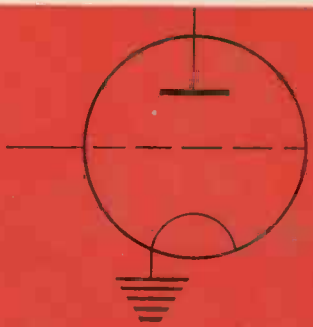
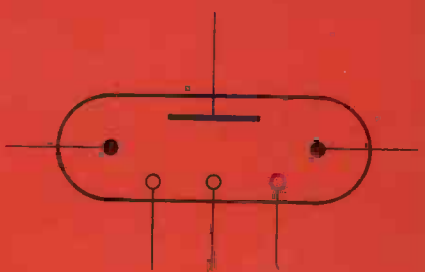
COMPUTERS



FOR THE

AMATEUR CONSTRUCTOR



R.H. WARRING

Basically the circuit elements used in computers are simple, and it is only the multiplicity of such circuits, cross-connected or interlinked, that makes the final computer a highly complex machine. Once it is appreciated that the amazing performance of a computer reduces to a mere basic function it becomes a subject for practical study and amateur construction, especially if we concentrate on the heart of the computer, its arithmetic block, which is the calculating machine that provides the answers in digital arithmetic or binary logic. It is readily possible to understand how such units work, and to duplicate them successfully without having any previous experience in electronic design other than the ability to follow simple circuit diagrams and to wire-up simple circuits.

This book sets out to provide a background knowledge of computers as a whole and their working principles, but it also includes descriptions of some subjects, such as memory devices, which are beyond the scope of amateur construction but which are necessary to obtain an overall picture of modern computers and their capabilities. The bulk of the book, however, is devoted to practical descriptions of computer circuits and circuit elements well within the capabilities of anyone to build. In particular, detailed circuits are given of two designs intended for amateur construction —one an adder/subtractor and the other a decimal counter. Both are easy to build, yet they are quite advanced computer designs.

20/- net

COMPUTERS
FOR THE AMATEUR CONSTRUCTOR

COMPUTERS
FOR THE
AMATEUR CONSTRUCTOR

R. H. WARRING

LONDON
MUSEUM PRESS LIMITED

First published in Great Britain by Museum Press Limited
39 Parker Street, London, W.C.2

PRINTED IN GREAT BRITAIN
BY EBENEZER BAYLIS AND SON LTD.
THE TRINITY PRESS, WORCESTER, AND LONDON
R.3880

CONTENTS

PREFACE

THIS is essentially an elementary book on computers. Basically the circuit elements used in computers are simple, and it is only the multiplicity of such circuits, cross-connected or interlinked, that makes the final computer a highly complex machine and provides it with the "intelligence" to solve problems. This it does with no more inherent ability than the basic circuits of a digital computer possess to distinguish between "on" and "off" or "stop" and "go."

Once we appreciate that the amazing performance of a computer reduces to a mere basic function it becomes a subject for practical study and amateur construction, especially if we concentrate on the heart of the computer, its arithmetic block, which is the calculating machine that provides the answers in digital arithmetic or binary logic. It is readily possible to understand how such units work, and to duplicate them successfully without any previous experience in electronic design other than the ability to follow simple circuit diagrams and to wire-up simple circuits involving such familiar components as resistors, capacitors and transistors. If we want to carry simplicity to extremes, we can make an elementary computer merely by wiring up switches.

This book sets out to provide a background knowledge of computers as a whole and their working principles, but it also includes descriptions of some subjects, such as memory devices, which are beyond the scope of amateur construction but which are necessary to obtain an overall picture of modern computers and their capabilities.

The bulk of the book, however, is devoted to practical descriptions of computer circuits and circuit elements well within the capabilities of anyone to build. In particular, detailed circuits are given of two designs intended for amateur construction—one an adder/subtractor and the other a decimal counter. Both are easy to build, yet they are quite advanced computer designs which will do as much as the arithmetic blocks of computers costing tens of thousands of pounds, although they can be constructed for only a few pounds. Cost is, in fact, directly related to the number of stages that are built into a home-made computer—each stage increasing its calculating capacity. The all-transistor circuitry of the adder/subtractor also makes it suitable for working off two 6-volt dry batteries, which obviate the additional cost, and bugbears, of high voltage supplies.

For those who wish to develop additional computers on a logic basis rather than for arithmetical working, practical circuit details

are given for a complete variety of logic switches or "gates"; and the basis of computer logic is fully described to show how such elements can be used in combination. It presents a new field of learning, one that is fascinating and becomes "alive" and divorced from its previous mystery when studied in a practical manner with working circuits. The book, indeed, reduces the subject of computer design and construction to the "everyday" level of a practical hobby interest, although there are, of course, distinct limits beyond which an amateur constructor cannot go.

The author would particularly like to express his thanks to Mullard Ltd., who are responsible for the design of the two main computer circuits described, and for their permission to include these as representative of the finest designs of their type, particularly as they are so suitable for constructing by the amateur. At the same time it should be pointed out that none of the circuits included is available in kit or component form, although all components used are standard and should be readily available from local radio shops or specialist suppliers for the home radio enthusiast.

CHAPTER 1

DEFINING COMPUTERS

PRIMARILY, computers are counters, but today most people associate the term with complex electronic devices capable of the most sophisticated functions at speeds so rapid as to be almost beyond belief. Nor is the intelligence of a computer limited to counting. Computers can be given "memory" so that they can store information, control fully automated processes, translate directly from one language into another, or even translate spoken words into commands.

Basically, however, the computer element is a simple device with extremely limited capability, its main virtues being positiveness of response (i.e. it does not make a mistake unless its working is faulty) and speed of response. It is only when a vast number of such computer elements are interlinked that the complete computer device becomes a highly sophisticated machine. Even then its reasoning or "brain" power is still limited to counting actions—an elementary process of seeking a solution compared with the intricate functioning of the human brain—but it scores in the speed with which it can arrive at solutions by "counting."

This basic "counting" action will be described in more detail in subsequent chapters, but it is worth recording here that the simple computer tackles problems the long way round and arrives at an answer rapidly only because of its speed of working. If commanded to multiply a given number by, say, 527, it would not actually carry out the conventional process of multiplication but would *add* that number 527 times. Note, also, that it is "commanded" and not "asked." A computer can only be programmed by human intelligence. It cannot reason in the human sense. It can only act on the information fed into it. It also needs to work on much simpler arithmetic than the human brain readily masters, if it is not to become too unwieldy.

Computers may be manual, mechanical, electro-mechanical, or electronic. The notched stick and knotted cord used by man several thousand years BC were elementary computers. The Egyptian sand table and the Arabian abacus were developments that made it easier to deal with larger numbers by "carrying" forwards and backwards in columns for units, tens, hundreds and thousands, and so on (Fig. 1). They were practical devices with a *positive* performance eliminating, or at least greatly reducing, the possibility of human

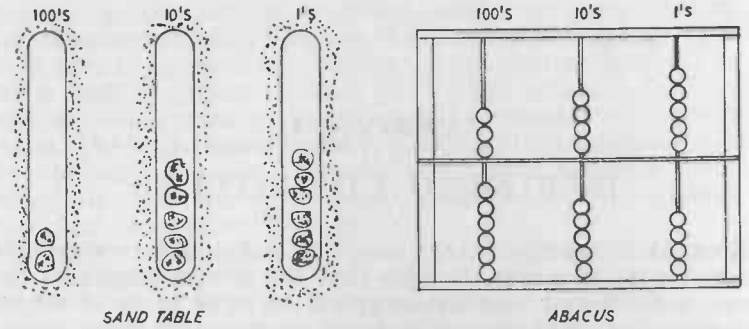


FIG. 1. TWO OF THE EARLIEST FORMS OF "COMPUTERS"

error. The abacus is still in use today, particularly for teaching elementary arithmetic.

It was not until thousands of years later that the same principle was extended to mechanical counting, using gear wheels instead of beads or counters in straight-line columns. In effect, this was merely bending the "columns" around in a circle, with gear teeth replacing the individual counters to give a more compact layout. However, it had the further advantage that intermeshing of the gear wheels provided automatic "carry forward" or "borrowing" from adjacent columns (Fig. 2). In this diagram only a "units" and "tens" wheel are shown working, but the "10s" wheel can be related to a "100s" wheel in the same manner, via a single pick-up tooth. Note also that an idler gear is necessary between each "column" so that the respective column wheels all rotate in the same direction.

This is the basis of the rotary calculator, which is a true computer and adaptable to a variety of units. It may be a decimal calculator, i.e. working in "columns" of 10 or in other units as required, e.g.

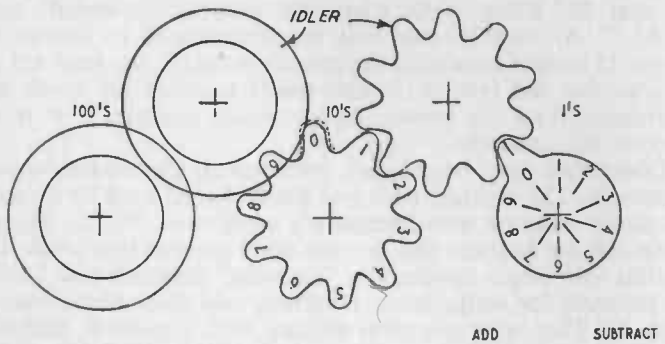


FIG. 2. THE MECHANICAL COUNTER IS A COMPUTER

pounds, shillings and pence. Its mechanical function may also be extended to ancillary services, such as recording individual entries as well as totalling, as in cash registers. It can also be adapted to multiplication and division by the principle of multiple addition or multiple subtraction, respectively, as previously described. And because it utilizes a rotary-type mechanism it can be made quite compact. The mileometer (odometer) fitted to a bicycle or behind the speedometer dial on a car is an example of just how compact a readable rotary calculator can be made with a counting capacity up to 99,999.

All such devices so far described are essentially adding machines working with integers. A similar function can be performed by simple electronic circuits, which permit considerable reduction in size and vastly increased speed of operation. Such circuits then form a *digital computer*, which literally counts numbers. This is one basic type of computer. The other is the *analog* computer, which measures a quantity, and again, this can be done mechanically or electronically. It is important to appreciate the difference between the two. The digital computer *counts* and the analog computer *measures*. There is also a difference in the performance of the two types. The digital computer is exact, but the analog computer, although simpler and more direct, is not a high-precision device and has only limited flexibility. Thus, each type has particular fields of application.

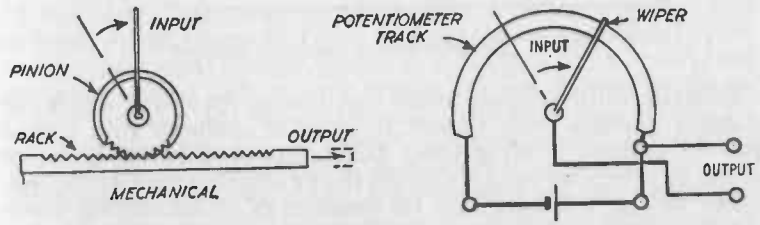


FIG. 3. TWO SIMPLE FORMS OF ANALOG RELATIONSHIP

Two elementary forms of analog computer are shown in Fig. 3. One is mechanical, the input being applied to a shaft to turn a gear wheel which produces a corresponding linear movement of the rack engaged with the gear wheel. Thus, the output (rack movement) measures the amount of input (amount of rotation applied to the shaft and gear wheel). In the electronic equivalent, the input shaft is connected to the wiping contact of a potentiometer. Input movement is measured by the change in resistance produced in the output circuit, amounting to a change in the voltage in this circuit. This can be measured and indicated on a voltmeter or ammeter, with the scale calibrated accordingly. In both cases the degree of accuracy of measurement can be quite high, but it will be appreciated that it is

not necessarily *exact*, although for many applications complete exactness is unnecessary.

One advantage the analog computer has is that, in addition to simple measurement (addition and subtraction), it can also tackle differentiation and integration in a single stage. Differentiation is merely a rate of change of a given variable with respect to some standard (or the finding of the derivative). Thus, speed (or, strictly speaking, velocity) is the rate of change of *distance* with respect to *time*. Velocity is the derivative in this case, and the relationship would be expressed in mathematical symbols as—

$$V = \frac{dD}{dT}$$

where V = velocity,
D = distance,
T = time,
and d represents an increment or change.

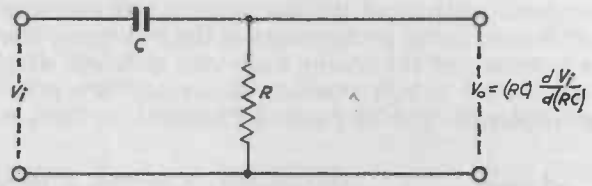


FIG. 4. DIFFERENTIATOR (ANALOG) CIRCUIT

Basically, differentiation means determining the rate at which one quantity changes with respect to another quantity (the second quantity commonly being time). If the first variable can be represented by a varying voltage (input), then a simple circuit of the type shown in Fig. 4 will perform the function of differentiation which can be "read" directly from the output voltage V_o, the complete equation also embracing the circuit constants R and C. This form of analog computer is known as a differentiator. It works by virtue of the fact that the charge on the capacitor C varies with variations of the input voltage V_1, and the value of C, together with that of the

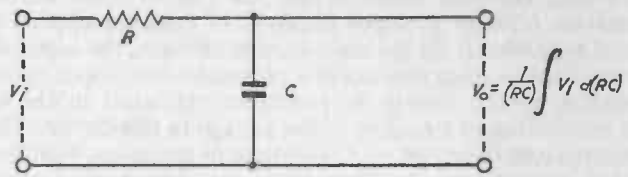


FIG. 5. INTEGRATOR (ANALOG) CIRCUIT

resistor R, renders the output voltage V_o as the rate of change of V_1 with time.

A similar circuit can be employed to perform integration (Fig. 5). Here the capacitor C stores the charge received from the input voltage V_1, and the voltage across C at any time (and thus the output voltage V_o) is the integral of the charging rate. Normally, an amplifier would be added to this circuit to prevent capacitor C discharging completely during integration.

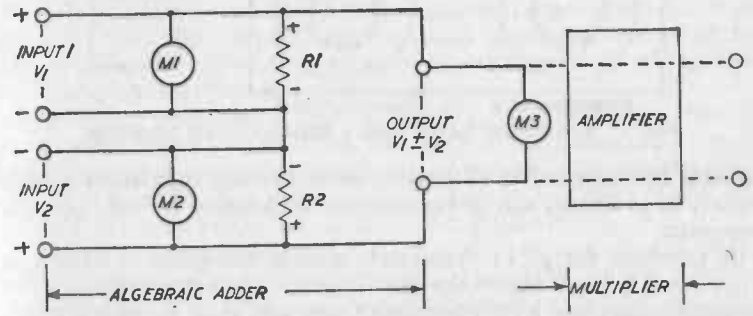


FIG. 6. A BASIC ANALOG ALGEBRAIC ADDER CIRCUIT

For normal adding functions the analog computer circuit takes a different form. Two inputs are connected to a common output, as in the basic circuit shown in Fig. 6. If resistors $R1$ and $R2$ are equal, then the same input voltage applied to each input (i.e. $V_1 = V_2$) will produce an equal voltage drop across $R1$ and $R2$ (i.e. voltmeter readings $M1$ and $M2$ will be the same). Since the two circuits are connected with opposite polarity these voltages will cancel each other out, and so there will be no output voltage ($V3 = 0$).

If, however, one input voltage is greater than the other, there will be a difference in the voltage across $R1$ and $R2$, which will be measurable at the output. This difference will be directly proportional to the difference in input voltages, so that $V3$ will actually equal $V1 - V2$. This difference will also take into account the sign of the difference, i.e. whether $V1$ is greater than $V2$, or $V2$ is greater than $V1$ by the direction of the output voltage. Thus, the circuit is an *algebraic adder*, giving $V3 = V1 \sim V2$. Such an analog circuit can perform addition and subtraction; also multiplication and division by repetitive algebraic addition. If an amplifier is introduced into the output circuit it can also perform multiplication directly by virtue of the gain of the amplifier.

Basically, this is all that analog computers consist of—passive networks of resistors (plus capacitors for differentiation or integration), a means of providing an input in analog form (varying voltage), and amplifiers to boost the output to a suitable level. By

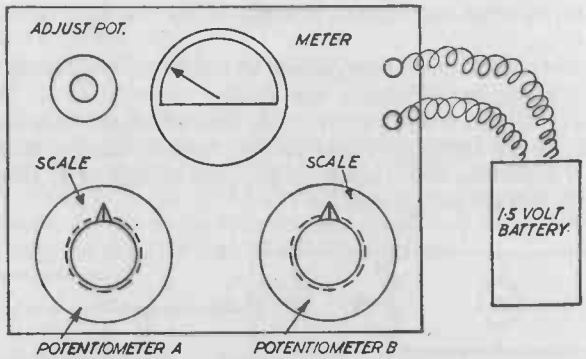


FIG. 7. COMPONENT LAYOUT FOR A SIMPLE ANALOG COMPUTER

suitable interconnexion of passive networks and amplifiers a wide variety of problems can be represented for solution by the complete computer.

A practical design of elementary analog computer is shown in Figs. 7 and 8. Fig. 7 shows the basic layout of the components—three potentiometers and a microammeter mounted on a suitable panel—and Fig. 8 the wiring diagram. Potentiometers *A* and *B* are of 25 kilohm (kΩ) value when, with a 0–500 microammeter, a single 1·5 volt battery can be used for the power supply.

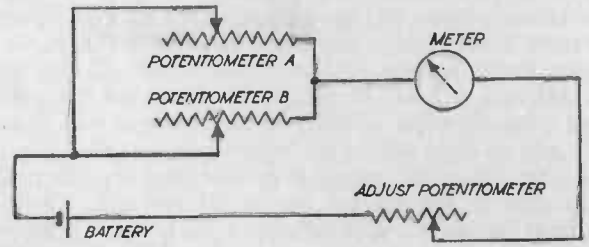


FIG. 8. WIRING DIAGRAM FOR A SIMPLE ANALOG COMPUTER

Once the circuit has been completed the analog computer must be calibrated. Potentiometers *A* and *B* are turned to maximum resistance and the "adjust" potentiometer then turned until the meter reads exactly 100 microamps (μA). A suitable value for this adjust potentiometer will be 1 or 5 kΩ. Once set up initially it should not need further adjustment.

With the set-up showing 100 μA, the pointer position for potentiometer *A* is now marked 100, and the pointer position of potentiometer *B* marked 0. This corresponds to the meter reading (100) equalling *A* plus *B* (100 + 0 = 100).

Potentiometer *A* is now adjusted until the meter reading advances

to 200 μA, and this position is marked as 200 on the potentiometer scale. Repeat the process to obtain calibrated positions for potentiometer *A* at 300, 400 and 500 μA. The potentiometer is then returned to its original (100) position.

Leaving potentiometer *A* in this position, potentiometer *B* is now turned until 200 is indicated on the meter scale. The pointer position at this setting is marked 100 for potentiometer *B*. This corresponds to the 100 position on *A* plus the 100 position on *B* = 200 on the meter (meter reading = *A* + *B*). Repeat the process for meter readings of 300, 400 and 500 to obtain calibration points for potentiometer *B* of 200, 300 and 400, respectively, i.e. with potentiometer *A* at its 100 position the calibration value of potentiometer *B* setting will always be 100 less than the meter reading.

The analog computer is now calibrated for working. Any settings of potentiometers *A* and *B* will be read on the meter as *A* + *B*. Unlike a digital computer, which can only count, the analog computer will give the sum over an infinite range of variation within the limits of the scale values, i.e. at any intermediate value.

This basic design lends itself to development in a number of ways. Different values of potentiometers and meter scales can be selected to extend the range of *A* + *B* that can be read or adapted to other units. The meter scale, for example, can be replaced by another pasted-on scale. Thus, in the elementary model described, a new scale could be calibrated for the meter, starting at zero for minimum reading (instead of 100). Further, if required, the *meter* scale could be calibrated to required positions (calibration values) on the two potentiometers.

Equally, the computer need not necessarily be confined to arithmetic working. It is only necessary to render a particular problem as an analogy to the basic relationship between current flow and variable resistance that the analog computer presents to make the computer read directly in the type of information required.

To extend the principle further, a more advanced analog computer could be designed, incorporating a number of fixed resistors rather than potentiometers, in the form of a plugboard. Each resistor value represents a different condition, value, or analogous data state, i.e. where the variation of one factor is analogous to the effect on current (the related or analogous factor) when resistance is varied.

Further extension in simple analog working is also possible by varying *voltage*. Since, basically,

$$\text{current} = \frac{\text{voltage}}{\text{resistance}}$$

it will be appreciated that varying the voltage is equivalent to multiplying or dividing, as far as the current value is concerned. This considerably extends the scope both for arithmetic and analogous working. This can be done via a variable voltage supply (e.g. a

tapped battery in a simple unit) for multiplication, or dropping resistors for division.

The analog computer is a very simple device but it is surprising what it can be made to perform. It is largely a matter of individual ingenuity as to how far it can be adapted or programmed to give the solutions required. Even with very simple designs such as the one described its scope is enormous. The elementary analog computer is also attractive because it is not expensive or complicated to make—although programming an elaborate pegboard of fixed resistor values can be a lengthy process—and in most cases it can be checked and calibrated directly.

The digital computer tends to be more complex than the analog computer because its basic element, which is nothing more than an "on-off" switch for counting, has to be incorporated in a logic circuit in order to perform the actual arithmetical operations required. The basic element must also be related to a memory or storage unit which retains the required program, or at least to an indicator circuit for visual presentation. The functions of these various sections will be described in later chapters.

Another complicating factor is that the switching circuits normally employed in digital computers can recognize only two states—"on" or "off"—and thus count in binary numbers rather than in decimals or tens. This is essentially logical, for it both reduces the number of switching circuits required and also utilizes them more fully, but it can cause confusion until the constructor has become familiar with the binary numbering system. It also means that the output device may be required to translate a binary number solution back into more familiar units or intelligence, demanding a knowledge of the relationship between the two. The following chapter is, therefore, devoted to a description of numbering systems.

CHAPTER 2

NUMBERING SYSTEMS

NUMBERING systems were developed for convenience. Pretty obviously the fingers of the hand were used as an aid to counting from very early times, and the Roman numeration system is based on this. The Roman symbols also *look* like upraised fingers—I, II, III and IIII (which was only later shortened to IV) and then V, said to represent the gap between thumb and fingers of a "full hand" of digits. Note, too, that this numbering system was to base 5, i.e. involved a carry over at intervals of five—thus the inclusion of L for fifty and D for five hundred in Roman numerals.

Although used for centuries (and still employed in a limited way) the Roman number system was too cumbersome and too complicated to be practicable for calculations. The Romans themselves, in fact, normally preferred the abacus for working out arithmetical problems and used their numerals only for recording answers. Even then, for higher numbers at least, translating a number often demanded *counting* rather than straightforward reading. Thus, MMDCXLVII demands a count of the thousands (Ms) and interpretation of the other numerals to a base of 5 in order to arrive at 2,647.

The decimal system, which is now the everyday numbering system adopted the world over, depended on the invention of the "zero." This seems to have evolved in ancient India and led to a revolution in mathematics with the eventual introduction of the Hindu-Arabic decimal system to base 10 (meaning virtually the use of ten basic symbols).

The decimal system (also known as the denary numbering system) represents a very compact way of writing numbers, particularly large numbers, and simplifies arithmetical calculations by virtue of the significance of the *position* of individual numbers. In effect, it is only doing what the abacus does in grouping numbers in "columns" of units, tens, hundreds, and so on, although rather than "columns of 10" the columns or groups refer to powers of 10, with the digits entered in their proper places.

Thus, the number 32,478 represents

$$\begin{aligned} 3 \times 10^4 &= 3 \times 10{,}000 = 30{,}000 \\ 2 \times 10^3 &= 2 \times 1{,}000 = 2{,}000 \\ 4 \times 10^2 &= 4 \times 100 = 400 \\ 7 \times 10^1 &= 7 \times 10 = 70 \\ 8 \times 10^0 &= 8 \times 1 = 8 \\ & \qquad\qquad\qquad\quad 32{,}478 \end{aligned}$$

Note that 10^0 (or any number to the power of 0) equals 1. This can also be written in columns—

10^4	10^3	10^2	10^1	10^0
(10,000)	(1,000)	(100)	(10)	(1)
3	2	4	7	8

Addition and subtraction follows logically from adding (or subtracting) numbers in their correct and respective columns, carrying (or borrowing) as necessary. Regardless of the size of the number this is all done with ten different digits—0,1,2,3,4,5,6,7,8 and 9—and a recognizable state or position for each digit.

Suppose, now, the decimal system is incorporated with a visual indicating system such as a series of light bulbs. Numbers can be entered on the column indicators by lighting up the appropriate bulbs. However, each column will require ten bulbs (one for each digit); and to enter the same number as above would require a set-up like this—

	10^4	10^3	10^2	10^1	10^0
0					
1					
2		ON			
3	ON				
4			ON		
5					
6					
7				ON	
8					ON
9					

This is a workable system, and additional numbers can be added or subtracted to work the board as an arithmetical computer with suitable carryover from one column to the next highest column. However, the solution will be represented only by one light on (one digit) in each column—a maximum of 5 in all, leaving 45 other bulbs idle. This involves a considerable waste of space and only 10 per cent (at most) use of individual bulbs.

