:

"Roses are red"	will mean	"All going as planned"
"Violets are blue"	will mean	"There is a change in plans"

The ability of computer systems to generate codes requires that computer systems be used in any attempts to break those codes. Codes generated by computer system are usually the result of the computer's ability to generate pseudo-random numbers. Breaking a computer-generated code is accomplished through sophisticated statistical techniques, using as a basis the fact that no computer can actually generate ***completely*** random numbers.

The two experiments which follow provide examples of simple coding and decoding programs for the international Morse Code.

Experiment 19: Encoder for Morse Code

This experiment will use MINIVAC to provide automatic letter-by-letter translation into Morse Code. The international Morse Code is:

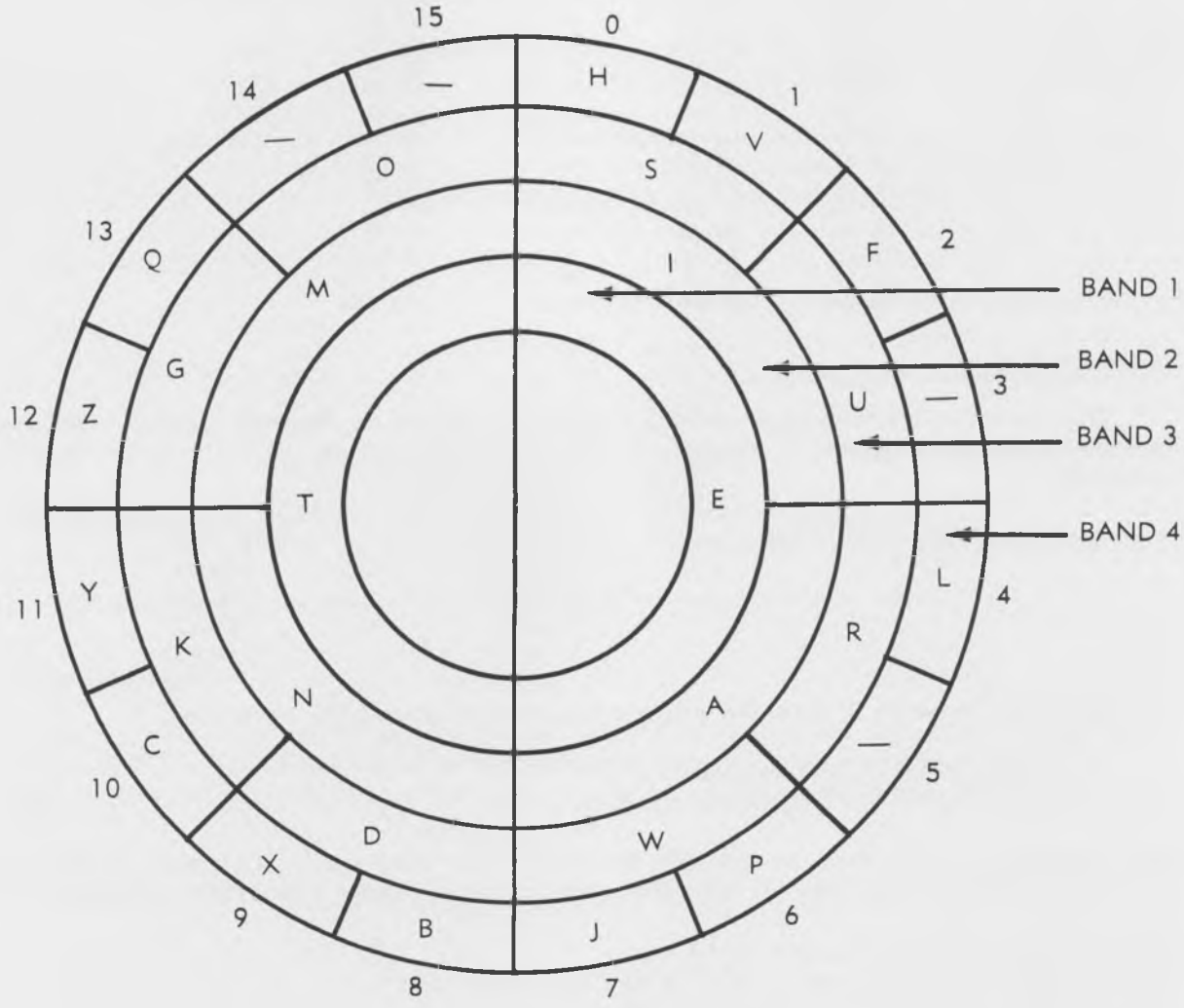
A	·—	N	—·
B	—···	O	———
C	—·—·	P	·——·
D	—··	Q	——·—
E	·	R	·—·
F	··—·	S	···
G	——·	T	—
H	····	U	··—
I	··	V	···—
J	·———	W	·——
K	—·—	X	—··—
L	·—··	Y	—·——
M	——	Z	——··

The program which follows will automatically display the Morse Code representation of any letter entered on the rotary switch. Output lights 3 through 6 will be the display lights.

The program for an Encoder is:

2C/3B	3G/D8	4com/5G	6F/D3
2C/3C	3H/3L	5A/5com	6G/6com
2F/D15	3L/4L	5B/6B	6H/6L
2G/4F	4A/4com	5C/6C	6H/6+
2H/2L	4B/5B	5F/D13	6com/D1
2H/3H	4C/5C	5H/5L	D1/D9
2K/5F	4F/D14	5H/6L	D2/D10
3A/3K	4G/4com	5K/6com	D3/D11
3B/4B	4H/4L	5com/6K	D4/D12
3C/4C	4H/5L	5com/D2	D5/D13
3E/D0	4K/5com	6A/6com	D6/D14
3F/3G	4com/D12	6C/6—	D7/D15
			D16/M+

To use this program most easily, the rotary switch dial should be fitted with a dial plate made by tracing the illustration below on a piece of paper.



MORSE CODE: ALPHABETIC DIAL PLATE

To use the program:

Set the rotary switch dial to 0 and turn power on. Turn the rotary switch clockwise until the pointer knob is pointing at the letter you wish to code. Read the Morse Code representation of the letter starting with output light 3. If the letter you wish to code is in band 1 of the dial plate, read only light 3. If the letter is in band 2, read lights 3 and 4. If the letter is in band 3, read lights 3, 4, 5. If the letter is in band 4, read lights 3, 4, 5 and 6.

A light on represents a dash.
A light off represents a dot.

Experiment 20: Decoder for Morse Code

This experiment uses the rotary switch dial plate described in the previous experiment. Letters entered in Morse Code on the slide switches will be indicated on the rotary switch.

The program for a Decoder is:

1A/1T	1W/4A	3G/D15	5J/D3
1B/1C	2B/3B	3J/D7	5K/D10
1B/2B	2C/3C	3K/D14	5N/D2
1C/2C	2F/4R	3N/D6	5S/6R
1F/2F	2G/4H	3S/4S	5V/6T
1G/3H	2H/5T	4C/5C	6C/6—
1H/5R	2J/6H	4F/5F	6G/D9
1J/5H	2K/4L	4G/D13	6J/D1
1K/3L	2L/5W	4J/D5	6K/D8
1L/5U	2N/6L	4K/D12	6N/D0
1N/5L	2V/2+	4N/D4	6S/6—
1R/3A	3B/4B	4S/6X	6X/D17
1S/2U	3C/4C	5C/6C	6Y/6+
1U/2A	3F/3R	5F/6F	D16/D19
1V/2W	3F/4F	5G/D11	D18/M—

To use the program:

With the dial plate from experiment 19 in place, turn power on. Set slide switches 1 and 2 to indicate in binary the number of Morse Code characters (dots and dashes) in the letter you wish to decode:

Number of Characters	Slide Switch 1	Slide Swith 2
1	0 (right)	1 (left)
2	1 (left)	0 (right)
3	1 (left)	1 (left)
4	0 (right)	0 (right)

Starting with slide switch 3, enter the dots and dashes according to the convention:

a slide switch RIGHT represents a DOT.
a slide switch LEFT represents a DASH

Push pushbutton 6: this gives the computer the instruction "read out." The rotary switch will turn to the correct spot on the dial. Read the alphabetic letter from the dial plate as follows.

Read the letter in Band 1 if output light 1 is on.
Read the letter in Band 2 if output light 2 is on.
Read the letter in Band 3 if output light 3 is on.
Read the letter in Band 4 if output light 4 is on.

Experiment 21: "Search and Track" Radar

In the previous section of this book the industrial uses of ***feedback*** were discussed. The importance of feedback in enabling a machine to deal directly with its environment makes it especially useful in military applications. An excellent example of the use of feedback in a military situation is provided in the basic operation of a "search and track" radar system used to detect and follow missiles and satellites.

