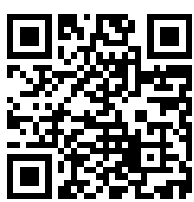
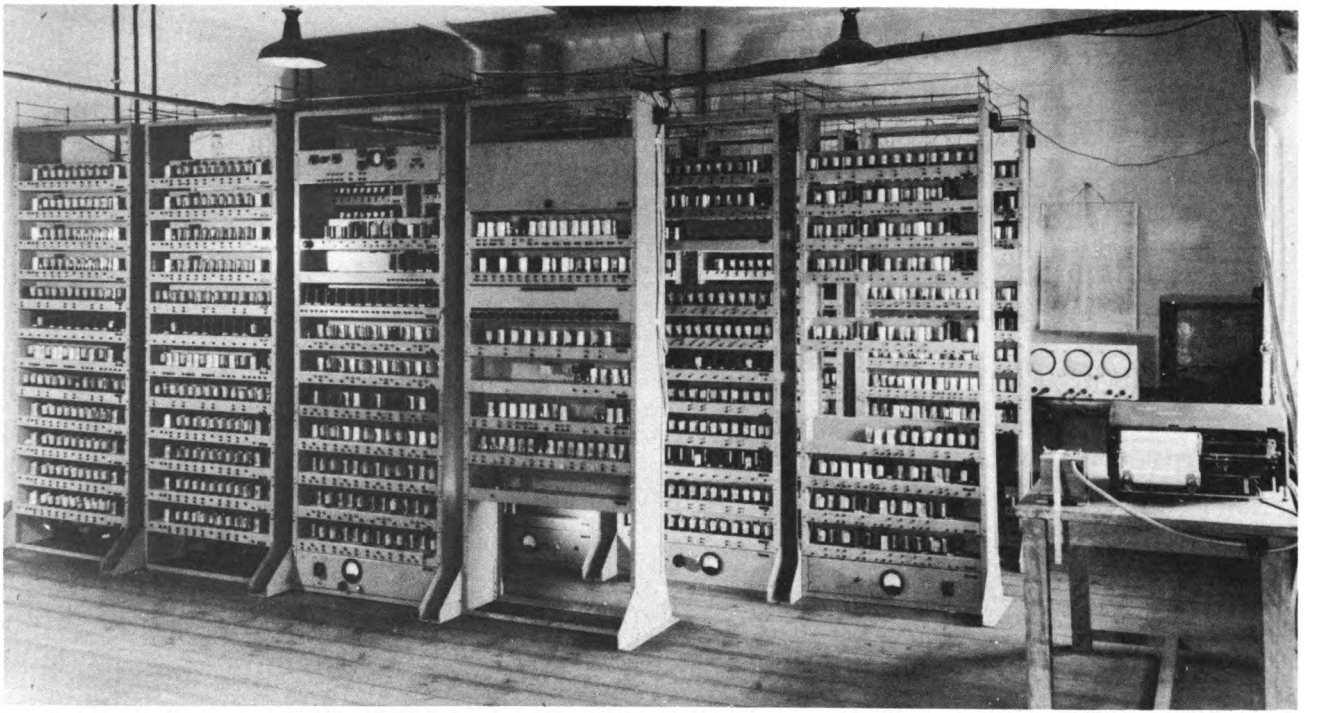
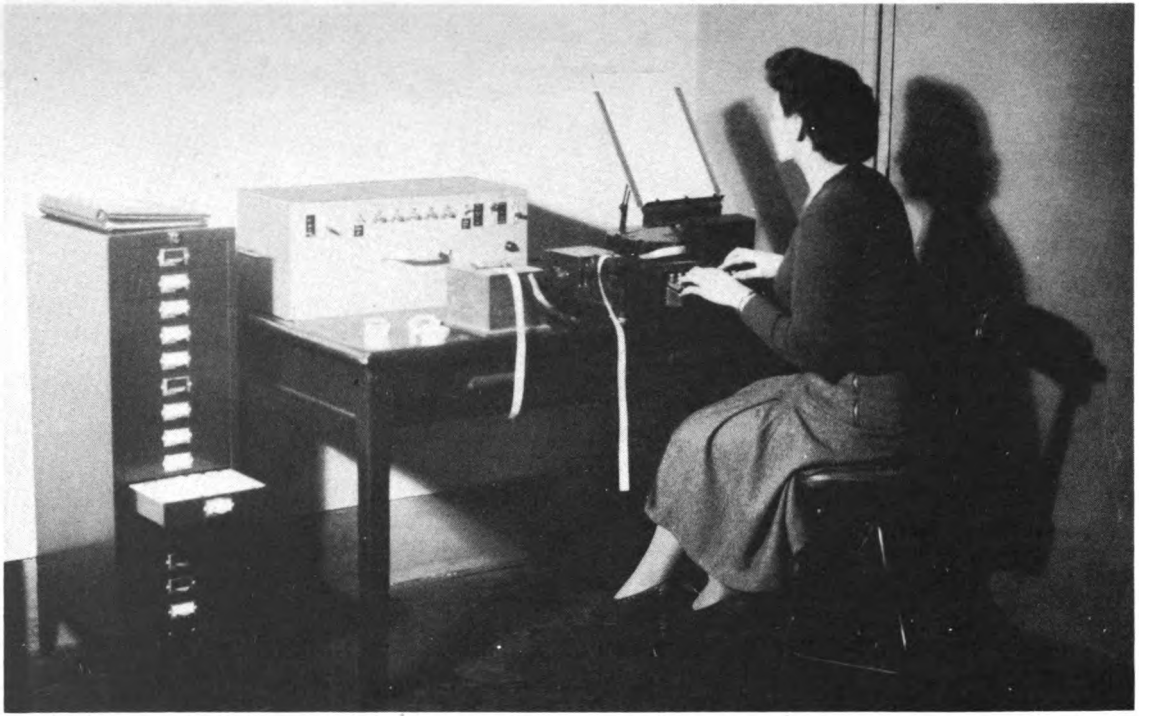


THE PREPARATION OF PROGRAMS FOR AN ELECTRONIC DIGITAL COMPUTER



A general view of the EDSAC. The racks in the front row contain (from left to right): part of the store (two racks), pulse generator, and input-output units. Behind are three racks containing the control, and, in the rear, the remainder of the store (two racks) and the arithmetical unit (three racks). On the extreme right of the photograph may be seen the tapereader for the input tape, and the teleprinter on which results are printed.



The library of tapes on which subroutines are punched is contained in the steel cabinet shown on the left. The operator is punching a program tape on keyboard perforator. She can copy mechanically tapes taken from the library on to the tape she is preparing by placing them in the tapereader shown in the center of the photograph.

THE PREPARATION OF PROGRAMS FOR AN ELECTRONIC DIGITAL COMPUTER

With special reference to the EDSAC and the use of a library of subroutines

by

MAURICE V. WILKES
Director of the Mathematical Laboratory of the University of Cambridge

DAVID J. WHEELER
and
STANLEY GILL

1951

ADDISON-WESLEY PRESS, INC.
CAMBRIDGE 42, MASS.

TABLE OF CONTENTS

PART I

PREFACE

The methods of preparing programs for the EDSAC described in this book were developed with a view to reducing to a minimum the amount of labor required, and hence of making it feasible to use the machine for problems which require only a few hours of computing time as well as for those which require many hours. This necessitated the establishment of a library of subroutines and the development of systematic methods for constructing programs with their aid. The methods are described in terms of the code of orders used in the EDSAC, but for the main part they may readily be translated into other order codes. It is hoped, therefore, that those who have charge of similar machines, or who are faced with the task of putting a new machine into operation, will find some of the ideas and methods presented here of assistance. It is hoped also that the book will be of use to those who wish to know something about the form in which problems are presented to an automatic digital calculating machine and who wish to assess the possibilities of the application of such machines to their own subjects.

Many workers in the Mathematical Laboratory have contributed to the development of the methods described in this book. We would mention especially the following: J. M. Bennett (to whom the material described in Appendix D is due), R. A. Brooker, E. N. Mutch, B. Noble, J. P. Stanley, and B. H. Worsley. To this list we would add the name of Professor D. R. Hartree, F.R.S. who has also very kindly contributed a foreword to this book. The following also assisted in the formation of the library of subroutines: K. N. Dodd, L. A. G. Dresel, A. E. Glennie, E. E. C. McKee, and C. M. Munford. We are especially grateful to Mr. Mutch for his assistance with the editorial work, in particular for undertaking the heavy task of preparing Parts II and III for the press.

We are deeply conscious of the debt we owe to our colleagues engaged on other projects, especially to those who were instructors of a course attended by one of us at the Moore School of Electrical Engineering, University of Pennsylvania, in 1946, and to Dr. J. von Neumann and Dr. H. H. Goldstine of the Institute for Advanced Study, Princeton, whose privately circulated reports we have been privileged to see.

We would also like to express our gratitude to Dr. Z. Kopal for the help he has freely given in proofreading and in seeing the book through the press. We are most grateful to the publishers and their staff for the care that they have taken and for the rapidity with which they have done their work.

M. V. W.
D. J. W.
S. G.

Cambridge, England
March, 1951

FOREWORD

by Professor D. R. Hartree, Ph.D., F.R.S.

To the potential user of an automatic digital calculating machine, the successful design and construction of the machine itself is only a first step, though certainly an essential one. In order that the machine should in practice be useful to him in the calculations he may desire to carry out with its aid, the provision of an adequate organization for using the machine is as important as the machine itself.

One form of such an organization is based on a library of subroutines for carrying out standard processes, and facilities for using it. Provision of such a library has two important effects. First, it relieves the user of the machine of the greater part of the work of programming calculations in detail; a library subroutine can be incorporated as a unit in his program, without it being necessary for him to work through the sequence of operations by which the calculation carried out by the subroutine is effected; and it is quite possible for eighty per cent of a complete program to be carried out by the use of such library subroutines. And secondly, in making up a program, use of library subroutines which have been thoroughly checked limits the possibilities of mistakes in programming, and correspondingly reduces the expenditure of time, both of the machine and of the programmer, in diagnosing and correcting mistakes. In order that such a library of subroutines should be practically useful, it seems desirable, if not indeed necessary, that the subroutines should be drawn up in a form which provides a certain amount of flexibility in their use, so that slight variations can be made in them in order to suit the contexts in which they may be used in particular applications.

The process of building up such a library of subroutines, and testing its value by practical use, appears to have proceeded further at the Mathematical Laboratory of the University of Cambridge than elsewhere, and in this book the authors, who together have been primarily concerned in this development, give an account of the present state of this aspect of the study of means of using an automatic calculating machine effectively. It is the result of a considerable amount of exploratory work on such matters as ways in which to specify operating instructions to the machine, and to draw up subroutines, so as to give the required flexibility, ways in which to enter and leave subroutines, and different types of subroutines.

The results of this work do not provide a unique system, nor are they to be regarded as forming a perfect one; they depend on the order code and other features of the functional design of a machine which were decided already two years ago, before some of the developments in programming had been envisaged. But that it is a practical and useful system has been tested by experience; it divests programming of the appearance of being something of a magic art, closed except to a few specialists, and makes it an activity simple enough to be undertaken by the potential user who has not the opportunity to give his whole time to the subject.

The subject is one which is still developing, but the authors are, I think, to be commended for drawing up this account of the present stage of their contribution to it, both in general ideas and in details, and so making this work available to others working in this same field.

D. R. Hartree

Cavendish Laboratory
Cambridge, England

January, 1951

PART I

CHAPTER 1

THE DESIGN OF PROGRAMS FOR ELECTRONIC COMPUTING MACHINES

1-1 Introduction.

A digital computing machine can perform only the basic operations of arithmetic, namely, addition, subtraction, multiplication, and division. In order to be able to solve a mathematical problem such as the integration of a differential equation it is first necessary to express the problem as a sequence of such operations. This may call merely for some expenditure of labor or it may involve considerable mathematical manipulation; for example, where derivatives or integrals are involved it may be necessary to replace the continuous variables by variables which change in discrete steps.

If the computation were to be performed by a human computer it would be possible to communicate the problem to him in a series of instructions or orders, each specifying an elementary arithmetical operation. It is convenient to use the same nomenclature when speaking of a machine but here the "instructions," or "orders," are groups of symbols punched on a paper tape or prepared in some other form which can be fed into a machine. A sequence of orders for performing some particular calculation is called the program. It must contain everything necessary to cause the machine to perform the required calculations and every contingency must be foreseen. A human computer is capable of reasonable extension of his instructions when faced with a situation which has not been fully envisaged in advance, and he will have past experience to guide him. This is not the case with a machine.

Since an automatic computing machine can perform only a very limited number of basic operations, the simplest mathematical calculation requires an extended sequence of orders. The labor of drawing up a program for a particular problem is often reduced if short, ready-made programs for performing the more common computing operations are available. These short programs are usually called subroutines, and they may be incorporated as they stand in the program, thus reducing the amount of work which has to be done ab initio. If it is intended that an electronic computing machine shall be used on a wide variety of problems it is worth-while to spend much effort on the establishment of an extensive library of such subroutines, together with a workable system whereby selected subroutines may be combined to form a program.

This book contains a detailed description of the library of subroutines used in the Mathematical Laboratory of the University of Cambridge in conjunction with the EDSAC (Electronic Delay Storage Automatic Calculator) and of the way in which programs can be constructed with its aid. There will be some discussion of the best way to construct subroutines for numerical quadrature, the integration of differential equations, and other processes, but the more theoretical problems that arise in numerical analysis are outside the scope of this book. Some of these, for instance those concerned with the convergence of iterative processes and with the accumulation of rounding-off

errors, are of great importance and interest and are likely to arise in acute form when planning the large-scale computing operations which an electronic machine makes possible. The present book, however, is concerned with the steps which must be taken to make the machine perform the numerical processes necessary to solve a problem when once it has been decided what those processes are.

There are naturally many ways, all similar in principle but differing in detail, in which subroutines may be used to construct programs, and no attempt will be made here to discuss all the possible alternative methods. It is hoped, however, that the account given of those at present being used with the EDSAC will be of general interest. The ideas and techniques described are applicable, with suitable adaptation, to other electronic calculating machines designed on the same general principles.

1-2 Types of automatic computing machines.

In large automatic computing machines which depend for their action on electromechanical devices the orders are usually punched in coded form on paper tape, one group of holes corresponding to each order. These holes are read by a sensing device and cause the machine to perform the operation called for; the tape is then advanced so that the next group of holes is under the reading head and the next order is similarly executed. In addition to a sensing mechanism for the main program tape, several other sensing mechanisms are usually provided. These can be used to read endless loops of tape which contain orders for performing parts of the program which have to be repeated a number of times. Control of the machine is passed from one tape to another as required. Machines which work in this manner are the Automatic Sequence Controlled Calculator at Harvard University, relay calculators built by the Bell Telephone Laboratories, the Aiken relay computer at Dahlgren, Md., and the IBM Selective Sequence Electronic Calculator.

Such a system, while admirable for controlling a relay machine, would not be fast enough for a machine in which the computation is performed by electronic means and in which it is desired to realize the very high speed which this makes possible. The ENIAC, which was the first purely electronic machine to be built, therefore used a system in which the various steps of the program were initiated by "program pulses" passed from one unit of the machine to another. For example, to cause a number standing in one register or "accumulator" to be added to the number standing in another accumulator, both accumulators needed to be stimulated by a program pulse, one to transmit and one to receive. When the operation was finished both accumulators emitted a pulse, and one of these (it did not matter which, since they both occurred at the same time) was used to stimulate the next action. Putting a problem on the machine consisted, therefore, of making a large number of connections by means of plugs and sockets and setting a number of switches. The main objection to this system is that it takes some time to change over from one problem to another. In all later machines, proposed or completed, the orders are expressed in a coded form and placed in advance in a quick-access store, or memory, from which they are subsequently taken and executed one by one. The orders are usually passed into the machine by means of a

punched tape, or some similar medium, but this is used simply as an intermediary in the process of transferring the program to the store, and does not control the computing action of the machine directly.

A store, or memory, is needed in automatic computing machines for the purpose of holding numbers, and in the EDSAC the same store is used to hold the orders; this is made possible by the device of expressing the orders in a numerical code. Several machines working on the same principles as the EDSAC are now in operation in the United States and in England. These principles derive from a report drafted by J. von Neumann in 1946 in connection with a new machine (the EDVAC) then projected by the Moore School of Electrical Engineering (University of Pennsylvania) where the ENIAC had been built. It is found that machines designed along the lines laid down in this report are much smaller and simpler than the ENIAC and at the same time more powerful. The methods by which programs are prepared for all these machines are, as might be expected, similar, although the details vary according to the different order codes used. Anyone familiar with the use of one machine will have no difficulty in adapting himself to another. All machines so far completed use the binary system for internal calculations but this is not an essential feature and several machines under construction use the decimal system. Even if the binary scale is used inside the machine, it is only rarely that the programmer needs to take notice of this fact, since input and output can be performed in the scale of ten, the necessary conversion being done by the machine itself as part of the program.

1-3 Description of the EDSAC.

In order to be able to construct programs, some knowledge of the main units of the machine and their interconnection is required, although it is not necessary to understand the precise mode of functioning of the various electronic circuits. There are, from the point of view of the programmer, four main parts to the machine: the store, or memory, the arithmetical unit, the input, and the output mechanisms. There is also the control unit which emits the electrical signals that control the action of the other units. Fig. 1 shows the connections between the various units. The store of the EDSAC, which is of the ultrasonic variety, was designed to have capacity for 1024 numbers of 17 binary digits each, although so far only half this capacity has been available. Negative numbers are represented inside the machine by their true complements and the most significant digit of any number is treated in the arithmetical unit as a sign digit. The sign digit is a zero if the number is positive and a one if it is negative. The 512 numbers are held in 512 "storage locations" numbered serially from 0 to 511 for reference purposes. The reference number of the storage location holding a number x is sometimes called the address of x. A special feature of the EDSAC is the possibility of combining any two consecutive storage locations (provided that the first has an even serial number) into a single long storage location capable of holding a number with 35 binary digits, one of which is a sign digit. Such a number is called a "long number" to distinguish it from a "short number" of 17 binary digits. It is possible to accommodate 35 digits in a long storage location, and not 34 only, since in the ultrasonic store of the EDSAC the digits of successive numbers are stored end to end and one digital position between each is left unused;

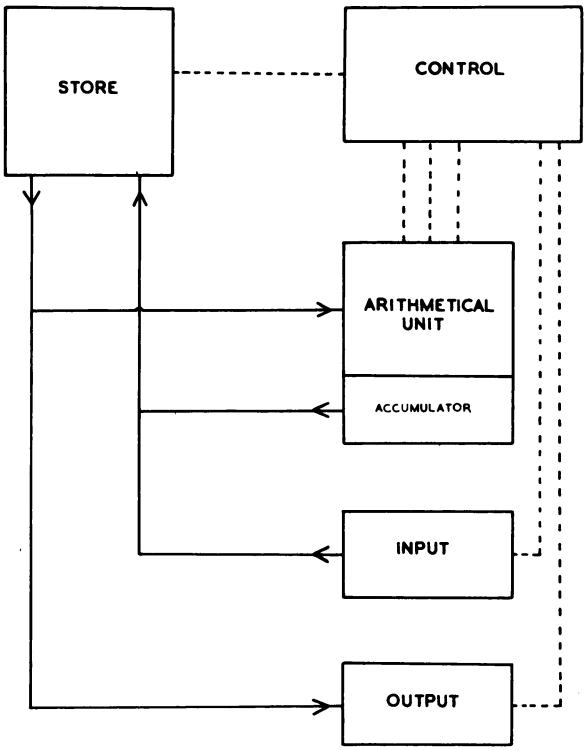


Fig. 1 Schematic diagram of the EDSAC

when two storage locations are combined this position can be used to contain an extra digit (sometimes called the "sandwich" digit). It may be noted that a long number contains the equivalent of about ten decimals and a short number the equivalent of about five decimals.