Transistor switches used in place of light bulbs in a digital computer have a similar "on-off" performance only. Thus, the same objection applies. Computing on the decimal system basis only, leads to unnecessary bulk and cost and low utilization of individual transistors, although it can be a workable scheme from the point of view of deriving arithmetical solutions. A far more satisfactory answer is to adopt a numbering system consistent with the intelligence of an on-off switching system so that each switching element can be fully and continuously employed.

Such a system is binary numbering, which computes to base 2. This means that there are only two digits—a 1 and a 0—and this matches exactly the two states of electricity—"on" and "off." It is

TABLE 1

Binary	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Decimal Equivalent	16,384	8,192	4,096	2,048	1,024	512	256	128	64	32	16	8	4	2	1
Set	1	1	1	1	1	1	0	1	1	0	1	1	1	1	0
Indicators	ON	ON	ON	ON	ON	ON	—	ON	ON	—	ON	ON	ON	ON	—

Decimal number indicated is:

16,384
8,192
4,096
2,048
1,024
512
128
64
16
8
4
2

32,478

not a new system; it was first employed by the Chinese over 4,000 years ago and then lapsed into disuse. It has been revived as the numbering system most easily adaptable for use in digital computers, and an appreciation of binary arithmetic is an essential part of understanding how such computers work.

With the modern binary system we have just two digits—1 and 0. To express a "bit" (i.e. a *bi*nary digi*t*) only one indicator is required. Thus, each "column" has only one line, but the equivalent *decimal* values of the individual columns rise in powers of 2, as in Table 1.

The same decimal number as before, 32,478, is shown entered on the "binary board" and is derived (as a decimal number) by the sum of the decimal equivalents to the binary presentation 1111110 11011110, as shown under the table.

It may look more complicated than the decimal system in breakdown, which it is, but this is only a matter of conventional relationship. As far as the digital computer is concerned it can work directly and very rapidly with binary numbers with a minimum of switching elements, and each element is continuously utilized (i.e. either ON or OFF).

There are several further advantages of the binary system. There are no "multiplication tables" for example, and very few rules to apply to cover any problem involving addition, subtraction, multiplication or division. It is simply a case of breaking the problem down into a series of individual operations, which, in fact, is just what a digital computer is suited to do via its individual "on-off" switching elements.

Addition is perfectly straightforward: $0 + 0 = 0$; $1 + 0$ or $0 + 1 = 1$; $1 + 1 =$ one-zero (10, *not* ten) involving a carry forward of 1 into the next column.

Example:	Decimal	Binary
	12	1100
	6	110
	18	10010

Where "carry 1" is involved it is often clearer to complete addition in two stages, first extracting the partial sum (line 3) and entering the 1s carried over in the appropriate places in the following line (line 4). The final addition then results from the sum of the two partial sums (lines 3 and 4).

Example:	Decimal	Binary	
	61	111101	
	20	10100	
	81	101001	partial sum
		1 1	carry
		1010001	

Subtraction may be done directly, or by complements. With direct subtraction it is only necessary to remember to borrow the 1 from the next (top) column when a 1 appears in the second line under a 0 in the first line.

Example:	Decimal	Binary
	12	1100
	−5	−101
	7	111

Subtraction by complements involves rewriting the number to be subtracted as a complement (which means changing all the 1s to 0s and the 0s to 1s, *adding* the two numbers (i.e. the first number and the complement of the second) and then adding the extra 1 or *end-around carry*. This is a very simple process, although the explanation of why it works is somewhat involved and beyond the scope of present description.

Example. To subtract binary 1011 from binary 1101—

i.e. 1101
−1011

write second line as complement 0100 and *add*

```
              1101
              0100
              ----
         ——— 10001
         ↓
carry round———→1
 and add
              ----
answer        0010
```

The business of *end-around* carry is a vital factor in computer subtraction, particularly as the conception of using the complement and *addition* matches the normal switching action of the basic digital computer switching circuit.

In multiplication, every product is equal to 0, except 1×1, which is equal to 1. Multiplication then follows from a series of shifts and additions. Note also that there are no carries, except when adding partial products. Each 1 is a shift left and an addition. Each 0 is a shift left but no addition.

Basic:

0	0	1	1
× 0	× 1	× 0	× 1
0	0	0	1

Example:	*Decimal*	*Binary*	
	12	1100	
	×5	× 101	
	60	1100	(first 1)
		0000.	(0-shift left)
		1100..	(second 1 shift left)
		111100	add

Demonstration of shifting left (and adding 0 to complete binary number):

	Binary	*Decimal*	*Significance*
	110	6	—
Shift left	110(0)	12	multiplied by 2
Shift left	1100(0)	24	multiplied by 2 again
Shift left	11000(0)	48	multiplied by 2 again

Thus, *each* shift to the left is equivalent to multiplying the original number by 2.

Division follows in a similar fashion but involves a shift to the right.

	Binary	*Decimal*	*Significance*
	1010000	80	—
Shift right	101000	40	divide by 2
Shift right	10100	20	divide by 2 again
Shift right	1010	10	divide by 2 again
Shift right	101	5	divide by 2 again

Division can also be worked out by conventional long division, bearing in mind that the divisor is subtracted from the dividend to yield a 1 or 0, as appropriate.

Example:

Decimal	*Binary*
21	10101
$5\overline{)105}$	$101\overline{)1101001}$
10	101
5	0110
remainder 0	101
	0110
	101
	0101
	101
	remainder 0

Note: A remainder will simply indicate the fractional number concerned relative to the denominator.

It is not easy to master the technique of binary number multiplication and division when first faced with it and previously accustomed to working with decimal numbers. However, the process is basically the same as decimals except that in the binary system a shift to right or left means multiplying or dividing, respectively, the original decimal by 2, and in the decimal system a shift means multiplying or dividing by 10. Thinking "to base 2" makes it quite clear that one shift left means multiplying by two, two shifts left multiplying by 4, and so on. Similarly, shifts to the right produce division in powers of 2.

Actually, although it is possible to convert from binary to decimal and back again, this is not always convenient or useful. In fact, the *octal* system is directly related to the binary system and much easier to work with. Its main use is for checking computer performance (binary arithmetic) against other calculators (mechanical or human).

The octal system is a numbering system to base 8. In other words it has eight digits, from 0 to 7, relative to the decimal system, although decimal 10 equals octal 8.

The advantage of octal numbers is that they can be written as groups of three binary digits, called *binary triplets*. Thus, conversion from a binary number to an octal number is direct and straightforward—

Octal	*Binary Triplet*		
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

To convert a binary number into its octal number the binary number is broken down into groups of three or triplets. If necessary, zeros are added in front of the number to complete a set of triplets. The corresponding octal number then follows from the equivalent of the various triplets.

Example:

Binary	10110011			
group in triplets	10	110	011	
add zero to complete	010	110	011	
corresponding octal numbers	2	6	3	

i.e. octal number = 263.

Octal numbers can be used to check computer arithmetical solutions by comparing the answers obtained by the two numbering systems. A worked-out example should make this clear.

Binary sum			*Octal sum*
110			6
+ 010			+ 2
1000			10 octal
001	000		
1	0	octal equivalent	

The two octal numbers agree, i.e. the one derived directly by octal number working and the other extracted as the octal equivalent of the binary sum solution. Thus, the binary arithmetic is correct.

TABLE 2: BINARY NUMBERS

DECIMAL	2^5	2^4	2^3	2^2	2^1	2^0
0						0
1						1
2					1	0
3					1	1
4				1	0	0
5				1	0	1
6				1	1	0
7				1	1	1
8			1	0	0	0
9			1	0	0	1
10			1	0	1	0
11			1	0	1	1
12			1	1	0	0
13			1	1	0	1
14			1	1	1	0
15			1	1	1	1
16		1	0	0	0	0
17		1	0	0	0	1
18		1	0	0	1	0
19		1	0	0	1	1
20		1	0	1	0	0
21		1	0	1	0	1
22		1	0	1	1	0
23		1	0	1	1	1
24		1	1	0	0	0
25		1	1	0	0	1
26		1	1	0	1	0
27		1	1	0	1	1
28		1	1	1	0	0
29		1	1	1	0	1
30		1	1	1	1	0

TABLE 2: BINARY NUMBERS
(continued)

DECIMAL	2^5	2^4	2^3	2^2	2^1	2^0
31		1	1	1	1	1
32	1	0	0	0	0	0
33	1	0	0	0	0	1
34	1	0	0	0	1	0
35	1	0	0	0	1	1
36	1	0	0	1	0	0

DECIMAL	2^6	2^5	2^4	2^3	2^2	2^1	2^0
37		1	0	0	1	0	1
38		1	0	0	1	1	0
39		1	0	0	1	1	1
40		1	0	1	0	0	0
41		1	0	1	0	0	1
42		1	0	1	0	1	0
43		1	0	1	0	1	1
44		1	0	1	1	0	0
45		1	0	1	1	0	1
46		1	0	1	1	1	0
47		1	0	1	1	1	1
48		1	1	0	0	0	0
49		1	1	0	0	0	1
50		1	1	0	0	1	0
51		1	1	0	0	1	1
52		1	1	0	1	0	0
53		1	1	0	1	0	1
54		1	1	0	1	1	0
55		1	1	0	1	1	1
56		1	1	1	0	0	0
57		1	1	1	0	0	1
58		1	1	1	0	1	0
59		1	1	1	0	1	1
60		1	1	1	1	0	0
61		1	1	1	1	0	1
62		1	1	1	1	1	0
63		1	1	1	1	1	1
64	1	0	0	0	0	0	0
65	1	0	0	0	0	0	1
66	1	0	0	0	0	1	0
67	1	0	0	0	0	1	1
68	1	0	0	0	1	0	0

TABLE 2: BINARY NUMBERS
(*continued*)

DECIMAL	2^6	2^5	2^4	2^3	2^2	2^1	2^0
69	1	0	0	0	1	0	1
70	1	0	0	0	1	1	0
71	1	0	0	0	1	1	1
72	1	0	0	1	0	0	0
73	1	0	0	1	0	0	1
74	1	0	0	1	0	1	0
75	1	0	0	1	0	1	1
76	1	0	0	1	1	0	0
77	1	0	0	1	1	0	1
78	1	0	0	1	1	1	0
79	1	0	0	1	1	1	1
80	1	0	1	0	0	0	0
81	1	0	1	0	0	0	1
82	1	0	1	0	0	1	0
83	1	0	1	0	0	1	1
84	1	0	1	0	1	0	0
85	1	0	1	0	1	0	1
86	1	0	1	0	1	1	0
87	1	0	1	0	1	1	1
88	1	0	1	1	0	0	0
89	1	0	1	1	0	0	1
90	1	0	1	1	0	1	0
91	1	0	1	1	0	1	1
92	1	0	1	1	1	0	0
93	1	0	1	1	1	0	1
94	1	0	1	1	1	1	0
95	1	0	1	1	1	1	1
96	1	1	0	0	0	0	0
97	1	1	0	0	0	0	1
98	1	1	0	0	0	1	0
99	1	1	0	0	0	1	1
100	1	1	0	0	1	0	0
101	1	1	0	0	1	0	1
102	1	1	0	0	1	1	0
103	1	1	0	0	1	1	1
104	1	1	0	1	0	0	0
105	1	1	0	1	0	0	1
106	1	1	0	1	0	1	0
107	1	1	0	1	0	1	1
108	1	1	0	1	1	0	0
109	1	1	0	1	0	0	1

TABLE 2: BINARY NUMBERS
(*continued*)

DECIMAL	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
110		1	1	0	1	1	1	0
111		1	1	0	1	1	1	1
112		1	1	1	0	0	0	0
113		1	1	1	0	0	0	1
114		1	1	1	0	0	1	0
115		1	1	1	0	0	1	1
116		1	1	1	0	1	0	0
117		1	1	1	0	1	0	1
118		1	1	1	0	1	1	0
119		1	1	1	0	1	1	1
120		1	1	1	1	0	0	0
121		1	1	1	1	0	0	1
122		1	1	1	1	0	1	0
123		1	1	1	1	0	1	1
124		1	1	1	1	1	0	0
125		1	1	1	1	1	0	1
126		1	1	1	1	1	1	0
127		1	1	1	1	1	1	1
128	1	0	0	0	0	0	0	0
129	1	0	0	0	0	0	0	1
130	1	0	0	0	0	0	1	0
131	1	0	0	0	0	0	1	1
132	1	0	0	0	0	1	0	0
133	1	0	0	0	0	1	0	1
134	1	0	0	0	0	1	1	0
135	1	0	0	0	0	1	1	1
136	1	0	0	0	1	0	0	0
137	1	0	0	0	1	0	0	1
138	1	0	0	0	1	0	1	0
139	1	0	0	0	1	0	1	1
140	1	0	0	0	1	1	0	0
141	1	0	0	0	1	1	0	1
142	1	0	0	0	1	1	1	0
143	1	0	0	0	1	1	1	1
144	1	0	0	1	0	0	0	0
145	1	0	0	1	0	0	0	1

TABLE 3: BINARY NUMBER EQUIVALENTS

Binary	Decimal	Binary	Decimal
2^0	1	2^{25}	33554432
2^1	2	2^{26}	67108864
2^2	4	2^{27}	134217728
2^3	8	2^{28}	268435456
2^4	16	2^{29}	536870912
2^5	32	2^{30}	1073741824
2^6	64	2^{31}	2147483648
2^7	128	2^{32}	4294967296
2^8	256	2^{33}	8589934592
2^9	512	2^{34}	17179869184
2^{10}	1024	2^{35}	34359738368
2^{11}	2048	2^{36}	68719476736
2^{12}	4096	2^{37}	137438953472
2^{13}	8192	2^{38}	274877906944
2^{14}	16384	2^{39}	549755813888
2^{15}	32768	2^{40}	1099511627776
2^{16}	65536	2^{41}	2199023255552
2^{17}	131072	2^{42}	4398046511104
2^{18}	262144	2^{43}	8796093022208
2^{19}	524288	2^{44}	17592186044416
2^{20}	1048576	2^{45}	35184372088832
2^{21}	2097152	2^{46}	70368744177664
2^{22}	4194304	2^{47}	140737488355328
2^{23}	8388608	2^{48}	281474976710656
2^{24}	16777216	2^{49}	562949953421312
		2^{50}	1125899906842624

CHAPTER 3

BINARY COUNTERS

A BASIC binary counter is a two-state electrical device which is either ON or OFF. The ON state condition represents a 1 in the binary numbering system, and the OFF state an 0. The working of a very elementary computer on this basis can be demonstrated by simple electrical circuits and switches, or relays. The relay provides a somewhat more realistic conception of a practical computer, since many of the early electronic computers were based on such devices. The typical circuit of Fig. 9 shows the set-up for an elementary binary adder, with the input commanded by manual operation of two switches.

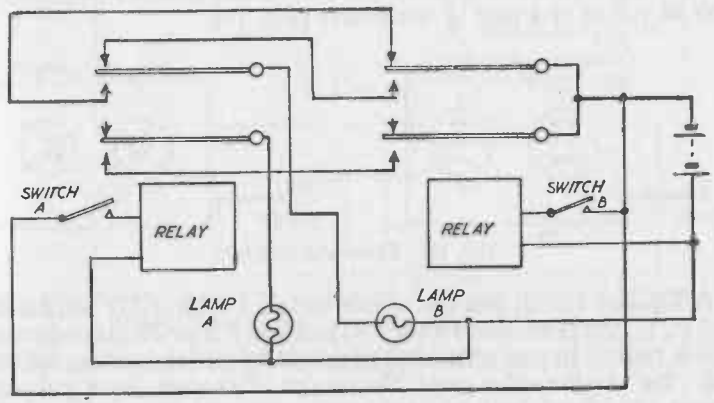


FIG. 9. RELAY-TYPE BINARY ADDER

Two counting elements are involved—the relays—each of which can be either ON or OFF, depending on whether or not the input circuit to the relay is completed by closure of the respective switch. For interpretation of the state of the devices (relays) an external circuit is used, connected to the relay contacts and using lamps as indicators.

This circuit now has the ability to count and indicate up to one–zero (10) in the binary numbering system, as indicated by the following possible conditions—

Input conditions		*Representing*	*Lamps*		*Representing*
Switch *A*	Switch *B*		*A*	*B*	
OFF	OFF	0 + 0	OFF	OFF	0
OFF	ON	0 + 1	OFF	ON	1
ON	OFF	1 + 0	OFF	ON	1
ON	ON	1 + 1	ON	OFF	10

The performance of such an elementary computer is distinctly limited. It can only count up to 10 (pronounced one-zero) in the binary numbering system, or the equivalent of 2 in the decimal system. It does, however, show the principle of binary addition by utilizing ON or OFF conditions in an electrical circuit; and it does fulfil the practical requirements of a computer in indicating the overall state or result. It could be extended to give greater coverage with additional relays and cross-links circuits, but the whole system would tend to become unwieldy if mechanical switching elements were used in this manner.

The modern electronic counter employs electronic elements for switching, and any two-state device such as valves, transistors or magnetic cores can be used. The basic circuit employed is normally a "flip-flop," which is a bistable multivibrator circuit using either a pair of valves or a pair of transistors (Fig. 10).

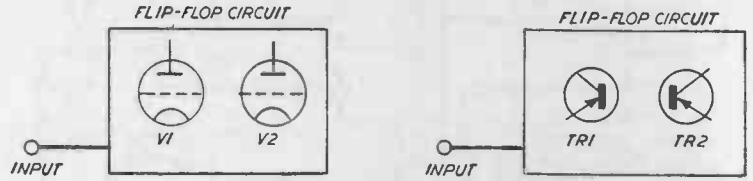


FIG. 10. FLIP-FLOP CIRCUITS

A flip-flop circuit has two stable states: *V*1 (or *TR*1) conducting and *V*2 (or *TR*2) off; and *V*1 (or *TR*1) off and *V*2 (or *TR*2) conducting. It will remain in one state until triggered by an input pulse, when it will "flip" to the other state. On receipt of another input pulse the circuit will "flop" back to its original state, and so on. Thus, every *two* pulses returns the circuit to its original state. The complete flip-flop counts by binary 10, or 2 in the decimal system, in the following manner—

Pulse	*V*1 (*or TR*1)	*V*2 (*or TR*2)	*Indication*
–	OFF	ON	0
*	ON	OFF	1
*	OFF	ON	0
*	ON	OFF	1
*	OFF	ON	0

and so on.

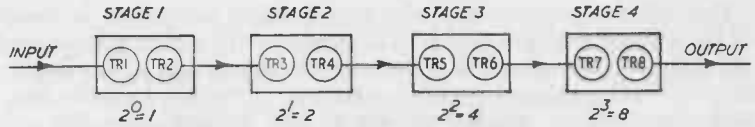


FIG. 11. FOUR STAGE FLIP-FLOP FOR BINARY COUNTING

The flip-flop circuit corresponds to a complete or single-stage counting by 2^0 (= 1 decimal). The range of counting can be extended by connecting a number of stages in series. The second stage will then count by 2^1 or 2; the third stage by 2^2 or 4; and so on. Thus, the four-stage flip-flop shown in Fig. 11 counts by 1 + 2 + 4 + 8 = 15. Note, however, that at the 16th pulse the complete circuit reverts to its original all-zero state.

To become a practical circuit the counter needs to count in either direction (i.e. add and subtract), and also a means of indicating the state of each stage so that the state of count can be read.

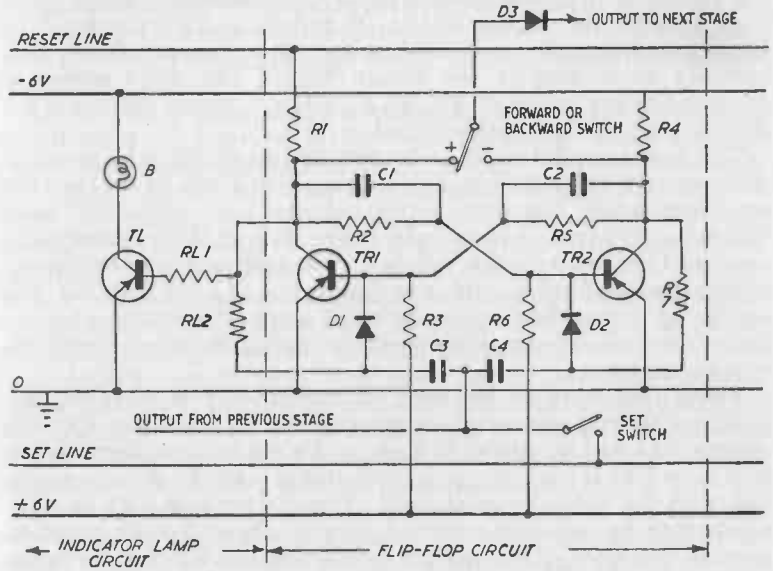


FIG. 12. COMPLETE COUNTER STAGE FOR AN ADDER/SUBTRACTOR

*R*1	1kΩ	*C*1	6·4μF
*R*2	4·7kΩ	*C*2	6·4μF
*R*3	220 kΩ	*C*3	0·22μF
*R*4	1kΩ	*C*4	0·22μF
*R*5	4·7kΩ	*TL*	OC72 or OC83
*R*6	220kΩ	*TR*1	OC71
*RL*1	6·8kΩ	*TR*2	OC71
*RL*2	2·2kΩ	*D*1	OA85 or OA81
		*D*2	OA85 or OA81
		*D*3	OA85 or OA81

The following description of a binary adder/subtractor is based on an original Mullard design developed specifically as a low-speed unit with a counting rate of about one per second and with a facility for both backward and forward counting. Provision has also been made for digits or whole numbers to be "written" into the unit by various means. It is a circuit designed for amateur construction and offers an excellent exercise both in the construction and working of a digital computer and in the application of binary arithmetic. It can be extended to any number of stages desired, when it can cope with arithmetic problems as an advanced commercial computer.

The circuit for a single stage, which includes its own indicator lamp, is shown in Fig. 12. Working of each binary stage is as follows. Positive input pulses are applied via the set switch connected to the set line or output from a previous binary stage, or from a pulse-shaping circuit. Pulses are applied to the junction of capacitors *C*3 and *C*4 and from thence to the base of the two transistors in the flip-flop circuit. If *TR*1 is conducting and *TR*2 non-conducting, the pulse is directed via the steering diodes *D*1 and *D*2 to *TR*1, causing it to become momentarily non-conducting. After a short delay *TR*2 becomes conducting or the circuit "flips." The delay period is governed by the values of *R*2, *R*5, *C*1 and *C*2, which determine the *time constant* of the bistable circuit.

This time constant needs to be arranged so that it is longer than the time over which the input pulse is applied. Then, by the time the switching action has been completed the input pulse will have decayed and the circuit remains in its new state until the second pulse appears. Consecutive pulses will then cause each transistor to become alternately conducting and non-conducting, i.e. the circuit will operate as a true flip-flop, changing its state after receipt of every pulse. The time constant of the particular circuit described is approximately one second.

Visual indication of the state of the flip-flop is given by the indicator lamp *B* connected to the collector of transistor *TR*1 via resistor *RL*1 and transistor *TL* (acting as a current amplifier). When transistor *TR*1 is conducting, its collector is virtually at zero potential, and the lamp remains unlit. When transistor *TR*1 is non-conducting its collector potential rises to about 5 volts negative, causing transistor *TL* to become conductive, and lighting the lamp.

The forward or backward switch provides the means of giving either forward counting (addition) or backward counting (subtraction). When set to the positive side, as shown in Fig. 12, the stage is set for forward counting. When transistor *TR*1 switches from a non-conducting to a conducting state, a positive pulse is fed via diode *D*3 to the next binary stage. If the switch is set to the negative side the second transistor *TR*2 controls the pulses fed to the next binary stage, and since this transistor is in opposite phase to *TR*1, positive pulses are fed to the next binary stage, when *TR*1 changes

from a conducting to a non-conducting state. Thus, in this condition the next stage adds the *complement* of digits, which is equivalent to subtraction (as explained in Chapter 2). With the switch set to the positive side the stage performs normal binary addition.

The complete circuit diagram for the binary adder/subtractor is shown in Fig. 13, together with a list of components required. The circuit (and components list) is complete for three stages, but it is a simple matter to extend the circuit to any number of stages required by the addition of further binary stages, and to determine the additional components required. This complete circuit also includes a pulse shaper, the purpose of which is described later.

The appearance of an additional switch, *S*5, will be noted in the complete circuit. This is to provide "cancellation" or to set zero facilities by connecting transistor *TR*6 to the $-6V$ rail through this switch (*S*5) which is normally in the made position. If the switch is operated (opened), transistor *TR*6 is disconnected from its collector supply voltage and its indicator lamp is extinguished. When the set zero switch *S*5 is returned to its normal (make) position transistor *TR*6 returns to its conductive stage, and the lamp remains extinguished.