A "search and track" radar system operates in two distinct modes: the "search" mode and the "track" mode. When operating in the search mode, the system constantly scans its environment, signalling the operator when it senses an object in the path of its scan. When an object is sensed, the system can be set in the track mode. Operating in the track mode, the system will find the object and continue to follow it if it moves.

While in the search mode, the radar system is a simple signalling system. Once in the track mode, the system utilizes feedback control to keep the desired object in "view." The actual operation of the system while in the track mode is as follows:

> The system scans the horizon until it picks up the signal which tells it there is an object in its path. It sweeps past the "object signal" until it has lost it and then reverses its direction, returning to the object signal. It again sweeps past, reverses and continues in this fashion until given other instructions or until the object signal is out of range.

The feedback in this case is information about the presence or absence of the object signal. The control action initiated by a computer system is to reverse the direction of scan immediately when the object signal is lost.

The program below illustrates a system which operates in two modes: a search mode and a track mode. The command to change mode must be given manually with this program; real search-and-track systems are often equipped to change modes automatically.

The program for a search-and-track system is:

1A/2E	2J/3H	3E/3J	5F/6G
1B/1—	2K/3N	3K/3+	5H/6H
2A/3E	2K/2—	3L/D18	5L/6J
2B/2—	2L/D17	4C/5C	6A/6V
2C/3C	2N/3K	4F/5K	6B/6—
2E/3G	3A/5E	4G/5E	6E/6—
2F/3F	3H/4F	4H/5H	6F/6U
2F/2H	3B/3—	5C/6E	6H/6+
2G/2+	3C/4C	5F/5G	6V/D16
			M+/D1, 2, etc.*

* Connection can be made between M+ and **any** point on the rotary switch dial.

To use the program:

Set slide switch 6 RIGHT and turn power on. The system is now operating in the search mode. Light 3 will flash on whenever the system detects an object signal (a connection on the rotary switch dial). You may move the object signal by changing the connection on the rotary switch dial.

Set slide switch 6 LEFT. The system is now operating in the track mode. Notice that the rotary switch oscillates about the object signal. If you now move the object signal, the system will continue to search until it finds the new location and will then resume tracking it.

Countdown Control:

One of the most highly publicized applications of specialized computer circuitry is the countdown used so dramatically on rocket test flights. The countdown is carried out not for effect, but because of the complexity of the equipment under test—as well as the complexity of the total test operation.

An actual countdown begins many months before "zero hour", and it is only at the final stages that specialized computer circuits are used for control. A computerized countdown control system is in essence a sequence control circuit. (see experiment 14) The inputs are the results of final checks; the output is either an "OK—step X" signal or an "ERROR" signal which immediately alerts the operators to the error and may, if further steps depend directly on the step in error, stop the countdown. The firing button is so programmed that it will not send the command "fire" until ***every*** step is correct.

Totally computerized check-out systems are built in to real quick-fire missiles. As soon as the "fire" command is given, the computer system begins to check out the individual parts of the missile, matching the check results against the pre-programmed requirements. If the computerized control system finds an error in the missile system, it automatically stops the firing procedure and signals an alert to the operators informing them of the location and nature of the error. The speed with which such control systems function makes it possible to reduce the countdown time on operational missiles to a few minutes.

The basis of any control system of this type is the simple AND circuit. Only after each individual switch is closed, through receipt of the correct input, will the "fire" command be executed.

4. COMPUTER APPLICATIONS IN THE SOCIAL AND POLITICAL SCIENCES

Vote Registering Machines

A computer-like device now familiar to many is the vote registering machine being used by some state legislatures. This device automatically registers the total vote on an issue without revealing the identity of the individual voter. As such, it is used in place of the "voice vote."

The basic circuit used in this vote registering machine is the Quantity Recognition circuit discussed in Book III. The inputs are supplied by the individual members of the legislative body, each of whom has a set of buttons with which he can indicate "yes," "no" or "abstain." Only the totals appear as output—usually displayed on a light board in the front of the room.

Although they "count votes," these vote registering machines are not functionally similar to the voting machines used in public elections. The vote registering machines use a basic computer circuit; the voting machines simply tabulate successive inputs.

Computers and Election Predictions

The national elections have spotlighted the capabilities of computers as predictors. Election prediction programs take advantage of the computer's ability to handle large amounts of data and to perform extremely rapid calculations. Many specialized techniques are used to extrapolate from early election return to the final outcome. However, the basic system is essentially a process of comparing early returns with historical data and performing various statistical operations to predict the final result.

When a computer is programmed for election prediction, it is fed information concerning the historical pattern of returns in certain precincts. These precincts are chosen on the basis of the relationship between returns in these precincts and returns in key national areas. As the returns from these precincts come in on election night, the computer compares the current ratio of returns to historical ratios. An extrapolation is then made on the basis of these comparisons.

The results of such a prediction scheme are stated as estimated conclusions with a specified degree of "confidence"—that is, a specified probability of occurrence of the predicted out-

come. When a computer produces a predicted result of this type, it is always very important to note the degree of confidence associated with that result. As the computer receives more and more information, its predicted results will have increasing degrees of confidence. The 100% degree of confidence is reached as a limit when the computer is given all the information—that is, when all returns are in.

The accuracy of the computer's predictions is a direct function of the program used to arrive at that prediction. The programs to predict the outcome of national elections are extremely involved mathematical statements designed to take into account as many factors as possible. Built into these programs is a vast amount of information about the social, economic and political status of samples of the key precincts and similar information about the country as a whole. Information about the candidates and their parties is also part of the total program, as well as information from previous elections concerning the vote-attracting characteristics of candidates and parties in various sections of the country.

Since a computer will accept only quantitative information, the hundreds of qualitative factors influencing a national election must be correctly stated in quantitative terms for inclusion in the total program. The analysis of these qualitative factors is in itself valuable work which is frequently made easier and/or speeded up by the use of computers. The knowledge gained from an attempt at one election prediction is not limited solely to knowledge of whether or not the program gave the correct result. Knowledge of indicative factors, changes in voting habits, shifts in attitudes towards national issues—all are valuable parts of the results of an election prediction program.

Language Translation

We have already examined a particular type of translation by a computer—the translation of messages into code. In terms of the circuitry involved in translation with a pre-determined code, this is merely a letter-by-letter substitution scheme. Translation between languages, however, requires that the computer be programmed to perform symbol-to-symbol substitution and, in addition, be capable of performing shifts in word order whenever necessary.

Symbols and Meaning

Before taking up the computer's problems in symbol-to-symbol substitution, let us briefly consider the nature of communication processes. In a normal day we encounter thousands of different symbols, each of which conveys a particular message. The red light of a traffic signal conveys the message "stop." The telephone bell indicates that someone is calling. Reading the word "water" brings a mental picture of a clear liquid made up of two hydrogen molecules for each oxygen molecule. The skull and cross-bones on a bottle warn that the contents are poisonous.

In each situation, we automatically translate a symbol into a message with particular meaning. We translate a color, a sound, a group of letters, or a picture into a meaningful idea. To a baby, though, these symbols have little meaning. The baby has not yet learned to associate a specific meaning or concept with these symbols; thus, the baby cannot "translate" the symbols. Teaching a baby to associate meanings with symbols is analogous to developing a computer program which will translate languages.

We must "teach" the computer to associate the correct meaning with a given symbol—and we must, just as in teaching the baby, teach each symbol in its proper environment. A baby will learn that a ringing bell does not ***always*** mean that someone is calling. A computer must be taught that certain symbols have different meanings depending upon the situation, or it will not be able to perform adequate translations.

Translations of Symbolic Combinations

Of the three successive stages of machine translation, we have already examined the first two:

1. Letter-by-letter translation as performed through the circuitry of the Morse Code decoder and encoder.

2. Translation of a pictorial symbol as performed through the circuitry of the Arabic Numeral Recognition program.

The third and most complex form of machine translation is that required for language translation: translation of one set of symbolic combinations into another. At this stage, the ***combination*** of symbols becomes important. The basic problem of interpretation between two symbolic sets is complicated by the "environment" of the symbols. That is, a given symbol may have different meanings depending upon the other symbols associated with it.

Perhaps the easiest way to see some of the problems inherent in machine language translation is to develop a few simple translation programs. We will do this using English and German.

Simple Translations between English and German

As a first step in developing a computer program for language translation, we will require a "dictionary." This will be ours:

ENGLISH	GERMAN
Pete	Piet
and	und
Eve	Eva
play	spielen
plays	spielt
with	mit
his	seinem
her	ihrem
their	ihrem
ball	Ball

Each English word or symbol has a German word or symbol corresponding to it and indicating the same idea or concept. In the simplest case, translation would involve merely a series of direct substitutions. We could program this elementary case by wiring pushbuttons to lights and letting the pushbuttons represent English words and the corresponding lights represent the corresponding German word. We could then translate from English to German by entering English input (pushing pushbuttons) and reading the German output from the lights.