The arithmetical unit may best be described as being an electronic version of an ordinary desk-type calculating machine. In it the operations of addition, subtraction, and multiplication may be performed; there is no divider in the EDSAC and the means used for performing division will be described later. The arithmetical unit contains an accumulator register, in which the results of additions, subtractions, and multiplications appear and in which a series of such results may be accumulated. There is another register which is used to hold the multiplier during the process of multiplication. The multiplier is so constructed that numbers are treated as though they lie in the range $-1 \leq x < 1$, that is, the binary point is assumed to come immediately to the right of the sign digit. The programmer should, therefore, rearrange the formulas before drawing up the program, so that all the quantities which need to be handled inside the machine are within the range $-1 \leq x < 1$. This may always be done if suitable positive or negative powers of two are introduced as multiplying constants; in the program these constants are represented by shift orders. An alternative procedure, although not one to be generally recommended, is for the programmer to adopt some other convention as to the position of the binary point and to program a shift after each multiplication; for example, if the binary point is assumed to be between the second and third digits to the right of the sign digit, each multiplication must be followed by a shift of 2 places to the left.

Five-hole punched tape, read by a photoelectric tape reader, is used for input to the EDSAC. All the orders and numbers required for the solution of a problem are punched on a single tape, which may, however, be divided into two or more pieces for insertion in the tape reader one after the other. Library subroutines are stored on separate short lengths of tape and copied

mechanically on to the program tape. The output mechanism is a teleprinter. Further information about the engineering of the EDSAC will be found in the papers listed on page 21.

1-4 The EDSAC order code.

The action of the machine proceeds in two stages; in stage I an order passes from the store into the control unit; in stage II the order is executed. The machine then proceeds automatically to repeat stage I, in general taking the order from the storage location following that containing the order just executed. An exception to this rule will be discussed in Section 1-6. Each order calls for one simple operation to be performed; for example, it may cause some number to be extracted from the store and added to whatever happens to be in the accumulator, the sum being left in the accumulator, or it may cause the contents of the accumulator to be transferred to the store. Some orders, for example left or right shift orders, do not involve the use of the store at all.

There are in the EDSAC code eighteen orders from which the programmer can build up his program. They are written in the form of a letter indicating the function of the order, and a number (the address) specifying the location (if any) in the store concerned. The address is followed by the code letter F if it refers to a short storage location, and by the code letter D if it refers to a long storage location. The full order code for the EDSAC is as follows:

Order Code

Where the code letter terminating an order is not shown it may be either F or D.

A n	Add the number in storage location n into the accumulator.
S n	Subtract the number in storage location n from the accumulator.
H n	Copy the number in storage location n into the multiplier register.
V n	Multiply the number in storage location n by the number in the multiplier register and add the product into the accumulator.
N n	Multiply the number in storage location n by the number in the multiplier register and subtract the product from the accumulator.
T n	Transfer the contents of the accumulator to storage location n and clear the accumulator.
U n	Transfer the contents of the accumulator to storage location n and do not clear the accumulator.
C n	Collate the number in storage location n with the number in the multiplier register and add the result into the accumulator; that is, add a "1" into the accumulator in digital positions where both numbers have a "1", and add a "0" in other digital positions.
*R D	Shift the number in the accumulator one place to the right; that is, multiply it by 2^{-1}.

**R 2^{p-2} F	Shift the number in the accumulator p places to the right; that is, multiply it by 2^{-p} ($2 \leq p \leq 12$).
R F	Shift the number in the accumulator 15 places to the right; that is, multiply it by 2^{-15}
*L F	Shift the number in the accumulator one place to the left; that is, multiply by 2.
**L 2^{p-2} F	Shift the number in the accumulator p places to the left; that is, multiply by 2^{p} ($2 \leq p \leq 12$).
L F	Shift the number in the accumulator 13 places to the left; that is, multiply by 2^{13}.
E n F	If the number in the accumulator is greater than or equal to zero, execute next the order which stands in storage location n; otherwise proceed serially.
G n F	If the number in the accumulator is less than zero, execute next the order which stands in storage location n; otherwise proceed serially.
I n	Read the next row of holes on the input tape and place the resulting integer, multiplied by 2^{-16}, in storage location n.
O n	Print the character now set up on the teleprinter and set up on the teleprinter the character represented by the five most significant digits in storage location n.
F n	Place the five digits which represent the character now set up on the teleprinter in the five most significant places in storage location n, clearing the remainder of this location.
*X	Ineffective; machine proceeds to next order.
*Y	Round-off the number in the accumulator to 34 binary digits; that is, add 2^{-35} into the accumulator.
*Z	Stop the machine.

* The addresses in these orders need not be zero.

** The addresses in these orders may be $k \cdot 2^{p-2}$ where k is odd, provided that the addresses do not exceed 2047.

1-5 Notes on the order code.

As a simple example of the use of this code, suppose that it is required to evaluate the expression x+y+xy, taking x and y to be the contents of the short storage locations 50 and 51, and to place the result in the long storage location 52. A program for doing this is as follows (it is assumed that the accumulator is clear at the beginning):

```
A 50 F
A 51 F
H 50 F
V 51 F
T 52 D
```

The accumulator has sufficient capacity to hold a number having 71 binary digits, of which one is regarded as a sign digit. As in the store, the binary point is immediately to the right of the sign digit. When two long numbers are multiplied together the resulting 69 digits are all available in the

accumulator. A U order or a T order will, however, transfer only the 35 most significant digits (or if the order is terminated by an F, the 17 most significant digits) to the store, although a T order always clears the whole of the accumulator. If it is desired to retain all the 69 digits which are obtained by multiplying two long numbers together, then the 35 most significant digits must first be transferred to the store by means of a U order and the contents of the accumulator shifted 34 places to the left; the 34 least significant digits are then in a suitable position to be transferred to the store by a T order. Note that it is necessary to use three left shift orders, since in the EDSAC the number in the accumulator cannot be shifted more than 13 places to the left by a single shift order.

If an A order is used to add a number x from the store to the number y standing in the accumulator the correct answer will be obtained only if $x+y$ satisfies the condition $-1 \leq x+y < 1$. If this condition is violated the number appearing in the accumulator will by $x+y-2$ if $x+y \geq 1$, and $x+y+2$ if $x+y < -1$. In a similar way, if the effect of any other order is to cause the capacity of the accumulator to be exceeded, the number which actually appears in the accumulator is that obtained by adding or subtracting a suitable multiple of 2 from the correct result.

The C order (collation) is useful when it is required to pick out specified groups of digits from a number. For example, the first four binary digits, not including the sign digit, of a given number may be isolated by collating the given number with the number 0111100000000000000000000000000000, that is, 15/16.

The number placed in the multiplier register by an H order remains there until it is replaced by another number introduced by another H order. Thus if a series of numbers are to be multiplied by a constant, one H order only is necessary to transfer the constant to the multiplier register at the beginning of the operation.

1-6 The use of conditional orders.

An exception to the rule that the machine executes orders in the sequence in which they stand in the store occurs when a conditional order (E or G) is encountered. The action then depends on the sign of the number in the accumulator; if this is negative an E order causes the machine to pass straight on to the next order, while if it is positive or zero the next order is taken from some other location in the store. In the latter case control is said to be transferred to the new storage location. The action of a G order is similar, except that control is transferred if the number in the accumulator is negative. The following program for finding the absolute value of the number in storage location 123 illustrates the use of a conditional order.

Location of order	Order	Notes
		the accumulator is assumed to be clear at the start
301	A 123 F	the number in 123 is added into the accumulator
302	E 305 F	the sign is tested

303	T	F	if negative, the number in the accumulator
304	S	F	is changed in sign
305	T	F	the result is placed in location 0

Conditional orders, however, are much more important than this example would indicate, since they enable the programmer to cause a group of orders to be repeated a number of times and to transfer control from one section of the program to another. Conditional orders thus provide facilities equivalent to those obtained by the use of endless loops of tape on the machines mentioned earlier. The following example shows how the operations called for by the sequence of orders held in storage locations 100 to 109 may be repeated six times.

Method. A number in the store is arranged to have the values -5, -4, ... 0 units after the group of orders has been obeyed once, twice, ... six times. Thus when this counting number becomes zero, the process has been performed six times.

It is assumed that storage location 0 can be used to hold the counter, and that storage locations 1 and 2 contain $6 \cdot 2^{-15}$ and 2^{-15} respectively.

Location of order	Order	Notes
		the accumulator is assumed clear at the start
97	S 1 F	places new value of counting number in storage location 0 (initially $-5 \cdot 2^{-15}$)
98	A 2 F	
99	T F	
100		orders to be repeated. It is assumed that they leave the accumulator empty
.		
109		
110	A F	test whether the counting number is zero
111	G 98 F	

In many cases it is not known in advance how many times the sequence of orders must be repeated. An example occurs in the calculation of a reciprocal root $a^{-\frac{1}{2}}$ from the iterative formula $x_{n+1} = 1/2\, x_n(3-ax_n^2)$. The iteration is to be started with a first approximation x_0 and stopped when $|x_{n-1}-x_n|<\varepsilon$, where ε is a positive quantity given in advance. This may be done by means of a sequence of orders which, given the value of x_n in a certain storage location, say m, calculates x_{n+1} and transfers it to m, where it replaces x_n. In addition, the quantity $|x_{n+1}-x_n|-\varepsilon$ is computed and left in the accumulator. If this quantity is positive or zero, the next order, which is an E order, transfers control back to the beginning of the sequence; otherwise control passes straight on. If storage location m contained x_0 before the sequence of orders was operated for the first time, this storage location will now contain $a^{-\frac{1}{2}}$.

1-7 Modification of orders by the program.

It has been explained that orders are expressed inside the machine in a numerical code, and that the numbers which represent them are held in the same store as other numbers needed in the calculation. If a number which

stands for an order is modified, for example by having a constant added to it, it then stands for a different order, and if the section of the program containing it is operated twice, once before and once after the modification, different operations will be performed. This facility of being able to modify the orders in the program by performing arithmetical operations on the numbers representing them is of great importance, and it is perhaps the feature most characteristic of program design for machines like the EDSAC. The operations required for this purpose are performed in the arithmetical unit in the same way as other arithmetical operations.

Some examples of the use which can be made of this facility are given below. It is first necessary, however, to explain the numerical code by which orders are represented inside the EDSAC. The order $\overline{X}$ n F (where $\overline{X}$ stands for any letter in the order code) is represented by the number $2^{-4}\,\overline{x}+2^{-15}\,n$, where the value of $\overline{x}$ for the various orders is given in the table below. The order $\overline{X}$ n D is represented by $2^{-4}\,\overline{x}+2^{-15}\,n+2^{-16}$.

$\overline{X}$	$\overline{x}$
A	-4
C	-2
E	3
F	-15
G	-5
H	-11
I	8
L	-7
N	-10
O	9
R	4
S	12
T	5
U	7
V	-1
X	-6
Y	6
Z	13

Thus A 50 F would be represented by the number $2^{-4}(-4)+2^{-15}\cdot 50$; this may be converted into the number representing A 51 F by adding 2^{-15} to it.

It is often convenient to drop the distinction between orders and the numbers representing them, and to speak, for example, of "the order contained in storage location n," and of orders being modified by having constants added to them.

A sequence of orders designed to be repeated a number of times may contain a group of orders which modify other orders in the same sequence. Each time the sequence is operated it will then cause a different set of calculations to be performed. In this way it is possible to use repetitive cycles to perform calculations which do not at first sight appear to lend themselves to such treatment. The advantage of doing this is that programs can often be constructed with many fewer orders than would otherwise be necessary, and therefore require less space in the store. As an example, suppose that the

sum of the contents of storage locations 100, 101, ... 149 is to be added to the contents of storage location 5.

Method. The contents of storage location 100 are added to those of storage location 5 by means of a group of orders containing the order A 100 F. The address specified in this particular order is then increased by one, and the group of orders repeated. Thus the contents of storage locations 100, 101, ... are added in succession to the contents of storage location 5. It is necessary to terminate this process, and a counter is used as in the previous example.

It is assumed that storage location 0 can be used to hold the counter, and that storage locations 1 and 2 contain $50 \cdot 2^{-15}$ and 2^{-15} respectively.

Location of order	Order	Notes
200	S 1 F	set counter (initially $-49 \cdot 2^{-15}$)
201	A 2 F	
202	T F	
203	A 100 F	the address in this order is increased by one each time the cycle is repeated
204	A 5 F	
205	T 5 F	
206	A 203 F	increase by one the address specified in order 203
207	A 2 F	
208	T 203 F	
209	A F	test for end of process
210	G 201 F	

This program may be shortened by using the variable order for counting. It then appears as below. Storage location 1 contains the number equivalent to the order A 150 F and storage location 2 contains 2^{-15}.

Location of order	Order	Notes
200	T F	clears accumulator
201	A 5 F	add appropriate number to the contents of storage location 5
202	A 100 F	
203	T 5 F	
204	A 202 F	increase the address specified in order 202 by one
205	A 2 F	
206	U 202 F	
207	S 1 F	test if the order contained in location 202 has become A 150 F; if not, repeat the process.
208	G 200 F	

This example contains nine orders. If it were written out in full, that is, if a repetitive cycle were not used, 52 orders would be necessary. A more complete discussion of methods of counting will be found in Appendix F.

Occasionally, where there are very few repetitions, it is better to write out the orders in full. This reduces the machine time taken by the process, since no time is consumed in modifying orders or in counting the number of repetitions, and this fact may be important if the whole process has to be

performed a large number of times. Moreover, if the accumulator is not required for counting and for modifying orders, the program can often be further shortened by making use of the facility of accumulating sums and products. The total number of orders may even be fewer than if a cycle is used.

1-8 Multiaddress codes.

In the EDSAC order code each order has reference to, at the most, one location in the store; it is thus described as a single-address code. Other machines have multiaddress codes in which each order may refer to several locations in the store. For example, one order in such a code might be:

A r s t — add the number in storage location r to the number in storage location s and transfer the result to storage location t.

This is an example of a three-address code. One order in such a code takes up more space in the store than an order in a single-address code (in the EDSAC it would require a long storage location instead of a short one) but it causes a more complicated set of operations to be carried out. Thus the single order A r s t has the same effect as the group of orders A r, A s, T t in the EDSAC order code, and requires one long storage location instead of three short ones. However, use of a three-address code does not always enable a similar saving to be made; for example, to add the four numbers in storage locations r, s, p, and t together and to place the result in storage location q the following three orders are required:

```
A r s q
A q p q
A q t q.
```

In the EDSAC order code the following group would be required:

```
A r
A s
A p
A t
T q
```

In this case the orders in the single-address code actually take less space than those in the three-address code, the reason being that when using the single-address code the programmer can take advantage of the fact that sums can be accumulated in the accumulator. On the whole it is doubtful whether more than a slight saving in the storage capacity required to hold the orders can be obtained by using a three-address code. Its use does, however, enable the speed of the machine to be increased slightly, since the number of orders which have to be extracted from the store is reduced. On the other hand, the complexity of the control section of the machine is increased.

From the point of view of the programmer there is very little to choose between the convenience of using single- and three-address codes; in particular, counting operations can be performed and orders modified in a three-address code by methods exactly analogous to those described in this chapter for use with a single-address code. The decision as to whether a machine

should have a single-address or a three-address code should rest rather with the designer than with the prospective mathematical user.

In most machines the orders are executed, as in the EDSAC, in the serial order in which they stand in the store, except when transfer of control is brought about by the action of a transfer order. An alternative system is to include in each order a specification of the location from which the next order is to be taken. This leads to a four-address code in which three of the addresses are used as in a three-address code and the fourth contains the address of the next order to be executed. This has advantages in the case of a machine which uses ultrasonic tanks (mercury memory) or a magnetic drum for its main store. With either of these stores numbers are available only at certain times in a fixed cycle. If a number or order is to be extracted from a random location there will therefore be a delay, equal on the average to half the circulation time in the case of the ultrasonic store and to half the rotation time in the case of the magnetic drum. If, however, the programmer has control over the location from which the next order is to be obtained, he can reduce this delay by placing the orders and numbers as far as possible in locations chosen so that they become available at the moment they are required. He is assisted in doing this if he is provided with a number of special storage registers which have an access time short compared with that of the main store; for example, a machine using an ultrasonic store may have a number of short mercury tanks, each accommodating a single number in addition to the long tanks of the main store, each of which holds 16 or 32 numbers. This procedure is sometimes called optimum programming and the first machine to be specially designed with a view to its adoption was the ACE (Automatic Computing Engine), of which a pilot model is now working at the National Physical Laboratory at Teddington, Middlesex, England. Optimum programming makes the work of the programmer more complicated, because it introduces considerations concerned with the timing of operations in the machine and thus confuses the essentially arithmetical nature of programming as stressed in this book. However, a compromise can be reached if it is possible for the library subroutines to be constructed in accordance with the principles of optimum programming and for the programmer to construct the other parts of the program in the ordinary way. In this way a high proportion of the gain in speed made possible by the use of optimum programming can be obtained without complicating the task of the programmer unduly. It should be especially noted that the provision of a four-address code of the kind described here and its use in conjunction with optimum programming technique are devices for mitigating the fundamental disadvantages of a delay-type store, and are of no assistance if a store of the electrostatic variety is used.

1-9 Binary-decimal conversion.

It has already been mentioned that conversion of numbers to and from the binary system is performed by the machine. Full details of how this is done may be found by examining the input and output subroutines in Part III of this book; a general explanation of the principles used will be given in the present section.

The paper tape used for input to the machine is prepared by means of a keyboard perforator. There are five positions across the tape in which holes may or may not be punched and one row of holes may therefore be said

to represent a five-digit binary number. The keyboard perforator has 32 keys, labeled with combinations of letters, figures, and other symbols, as in the case of an ordinary teleprinter keyboard. Each key causes one row of holes to be punched on the tape according to the code given in Appendix A. The corresponding five-digit binary numbers are also given in this Appendix.* It will be seen that the figures from 0 to 9 are represented by their binary equivalents. For example, 5 is represented by 00101, 6 by 00110, etc.

Suppose that it is required to put the number 0.21973 into the machine. The successive digits of this number are punched in order on the input tape. When the tape is read by the machine acting under the control of a succession of I orders in the program, the binary equivalents of the following numbers will be transferred to the store in succession:

$$2 \times 2^{-16}$$
$$1 \times 2^{-16}$$
$$9 \times 2^{-16}$$
$$7 \times 2^{-16}$$
$$3 \times 2^{-16}$$

The program contains orders which cause the first of these numbers to be multiplied by 10^4, the second by 10^3, the third by 10^2, the fourth by 10, the last by 1, and the results to be added together. This calculation is carried out in the binary scale so that the binary equivalent of 21973.2^{-16} is now to be found in the store. A further multiplication by $10^{-5} \cdot 2^{16}$ forms the required number in its binary form. It will be seen that the decisive step in the conversion of the number to the binary scale takes place in the keyboard perforator, which converts the individual decimal digits of the number to their binary form.

In drawing up the program for this conversion it is necessary to avoid the use of numbers that lie outside the range $-1 \leq x < 1$. For example, it is not possible to multiply by 10 directly; instead, it is necessary to multiply by 10/16 and to shift the result four places to the left.

Conversion of binary numbers to their decimal form during output is done in an analogous manner. The teleprinter accepts a five-digit binary number (actually the five most significant digits in the storage location specified in the output order) and prints the corresponding character. Here again the code is so chosen that the binary numbers from 0 to 9 are printed as the corresponding decimal figures; for example, 00101 is printed as 5, 00110 as 6, etc. The program must therefore cause the successive decimal digits of the given number to be calculated in their binary form; final conversion to decimal form can then take place in the teleprinter.

The principle of the method used to obtain successive decimal digits is to multiply the number (which is assumed to be positive and less than unity) repeatedly by ten and to remove the integral part each time. If the number is expressed as a decimal fraction this method clearly isolates the successive digits, beginning with the most significant. The same is true if the number is

*A hole in the tape represents the binary digit 1, except in the case of the most significant digit, where a "1" is represented by the absence of a hole. This is done in order to avoid having to represent the number 0 by blank tape.

expressed as a binary fraction (the multiplication being by ten in its binary form, that is, by the binary number 1010), except that the digits are then obtained in the form of the corresponding binary numbers. When this method is programmed for the EDSAC it is necessary, in order to avoid using numbers outside the range $-1 \leq x < 1$, to multiply by 10/16 instead of by 10 and to take the four digits which come immediately after the binary point. The remainder is shifted four places to the left before a further multiplication is performed.