All the binary stages in the complete circuit are identical and are intercoupled by diodes to avoid "back triggering" effects between adjacent stages. Stages read, from left to right; switch *S*1 setting 2^0 or 1; switch *S*2 setting 2^1 or 2; and switch *S*3 setting 2^{n-1}, where n equals the number of stages actually used. Thus, for a three-stage circuit, switch *S*3 sets $2^{3-1} = 4$. It is recommended, however, that a minimum of *four* stages be constructed (giving a total count of 15), and as many more as is economically practicable. Five stages will give a total count of 31; six stages a total count of 63; and seven stages a total count of 127 (*see also* Table 2, Chapter 2).

As regards construction, component layout is completely non-critical but should obviously follow the theoretical circuit as closely as possible for grouping, i.e. be laid out in adjacent stages. It is adaptable to printed circuit construction or Veroboard assembly, or tagboard assembly on a printed circuit panel (*see* Chapter 12).

Power requirements are quite moderate and can be met by 6-volt dry batteries. Current requirements per stage are approximately 40 μA at + 6 volts; and 65mA at −6 volts. Alternatively, a centre tapped 12-volt d.c. transformer/rectifier can be used connected to mains supply.

Although the working of the complete counter should be fairly obvious, once the principles of binary arithmetic have been grasped, the following notes on the method of use may be helpful.

Addition

Set the add/subtract switch to "add" and operate the set zero switch to extinguish all lights.

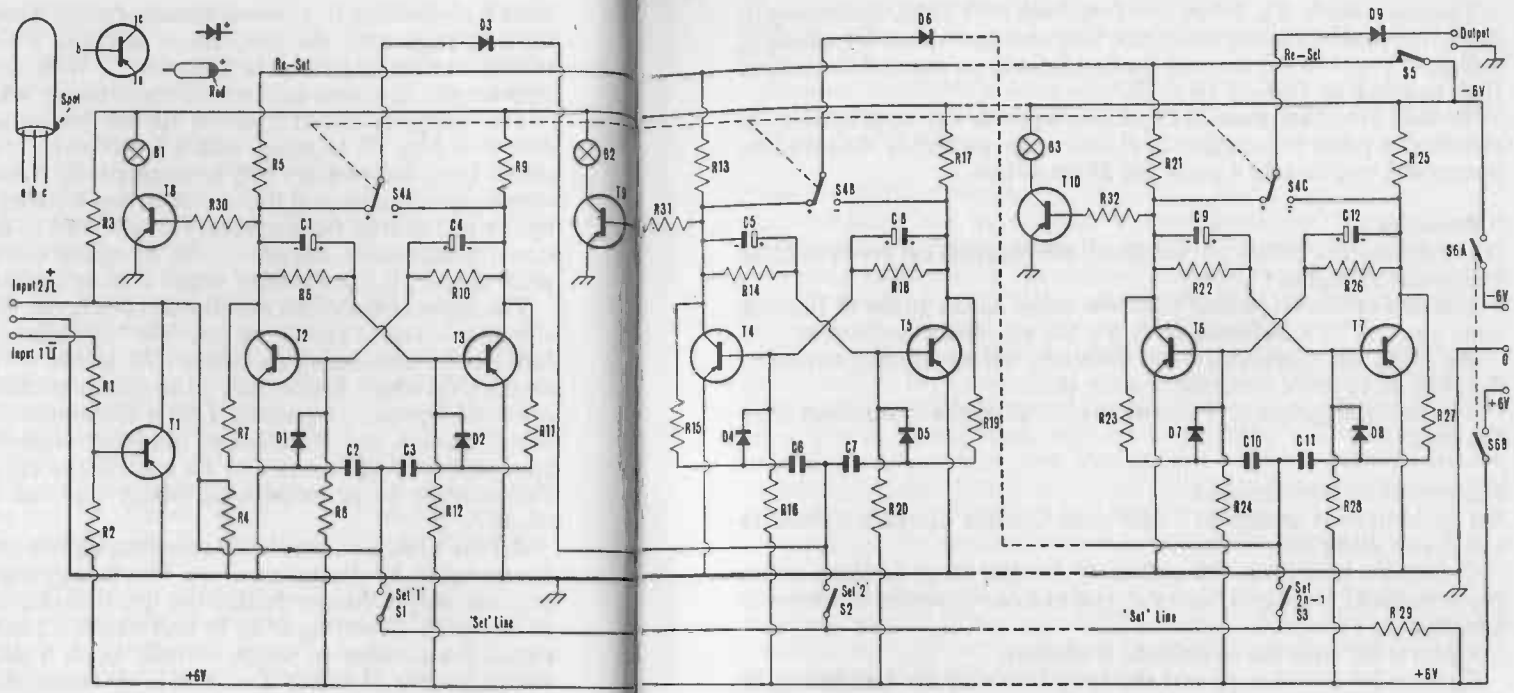


FIG. 13. COMPLETE CIRCUIT DIAGRAM FOR MULLARD ADDER/SUBTRACTOR

Resistors

*R*1	15kΩ	R8, 16, 24 etc.	220kΩ
*R*2	220kΩ	R9, 17, 25 etc.	1kΩ
*R*3	2·2kΩ	R10, 18, 26 etc.	4·7kΩ
*R*4	330kΩ	R11, 19, 27 etc.	2·2kΩ
*R*5, 13, 21 etc.	1kΩ	*R*12, 20, 28 etc.	220kΩ
*R*6, 14, 22 etc.	4·7kΩ	*R*29	1kΩ
*R*7, 15, 23 etc.	2·2kΩ	*R*30, 31, 32 etc.	6·8kΩ

All resistors ¼ watt, 10% tolerance.

Capacitors

*C*1, 5, 9 etc.	6·4μF	25V wkg. electrolytic
*C*2, 6, 10 etc.	0·22μF	
*C*3, 7, 11 etc.	0·22μF	
*C*4, 8, 12 etc.	6·4μF	25V wkg. electrolytic

Transistors, Diodes and Bulbs

*T*1	Mullard OC71 transistor
*T*2, 4, 6 etc.	Mullard OC71 transistor
*T*3, 5, 7 etc.	Mullard OC71 transistor
*T*8, 9, 10 etc.	Mullard OC72 or OC83 transistor
*D*1, 4, 7 etc.	Mullard OA85 or OA81 diode
*D*2, 5, 8 etc.	Mullard OA85 or OA81 diode
*D*3, 6, 9 etc.	Mullard OA85 or OA81 diode
*B*1, 2, 3 etc.	6V, 0·05A bulb M.E.S. fitting

Switches

*S*1, 2, 3 etc.	Two-way, single-pole toggle switch *biased to break position*
*S*4, *A*, *B*, *C* etc.	Two-way, multi-pole rotary switch
*S*5	Two-way, single-pole toggle switch *biased to make position*
*S*6, *A*, *B*	Two-way, two-pole on/off switch

Miscellaneous

4mm terminals, M.E.S. bulb holders.

Note: This components list is complete for three binary stages and is arranged so that the values of additional components required for additional stages are easily extracted.

Operate switch *S*1, when the first bulb will light, indicating 1. Repeat to add in any further digits. The next operation, for example, will yield 1 + 1 = 10, causing the first light to go out and the second light to come on (binary 10 = 2).

To add numbers instead of digits, operate the appropriate set switches to pulse the number of digits corresponding to the numbers concerned, e.g. to add 4 press the 2^2 set switch.

Subtraction

Set add/subtract switch to "subtract" and operate set zero switch to extinguish all lights.

Operate switch *S*1, which will now cause *all* the bulbs to light up since the unit now indicates 0—1, i.e. the negative complement.

By using the appropriate set switches, extinguish all unwanted lights so as to leave a number on the unit.

Use the appropriate set switches to subtract digits or numbers from the unit.

Subtraction by complements

Set add/subtract switch to "add" and operate set zero switch to extinguish all lights.

Connect a lead from the output of the *last* stage to the positive input terminal to act as a "carry 1" (i.e. end carry-over as described in Chapter 2).

Operate set switches to indicate a number.

Operate set switches to add the *complement* of the number to be subtracted. The unit will then show the difference between the two numbers.

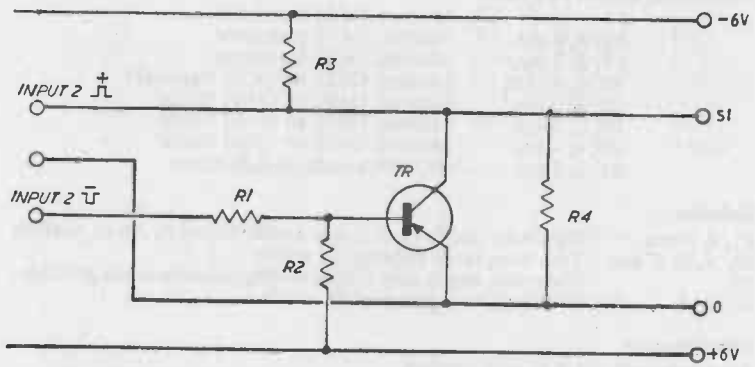


FIG. 14. PULSE SHAPER CIRCUIT FOR BINARY ADDER/SUBTRACTOR

*R*1	15kΩ	*TR*	OC71
*R*2	220kΩ		
*R*3	2·2kΩ		
*R*4	330kΩ		

Multiplication and Division

Multiplication is carried out by repetitive addition, e.g. 3 × 4 = 3 + 3 + 3 + 3.

Division is carried out by repetitive subtraction, e.g. 11 ÷ 3 = 11 − 3, − 3, −3 = three times, or 3. In this case there is a remainder of 2 or, specifically, 2/3.

Extension of Input Facilities

The computer can be adapted to read inputs other than simple manual pulses applied via the set switches. However, in order to get consistent results from the various possible forms of positive and negative waveforms of various shapes which might be required to be interpreted as "input," a *pulse shaping circuit* is required.

This is quite an elementary addition, a suitable circuit for which is shown in Fig. 14. It comprises a single transistor positively biased to cut off via a resistor connected to the +6 volt supply. Negative pulses up to 6 volts in amplitude, applied to the base of the transistor via another resistor, cause the transistor to conduct the pulse. On becoming non-conductive again, the transistor produces the required positive pulse for the first binary stage.

In general, for most applications, various shaped waveform inputs can be applied across input 2 terminals, the required switching action will then be produced when the waveform changes from negative to positive. Positive pulses with sharp rise times (e.g. square waves) and a maximum amplitude not greater than 6 volts can be applied directly to the input of the first binary stage.

This binary counter could be used as the arithmetic unit in a highly advanced computer design by incorporating a suitable number of stages to accommodate the range of numbers likely to be required and by relating them to suitable read-in, read-out and storage units. For reasonably straightforward amateur construction, however, the binary adder/subtractor represents a complete unit on its own, with read-out obtained from the indicator lamps. This will, of course, involve "translating" the binary number answer into decimal numbers.

For straightforward working, input is manual (i.e. manual operation of the appropriate set switches that feed the binary numbers representing the sum into the appropriate circuits). However, operation can be speeded by using a telephone dial for the input coder, as described in Chapter 8. The use of such a device automatically renders decimal numbers in digital (pulse) form. Equally, with the addition of the pulse shaper, any form of waveform input can be applied to the circuit, offering considerable scope for ingenuity in rendering the data to be dealt with arithmetically in pulse or waveform. For direct counting of high-frequency pulses, as opposed to arithmetic solutions, however, the decade scaler described in Chapter 4 would be a more logical form of computer.

CHAPTER 4

DECADE AND OTHER COUNTERS

BINARY counters can, if required, be adapted to count in the conventional decimal system, such devices normally being termed *decade counters*. In the case of a binary counter, as described in Chapter 3, the device is fed with digits or binary numbers and the answer is also given as a binary number. Thus if the problem is originally presented in decimal numbers these must be converted into their equivalent binary numbers before being fed into the counter; also the answer given by the counter must be converted from the binary number arrived at into its decimal equivalent. This "conversion" is done automatically in the decade counter. It still works in terms of binary arithmetic, but suitable interconnection of the appropriate stages provides *preset*. This means that the counter will always start counting at the preset number instead of "1" and continue to the rest of the available count when it will give an output to the next stage (and at the same time revert to its preset figure).

TABLE 4: BINARY COUNTERS

DECIMAL	PRESET	STAGE 1	STAGE 2	STAGE 3	STAGE 4
0		0	0	0	0
1		1	0	0	0
2		0	1	0	0
3		1	1	0	0
4		0	0	1	0
5		1	0	1	0
6	↓	0	1	1	0
7	1	1	1	1	0
8	2	0	0	0	1
9	3	1	0	0	1
10	4	0	1	0	1
11	5	1	1	0	1
12	6	0	0	1	1
13	7	1	0	1	1
14	8	0	1	1	1
15	9	1	1	1	1
16	10	0	0	0	0

TABLE 5: FOUR-STAGE COUNTER PRESETS AND COUNTS

DECIMAL	PRESETS															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0															
1	1	0														
2	2	1	0													
3	3	2	1	0												
4	4	3	2	1	0											
5	5	4	3	2	1	0										
6	6	5	4	3	2	1	0									
7	7	6	5	4	3	2	1	0								
8	8	7	6	5	4	3	2	1	0							
9	9	8	7	6	5	4	3	2	1	0						
10	10	9	8	7	6	5	4	3	2	1	0					
11	11	10	9	8	7	6	5	4	3	2	1	0				
12	12	11	10	9	8	7	6	5	4	3	2	1	0			
13	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
14	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
15	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
COUNT	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

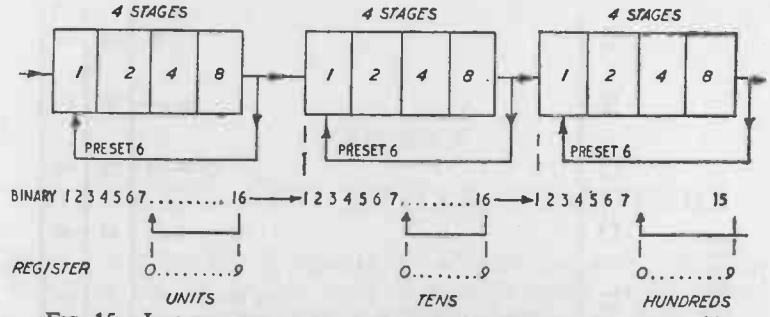


FIG. 15. ILLUSTRATING THE PRINCIPLE OF PRESET FOR COUNTING BY 10s

In practice there are various ways of feeding in the preset. For example, the output at the required count of 10 could be arranged to trigger-in 6 pulses at the same time, to establish the denary zero (binary 110) ready for the next input/pulse. Equally, the multistage binary counter could incorporate adjustable preset to read out after any number between 1 and the full count number. Thus, a four-stage binary counter could be preset to read out at *any* number up to 16. *See* Table 5.

This method of preset is an extremely useful function, for it means that the basic binary counter can be adapted to deal with almost any units and combinations of units. Thus, the decade counter (preset to 6 in a four-stage unit) will deal with decimal counts up to 10, with extension of the 10s count to 100s, and so on, by additional blocks

TABLE 6: BINARY COUNTERS

DECIMAL	PRESET			STAGE 1	STAGE 2	STAGE 3	STAGE 4
	Base	*Base*	*Base*				
0	4	6	8	0	0	0	0
1				1	0	0	0
2				0	1	0	0
3				1	1	0	0
4				0	0	1	0
5				1	0	1	0
6				0	1	1	0
7				1	1	1	0
8			↓	0	0	0	1
9			1	1	0	0	1
10		↓	2	0	1	0	1
11		1	3	1	1	0	1
12	↓	2	4	0	0	1	1
13	1	3	5	1	0	1	1
14	2	4	6	0	1	1	1
15	3	5	7	1	1	1	1
16	4	6	8	0	0	0	0

(Fig. 15). By adjusting the preset of the individual blocks the computer can count and read out in different units. Table 6 shows the condition in a four-stage binary counter for reading in numbering systems to base 4, base 6 and base 8.

Fig. 16 shows in block diagram form the arrangement required for preset counters reading directly in mixed units—pounds, shillings and pence. Exactly the same principle can be extended in a variety of other ways. All the working, however, is still accomplished by binary stages, and the binary counter described in Chapter 3 could be adapted directly for any such systems. Each "block," in effect, becomes a separate binary counter with its own individual preset.

It will be appreciated, however, that this can become wasteful. For example, for counting shillings and giving outputs in pounds, a five-stage binary counter is required, and the preset of 12 means that only twenty-four thirty-seconds, or 75 per cent, of the full counting capability of the complete counter is being utilized. Nevertheless, the advantage of having a direct readout in the units required can often be far greater than the theoretical advantage of full working with binary numbers.

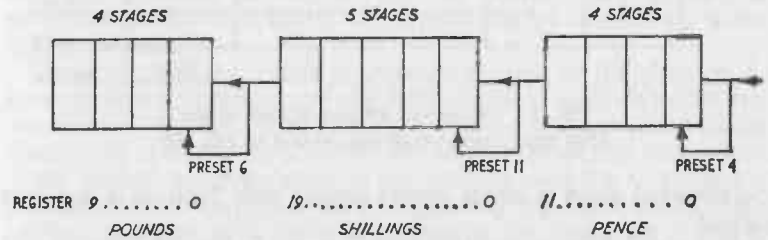


FIG. 16. PRESET FOR COUNTING IN £ s d

The decade counter, in particular, is an extremely useful device which can also be designed on somewhat different lines as a *scaler* with an extremely rapid counting rate. The circuit description which follows is, again, a Mullard design suitable for amateur construction, and it can be extended to any number of stages required. The cost of components for a four-stage unit with a total count of 10^7 (10 million) should not be more than £25. Construction of the power supply and first stage only can be completed for very much less, and further stages can be added later, if required.

The Mullard scaler has a maximum count *rate* of 400 cycles per second (c/s) and will accept and count most standard waveforms. A visual indication of the count is given by a standard electromechanical register driven from the final output stage. Power supplies are derived from a single 12 volt d.c. source, e.g. a battery or mains transformer/rectifier. The scaler actually requires four different supply voltages: + 500 V, + 45 V, −9·1 V and −64 V, but these can

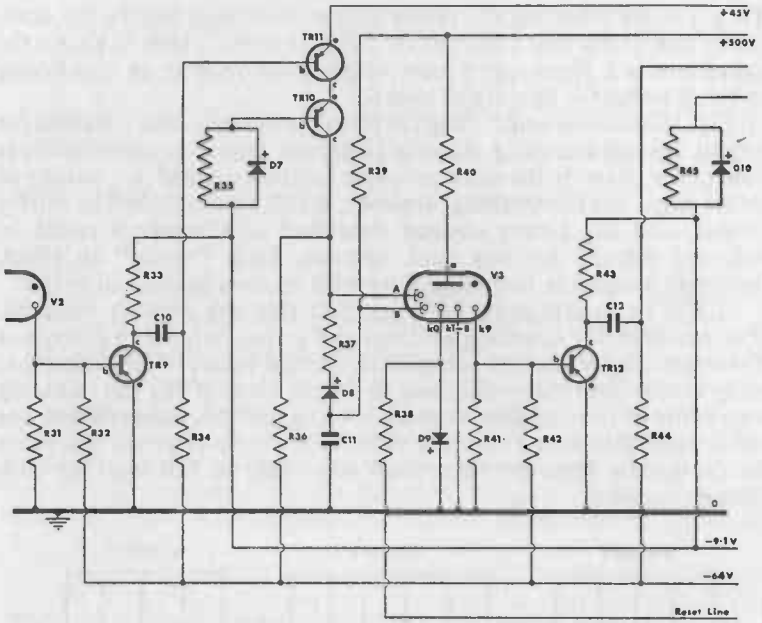


FIG. 17. MULLARD DECADE SCALER STAGE
(For component values see caption to Fig. 20)

be obtained from a single power supply unit built as a separate circuit.

One stage of the complete circuit is shown in Fig. 17, where it will be noticed that a special counter tube is employed. This, in its essentials, comprises thirty identical rod-shaped cathodes arranged in a circle concentric with a common circular-plate anode. The cathodes are divided into three groups of ten, so connected that every third electrode belongs to the same group. These groups (k_0, k_1, etc., up to k_{10}) comprise the main cathodes, guide *A* cathodes and guide *B* cathodes. All the guide *A* and guide *B* cathodes are connected internally and brought out, in each case, as a single connexion on the valve.

The glow discharge, visible as an orange light, normally rests on a main cathode, and its position on the ring of cathodes is determined by the number of input pulses applied to the tube. In other words, the position of the glow corresponds to 0 to 9 (or 1 to 10), with reference to the number of pulses applied to the circuit, giving a visual indication of the count.

The glow discharge is transferred from one (main) cathode rod to the next, through the guide cathodes. An input signal applied to the

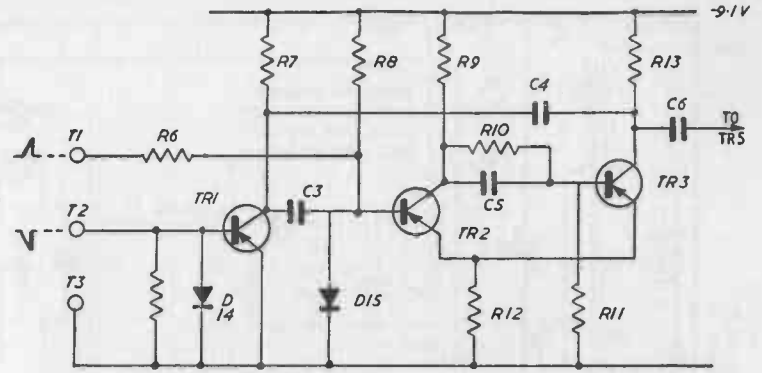


FIG. 18. INPUT CIRCUIT FOR DECADE SCALER
(For component values see caption to Fig. 20)

base of *TR*9 is amplified, shaped, and then converted into two negative pulses by the circuit. The first pulse is applied to guide *A* cathodes, followed immediately by the second pulse applied to guide *B* cathodes. This causes the glow discharge to move on to the next main cathode.

The main cathode circuit is arranged so that as the glow on k_9 is extinguished a negative pulse is fed from this electrode to the first transistor of the next stage (*TR*12). Cathode k_0 is connected to the other main cathodes via a diode (*D*9) so that, as well as taking part in the normal counting cycle, it can also be used for resetting the discharge to the zero position by operating the reset switch. This applies a negative pulse direct to k_0, causing the discharges to step to the zero position.