However, we could not simply reverse the process to translate from German to English. Notice that the German "ihrem" translates as ***either*** "her" or "their." The simple program outlined above will not produce an adequate translation; we must add to the program a rule for selecting the appropriate English translation of "ihrem."

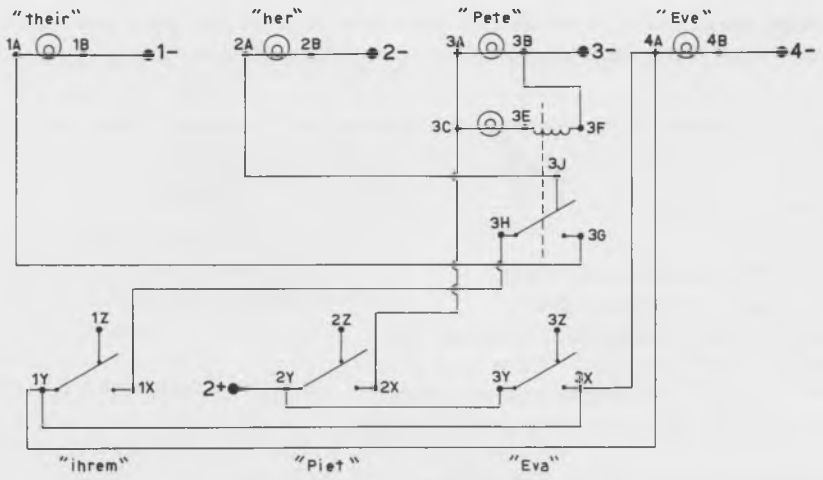
The basic rule which we must teach the computer is: a plural subject requires the use of "their" as an adjective; a singular feminine subject requires the use of "her" as an adjective. This can be stated logically as:

$P \times E = \text{their} \quad \text{and} \quad E \times \overline{P} = \text{her}$

If we now let the pushbuttons and lights represent words as follows, we will be able to program the rule for proper selection of an adjective, given the subject of the sentence.

Output light 1 on represents "their"
Output light 2 on represents "her"
Output light 3 on represents "Pete"
Output light 4 on represents "Eve"
Pushbutton 1 represents "ihrem"
Pushbutton 2 represents "Piet"
Pushbutton 3 represents "Eva"

The program and circuit diagram are:



CIRCUIT DIAGRAM—SELECTION RULE FOR TRANSLATION OF "IHREM"

Program:

1A/3G	2A/3J	3A/3C
1B/1—	2B/2—	3B/3—
1X/3H	2X/3C	3B/3F
1Y/3X	2Y/3Y	4B/4—
1Y/4A	2Y/2+	

To use this program:

Select a singular or plural subject by pushing pushbuttons 2 and/or 3. Enter the adjective "ihrem" by pushing pushbutton 1. The English Translation of the German input will appear in the output lights.

To further complicate matters, consider the expansion of our limited vocabulary to include the past perfect as well as the present tense of the verb. Direct translation of the past perfect tense is:

has played—hat gespielt
have played—haben gespielt

However, in German the word order changes when the past perfect tense is used. The auxiliary verb is placed as usual within the sentence, but the associated past participle must be placed at the end of the sentence. For instance:

Eve and Pete have played with their ball.
Eva und Piet haben mit ihrem Ball gespielt.

Thus, a translation program must be able to recognize the existence of the past perfect tense and modify the word order accordingly.

By limiting the number of words permissible in a sentence, we can develop a program to implement this requirement. Different slide switches will be used to communicate the English nouns and verbs. In actual translating programs, the grammatical identification of each word is supplied through an auxiliary coding scheme. Positioning will be accomplished by programming the German sequence so that "hat" and "haben" are required to appear after the subject and before the predicate if and when they occur.

The program which follows incorporates the word order requirements of the past perfect tense with the selection rule for the appropriate adjective. The slide switches will be used to provide information as to the players and whether they are playing now or played at some time in the past. The correct German expression of the English idea expressed will be read from the output lights and the relay indicator lights.

The translation program is:

1A/2V	1S/2S	2S/3S	3F/6A
1B/1E	1T/2V	2T/3L	3F/3R
1B/2B	2A/3G	2W/4A	3S/3+
1C/2U	2B/3B	3A/3N	4B/5B
1C/4C	2C/3K	3B/4B	5A/5+
1E/2E	2E/3E	3C/3J	5B/6B
1R/2U	2R/3H	3E/4E	6B/6—

Slide switch 1 RIGHT indicates "Pete"
Slide switch 1 LEFT indicates "Eve"
Slide switch 2 RIGHT indicates "Pete or Eve"
Slide switch 2 LEFT indicates "Pete and Eve"
Slide switch 3 RIGHT indicates present tense
Slide switch 3 LEFT indicates past perfect tense

The German sentence will be read in this sequence as follows:

Output light 1 on represents "Piet"
Relay light 1 on represents "Eva"
Output light 2 on represents "haben mit"
Relay light 2 on represents "hat mit"
Output light 3 on represents "spielt mit"
Relay light 3 on represents "spielen mit"
Output light 4 on represents "seinem"
Relay light 4 on represents "ihrem"
Output light 5 on represents "Ball"
Output light 6 on represents "gespielt"

To use the program:

Enter an English idea by moving slide switches 1, 2 and 3 appropriately. Read the German sentence from the lights as indicated above. The German sentence will be in the proper order and will be complete except for the conjunction which must be supplied if Pete and Eve are indicated as input: supply "und" for "and"; "oder" for "or."

From these simple examples you can see the problems encountered in a large-scale translation program. In addition to the grammatical problems which can be solved by converting the rules of grammar to logical statements for programming, a large-scale translation program must also be capable of recognizing the subtle shades of meaning conveyed by various words and phrases.

Present work on the development of language translation programs is by no means complete. Various research teams are at work on different languages, attempting to perfect present programs and developing new and better programs for the rapid translation of languages.

Job Selection

Computers' ability to perform high-speed comparisons has led to their application in personnel work, matching the qualifications of individuals with the requirements of specific jobs. In applications of this type, the results of computer comparisons are generally used to assist the personnel staffs of very large firms who find it necessary to process many job applications in order to select people to fill specific jobs. Seldom, if ever, are the computer's results taken as the final word on who is to be hired. The true value of the computer in this application is to point out those best qualified for a particular job—in essence, a narrowing-down procedure.

A computerized selection program requires two inputs for comparison:

1. The requirements of the specific job
2. The qualifications of the individual applicant

The job requirements are supplied by the personnel department. The applicant's qualifications are provided by asking the applicant to fill out a specially designed questionnaire. Such a questionnaire might look like this.

PERSONAL QUALIFICATION FORM

Check only one for each question

1. Which would you rather do:
   a. Add a column of numbers ——— a.
   b. Count the number of items in a row ——— b.
2. Which would you rather do:
   a. Write a short story ——— a.
   b. Paint a bookcase ——— b.
3. Which would you rather do:
   a. Be responsible for selling 100 items by yourself ——— a.
   b. Be responsible for selling 500 items with four other people ——— b.
4. Which would you rather do:
   a. Set type for a printing press by hand ——— a.
   b. Correct and rewrite articles for a newspaper ——— b.
5. Which would you rather do:
   a. Make final decisions and be responsible for the consequences ——— a.
   b. Make recommendations to the person above you ——— b.
6. Which would you rather do:
   a. Be responsible for hiring and firing close business associates ——— a.
   b. Be responsible for preparing the payroll for a company ——— b.

The job requirements information is generally in standardized form, with certain key factors specified. Some organizations use a standard job evaluation form, an example of which appears below:

JOB EVALUATION FORM

check only one for each question

1. Does the job require more:
   a. Accuracy and attention to detail ——— a.
   b. Speed with minimum accuracy ——— b.
2. Does the job require more:
   a. Creativity ——— a.
   b. Routine performance ——— b.
3. Is job performance more dependent upon:
   a. Manual dexterity ——— a.
   b. Working cooperatively with others ——— b.
4. Does the job require more:
   a. Direct supervisory activity ——— a.
   b. Mental effort ——— b.
5. Does the job require:
   a. Major responsibility for policy decisions ——— a.
   b. Minor responsibility for policy decisions ——— b.
6. Does the job require:
   a. Direct supervisory activity ——— a.
   b. Little or no supervisory activity ——— b.

Using the two forms above, we can write a comparison program designed to match individual qualifications with job requirements. The questionnaires above are, of course, much simpler than any in actual use.