1-10 Checking facilities.

The EDSAC was designed with the understanding that the programmer would incorporate in his program such mathematical checks as he might consider necessary, or arrange for them to be carried out afterwards. No special checking devices are therefore provided inside the machine. It is, however, desirable that there should be some means available whereby the programmer can verify that a number computed and held in the store of the machine has been correctly transferred to the teleprinter. For this reason there is an order (the F order) which enables the number transferred to the teleprinter by the last output order to be read back into the store. By making use of this order it is possible to arrange that an indicating symbol, for example a question mark, shall be printed if the number has been incorrectly transferred to the teleprinter. Examples of how this is done will be found in the output subroutines given in detail in Part III of this book. It is of course possible that even though the correct number has been transferred to the teleprinter a wrong character will be printed. The design of the Creed teleprinters used in conjunction with the EDSAC is such, however, that the possibility of an error occurring beyond the point at which the check is made is remote.

CHAPTER 2

INPUT OF ORDERS

2-1 Initial orders.

This chapter is concerned with the process by which orders are read from the tape and placed in the store of the EDSAC. The only way in which a symbol punched on the tape can be read is by the operation of an I order. To enable a program tape to be read, therefore, means are provided whereby a short group of orders, known as the "initial orders" or sometimes as the "initial input routine," can be placed in the store independently of the input mechanism. These orders are wired in binary form on a set of stepping switches (uniselectors), and are automatically transferred to the store (and called into action) when the starting button is pressed. The initial orders are needed only while the program tape is being read, and the space they occupy in the store may be used again for other purposes during the course of the calculation.

An order is punched exactly as it is written, the address being in decimal form. The initial orders must therefore be such as to convert the address to binary form, to assemble the complete order, and to place it in the correct storage location. It is important to distinguish between the coded form in which orders are punched on the tape and that in which they appear in the store, and to realize that the relation between these two forms is determined solely by the initial orders. By making the two forms more similar (for example, by punching the address in the scale of 8 or 16) it would be possible to simplify the initial orders. There would, however, be no advantage in doing this and it would mean that more work would be left to the programmer, who would have to carry out tedious conversions when constructing the program. It is highly desirable that the machine itself should carry out as much of this work as possible; the chance of error is then reduced and the programmer is left free to concentrate his attention on the more essential aspects of the program.

The choice of the initial orders, and thus of the form in which orders are punched, is therefore a matter for careful consideration, since upon it depends the ease with which all programs are constructed. Once the choice has been made, a library formed, and several large jobs begun, a change in the form of writing and punching orders would entail a big reorganization. The form used with the EDSAC was changed in September 1949, after only a few simple programs had been run; now, however, any substantial change would be practically out of the question, even if it were desired (see reference 10).

The form in which orders referring to specific storage locations are punched has already been described. First, there is a letter indicating the function of the order, then the address in its decimal form, and finally a code letter which is either F or D according as the address refers to a short or a long location. It should be noted that in the address zeros are not punched in front of the first significant digit, for example A 50 F is punched, not A 0050 F; if the order referred to the number in storage location 0 it would be punched A F.

The action of the initial orders can now be described. When an order such as A 50 F or A F is being transferred from the tape to the store, the first character to be read is the function letter, and the corresponding binary number is placed by the initial orders in a suitable location for temporary storage. The next character may be either a digit of the address or a code letter F or D. These can be distinguished by the fact that F and D correspond to binary numbers which are greater than ten. The character just read is therefore tested by having 10-1/2 subtracted from it; if the result is negative the character must represent a digit of the address, otherwise it represents a code letter. As the successive digits are read the address is built up progressively in binary form. When the code letter is encountered the address and the number representing the function letter are added together. If the code letter is F the result represents the complete order and is transferred to the store as it stands. If the code letter is D, 2^{-16} is added to the result before it is transferred to the store.

In addition to the code letters F and D so far referred to, there are thirteen other code letters which may be used to terminate an order. The object of these code letters is to facilitate the use of subroutines in ways which will be described later. Each causes the contents of a certain storage location to be added to the order before it is transferred to the store. The complete list of code letters is as follows:

Code letter	Location whose content is added to the order	Number added
F	41	zero
θ	42	variable
D	43	2^{-16}
ϕ	44	
H	45	
N	46	
M	47	
Δ	48	
L	49	variable
X	50	
G	51	
A	52	
B	53	
C	54	
V	55	

Storage location 41 contains zero, so that the code letter F leaves the order unchanged. Storage location 43 contains 2^{-16}, so that code letter D causes 2^{-16} to be added to the order. These two code letters thus have the effect described earlier.

All the above code letters indicate the end of an order, and cause it to be placed in its correct location in the store. The code letter π causes 2^{-16} to be added to the order (in this it resembles D) but must be followed by another code letter to indicate the end of the order. It is thus possible by using π to cause both 2^{-16} and some other number to be added to the order before it is put away in the store.

2-2 Pseudo-orders.

A converse of the fact that orders are represented in the machine by numbers is that numbers may be represented outside the machine by "pseudo-orders," that is, tape entries which are punched in the same form as orders but which are merely intended to be used as constants and are never to be obeyed as orders. For example, the pseudo-order P n F is equivalent to the number $n \cdot 2^{-15}$, since P corresponds to zero (see Appendix A); P n D is equivalent to $(n+1/2) \cdot 2^{-15}$, and X F (where X stands for any letter) is equivalent to $x \cdot n^{-4}$, where x is the numerical equivalent of X. This is often a convenient way of putting constants into the machine. It should also be noted that genuine orders which are obeyed at one point in a program may also be used as constants in other parts of the program.

When each order has been built up by the initial orders it is transferred to its correct location in the store. The particular order in the initial input routine which causes this transfer to take place will be referred to in what follows as "the transfer order." The address specified in the transfer order is increased by unity each time an order is placed in the store, so that successive orders are placed in successive storage locations.

2-3 Examples.

The following examples show orders and pseudo-orders as they are punched and in the binary form in which they are held in the store.

Punched on tape				Held in store														
				Function letter				Address										Long/short digit
A	6	F	≡	1.1	1	0	0	0	0	0	0	0	0	1	1	0	0	
A	6	D	≡	1.1	1	0	0	0	0	0	0	0	0	1	1	0	1	
T		D	≡	0.0	1	0	1	0	0	0	0	0	0	0	0	0	0	1
P	6	F	≡	0.0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
P		F	≡	0.0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

If storage location 45 contains the number 80,

T	6	H	≡	0.0	1	0	1	0	0	0	0	1	0	1	0	1	1	0	0

If it is desired to modify an order by means of a code letter other than D and at the same time to make the order refer to a long storage location this can be done by punching π immediately before the code letter. Thus

T	6πH		≡	0.0	1	0	1	0	0	0	0	1	0	1	0	1	1	0	1

2-4 Control combinations.

Among the orders on the tape are punched groups of symbols called "control combinations." These are distinguished from orders by the initial input routine; they are not placed permanently in the store but direct the manner in which the input process is carried out. For example, the control combination T m K, where m is an integer punched in decimal form like the address of an order, causes the address in the transfer order to be replaced by m, so

that the next order is placed in storage location m regardless of where the previous order was placed. Succeeding orders go into storage locations m+1, m+2, etc. For example, suppose that it is required to place the pseudo-orders P 5 F and P 10 F in storage locations 45 and 46, and then to place a sequence of orders in storage locations 100, 101, etc. The following will then be punched on the tape in front of the orders:

T 45 K
P 5 F
P 10 F
T 100 K

The control combinations in most common use are given below. A further list of control combinations is given in Appendix C. The initial orders themselves, together with notes on their operation, are given in Appendix B.

T m K	this causes the next order on the tape to be placed in storage location m
G K	this causes the address in the transfer order to be copied into 42, which corresponds to the code letter θ
T Z	this causes the address held in 42 (m, say) to replace that in the transfer order, so that the next order is sent to storage location m.
E m K P F	this stops the reading of orders, and causes control to be transferred to storage location m with the accumulator cleared
E 25 K T m H	this causes the next order on the tape to be placed in storage location m+h, where $h \cdot 2^{-15}$ is the number in 45. H and 45 may be replaced by any other code letter and the associated storage location.

2-5 Starting the program.

The first few inches of a program tape are always left blank and the tape is inserted into the EDSAC tape reader with the reading head somewhere on the blank portion. It is not necessary to set the first row of holes under the reading head because the initial orders are designed to have the property of ignoring blank tape, in the sense that they do not erase anything of importance from the store when it is read. It is, however, necessary to punch a control combination at the end of the blank tape and immediately in front of the orders. The usual control combination is P K T m K, and in this case orders go into the store starting at storage location m. If the control combination P F is used orders will go into 45, 46, etc.; P Z will cause them to go into 44, 45, etc. The action of the initial orders when reading blank tape is described in detail in Appendix B.

The control combination E a K P F is punched at the end of the orders to cause control to be transferred to a, which is supposed to contain the first order of the program. The initial orders may be called in again, if required, to read further orders from the tape by transferring control to storage location 25. The accumulator need not be empty, but the first combination to be

read from the tape should be a control combination which will replace the transfer order, for example T n K. If it is intended to use the initial orders again in this way care must be taken to see that they are not written over by numbers during the course of the program. If the initial orders have been written over they may be replaced (after the machine has stopped) by pressing the starting button again; the contents of other parts of the store will be left undisturbed.

2-6 Use of code letters.

The code letter θ has a special use in connection with subroutines. In any subroutine it will be found that the addresses specified in some of the orders are those of other orders or pseudo-orders in the same subroutine, and therefore depend on the location in the store of the subroutine as a whole. In order to illustrate this, a subroutine for replacing the number in storage location 0 by its modulus is given below. It is shown with its first order placed in storage location m and it will be seen that the address specified in the second order depends on m.

m	S		F
m+1	G	m+3	F
m+2	T		F
m+3	T	1	F

Subroutines punched in this way would not be suitable for forming into a library, since each could be used in one place in the store only. This difficulty is overcome by punching the subroutine in the following form:

G		K
S		F
G	3	θ
T		F
T	1	F

The control combination G K at the head of this subroutine causes the address specified in the transfer order, m say, to be recorded in storage location 42. The orders of the subroutine are then placed in the store in order, the first going into storage location m. When the second order is taken in, the code letter θ which terminates it causes the number m in storage location 42 to be added to its address. The order then refers to the correct location within the subroutine. It should be clearly understood that the modification of an address brought about by the action of a code letter takes place at the time that the order is being transferred from the tape to the store, and not at the time that it is executed.

The use of the system described in the last paragraph enables library subroutines to be stored in the form of short lengths of punched tape which can be copied mechanically on to a program tape. A description of the equipment used for this purpose is given in Chapter 6.

The main purpose of the code letters, other than θ, is to make it possible for parameters to be incorporated in subroutines during input. This matter will be taken up more fully in section 3-3, but a simple example will be given here. The following subroutine is designed to replace the number in

storage location h by its modulus, where 2^{-15} h is the number in storage location 45. Were it not too trivial to warrant such treatment, this subroutine might be contained in the library.

```
T     Z
S     H
G  3  θ
T     H
T     F
```

If this subroutine were in the library and if a programmer wished to use it for replacing the number in storage location 150 by its modulus, he would copy it mechanically from the library tape on to his program tape, with the following punched immediately in front:

```
G     K
T  45 K
P 150 F
```

Note that the subroutine starts with the control combination T Z, which restores the address of the transfer order to the value it had before the pseudo-order P 150 F was placed in 45.

2-7 Constants.

When the initial orders have finished their work the following constants are left in the store:

storage location 2	P 1 F
storage location 3	U 2 F

These constants are used by most library subroutines and it is important that they should be left undisturbed during the program.

2-8 Notation.

The following abbreviations will be used in this book.

n	short storage location having serial number n (the alternatives nF or S(n) may sometimes be necessary to avoid confusion)
nD	long storage location formed by combining short storage locations n and n+1, n being even (the alternative n' was used in earlier literature)
C(n)	content of storage location n
C(nD)	content of storage location nD (the alternative C(n') was used in earlier literature)
C(Acc.)	content of the accumulator
C(R)	content of the multiplier register
nH	storage location (n+h) where $h \cdot 2^{-15}$ is the number in 45 during input of the part of the program concerned; alternatively long storage location (n+h)D if $(h+0.5) \cdot 2^{-15}$ is in 45 during input

nπH	long storage location (n+h)D where h · 2^{-15} is the number in 45 during input of the part of the program concerned
C(nH)	content of nH
C(nπH)	content of nπH

(In the last four cases, H and 45 may be replaced by any other code letter and its associated storage location)

REFERENCES

The following is a list of publications about the EDSAC:

1. Wilkes, M.V., The design of a practical high-speed computing machine, The EDSAC. Proc. Roy. Soc. A 195, 274 (1948)

2. Wilkes, M.V. and Renwick, W., An ultrasonic memory unit for the EDSAC. Electronic Engineering 20, 208 (1948)

3. Wilkes, M.V., Progress in high-speed calculating machine design. Nature 164, 341 (1949)

4. Wilkes, M.V., Electronic calculating machine development in Cambridge. Nature 164, 557 (1949)

5. Wilkes, M.V., Programme design for a high-speed automatic calculating machine. J. Sci. Instr. 26, 217 (1949)

6. Wilkes, M.V. and Renwick, W., The EDSAC, an electronic calculating machine. J. Sci. Instr. 26, 385 (1949)

7. Bennett, J.M., High-speed digital calculating machines. Distribution of Electricity 22, 251 and 276 (1950)

8. Wilkes, M.V., and Renwick, W., The EDSAC (Electronic Delay Storage Automatic Calculator). M.T.A.C. 4, 61 (1950)

9. Wilkes, M.V., The use of the EDSAC for mathematical computation. Appl. Sci. Res. B1, 429 (1950)

10. Wheeler, D.J., Programme organization and initial orders for the EDSAC. Proc. Roy. Soc. A 202, 573 (1950)

11. Gill, S., A process for the step-by-step integration of differential equations in an automatic digital computing machine. Proc. Camb. Phil. Soc. 47, 96 (1951)

12. Wilkes, M.V., Automatic computing. Nature 166, 942 (1950)

13. Gill, S., The diagnosis of mistakes in programmes on the EDSAC. Proc. Roy. Soc. A (in press)

A list of references of a more general nature will be found in

Hartree, D.R., Calculating instruments and machines. University of Illinois Press, 1949, and Cambridge University Press, 1950

CHAPTER 3

SUBROUTINES AND PARAMETERS

3-1 Open subroutines.

The simplest form of subroutine consists of a sequence of orders which can be incorporated as it stands into a program. When the last order of the subroutine has been executed the machine proceeds to execute the order in the program which immediately follows. This type of subroutine is called an "open" subroutine.

3-2 Closed subroutines.

A "closed" subroutine is one which is called into use by a special group of orders incorporated in the master routine or main program. It is designed so that when its task is finished it returns control to the master routine at a point immediately following that from which it was called in. The subroutine itself may be placed anywhere in the store. There are various methods of arranging the operation of entering a closed subroutine and returning to the master routine after the operation of the subroutine has been completed. The method chosen for use with the EDSAC is given below; n is the address of the first order of the subroutine.

Number of storage location	Order	Explanation (Accumulator contains zero at this point)
m	A m F	adds number representing A m F into the accumulator (this is negative, since A corresponds to -4/16)
m+1	G n F	transfers control to n, since number in the accumulator is negative

The orders in the subroutine are as follows:

	G K	control combination; puts the value of n in 42
n	A 3 F	adds U 2 F to contents of accumulator (A m F) forming E m+2 F (link order) since A ≡ -4/16, U ≡ 7/16, whence A + U ≡ 3/16 ≡ E
n+1	T p+2 θ	plants link order in (n+p+2) (code letter θ causes C(42), i.e. n, to be added to address during input)
n+2 ... n+p+1		operational orders of the subroutine, p in number. These leave the accumulator empty
n+p+2	Z F	becomes E m+2 F (link order) as result of order in (n+1)

Any order may be punched on the tape for the last order of the subroutine, since it is overwritten by the link order. Often Z F or P F is used and this

has the advantage that if by reason of an error the link order is not planted in the correct position the machine will stop and by the place of its stopping give some indication of what is wrong. Orders which are intended to be changed during the program are usually written in brackets.

3-3 Preset parameters.

It is desirable to be able to make library subroutines of as wide a utility as possible in order that the total bulk of the library may be kept small. This may be done by including in a subroutine one or more parameters which may be given different values on different occasions. For example, D4 (see Part II) is a division subroutine which divides the contents of a certain storage location hD by the contents of the accumulator, where h is a parameter which may be set in advance. Parameters of this kind are called preset parameters and it is arranged that they are set to their correct values during input by making use of the facilities described in Chapter 2 under the heading Code Letters. In the case of the division subroutine mentioned, the following combinations must be punched on the tape in front of the subroutine:

G K
T 45 K
P h D

The subroutine itself is headed by the control combination T Z instead of the usual G K. The effect is that the number corresponding to P h D (i.e., $h \cdot 2^{-15} + 2^{-16}$) goes into storage location 45 and is therefore added to the address specified in any order in the subroutine which is terminated with the code letter H. In this way the numerical value of the constant h can be incorporated in the subroutine during input. When a subroutine has a number of parameters they are punched in order after the control combination G K T 45 K; they then go into storage locations 45, 46, 47, etc., and are added to the addresses specified in orders terminated by the code letters H, N, M, etc, respectively.

3-4 Program parameters.

The values of preset parameters are incorporated in a subroutine during the process of input and are therefore fixed for the whole of a program. If it is desired to include in a subroutine a parameter which can be given different numerical values at different points in the same program, then a different method must be used. Such a parameter is called a program parameter and is normally placed immediately after the orders which call in the subroutine. An example is to be found in subroutine P 1 which prints the positive number in OD to n places of decimals. This subroutine, assumed to have its first order in p, is called in by the following group of orders (the accumulator containing zero at this point):

m	A m F	adds number corresponding to A m F into the accumulator
m+1	G p F	transfers control to p (that is, to the beginning of the subroutine), since the number in the accumulator is negative
m+2	P n F	program parameter specifying n
		program parameter specifying n

The subroutine is so constructed that an order A m+2 F is formed and planted in a suitable position within the subroutine. When this order is obeyed it extracts the program parameter from the master routine and enables it to be used by the subroutine. The link order formed is E m+3 F, instead of E m+2 F as described in Section 3-2, since it must return control to the location which follows immediately after that containing the program parameter. The actual orders used in P1 are as follows:

	G		K	control combination; puts p in 42 during input. Subsequent orders can therefore be numbered relative to the next order as zero
0	A	18	θ	adds U 3 F to C(Acc) (which is A m F as in all closed subroutines on entry) forming E m+3 θ, the required link order
1	U	17	θ	plants link order in 17θ (i.e., in p+17) and leaves E m+3 F in the accumulator
2	S	20	θ	subtracts M 1 F from C(Acc) forming S m+2 F (since E = 3/16, M = -9/16, whence E-M = 12/16 = S).
3	T	5	θ	plants S m+2 F in 5θ
4	H	19	θ	first operational order of subroutine
5	(P		F)	becomes S m+2 F as result of order in 3θ
⋮				other operational orders of subroutine
16				
17	(E		F)	becomes link order E m+3 F as result of order in 1θ
18	U	3	F	pseudo-orders forming constants required in the operation of the subroutine.
19	J		F	
20	M	1	F	

CHAPTER 4

LIBRARY SUBROUTINES AND THEIR USE IN CONSTRUCTING PROGRAMS

4-1 Library catalog.

The library catalog used in the Laboratory is drawn up in two sections. One gives a concise specification of the purpose of each subroutine together with sufficient information to enable a programmer to make use of it; this includes information about the operating time and storage space occupied. The second section gives the orders of each subroutine in full. The catalog is contained in loose leaf books so that new sheets can be inserted as new subroutines are added to the library. A condensed version of the catalog is given in Parts II and III of this report. Part II consists of specifications of all subroutines now in the library except for some which are obsolete. Most of these specifications are given in full but an abbreviated version only is given in the case of some subroutines which can be regarded as variants of others that are specified completely. Part III contains full program sheets for about half the total number of subroutines in the library and includes all those which are thought to contain points of special interest.