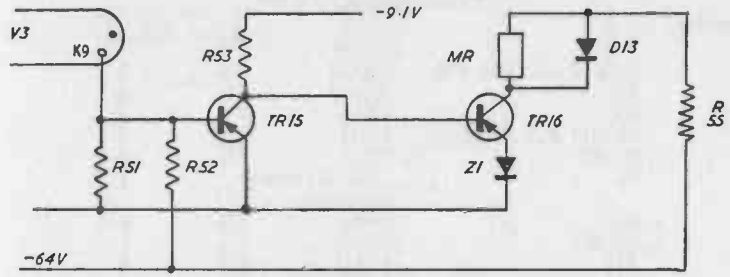


FIG. 19. OUTPUT CIRCUIT FOR DECADE SCALER WITH ELECTRO-MAGNETIC REGISTER

*R*51	33kΩ	*TR*15	OC75
*R*52	330kΩ	*TR*16	ACY17
*R*53	5·6kΩ	*D*13	OA81
*R*55	3kΩ *minus MR* resistance	*Z*1	OAZ200

MR High resistance electro-magnetic register 2–3 kΩ resistance

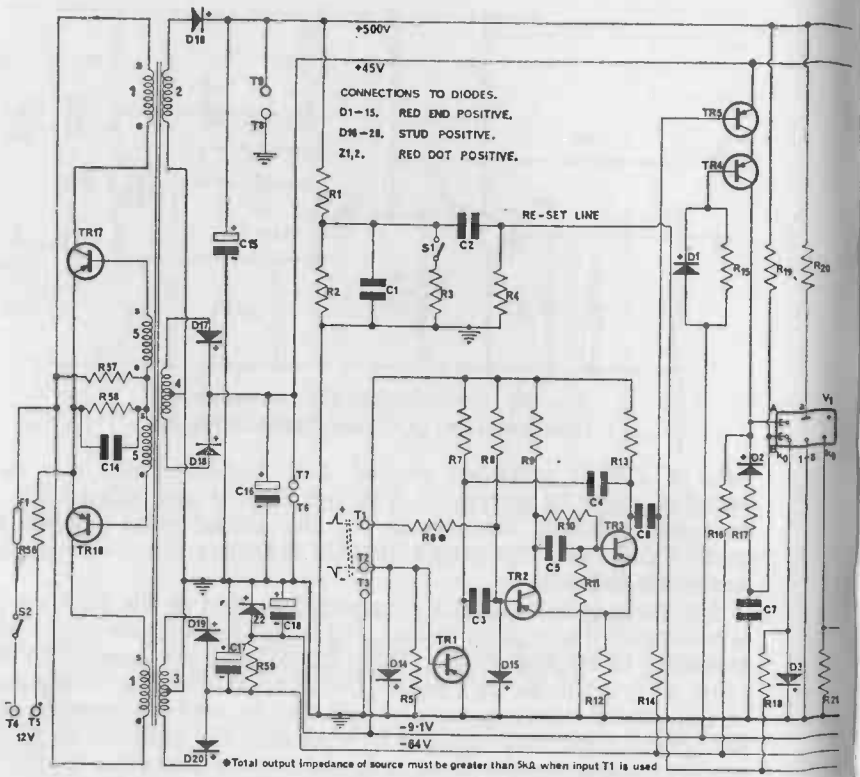
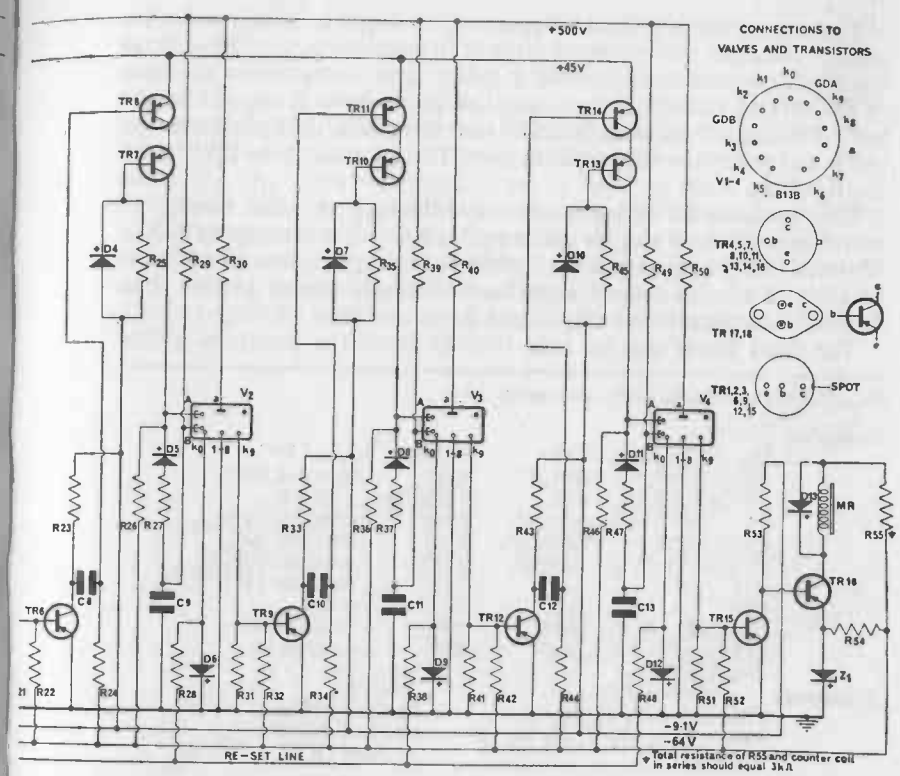


FIG. 20. COMPLETE CIRCUIT FOR DECADE SCALER

COMPONENTS LIST

Resistors

No.	*Value*	*Tol.*	*Watts*
R1 R20 R30 R40 R50	820kΩ	7%	¼
R2	470kΩ	7%	¼
R3	100Ω	7%	¼
R4 R17 R27 R37 R47	47kΩ	7%	¼
R5 R7	12kΩ	7%	¼
R6	*R6* plus source impedance > 5kΩ		¼
R8	10kΩ	7%	¼
R9 R13	1·5kΩ	7%	¼
R10	4·7kΩ	7%	¼
R11	6·8kΩ	7%	¼
R12	270Ω	7%	¼
R14 R24 R34 R44	270kΩ	7%	¼
R15 R25 R35 R45	120kΩ	10%	¼
R16 R26 R36 R46	9·1kΩ	7%	1
R18 R28 R38 R48	150kΩ	10%	¼
R19 R29 R39 R49	10MΩ	10%	½



Resistors

No.	*Value*	*Tol.*	*Watts*
R21 R31 R41 R51	33kΩ	7%	¼
R22 R32 R42 R52	330kΩ	5%	¼
R23	2·2kΩ	7%	¼
R33 R43 R53	5·6kΩ	7%	¼
R54	47kΩ	7%	½
R55	To total 3kΩ with register *MR*		¼
R56	2 × 5·6Ω in parallel	5%	6 w.w.
R57	1·5kΩ	5%	½
R58	20Ω	5%	2 w.w.
R59	2·2kΩ	5%	2 w.w.

w.w. indicates wire wound

Capacitors

No.	*Value*	*Voltage*	*Type*
C1	1μF	250V	paper
C2	0·22μF	125V	paper
C3	0·47μF	125V	moulded 10%

Caption continued overleaf

The input circuit is shown separately in Fig. 18. This is a monostable vibrator circuit employing three transistors. The circuit remains quiescent until a positive going wave is applied to the base of the second transistor, or a negative going wave is applied to the base of the first transistor. In either case the circuit then produces one pulse and returns to its quiescent state. Output is taken to *TR*5 in the main circuit.

This monostable vibrator circuit will accept a wide variety of waveforms of most shapes and amplitudes within a range of 0·5 to 10 volts, but the input resistor for the positive-going waveform must be selected to give a total impedance of signal source greater than 5kΩ (i.e. *R*6 plus source impedance is greater than 5*k*Ω).

The total count can be read directly from the positions of the

FIG 20. (Components List)—*continued*

Capacitors

No.	*Value*	*Voltage*	*Type*
*C*4	0·01μF	125V	moulded 10%
*C*5	470pF	125V	mica 5%
*C*6	0·022μF	125V	moulded 10%
*C*8 *C*10 *C*12	0·018μF	125V	moulded 10%
*C*7 *C*9 *C*11 *C*13	1200pF	125V	moulded 10%
*C*14	1μF	125V	moulded 10%
*C*15	1μF	500V	paper
*C*16 *C*17	64μF	64V	electrolytic
*C*18	32μF	64V	electrolytic

Transistors

*TR*1, 2, 3, 6, 9, 12, 15	Mullard OC75
*TR*4, 5, 7, 8, 10, 11, 13, 14, 16	Mullard ACY17
*TR*17, 18	Mullard OC35

Diodes

*D*1, 2, 4 5, 7, 8, 10, 11, 13, 14, 15	Mullard OA81
*D*3, 6, 9, 12	Mullard OA202
*D*16	Mullard OA211 or BY100
*D*17, 18, 19, 20	Mullard OA210

Zener Diodes

*Z*1	Mullard OAZ200
*Z*2	Mullard OAZ207

Counting Tubes

*V*1-4	Mullard Z504S
Valve bases	4 × B13B

*S*1 Single-pole, spring-loaded, press to make toggle switch
*S*2 Single-pole, on/off switch
MR High resistance electromagnetic register—resistance 2–3kΩ

Transformer Core—2 Mullard Ferroxcube E cores type FX1819
*T*1–5 Screw down terminal with 4 mm sockets
*T*6–9 4mm shrouded sockets
*F*1 2·5A fuse and holder

glow discharges on the counter tubes, but electro-mechanical counting can be coupled to the output if preferred. A suitable output circuit is shown in Fig. 19. This utilizes the positive pulses from cathode k_9 of the counting tube, which are applied to the base of a transistor (*TR*15), which then cuts off. This results in a negative pulse being applied to transistor *TR*16, which conducts a large current, energizing the electro-magnetic register *MR*. A zener diode *Z*1 is included in this circuit to provide additional bias to transistor *TR*16.

The complete circuit for a four-stage decade scaler is shown in Fig. 20. This employs four counting tubes (each stage or counter tube circuit being identical) and electromagnetic readout. The electromagnetic resister needs to be of high resistance type (2,000 to 3,000 Ω).

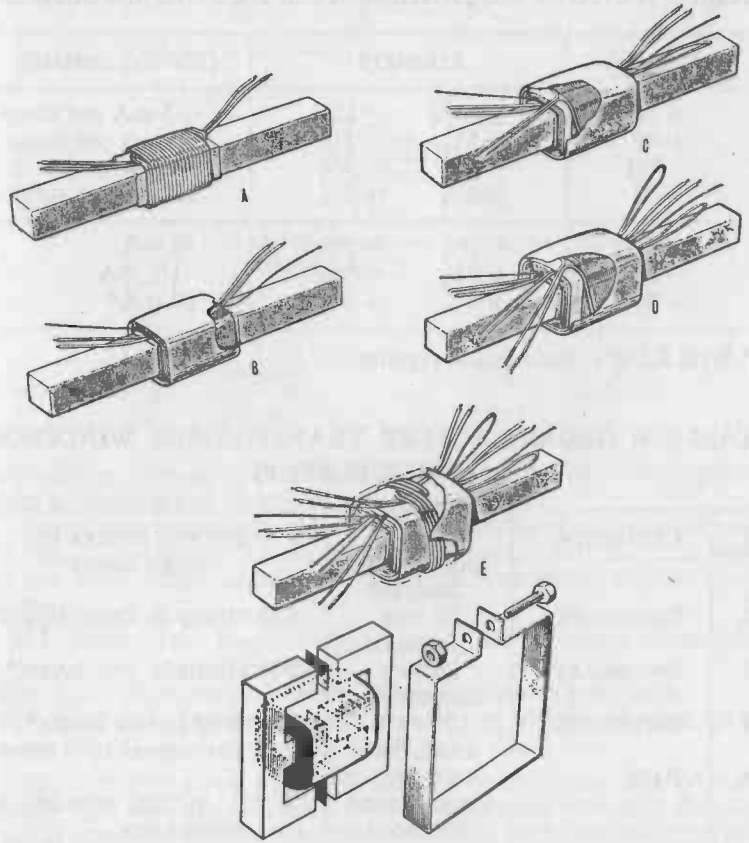


FIG. 21. TRANSFORMER CONSTRUCTIONAL DETAILS FOR DECADE SCALER POWER SUPPLY

Power requirements, summarized in Table 7, are best provided by a special design of transformer, details of which are summarized in Fig. 21 and Table 8. The windings are made over a half-inch square mandrel, over an initial wrapping of thin card, and comprise five separate windings laid one on top of the other. It is important that these windings should not be made so tight that the finished windings cannot be withdrawn from the mandrel.

The complete bobbin so formed is then mounted on Mullard ferroxcube E cores type FX1819, entered from each side of the bobbin. Between the two cores a small air space is required. This can be provided by inserting a piece of thin card between the faces when

TABLE 7: POWER REQUIREMENTS FOR DECADE SCALER

VOLTAGE	STABILITY		CURRENT DEMAND
+500	+10%	−15%	0·5 mA per stage
+45	+5%	−5%	12 mA per stage
−9·1	+5%	−5%	5 mA per stage
−64	+5%	−5%	12 mA per stage
−9·1	+5%	−5%	3 mA
−9·1	+5%	−5%	1·5 mA
−64	+5%	−5%	10 mA*

* With 2,500 Ω mechanical register.

TABLE 8: DECADE SCALER TRANSFORMER WINDINGS (POWER SUPPLY)

1	COLLECTOR	24 swg single strand insulated	2 + 9 TURNS BIFILAR in single layers
2	SECONDARY	36 swg enamelled	520 TURNS in three layers*
3	SECONDARY	24 swg enamelled	140 TURNS in two layers*
4	SECONDARY	24 swg enamelled	96 TURNS in two layers* centre tapped at 48 turns
5	BASE	24 swg single strand insulated	2 × 2 TURNS BIFILAR in single layer

* With paper interlay.

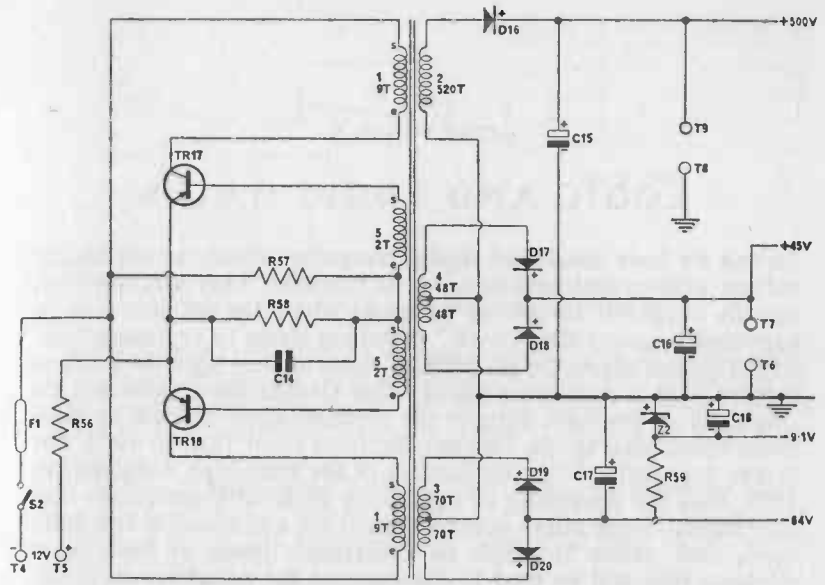


FIG. 22. TRANSFORMER CIRCUIT FOR DECADE SCALER

R56	2 × 5·6Ω (paralleled)	*F1*	2·5A fuse
R57	1·5kΩ	*D16*	OA211
R58	20Ω	*D17*	OA210
R59	2·2kΩ	*D18*	OA210
C14	1μF 125V	*D19*	OA210
C15	1μF 500V	*D20*	OA210
C16	64μF 64V	*Z2*	OAZ207
C17	64μF 64V	*TR17*	OC35
		TR18	OC35

assembling. The complete transformer can then be held together with a simple metal clamp.

The transformer is then wired into a d.c. converter circuit, as shown in Fig. 22. All the semiconductors in this circuit, with the exception of the zener diode, require mounting on heat sinks, suitable sizes being—

TR17, *TR18* 4-in. length of 1½ in. × 1½ in. L-shaped aluminium angle ⅛ in. thick.

D16 ¾ in. diameter or ¾ in. square of aluminium ⅛ in. thick.

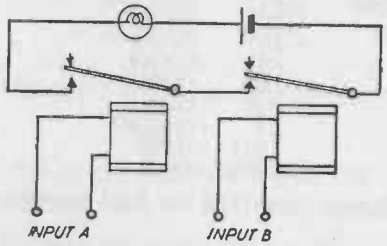
D17, *D18*, *D19*, *D20* ⅝ in. diameter or ⅝ in. square of aluminium 1/16 in. thick.

The complete power supply circuit can be mounted on a Paxolin panel, with the heat sinks and other components mounted directly on the panel. Connexions to the main circuit are obvious from the circuit markings, but the power supply is also shown complete and properly connected in the main circuit diagram of Fig. 20.

CHAPTER 5

LOGIC AND LOGIC GATES

So far we have considered digital computers simply as devices for solving arithmetical problems, i.e. as counters. They are, however, equally adaptable to solving problems where the solution can be extracted by using the "on-off" switching states to represent "yes-no." This introduces the principle of mathematical logic, or Boolean algebra as it is sometimes called (after George Boole who was an originator of symbolic logic or the representation of logic by algebraic equations). Boole, incidentally, lived from 1815 to 1864, but it was not until the first conception of the relay-type computer, in 1938, that the possibility of combining logic with mechanics was appreciated. Since relays operate directly on a mechanical two-state basis, their action is simple to understand; hence as basic logic elements they will be used to demonstrate the principles involved.



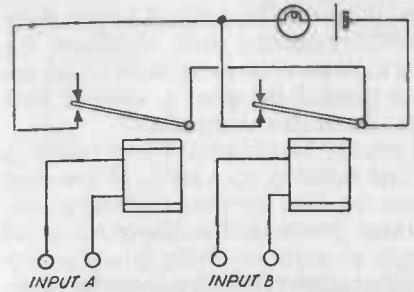
INPUT A	INPUT B	OUTPUT
0	0	0
0	1	0
1	0	0
1	1	1

TRUTH TABLE

FIG 23. SERIES "LOGIC" CIRCUIT AND TRUTH TABLE

Switches could be used equally, since the working of a basic logic circuit, or *gate* as it is called, depends entirely on whether the circuit is complete, ON, or broken, OFF.

Consider the series circuit in Fig. 23, which is shown both as a mechanical circuit and as a statement of possible conditions, or *truth table* as it is termed, using the standard computer designation of 1 for ON and 0 for OFF. The circuit is completed and thus there is an output only when both contacts are closed, i.e. both relays are energized by an input signal. In simple language this can be expressed as output given when there are inputs to relay 1 AND relay 2. Thus, the circuit is that of an AND gate.



INPUT A	INPUT B	OUTPUT
0	0	0
0	1	1
1	0	1
1	1	1

TRUTH TABLE

FIG. 24. PARALLEL "LOGIC" CIRCUIT AND TRUTH TABLE

Now consider the parallel circuit shown in Fig. 24, represented both by the physical circuit and its equivalent truth table. Again, there are four possible conditions, but an output is given (i.e. the circuit is complete) when there are inputs to relay *A* OR relay *B* OR both relays. This circuit, therefore, forms an OR gate.

In terms of generalized Boolean algebra or "mathematical logic" a series condition is represented by the symbol . and a parallel condition by +. On this basis the truth table may be written in another form.

In the series circuit or AND gate

Logic symbol	*Meaning*
$0 \cdot 0 = 0$	OFF in series with OFF = no output
$0 \cdot 1 = 0$	OFF in series with ON = no output
$1 \cdot 0 = 0$	ON in series with OFF = no output
$1 \cdot 1 = 1$	ON in series with ON = output ON.

The circuit OFF, or no output condition, can also be represented by introducing another symbol ¯ which means *negation* or opposite, sometimes referred to as "not." Thus, the circuit is completed by both (relays) *A* and *B* being energized (or both switches being closed if switches are considered instead of relays).

$A.\bar{A} = 0$, meaning that one set of contacts, *A*, in series (.) with its negation ($\bar{A}$) gives no output.

In the parallel circuit or OR gate we have the following—

Logic symbol	*Meaning*
$0 + 0 = 0$	OFF in parallel with OFF = no output
$0 + 1 = 1$	OFF in parallel with ON = output ON
$1 + 0 = 1$	ON in parallel with OFF = output ON
$1 + 1 = 1$	ON in parallel with ON = output ON
Generally $A + \bar{A} = 1$	ON in parallel with its negation = output ON.

It is important not to confuse logic symbols with ordinary arithmetic. Thus, the sign . does not mean "multiply" but logic AND

(with the implication that all inputs have to be fulfilled before there is an output or completed circuit through the gate). Similarly, the sign + does not mean "add" but logic OR (one or more or all inputs together) will complete the circuit through the gate. A series of such gates can be used for routing data through a computer.

An elementary computer can readily be designed using relays as gate elements (or just switches) and building up a series of AND and OR gates to sort out the logic from the data involved and to provide an answer to a particular problem involved. For the purpose of demonstration we will take a simple example involving qualifications required for a job, and render the computer, which solves the problem of whether a candidate is suitable or not, in the form of switches only.

Qualifications required: 18–20 years old
AND
6 "O" levels OR 3 "O" levels and 1 "A" level
AND
must have maths at "O" level.

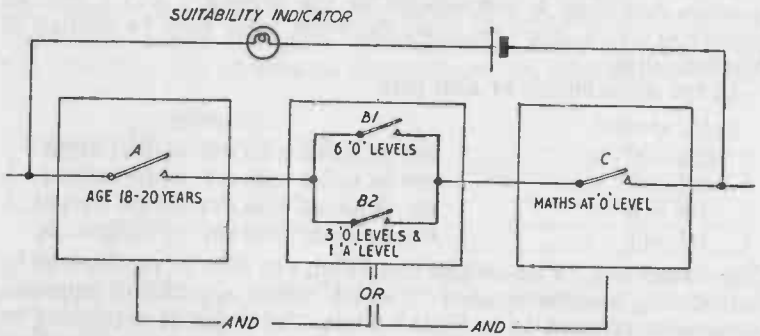


FIG. 25. ANALYSIS OF JOB SUITABILITY BY ELEMENTARY LOGIC COMPUTER SWITCHING

The corresponding "computer" circuit is shown in Fig. 25. If the first qualification (age) is met, switch *A* is closed. If one or other of the academic qualifications is met, either switch *B*1 or *B*2 is closed. Finally, if the candidate has also passed "O" level maths, switch *C* is closed. The circuit is then completed and the indicator lamp lights to show that the candidate has the necessary qualifications.

So far as logic elements are concerned, *B*1 and *B*2 form an OR gate. *A* and *B*1/*B*2 together form an AND gate. *B*1/*B*2 and *C* together also form an AND gate. Thus, the circuit involves two AND and one OR gates, consistent with the "ands" and "ors" in the list of qualifications.

This is, of course, only a very simple example of applied computer

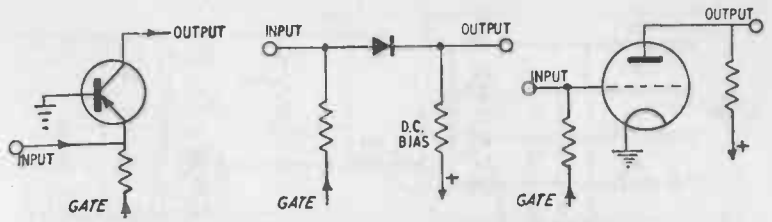


FIG. 26. TRANSISTOR, DIODE AND TRIODE AS SWITCHES

logic, but it does illustrate the principles of AND and OR gating or switching. It could be extended or modified to cope with other sorts of problems of a similar nature involving a required combination of "ands" and "ors," still with very elementary circuitry. It is a very slow computer, however, for each switch has to be operated manually (where appropriate). Relays would be much faster since the inputs could be fed by pulses (pulse for "yes," no pulse for "no"). Still greater speed, and far greater compactness, can be realized by using electronic elements instead of switches, particularly transistors, which have the advantage of requiring lower power for operating the computer and, besides acting as "on-off" switches, can also amplify the currents they pass.

Note, too, that a gate performs only a simple "on-off" switching function—like switch *A* and switch *C* in Fig. 25 for example. Two such simple gates can then perform AND or OR logic function if connected in series or parallel, respectively. There are also several other modes of interconnexion or gate circuit design capable of providing different logic.

Fig. 26(*a*) shows a transistor connected in grounded base configuration to perform the function of a simple gate. The collector is negative, and when the emitter voltage rises to slightly above zero the transistor conducts. Thus, a positive gate voltage applied to the emitter then allows the transistor to pass a signal. The transistor is working as a simple "on-off" or "stop-go" switch. A similar simple switching function can also be provided by a diode, as shown in Fig. 26(*b*), or a triode, as shown in Fig. 26(*c*). Equally, any one of these basic circuit elements, transistor, diode, or pentode, can be used in AND or OR gate circuits. For the further development of logic gates, however, we will work with transistors.

Consider the circuit shown in Fig. 27. This is an AND gate by virtue of the fact that the output circuit is completed (i.e. an output signal produced) if all the inputs are signalled, and *only* if all are signalled. Component values are given for a practical working circuit, four inputs being shown. Any number of inputs can be incorporated, from two upwards, merely by adding to the diode-resistor circuits.

The circuit works as follows. In the absence of any input at *A*, *B*, *C*, etc., the transistor is biased to cut-off, and the collector

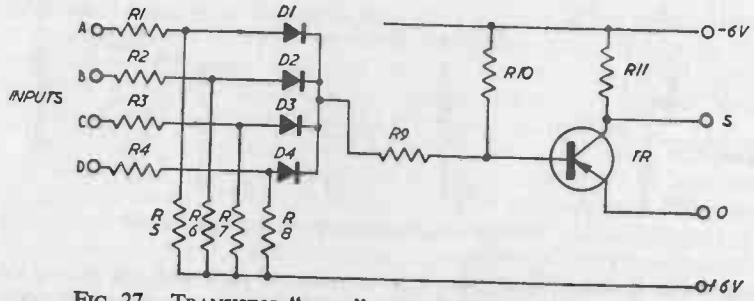


FIG. 27. TRANSISTOR "NAND" GATE CIRCUIT WITH FOUR INPUTS

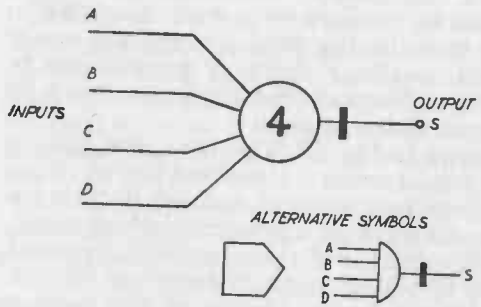
*R*1	15kΩ	*R*7	47kΩ
*R*2	15kΩ	*R*8	47kΩ
*R*3	15kΩ	*R*9	1·5kΩ
*R*4	15kΩ	*R*10	100kΩ
*R*5	47kΩ	*R*11	2·2kΩ
*R*6	47kΩ	*TR*	OC71

output is −6 volts with respect to earth. This bias can be removed only if *all* the inputs are connected to a potential of −6 volts with respect to earth, when the transistor conducts to provide an output signal.

However, this output is in anti-phase to the input, and in terms of logic symbols this is represented by the equation.

$$A.B.C.D. = \bar{S}$$

where S is the output and $\bar{}$ is the negater or anti-phase symbol. This condition is designated by a vertical negater bar on the symbolic drawing of the gate, Fig. 28, which diagram also includes the truth table for this particular gate with four inputs *A*, *B*, *C* and *D*. The number of inputs is noted in the gate symbol (circle).