The program makes a direct comparison between questions on the two forms. When an applicant's answer is consistent with a particular job's requirements, a light will come on. The best-qualified applicants for the job will be those for whom the most lights come on.

The program for ***section 1*** is:

```
1A/1S      1Y/1+      1T/1Z
1B/1—      1R/1X
```

Repeat this program for sections 2 through 6 on MINIVAC.

To use the program:

From a completed Job Evaluation form, enter the job requirements as follows:

Set the slide switch corresponding to the question number
LEFT if the answer was "a"
RIGHT if the answer was "b"

To check an applicant's qualifications, enter his replies to the Personal Qualifications form as follows:

Push the pushbutton corresponding to the question number for each question which was answer "a". If the question is answered "b" do not push the corresponding pushbutton. As the answer to each Personal Qualification question is communicated to the machine note whether the corresponding light goes on or off. (Note: some of the output lights will be on when only the job evaluation information has been indicated. Record these lights only if they do not go off when the applicant's replies are entered.) After the replies to all six questions have been tested, record the total number of lights that went or stayed on while the applicant's qualifications were tested.

Those persons most highly qualified for the specific job will be those with the greatest number of positive comparisons with the job requirements as indicated by the number of lights recorded.)

Mate Selection

The job selection technique has been used on one well-known television program and in some social science situations to provide a means of comparing the interests and orientations of a number of couples to determine their compatibilities. The assumption in this case is that compatibility will determine marital happiness.

The job selection program can be used to test the compatibility of a couple by changing the questionnaires. When used as a mate selection program, the replies of one individual are entered on the slide switches and the replies of the second individual are checked against the first individual's replies using the pushbuttons. The number of lights recorded as on will give an indication of compatibility. (Total compatibility is indicated by all lights recorded as on.)

Two questionnaires are presented below as examples of the characteristics which are compared in this type of program. The replies to questionnaire A are first compared, then the replies to questionnaire B. The total compatibility figure is reached by adding the results of A and B.

QUESTIONNAIRE A

	check only one for each question
1. Which is more important to you:	
a. A person's level of education	a. _____
b. A person's physical appearance	b. _____
2. Do you feel that you have:	
a. A liberal political orientation	a. _____
b. A conservative political orientation	b. _____
3. Would you prefer to have:	
a. A large family	a. _____
b. A small family	b. _____
4. Would you prefer to live in:	
a. An urban area	a. _____
b. A suburban or rural area	b. _____
5. Do you think a wife should be:	
a. Career-oriented	a. _____
b. Family-oriented	b. _____
6. Do you feel that you have:	
a. A fundamentalist religious orientation	a. _____
b. A liberal religious orientation	b. _____

Slide switch setting for questionnaire A:

Set the slide switch corresponding to each question

LEFT if the reply is "a"
RIGHT if the reply is "b"

Push a pushbutton for each "a" reply.

QUESTIONNAIRE B

Indicate whether or not you like:	YES	NO
1. Outdoor activities	a. _____	b. _____
2. Music and art	a. _____	b. _____
3. Sports	a. _____	b. _____
4. Political activity	a. _____	b. _____
5. The theater	a. _____	b. _____
6. Reading	a. _____	b. _____

Slide switch setting for questionnaire B:

Set the slide switch corresponding to each question

LEFT if the reply is "yes"
RIGHT if the reply is "no"

Push a pushbutton for each "yes" reply.

Behavioral Simulations

Some research is being done in the use of computers to simulate various behavioral phenomena normally associated with intelligent beings. This work raises interesting questions about the ability of a computer to exhibit certain kinds of intelligent behavior. For the purposes of this discussion, we will consider only two specific examples of machine simulation of intelligent behavior.

Associative Memory

The psychological process of conditioning can be simulated by MINIVAC through the program which follows. This program enables the computer to associate a stimulus and a response once the two elements have been presented together.

MINIVAC is programmed so that if pushbuttons 1, 2 or 3 are pushed separately lights 1, 2 or 3 respectively will come on. However, if two or three pushbuttons are pushed simultaneously, the computer will remember them in association. Once any combination of pushbuttons has been pushed, the computer will turn on all associated lights whenever any one of the associated pushbuttons is pushed. MINIVAC will remember the association until instructed to forget it.

This is analogous to the famous experiments carried out with dogs by the psychologist Pavlov. Like Pavlov's dog, the computer "learns" to associate stimulus and response. If two stimuli are presented simultaneously, the computer "learns" to associate the two responses originally associated with each of the two stimuli. Thereafter, the computer will execute both responses whenever either stimulus is presented. Unlike Pavlov's dog, the computer can be instructed to remember or forget various stimuli at will.

The program for an associative memory circuit is:

1A/1K	1Y/2Y	3A/3K	4H/6G
1B/1—	2A/2K	3B/3—	4L/5L
1C/2C	2B/2—	3C/4C	5C/6C
1F/4H	2C/3C	3F/6H	5F/5K
1F/1X	2F/5H	3F/3X	5G/6H
1G/6F	2F/2X	3G/5F	5L/6L
1H/3K	2G/4F	3L/4L	6C/6—
1K/2H	2K/3H	4C/5C	6F/6K
1L/2L	2L/3L	4F/4K	6L/6Z
1L/1Y	2Y/3Y	4G/5H	6Y/6+

To use the program:

Turn power on. Push pushbutton 1, 2 and 3 individually. Notice that each pushbutton turns on the single light associated with it. Now push two pushbuttons simultaneously— for example, pushbuttons 2 and 3. From now on, pushing ***either*** of the associated pushbuttons will turn on ***both*** of the associated lights.

Push all three pushbuttons simultaneously. Now pushing any one of the pushbuttons will turn on all three lights. To instruct the computer to forget the associations, push the "forget" button—pushbutton 6. The computer is once again ready to "learn" associations.

A simulated Maze Solver

The program which follows permits the computer to "learn" which of three possible paths is the correct path to a goal, and to "remember" the correct path until instructed to "forget" it. This is analogous to experiments in which an animal learns by trial and error the correct path to a goal through a maze.

The "maze" will consist of a start position and three possible "paths." The start position will be the "1" setting on the rotary switch dial. The possible "paths" will be:

Path A: from 1 to 3 on the rotary switch dial
Path B: from 1 to 5 on the rotary switch dial
Path C: from 1 to 7 on the rotary switch dial

The computer will be allowed to "explore" the possible paths. Then one path will be indicated as the correct path. The computer will find this correct path and will then always take that path until it is instructed to forget it.

The program for a simulated maze solver is:

1A/D8	3E/4E	4H/D7	6H/5Z
1B/1+	3E/6N	4L/D5	6Y/D1
2E/4Z	3F/4F	4Y/5G	6Z/D19
2F/2+	3F/3+	5B/5+	D2/D4
2G/6L	3H/D3	5E/6E	D4/D6
2H/2—	3J/5A	5F/6F	D6/D8
2K/4Y	3K/D2	5H/5—	D16/M—
2L/2—	3L/3—	5Y/5—	D17/M+
3A/4J	4A/4N	6F/6+	D18/M—
3B/3+	4B/4+	6G/D19	6E/free*

* This programming wire may be connected to D3, D5 or D7 to indicate the goal.

To use the program:

Turn power on. The computer will go directly to the start position. Push pushbutton 6. The computer is now "exploring" the possible paths. Release pushbutton 6 and indicate a "correct" path by connecting the wire from 6E to D3, D5 or D7. Push pushbutton 6 again. The computer selects the "Correct" path and stops.

Return the computer to its start position by pushing pushbutton 5. If you now push pushbutton 6, the computer will go directly to the "correct" path without exploring possible paths.

To set the computer to the "exploring" stage again, remove the connection from 6E to the path you selected and push the "forget" button—pushbutton 4.

5. COMPUTER APPLICATIONS IN SCIENCE FICTION FILMS

Science fiction fans seem always to be entranced by equipment which flashes lights and makes peculiar sounds. To satisfy these fans, the film industry has designed a great variety of "computers" for leading roles in science fiction films. Computers are well-adapted for such roles since it is possible to produce flashing lights ***and*** peculiar noises while operating a real computer.

However, the latest computers are made up of solid-state elements which operate silently; and output is more likely to be a reel of magnetic tape than a series of blinking lights, blips on a monitor screen and wierd choking and whirring noises.

Lest your friends—or you yourself—be disappointed at the relative quiet and tranquility of the preceding programs, the following two programs are presented. Neither program performs a useful function, but both provide the elements of science fiction machines.

The Flashing Lights Circuit

This program provides a maximum number of flashing lights. With this circuit programmed on MINIVAC, turn the power on. MINIVAC will perform as a blinking light machine.