Although much labor can be saved by making use of the library in its present form, it is still in many respects incomplete, and new subroutines are continually being added. In particular, it is hoped that subroutines for performing the following calculations will shortly be included: multilength arithmetic, calculation of hyperbolic functions, solution of algebraic equations, etc.

It will be found in a number of cases that there are two or more subroutines which perform very similar operations. Usually they differ in time of operation and in storage space occupied. Normally the one with the shortest operating time would be chosen for any particular application, but if the program occupies nearly all the store, then storage space may become the major consideration. Time of operation is of great importance only if the subroutine is called in many times during the program, thus consuming a large proportion of the total time.

Subroutines may also differ in numerical accuracy and in the number of parameters which may be varied to suit particular applications. If a subroutine has many parameters it is often useful to have a separate subroutine to deal with any case that commonly arises, since a subroutine with fewer parameters is shorter and simpler to use.

4-2 Input and output subroutines.

All input subroutines read numbers punched on the tape in the scale of ten, convert them to the scale of two, and place them in the store. Some subroutines read a single number only on each occasion they are called in, while others read a series of numbers and place them in consecutive locations in the store. The subroutines may be further classified into those which read decimal fractions from the tape and those which read integers; in the latter case, when an integer n is read from the tape the number $n \cdot 2^{-16}$ or $n \cdot 2^{-34}$ is put into the store according as short or long numbers are being used. The

conversion of decimal fractions is slightly simpler if the least significant digit is read first and subroutines R5 and R7 are designed in this way. The number tape can, however, be punched in the ordinary way with the most significant digit first and reversed during the process of copying onto the final tape.

Many subroutines contain numbers as well as orders. Short numbers are best put in as pseudo-orders but long numbers may involve the use of an input subroutine. In some cases, library subroutine R2 is included in such a way that it is overwritten when the numbers have been taken in. Since, however, several subroutines in a program may need long numbers it is now common practice to draw up subroutines on the assumption that R9 has first been put into the store. R9 is a modified form of R2 which allows the input of long integers during input of orders. It is always placed in locations 56 to 70 inclusive.

When a subroutine contains only one or two long constants, an alternative to the use of an input subroutine is to put the constants in as two short numbers. A difficulty arises, however, because there is an unused digit between each two short numbers (the "sandwich digit"). The method can only be used, therefore, to put in a number which has a zero in the 17th position after the binary point. This is not a serious limitation, since either the number itself or its complement is of this form unless it is an odd multiple of 2^{-17}. The long storage location intended to receive the constant must first be cleared (to make sure that the sandwich digit is zero), and the two short numbers planted one after the other. This method of putting in long numbers is not often used now but examples will be found in A1 and P7.

The output subroutines convert numbers in the store to the scale of ten and cause them to be printed. Again they may be divided into those which deal with decimal fractions and those which deal with integers (some of the latter suppress nonsignificant zeros). Some output subroutines print the numbers in a special layout; in other cases the output subroutine prints a single number only and it is left to the programmer to arrange his own layout. Most of the later output subroutines make use of the F order to verify that the digits have been correctly printed. If this check fails then some special indication is given, usually an extra line-feed.

By the use of suitable scale factors the input and output subroutines can be made to handle numbers having the binary point in positions other than those mentioned above.

4-3 Division subroutines.

The order code of the EDSAC does not include an order for division, which must therefore be carried out by means of a division subroutine, several of which are available in the library. One of these, D6, uses an iterative formula and is arranged to give the greatest possible accuracy. Others use a repetitive cycle to divide directly and contain fewer orders than D6. Errors can build up in this process, however, and these subroutines are not as accurate as D6, although the result is usually reliable to about as many figures as exist in the dividend.

4-4 Trigonometrical and other functions.

When values of a trigonometrical or similar function are required for arbitrary values of the argument it is usually better to use a subroutine which calculates them from first principles rather than to place a table in the store and to interpolate from it. In many cases it is quickest and simplest to use a power series. In the earlier subroutines Taylor series were used, but later subroutines use series based on Tchebycheff polynomials since in this way the same accuracy can be obtained with fewer terms. An example of this will be found in T7.

Sometimes repetitive methods based on very simple formulas and needing very few orders are available; they are, however, usually rather slow. Examples are to be found in subroutines E2 and S1. In the latter case the required answer is built up digit by digit.

When a series of sines or cosines is required with equal increments of the argument, subroutines T5 or T6, which are based on a recurrence formula, may be useful. This situation occurs when a differential equation involving a sine or cosine of the independent variable is being solved.

For most trigonometrical subroutines the angle corresponding to the argument must be in the first quadrant. One routine (T4), however, can be used for arguments of any magnitude.

An example is given in Section 7-1 of a program built up from the subroutines already discussed. The program causes a series of numbers x (<1) punched on the tape to be read and the quantities $e^{-\sin x}$ to be computed and printed.

4-5 Quadrature.

Q1, Q2, and Q3 are subroutines for computing definite integrals. An auxiliary subroutine for calculating values of the function to be integrated must be constructed; this is called in as required by the integration subroutine. Q2 and Q3 are based on Gauss' 5-point and 6-point integration formulas and one of these is ordinarily the best subroutine to use. The usual objection to Gauss' formulas, namely that the function has to be computed for awkward values of the argument, is of no account when using an automatic machine (an illustration of the different considerations which apply when selecting methods for automatic as compared with hand computing). However, in some cases, for example if the function to be integrated is obtained by integrating a differential equation, a formula which uses equally spaced ordinates may be more suitable. In these cases Q1, which is based on Simpson's rule, may be used.

A simple example of the use of such a subroutine will be found in Section 7-2, which gives a program for the calculation of π from the formula

$$\frac{1}{4}\pi = \int_0^1 (1 + x^2)^{-1}\, dx.$$

4-6 Assembly subroutines.

The example given in Section 7-1 illustrates two alternative methods of assembling a program. In each of these the programmer has to decide where the master routine and each subroutine are to go in the store and to insert

the correct addresses in the orders in the master routine which call in the subroutines. The object of an assembly subroutine is to relieve the programmer of these and other mechanical tasks.

4-61 Principle of operation of assembly subroutine M1. Any complete program contains one or more of the following components:

1. Sequences of numbers
2. A master routine
3. Closed subroutines of two kinds:
   (a) those made specially for the program
   (b) those taken from the library.

When the subroutine M1 is used, the numbers in each sequence are numbered 0, 1, 2, ..., and are distinguished in the master routine and in the first group of subroutines by code letters (H, N, M, ...) punched after each address, one code letter being used for each number sequence. The closed subroutines are numbered 1, 2, 3, etc., and are called in by orders A m_1 θ, G 1 ϕ; A m_2 θ, G 2 ϕ; etc., where m_1, m_2, etc. are the addresses of the storage locations in the master routine from which the subroutines are called in.

M1 is first punched on the tape and is followed by two parameters P r F and T s K; $r \cdot 2^{-15}$ goes into 44 and its meaning will appear later. The components of the program are then punched in the above order, each being preceded by a control combination; the last component is followed by a control combination which starts the program. The components eventually go into the store head-to-tail in the order in which they are punched. Suppose that the first numbers of the various number sequences (H, N, M, ...) go into locations n_1, n_2, ..., etc., that the first order of the master routine goes into m, and that the first orders of the closed subroutines go into s_1, s_2, s_3, etc.

M1 is called in by the control combination punched at the head of each component. It causes a record to be made of the location into which the first order or pseudo-order of the component is about to go, in the following form:

1. When the number sequences go into the store, pseudo-orders P n_1 F, P n_2 F, etc., go into 45, 46, etc.
2. When the master routine goes into the store, the order E m F goes into r.
3. When the closed subroutines go into the store, orders G s_1 F, G s_2 F, etc., go into (r+1), (r+2), etc.

M1 then returns control to the initial orders, and the orders or pseudo-orders which follow on the tape are read in the usual manner.

The control combination E 25 K E ϕ P F is punched at the end of the tape. This sends control to r, where there is an E order which sends control to the beginning of the program. When the pth subroutine is called in, control passes to (r+p), where there is a G order which sends it to the beginning of the correct subroutine. Orders in the master routine and in the subroutines which refer to the various number sequences (distinguished by different code letters) have their addresses corrected in the usual way by the addition of C(45), C(46), etc.

Normally, M1 causes the first order or pseudo-order of each component to be placed in an even location. This means that here and there short storage locations will be left unused. If this is considered undesirable, M1 may be

called in in such a way that the first order or pseudo-order of the next component goes into the next available location, whether it be odd or even.

4-62 Directions for use of M1. The method of use of M1 is best explained by giving a schedule for the punching of the tape. Two number sequences and two subroutines are shown, but others may be added.

Notation: a location of first order of M1
r location of reference order for master routine
s location of first order or pseudo-order to be placed in the store

Tape	Notes
P K T a K Assembly subroutine M1	copied from library tape
P r F T s K	parameters
space	
P Z G K E a K T F number sequence (H)	calls in M1. which places P n_1 F in 45
space	
P Z G K E a K T F number sequence (N)	calls in M1, which places P n_2 F in 46
space	
P Z G K T(a+10)K T ϕ	sets M1 ready to place a reference order in r
E a K I F Master routine	
space	
P Z G K E a K P F subroutine no 1	calls in M1, which places G s_1 F in r+1
space	
P Z G K E a K P F subroutine no 2	calls in M1, which places G s_2 F in r+2
E 25 K E ϕ P F	sends control to master routine via r

Notes:

1. If the combination E a K T D is punched in front of the first number sequence instead of E a K T F, P n_1 D will be placed in 45 instead of P n_1 F and similarly for the other number sequences.

2. If it is desired to place a component of the program in the next available storage location regardless of whether it is odd or even, M1 should be called in by E (a+1) K, instead of E a K.

3. If storage space is short, M1 may be placed where it will be overwritten by the last subroutine of the program.

4. Where spaces are shown on the tape at least two blank rows must be left. If desired, the spaces may be omitted altogether, in which case the combination P Z should also be omitted.

5. If it is desired to leave a gap in front of any component, the combination G n K should be punched instead of the G K immediately before E a K; n storage locations will be left unused.

6. If there are no number sequences, the control combination G K T(a+10) K T ϕ should follow directly after the parameter T s K.

7. M1 makes the address of the transfer order equal to C(42) (increased by 1 if necessary); that is, it has a similar effect to T Z. Since it leaves the address of the transfer order equal to C(42) it need not be followed by G K.

8. If the first number sequence is to go into the store immediately after M1 the parameter T s K should be T a+16 K.

9. The above proforma shows the normal way in which it is intended that M1 should be used. Various other possibilities will suggest themselves; for example, several E orders may be stored as reference orders in addition to the one used for the master routine. It is to be noted that if M1 is called in by the control combination E a K X q F, where X is any letter, the reference order manufactured and placed in the store will be (X+G) q F.

As an example, the program for the computation of

$$\int_0^1 (1 + x^2)^{-1} \, dx$$

given in Section 7-2 is given in Section 7-3 in a revised form making use of M1.

4-63 Principle of operation of assembly subroutine M2. This subroutine handles the input of subroutines in a similar manner to M1, but does not apply to number sequences. It requires fewer control combinations preceding subroutines than does M1. M2 slightly modifies the initial orders and enables use to be made of the code letter S. Control combinations terminated by S operate as follows:

Case (1). Control combinations with zero address. First the address specified in the transfer order (order 22) is copied into 42; this is the same effect as that of the control combination G K. Next, a reference order is stored, having the same address as that specified in the transfer order, and the same function letter as the control combination. Thus the control combination X S, when the transfer is T n F, will put P n F in 42 and store the reference order X n F.

Case (2). Control combinations with address 1. The address specified in the transfer order, if even, is left unchanged, and if odd, is increased by 1. Thereafter the effect is the same as in case (1).

Reference orders are always stored by M2 immediately after itself. Thus if M2 commences in m, the reference orders are placed in (m+16),

(m+17), ..., etc. The parameter P (m+16) F is automatically placed in 44 when M2 is fed in, so that the code letter ϕ will refer to the reference orders. For example, G 2 ϕ will switch control to the third reference order.

For a normal closed subroutine the reference order will be a G order, switching control to the start of the subroutine. For the master routine an E order may be used to direct control to the beginning of the program. In special cases other letters may be useful.

M2 must be so placed in the store that room is left for the reference orders which follow it. The first fourteen orders of M2 may be written over by the last subroutine on the tape; the last two orders must remain undisturbed until the tape is read, otherwise the code letter π will not be read correctly. The reference orders must remain throughout the program.

Orders punched immediately after M2 will be placed in 45 onwards.

4-64 Directions for use of M2. The following example shows how a tape could be arranged for a simple program consisting of two closed subroutines and a master routine.

Tape	Notes
P K T a K subroutine M2	places M2 in store, commencing at a, and puts P (a+16) F in 44
blank tape	
P Z T s K E S T 45 K parameters for master routine	places parameters in 45 onwards
T Z master routine	places master routine in s, sets C(42) to P s F and stores E s F in ϕ
blank tape	
P Z G 1 S T 45 K parameters for first subroutine first subroutine	places parameters in 45 onwards, and subroutine in next even location following master routine. Places G order in 1ϕ
blank tape	
P Z G S second subroutine	places subroutine in next location, odd or even. Places G order in 2ϕ
blank tape	
P Z E 25 K E ϕ P F	switches control to ϕ, and thence to the beginning of the master routine.

Notes:

1. m_1 A m_1 θ
G 1 ϕ — calls in first subroutine

   m_2 A m_2 θ
G 2 ϕ — calls in second subroutine

2. Blank tape means at least two rows. If desired it may be omitted, in which case the following P Z should also be omitted.

3. When preparing a subroutine (or master routine) with no preset parameters for use with M2, the G S (or E S) may be included at the head, in place of G K.

4. Initial orders 27 and 28 are altered by M2.

4-7 Integration of differential equations.

There are in the library four subroutines for integrating ordinary differential equations (not necessarily linear) by step-by-step processes. G3 and G4 enable second-order differential equations with the first derivative absent to be integrated; they are based on conventional methods using difference formulas in which use is made in each interval of values of the function calculated in previous intervals. They have the disadvantage that special methods are needed for starting the integration.

G1 and G2 are subroutines for integrating sets of simultaneous first-order differential equations using a modified Runge-Kutta method (ref. 12) which is described below. This method has the advantage that a special starting procedure is not necessary and, since any differential equation or set of differential equations can be reduced to a system of first-order equations, it is of wide utility. In cases where both are applicable it is, however, somewhat slower than the method used in G3 and G4.

4-71 Library subroutines G1 and G2. The modified Runge Kutta process.

This process handles a set of simultaneous first-order ordinary differential equations, in which each derivative is expressed explicitly in terms of the variables

$$y_1' = f_1(y_1, y_2, \ldots\ldots y_n),$$
$$y_2' = f_2(y_1, y_2, \ldots\ldots y_n),$$
$$\cdots\cdots$$
$$y_n' = f_n(y_1, y_2, \ldots\ldots y_n).$$

Any equation or set of equations must be expressed in this form before the process can be applied. For example,

$$y'' = -w^2 y$$

may be written

$$y_1' = wy_2,$$
$$y_2' = -wy_1,$$

where $y_1 = y$ and $y_2 = y'/w$.

The case in which the functions f involve the independent variable can be treated by the method described in Section 4-72.

The subroutine G1 or G2 carries out one step of the integration each time it is called in. In doing so, it makes use of an auxiliary subroutine which evaluates the functions $f_1 \ldots f_n$. The auxiliary subroutine must be provided for the individual problem. It is called into play four times during each step.

The auxiliary subroutine is asked to provide the quantities hy' multiplied by a suitable scale factor 2^m, where h is the length of the interval, and m is chosen to be as high as possible without exceeding capacity.

Apart from the 2n storage locations used to store y and $2^m hy'$, further n locations are used as working positions by the integration subroutine (to hold the quantities $2^m q$, see Section 4-73). The numbers left standing in these locations after the end of a step are 3×2^m times the rounding-off errors of the quantities y; they are taken into account during the following step, and serve to prevent the rapid accumulation of rounding-off errors. As a result, the effective numerical accuracy is m digits more than the capacity of the storage locations. At the beginning of a range these working positions must be cleared, otherwise the integration routine will add spurious "corrections" to the variables. Apart from planting the initial values of the variables, this is the only preparation required before starting an integration.

The truncation error in one step is of order h^5. Ordinarily it is about $10^{-2}h^5$, so that maximum accuracy is obtained with $h = 2^{-7}$ or 2^{-8} for long numbers, and $h = 2^{-3}$ or 2^{-4} for short numbers. If the functions are very sensitive to variations in y, or if the number of equations is very large, smaller steps will probably be necessary.

4-72 The independent variable. If the independent variable occurs in the functions f, it may be obtained by integrating the equation $x' = 1$. x is treated as an additional dependent variable, for which the auxiliary subroutine has to provide the quantity $2^m hx' = 2^m h$. In point of fact the latter may be planted at the beginning of the integration and left there, so that the auxiliary subroutine is relieved of the task. If the independent variable does not appear in any of the f's but is merely wanted for indication purposes, it is quicker to use a simple counter in the master routine.

When x is generated by integrating $x' = 1$, the values which it assumes during the four applications of the auxiliary subroutine within one step are x_0, $(x_0+\frac{1}{2}h)$, $(x_0+\frac{1}{2}h)$, and (x_0+h) respectively, where x_0 is the beginning of the step. This has two implications. First, if time is of great importance, x may be generated by using a binary switch in the auxiliary subroutine, so that $\frac{1}{2}h$ is added every other time the subroutine is used; x may then be used in calculating the f's, but does not require the introduction of an additional "dependent" variable. Second, if the f's involve a function of x which is tabulated at equal intervals, it will only be necessary to employ the tabulated values, or values interpolated at simple fractions of the tabular interval.

In the case of G1, if either of the suggestions in the preceding paragraph is carried out, G1 should be placed in the upper half of the store to obtain maximum accuracy (the ideal position is 386 onwards). This is because one of the orders forms part of a constant which thus depends slightly on the location of the routine. In normal use the effect is quite negligible, but it does mean that the last value of x in each step may differ from (x_0+h) by at most 2^{-33}.

If h cannot be expressed exactly in binary form, there is a numerical advantage in generating x by integrating $x' = 1$. Owing to the high digital accuracy afforded by the "bridging" values of $2^m q$ which are carried over from one step to the next, the accumulation of rounding-off errors in x occurs much less rapidly than it would if x were obtained by the repeated addition of h.

4-73 Definition of the process. The process is defined by the equations below. y_{i0} is the value of the ith variable at the beginning of a step; y_{i4} is its value at the end of the step. While the $2^m k_{ip}$'s for one value of p are being

calculated by the auxiliary routine, the corresponding y_{ip} and $2^m q_{ip}$ (i = 1...n) are stored. The quantities r_{ip} are only used in the formation of the corresponding y_{ip} and q_{ip}, and do not need to be carried over to the following value of p. Each r is rounded off to the same number of places as y.

y_{i4} and q_{i4} become y_{i0} and q_{i0} for the following step. The scale factor 2^m employed in storing k and q is left out for simplicity.

$$
\left.
\begin{aligned}
k_{i0} &= hf_i(y_{00}, y_{10}, \ldots.) \\
r_{i1} &= (1/2)k_{i0} - wq_{i0} \\
y_{i1} &= y_{i0} + r_{i1} \\
q_{i1} &= q_{i0} + 3r_{i1} - (1/2)k_{i0} \\
k_{i1} &= hf_i(y_{01}, y_{11}, \ldots.) \\
r_{i2} &= (1 - \sqrt{1/2})(k_{i1} - q_{i1}) \\
y_{i2} &= y_{i1} + r_{i2} \\
q_{i2} &= q_{i1} + 3r_{i2} - (1 - \sqrt{1/2})k_{i1} \\
k_{i2} &= hf_i(y_{02}, y_{12}, \ldots.) \\
r_{i3} &= (1 + \sqrt{1/2})(k_{i2} - q_{i2}) \\
y_{i3} &= y_{i2} + r_{i3} \\
q_{i3} &= q_{i2} + 3r_{i3} - (1 + \sqrt{1/2})k_{i2} \\
k_{i3} &= hf_i(y_{03}, y_{13}, \ldots.) \\
r_{i4} &= 1/6(k_{i3} - 2q_{i3}) \\
y_{i4} &= y_{i3} + r_{i4} \\
q_{i4} &= q_{i3} + 3r_{i4} - (1/2)k_{i3}
\end{aligned}
\right\} \quad i = 1....n
$$

The coefficient w appearing in the expression for r_{i1} is not critical. The best value is actually 1, but G1 and G2 use the value 1/2, as it simplifies the program.