INPUTS				S
A	B	C	D	
0	0	0	0	1
1	0	0	0	1
0	1	0	0	1
0	0	1	0	1
0	0	0	1	1
1	1	0	0	1
1	0	1	0	1
1	0	0	1	1
0	1	1	0	1
0	1	0	1	1
0	0	1	1	1
1	1	1	0	1
1	0	1	1	1
1	1	0	1	1
0	1	1	1	1
1	1	1	1	0

FIG. 28. "NAND" GATE SYMBOLS AND TRUTH TABLE FOR FOUR INPUTS

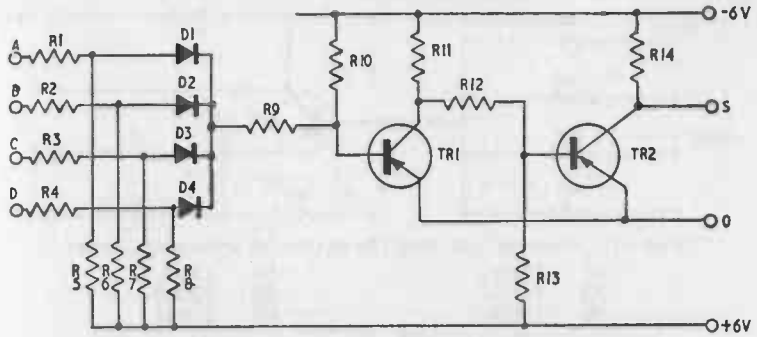


FIG. 29. TRUE "AND" GATE CIRCUIT WITH FOUR INPUTS

*R*1	15kΩ	*R*9	1·5kΩ
*R*2	15kΩ	*R*10	100kΩ
*R*3	15kΩ	*R*11	2·2kΩ
*R*4	15kΩ	*B*12	15kΩ
*R*5	47kΩ	*B*13	220kΩ
*R*6	47kΩ	*R*14	2·2kΩ
*R*7	47kΩ	*TR*1	OC71
*R*8	47kΩ	*TR*2	OC71

Strictly speaking, since the output is complementary to the input (i.e. in anti-phase to it), this is a NOT–AND gate or NAND gate.

A true AND gate circuit is shown in Fig. 29. Basically its performance is the same as a NAND gate, except that output and inputs vary in phase, the corresponding equation being

$$A.B.C.D. = S.$$

In physical conception it comprises a NAND gate in series with a single input NOR or NOT gate. The first gate (NAND gate) then gives

$$A.B.C.D. = \bar{S}_1.$$

The second (NOR) gate gives

$$S_1 = \bar{S} \text{ or } \bar{S}_1 = S_1$$

Whence by substitution

$$A.B.C.D. = S.$$

The symbol for the true AND gate is shown in its usual form in

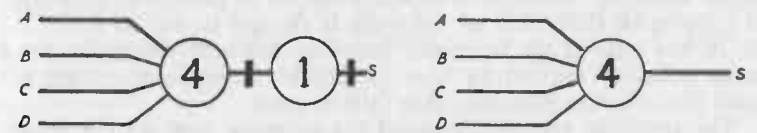


FIG. 30. "AND" GATE SYMBOLS

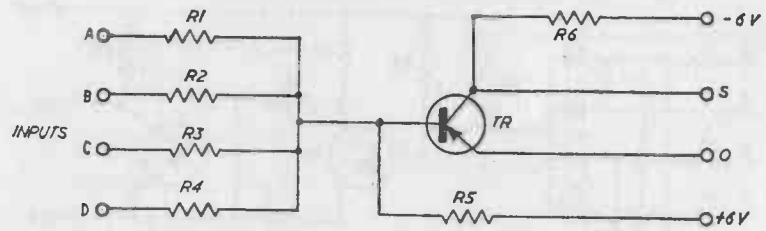
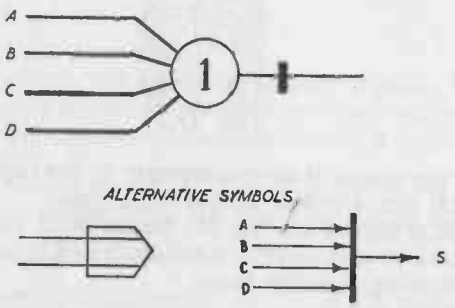


FIG. 31. "NOT-OR" OR "NOR" GATE CIRCUIT WITH FOUR INPUTS

*R*1	15kΩ	*R*4	15hΩ
*R*2	15kΩ	*R*5	220kΩ
*R*3	15kΩ	*R*6	2·2kΩ

TR OC71



INPUTS A	B	C	D	S
0	0	0	0	1
1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	0	1	1	0
1	0		1	0
1	1	0	1	0
1	1	1	0	0
0	1	1	1	0
1	1	1	1	0

FIG. 32. "NOR" GATE SYMBOLS AND TRUTH TABLE

Fig. 30, which also shows a combination of a NAND gate and a single input NOR gate.

OR gates, similarly, can have anti-phase or in-phase characteristics. Thus, Fig. 31 shows a practical circuit for a NOT–OR or NOR gate with two or more inputs, the output changing if any one input is signalled, but in anti-phase to changes in input. The corresponding equation is—

$$A + B + C + D \ldots N = \bar{S}.$$

In this case the potential difference at the collector of the transistor is −6 volts with respect to earth in the absence of any input signal since the transistor is biased to cut off by the +6 volts supply. If a potential difference of −6 volts is applied to any of $A,B,C \ldots$ or N, the base of the transistor becomes sufficiently negative for a large collector current to flow. If suitable component values are used the collector potential thus falls to zero.

The symbolic representation of the inclusive NOR gate is shown in Fig. 32 together with its truth table, which is shown for four

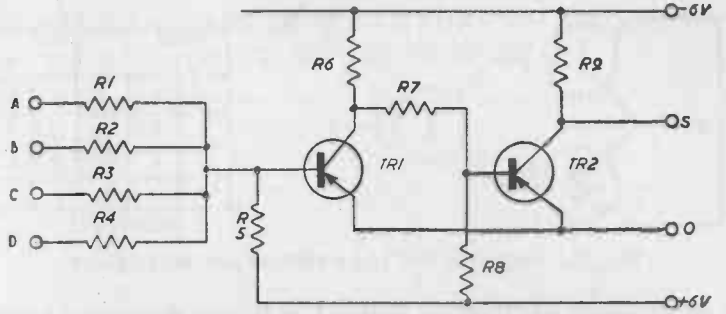


FIG. 33. INCLUSIVE "OR" GATE CIRCUIT WITH FOUR INPUTS

*R*1	15kΩ	*R*7	15kΩ
*R*2	15kΩ	*R*8	220 kΩ
*R*3	15kΩ	*R*9	2·2kΩ
*R*4	15kΩ	*TR*1	OC71
*R*5	220kΩ	*TR*2	OC71
*R*6	2·2kΩ		

inputs. The 1 in the circle denotes that only one input needs to be energized to provide an output. The negater sign indicates that the output is in anti-phase.

The complement of this is the inclusive OR gate, which is identical in function except that input and output are in phase. It is thus represented by the equation

$$A + B + C + D \ldots N = S.$$

The corresponding circuit is shown in Fig. 33, in which the OR gate comprises a multi-input NOR circuit in series with a single-input NOR circuit, giving the following—

$$\text{First gate: } A + B + C + D = \bar{S}_1$$
$$\text{Second gate: } S_1 = \bar{S} \text{ or } \bar{S}_1 = S$$

whence

$$A + B + C + D = S.$$

The two methods of representing the inclusive OR gate by symbols are shown in Fig. 34.

There is also an *exclusive* OR gate which is similar to the inclusive OR in that the output is produced by a signal to any one input; but

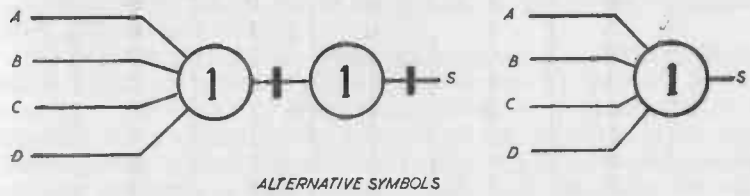
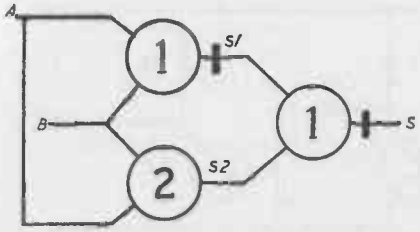


FIG. 34. ALTERNATIVE SYMBOLS FOR INCLUSIVE "OR" GATE



INPUT A	INPUT B	OUTPUT S1	OUTPUT S2	OUTPUT S
0	0	1	0	0
1	0	0	0	1
0	1	0	0	1
1	1	0	1	0

TRUTH TABLE

FIG. 35. EXCLUSIVE "OR" GATE SYMBOLS AND TRUTH TABLE

if *all* the inputs are identical (either 1 or 0) then the output returns to zero. This is shown, in symbolic form for a two-input gate, together with its truth table, in Fig. 35, in which it will be seen that basically it comprises an AND gate and two NOR gates. If inputs *A* and *B* are *both* energized the AND gate is also energized, which prevents or *inhibits* the final NOR gate from changing its output condition.

Although AND and OR are the main logic functions, the mixing or combination of the two in various ways can produce a variety of different gate performances. It is on this basis that logic circuits are

TABLE 9: TRUTH TABLE FOR "AND" GATES (FOUR INPUTS SHOWN)

INPUTS A	B	C	D	S
0	0	0	0	0
1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
1	1	0	0	0
1	0	1	0	0
1	0	0	1	0
0	1	1	0	0
0	1	0	1	0
0	0	1	1	0
1	1	1	0	0
1	0	1	1	0
1	1	0	1	0
0	1	1	1	0
1	1	1	1	1

TABLE 10: TRUTH TABLE FOR "OR" GATES (FOUR INPUTS SHOWN)

INPUTS A	B	C	D	OUTPUT S
0	0	0	0	0
1	0	0	0	1
0	1	0	0	1
0	0	1	0	1
0	0	0	1	1
1	1	0	0	1
1	0	1	0	1
1	0	0	1	1
0	1	1	0	1
0	1	0	1	1
0	0	1	1	1
1	1	1	0	1
1	0	1	1	1
1	1	0	1	1
0	1	1	1	1
1	1	1	1	1

developed to complete a computer circuit designed to accept a variety of data as inputs and to extract the logical answer as an output. This may involve arithmetical calculations or general logic.

Thus, the gate circuits described above can be employed in suitable combinations; for example a combination of gates could be used to analyse the same problem dealt with by Fig. 25—a combination of AND and OR logic. The resulting circuit would be much more complex than the original switching circuit designed to provide the same solution, but this is largely because the problem involved is basically a simple one; hence the comparison is only illustrative of the two methods. The electronic computer has far more scope for dealing rapidly with more complex problems, or, rather, more complex variations of problems of logic, by breaking down the problem into a variety of stages which can be interpreted by the "logic" available from different gate circuits.

All the logic gate circuits described, together with the component values listed, are practical circuits that can readily be built by the amateur. The design of a computer using logic gates thus only demands a knowledge of the function of the individual gates and then a mixing or grouping of gates, as required, to cover the specific problem involved. A little further thought will also show that the function of different gates may be similar. Thus, a NOR circuit can

be either AND or OR, depending on conditions, e.g. positive AND is the same as negative OR. Similarly, the function of a two-input AND gate has an identical function to that of three NOR gates, the first two in parallel, and connected in series to the third.

The methods by which gates can be displayed or made to indicate their condition are numerous. Thus, meters can be employed in both the input and output circuits, although this arrangement is rather cumbersome and expensive. A simpler, and generally more suitable, method is to arrange for visual presentation by means of an indicator lamp in the output circuit of each gate (or such gates as require indicators).

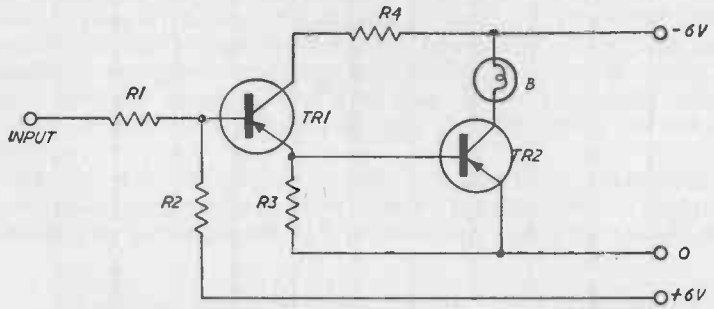


FIG. 36. INDICATOR LAMP CIRCUIT FOR USE WITH TRANSISTOR GATE CIRCUITS

*R*1	100kΩ	*TR*1	OC71
*R*2	220kΩ	*TR*2	OC72 or OC78
*R*3	1kΩ	*B*	6·3V 0·05A bulb
*R*4	4·7kΩ		

A suitable circuit matching any of the gate circuits described in this chapter is shown in Fig. 36. The indicator is a 6·3 volt 50 mA bulb connected via an OC72 (or OC78 or OC83) transistor across the output and −6 volt supply. The inclusion of a transistor in the bulb circuit ensures that the bulb will not draw any significant current from the gate circuit to which it is connected. The bulb will then indicate a 0 or 1 condition of the output by remaining off or lighting up, respectively.

CHAPTER 6

LOGIC CIRCUITS

THE digital computer designed for solving arithmetical problems is simply a counter; or at least the arithmetic block of such a computer is a counter. It works with 1s and 0s, and exactly the same conditions apply as in formal logic. A statement is either "true" or "false," hence the same type of circuit can be used to analyse statements embodying formal logic on the basis that a 1 is "true" and a 0 is "false." Thus, 1s, or continuing true statements, can be counted by a binary counter or passed by a gate circuit, but such a train is interrupted or stopped, as appropriate, by the appearance of a false bit (i.e. a 0).

All this seems very simple, and reduced to the basic operation of a logic computer it *is* simple. The main problem lies in reducing the problem requiring solution to the elementary steps involving formal logic, and then to provide the necessary computer circuits to cope. It has to deal with original truths, called *propositions*, and put these together to form a deduction or a *syllogism* based on a series of truths. This may involve *connectives* expressing the relationship, the chief of which are AND and OR.

This is quite different from an arithmetic counter, which has to deal with 1s and 0s that represent numerical digits. For that reason it is impossible to design a generalized type of logic computer, as can be done with counters. The computer has to be designed either to deal with a specific set of propositions and corresponding syllogisms, which can be met by a simple computer built with a knowledge of logic gate circuits; or be sufficiently ubiquitous in circuitry to enable it to be programmed to accept different specific sets of propositions and deal with their respective syllogisms, which calls for an extremely complex and expensive computer which will involve more than a simple logic block.

As an illustration of formal logic in its most elementary form, consider the following propositions—

(i) Some animals need a licence.
(ii) Dogs and cats are animals.
(iii) Dogs need a licence.

The syllogism is that if the subject is an animal *and* a dog it needs a licence. This could be analysed by an AND gate, as shown in Fig. 37.

The main subject is "animals," and the question is whether the household pet "Tom" needs a licence. "Tom" is an animal, so there is the possibility that he does need a licence, thus completing one input to the AND gate when this information is fed to the computer. "Tom" is also a dog, so by proposition (iii) this completes the second input to the AND gate. The two "bits" together produce an output from the gate that indicates a licence is required. On the other hand, had "Tom" been a cat, only the first input would have been completed to the gate, with no output resulting (i.e. no licence required). The complete computer to deal with such an elementary problem of logic is a single AND gate. It could deal with similar problems of logic, with more inputs if necessary, or provide just one stage of the analysis of a more complex relationship with additional connectives. The initial problem in the design of a suitable computer circuit is, then, a matter of breaking down the logic involved by suitable, and necessary, connectives. And the easiest way to do this is to convert the logic into Boolean algebra or symbolic logic, on the principles described in Chapter 5.

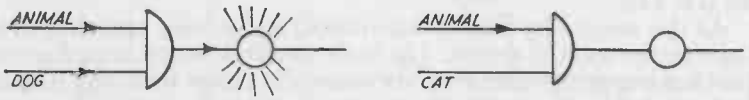


FIG. 37. THE "AND" GATE AS A SIMPLE LOGIC COMPUTER

Consider, for example, the elementary problem dealt with by the relay-type logic computer of Fig. 25. The propositions involved are—

A. Age 18–20 years.
*B*1. 6 "O" levels, or
*B*2. 3 "O" levels and 1 "A" level.
C. Maths at "O" level.

With the required connectives, and expressed in symbolic logic, this reduces to an equation

$$\text{Output (or answer)} = A.(B1 + B2).C$$

This means *A*, together with *B*1 or *B*2 and *C*; or, in terms of actual through circuit connexions, *A* in *series* with *B*1 and *B*2; *B*1 and *B*2 in *parallel*; and *B*1 and *B*2 in *series* with *C*. This is the actual switching circuit involved. The path can equally be considered in terms of gates, since series connexion is the equivalent of AND, and parallel connexion is the equivalent of OR. Thus, in the original equation—

$B1 + B2$ is an OR gate,
A and $(B1 + B2)$ is an AND gate,
$(B1 + B2)$ and C is an AND gate.

Reducing the problem of logic to symbolic logic (based on Boolean algebra) does, therefore, establish the circuit design, whether using

switches, relays or electronic gates. In other words, the development of a logical statement provides the basis of routing data through a logic computer.

However, development of a logical statement may often introduce an unnecessary number of steps or individual gates. In other words, the original statement or logic equation may be capable of simplification, just as ordinary algebraic equations can often be simplified for easier working. This can arise when one or more of the original propositions have no bearing on the result, or are merely a repetition of a fact already established elsewhere.

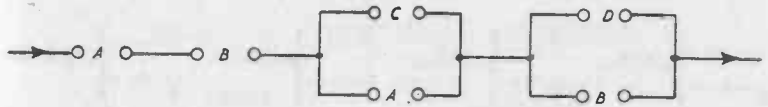


FIG. 38. DEVELOPMENT OF CIRCUIT RELATIONSHIP FOR "LOGIC"

Consider the circuit path shown in Fig. 38, the symbolic logic equation for which is—

$$A.B.(A + C).(B + D).$$

If *A* and *C* are closed, the path is completed to $(A + C)$, but *A* is already closed, so the path is completed to $(B + D)$. Here, *B* is already closed, so the whole path is completed. In fact, *C* and *D* have no bearing on the analysis of the problem, so this particular equation does, in fact, simplify to—

$$A.B.$$

In practical terms, this means that this particular problem can be dealt with by a single AND gate rather than a series of two AND and two OR gates, as originally analysed. Both would give the same result, but the simplified circuit is obviously to be preferred.

The significance of negation must also be understood since this applies an opposite sign or a NOT. Thus, $\bar{A}$ is the opposite of *A*, or not *A*, in logic, namely—

if $A = 1$, $\bar{A} = 0$
if $A = 0$, $\bar{A} = 1$.

This is important, for in a series connexion (AND gate)

$$A.\bar{A} = 0$$

and with parallel connexion (i.e. an OR gate)

$$A + \bar{A} = 1.$$

In terms of circuitry, this means that a switch in series with its negation produces an open circuit, and a switch in parallel with its negation produces a closed circuit.

All types of computer gate functions can be drawn with respect to the possible outputs, including an inverter I. These are shown symbolically for a two input AND gate in Fig. 39; and a two input OR gate in Fig. 40. These represent the four possible outputs for logical AND and logical OR, respectively.

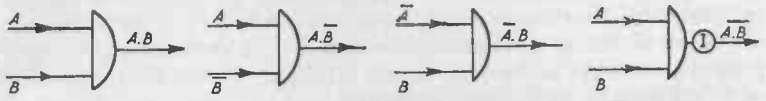


FIG. 39. THE FOUR POSSIBLE OPERATING CONDITIONS OF AN "AND" GATE

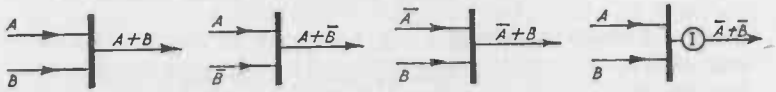


FIG. 40. THE FOUR POSSIBLE OPERATING CONDITIONS OF AN "OR" GATE

Suitable combinations of such gates can be employed to analyse almost any problem broken down into basic logic, a 1 passed by a particular gate being a "go," or input, to the next, or related gate, and so on. The complete network will involve a series of passages for the information or directions fed to the circuit, some of which may involve dead ends, i.e. no further progress is possible and thus there is no "logical" answer for that set of data, and others which may involve alternative directions or alternative commands in order to progress through the remainder of the circuit.

In general, it is easiest to consider the network design in terms of simple switching circuits first, to establish the required path and also to show where the symbolic logic may be simplified, if possible. The particular advantage of gate circuits is the readiness with which they can accept a large number of inputs, if necessary, thus reducing the possibility of developing "dead end" circuits with waste of components.

TABLE 11: SERIES ("AND") CIRCUITS

$A = 1 \quad B = 1$

$A \cdot B$	1·1	1
$A \cdot \bar{A}$	1·0	0
$\bar{A} \cdot A$	0·1	0
$A \cdot \bar{B}$	1·0	0
$\bar{A} \cdot B$	0·1	0
$\bar{A} \cdot \bar{B}$	0·0	0
$A \cdot \bar{A}B$	1·0	0
$A\bar{B} \cdot B$	0·1	0

TABLE 12: PARALLEL ("OR") CIRCUITS

$A = 1 \quad B = 1$

$A + B$	1 + 1	1
$A + \bar{A}$	1 + 0	1
$A + \bar{A}B$	1 + 1	1
$B + \bar{B}$	1 + 0	1
$\bar{B} + B$	0 + 1	1
$A\bar{B} + B$	1 + 1	1
$\bar{A} + \bar{B}$	0 + 0	0

TABLE 13: "AND" GATE FUNCTION

INPUT	OUTPUT TO NEXT GATE OR STAGE
True (or Yes) and True (or Yes)	Go
True (or Yes) and False (or No)	Stop
False (or No) and False (or No)	Stop

TABLE 14: "OR" GATE FUNCTION

INPUT	OUTPUT TO NEXT GATE OR STAGE
True (or Yes) and True (or Yes)	Go
True (or Yes) and False (or No)	Go
False (or No) and False (or No)	Stop

CHAPTER 7

DIODE AND VALVE CIRCUITS

A DIODE, with its capacity to pass current in one direction and block current flow in the opposite direction, is the simplest of electronic gate elements. Thus, if a 1 is represented by a positive direction of current and a 0 by a negative direction of current a diode, suitably connected, will pass 1s and cut off 0s.

notation	1	0
current	+	−
output	1	0

This can apply both for steady currents, and for pulses (i.e. positive and negative pulses). In a practical circuit it is merely necessary to bias the diode in the appropriate manner to make it perform as a simple switching gate regulating the transfer of information (pulses) from input to output (cf. Fig. 26).

Diode switching action is further illustrated in Fig. 41, where corresponding switch circuits are shown. At (*a*) the diode offers high resistance to the input signal, equivalent to an "open" switching action. Thus, the input signal is directed to the output.

At (*b*) the diode configuration is such that it can conduct, which is equivalent to a closed switching circuit. The input thus flows primarily through the diode with only a small proportion applied to the output. This is a particularly useful form of switching action since the diode itself remains essentially outside the input-output circuit path and merely acts as though it were an "external" switch.

Fig. 42 shows diodes connected in the form of an AND gate with four inputs. There are two possible configurations for the diodes, one of which will yield a positive AND gate and the other a negative

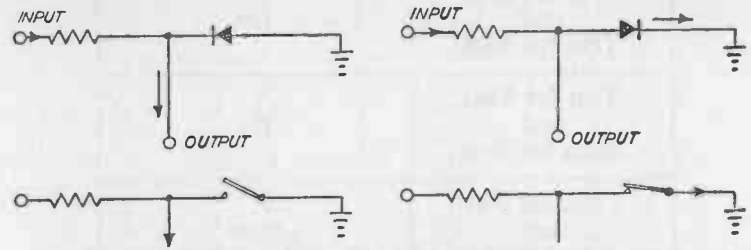


FIG. 41. (a) left and (b) right. DIODE SWITCH ACTION COMPARED WITH A MECHANICAL SWITCH CIRCUIT

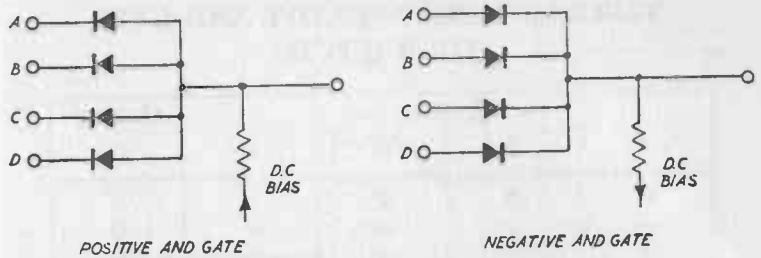


FIG. 42. DIODE "AND" GATE CIRCUITS

AND gate. In the first an output will be given when all the four inputs are positive, and in the second when all the four inputs are negative. In practical circuits this may involve both negative and positive voltages, when the response is best considered in terms of "high" and "low" voltages representing a 1 and 0, respectively. Thus, a positive AND gate will produce an output when the inputs all have a positive "high" voltage, and a negative AND gate when all the inputs have a negative "high" voltage (i.e. in this case "high" means that the input voltage is *more* negative than the "low" negative voltage). *See also* tables 15 and 16.