The program for flashing lights is:

1A/D1	4A/D4	D1/D12	D8/D14
1B/1—	4B/4—	D2/D11	D10/D0
2A/D2	5A/D5	D3/D10	D11/D15
2B/2—	5B/5—	D4/D9	D16/M+
3A/D3	6A/D6	D5/D8	D17/M+
3B/3—	6B/6—	D6/D13	D18/M—

The "Super" Circuit

This program is presented for those who are not satisfied with the quiet operation of the previous program. Like the program above, this one does absolutely nothing constructive, although it appears to be diligently operating to solve the problems of the world. Program this circuit on MINIVAC and enjoy the ultimate in science fiction machines.

The program for the "Super" Circuit is:

1A/1E	2H/4N	4E/4J	6H/D16
1B/1—	2K/D4	4H/5K	6J/6K
1C/2C	2L/3N	5B/5—	6J/D17
1E/2G	3A/3E	5C/5com	6L/M—
1F/1H	3B/3—	5F/5—	6N/D18
1F/2F	3C/4C	5G/6E	D0/D2
1G/1+	3E/4G	5H/4—	D2/D6
1J/2H	3F/3H	5L/5+	D3/D15
1K/D1	3F/4F	5com/D13	D3/D5
1L/3K	3G/3+	6A/D0	D5/D9
1N/D8	3H/4L	6B/6—	D6/D7
2A/D14	3J/4H	6C/6G	D7/D11
2B/2—	3L/6com	6E/6com	D9/D10
2C/3C	4A/4E	6F/6—	D11/D12
2E/5A	4B/4—	6G/6N	D14/D15
2E/2J	4C/3—	6H/6+	

APPENDIX

PROGRAMMING LANGUAGES

In the discussions of programming in this series we have examined instructions which are communicated to a computer as part of its program. It has been emphasized that a computer must be given complete instructions for each step which it is to perform in solving a problem.

In Book II, three types of programs commonly used by electronic data processing machines were discussed: Wired programs, Coded programs, and Stored programs.

We have not yet commented on the various coding systems developed to permit instructions to be communicated to a computer in other than the 1's and 0's of a binary code. It is important to remember that once communicated to the computer, ***all*** instructions will eventually be reduced to 1's and 0's which will be interpreted by the computer as on-off switches. The computer cannot directly "understand" the meaning of the word "add." However, it can interpret the letters "ADD" as 3 binary coded decimal or "BCD" characters according to the BCD code presented in Book II. A computer programmed to understand BCD can translate the three letters of the word "ADD" into a combination of 0's and 1's.

Letters	BCD Equivalents
ADD	010 001 010 100 010 100

Once the word "ADD" has been converted into binary information using the BCD code, the BCD equivalent of the word "ADD" may be stored in a storage register of the computer. An interpretive program may then translate the BCD into a binary operating code which the computer interprets as an instruction directing it to utilize certain pre-programmed circuits in a specified way.

There are as many different interpretive routines as there are different computers. Most large computers have interpretive programs written for them so that they are able to receive instructions in the form of letters such as the "add" combination above and translate these letters into binary bits using BCD. Through the interpretive program, the computer can then translate the coded binary bits into a single instruction code which the computer interprets as an instruction

directing it to use certain logical or arithmetic circuits which have been pre-programmed into the computer. Once the interpretive program has reduced the letters into a simple instruction code, the large computer is operating with a basic input signal such that a 0 in the instruction code is comparable to a pushbutton in the up position on MINIVAC and a 1 in the instruction code is comparable to a pushbutton in the down position on MINIVAC.

The interpretive program converts information communicated to it in letters which the human operator can understand into simple 0 and 1 signals which the machine can interpret as basic instructions. These basic instructions determine which pre-programmed circuits will be used, to effect the action which the human operator desires.

Some examples from the basic language routine used in conjunction with the IBM 7090 computer will be used to demonstrate the functions of an interpretive program. The combinations of letters below are a portion of a program prepared for communication to the 7090 using the alphabetically coded language which is translated by the interpretive program prepared for operation on that computer.

```
    CAL   A
    SUB   B
    STO   C
A   PTH
B   PTW
C   PZE
```

A "SCAT" CODED 7090 PROGRAM

The letters on each line of the program are punched on a single input card. As the cards are read into the computer, each of the letters is translated into a BCD representation which is stored in the computer. A decoding program then interprets the BCD code and establishes the meaning for each of the instructions explained below.

CAL A This instruction is interpreted by the computer as: define one register in storage as register "A". Clear the processing register (set all of the bits of the processing register to zero) and add the contents of storage register "A" to the processing register. If we consider the pushbuttons of MINIVAC to be register "A", this is equivalent to telling the 7090 to select a circuit comparable to one which will permit the contents of the pushbuttons to be entered in the processing relays.

SUB B This instruction is interpreted by the computer as: define a single register of storage as register "B". Then subtract the contents of storage register "B" from the contents of the processing register. If we consider the pushbuttons of MINIVAC to be storage register "B", this instruction is equivalent to telling the large computer to perform that operation wherein the contents of the pushbuttons are subtracted from the contents previously stored in the processing relays.

STO C This instruction is interpreted by the 7090 as: define a single storage register as register "C". Then transfer the contents of the processing register to storage register "C". If the binary output lights of MINIVAC are defined as storage register "C", this is equivalent to directing MINIVAC to display the contents of the processing relays on the binary output lights.

The last three instructions in the program indicated above are used to direct the 7090 to store specific numerical values in the registers which it has defined as A, B, and C, respectively.

PTH Indicates that binary 3 is to be stored in the register defined as storage register "A".
PTW indicates that a binary 2 is to be stored in the register defined as storage register "B".
PZE directs the 7090 to make the initial contents of storage register C equivalent to a positive 0.

Many instructions in addition to those indicated in the example above may be used to communicate programming ideas to the 7090. In each case the instruction is first converted to the BCD code so that it may be stored in binary form in the computer. The BCD code is then trans-

lated by the interpretive routine into a simple binary instruction which can be performed by the computer.

The binary instruction signals the large computer just as pushing a pushbutton on MINIVAC signals it to select certain pre-programmed circuitry and perform the operations which the program is prepared to execute.

Coding languages may be used to communicate logical as well as arithmetic instructions to a computer. The first sequence of instructions below is interpreted by the interpretive program of the 7090—and a series of binary instructions are generated. These instructions cause the 7090 to perform an "AND" operation with the contents of storage register "C" set equal to the contents of storage register "B" ***AND*** the contents of storage register "A". The second program causes the computer to store the contents of storage register "B" combined according to the rules of the ***"OR"*** function with the contents of storage register "A" in storage register "C".

Program 1:

```
CAL  A
ANA  B
STO  C
```

Program 2:

```
CAL  A
ORA  B
STO  C
```

TWO SCAT-CODED LOGICAL PROGRAMS

The interpretive programs discussed so far permit instructions to be given to the computer expressed as a combination of letters, rather than as binary words. More complex interpretive programs have been developed to permit the computer to perform an additional step in translating information communicated to it by the human programmer. These programs translate information into binary control words which the computer can interpret as control instructions directing it to select specific pre-programmed operating circuits. An example of one such "compiler program" is the "Fortran" System prepared for use with the IBM 7090. The statement below is a Fortran coded representation of the relationships expressed in the SCAT coded 7090 program discussed at the beginning of this section.

$$C = A - B$$

The "Fortran" statement which is punched on a single input card is a mathematical statement of the relationship which the computer is to represent by the steps in its program. The Fortran compiler first reads the alphabetic and special characters punched on the card and interprets them using BCD into a series of 0's and 1's which can be stored in the registers of the computer. The compiler program then interprets each of the special characters (in the example above the "—" sign and the "=" sign) as directions to establish a series of instructions equivalent to the SCAT coded 7090 program instructions.

Once a series of instructions equivalent to the SCAT coded instructions are established by the compiler, an interpretive routine directly equivalent to that used to interpret the SCAT coded program is employed to establish the equivalent instructions which are then executed by the computer.

The examples above are simple ones. The capabilities of the various programming languages developed for use with large scale machines are quite comprehensive. Although these languages solve some problems by making it unnecessary to communicate to the machine directly in binary language which the machine can understand, they create others by making it necessary to comply with a number of conventions in a specific manner. These conventions are necessary in order to insure that the interpretive routine will interpert the letters given to it on the punch cards in ***exactly*** the manner in which the programmer wishes to have them interpreted.

The important point to remember in considering the variety of machine coding systems which exist is that they are not read or executed by the computer in the form in which they are communicated to it. Compiling and interpretive routines are used to ***translate*** the information presented to the computer into the basic 0 and 1 codes which the computer is able to use as a basis for its operation.