An example of the use of these subroutines is given in Section 7-6.

4-8 Processes. Interpretive subroutines.

There are in the library a number of subroutines which, when called in, execute series of operations according to sets of parameters in the store. The codes by which these parameters are interpreted are determined by the design of the routines themselves, and are arranged to simplify the coding of such operations as the handling of complex numbers and numbers in floating point form (see below).

These subroutines are usually called in by the method used for the closed type, the parameters following the orders which call in the routine. The routines do, however, form a distinct class, and have been labelled "interpretive." Such a routine is defined as one which executes an operation defined by each parameter according to a code which is independent of the position of the parameter in the series. Usually the series is of indefinite length, being terminated by a special parameter.

Each parameter may be regarded as an "order," and thus the use of interpretive routines effectively extends the order code of the machine by increasing the complexity of the operations which may be performed in response to a single "order." The resulting gain in expediency of programming is offset by an increase in the time required by the machine to carry out the calculation, due to the higher percentage of orders concerned purely with organizing the operations.

4-81 Operations on complex numbers. Subroutines B1 and B2 are interpretive subroutines which enable operations to be performed on complex numbers whose real and imaginary parts are stored in consecutive long storage locations. The orders which define operations on complex numbers are placed in the master routine directly after the orders used to call in the subroutine. By the use of these subroutines the processes of addition, subtraction, transfer, and shifting may be carried out on complex numbers. Subroutine B2 will also carry out complex multiplication. The order code used is the normal order code of the EDSAC with certain small exceptions described fully in Part II.

Further subroutines dealing with complex numbers are described in the next paragraph.

4-82 Floating point subroutines. One difficulty which arises in programming complicated problems is the control of the magnitudes of the numbers involved. With the binary point at the extreme left-hand end of the accumulator, repeated addition may cause the accumulator to overflow at the left-hand end and repeated multiplication may cause loss of significant figures at the right-hand end. To prevent this, it is necessary to place the number in a suitable digital position within the arithmetical unit. In complicated programs this may be difficult or impossible to estimate in advance. Subroutines have therefore been prepared that will automatically adjust the scale factors associated with particular numbers or groups of numbers.

These subroutines carry out arithmetical operations with real or complex numbers expressed in the "floating decimal" form $a \cdot 10^p$, where $|a|$ is restricted to lie between two limits such as 1 and 10. In this form it is possible to represent numbers having a wide range of values fairly accurately over the entire range. The digits representing the exponent and those representing the numerical part together form the digits of one long number (or two long numbers if a is complex).

The main library subroutines dealing with numbers in floating decimal form consist of the following three groups.

A1 - A4

A3 and A4 are two versions of a subroutine to carry out special arithmetical operations (described in detail in the specification of A3 in Part II) on real numbers expressed in the following standard floating decimal form:

$$X = X_0 \cdot 10^p,$$

where X_0 is a seven decimal-digit number and p is an integer such that

$$4 > |X_0| \geq 0.4,$$
$$512 > p \geq -512.$$

The two parts of the number X are packed into a single long storage location, ten digits being allocated to the signed exponent, p, and the remaining 24 to the numerical part, X_0. That is, in the store, X is represented by the number $2^{-12} \cdot X_0 + 2^{-9} \cdot p$.

The subroutines unpack each number when it is required and place the numerical part and the exponent in separate long storage locations or "registers" ready for the special arithmetical operations to be performed. Similarly, answers are packed up and stored if they are not to be used again immediately. The special Read and Print subroutines A1 and A2 provide these packing facilities for the input and output of data to be used by A3 and A4. In A1, p is further restricted to the range $256>p>0$.

A5 - A8

A5 and A8 are similar to A3 and A4 but operate on complex numbers expressed in the following standard floating decimal form:

$$Z = Z_0 \cdot 10^p = (X_0 + iY_0) \cdot 10^p,$$

where X_0 and Y_0 are seven decimal-digit numbers and p is an integer such that

$$4 > \left| \sqrt{X_0^2 + Y_0^2} \right| \geq 0.4,$$

$$2048 > p \geq -2048$$

Two long storage locations are used to hold one such complex number, the first 28 digits of each representing X_0 and Y_0, and the remaining total of 12 being used for the exponent, p. That is, in the store, the number Z is represented by $2^{-2} \cdot X_0$ in rD and $2^{-2} \cdot Y_0$ in (r+2)D, each rounded off to 28 binary places. The last 6 digits of each of these storage locations contain the most and least significant halves respectively of the 12-digit integer p, the left-hand digit of which is treated as a sign digit.

A6 and A7 are special Read and Print routines for the input and output of complex data to be used by A5 and A8. In A6, p is further restricted to the range $2048>p\geq 0$.

A9 - A11

A11 is an interpretive subroutine which performs the arithmetical operations of addition, subtraction, multiplication, and transfer on real numbers expressed in floating decimal form, in accordance with a code of program parameters detailed in the specification in Part II. Each number is expressed in the form $a \cdot 10^p$ and is represented in the store by the long or short number $a \cdot 2^{-11} + p \cdot 2^{-6}$.

A9 and A10 provide for the input and output of data to be used by A11.

One other subroutine, K8, also uses floating point arithmetic for special operations on power series. These are described in Part II.

Floating point subroutines help to preserve accuracy by retaining a fixed number of significant figures in the most advantageous position within the machine. They can, however, do nothing to prevent the inherent loss of accuracy which results from the subtraction of two nearly equal quantities. For example, in the subtraction

$$3.214567 \times 10^5 - 3.214032 \times 10^5 = 0.000535 \times 10^5,$$

the difference is afterwards converted to standard form (that is, 0.535000×10^2),

before use in another calculation. The last three figures in this case are meaningless and the accumulation of such nonsignificant figures may still present a problem for the programmer.

A1 - A8 were developed for the problem of locating zeros of arbitrary real or complex functions by the "root squaring" method (see Section 7-8). A9 - A11 are intended for more general use.

4-83 Packing of orders used with interpretive subroutines. It often happens that the "orders" used in connection with an interpretive subroutine are few in number and refer only to a limited number of addresses. This is especially so with vector operations, where a single address can be used to specify a complete vector or matrix, but it is by no means restricted to such cases.

In such an event space may sometimes be saved by packing two or more orders into a single storage location. The associated interpretive subroutine must then also incorporate unpacking facilities. A special input subroutine must also be provided unless the orders used are translated into the form normally accepted by the initial orders - a laborious process if many orders are involved.

No subroutines using this principle have yet been included in the library but a short account of one problem to which these methods have been applied will be found in Appendix D.

CHAPTER 5

PITFALLS

Even a first-class computer will sometimes make a mistake (although he will not allow it to go undetected for long). In the same way a programmer will sometimes make a mistake in the master routine, in a subroutine, or in the make-up of the tape. Some mistakes may cause the answer to be in error. Others may cause the machine to obey a wrong sequence of orders, or to try to obey some constant order or pseudo-order not intended for such use. In the latter case the machine will stop on a meaningless order, or perhaps go into a closed loop. The machine may print a number of symbols or it may print nothing at all.

Experience has shown that such mistakes are much more difficult to avoid than might be expected. It is, in fact, rare for a program to work correctly the first time it is tried, and often several attempts must be made before all errors are eliminated. Since much machine time can be lost in this way a major preoccupation of the EDSAC group at the present time is the development of techniques for avoiding errors, detecting them before the tape is put on the machine, and locating any which remain undetected with a minimum expenditure of machine time.

Library subroutines are all checked on the machine before being put into the library and are presumably free from error. This in itself would be a sufficient reason for having a library, quite apart from any other considerations. When subroutines are specially made for a particular program it is good practice to test them beforehand by means of short programs constructed for the purpose.

It is easier to avoid and detect errors if the program is drawn up in an orderly and logical manner. For example, if six quantities x_1, x_2, x_3, y_1, y_2, y_3 occur, they should be placed in consecutive storage locations and not scattered about the store. Similarly, orders and pseudo-orders used for counting purposes should be arranged on some plan and not placed at random in the store. When drawing up a complicated program the programmer should not hesitate to copy it out in a more logical layout whenever necessary. The parallel case of hand computation will suggest itself; good computers usually pay great attention to the arrangement of their work sheets.

It is of great assistance, both to the programmer and to a person checking the program, to provide notes describing the actions of the orders, as is done for all library subroutines (see Part III). The notation for entry points, etc., given at the beginning of part III, is also designed to help in understanding programs.

5-1 Proofreading of programs. Points to be checked.

Some idea of the types of mistake which can occur is given by the following list of points that should be checked before a program is punched. Many of these are of a purely clerical nature, and could be checked by a person without great mathematical ability. Others may require an understanding of the particular calculation.

1. No two subroutines may occupy the same storage locations, unless one is only used temporarily before the other is inserted.

2. All conditions contained in the specification of each library subroutine used must be met. For example, if it is necessary that the subroutine start in an even location, this point should be checked, and it should be made certain that all parameters have been correctly specified.

3. When calling in a closed subroutine, (a) the accumulator must first be cleared, (b) the A and G orders must specify the correct addresses.

4. Where necessary, addresses must be corrected after renumbering.

5. Counting operations should give the correct number of repetitions, and control must be transferred to the correct point when repeating a cycle.

6. The program should be prepared so as to leave a location for any order which is to be planted by the program itself. This is usually done by writing a dummy order such as (Z F) or (P F).

7. Control must be directed to the correct place to start the program.

8. No item of information in the store should be overwritten until it is no longer required. In particular, no wanted information should be left in a location that is used as a working position by a subroutine.

9. The contents of the multiplier register must not be assumed to be unaltered by a subroutine.

5-2 Location of mistakes in a program.

It might be thought that a good way of finding errors in a program would be to make the machine proceed order by order under the control of the "Single E.P." button (see Section 6-4), and to study the numbers in the machine by watching the monitors attached to the arithmetical unit and store. This, however, usually turns out to be a very slow and inefficient process, especially as the numbers are displayed in binary form. Methods have therefore been developed which permit the machine to proceed unhindered by the operator, whilst printing on the teleprinter a permanent record that can be studied at leisure, and that will assist in understanding the nature of the mistake.

One such method is to wait until the machine has stopped (or to stop it deliberately) and then, without clearing the whole store, to insert (by pressing the starter button again) a small program which will print, in suitable form, the contents of part of the store. This has come to be known as the "post-mortem" method. Tapes are kept available near the EDSAC for printing the function letters, or address parts, of orders in consecutive storage locations. Programmers may also prepare their own post-mortem tapes.

This method yields only a static picture of the store as it was when the calculation stopped. Other methods have been derived to provide information about the whole course of the calculation. These necessarily involve modifying the program to cause the extra printing, and therefore a new tape must be prepared and presented to the machine. This, however, is no hardship, since the machine will read an average tape in about a minute, and the preparation need not take more than a few minutes of the programmer's time.

5-21 Method using extra output orders. One simple and very useful plan is to place an output order at the beginning of the master routine and in front of each subroutine so that the completion of the various stages of the program

will be recorded by the printing of suitably chosen symbols. If by reason of a programmer's blunder the machine stops in the middle of the program, the symbols printed will enable the error to be localized. Letter and figure shifts must also be inserted if letters are required for indication purposes while the ordinary printing of numbers called for by the program takes place correctly.

As an example the program given in Section 7-2 is repeated in Section 7-4 with the extra print orders incorporated.

When the program has been made to work correctly, the extra printing may be eliminated by omitting the extra orders from the tape. If an assembly subroutine (or the second method of assembling a program described in Section 7-1) is used, no renumbering is necessary. Note that two extra orders must then be placed in front of any subroutine which is required to have its first order in an even location.

It is a good plan to include extra printing of the kind described here in all new programs when they are first drawn up rather than to wait until the program has been tried and found to fail.

5-22 Subroutines for checking programs. Methods like the foregoing are too limited to deal with many of the questions that arise. In such cases a considerable modification of, or addition to, the original program is necessary. It has been found possible to construct subroutines which incorporate all these modifications and additions, and which are sufficiently general to be applied to any program. These form category C of the library and fall into two classifications, those that check the sequence of operations and those that check the numbers operated upon.

For their operation, these methods depend largely on the technique used in interpretive subroutines (see Section 4-8), namely, the repeated selection of "parameters" from another part of the store. In this case, however, the "parameters" are simply the orders of the original program, and they are selected and carried out in exactly the same manner and sequence as if they were being obeyed directly. The purpose here is not to enable new operations to be initiated by each order, but to make it possible (by suitably designing the checking routine) to interpose the printing of extra information for checking purposes. Another technique employed is the planting in the original program of an E order (or "blocking order") which switches control to the checking routine.

For a full discussion of checking routines see ref. 13.

5-23 Subroutines that check the sequence of orders carried out. Certain checking subroutines print the function letters of orders as they are obeyed, this being the most convenient way of checking the sequence of operations. Subroutine C11 is the simplest; it checks through the whole program without a break. Letters are printed in a line across the page until a transfer of control occurs, when a new line is started. An example of the use of C11 is given in Section 7-5.

C7, C9, and C12 are rather more elaborate versions of C11, and provide for the suppression of checking or printing during irrelevant parts of the program to save time.

5-24 Subroutines that provide numerical checks. Some mistakes in programs cause the numbers operated upon to be in error, without immediately

affecting the sequence in which the orders are obeyed. It cannot therefore be assumed that if a program apparently operates correctly it is giving correct results, and careful numerical checks must always be applied. Moreover, the diagnosis of such mistakes can be as difficult as that of mistakes which affect the order sequence.

A numerical fault may be due to a mistake in a single order or in a constant, or to a more fundamental mistake, such as a wrong choice of scale factors that causes a number in the machine to exceed unity. Some knowledge can be gained by printing several intermediate results, and it is usually advisable to include such extra printing in the first draft of any program.

If this is insufficient, subroutines C1, C8, and C10 can be used, in conjunction with the program, to cause the printing at frequent intervals of numbers involved in the calculation.

5-3 Counting operations.

Even the simplest of programs usually contains cycles of orders which have to be repeated a certain number of times. The methods commonly employed to ensure that the correct number of repetitions is carried out are explained in Appendix E.

Some programs involve rather complicated counting operations, and it is easy to make an error in these. As a means of simplifying the preparation of such programs a number of counting subroutines, U1 to U5, have been incorporated in the library. The use of the closed counting subroutines U1 and U4 undoubtedly enables the layout to be improved but at the expense of making the program rather longer than it otherwise would have been. The open routines U2, U3, and U5 will probably be of more general utility and their use should enable many errors of counting to be avoided.

CHAPTER 6

USE OF THE EDSAC AND ITS ASSOCIATED EQUIPMENT

6-1 Tape punching and editing facilities.

This section will deal with the preparation of a punched tape from a program sheet which has been prepared in the manner described in the previous sections. So far no attempt has been made to set up in the Mathematical Laboratory an elaborate organization for punching tapes. Most of the punching is done by the users themselves, although some assistance is available when necessary. The problems dealt with so far have not required a great deal of input, although in one or two cases several hundred numbers have had to be read for one run.

The main pieces of equipment provided for the preparation and editing of tapes are described below.

6-11 Keyboard perforator. Use is made of standard 5-hole teleprinter keyboard perforators modified so as to conform to the special EDSAC code. Several are available for use.

6-12 Tape Duplicator. This name is given to a device used (a) to prepare a corrected copy of a tape and (b) to build up a complete program tape from number sequences, the master routine, and subroutines which have been punched separately. It incorporates a keyboard perforator which has been fitted with five solenoids (one for each hole) in addition to the usual keys. The solenoids are linked to a tape reader and the operator may prepare a new tape partly by operating the keys in the ordinary way and partly by copying data from a separate piece of tape placed in the tape reader. When copying he can make the duplicator run continuously or in single steps. If he wishes he can also make the tape reader advance the original tape without copying it onto the program tape. If a switch marked "Ignore 11111" is closed any row of the original tape in which all five holes are punched will be passed over and omitted from the copy. This enables a row of five holes to be used as an erase sign when punching. If another switch marked "Ignore 00000" is closed the tape reader will automatically pass over blank tape. A further facility is a key which causes blank tape to be fed automatically from the punch.

Two duplicators are available and a standard tape reperforator for producing straight copies of a tape is also provided. Two tape readers and a changeover switch are provided with each duplicator for use when building up a complicated tape from short pieces.

6-13 Tape comparator. This device enables two tapes which are supposed to be identical to be checked one against the other. They are placed in separate tape readers and when a switch is depressed are advanced automatically as long as the symbols punched on them are identical. When a discrepancy is encountered the tape readers stop. Switches are provided for advancing either tape independently in single steps and for ignoring blank tape. Two comparators are available.

It is normal practice to punch the various number sequences, master routine, and subroutines which go to make up a program tape separately and to combine them later. Each part is punched twice and the two are checked by means of a comparator to make sure they agree. In this way errors of punching can be detected. Of the various errors which occur when preparing a problem for the EDSAC errors in punching are the least excusable.

6-2 Storage of library subroutines.

Subroutines in the library are punched on colored tape so that they can easily be distinguished from program tapes, which should be white. Several copies of each subroutine are provided and when not in use each copy is rolled in a small cardboard box. The boxes are filed in serial order in a steel cabinet. The master copy of each subroutine is kept under lock and key and is used only when all existing copies of the subroutine are damaged. The master tape is then used to prepare further copies by means of a duplicator. All copies must be checked against the master, by means of a comparator, before being put into the library for general use.

6-3 EDSAC organization.

The following brief note on the organization of the machine room may be of interest. When a program tape is ready to be put on the machine the programmer writes out a ticket saying what he expects the machine to print and giving any other information which the operator may need. He then hangs the tape with its ticket from a clip running on a horizontal wire. The various tapes hanging from the wire form a queue and the machine operator puts them through the EDSAC in order, subject to any overriding instructions about priority. If the machine prints what is expected, the output sheet is placed in a rack ready for collection by the programmer. If the machine stops unexpectedly the operator notes on the ticket the place at which it has stopped (that is, the number in the sequence control register) and then proceeds to the next tape in the queue.

A number of test tapes are available by means of which the operator can make regular checks on the operation of the machine. Should one of these tapes reveal an error the maintenance staff is called upon to rectify the fault.

6-4 EDSAC controls.

The EDSAC requires no preliminary "setting up" for a particular program. The procedure when a program tape is to be run is as follows.

1. The tape is inserted in the tape reader.
2. The "clear store" button is pressed to clear out any information previously in the store.
3. The start button is pressed and causes the initial orders to be placed in the store. Under control of these orders the tape is then read and the program carried out according to the orders on the tape.

The purpose of the "clear store" button is to ensure that the machine is always in the same condition when a program is started, and should therefore always react in the same way to the same tape. If, during a program, the

machine is suspected of being faulty, the program can be repeated and if consistent results are not obtained the fault is known to lie with the machine and not with the program. Other push button controls provided for occasional use are

Stop	- stops the machine in exactly the same way as a Z order.
Reset	- used to restart the machine after a "stop" operation or a Z order.
Single E.P.	- may be used after a "stop" operation or a Z order. Every operation of this control causes the machine to execute one single order.

CHAPTER 7

EXAMPLES

Sections 7-1 to 7-5 describe how some relatively simple calculations might be programmed for solution on the EDSAC, making use of library subroutines. Sections 7-6 to 7-8 are examples of actual problems prepared for the EDSAC. In all the program sheets the notation used is that described at the beginning of Part III.

7-1 Example 1. Calculation of $e^{-\sin x}$ (see Section 4-4).