TABLE 15: DIODE POSITIVE AND GATE (FOUR INPUTS)

INPUTS				OUTPUT
A	*B*	*C*	*D*	*S*
0	0	0	0	0
+	−	−	−	0
−	+	−	−	0
−	−	+	−	0
−	−	−	+	0
+	+	−	−	0
+	−	+	−	0
+	−	−	+	0
−	+	+	−	0
−	+	−	+	0
−	−	+	+	0
+	+	+	−	0
+	−	+	+	0
+	+	−	+	0
−	+	+	+	0
+	+	+	+	1

+ = "High" = 1
− = "Low" = 0

TABLE 16: DIODE NEGATIVE AND GATE (FOUR INPUTS)

Inputs				Output
A	*B*	*C*	*D*	*S*
0	0	0	0	0
−	+	+	+	0
+	−	+	+	0
+	+	−	+	0
+	+	+	−	0
−	−	+	+	0
−	+	−	+	0
−	+	+	−	0
+	−	−	+	0
+	−	+	−	0
+	+	−	−	0
−	−	−	+	0
−	+	−	−	0
−	−	+	−	0
+	−	−	−	0
−	−	−	−	1

− = "High" = 1
+ = "Low" = 0

The negative AND diode gate is also the equivalent of a positive OR gate, and which function it performs depends on how the circuit is used, i.e. the respective significance of "high" and "low" voltages. Equally, the positive AND diode gate is the same in function as a negative OR gate.

Although simpler circuit elements than transistors, the "gate" working of diodes is often a little obscure until evaluated on the basis of the significance of "high" and "low" voltage response. Transistor gates are, therefore, usually preferred for direct working with semiconductors and can also provide amplification at the same time. Nevertheless, the diode remains a very useful computer circuit element and is often used in conjunction with transistors in gate circuits, and also as a blocking device at appropriate parts of the circuit.

Thermionic valves can be used for switching elements in computer circuits. Their chief disadvantage, particularly from the amateur constructor's point of view, is that they require higher operating voltages than transistors to perform similar functions. Nevertheless, they can still provide compact circuits, particularly when using sub-miniature "trigger" tubes for providing flip-flop circuits. Also, for

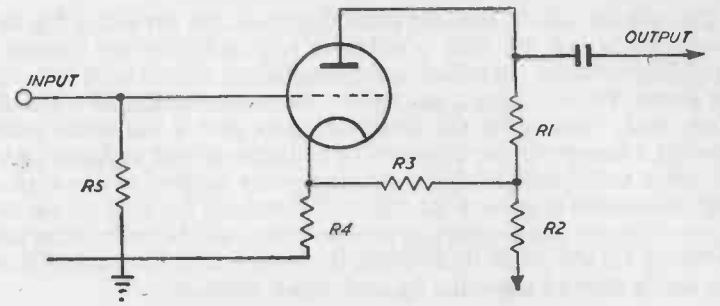


FIG. 43. THE TRIODE VALVE AS A SWITCH ELEMENT

decade counters a special valve or valves can be used to indicate visually the state of count over a range of ten digits or, more directly, in decimal numbers (*see* Chapter 4), when they are called *numicator* valves.

Fig. 43 shows a triode valve used as a switch element. *R*2, *R*3 and *R*4 together form a voltage divider, with values selected so that in the normal state the cathode voltage is positive and the valve is cut off. A positive signal applied to the control grid greater than this bias will then make the valve conduct and produce an output across *R*1. In other words, the application of a signal of the correct type (correct positive voltage) switches the valve from OFF to ON. In practice the signal would normally be in the form of a pulse, which is reproduced at the output.

Fig. 44 shows this principle carried a stage further, using a tetrode valve with the cathode bias adjusted so that *one* positive input signal *V*1 *or V*2 applied to the grid is insufficient to overcome the bias, and so the valve remains non-conductive; but both *V*1 *and V*2 together will produce conduction (by overcoming the cathode bias) and give an output. This circuit is, therefore, an AND gate. Similarly, with a lower standing bias on the cathode the same circuit could act as an OR gate (the valve conducting when either *V*1 *or V*2 is applied to the grid). In addition to performing a logic function the circuit will also provide amplification of the input signal.

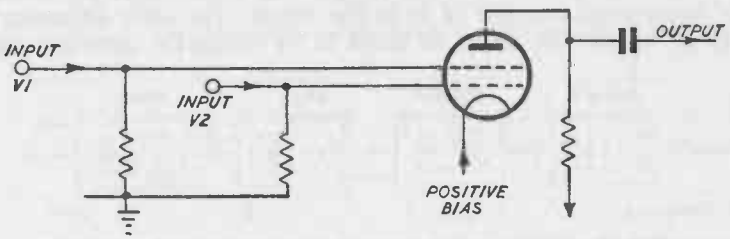


FIG. 44. TETRODE VALVE "AND" GATE CIRCUIT WITH TWO INPUTS

Two similar valves, suitably interconnected, can provide a flip-flop circuit, as in Fig. 45. This is identical in function to the transistor flip-flop previously described and is basically a switch with two output states. With no input, one valve conducts more heavily than the other, and, because of the feedback, goes into a saturated state, drawing a heavy current through the cathode resistor and cutting off the other valve completely. A positive pulse applied to the grid of both valves will then have the effect of switching the state of the two valves—the one not conducting now conducts and becomes saturated switching off the other. The circuit is bi-stable and is switched from one stable state to the other by each input pulse.

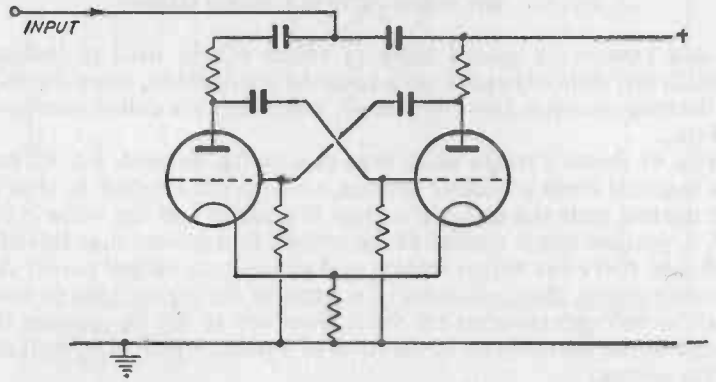


FIG. 45. FLIP-FLOP VALVE CIRCUIT USING TRIODES

This, of course, is the basic counter for a binary computer. To extend the range of counting it is only necessary to add additional stages (Fig. 46), as with the transistor binary computer (Chapter 3). By introducing *preset*, together with a suitable number of stages, the complete circuit can be adapted to count directly in decimals (decade counter), or any other numbering system that is required. Further, the preset may be adjustable, enabling the same binary counter to be used for different number systems.

The basic requirement is that the preset is established by inserting the appropriate number of 1s in the circuit. The most elementary way in which this could be done is by manually signalling the

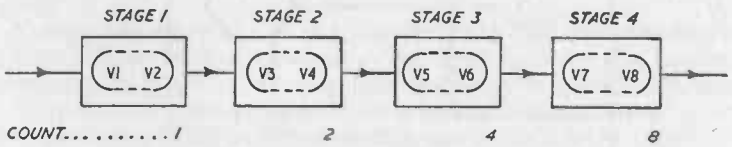


FIG. 46. FOUR-STAGE FLIP-FLOP WITH DOUBLE-TRIODE VALVES

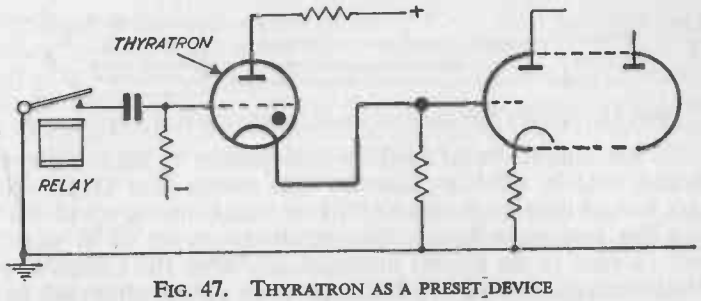


FIG. 47. THYRATRON AS A PRESET DEVICE

appropriate preset after the completion of each count through the number of stages available, although in practice this would be too slow and liable to error. Normally, the preset would be established electronically, using a separate valve (or transistor) as a switch element. Thus, Fig. 47 shows a thyratron preset switch. The thyratron circuit is open until the relay is pulled in, and thus normally has no part in the counter working. The relay itself is connected to the output at the end of the counter stages and is thus pulled in only when the count is completed through the stages.

Closing the relay contacts completes the thyratron circuit, which then applies a positive pulse to the grid of the appropriate "stage" valve for which the thyratron is acting as a preset switch, "flipping" (or "flopping") that stage and thus presetting it with a 1. With a suitable connexion to appropriate stages preset can be applied automatically to such stages as required at the completion of each count. Also, for variable preset, the individual thyratron circuits can be connected to their appropriate stage valves via mechanical switches. Thus, the desired preset number can be selected by turning a knob controlling the mechanical switching. Exactly the same principle applies to transistor circuits for deriving automatic preset, whether adjustable or not.

When a *fixed* preset is required, then separate preset switching is not always necessary as the same function can be performed by feedback. This can be demonstrated by the basic circuit required for adapting a binary counter to a decimal (decade) counter. To count

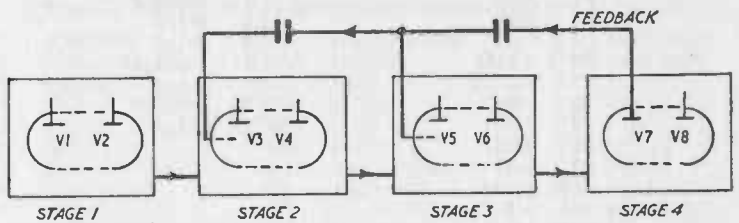


FIG. 48. PRESET BY FEEDBACK IN A FOUR STAGE FLIP-FLOP FOR DECIMAL COUNTING

FIG. 49. GTR 120W TRIGGER VALVE, APPROXIMATELY ACTUAL SIZE

by 10, the counter must initially incorporate at least a ten-digit capacity, and in a binary counter this means four stages—three stages would only count up to 8; four stages count up to 16. To adapt the four-stage binary counter to count by 10 it must be *preset* to read in six counts automatically after the completion of each (16) count, so that, in effect, it gives an output after each count of 10. Fig. 48 shows how this can be done with interconnexion between stages providing feedback.

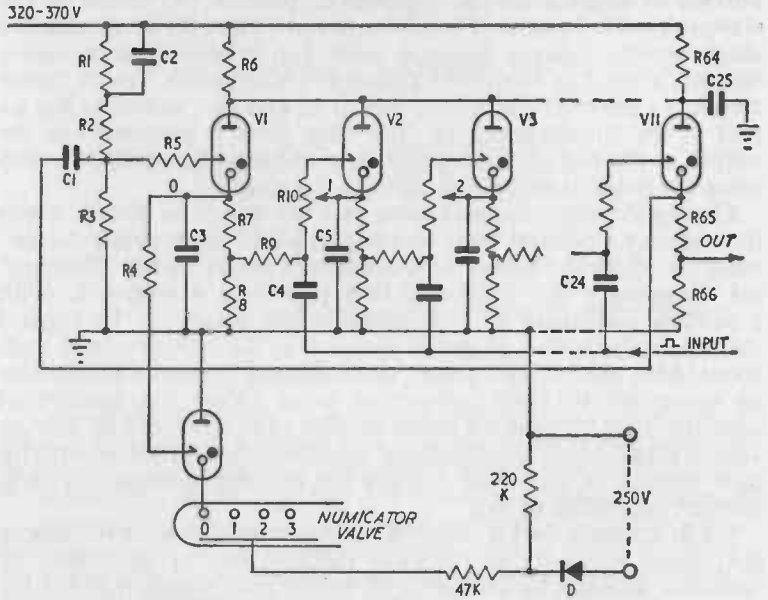


FIG. 50. COMPLETE CIRCUIT FOR DECADE COUNTER

*R*1	1μΩ	*C*1	0·002μF
*R*2	1μΩ	*C*2	0·1μF
*R*3	1μΩ	*C*3	0·1μF
*R*4	4·1kΩ	*C*4	0·002μF
*R*5	100kΩ	*C*5	0·1μF
*R*6	10kΩ	*C*24	0·002μF
*R*7	15kΩ	*C*25	0·1μF
*R*8	22kΩ		
*R*9	1μΩ		
*R*10	100kΩ		
*R*64	3·3μΩ		
*R*65	15kΩ	*V*1, *V*2 etc.	GTR120W
*R*66	15kΩ	Numicator valve	XN3

Feedback is taken from the anode of *V*7 in the four-stage binary counter, back to the grids of *V*3 in stage 2 and *V*5 in stage 3. The effect of this is that the counter starts at 0110 (decimal 6) instead of 0000. After nine pulses the count will be 1111. The next (tenth) pulse will then flip stage 4 to provide a negative output pulse from *V*8, with *V*7 positive. This positive pulse on *V*7 is transmitted back to *V*3 and *V*5 to establish the preset simultaneously with the output. In other words, there is an output pulse after the tenth pulse is applied, while a preset of 0110 is set up at the same time ready for starting the next count of up to ten pulses.

A valve-type decade counter, which is particularly suitable for amateur construction, can be based on trigger valves or a special type of thyratron. These valves are diminutive in size, a suitable type specially designed for computer work being the Ericson Type

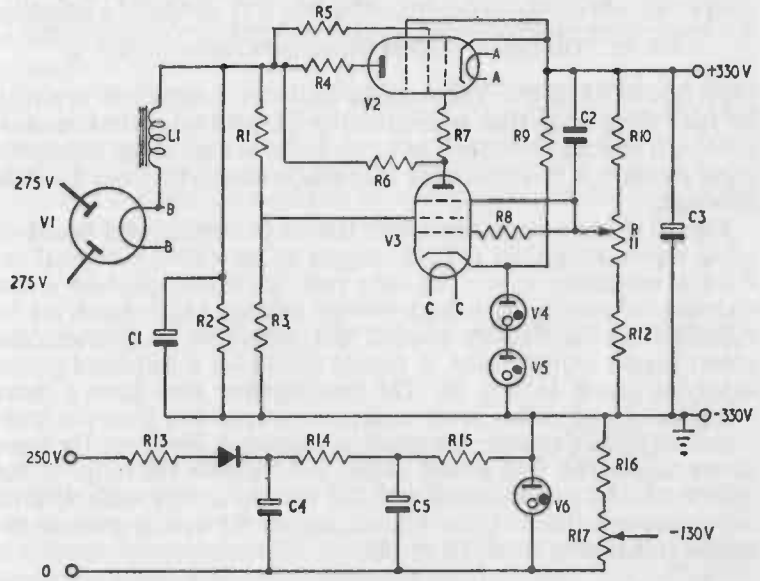


FIG. 51. POWER SUPPLY FOR DECADE COUNTER CIRCUIT OF FIG. 50

*R*1	100kΩ1W	*R*11	10kΩ variable	*C*4	16μF 450V
*R*2	10kΩ25W	*R*12	100kΩ	*C*5	16μF 450V
*R*3	150kΩ1W	*R*13	100Ω	*U*1	SU4G
*R*4	15Ω	*R*14	470Ω	*U*2	FL81
*R*5	150kΩ	*R*15	68kΩ	*U*3	EF91
*R*6	1μΩ	*R*16	68kΩ	*U*9	XC12T
*R*7	220Ω	*R*17	10kΩ variable	*U*5	XC12T
*R*8	220Ω	*C*1	16μF 600V	*U*6	150C4
*R*9	150kΩ	*C*2	0·5μF 500V	*A-A*	6·3V 2·5A
*R*10	100kΩ	*C*3	8μF 450V	*B-B*	5V 2A
				C-C	6·3V 2·5A

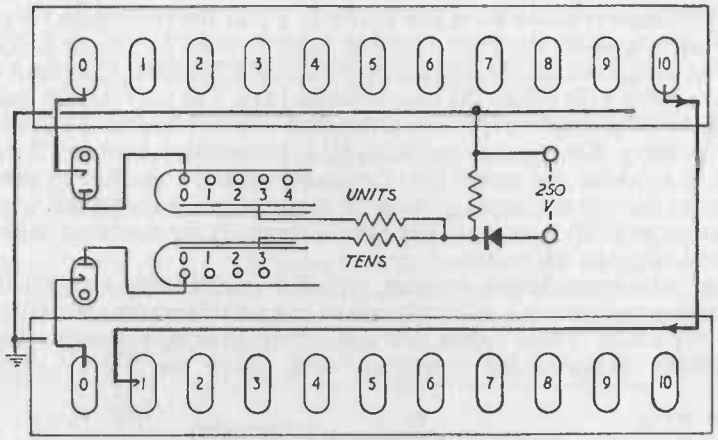


FIG. 52. THE DECADE COUNTER EXTENDED TO COUNT TO 10 × 10

GTR 120 W (Fig. 49). They can be soldered directly into a circuit by their three leads (like transistors) or plugged into sub-miniature holders. The only precaution to be observed in this respect is that the wires should not be soldered or bent nearer than ½ in. from the glass envelope.

Fig. 50 shows a complete circuit for the decade counter based on using eleven such cold cathode trigger tubes with direct read-out from a numicator tube. The only real limitation involved is the necessity of employing a high-voltage supply, which needs to be stabilized for satisfactory results, and somewhat complicates the power supply requirements. A typical circuit for a stabilized power supply is shown in Fig. 51. The two together thus form a more complicated and rather more expensive proposition than the transistorized decade counter described in Chapter 4. However, the same power supply can feed several stages and increase the range of the decade counter by "doubling up" the decade blocks, each with its own numicator tube. Thus, Fig. 52, shows the decade counter extended to count to 10 × 10 = 100.

CHAPTER 8

THE ARITHMETIC UNIT

THE arithmetic unit embodies the calculating circuits of the computer, a series of two-stage devices (e.g. flip-flops) dealing with the numbers fed to them to add, subtract, multiply, divide or compare. Where the computer is based on logical analysis the same principle applies, but the circuit becomes a logic unit activated by 1 or 0 input signals (pulses) which are operated on in a logical rather than an arithmetical fashion. It is generally easier to consider the operations involved in terms of numbers, and so description in this chapter will be confined to the arithmetic unit.

Binary numbers can be dealt with either in series or in parallel. That is to say, the input pulses (1s or 0s) can follow one after the other on a single path, or each can be routed through separate paths (Fig. 53). The advantage of parallel routing is that it is much faster since all the digits involved can be operated on simultaneously, but it does require a more elaborate circuit.

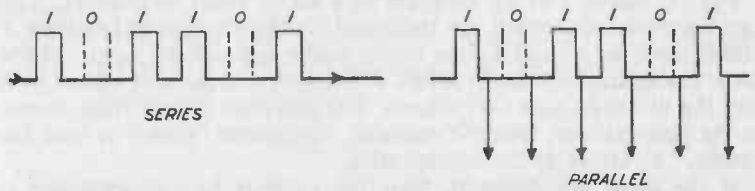


FIG. 53. SERIES AND PARALLEL READING OF BINARY DIGITS

Each pulse represents a *bit* (binary digit). These can be grouped in threes (binary triplets), the number of such triplets depending on the number to be represented in binary terms. Thus, each triplet is represented by binary 000 to 111 or 0 to decimal number 7, in fact giving the equivalent octal numbering system described in Chapter 2 (*see* Fig. 54).

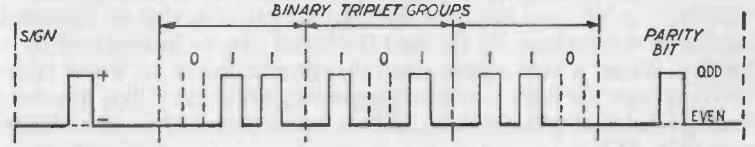


FIG. 54. A COMPUTER WORD COMPRISING BINARY TRIPLETS PLUS SIGN AND PARITY BITS

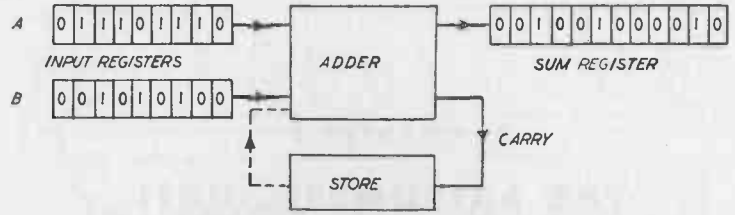


FIG. 55. BLOCK DIAGRAM OF A SERIES ADDER/SUBTRACTOR

Complete "intelligence" is then provided by a separate single digit or bit *preceding* the required number of groups, and a single bit *following* the number of groups. The leading bit conveys the sign of the following (octal) number: 1 for + and 0 for −. The trailing bit, or last digit, is called a *parity bit* and is really a checking pulse. If there is an odd number of pulses in the preceding groups the parity bit will be a 1. If there is an even number of pulses in the preceding groups the parity bit will be a 0. This is so that the total number of pulses is *always* even, which is a simple means of checking that the combination of groups, known as a computer *word*, is correct. Thus, computer *words* should always have an even number of pulses. If a check shows that they have not, then there is a fault or error somewhere in the word. Computer words thus derived are then fed to the next stage for calculation or logic.

Fig. 55 shows a block diagram of a series adder/subtractor. The two numbers concerned are indicated on the two input registers *A* and *B*, and fed one at a time to the adder unit. If the signs are the same (as defined by the first bits of the two words) *A* is added to *B* and the common sign (+) affixed. The resultant sum is then passed to the sum register, where it remains. A separate "store" is used for "carry," to arrive at the correct sum.

If the signs are different, then the number to be subtracted is converted into its complement and then added, with end carry round. In other words, addition or subtraction is performed by an *adder* unit.

In parallel addition/subtraction, exactly the same principle is involved except that there are as many adders as there are digits on the input registers *A* and *B*, to deal with all the digits simultaneously. The adders are series connected to feed from one to the next for "carry," each feeding its respective "column" in the sum register.

Evaluation of signs involves the use of logic. The sign of the octal number is represented by the first digit and can be interpreted by a flip-flop. Thus, a sign comparison circuit can derive its input from flip-flops (one for each number interpreting the sign of that number) feeding four AND gates which, in turn, are connected in pairs to OR gates (Fig. 56).

With two numbers, there are four possible combinations of signs:

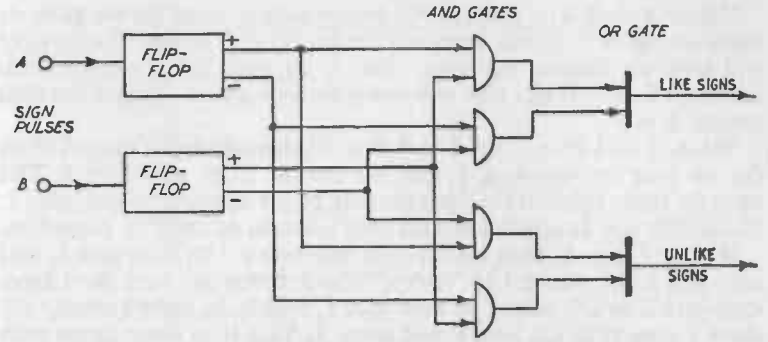


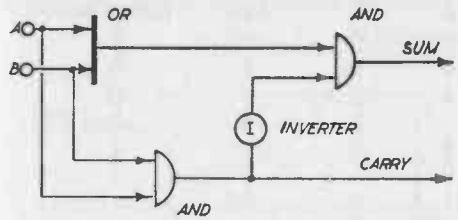
FIG. 56. USE OF FLIP-FLOPS AND GATES FOR SIGN EVALUATION

+ and +, + and −, − and +, − and −. If the two flip-flops have the same output, AND-gate 1 will have an output if the signs are +, and AND-gate 2 will have an output if the signs are −. The coupled OR-gate will then have an output corresponding to "like" signs. In a similar manner, if the two signs are unlike (+ and −, or − and +), either AND-gate 3 or AND-gate 4 will have an output, passed on by the corresponding OR gate. In the first case the output ("like" signs) implies a straightforward addition of digits in the sum register. In the second case the output ("unlike" signs) signals that the number to be subtracted must be turned into its complement before being added.

Both "half" and "full" adders can be used for the next step. A *half adder* can deal with two digits, giving four possibilities—

Number *A*	Number *B*	Sum	Carry
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

A basic half-adder circuit is shown in symbolic form in Fig. 57, and comprises an OR gate, two AND gates and an *inverter*. The inverter is simply a device which changes a 1 to a 0, or a 0 to a 1. The circuit then works as follows.