The example below provides a summary of the basic steps performed by a compiler and interpretive routine in translating a single card containing a Fortran statement into instructions which the 7090 is able to perform.

The FORTRAN Expression

$$C = (A + B) \quad * \quad D + F/(E - A)$$

is interpreted by the compiler program as equivalent to the following series of SCAT instructions.

```
CLA  E          CLA  A
FSB  A          FAD  B
STO  Temp       XCA
CLA  F          FMP  D
FDP  Temp       FAD  Temp
STO  Temp       STO  C
```

Once this series has been generated, each instruction is assigned to a storage register in which the binary code representing that instruction will be stored. The binary code is ***not*** the BCD coding of the letters of the instruction but a code ***representing*** the BCD instruction. For example, CLA E which is BCD coded by the machine as 010 011 100 011 011 001 010 101 is coded by the interpretive routine as 000 000 111—101 000 000—000 000 001 where storage register 001 has been assigned to Variable "E" and the instruction "CLA E" has been assigned storage register 111. The code 101 000 000 is the operation code derived from "CLA" which indicates to the machine "select the pre-programmed circuitry designed to place the contents of the indicated data register in the processing register (Accumulator)."

The Machine language coding of the entire SCAT program appears below:

Instruction Location	Operation Code	Data Location
000 000 111	101 000 000	000 000 001
000 001 000	011 000 010	000 000 010
000 001 001	110 000 001	000 010 100
000 001 010	101 000 000	000 000 011
000 001 011	010 100 001	000 010 100
000 001 100	110 000 000	000 010 100
000 001 101	101 000 000	000 000 010
000 001 110	011 000 000	000 000 100
000 001 111	001 011 001	000 000 000
000 010 000	010 110 000	000 000 101
000 010 001	011 000 000	000 010 100
000 010 010	100 000 001	000 000 110

This machine language representation of the program is the only form of the program which the computer can actually execute. Until this form is derived the computer— in running the compiler and interpretive programs used to reduce the FORTRAN statement given above to the Machine language—is simply translating BCD data and coding this data in accordance with a series of procedures which it has been previously programmed to follow. Once it has produced the machine language program above it can be given this new program and directed to execute the indicated instructions.

BOOK VI

MINIVAC Games

PREFACE

This book contains examples of several types of games which can be played using MINIVAC 601. Once you have become acquainted with these games you will discover that many different games can be developed using the gaming capabilities of the MINIVAC. Realizing that hundreds of games have been designed using the 52 cards of a deck of playing cards, you can imagine the number of games which can be developed using the hundreds of contact points on the MINIVAC 601. Each of the games appearing in this book demonstrates a principle of computer gaming which may be expanded to produce additional games.

Most of the games with and against MINIVAC depend either on the players' skill (as in the "reaction time tester" or "philosophic tug of war") or MINIVAC's "skill" (as in the "Match Game" or "Tic-Tac-Toe"). Some games depend on chance and make use of MINIVAC as a source of random numbers (as in the "Fortune Teller" program).

1. MINIVAC AS AN OPPONENT

The Secret Code

The object of the game is to find the code; that is, the correct order in which the pushbuttons must be pushed to make the "Code Solved" light come on. Using the slide switches, you can generate 63 different codes; by simple re-programming you can generate 720 different codes.

Basic to this game as well as to the two games which follow, is a ***sequence recognition circuit.*** Once this basic circuit is wired, different games can be created using the pushbuttons, game matrix, rotary switch and slide switches. For the secret code, the slide switches and pushbuttons are used.

The program for the basic sequence recognition circuit is:

1A/1G	2F/3K	3com/4L	5F/6K	6F/6com
1B/1—	2H/3G	4C/5C	5H/6G	6H/6+
1C/2C	2N/3N	4F/4G	5N/6N	
1F/1G	2com/3L	4F/5K	5com/6L	
1F/2K	3C/4C	4H/5G	6A/6J	
1H/2G	3F/3G	4N/5N	6B/6—	
1com/2L	3F/4K	4com/5L	6C/6—	
2C/3C	3H/4G	5C/6C	6E/6N	
2F/2G	3N/4N	5F/5G	6F/6G	

For the Secret Code, ***add*** the following connections:

1R/6V	2V/3R	4T/4U	6R/6W
1R/1W	2Y/3Y	4V/5R	6T/6U
1S/1X	3R/3W	4Y/5Y	6Y/6+
1T/1U	3S/3X	5R/5W	2T/1com*
1T/2S	3T/3U	5S/5X	6W/2com*
1V/2X	3T/4S	5T/5U	4T/3com*
1Y/2Y	3V/4X	5T/6S	2W/4com*
2R/2W	3Y/4Y	5V/6X	6T/5com*
2T/2U	4R/4W	5Y/6Y	4W/6com*

* These connections may be interchanged for new codes. Be sure though, that there are connections to ***each*** of the six common terminals.

To use the Secret Code program:

Wire the basic circuit onto the MINIVAC; then ***add*** the connections directly above. Program a code by setting the slide switches in various positions. For example, with the above program and all six slide switches to the right, the code is solved by pushing the pushbuttons in sequence from 6 through 1. When the code has been correctly solved, light 1 comes on indicating "code Solved." If light 6 comes on, an error has been made and MINIVAC will automatically re-set for the next attempt.

Once you have set the code by moving the slide switches, you can challenge your friends to "break the code." They can then try to find the correct sequence of pushing the six pushbuttons to break the code. When your code is broken, you can change it by moving the slide switches.

If you want to create more new codes, you can interchange the starred connections to the common terminals. For example, by changing:

2T/1com to 2T/2com
and
6W/2com to 6W/1com

you will be able to generate 63 different codes.

The Combination Lock

Using the basic sequence recognition circuit of the Secret Code, in combination with the rotary switch dial, you can design an electronic combination lock which works just like those used on vaults, padlocks, etc.

Light 1 comes on to indicate that the code has been solved; light 6 comes on when an error is made, and the machine then automatically re-sets for the next attempt.

First, program the basic sequence recognition circuit:

1A/1G	2H/3G	4F/4G	5N/6N
1B/1—	2N/3N	4F/5K	5com/6L
1C/2C	2com/3L	4H/5G	5A/6J
1F/1G	3C/4C	4N/5N	6B/6—
1F/2K	3F/3G	4com/5L	6C/6—
1H/2G	3F/4K	5C/6C	6E/6N
1com/2L	3H/4G	5F/5G	6F/6G
2C/3C	3N/4N	5F/6K	6F/6com
2F/2G	3com/4L	5H/6G	6H/6+
2F/3K	4C/5C		

(This is the same basic circuit as was used in the previous game)
Now ***add*** the connection D16/M+

Now, design a combination and program it as follows:

Combination	Connections
left to 13	D13/6com
right to 2	D2/5com
left to 10	D11/4com
	D10/3com
right to 6	D6/2com
left to 9	D9/1com
	D8/6E*

* This last connection is optional; it will cause the machine to indicate an error if 8 is passed at ***any*** point.

When you are designing a combination, you must not include any connection which will touch off a connection further along. Also, you must make connections to each of the six common terminals in order.

Once you have designed a combination and programmed it onto MINIVAC, you can challenge your friends to attempt to solve the combination. Light 1 will come on when the combination is successfully solved; light 6 will come on to indicate an error, and the player must start over again.

You may wish to add dummy connections to the rotary switch dial so that the pertinent connections will not be obvious.

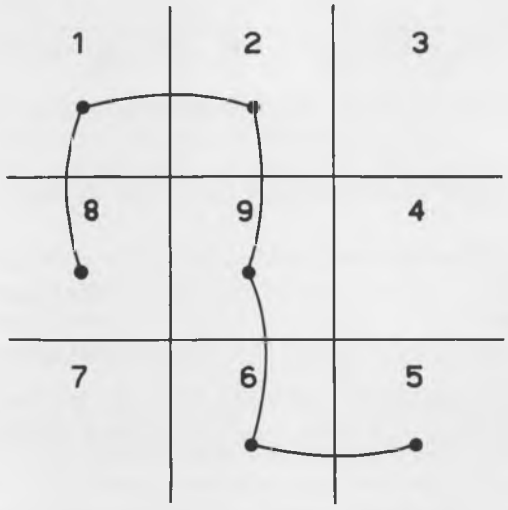
The Electronic Maze

Again using the basic sequence recognition circuit, this time in combination with the game matrix, we can design another game. The object of this game is to find the correct path through the game matrix. Light 1 indicates success; light 6 indicates error and re-set.