This process causes a series of positive numbers x (<1) to be read from the tape and the quantities $e^{-\sin x}$ to be calculated and printed to nine decimal places. Four values of x have been chosen: 0.1234, 0.986, 0.74281079, and 0.84314763. Each of these will be read in turn and the corresponding value of the function printed before the next value of x is read. After printing the fourth value of the function the machine will stop. Five library subroutines and a master routine specially constructed for this problem are used and positioned in the store as follows:

Subroutines, etc.	Location of first order	Number of storage locations occupied
R9	56	15
T7 (sine, rapid)	72*	36
E4 (exponential, fast)	108*	36
R3 (input one signed decimal fraction)	144*	41
P11 (print signed decimals in preset layout)	185	55
Master routine	240	

*First order must be in an even location.

7-11 Make-up of tape.

R9	R9 begins with P K T 56 K so that it is automatically placed in locations 56 to 70.
P F	Extra pseudo-order put in to bring first order of T7 into an even storage location.
space P Z	
T7	
space P Z	
E4	
space P Z	
R3	
space P Z	

G K	Places in 42 the address specified in the current transfer order.
T 45 K	Sets transfer order so that parameters following go into 45 to 48.
A 258 D (H parameter) P 20 F (N parameter) P 47 θ (M parameter) P3104 F (Δ parameter)	4 columns 3 spaces between columns digit layout: 4 digits, space, 5 digits] parameters used by P11
P11	P11 begins with T Z, so that the address stored in 42 is replaced in the transfer order. P11 is therefore placed in 185 to 239.
space P Z	
Master routine	
space P K	
E 240 K P F	When this control combination is read, control is switched to 240, which contains the first order of the master routine.
1234 + 986 + 742 81079 + 843 14763 +	values of x. These are not placed in the store during input of orders but are read one by one under the control of R3 when that subroutine is called in by the master routine.
1 -	the master routine is drawn up so as to stop the machine if the number read from the tape is negative, and this stops the program.

7-12 Master routine.

Start		G		K	
17 →	0	A		θ	calls in R3, which reads x from the tape and places it in OD.
	1	G	144	F	
R3 →	2	A		D	x to accumulator.
	3	E	5	θ	stops process if x is negative.
	4	Z		F	
3 →	5	R		D	$\frac{1}{2}$x to 4D ready for T7.
	6	T	4	D	
	7	A	7	θ	calls in T7, places $\frac{1}{2}$ sin x in 4D.
	8	G	72	F	
T7 →	9	S	4	D	-sin x to OD.
	10	L		D	
	11	T		D	
	12	H		D	-sin x to multiplier register ready for E4.

	13	A	13	θ	calls in E4, forms $e^{-\sin x}$ and places it in OD.
	14	G	108	F	
E4 →	15	A	15	θ	calls in P11, prints $e^{-\sin x}$.
	16	G	185	F	
P11 →	17	E		θ	transfers control to 0θ as accumulator is clear.
		E	69	K	places decimal round-off number required by P11 (i.e., 5.10^{-10}) in 258D.
		T	258	D	
			9	π	

7-13 Notes.

1. "Space" indicates that a few rows of blanks (there must be more than one) are left on the tape. The object is to enable the subroutines to be identified easily when checking the tape or making corrections (see Chapter 2). The control combination P Z which follows each space sets the initial orders back into the condition they were in before the space. If spaces are not required on the tape these control combinations should be omitted.

2. The machine can be stopped by pressing the stop button when reading the blank tape following the main program and can be restarted by pressing the reset button. This does not affect the content of the store. In certain circumstances it is convenient to divide the tape into two parts, an order tape, going as far as the main program and ending with a length of blank tape, and a number tape, beginning with P K E 240 K P F, followed by the numbers. The machine would then be stopped on blank tape after the master routine had been read, the number tape would be inserted in the tape reader with blank tape under the reading head, and the machine would be restarted by pressing the reset button. In this way a great many number tapes could be used with one order tape.

3. This program consists of a total of 202 orders, but only the 18 orders of the master routine have to be drawn up especially for this calculation.

7-14 Alternative method of making up a tape. In the above method of making up a program tape the subroutines automatically follow one another into the store head-to-tail. An alternative method is to place each subroutine into a definite place in the store by means of a control combination of the form T m K. If this method is used, the make-up of the tape is as follows:

R9

space

P K T 72 K

T7

space (If spaces are not required the P K's may be omitted.)

P K T 108 K

E4

space

```
P K T 144 K
[R3]
space
P K T 185 K
G      K
T  45  K
A 258  D
P  20  F
P  47  0
P3104  F
[P11]
space
P K T 240 K
[Master routine]
space
P K E 240 K P F
      1234  +
       986  +
742  81079  +
843  14763  +
         1  -
```

This method of making up a tape puts the subroutines into the same locations as the previously described method. A few spare locations can, however, be left between the subroutines if desired. This reduces the possibility of error arising because of a miscalculation of the locations required by a subroutine and enables corrections involving a slight increase of length to be made to a subroutine without renumbering. Such corrections often must be made to subroutines which have been specially constructed for the program.

Another method of making up a tape for a complete program is exemplified in Section 7-3.

7-2 Example 2. Calculation of π by evaluation of definite integral.

The formula chosen is $\frac{1}{4}\pi = \int_0^1 (1 + x^2)^{-1}\, dx$. In order that all numbers concerned in the calculation shall be less than unity this must first be written in the form

$$\frac{\pi}{10} = \int_0^{1/2} \frac{3/16}{(15/16)(0.25 + x^2)}\, dx.$$

Evaluation of the integral is carried out by subroutine Q2, which requires an auxiliary subroutine to calculate the integrand. This must be so designed that it evaluates the integrand for the value of x given by C(OD) and places it in OD; it is called into use by Q2 as required. The auxiliary subroutine is given in full in Section 7-23. It requires a division subroutine, and the one chosen is D6.

The program requires two other subroutines, R9, which is used by Q2, and P1, which is used to print the result, which consists of one ten-digit number.

7-21 Make-up of tape.

R9	R9 begins with P K T 56 K, so that it is automatically placed in locations 56-70.
space	
P K T 72 K	Sets transfer order so that master routine goes into store starting at 72.
Master routine	
space	
P K T 96 K	
Auxiliary subroutine	Auxiliary goes into store starting at 96.
space	
P K T 112 K	Q2 goes into store starting at 112.
G K T 45 K	Sets transfer order so that following parameters, required by Q2, go into 45 and 46.
P 92 D (H parameter) G 96 F (N parameter)	
Q2	Q2 begins with T Z, so that transfer order is reset after planting of parameters.
space	
P K T 164 K	P1 goes into store starting at 164.
P1	
space	
P K T 185 K	D6 goes into store starting at 185.
D6	
E 72 K P F	When this control combination is read control is transferred to the order in 72, that is, to the beginning of the master routine.

7-22 Master routine.

	G	K	
Start → 0	0	15 θ	figures
1	0	13 θ	carriage return] (see note 5)
2	0	14 θ	line feed]
3	T	20 $\pi\theta$	] Sets limits of integration: 0 to 20$\pi\theta$,
4	A	19 θ	1/2 to 22$\pi\theta$. (Note: started with accumu-
5	T	22 $\pi\theta$	] lator clear.)

	6	A	6	θ	calls in Q2, which places integral in OD.
	7	G	112	F	
Q2→	8	A	8	θ	calls in P1, which prints integral.
	9	G	164	F	
	10	P	10	F	parameter for P1.
P1→	11	O	15	θ	extra output order to print last figure.
	12	Z		F	
	13	θ		F	carriage return
	14	Δ		F	line feed
	15	π		F	figures
	16	K	2048	F	$15 \cdot 2^{-4}$
	17	R		F	$4 \cdot 2^{-4}$
	18	E		F	$3 \cdot 2^{-4}$
	19	I		F	$8 \cdot 2^{-4}$
	20	(P		F)	set by 3
	21	(P		F)	
	22	(P		F)	set by 5
	23	(P		F)	

7-23 Auxiliary subroutine

	G		K	
0	A	3	F	
1	T	15	θ	plant link
2	H		D	x^2
3	V		D	
4	A	17	M	
5	Y		F	
6	T		D	$0.25 + x^2$ to OD
7	H	16	M	
8	V		D	
9	Y		F	
10	T	4	D	$15/16 \cdot (0.25 + x^2)$ to 4D
11	A	18	M	
12	T		D	3/16 to OD
13	A	13	θ	calls in D6, which places integrand in
14	G	185	F	OD.
D6→15	(Z		F)	link

Notes:

1. If spaces are not required the P K's may be omitted.

2. The mechanism of the teleprinter output system is such that an O order sets up on the printer the character next to be printed and at the same time prints the character set up by the previous O order. Thus, at the end of a program an extra O order must be supplied in order to print the final character. Some print subroutines do this automatically by printing one or more spaces after each number. P1, used in this program, does not, and an extra O order is therefore supplied in the master routine.

3. "Carriage return" must precede "line feed," since it takes longer than the time required for other teleprinter operations.

4. The master routine must start at an even location, since locations 20θ and 21θ are combined to form one long storage location.

7-3 Alternative method for Example 2.

The example given in Section 7-2 will now be repeated in a revised form making use of assembly subroutine M1. The components of the program are

H sequence,
Master routine,
Auxiliary subroutine,
Library subroutines R9, Q2, P1, and D6.

R9, however, is not dealt with by M1 but is automatically placed in its usual position (locations 56-70). The H sequence consists of a number of pseudo-orders which, in Section 7-2, were included at the end of the master routine.

Storage space is allocated as follows:

56-70	R9		
71-75	unused		
76	reference order for	master routine	see Section 4-62
77	do.	auxiliary	
78	do.	Q2	
79	do.	P1	
80	do.	D6	
82-97	M1		
98-	H sequence		

7-31 Make up of tape.

R9	R9 begins with P K T 56 K, so that it is placed in locations 56-70.
space	
P K T 82 K	First order of M1 goes into 82.
M1	
P 76 F	Reference order of master routine goes into 76.
T 98 K	First order of H sequence goes into 98.
space	
P Z G K	
E 82 K T F	Calls in M1, which places P 98 F in 45.
H sequence	Placed with first order in 98.
space	
P Z G K	
T 92 K Tϕ	Sets M1 ready to deal with master routine.
E 82 K I F	Calls in M1, which places reference order in 76.

parameters used by M1

Master routine

space

P Z G K
E 82 K P F — Calls in M1, which places reference order in 77.

Auxiliary

space

P Z G K
E 82 K P F — Calls in M1, which places reference order in 78.

T 45 K
P 72 D
G 1 ϕ — Plants parameters required by Q2.

Q2

space

P Z G K
E 82 K P F — Calls in M1, which places reference order in 79.

P1

space

P Z G K
E 82 K P F — Calls in M1, which places reference order in 80.

D6

E 25 K
E ϕ P F — Sends control to the first order of the master routine.

7-32 H sequence.

H	0	Δ		F	line feed
	1	θ		F	carriage return
	2	K	2048	F	$15 \cdot 2^{-4}$
	3	R		F	$4 \cdot 2^{-4}$
	4	E		F	$3 \cdot 2^{-4}$
	5	I		F	$8 \cdot 2^{-4}$
	6	π		F	figure shift

7-33 Master routine.

		G		K	
Start →	0	O	6	H	figures
	1	O	1	H	carriage return
	2	O		H	line feed

```
        3 |  T  72  D |
        4 |  A   5  H |   Sets limits of integration: 0 to 72D,
        5 |  T  74  D |     1/2 to 74 D.
        6 |  A   6  θ |
        7 |  G   2  φ |   calls in Q2.
Q2 ---> 8 |  A   8  θ |
        9 |  G   3  φ |   calls in P1.
       10 | ‖P  10  F |   parameter for P1.
P1 ---> 11 |  O   6  H |   extra output order to print last figure.
       12 |  Z      F |
```

7-34 Auxiliary subroutine

```
          |  G      K |
        0 |  A   3  F |
        1 |  T  15  θ |   plant link
        2 |  H      D |
        3 |  V      D |   x^2 to accumulator
        4 |  A   3  H |   4·2^-4 + x^2
        5 |  Y      F |
        6 |  T      D |   0.25 + x^2 to OD
        7 |  H   2  H |   15·2^-4
        8 |  V      D |
        9 |  Y      F |
       10 |  T   4  D |   15/16·(0.25 + x^2) to 4D
       11 |  A   4  H |
       12 |  T      D |   3/16 to OD
       13 |  A  13  θ |   Calls in D6, which places the integrand
       14 |  G   4  φ |     in OD.
D6 ---> 15 | (Z      F) |  link
```

7-4 Example 2, with extra print orders for checking.

The example of Section 7-2 is given below in a modified form in which extra print orders are included as described in Section 5-21. The letters printed by the orders preceding the various subroutines are as follows: A for the auxiliary, Q for Q2, P for P1, and D for D6. These letters are stored in locations numbered, for convenience, with respect to the code letter M. In addition, an order which operates the letter-shift of the teleprinter is included in front of Q2 (the subroutine operated first), and an order which puts it back on to figures is included in front of P1.

This example will print the following:

QADADADADADP3141592653

7-41 Make-up of tape.

R9

T 47 K
P 228 F (M parameter)

space

```
P K T 72 K
[Master routine]
space
P K T 96 K
O M                          Print A. This order goes into 96.
[Auxiliary subroutine]       Auxiliary starts at 97.
space
P K T 114 K
O 4 M                        Letter shift  ]  extra print orders in 114
O 1 M                        Print Q       ]    and 115.
G K T 45 K
P 90 D (H parameter)         ]
G 96 F (N parameter)         ]  Parameters for Q2.
[Q2]                         Q2 starts at 116.
space
P K T 168 K
O 2 M                        Print P       ]  extra print orders in 168
O 5 M                        Figure shift  ]    and 169.
[P1]                         P1 starts at 170.
space
P K T 191 K
O 3 M                        Print D. This order goes into 191.
[D6]                         D6 starts at 192.
space
        | P K T 228 K
0M      | A F                Denotes auxiliary  ]
1       | Q F                Denotes Q2         ]  Code letters to be
2       | P F                Denotes P1         ]  printed by the extra
3       | D F                Denotes D6         ]  print orders.
4       | K 2048 F           Letter shift
5       | π F                Figure shift
        | E 72 K P F         Transfers control to the master routine
```

7-5 Application of checking subroutine C11 to Example 2.

C11 calls for a small alteration to the original tape of Section 7-21. It is merely necessary to remove the control combination at the end of the tape and replace it by C11. The point of entry into the master routine is now specified by an E order at the end of the tape, following C11. In this example it is necessary to avoid the first order of the master routine because this order causes a figure shift, and the teleprinter is set by C4 to print letters. Hence the tape is terminated by E 73 F. The end of the tape appears thus:

. . . . . .

D6

C11

E 73 F

The first few rows of printing produced by this tape would be as follows.

Printed by teleprinter	Corresponding orders	
O	master routine	1 and 2
TATAG	master routine	3 to 7
ATTSAUATAHVYTAG	Q2	0 to 14
ATHVAYTHVYTATAG	Aux.	0 to 14
ATSETSE	D6	0 to 7
SE		2 and 3
LE		13 and 14
TALTALE		8 to 14
TALTALERULATE		8 to 20
HSNAYG		25 to 30
UNATHSNAYG		21 to 30
UNATHSNAYG		21 to 30
UNATHSNAYG		21 to 30
UNATHSNAYGSVTE		21 to 34
E	Aux.	15
HVAYTASG	Q2	15 to 22
AUATAHVYTAG		4 to 14
ATHVAYTHVYTATAG	Aux.	0 to 14
ATSETSTE	D6	0 to 7
SE		2 and 3
LE		13 and 14
TAL . . . etc.		8 to . . .

7-51 Notes:

1. The carriage return and line feed at the beginning of the master routine each affect the teleprinter after the corresponding "O" has been printed. The result of this is that both O's are printed in the same position on the paper, and the following letters on the next line.

When P1 is reached, the decimal digits which it sends to the teleprinter will be printed as letters immediately after the "O" indicating the print order in P1. They will not, however, be the digits obtained in the original example, because P1 employs the F order and therefore fails when C11 is used.

2. The exact number of repetitions of the groups of orders 8 to 14 and 21 to 30 of D6 depends on the numbers operated upon. The above example shows a probable course of the calculation.

7-6 Example of integration of an ordinary differential equation.

7-61 Statement of problem.

The equation considered is

$$\frac{dy}{dx} = \frac{y(1+2y-4x)}{x(y-x+N)},$$

where N is a constant. This equation occurs in theoretical astrophysics.

In the vicinity of the origin, a solution for any given value of N has the behavior $y = (Bx)^{1/N}$, where B is arbitrary. Solutions are required for a set of values of N and, for each value of N, for a set of values of B. Each solution is to be tabulated at an interval of 0.01 in x until either $y>1$ or $dy/dx<-1$, values of y being correct to five decimals.

7-62 Method.

The formula for dy/dx is formally indeterminate at $x = 0$, so that it is necessary to start the numerical integration from some small value x_0 of x, at which the value y_0 of solution y can be evaluated from a series expansion. This starting point was taken as $x_0 = 0.01$; the corresponding values of y_0 for different values of B were calculated separately and furnished to the machine as part of the number input. The program is so arranged that the machine evaluates automatically the whole set of solutions for a given value of N and different values of y_0.

An input subroutine is required to take in the values of x_0, N, and the set of values of y_0 for which solutions are required. Subroutines for step-by-step integration of a first-order differential equation and for printing are also required. The subroutines used are R3, G1, and P11. G1 requires an auxiliary subroutine for calculating $2^m h(dy/dx)$, and this auxiliary subroutine has to be programmed in detail; it involves a division process and for this D7 is used. Assembly subroutine M1 is used to organize the various subroutines into a complete program.

7-63 Allocation of storage locations.

Locations 0D, 4D, 6D are used by D7, P11, and R3; they are also used by G1, which in addition requires six storage locations for y, x, $2^m h(dy/dx)$, $2^m h$, $2^m q_1$ and $2^m q_2$ (q_1 and q_2 being intermediate quantities calculated in the course of the use of G1). These have been taken as 10D, 12D, ..., 20D. Storage for N, x_0, and a round-off number are provided at 22D, 24D, and 26D.

7-64 Auxiliary subroutine.

This subroutine must put

$$2^m h \frac{dy}{dx} = 2^m h \frac{y(1 + 2y - 4x)}{x(y - x + N)}$$

into location 14D. The quantities x and y are in the range (0,1) and, for the solutions required, N and $2^m h$ are less than 1. However, 4x may exceed unity, so we must introduce a scaling factor 2^{-2} and calculate

$$2^m h \frac{y(0.25 + 0.5y - x)}{x(0.25(y-x) + 0.25N)}$$

The sequence of operations by which this quantity is evaluated must be planned with care, to ensure that all intermediate quantities remain within the capacity of the accumulator. The method adopted is to test whether $|y(0.25 + 0.5y-x)| > x(0.25(y-x)+0.25N)$. If this condition is satisfied, the multiplication

of the numerator by $2^m h$ is done first and is followed by the division. If the condition is not satisfied, the division is carried out first and the result is multiplied by $2^m h$. The reason for this procedure is as follows. If two small quantities are multiplied and give a product less than 2^{-34}, this appears as zero, and subsequent division by a small number will still give a zero result although the correct result may be much greater than 2^{-34}. Hence in calculating ab/c, it is advisable to carry out the division first if $|b/c|<1$.

7-65 Master routine. This is straightforward, and stops the integration when either $y \geq 1 - 2^{-34}$ or $(dy/dx)<-1$. The value of x for which the integration is stopped is printed in brackets at the end of the table of results.

7-66 Mathematical checks. These are required to verify that the interval is small enough for the step-by-step integration process and that the solution is stable despite rounding-off errors. These checks are not programmed but must be carried out by hand outside the machine. Checks used are the evaluation, at selected points, of (dy/dx) from the differential equation and also from a central-difference formula. Further, results for one special case, namely N = 0.4, $x_0 = 0.074515$, $y_0 = 0.049793$, can be obtained from the tabulated solution of Emden's equation. When this problem was run on the EDSAC, agreement with these results to the required degree of accuracy was obtained.

7-67 Master routine.