A	B	SUM	CARRY
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

FIG. 57. BASIC HALF-ADDER CIRCUIT AND TRUTH TABLE

When $A = 0\ B = 0$ there will be no output from the OR gate, or from AND-gate 2. Hence there will be no "carry" pulse. The inverter will have an output, however, since it changes the 0 output from AND-gate 2 to a 1, but this will not pass AND-gate 1. Hence the sum output is 0.

When $A = 1\ B = 0$, or $A = 0\ B = 1$ there will be an output from the OR gate to AND-gate 1, but no output from AND-gate 2. The inverter turns this 0 from the AND gate to a 1 applied to AND-gate 1. Thus, with two 1s applied to AND-gate 1, a sum of 1 will be passed on.

If $A = 1\ B = 1$, then the OR gate will pass a 1 to AND-gate 1, and AND-gate 2 will pass a 1 to "carry." The inverter will turn the 1 from AND-gate 2 to a 0 passed to AND-gate 1, which, in consequence, will show a sum of 0, i.e. sum 0 and carry 1. This is in accordance with the truth diagram.

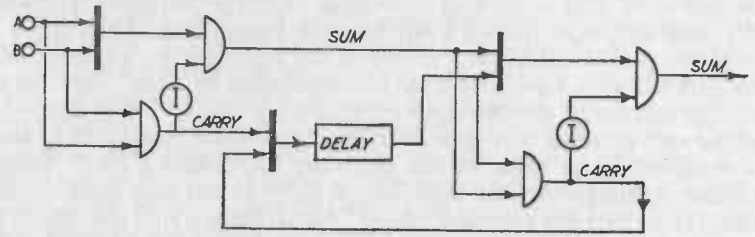


FIG. 58. BASIC FULL-ADDER CIRCUIT

Two half adders together form a full adder (Fig. 58). This circuit first produces a partial sum and then adds the carry, shifted one place to the left, to provide the final sum. A delay element is required to store or hold the carry for one-pulse time so that the carry is passed on as the adder receives the next pair of pulses.

For dealing with three input digits exactly the same principles apply. Logic elements are introduced to conform to the various possible combinations and to give the correct output. Thus, a full-adder for three inputs (three-digit numbers) is shown in Fig. 59, and the corresponding truth table is—

	Binary				
Decimal	*A*	*B*	*C*	Sum	Carry-out
0	0	0	0	0	0
1	1	0	0	1	0
2	0	1	0	1	0
3	0	0	1	1	0
4	1	1	0	0	1
5	1	0	1	0	1
6	0	1	1	0	1
7	1	1	1	1	1

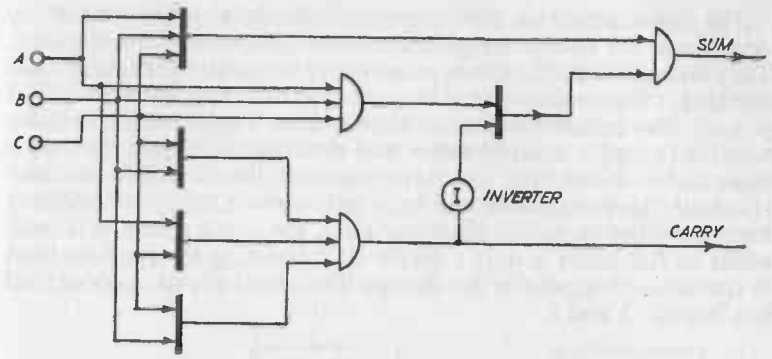


FIG. 59. FULL-ADDER CIRCUIT FOR THREE INPUTS

Multiplication can be accomplished by repeated addition or, more simply by "shifting." In the case of binary numbers, shifting one place to the left is equivalent to multiplying by 2. Thus—

Decimal		Binary
6		110
× 2	Shift left	1100 = decimal 12
Answer 12		

Division can be accomplished by repeated subtraction or shifting in the opposite direction. Shifting one place to the right is the equivalent of dividing by 2, in binary numbers. Thus—

Decimal		Binary
16		10000
÷ 2	Shift right	1000 = decimal 8
Answer 8		

In a binary computer, shifting is done in the *shift register*, which is simply a series of flip-flops connected together in series to transfer information from one to the next when the advance or "shift" input is pulsed. A delay element is also incorporated, as with the full adder, to store the transfer for one-pulse time (Fig. 60), and the number of stages required is equal to the number of bits in the "word."

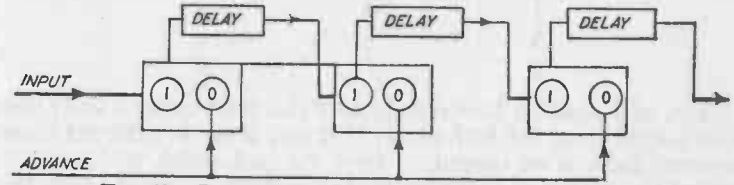


FIG. 60. BLOCK DIAGRAM OF SHIFT REGISTER OPERATION

The introduction of shift registers does, however, considerably complicate the circuit design and number of components required. They are a standard feature of commercial computers for arithmetical working, where costs may run into tens and hundreds of thousands of pounds. For amateur computer construction it is generally far more suitable to tackle multiplication and division by "repetition" on a basic adder circuit. The individual elements involved then embrace standard flip-flop circuits and logic gate circuits, all of which have been described in earlier chapters. Thus, the construction of a half-adder or full-adder is only a matter of translating the symbols used in the circuit diagrams in this chapter into actual circuits as described in Chapters 3 and 5.

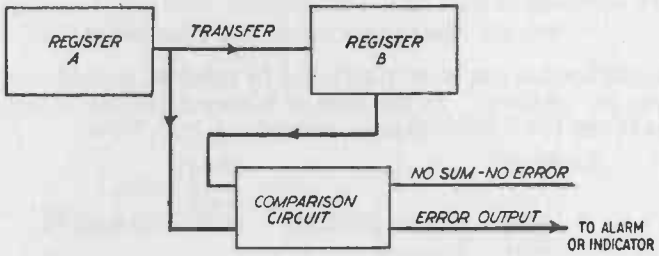


FIG. 61. THE COMPARATOR USED FOR DETECTING ERRORS

Mention should be made, however, of the use of the half-adder circuit as a comparator. A comparator, or comparison circuit, is a device for detecting errors in passing on a number or computer word. Supposing, for example, a word is to be passed from register *A* to register *B*. The addition of connexions (i.e. inputs derived from each register) to a half-adder circuit then provides a means of detecting whether these inputs are the same (equal, and thus no error) or unequal (and thus in error). This follows from the truth table of the half-adder circuit, which is rewritten below in terms of the conditions of register *A* and register *B*, and half-adder output. The comparison circuit is shown in Fig. 61.

Register *A*	Register *B*	Half-adder output
0	0	0
1	0	1
0	1	1
1	1	0

Thus, with equality between *A* and *B* (i.e. both either 0 or 1) there is no output from the half-adder. If *A* and *B* are in different states, however, there is an output, 1, from the half-adder, which can be applied to operate an alarm, indicator or other warning device that

the transfer from register *A* to register *B* has produced an error at this stage of the circuit.

Comparator circuits are simple—merely comprising a standard half-adder circuit—and can be used to advantage for checking at various stages of a computer circuit. And the half-adder circuit consists of a combination of two AND and one OR gates, the layouts for which have already been described in Chapter 5. Inversion can be carried out by an inclusive NOR gate, utilizing just one input.

CHAPTER 9

MEMORY DEVICES

SIMPLE computers may have direct read-out. That is to say, the answer to a straightforward arithmetical or logic problem is indicated by lights at each stage (e.g. the binary counter of Chapter 3) or an electro-mechanical counter (e.g. the decade counter of Chapter 4). With more complex computers and more complex problems it becomes necessary for the computer to store information. This may be a momentary storage to perform an individual calculation; a larger storage for information which is not immediately required; or bulk storage into which masses of data are fed to the computer for permanent storage so that it can later be extracted to obtain answers, either as single bits of data or in combinations, to provide a logical answer to a set of input data (questions). Equally, storage devices may be used to provide the input signals to a computer. Thus, a complete sequence of operations may be stored on tape, or some other device, and fed into a computer whose output commands a fully automated system. The whole system—input, command (by computer) and output (controlling external actions)—is then fully automated.

Storage or "memory" devices are quite commonplace and by no means confined to computer work. An ordinary filing cabinet is a memory device, the placing and selection of data both being performed manually. It can be made faster and more reliable by storing data on punched cards, when individual data can be placed (stored) and selected only on the basis of the code adopted, i.e. the position of the punched holes. The usefulness of such a system can be further improved by combining it with an electro-mechanical or electronic reading device. There are numerous other possibilities, such as magnetic tape, magnetic drums and magnetic cores as storage media. Any such storage method which works on a simple "on-off" or "yes-no" basis is directly applicable to binary computer storage.

The punched card is a "yes-no" device, for holes are punched in the card at points appropriate to the appearance of specific information. Thus, an elementary 10-code system would involve ten possible positions for a punched hole to appear in a line on the card (or more specifically a paper tape), when the position of the hole would determine the digit stored. Thus, in Fig. 62 the figure 7 is stored on the tape.

In practice the code can be extended to any suitable length, e.g. to

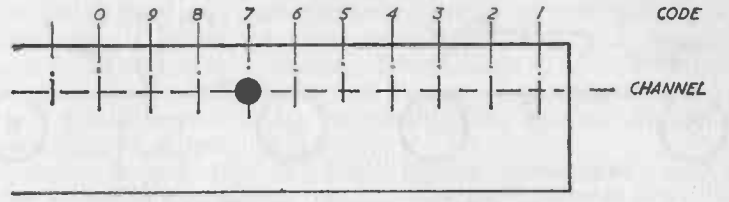


FIG. 62. SIMPLE PUNCHED-TAPE CODE

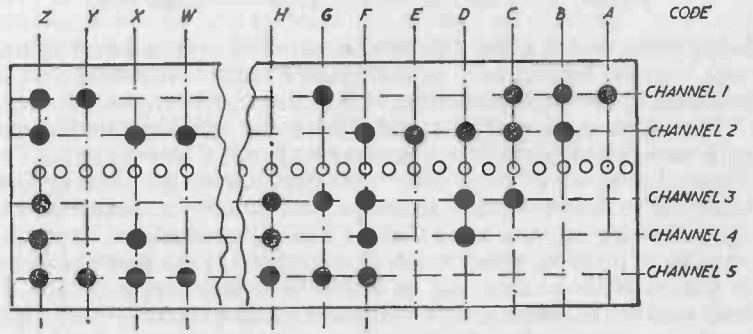


FIG. 63. FIVE-CHANNEL PUNCHED TAPE

cover a complete alphabet, and it can incorporate more than one possible "line" position, known as a channel. Fig. 63 shows a five-channel paper tape, which would normally also incorporate an additional line of uniformly spaced holes, not to convey storage information as such but merely to serve as a method of guiding the tape accurately through the reader. These are known as location holes.

Two simple forms of reader are shown in Fig. 64. The first, (*a*), employs a wire brush rubbing lightly on the surface of the tape. There is no contact between the brush and the underlying metal plate until a punched hole appears under the brush, when the electrical circuit is completed and an output pulse is produced. The photocell reader (*b*) operates on the same basis, except that the output circuit is triggered by a hole appearing in front of the photocell. It has the advantage of avoiding mechanical contacts and mechanical wear, but is more expensive, which can be a significant factor when there is a large number of channels on the tape and each channel has to be read separately.

Punched card and paper tape storage, together with their appropriate readers, are widely used in accounting machines, and similar devices and are, in fact, a form of mechanical computer. Punched paper tapes are also widely used for information storage, particularly *input* storage, with electronic computers. They have the advantage of

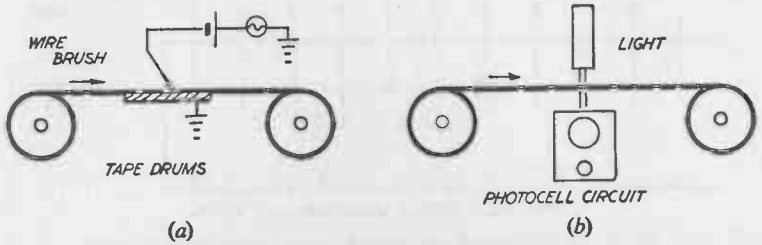


FIG. 64. TWO BASIC METHODS OF READING PUNCHED TAPES

being permanent (i.e. the data can be extracted over and over again), with capacity limited only by the length of tape that can be accommodated. Their main limitation is that they are low-speed devices.

The magnetic tape offers exactly the same possibilities for storage, with some advantages. It is a permanent form of storage again, but it can also be wiped clean and "rewritten." Since it is dealing with binary information it needs to record only pulses, i.e. 1s and 0s, and again can incorporate a number of individual channels. It is also possible to pack the pulses much closer together than punched holes, so that much more data can be stored in a given length of tape. In fact, with pulse-packing, something like 2,000 to 3,000 binary digits can be stored in each single foot of tape.

As with paper tapes, the speed at which a particular bit can be found is low (compared with electronic storage devices). Typical computer tapes may run to 2,000 to 3,000 *feet* in length, and if the required bit is at the opposite end of the tape the whole length has to be run through. Running the tape at very high speeds presents problems, notably the necessity of stopping the tape at exactly the right point with no lost motion, and so the drive is quite different from that of a conventional tape-recorder.

A typical system is shown in Fig. 65, where the tape "floats" in two loops controlled by pairs of photoelectric switches, each tape drum being driven by its own servo motor. Thus, for starting, each

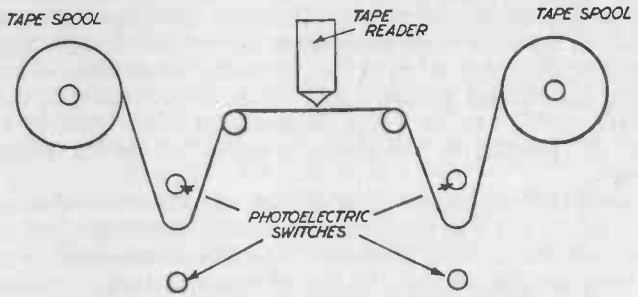


FIG. 65. TAPE SYSTEM FOR HIGH-SPEED OPERATION

drum has to draw only from the freely draped loop and not from the whole tape, and at the same time the other drum starts to maintain the loops at the correct positions. Quick starts and stops are thus possible without putting a strain on the tape. A typical start or stop time is 5 milliseconds, while for searching the tape can be run at speeds of up to 10 feet per second.

Computer storage tape also differs from normal magnetic tape in being wider ($\frac{1}{2}$ in. is usual), and it is "printed" without regard to sound quality. It has only to record pulses, hence high currents are used in the writing head to ensure saturation of the tape. Nevertheless, an ordinary tape-recorder can be used as a storage device for simple computers (particularly computer input), although it will suffer from limited speed of operation.

For computer data storage with magnetic tapes, data are recorded in blocks of computer words normally occupying approximately one inch of tape, individual blocks being separated by blank lengths of tape. Block positions are allied to the counter circuit. Any stored information can therefore be located by commanding the tape to run to a specified block number, and thence to the appropriate word in that block. This is done by an *address selector* pulsing the required "count" to position the tape. This would normally be applied to a comparator circuit to check that the position at which the tape stops is correct before the word is read by the tape head.

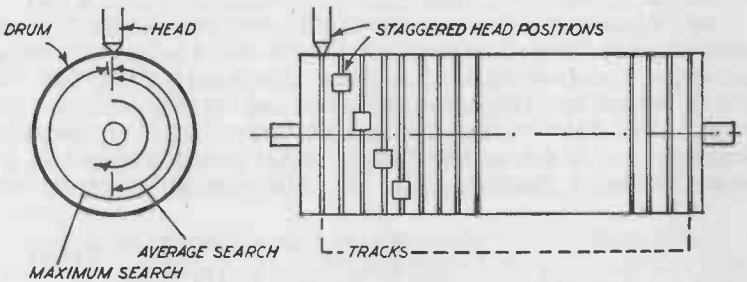


FIG. 66. MAGNETIC-DRUM MEMORY STORAGE

Very much faster operation, or shorter *access time*, is possible if the tapes are arranged as tracks around a drum, forming what is called a magnetic drum (Fig. 66). Although this reduces the overall length of tape that it is possible to incorporate on a practical size of drum, the drum itself can be run at speeds of up to 15,000 rev/min and the information required is always less than one revolution away. Thus, the *maximum* access time with drum storage may be only a matter of 2 milliseconds or even less, with the *average* time one half of this figure.

Individual tracks can be quite crowded on the drum—as many as one hundred tracks per inch—each with its own recording-playback

head arranged in staggered formation to obtain the necessary clearance. With pulse packing, typical sizes of magnetic drums running at speeds of 3,600 rev/min (approx. 4 msec access time) can have a memory capacity of anything between a quarter of a million and a million bits. Larger diameter drums can store several million bits.

As an alternative to magnetic drums, magnetic discs may be used, similar to ordinary gramophone records but with a much greater number of tracks per inch (radial). To increase the storage capacity it is only necessary to increase the number of discs and heads, while still maintaining a very fast access time. In both cases the required word is found by an address selector operating on the same basis as that described for magnetic tapes.

Alternative methods of storage utilize relays or magnetic cores. The earliest electronic computers used relays, which are simple "on-off" switches. Thus, in one position (energized) the relay represents a 1 condition, and in the other position (de-energized) a 0. If the relay is a *latching* type it will remain in the position to which it was triggered to retain a 1 condition, and be unlatched to provide a 1 signal by the appropriate "search" signal.

Although this is a direct and easily applied method of storage, the great disadvantage of such a system is that one relay is required to store each bit, and thus a memory with reasonable capacity would require a vast number of relays, becoming extremely bulky and expensive.

The magnetic core, by comparison, is an extremely compact device and comprises, basically, a ring or toroid made of a suitable magnetic material. It can be energized (i.e. "permanently magnetized") by the current carried by a single turn of wire around the coil. If the current flows in one direction the core will be magnetized in a clockwise direction, and if the current flow is in the opposite direction, in an anticlockwise direction (Fig. 67). The core will then remain

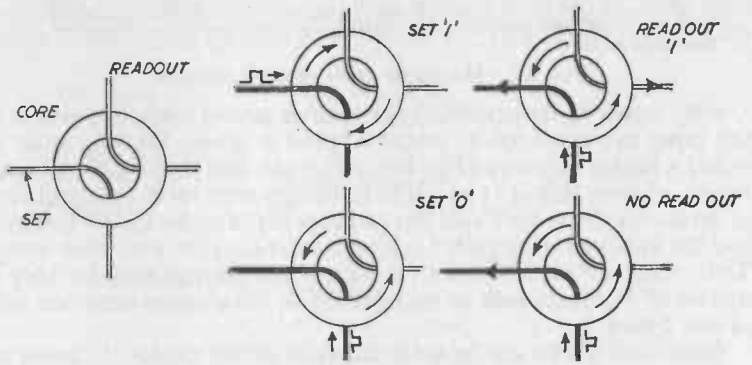


FIG. 67. PRINCIPLE OF CORE MEMORY SETTING AND READ OUT

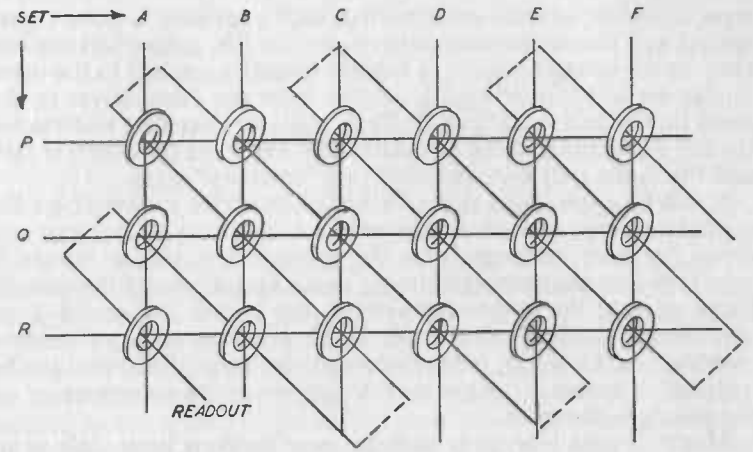


FIG. 68. A SIMPLE MEMORY CORE PLANE OR SCREEN

magnetized with this polarity until demagnetized by a current pulse through the coil in the opposite direction. Thus, it will store either a 1 or a 0, according to the direction in which it was originally magnetized.

The method of reading out the "bit" information is then as follows. The input is provided by one wire threading the core. A pulse of current will then magnetize the core in one direction, representing either a 1 or 0. Suppose a positive pulse represents a 1. A separate read-out wire also threads the core, and if a *positive* pulse is applied to the read-out wire this will have no effect. If a *negative* pulse is applied this will demagnetize and *reverse* the magnetism of the core, changing the core state from a 1 to a 0 and producing an output.

Each core provides a memory for one bit, and a large number of individual cores can be assembled in the form of a screen or *plane* in the manner shown in Fig. 68, each one being located by its grid reference, e.g. AQ, representing the core defined by the intersection of A (vertical) and Q (horizontal). The physical size of such cores is extremely small, rather less than $\frac{1}{16}$ in. diameter on average, and so a 10 in. square panel will accommodate something like 10,000 cores. Thus, one plane of this size would have a storage of 10,000 bits, and 100 planes together 1 million bits. Equally, with a "memory density" of approximately 100 bits per square inch, the size of each plane can be reduced, and the number of planes selected to cover computer words of the required number of bits together with a leading plane for the sign bit and a final parity bit plane.

To store the memory or bit in the appropriate core a core-plane driver is employed. The principle of this is shown in Fig. 69. Two

separate driver sources are involved, each providing a pulse representing *half* the current necessary to energize (i.e. magnetize) any one core. In the example shown, ½ current (pulse) is applied to the cores on the vertical line *H*; and ½ current from the other driver to the cores on the horizontal line *D*. Thus, the *only* core that will receive the full current (half from one driver and half from the other) is *DH*, and this is the only core switched to a "memory" state.

It will be appreciated that to achieve full access to every core for memory storage one driver is needed for each vertical line and one driver for each horizontal line. In practice, a switching matrix is used to reduce the number of driver circuits required and this usually forms part of the addressing system that locates the stored data. The circuits involved are complex and beyond the scope of amateur construction. However, memory core planes are available and can be included in amateur designs with simple pulse drivers operating on the principle discussed.

Magnetic core storage is used by most modern large-scale computers, although there are advances on the original form of construction. The latest forms of magnetic cores are based on plates of ferro-magnetic material, with small holes punched in them and the magnetic field circulating in the plate around each hole.

Although memory storage is an essential feature of the working of a typical commercial computer such devices are largely outside the scope of amateur construction, largely because of the cost involved. There is no reason, however, why simplified methods should not be employed for input storage, e.g. a punched tape or magnetic tape for providing a "programmed" input to any of the simple computers described. Input and output devices are more fully described in the next chapter.

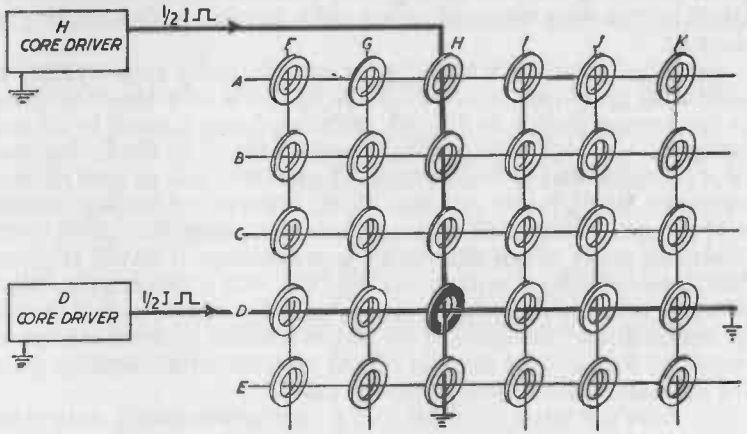


FIG. 69. CORE DRIVERS USED WITH A CORE PLANE

CHAPTER 10

INPUT AND OUTPUT DEVICES

MUCH of the advantage offered by a computer is lost if the methods of feeding data into it is slow or complicated. Similarly, it is desirable that the output be displayed or "read out" in an easily understood form. Unfortunately, to provide for high-speed fully automatic working considerable complication is usually involved. This does not matter in commercial computers, where the extra cost is more than justified by the results achieved, but does limit the scope offered by simpler circuits. Amateur construction, generally, stops with the arithmetic or logic block.

However, this does not necessarily mean that computer working is limited to manual operation and slow speeds. The digital computer can work on any information supplied in pulse form and can work as fast as the pulse rate (consistent with the limits of the computer circuit time constant). Thus, in the practical form of binary counter described in Chapter 3 the time constant is such that the circuit can respond to input pulses far faster than can be applied by manual operation of the respective set keys. To use waveforms as input, a pulse-shaper circuit was also described in Chapter 3, thus considerably expanding the input possibilities of the counter. The same comment applies to the operation of the basic flip-flop circuit, in association with various logic gates, to produce a computer giving logic solutions rather than arithmetical solutions.