First, program the basic sequence recognition circuit:

1A/1G	2H/3G	4F/4G	5N/6N
1B/1—	2N/3N	4F/5K	5com/6L
1C/2C	2com/3L	4H/5G	6A/6J
1F/1G	3C/4C	4N/5N	6B/6—
1F/2K	3F/3G	4com/5L	6C/6—
1H/2G	3F/4K	5C/6C	6E/6N
1com/2L	3H/4G	5F/5G	6F/6G
2C/3C	3N/4N	5F/6K	6F/6com
2F/2G	3com/4L	5H/6G	6H/6+
2F/3K	4C/5C		

(This is the same basic circuit as was used in the two previous games.)
Now ***add*** a programming wire at M+, leaving one end free.
This is the "play wire."
Design a path through the game matrix. For example: from 8 to 1 to 2 to 9 to 6 to 5



This is programmed as follows:

from 8	M8t/6com
to 1	M1t/5com
to 2	M2t/4com
to 9	M9t/3com
to 6	M6t/2com
to 5	M5t/1com

The player now takes the free end of the "play wire" which is connected to M+ and touches it to the ***top*** terminals of the game matrix squares. (The wire need not be plugged into the terminal; touching the terminal will make the contact.) You may tell the player his START and GOAL points; if you wish to make the game more difficult, however, you need not give him this information.

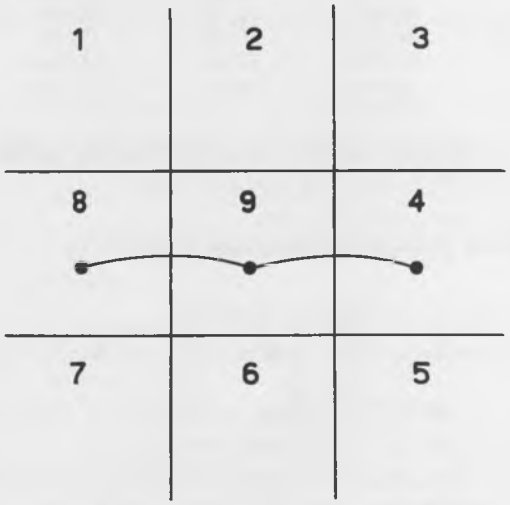
Since connections are made only to those squares which are a part of the path, you may wish to add dummy connections to those not involved in the actual maze. For example, in the above program, you could add:

M3t/6R
M4t/6S
M7t/6T

This will not affect the maze, but conceals the fact that squares 3, 4 and 7 are not part of the path.

If you wish to program a path which has less than six steps, connect this path to the lower common terminals and manually push the remaining relays to the left before beginning play. For example, consider the following path:

from 8 to 9 to 4:



Make the connections:

from 8	M8t/3com
to 9	M9t/2com
to 4	M4t/1com

Before beginning the play, manually push relays 4, 5 and 6 to the left. Whenever an error is made, relays 4, 5 and 6 must be pushed to the left before play is resumed.

The Match Game

In the original game two players alternate in taking matches from a pile. At each move a player can take ***either*** one or two matches. The object is to take the last match. This program permits the MINIVAC to play against a human opponent. The number of matches is represented by the rotary switch dial.

The game is started with the rotary switch dial at 15. The player takes the first move and indicates his move (1 or 2 matches taken) by pushing the appropriate pushbutton. MINIVAC 601 will always win.

The program for the Match Game is:

3A/3E	4Y/5Y	6Y/6+
3B/3—	5C/6C	D1/D4
3E/3G	5E/D13	D2/D5
3F/3—	5F/5X	D3/D6
3G/D0	5F/5G	D4/D7
3H/3Z	5H/6H	D5/D8
3Y/4Y	5K/6K	D6/D9
4C/5C	5L/6L	D7/D10
4E/D14	5Y/6Y	D8/D11
4F/4X	6C/6—	D9/D12
4F/4G	6E/D15	D10/D13
4H/4L	6F/6X	D11/D14
4H/5H	6F/6G	D12/D15
4K/5K	6H/6+	D16/M+
4L/5L	6K/D18	D17/M—

Pushbutton 3 is the re-set button.
Pushbutton 4 is used for "player removes 1 match."
Pushbutton 5 is used for "player removes 2 matches."
Pushbutton 6 is used for "machine moves."

Light 3 is the "win" indicator. The last player to move before light 3 comes on wins the game.

To play the Match Game:

Push the re-set button (pushbutton 3). The rotary switch dial will turn to 15, indicating that there are 15 matches in the pile.

Player makes the first move and indicates:
removing 1 match by pushing pushbutton 4
or
removing 2 matches by pushing pushbutton 5

After the player indicates his move, the rotary switch dial will indicate the number of matches remaining.

MINIVAC moves next: player must push pushbutton 6; the rotary switch will turn to the number of matches remaining after the machine's move.

Play continues as above, with the player and MINIVAC removing matches in turn.

Light 3 will come on after the last move has been made. The last player to move before light 3 comes on wins the game.

To start another game, push the re-set button (pushbutton 3).

Tic-Tac-Toe

MINIVAC can be programmed to play tic-tac-toe against a human opponent. A warning though: with this program, MINIVAC can not lose. The human opponent may tie the game, but he can never win. This is because of the decision rules which are the basis of the program.

The MINIVAC is so programmed that the machine will move 5 squares to the right of its own last move if and only if the human opponent has blocked that last move by moving 4 squares to the right of the machine's last move. If the human player did not move 4 squares to the right of the machine's last move, MINIVAC will move into that square and indicate a win. If the human player consistently follows the "move 4 to the right" rule, every game will end in a tie.

This program requires that MINIVAC make the first move; the machine's first move will always be to the center of the game matrix. A program which would allow the human opponent to move first would require more storage and processing capacity than is available on MINIVAC 601. Such a program would, of course, be much more complex than the program which permits the machine to move first.

The program for Tic-Tac-Toe is:

1A/1N	2L/D18	4F/5F	5L/M4t
1B/1—	2N/D19	4G/D5	5G/6N
1C/2C	2X/3X	4H/M1b	5Y/M10
1F/M11	2Y/3Y	4H/M5t	6+/6Y
1G/2A	3C/4C	4J/4K	6—/6C
1H/5X	3F/4F	4K/D6	6G/D1
1J/2F	3G/D7	4L/M2b	6H/M1t
1L/1Z	3G/4N	4L/M6t	6H/M5b
1N/3F	3H/M3b	4G/5N	6J/6K
1X/2H	3H/M7t	4Y/5Y	6K/D2
1X/2X	3J/3K	5C/6C	6L/M2t
1Y/2Y	3K/D8	5F/6F	6L/M6b
1Y/2G	3L/M4b	5G/D3	6X/M10
2B/2—	3L/M8t	5H/M7b	6com/D9
2C/3C	3N/6G	5H/M3t	M9b/M11
2E/D16	3X/4X	5J/5K	M—/D17
2F/2H	3Y/4Y	5K/D4	M10/free*
2G/2K	4C/5C	5L/M8b	M11/free**

* this is the machine's "play" wire
** this is the player's "play" wire

To play:

(Note: it will probably be most convenient to keep track of the game on a piece of paper.)
Turn power ON.
Place the free end of the machine's "play" wire in 6 com.

Push pushbutton 6 and hold pushbutton 6 down while you push pushbutton 1. The pointer knob will turn to indicate the machine's first move.
Release both pushbuttons 1 and 6.

All succeeding moves are made as follows:

Place the free end of the machine's "play" wire in the ***bottom*** contact of the matrix terminal corresponding to the machine's last move (as indicated by the pointer knob).
Player now selects his move and indicates this move by placing the free end of the player's "play" wire in the ***top*** contact of the matrix terminal corresponding to his move.
Push pushbutton 6 and hold pushbutton 6 down while you push the pushbutton corresponding to the move (pushbutton 2 the second time, pushbutton 3 the third time, etc.)
The play continues as above until either light 1 comes on—indicating that MINIVAC wins; or light 2 comes on—indicating a tie.
To begin another game, be sure that the player's "play" wire and the machine's "play" wire are both free.

2. MINIVAC AS A REFEREE

The Philosophic Tug of War

This game is played with two persons; MINIVAC acts as the game board and referee. The object of the game is to make the rotary switch dial stop at the WIN position for the player. If the dial stops at 9, player 1 wins; if the dial stops at 7, player 2 wins.

Player 1 uses pushbutton 1; player 2 uses pushbutton 6. The rotary switch knob turns:
counterclockwise (towards 9) if pushbuttons 1 and 6 are in the ***same*** position (both up or both down.)
clockwise (towards 7) if pushbuttons 1 and 6 are in ***different*** positions.

In other words, player 1 tries to match his pushbutton position to that of player 2's. At the same time, player 2 tries to keep his pushbutton position opposite to that of player 1's.