		G		K	
	0	A		θ	call in R3 to read 2^3h
	1	G	4	φ	
R3→	2	A		D	2^3h to 16D
	3	T	16	D	
	4	A	4	θ	read round-off constant and send to 26D
	5	G	4	φ	
R3→	6	A		D	
	7	T	26	D	
	8	A	8	θ	read x_0 and send to 24D
	9	G	4	φ	
R3→	10	A		D	
	11	T	24	D	
	12	A	12	θ	read N
	13	G	4	φ	
R3→	14	A		D	send $\frac{1}{4}$N to 22D
	15	R	1	F	
	16	Y		F	
	17	T	22	D	
56→	18	A	24	D	reset x to x_0
	19	T	12	D	
	20	A	5	φ	form order 23**
	21	A		H	
	22	T	23	θ	
	23	(P		F)	becomes T q+20 F
	24	A	24	θ	read next y_0
	25	G	4	φ	

set to print new block*

read constants

evaluate one solution of equation

R3→26	A		D	y_0 to accumulator	evaluate one solution of equation
27	E	31	θ		
28	T		F	stop if $y_0<0$	
29	O		H		
30	Z		F		
27→31	T	10	D	set $y = y_0$	
32	T	18	D	clear 18D	
33	T	20	D	clear 20D	
47→34	T		F	clear acc.	one step of integration
35	A	10	D	y to 0D	print y
36	T		D		
37	A	37	θ	call in P11	
38	G	5	ϕ		
P11→39	A	39	θ	call in G1	
40	G	3	ϕ		
G1→41	A	10	D	jump to 48 if $y \geq 1 - 2^{-34}$	
42	A		2πH		
43	G	48	θ		
44	T		F	clear accumulator	
45	A	14	D	return to 34 if $y' \geq -1$	
46	A	16	D		
47	E	34	θ		
43→48	T		F	clear accumulator	
49	A	12	D	x to 0D	print (x)
50	T		D		
51	O	1	H	print (	
52	A	52	θ	call in P11	
53	G	5	ϕ		
P11→54	O	4	H	print)	
55	O	6	H	line feed	
56	E	18	θ	return to 18	

*See P11, note 7 (Part II).

**C(5ϕ) = G q F (if P11 starts in q), C(OH) = J 20 F. Thus C(5ϕ) + C(OH) = (G + J) q + 20 F = T q + 20 F.

7-68 Auxiliary subroutine. Puts 2^3h(dy/dx) in 14D where

$$\frac{dy}{dx} = \frac{y(0.25 + 0.5y - x)}{x(0.25(y-x) + 0.25N)}$$

	G		K		
0	A	3	F	plant link	
1	T	46	θ		
2	A	10	D	$\frac{1}{4}(y-x) + \frac{1}{4}N$ to 0D	
3	S	12	D		
4	R	1	F		
5	A	22	D		
6	Y		F		
7	T		D		

	8	H	12	D	denominator to 4D
	9	V		D	
	10	Y		F	
	11	T	4	D	
	12	A	10	D	$\frac{1}{4} + \frac{1}{2}y - x$ to 0D
	13	R		D	
	14	S	12	D	
	15	A	5	H	
	16	Y		F	
	17	T		D	
	18	H	10	D	numerator to 0D
	19	V		D	
	20	Y		F	
	21	U		D	
	22	E	25	θ	modulus of numerator to accumulator
	23	T	14	F	
	24	S		D	
22→	25	S	4	D	subtract denominator*
	26	E	36	θ	jump if $\lvert dy/dx \rvert \geq 1$
	27	T	14	F	clear accumulator
	28	A	28	θ	call in D7 to form (dy/dx)
	29	G	2	ϕ	
D7→	30	T	4	F	clear accumulator
	31	H	16	D	$2^3 h(dy/dx)$ to 14 D
	32	V		D	
	33	Y		F	
	34	T	14	D	
	35	E	46	θ	to link
26→	36	T	14	F	clear accumulator
	37	H	16	D	multiply numerator by $2^3 h$
	38	V		D	
	39	Y		F	
	40	T		D	
	41	A	41	θ	call in D7 to form $2^3 h(dy/dx)$
	42	G	2	ϕ	
D7→	43	T	4	F	clear accumulator
	44	A		D	$2^3 h(dy/dx)$ to 14D
	45	T	14	D	
35→	46	(Z		F)	link order

$\lvert dy/dx \rvert < 1$

$\lvert dy/dx \rvert \geq 1$

*In the range considered, the denominator is always positive.

7-69 H sequence, make-up of program and number tapes. H sequence:

	T	2πZ		
2				
3	‖P	F		(See Chapter 2 under the heading sandwich digit.)
	T	Z		
0	‖J	20 F		
1	‖K	F		

2	‖ P		D	2^{-34} (See Chapter 2 under the heading
3				sandwich digit.)
	T	4	Z	
4	‖ L		F	)
5	‖ R		F	= 1/4
6	‖ Δ		F	line feed

Make-up of program tape:

P K T 60 K
M1
P 52 F
T 76 K

space

P Z G K
E 60 K T F
H sequence

space

P Z G K
T 70 K Tφ
E 60 K I F
Master routine

space

P Z G K
E 60 K P F
Auxiliary subroutine

space

P Z G K
E 60 K P F
D7

space

P Z G K
E 60 K P F
G K T 45 K
P 12 D
P 4 F
P 4 F
P 4 F
P 2 F
P 1 φ
G1

preset parameters for G1

space

P Z G K
E 60 K P F
R3

space

P Z G K
E 60 K P F
G K T 45 K
A 26 D
P 25 F
P 45 θ
P 1552 F
P11

preset parameters for P11

Make-up of number tape:

P K E 25 K
E ϕ P F

08 +	2^3h
00000000005 +	round-off
01 +	x_0
4 +	N
---	y_0 first value
---	y_0 second value
.	
.	
.	
. etc.	
---	y_0 last value
1 -	stop

numbers

7-7 Evaluation of a definite integral.

7-71 Statement of problem. The following integral, which occurs in the theory of the ionization of an exponential atmosphere by solar radiation, is to be tabulated as a function of X and x for $X = 90°(-1°)50°$ and x = 200(20)280. The table is to be printed in five columns, each giving values of the integral for a fixed value of x.

$$\int_0^X \exp\left\{-x\,\frac{\sin\lambda - \sin X}{\sin\lambda}\right\}\operatorname{cosec}^2\lambda\,d\lambda.$$

7-72 Method. The integral is written in the following form, in which all quantities to be handled in the machine are numerically less than unity:

$$-\int_X^0 e^{-2^4 u}\,\operatorname{cosec}^2\lambda\,d\lambda,$$

where

$$u = \frac{x}{2^4}\,\frac{\frac{1}{2}\sin X - \frac{1}{2}\sin\lambda}{\frac{1}{2}\sin\lambda}.$$

The integration is performed by Simpson's rule, using subroutine Q1. The following subroutines from the library are also used: E3, P11, T7, and D7. In addition a master routine, and a subroutine for computing the integral need

to be constructed specially. The integrand rapidly becomes small as λ decreases, and the integration is stopped by a conditional operation in the auxiliary subroutine, which returns contol to the master routine when u exceeds a certain quantity (see note 4 to the specification of Q1 in Part II).

7-73 Constants: N sequence.

0N	P 200 F		starting value of $x \cdot 2^{-15}$
1	P 20 F		increment of $x \cdot 2^{-15}$
2	(P 200 F)		current value of $x \cdot 2^{-15}$
3	(P 900 F)		current value of $10X \cdot 2^{-15}$
4	P 10 F		negative increment of X
5	P 510 F		final value of X plus 1
6	P 5 F		
7	(P F)		column counter
8	K F		14.2^{-4}
9	P 19 F		

In addition, two long numbers are taken in by subroutine R5 and placed in 60D and 62D.

7-74 Master routine.

		G		K	
Start, 30 →	0	S	6	N	
	1	T	7	N	plant column counter
20 →	2	A	3	N	
	3	T	6	θ	plant current value of X
	4	A	4	θ	calls in Q1, which places integral in 0H.
	5	G	2	ϕ	
	6	(P		F)	current value of X
	7	P	5	F	strip width
	8	P		F	upper limit of integration
Q1 →	9	S		H	
	10	T		D	integral to 0D
	11	A	11	θ	calls in P11, which prints value of integral.
	12	G	4	ϕ	
P11 →	13	A	7	N	column count
	14	A	2	F	
	15	E	21	θ	jump after 5th column
	16	T	7	N	
	17	A	2	N	increase x
	18	A	1	N	
	19	T	2	N	
	20	E	2	θ	accumulator empty: jump to 2θ.
15 →	21	A	3	N	increase X
	22	S	4	N	
	23	U	3	N	
	24	S	5	N	
	25	E	27	θ	test for end
	26	Z		F	Stop

25 →	27	T		F	clear accumulator
	28	A		N	reset x
	29	T	2	N	
	30	E		θ	

7-75 Auxiliary subroutine.

		G		K	
	0	A	3	F	
	1	T	51	θ	plant link
	2	H	62	D	$(\pi/180)\cdot(2^9/10)$ to register
	3	V		D	$10\lambda\cdot(\pi/180)\cdot(2^9/10)$ to accumulator
	4	L	8	F	$\frac{1}{2}\lambda$ (radians) to 4D
	5	Y		F	
	6	T	4	D	
	7	A	7	θ	calls in T7, which places $\frac{1}{2}\sin\lambda$ in 4D
	8	G	5	φ	
T7 →	9	A	4	D	$\frac{1}{2}\sin\lambda$ to 2H
	10	T	2	H	
	11	H	62	D	$10X\cdot(\pi/180)\cdot(2^9/10)$ to accumulator
	12	V	3	N	
	13	L	8	F	
	14	Y		F	
	15	T	4	D	$\frac{1}{2}X$ (radians) to 4D
	16	A	16	θ	calls in T7, which places $\frac{1}{2}\sin X$ in 4D
	17	G	5	φ	
T7 →	18	A	4	D	$\frac{1}{2}\sin X$ to 4H
	19	T	4	H	
	20	A	2	H	$\frac{1}{2}\sin X - \frac{1}{2}\sin\lambda$ to 0D
	21	S	4	H	
	22	T		D	
	23	A	2	H	$\frac{1}{2}\sin\lambda$ to 4D
	24	T	4	D	
	25	A	25	θ	calls in D7, which places
	26	G	6	φ	$(0.5\sin X - 0.5\sin\lambda)/(0.5\sin X)$ in 0D
D7 →	27	A		N	$x\cdot 2^{-15}$ to accumulator
	28	L	8	F	
	29	T	4	D	
	30	H	4	D	$x\cdot 2^{-10}$
	31	N		D	$-x[(\sin X - \sin\lambda)/(\sin X)]2^{-10} = -2^{-6}u$
	32	L	16	F	
	33	U	6	H	-u to 6H
	34	A	8	N	$14\cdot 2^{-4} - u$ in accumulator
	35	G	52	θ	
	36	T		F	clear accumulator
	37	H	6	H	
	38	A	38	θ	calls in E3, which places $e^{-2^4 u}$ in 0D
	39	G	3	φ	
E3 →	40	A		D	$\frac{1}{4}e^{-2^4 u}$ in 0D
	41	R	1	F	
	42	T		D	

	43	H	2	H	$\frac{1}{4}\sin^2\lambda$
	44	V	2	H	
	45	L	1	F	$\sin^2\lambda$
	46	N	2	H	$\sin^2\lambda - \frac{1}{4}\sin^2\lambda$
	47	Y		F	
	48	T	4	D	$\frac{3}{4}\sin^2\lambda$ to 4D
	49	A	49	θ	calls in D7, which places $\frac{1}{3}e^{-2^4u}$ cosec^2 in 0D
	50	G	6	ϕ	
D7 →	51	(Z		F)	link (E m+2 F)
35 →	52	T		F	plants E m+21 F in 56 if $2^4u>14$
	53	A	51	θ	
	54	A	9	N	
	55	T	56	θ	
	56	(Z		F)	becomes E m+21 F. Returns control to master routine via link order of Q1 (see note 1).

Notes:

1. When 2^4u exceeds 14 the auxiliary subroutine causes control to return to the master routine via the link order of Q1. When assembly subroutine M1 is used this is more convenient than returning control directly, since it avoids the necessity for more than one entry point in the master routine.

2. It will be noted that in the auxiliary subroutine all the intermediate quantities are placed in separate storage locations. Some of these could be written over others, but by placing each in a separate location it is much easier to arrange for them to be printed out should this be desirable when looking for errors in the program.

7-76 Make-up of tape.

P K T 64 K

R5

R5 is used to take in two constants and is afterwards overwritten

T 62 D

07758 06398 +

Conversion factor (degrees to radians) $2^9\pi/1800$; goes into 62D.

500000 +

Round-off number $5\cdot10^{-6}$; goes into 60D (these numbers appear backwards on this tape; see specification of R5, Part II).

X T Z

M1

P 50 F
T 80 K

space

P Z G K
E 64 K T F

G 16 K	This causes spaces to be left in the store for 8 long numbers which can be referred to by the addresses 0H, 2H, ..., 14H (see note 5 in Section 4-62).
E 64 K T F	
N-sequence	
space	
P Z G K T 74 K Tϕ E 64 K I F	
Master routine	
space	
P Z G K E 64 K P F	
Auxiliary subroutine	
space	
P Z G K E 64 K P F T 46 K P 1 ϕ	N parameter for Q1; the H parameter has already been set by the assembly subroutine.
Q1	
space	
P Z G K E 64 K P F T 45 K P 4 F	Parameter for E3.
E3	
space	
P Z G K E 64 K P F T 45 K A 60 D P 25 F P 46 θ P 1024 F	Parameters for P11.
P11	
space	
P Z G K E 64 K P F	

T7

space

P Z G K
E 64 K P F

D7

E 25 K
E ϕ P F

This program contains a little over 300 orders and pseudo-orders; of these only 98 need to be drawn up specially.

7-8 Program to facilitate the solution of algebraic equations.

In Graeffe's method for the solution of the equation

$$G_0 = a_0 + a_1x + ... a_{s-1}x^{s-1} + a_sx^s = 0,$$

whose roots are λ_1, λ_2 .....λ_s, an equation $G_m = 0$ having roots λ_1^t, λ_2^t.... λ_3^t is formed ($t = 2^m$ where m is an integer). The program given below is designed to calculate the coefficients of $G_m = 0$. The subsequent numerical analysis necessary to find the roots must be performed independently and will proceed along lines which can only be determined after inspection of the coefficients of $G_m = 0$.

The program uses the floating decimal subroutines A1, A2, and A4 and involves four tapes.

1. Input program tape. This puts into the store a short program which causes the coefficients (punched on the following tape) to be read and placed in 300D, 302D, ... (300+2s)D. When all the orders on this tape have been read the machine is stopped by pressing the stop button, blank tape being under the reading head.

2. Coefficient tape. This tape is placed in the tape reader when the input program tape has been read, and the machine is restarted by the reset button. If desired it may be combined with the input program tape.

3. Master tape. This is taken in by pressing the start button again when the coefficients are in the store. It first causes the coefficients of $G_1 = 0$ to be computed and placed in (302+2s)D, (304+2s)D, ... (300+4s+2)d. The coefficients of $G_2 = 0$ are then computed and placed in the locations formerly occupied by those of $G_0 = 0$. This process continues, the two sets of locations being used alternately until the coefficients of $G_m = 0$ have been computed. The machine then stops and two sets of coefficients (those of $G_{m-1} = 0$ and $G_m = 0$) are available for printing out.

4. Printing-out tape. This is placed in the tape reader when the machine has stopped and the start button is pressed again. The coefficients of $G_{m-1} = 0$ and of $G_m = 0$ are printed out in a single column of 2s+2 numbers.

Notes:

1. The program is drawn up in such a way that it can be adjusted to any value of s by giving a suitable value to one of the parameters at the head of the input program tape.

2. If, after the results have been printed out, it is desired to carry the root-squaring process further, the master tape, modified by punching P K T 49 K P 2s F at its head, may be inserted a second time. The coefficients of $G_{2m-1} = 0$ and $G_{2m} = 0$ will then be computed and may be printed out by means of tape 4.

3. If sufficient storage space is available, it is possible to combine tapes 1, 3, and 4.

7-81 Make-up of input program tape.

```
P      F
T  50  K
G      K
T  49  K
P  2s  F          L parameter
T  45  K
P 300  L          equivalent to P 300+2s F; H parameter
P   2  L          equivalent to P 2s+2 F; N parameter
```

A1

7-82 Make-up of coefficient tape.

```
P      K  ]
E  51  K  ]       control combination initiating the program
Z      F  ]
```

a_s
a_{s-1}
a_{s-2}
a_{s-3}
.
.
.
.
a_1
a_0

7-83 Make-up of master tape. (P 2s F is in 49 when this tape is taken in.)

```
P      K
T 200  K
G      K
T  45  K
P 300  F          H parameter
P 302  L          N parameter
P      L          M parameter
P  50  F          Δ parameter
```

Auxiliary subroutine

```
T  50  K
```

A4

T 280 K

master routine

E	Z	control combination switching control to first
P	F	order of master routine.

7-84 Make-up of printing-out tape. (P 2s F is in 49 when this tape is taken in.)

	P		K	
	T	50	K	
	G		K	
→50	A	49	F	these orders are stored temporarily in 50 to 53 and place P 4s F in 49 (L parameter).
51	L		D	
52	T	49	F	
53	E	25	F	returns control to initial orders.
	E	50	K	switch combination
	P		F	
	T	45	K	
	P	302	L	H parameter
	P	4	L	N parameter
	P	1	F	M parameter
	P	8	F	Δ parameter
	P		F	L parameter

A2

E	1	Z	tape combination initiating printing
P		F	program.

Note: P 4s F will not be in 49 after this tape has been taken in.

7-85 Master routine.

	G		K	
0	S	16	θ	set count of squarings
14→ 1	T	33	F	
2	T	10	D	clear floating decimal accumulator
3	T	9	F	
4	A	4	θ	switch to auxiliary subroutine
5	G	200	F	
Aux.→ 6	A	266	F	interchange roles of P u D and P v D in locations 66θ and 67θ of auxiliary subroutine.
7	T	34	F	
8	A	267	F	
9	T	266	F	
10	A	34	F	
11	T	267	F	
12	A	33	F	test for s squarings
13	A	2	F	
14	G	1	θ	
15	Z		F	
16	‖P	6	F	squaring count

7-86 Description of the auxiliary subroutine. Given the coefficients $a_0^{(0)}, a_1^{(0)}, a_2^{(0)}, \ldots, a_s^{(0)}$, of an equation G_0, in locations (u+2r)D, r = 0, 1, ...s, the subroutine computes and places the coefficients $a_0^{(1)}, a_1^{(1)}, a_2^{(1)}, \ldots, a_s^{(1)}$ of the equation G_1, in locations (v+2r)D, r = 0, 1, ...s. The $a_n^{(1)}$ are defined as follows:*

$$a_n^{(1)} = \left[a_n^{(0)}\right]^2 - 2\sum_{r=1}^{j}(-1)^{r-1}a_{n+r}^{(0)}a_{n-r}^{(0)} \qquad \text{(i)}$$

where j = n if $n \leq s-n$
j = s-n if $n > s-n$.