The waveform input (modified to suit the circuit characteristics by being passed through the pulse shaper) can be of a variety of forms. Thus, coupled to a waveform generator the counter would count the number of waves or pulses in a given time, the limitation to usefulness here being the number of binary stages necessary to deal with larger numbers. Also, the count is rendered in binary digits, which need converting to equivalent decimal numbers to give a "real" answer. Thus, for high-speed waveform counting the decade scaler of Chapter 4 is a more practical proposition, capable of accommodating a much higher total "count," and also rendering the answer in decimal figures through the electro-mechanical counter unit coupled to the output (or read directly from the position of the glow discharge on the various counter tubes).

Manual input via pushbuttons or switches is undoubtedly the simplest and most straightforward input method for putting numbers into a computer, and it is widely used on commercial computers for

arithmetical work. It is certainly the simplest, and cheapest, method for amateur computer working with the aforementioned binary counter. Any extension of service can then be put into increasing the number of binary stages, and, thus, the capacity of the unit.

For simple computer working, i.e. the type of computer to which the amateur constructor is usually limited, a telephone dial can make an excellent input device, particularly as it automatically converts decimal numbers into binary pulses. Thus, if a telephone dial is connected into the input circuit of a binary counter, numbers to be added or subtracted can be dialled directly into the computer. The only necessary precaution would be to see that the "forward" and "backward" switch was in its correct position for addition and subtraction.

Assuming addition was required, and the sum was $7 + 4 + 6$, these numbers (7, 4 and 6) could be dialled one after the other, feeding a total of $7 + 4 + 6 =$ digits into the computer. The answer would be in binary numbers (10000), but the original numbers would not have to be converted from decimal to binary numbers (or signalled as single manual pulses) as the dial does this automatically. Also, to accommodate subtraction at any step the computer switch is moved to "backward" counting before dialling the number to be subtracted.

One practical point must be noted when using a telephone dial as an input device. Most dials do, in fact, transmit one more digit (pulse) than the dialled number, i.e. the first quarter rotation of the dial transmits an extra pulse before the 1-hole reaches the finger stop. This can be checked by examining the make-and-break action on the back of the dial and either modifying it to eliminate the extra pulse or, more simply, repositioning the finger stop to restrict the rotation and this prevent the unwanted extra pulse being made when dialling any number.

More complex problems may need *programming* (*see* Chapter 11), and speed or other requirements of working may demand the use of input devices other than manual operation of single-stage switches. All this really implies is that in digital computers the input data must be rendered in digital form—"on-off", "yes-no", "plus-minus", etc. In some cases the data is digital, e.g. waveforms representing "plus-minus," when it can be used directly as input (via a pulse shaper, if necessary). In other cases the data may be more obscure, when it will need breaking down into digital form (programming). If the data is in analog form then it will have to be converted into digital form before it can be used in a digital computer. On the other hand, analog data can be fed directly into an analog computer.

The memory devices described in Chapter 9 all provide digital data and thus can be used to provide the input in the form of a series of signal pulses represented by the coding adopted, and rendered in digital form by the reader. It is merely a matter of ensuring that the

coding conforms to the computer circuit functions and capacity. The modern commercial computer extends this facility to a remarkable degree. Thus, besides reading punched tapes, magnetic tapes, etc., readers can be devised to interpret lettering printed in magnetic ink in terms of digital data (cheque-reading computers), or even to read actual printed letters and words in terms of bits. If the amateur constructor has a matter of £10,000 or so to spare, this is a fruitful field for further experiment.

More realistically, there is no reason why the simple binary counter or its development with logic gates cannot be operated from an input programmed on an ordinary tape recorder, bearing in mind that the data needs only to be in pulse or "on-off" form. This is merely another application of basic logic. The computer circuit will have been designed to work on a logical basis, and the input must be rendered in similar logic *within the capabilities of the circuit*. Thus, it is no good feeding AND and OR signals into a logic network which has a capability of dealing with, say, only OR logic.

The application of *measured* data to a digital computer network is a little more difficult, for such a computer counts instead of measures. Yet measurement is a very convenient method of representing quantities, e.g. by meters. Measured quantities like this are, of course, analog data, and are very commonplace. To meet the demands of digital computers meters of various types (i.e. capable of measuring different quantities, such as volts or amperes) are, in fact, now produced capable of rendering readings in digital units and providing a corresponding output in the form of digital pulses. Naturally they are more expensive than simple analog meters, but they can be coupled directly to digital computers. The only alternative when dealing with

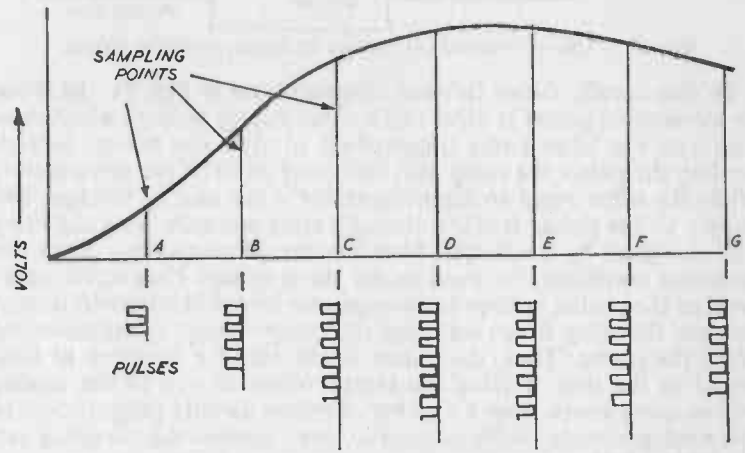


FIG. 70. ANALOG TO DIGITAL READING BY SAMPLING TECHNIQUE

analog data for feeding a digital computer is to convert the analog form of measurement to digital presentation or its equivalent.

A method by which this can be done is "sampling." Basically this implies using the analog value at any instant as a "time quantity" to drive a pulse generator. The pulse generator may be mechanical (e.g. a form of clock), electro-mechanical or purely electronic. Thus, if the analog quantity is represented by the changing voltage form shown in Fig. 70, "samples" are taken at points *A*, *B*, *C*, etc. and the pulse generator activated at each point to provide a supply or "count" of pulses equivalent in number to the actual voltage at that point. Thus, conditions *A*, *B*, *C*, etc. are rendered in terms of digital data—so many pulses at each point. In this form the data can be accepted as input in a digital computer.

Although this may appear simple in principle, it is a little tricky to reproduce in practical working. With the speed likely to be required for simple computer work a mechanical (clock) pulser will be quite adequate. The difficulty which then arises is how to drive the clock for the exact amount of time equivalent to the analog voltage at the sampling point. A simple way out of this is to call on the ubiquitous comparator circuit again (Chapter 5).

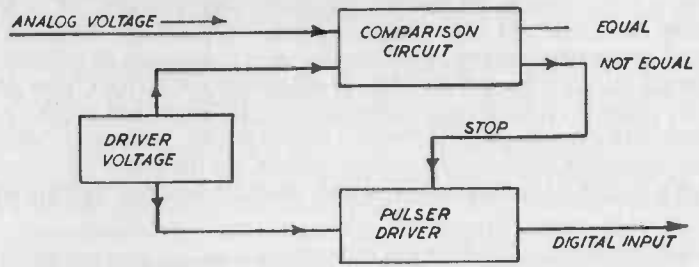


FIG. 71. USE OF COMPARISON CIRCUIT TO COMMAND PULSE DRIVER

In this circuit, shown in block diagram form in Fig. 71, the clock or mechanical pulser is driven by a ramp supply voltage which continues to rise after being triggered-on to start the pulser. Besides feeding the pulser the ramp also feeds one input of the comparator, while the other input to the comparator is the analog voltage. The supply to the pulser is taken through relay contacts, with the relay coil energized by the output from the comparator circuit in an unbalanced condition. As soon as the ramp voltage rises to the same level as the analog voltage the comparator circuit is triggered to zero output, the relay drops out, and the ramp voltage is disconnected from the pulser. Thus, the pulser drives only for a period of time equal to the time it takes the ramp voltage to rise to the analog voltage, and hence gives a number of pulses directly proportional to the analog voltage. With automatic reset, continuous sampling can be achieved, if required.

Output devices are, basically, input devices worked the opposite way round, and their choice is usually based on convenience, or, in amateur construction, expense. Indicator lamps, rather than meters, are usually the simplest and most satisfactory since they need draw very little current if coupled to a transistor amplifier (*see* Chapters 3 and 5).

Output will be in digital form, and in commercial computers quite complex translaters may be employed to provide final output, through electro-mechanical devices, in directly intelligible form. Thus, a *printer* may be used to render the digital (output) information in standard type form—decimal figures or letters. In essentials this is nothing more than a typewriter adapted to respond to digital signals following a predetermined code, but it is constructed so that it can operate very much faster than a conventional typewriter. The fastest are wheel-type printers, which can print at a rate of over 100,000 characters a minute, the equivalent of some three pages of normal single-spaced typing a second.

CHAPTER 11

COMPUTER PROGRAMS

ALTHOUGH most computers operate on digital logic, and where necessary can be given a "memory" capacity, they cannot themselves think or *reason*. Equally, they can only work on a step-by-step basis and cannot work at all until fed with the necessary instructions or input signals. Presentation of the necessary information for completing the step-by-step process is called *programming*, and the actual sequence involved, a computer program, which may include both the input and output.

The program is obviously influenced by the design of the computer. The problem of the program designer or *programmer* is to break down the instructions into a series of simple steps which can be represented by 1 or 0 in the form of a *flow chart*, representing the full path of the data through the various computer sections. The simpler the computer the simpler the program required. Thus, in the binary counter the only programming required is to set up the numbers to be added or subtracted on the respective set switches, in turn, having first converted these into binary numbers and read the answer in binary numbers, and to convert them back to decimal numbers. Programming, in fact, becomes significant only when the computer incorporates a memory storage and the solution to be extracted demands temporary storage of parts of the problem, broken

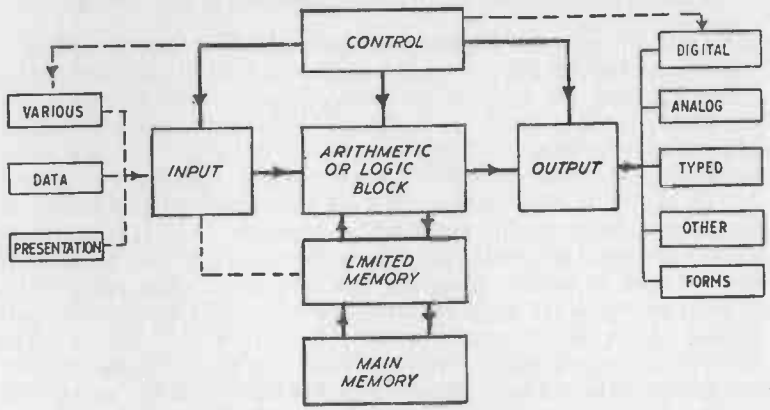


FIG. 72. TYPICAL FULL-SCALE COMPUTER SHOWN IN BLOCK DIAGRAM FORM

down into steps, or drawing on permanent storage for specific information.

This difference can be exemplified by comparing the programming of an elementary arithmetical computer and a more complex computer. The first involves only the arithmetic unit; the second a similar arithmetical unit with memory storage, input and output blocks (read-in and read-out) and a control block (Fig. 72).

Suppose the problem to be solved is the equation $X = A \times B$ where A equals, say, 2 and B equals 6. In the simple arithmetical computer this involves setting up A as a binary number (10) on the respective keys and then repeating this operation 6 times.

Decimal	Binary	
2	10	
	10	(add once = × 2)
4	100	
	10	
6	110	(add twice = × 3)
	10	
8	1000	(add three times = × 4)
	10	
10	1010	(add four times = × 5)
	10	
12	1100	(add five times = × 6)

To deal with the same problem the more complex computer may be programmed like this—

(i) Take number A from permanent storage address 10 and send to arithmetic unit.
(ii) Command "multiply by 6" (either via shift register and/or repetitive addition).
(iii) Transfer answer to output.
(iv) Justify.
(v) Return original number to permanent storage.
(vi) Read out answer as a decimal number.

Obviously, with such a simple problem the process is more complicated and far more computer function is being employed than need be. These facilities only benefit when the problem is more complex and beyond the capability of a simple arithmetical unit to handle on its own without a "memory" facility. Compare the difference, for example, if the problem involved a quadratic equation—

$$Y = AX^2 + BX + C.$$

The simple computer could give an answer for $X \times X = X^2$ by adding X "X" times; then AX^2 by adding the number obtained A times. It could find BX by adding B "X" times. A further operation would add AX to BX. It could then add C to complete the equation. This would mean, however, extracting or reading out six separate answers from the computer: $X^2, AX^2, BX, AX^2 + BX, (AX^2 + BX) + C$.

The more complex computer would probably be programmed like this—

(i) Read A,B,C and X into the memory unit.
(ii) Take X from the memory and transfer to the main register as a binary number.
(iii) Multiply X by X.
(iv) Multiply (iii) by A and store in temporary memory.
(v) Read B into main register as a binary number.
(vi) Multiply by X to give BX.
(vii) Extract AX^2 from the memory store and add to BX in the main register.
(viii) Extract C from the memory as a binary number.
(ix) Add to (vii) to give Y.
(x) Read out Y as a decimal number.

More stages, but the whole process can be accomplished in the computer circuits in a fraction of a second. Equally, the answer Y could be stored for future use and, fed initially with enough data to give solutions for Y for all possible combinations of A,B,C and X, when required answers to an appropriate value of Y are immediately available from the memory storage without having to go through the intermediate steps. The program then merely has to provide the correct address consistent with the given values of A,B,C, and X.

Exactly similar considerations apply to logic as well as arithmetical computers. In addition to solving individual problems on a step-by-step basis, solutions for specific combinations of logic can be stored permanently to be called upon virtually instantaneously at any future date, provided the storage is of permanent type. Thus, punched tape or magnetic tape or storage drum data is stored permanently. Magnetic core data is destroyed once a core is triggered, unless the core is re-pulsed to restore it to its "memory" condition after being triggered by the read out.

As noted previously, however, memory storage is something that is virtually an economic impossibility in amateur computers, but there is quite a lot that can be done to extend the scope of even simple computers, particularly those working on logic gates, even without storage. At the same time, a knowledge of what "memory" is used for, and storage capability, is essential for an appreciation of the working of the more elaborate commercial computers and the significance of programming.

Most modern computers work on a stored-program basis, which

means, in effect, that all the necessary digits or characters have already been fed into the machine and retained in a memory unit. The process of programming then involves directing the control unit to the right address, bringing the data to the arithmetic or logic unit, and subjecting it to the necessary step-by-step processes on the lines described above. An address as such is only a location in the storage unit, but each location would normally hold a computer word complete, which may be either a number or an instruction. When called up, the control unit has to decode this, act as necessary, and proceed to the next stage.

With the elementary types of computers suitable for amateur construction the operator has, basically, to take the place of the memory and the control unit. On the other hand, in a "logic" computer concerned with deriving a logical answer rather than arithmetic solutions, the computer will have to be designed around the program in order to deal with the input data at all. This applies whether the computer is a very elementary one, or an extremely complex machine. And in digital computers both programming and logic circuit design involve thinking in terms of binary numbers. With an arithmetical unit programming is restricted to feeding in digital signals, plus whatever other control may be necessary (e.g. operation of the forward and backward switch on the binary adder/subtractor of Chapter 3, for addition or subtraction, as required).

CHAPTER 12

BINARY COMPUTER CONSTRUCTION

THE complete theoretical circuit for a binary adder/subtractor has been described in Chapter 3. This chapter is devoted to describing the same circuit in terms of physical layout and construction for those readers who find it difficult to translate theoretical circuits into physical layouts. Layout description is confined to a single stage. Thus, to build up any number of required stages it is only necessary to duplicate the single stage the required number of times, end to end. This can most conveniently be done by extending the main bus bars or common lines—+6 volts, 0 (or earth) and −6 volts on the layout panel. Alternatively, each stage may be built on a separate panel, if preferred. Single panel construction is also useful for adding further stages at a later date.

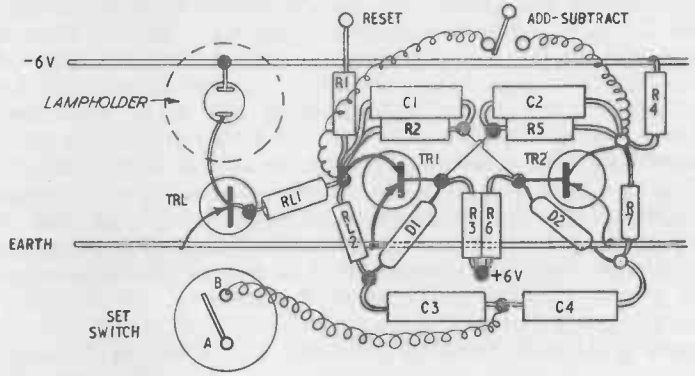


FIG. 73. TAGBOARD ASSEMBLY FOR CONSTRUCTION OF BINARY COUNTER STAGE

*R*1	1kΩ	*C*1	6·4μF
*R*2	4·7kΩ	*C*2	6·4μF
*R*3	220kΩ	*C*3	0·22μF
*R*4	1kΩ	*C*4	0·22μF
*R*5	4·7kΩ	*D*1	OA85 or OA81
*R*6	220kΩ	*D*2	OA85 or OA81
*RL*1	6·8kΩ	*TL*	OC72 or OC83
*RL*2	2·2kΩ	*TR*1	OC71
		*TR*2	OC71

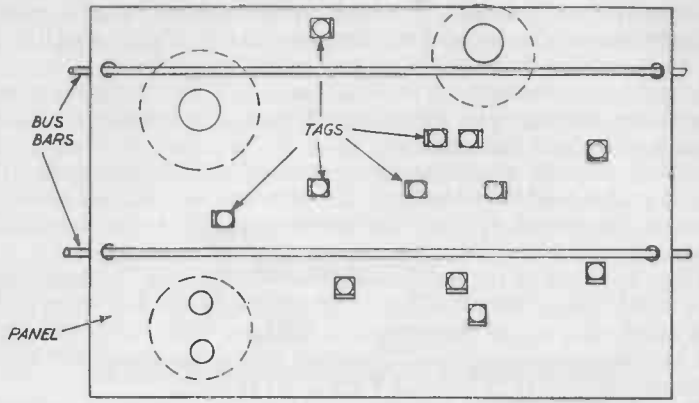


FIG. 74. PANEL FOR ASSEMBLY SHOWING BUS BARS AND SOLDER TAGS MOUNTED

The basic circuit is adaptable to tagboard or printed circuit construction. Both will be described. Another alternative is "Veroboard" construction, which is a form of prefabricated printed circuit board. Layout on a "Veroboard" panel can be based on the printed circuit construction.

Fig. 73 shows the component layout for tagboard construction, approximately actual size, with component values listed. Fig. 74 shows the tagboard complete with bus bars and tags mounted in place ready for soldering the components in place. The board itself is a Paxolin panel approximately 3 in. by 2 in. The tags are riveted in place in the positions shown, or they can be bolted in place. The main bus bars are of 16 gauge tinned copper wire soldered to the appropriate tags. Components are then soldered to the remaining tags to conform to the layout shown in Fig. 73.

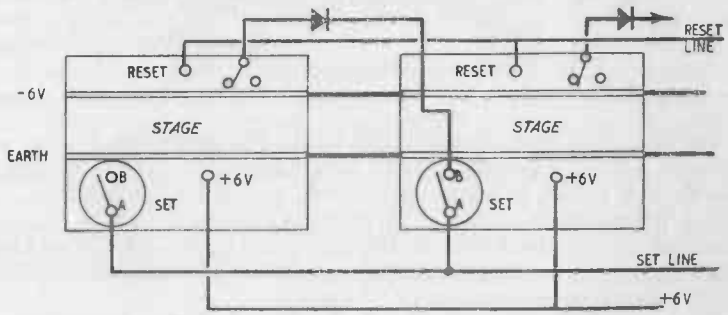


FIG. 75. INTERSTAGE CONNEXIONS FOR BINARY ADDER/SUBTRACTOR

Switches (for "set" and "forward" or "backward" counting) and the bulb holder are mounted on the opposite side of the panel, i.e. the plain side, which is the "top" or face of the assembly.

Interstage connexions are made as shown in Fig. 75 with insulated wire soldered to the appropriate tags. Note that the output from each stage is taken from the common side of the "forward" or "backward" switch via a diode, which should be soldered directly to the tag. The output lead should be soldered to the other end of the diode and then taken to the centre of the multivibrator circuit on the next stage (physically, the normally open side of the "set" switch).

The other side of the "set" switch is "commoned" in each stage, i.e. a wire is taken from this side of the switch on the first stage to the next stage, and so on through all the stages. This forms the "set" line in the complete circuit. After the *final* stage the "set" line is connected to +6 volts bus bar via a 1 kΩ resistor.

The other "commoning" connexion to be made between stages is the "reset" line. This is taken from the appropriate tag, connecting the same tag stage by stage and, finally connecting it to the reset switch itself after the final stage. The other side of the reset switch connects to −6 volts bus bar.

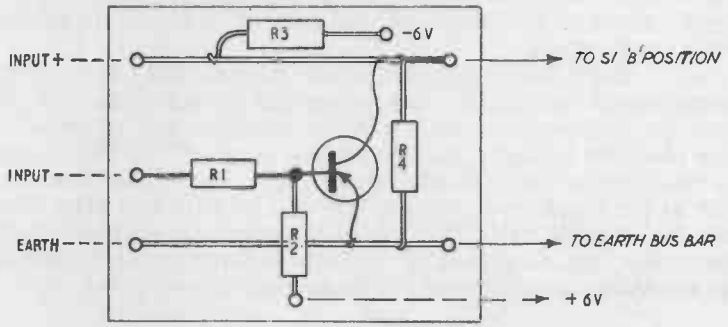


FIG. 76. COMPONENT LAYOUT FOR PULSE-SHAPER CIRCUIT

*R*1	15kΩ	*R*3	2·2kΩ
*R*2	220kΩ	*R*4	330kΩ

TR OC71

The circuit is then complete and ready for connexion to the +6 volts, 0, −6 volts battery supply, which should be taken through a double-pole switch to act as an "on-off" switch for the battery.

If a pulse-shaper circuit is to be incorporated it can be accommodated at the front end of the first stage (Fig. 76). Otherwise, input signals are applied via the appropriate set switches, as described in the operating instructions in Chapter 3.

For printed circuit assembly the layout can be made somewhat more compact. However, for convenience, and to avoid overcrowding, a similar layout can be adapted. With printed circuit assembly the bulb-holder and switches can be mounted on the same side as the components, but they are preferably mounted on a separate face panel. The latter arrangement will usually be neater and more convenient since the separate panel can then form the face of the cabinet or box accommodating the complete circuit.

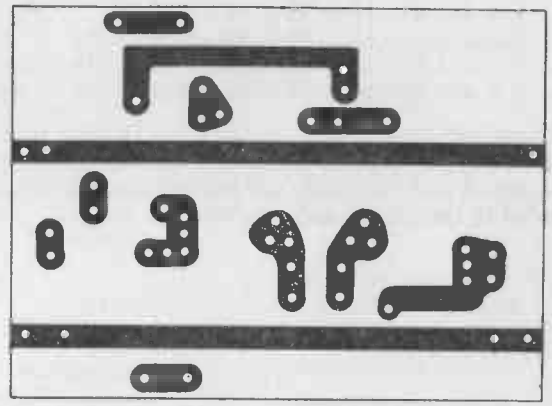


FIG. 77. PRINTED CIRCUIT PATTERN FOR BINARY COUNTER STAGE

The printed circuit layout (copper side) is shown in Fig. 77. This is a full-size pattern for tracing and transfer to printed circuit stock for preparation of the actual circuit. It does, however, cover only one stage, requiring interstage connexions by flying leads, as described above for tagboard assembly. In a practical design it would be neater and better practice to accommodate all the stages on a single panel (or as many stages as convenient on a given or required panel length), and also to accommodate the "reset" and "set" lines on the panel. Thus, the printed circuit layout would follow the full theoretical diagram of Fig. 75.

It should be emphasized that there is no limit to the number of stages that can be added on to increase the capacity of the counter, except the expense involved. However, for practical use a *minimum* of four stages is recommended, which will give a total count of 15. Seven stages are a preferred minimum for general use since this will increase the count to 64 + 32 + 16 + 8 + 4 + 2 + 1 = 127. The actual number of stages required for a particular coverage can be decided on the following basis—

Number of stages	Total count
1	1
2	$1+2=3$
3	$1+2+4=7$
4	$1+2+4+8=15$
5	$1+2+4+8+16=31$
6	$1+2+4+8+16+32=63$
7	$1+2+4+8+16+32+64=127$
8	$1+2+4+8+16+32+64+128=255$
9	$1+2+4+8+16+32+64+128+256=511$
10	$1+2+4+8+16+32+64+128+246+512=1{,}023$

and so on.

From this it will be seen that to accommodate decimal numbers up to a total count of one thousand (and just slightly beyond) ten stages will be needed in the binary adder/subtractor.

INDEX