The program for the Philosophic Tug of War is:

1X/6X	4Z/D9	6G/6N	6N/D18
1Y/1+	6E/6Y	6H/6+	D7/D8
1Z/6Z	6F/6—	6J/6K	D8/D9
4Y/D18	6F/6L	6J/D17	D16/D19

To play:

Player 1 uses pushbutton 1
Player 2 uses pushbutton 6

To start:

Push pushbutton 4; this is the re-set button. Release pushbutton 4 as the rotary switch knob starts to turn. As soon as the knob starts to turn, each player tries to control its direction using his pushbutton. Player 1 will win if the knob reaches 9; player 2 will win if the knob reaches 7.

The game continues with player 1 trying to match his pushbutton position to that of player 2, while player 2 is trying to keep his pushbutton position different from that of player 1.

The game ends when the pointer knob stops—either at 9 (player 1 wins) or at 7 (player 2 wins).

To start another game, push the re-set button (pushbutton 4).

The Mind Reading Trick

The "Mind Reading Program" of Book III is repeated here with a variation you may wish to try. (See Book III for the development of the program.)

The program for the Mind Reading Trick is:

5V/4S	6V/5V	4W/5S	D0/6N
4R/4A	6U/6C	5R/6H	D1/6K
4B/4—	5U/5C	5T/6L	D2/6J
5C/5A	5F/6F	6R/5H	D3/6G
5B/5—	6F/6—	6T/5L	D4/5N
6A/6C	M—/D18	M+/6Y	D5/5J
6B/6—	D18/4V	6X/D17	D6/5K
6X/6V	4U/6S	D16/D19	D7/5G

To use the program:

Ask a friend to think of a number between 0 and 7. Ask your friend to answer "yes" or "no" to the following questions about the number:

A. Is the number greater than 3?
B. When the number is divided by 4, is the ***remainder*** greater than 1?
C. Is the number odd?

Indicate the answers to the questions as follows:
If the answer to question A is YES, move slide switch 4 to the LEFT.
If the answer to question A is NO, move slide switch 4 to the RIGHT.
If the answer to question B is YES, move slide switch 5 to the LEFT.
If the answer to question B is NO, move slide switch 5 to the RIGHT.
If the answer to question C is YES, move slide switch 6 to the LEFT.
If the answer to question C is NO, move slide switch 6 to the RIGHT.

After the replies to the questions have been indicated, push pushbutton 6. The pointer knob of the rotary switch will turn to the number your friend had in mind.

A variation:

The same program can be used substituting names for numbers as follows:

0 = Karen
1 = David
2 = Catherine
3 = Robert
4 = Peggy
5 = Andy
6 = Dorothy
7 = Hartley

Ask a friend to mentally select one of the above names, and then ask him to answer the following questions:

A. Does the name end in "Y"?
B. Does the name have more than 5 letters?
C. Is the name a boy's name?

Indicate the replies to the questions exactly as before. Push pushbutton 6. The pointer knob will turn to the number corresponding to the name which your friend selected.

The Fortune Teller

With this program, MINIVAC will answer questions—after "thinking" about them. The answer to the question will appear on the lights as follows:

light 1—Absolutely No
light 2—Doubtful
light 3—Perhaps
light 4—Definitely Yes

Lights 5 and 6 are MINIVAC's "contemplation" lights.

The program for the Fortune Teller is:

1A/3G	3A/3J	5C/6C
1B/1—	3B/3—	5E/6K
1C/3E	3C/4C	5F/6N
1E/2E	3F/5G	5F/5G
2A/3K	3H/4K	5H/5+
2B/2—	3L/4N	5K/6F
2C/4E	3N/4A	5N/6E
2E/3C	4B/4—	6A/6E
2F/2X	4C/5C	6B/6—
2G/6L	4F/6G	6C/6—
2H/2L	4L/4X	6F/6G
2K/5L	4Y/4+	6H/6+
2L/2Y	5A/5E	
2Y/2+	5B/5—	

To use the Fortune Teller program:

Ask MINIVAC a yes-or-no question.

Push the "contemplate" button (pushbutton 2). Hold the "contemplate" button down while MINIVAC considers the question. When you feel that MINIVAC has had sufficient time to consider the question, release the "contemplate" button and push the "answer" button (pushbutton 4).

MINIVAC's answer will appear in the lights as indicated above. (You may wish to label the lights by placing a piece of paper with the appropriate indication above the lights).

The Random Number Generator

The program for the Fortune Teller can also be used for any game which requires generating a random number from 1 through 4. Using the preceding program as a random number generator is just the same as throwing a four-sided die, or spinning a dial with four possible stops.

To generate a random number:

Program MINIVAC as for the Fortune Teller program. Push pushbutton 2 to allow MINIVAC to generate a random number. Release pushbutton 2 and push pushbutton 4 to allow MINIVAC to display the number it has generated. (Light 1 indicates "1", light 2 indicates "2", light 3 indicates "3" and light 4 indicates "4".)

Scissors, Paper or Stone

In this modern version of a very old game, MINIVAC acts as referee and indicates "win" or "tie". Two people play the game, and each player indicates his play (scissors, paper or stone) by pushing the appropriate pushbutton.

	Player 1	Player 2
Scissors	pushbutton 1	pushbutton 4
Paper	pushbutton 2	pushbutton 5
Stone	pushbutton 3	pushbutton 6

The rules of the game are:

Scissors cut paper (scissors will win over paper)
Paper covers stone (paper will win over stone)
Stone breaks scissors (stone will win over scissors)

After both players have indicated their moves, MINIVAC will indicate that:

Player 1 wins: light 1 comes on
or
Player 2 wins: light 6 comes on
or
Tie: lights 3 and 4 come on

The program for Scissors, Paper or Stone is:

1A/1G	2C/4X	3F/4F	4G/6G
1B/1—	2F/3F	3G/5G	4Y/5Y
1C/2C	2G/4G	3H/6H	5C/6C
1F/2F	2H/5H	3K/5K	5F/6F
1G/3G	2X/3L	3L/5H	5Y/6Y
1H/4H	2Y/3Y	3X/5L	6A/6G
1H/5L	3A/5K	3Y/4Y	6B/6—
1K/3K	3A/4A	4B/4—	6C/6X
1L/1X	3B/4B	4C/5X	6F/6—
1L/3H	3C/4C	4F/5F	6Y/6+
1Y/2Y			

To Play:

Each player decides upon a move (scissors, paper or stone) and indicates it by pushing his appropriate pushbutton. Player 1 should move just ***before*** player 2 moves since the relays will move after player 2 indicates his move. MINIVAC will indicate a win or a tie on the lights.

You may find it convenient to fix a tall piece of cardboard between pushbuttons 3 and 4 to keep players from observing their opponent's move in advance.

Reaction Time Tester

This game uses MINIVAC to judge which of two players has the faster reaction time. When lights 3 and 4 come on, each player tries to push his pushbutton down ***first.*** MINIVAC will indicate a winner, and will also indicate a foul—that is, when a player moves before the signal lights (3 and 4) come on.

Player 1 uses pushbutton 1
Player 2 uses pushbutton 6
Pushbutton 3 is the re-set button

Light 1 indicates that player 1 wins.
Light 2 indicates that player 1 has fouled.
Light 5 indicates that player 2 has fouled.
Light 6 indicates that player 2 wins.
Lights 3 and 4 give the signal "START".

The program for the Reaction Time Tester is:

1A/3F	2B/2—	3G/5G	5B/5—
1B/1—	2E/3E	3H/4H	5E/6G
1E/2E	2F/2G	3J/6Y	5F/5—
1E/1—	2F/5A	3K/4K	5L/6X
1F/2A	2G/5N	3L/4L	6B/6—
1F/1G	2H/3H	3Y/3+	6E/6G
1G/5J	2K/3K	3Z/6H	6E/D2
1H/2H	2L/3L	4B/4—	6F/6—
1H/3Z	3A/6K	4F/6A	6L/6+
1K/2K	3A/4A	4F/4G	6L/D17
1L/2L	3B/4B	4G/5K	D16/D17
1X/5H	3E/4E	4K/D19	D18/M—
1Y/4J	3F/3G	4L/4—	

To play:

Each player should be ready at his pushbutton.
Push the re-set button (pushbutton 3).
As soon as lights 3 and 4 come on, each player tries to be the first to push his pushbutton. MINIVAC will indicate the winner on the lights.
If either player pushes his pushbutton ***before*** lights 3 and 4 come on, MINIVAC will indicate who has fouled and the game must be begun again.
To start another game push the re-set button and proceed as above.

MINIVAC 601 AND THE MINIVAC MANUAL

ARE PRODUCTS OF:

372 Main Street, Watertown, Massachusetts