7-87 The auxiliary subroutine. Closed; 80 storage locations; working positions 30, 31, 32. Preset parameters:

		G		K	
		T	45	K	
45	H	P	u	F	
46	N	P	v	F	
47	M	P	2s	F	s is the degree of the equation
48	Δ	P	w	F	address of first order of A4

		T		Z	
	0	A	3	F	] plants link
	1	T	65	θ	]
	2	S	71	θ	] set n=0 initially
64→	3	A	71	θ	]
	4	U	30	F	$n \cdot 2^{-14}$ to 30
	5	A	66	θ	]
	6	U	40	θ	]
	7	U	41	θ	] these orders plant P u+2n D in 40θ, 41θ,
	8	U	55	θ	] 55θ, and 56θ.
	9	T	56	θ	]
	10	A	30	F	]
	11	A	67	θ	] P v+2n D is formed and planted in 61θ.
	12	T	61	θ	]
	13	S	30	F	]
	14	S	30	F	] $(s-2n)2^{-14}$ to accumulator
	15	A	70	θ	]
	16	E	76	θ	jump if $n \leq s-n$
77→	17	A	30	F	]
	18	T	32	F	] P 2j F is planted in 32
	19	T	20	D	clear 20D

*See for example Whittaker and Robinson, pp. 106-109, 3rd edition (1940), Blackie. It is more convenient to use the relation (i) in the form

$$a_n^{(1)} = \left[a_n^{(0)}\right]^2 + 2(-1)^j p_j,$$

where p_j is the jth term of the sequence

$$p_r = a_{n+r}a_{n-r} - p_{r-1}, \quad p_1 = a_{n+1}a_{n-1}.$$

20	T	31	F	clear 31
21	E	47	θ	jump to 47θ
49→22	A	32	F	modify counter; $r \cdot 2^{-14}$ to 31
23	A	68	θ	
24	T	31	F	
25	A	40	θ	form and plant P u+2n+2r D
26	A	68	θ	
27	T	40	θ	
28	A	40	θ	form and plant P u+2n-2r D
29	S	68	θ	
30	T	40	θ	
31	S	69	θ	test m, the two-way switch
32	E	78	θ	jump if r is even
33	T	69	θ	restore two-way switch
34	A	2	F	
79→35	A	75	θ	plants P 73 θ in 58θ if r is even. Plants P 74 θ in 58θ if r is odd.
36	T	58	θ	
37	A	37	θ	enter A4
38	G		Δ	
39	E	3	Δ	$a_{n+r} \cdot a_{n-r}$ to floating decimal accumulator
40	(P		F)	
41	(P		F)	
42	E	3	Δ	
43	P	72	θ	$-p_{r-1}$
44	P	20	D	
45	E	13	Δ	p_r to 20D
46	P	20	D	
21→47	A	31	F	test for r = j
48	S	32	F	
49	G	22	θ	
50	A	68	θ	reset two-way switch
51	T	69	θ	
52	A	52	θ	enter A4
53	G		Δ	
54	E	3	Δ	form $\left[a_n^{(0)}\right]^2$ in floating decimal accumulator
55	(P		F)	
56	(P		F)	
57	E	3	Δ	$+2(-1)^j p_j$
58	(P	74	θ)	
59	P	20	D	
60	E	13	Δ	
61	(P		F)	plant $a_n^{(1)}$ in (v+2r)D
62	A	30	F	test for last coefficient
63	S	70	θ	
64	G	3	θ	
65	(E		F)	link
66	P		πH	becomes P u D
67	P		πN	becomes P v D
68	P	2	F	
69	(P	2	F)	two-way switch

70	P		M	becomes P 2s F
71	P	2	M	becomes P 2s+2 F
72	V	2040	F	≡-1 in floating decimal representation
73	P	16	F	≡+2
74	V	2032	F	≡-2
75	P	73	θ	
16→76	T		F	control switched to these orders if n≤s-n
77	E	17	θ	
32→78	T	69	θ	control switched to these orders if r is even
79	E	35	θ	control being returned to 35θ so that P 73 θ is planted in 58θ. This corresponds to taking $2(-1)^r = +2$. When the last product is formed r = j and $2(-1)^j$ is in location 73θ if j is even, and 74θ if j is odd.

PART II

SPECIFICATIONS OF LIBRARY SUBROUTINES

Each subroutine is distinguished by a letter denoting its category and a serial number within that category. The categories are as follows.

Category	Subject
A	Floating point arithmetic.
B	Arithmetical operations on complex numbers.
C	Checking.
D	Division.
E	Exponentials.
F	General routines relating to functions.
G	Differential equations.
J	Special functions.
K	Power series.
L	Logarithms.
M	Miscellaneous.
P	Print and layout.
Q	Quadrature.
R	Read (i.e., Input).
S	nth root.
T	Trigonometrical functions.
U	Counting operations.
V	Vectors and matrices.

In the specifications on succeeding pages the following information is given in abbreviated form immediately beneath the title of each subroutine:

1. Type of subroutine, i.e., whether open, closed, interpretive, or special.
2. Restriction on address of first order. If the word "even" appears it denotes that the first order must have an even address; if no note appears it indicates that the address may be either odd or even.
3. Total number of storage locations occupied by the subroutine.
4. Addresses of any storage locations needed as working space by the subroutine.
5. Approximate operating time (not possible to state in all cases).

The gaps in the numbering within each category correspond to subroutines which have become obsolete.

A. Subroutines to carry out floating point arithmetic.

A1 Input of a sequence of s real numbers in floating decimal form (used with A3 or A4).
Closed; even; 77 storage locations; working positions 0D, 4D, 6D, and 10D.

Given a sequence of s numbers (>1) punched in floating decimal form, this subroutine assembles them in a standard form* and places them in nD, (n+2)D, (n+4)D ..... (n+2s-2)D.

Preset parameters:					
45	H	P	n+2s-2	F	
46	N	P	2s	F	

Notes: *1. For further details see Part I, Section 4-82.
2. A typical number $X \cdot 10^p$, where $256>p>0$ and $4>|X|\geq 0.4$, is punched as: X, sign, p, F. In X the decimal point is immediately after the first digit punched. Any number of digits up to ten may be punched for X. More than ten will exceed the capacity of the accumulator.
3. The numerical part of each number is eventually rounded off to 24 binary places.

A2 Print sequence of s floating decimal real numbers in a preset layout (used with A3 or A4).
Closed; even; 111 storage locations; working positions 0D, 4D, 8.

Prints the sequence of s numbers packed in floating decimal form in storage locations nD, (n+2)D, (n+4)D .... (n+2s-2)D. Layout: numerical part printed to d digits preceded by sign and followed, after one space, by the positive* integral exponent (up to five figures with suppression of nonsignificant zeros). Two spaces separate columns of complete numbers. Decimal point in numerical part is after first digit printed.

Preset parameters:						
45	H	P	n+2s-2	F		
46	N	P	2s	F		
47	M	P	c	F	number of columns (≤ 4)	
48	Δ	P	d	F	number of digits	
49	L	P	x	F/D	$= q \cdot 2^{-16}$	

Notes: 1. Teleprinter must be on figure shift.
*2. A2 prints out positive exponents only. To ensure this provision is made whereby all exponents may be increased by a preset amount, q, before printing.
3. No round-off is provided. Any number of figures may be printed.
4. $c(d+9) \leq 72$.

A3 Special arithmetical operations on real numbers in floating decimal form.
Closed; even; 126 storage locations; working positions 0D, 4D, 6D, 8D, 10D, 12D; time: part 1 = 85 msecs; part 2 = (64+24q) **msecs.

Enables special arithmetical operations to be carried out on real numbers expressed in standard floating decimal form.* A3 is in two parts and has two entry points p and p+76, where p is the location of the first order. Part 1 is entered by:

```
m   | A m F
m+1 | G p F
```

and executes the arithmetical operation: xy + C(A) to A, where A refers to a floating decimal "accumulator" in the store and x and y are stored in rD and sD respectively.

Part 2 is entered by:

```
m   | A  m   F
m+1 | G p+76 F
```

and assembles C(A) in standard form* which is then placed in tD. r, s, and t are specified by parameters P r F/D, P s F/D, and P t F/D which may refer to either long or short floating numbers. These parameters are stored in preset locations.

Preset parameters:

```
45 | H | P a F   address of P r F/D
46 | N | P b F   address of P s F/D
47 | M | P c F   address of P t F/D
```

Notes: *1. See Part I, Section 4-82.

2. No more than two "Part 1" operations may be carried out in succession without following a "Part 2" operation.

**3. q is the number of significant zeros arising from cancelation in the sum in the accumulator.

4. See Part III for detailed program.

A4 Special arithmetical operations on real numbers in floating decimal form. Interpretive version of A3.

Interpretive; even; 150 storage locations; working positions 0D, 4D, 6D, 8D, 10D, 12D; time: part 1 = 100 msecs; part 2 = (80+24q)msecs.

This subroutine consists of A3 preceded by a supplementary subroutine which enables it to be used with program parameters. Floating point operations using this subroutine can then be coded as follows:

```
m   | A m F
m+1 | G p F    calls in A4.
```

Thereafter, when required, control may be switched to Part 1 (see A3) by

```
m+2 | E p+3 F
m+3 | P r F/D*
m+4 | P s F/D*
```

Control will afterwards be returned to m+5, whence a further Part 1 operation can be called in by another triplet of orders similar to those above or, alternatively, a Part 2 operation may be initiated by

```
m+5 | E p+13 F
m+6 | P t  F/D*
```

Control will then be returned to m+7.

*P r F/D, etc., refer to either short or long floating numbers.

A5 Special arithmetical operations on complex numbers in floating decimal form.
Closed; even; 206 storage locations; working positions 0D, 4D - 18D; time: Part 1 = 150 msecs; Part 2 = (90+31q)* msecs.

Similar to A3 but operates on complex numbers expressed in standard form (see Part I, Section 4-83). Part 1 carries out the operation

$$z_1 z_2 + C(A) \text{ to } A,$$

where z_1 is stored in rD and (r+2)D and z_2 in sD and (s+2)D. Part 2 assembles C(A) in standard form and transfers it to tD and (t+2)D.

Preset parameters:				
	45	H	P a F	address of P r F
	46	N	P b F	address of P s F
	47	M	P c F	address of P t F

Notes: 1. Not more than two Part 1 operations may be carried out without following with a Part 2 operation.
*2. q is the number of multiplications by 10 which are necessary when assembling the number to bring the modulus within the standard range.
3. A5 is entered in the same way as A3, Part 2 being entered at p+128.
4. See Part III for detailed program.

A6 Input of a sequence of complex numbers in floating decimal form (used with A5 or A8).
Closed; even; 98 storage locations; working positions 0D, 4D, 6D, 14D, 16D, 18D.

Given a sequence of s complex numbers (>1) punched in floating decimal form, A6 assembles them in standard form* and places them in nD, (n+4)D, (n+8)D, ... (n+4s-4)D.

Preset parameters:			
	45	H	P n+4s-4 F
	46	N	P 4s F

Notes: *1. See Part I, Section 4-83.
2. A typical number $(X + iY) \cdot 10^{p}$, where $4 > \left|\sqrt{X_0^2 + Y_0^2}\right| \geq 0.4$ and $2047 \geq p \geq 0$, is punched as X_0, sign; Y_0, sign; p. In X_0 and Y_0 the decimal point is immediately after the first digit punched. Any number of digits up to ten may be punched for X_0 and Y_0; more than ten will exceed the capacity of the accumulator.
3. X_0 and Y_0 are eventually rounded off to 28 binary places.

A7 Print sequence of s floating decimal complex numbers (used with A5 or A8).
Closed; even; 125 storage locations; working positions 0D and 4D.

Prints the sequence of s complex numbers packed in floating decimal form in storage locations nD, (n+4)D, (n+8)D, ... (n+4s-4)D. Layout: 2 columns of complete numbers. Each number consists of ± real part, space, ± imaginary part, space, positive* exponent.

Preset parameters:				
	45	H	P n+4s-4 F	address of last number of sequence
	46	N	P 4s F	s numbers
	47	M	P x F/D	$= q \cdot 2^{-16}$
	48	Δ	P d F	number of digits.

Notes: 1. Figure shift is called during input.
*2. As in A2, provision is made for the addition of a preset amount q to the exponent of each number before printing.
3. No round-off is provided. Any number of digits may be printed.
4. $c(d+9) \leq 72$.

A8 Special arithmetical operations on complex numbers in floating decimal form. Interpretive version of A5.
Interpretive; even; 230 storage locations; working positions 0D, 4D to 18D; Time: Part 1 = 165 msecs, Part 2 = 105 + 31q msecs.

This subroutine consists of A5 preceded by a supplementary subroutine which enables it to be used with program parameters. Floating point operations using this subroutine can then be coded as follows:

m	A m F	
m+1	G p F	calls in A8

Thereafter, when required, control may be switched to Part 1 (see A5) by

m+2	E p+3 F
m+3	P r F/D*
m+4	P s F/D*

Control will afterwards be returned to m+5, whence a further Part 1 operation can be called in by another triplet of orders similar to those above; alternatively, a Part 2 operation may be initiated by

m+5	E p+13 F
m+6	P t F/D*

*P r F/D, etc., may refer to either short or long floating numbers.

A9 Input of a sequence of numbers in floating decimal form during input of orders (used with A11).
Special; even; 31 storage locations.

The numbers are punched on a separate data tape in the following form: character representing exponent; sign; numerical part (the decimal point being after the first digit). For example,

512 would be punched as W + 512 or 2 + 512 = $10^2(5.12)$,
-.0012 " " " " B - 12 = $10^{-3}(1.2)$.

The first number in the sequence is preceded by Z T X. After the subroutine, T m D is punched, followed by the data tape which is copied in the reverse direction. The numbers are then placed in the store in floating decimal form in storage locations mD, (m-2)D, etc., so that mD is the location of the number originally punched last.

Accuracy: the numerical part of each number is represented by 23 binary digits - equivalent to almost 7 decimal digits.

Notes: R9 must be in the store when A9 is read.

A10 Print single floating decimal number (used with A11).
Closed; even; 63 storage locations; working position, 4D; time = 2 secs per number.

Prints the signed exponent followed by the signed numerical part of the number stored in floating decimal form in 0D. Each number is printed as: negative sign, or space; exponent (2 figures); 2 spaces; negative sign, or space; integer part (1 figure); space; 6 decimal figures.

Accuracy: the number is rounded off to 7 figures (including integral part).

Notes: 1. If the numerical part is exactly 10 it will be printed as # 000000.

2. The last order on the tape is the digit layout parameter for the numerical part (as in P11), and may be altered if required.

3. Normally, before the number is printed, carriage return and line feed will occur. They may be omitted by entering the routine at its third order. One space only is printed after each number. Not more than four numbers may be printed on one line.

A11 Arithmetical operations on real numbers expressed in floating decimal form.
Interpretive; even; 128 storage locations; working positions 0D*, and H and N for floating accumulator; time, see Note 1.

Operations: A11 carries out the operations specified individually by the program parameters according to the following code:

Program parameter	Operation
A m F/D*	add to the number in floating decimal accumulator the number represented by C(m).
B m F/D*	subtract from the number in floating decimal accumulator the number represented by C(m).
V m F/D*	multiply the number in floating decimal accumulator by the number represented by C(m).
T m F/D	transfer the number in floating decimal accumulator to S(m), and clear the accumulator.
E m F	switch control to m with accumulator clear (previous parameter need not be T).

*$m \leq 511$

Representation of numbers. Each number is expressed in the form $a \cdot 10^p$, where a is the numerical part and p the exponent (an integer). In the store the number is represented by the long or short number $a \cdot 2^{-11} + p \cdot 2^{-6}$. The routine uses positions H (long) and N (short) as a "floating decimal accumulator," or "f.d.a.," in which the above numbers would appear as $-a \cdot 2^{-11}$ in H and $p \cdot 2^{-14}$ in N.

Range of values: In the f.d.a. $|p| < 16000$ approx. and $-2048 \leq a < 2048$. In the store $-63 \leq p < 63$ and $|a| \leq 10$, but when a number is transferred to the store from the f.d.a. it is always represented in such a way that either $1 < |a| \leq 10$, or $a = 0$ and $p = -63$.

Capacity of registers. If a T parameter is read, and the number in the f.d.a. exceeds 10^{63}, the machine will normally come to a dynamic stop. If this is undesirable, the preset L parameter E 56 *0* may be replaced by any

E order transferring control to a suitable point in the store in the event of exceeded capacity (the accumulator not being empty).

If the number in the f.d.a. is less than or equal to 10^{-63}, a T parameter will place the representation of zero in the store (0.10^{-63}).

It is possible to exceed the capacity of the numerical part of the f.d.a. without the number actually represented by the f.d.a. exceeding the range of possible values. The rules for avoiding this are as follows. After a T parameter, $|a| = 0$; an A or B parameter may increase $|a|$ by 10, and a V parameter may multiply $|a|$ by 10, in the worst cases; hence the sequence of parameters should be such as to ensure that a can never reach 2048.

Accuracy: when using long numbers, a has between 23 and 27 binary digits, that is, 7 or 8 decimal digits. When using short numbers, a has between 5 and 9 binary digits, that is, about 2 decimal digits.

Notes: 1. Times of operation:

Parameter	Time in secs
A	.066
B	.066
T	.050 + m(.015), where m is the number of decimal shifts necessary to convert the number to the form required in the store.
V	.050
E	.012

2. 0D may be used as a temporary store only if no A, B, or V parameters are used between planting and using; for example,

```
T D          T  D
A D   or     T 12 D
             B  D     is permissible,
```

but not

```
T  D
A 4 D
B  D
```

Preset parameters:

```
H  P  s  D
N  P  t  F
M  P 103 θ  ┐
Δ  P     θ  ├  set and used by subroutine
L  E  56 θ  ┘
```

location of numerical part of f.d.a.

location of exponent of f.d.a.

B. Subroutines to carry out arithmetical operations on complex numbers.

B1 Complex Operation No. 1.
Interpretive; 16 storage locations; working positions hD, (h+2)D; time = 21 msec per order.

This subroutine is entered in the usual manner and executes the orders following the entry, operating on long or short complex numbers. Each order operates on the complex number C(n) + iC(n+2), where n is the address specified

in the order. This subroutine enables operations to be carried out on complex numbers by the following orders: A, S, T, U, L, R. Multiplication by a real constant (placed in the multiplier register before the subroutine is entered) can also be used.

Preset parameter: 45 | H | P h D (h even)

Notes: 1. Shifting, up to 6 places must be effected by a series of single or double shifts (i.e., L D or L 1 F). A shift of 7 places or more may be obtained by a pair of orders, e.g., L 2^{n-5} F, L(2^{n-5} - 2) F.

2. Y order is meaningless.

3. T D, U D must not be used, since they will destroy the contents of 2D.

4. Exit from the subroutine is made by an E order immediately following a T order. Control is transferred to the address specified in the E order.

5. Care should be taken that the first time B1 is called in storage locations hD and (h+2)D are cleared.

B2 Complex Operation No. 2.
Interpretive; even; 54 storage locations; working positions 0 and hD to (h+10)D; time = approx. 75 msec per order.

Similar to B1 but makes available the following orders: A, S, T, U, V, N, Y, and right or left shift of one or two places.

Preset parameters: 45 | H | P h D (h even)
46 | N is also used by subroutine

Notes: 1. There is no H order. The role of "complex multiplier register" is undertaken by storage locations hD and (h+2)D, which may be filled by the order T (or U) hD.

2. See also notes 1, 3, 4, and 5 for subroutine B1.

3. See Part III for detailed program.

C. Checking subroutines.

C1 Cycle check, examines one storage location.
Special; even; 44 storage locations; time = about 2.5 secs per number.

May be applied to a program in order to print C(hD) immediately before obeying the order in n. Numbers are printed in a single column; printing ordered by the master routine occurs to the right of the previous check number. The accumulator must be empty before C(n) is obeyed. C(n) must not be altered by the original program or used in any way other than as an order.

C7 Check function letters, with localized print suppression.
Special; 61 storage locations; time, see Note 5.

Performs a given program order by order, and prints the function letters of those orders which are drawn from certain specified parts of the store; other orders are obeyed silently. The store may be divided into four regions, orders in two of which have their function letters printed.

Preset parameters:							
	45	H	P	b	F		
	46	N	P	(c-a)	F		See Note 1
	47	M	P	(c-b)	F		
	48	Δ	P		θ	print low	See Note 1.
		or	Δ		θ	print high	
	49	L	P	m	F	start at m	

Notes: 1. The regions of the store are specified by the parameters a, b, c as follows:

(i) $n<a$
(ii) $a\leq n<b$
(iii) $b\leq n<c$
(iv) $c\leq n$

The subroutine will either "print low," i.e., print function letters of orders in (i) and (iii), or "print high," i.e., print function letters of orders in (ii) and (iv).

2. Print routines in the original program must be arranged to lie in regions from which the function letters are not printed. Characters printed by such routines will appear as figures.

3. A new line of printing is begun at each transfer of control; a clear line is left where orders have been obeyed silently unless such orders themselves cause printing to appear on this line.

4. C7 only tests the locations of orders at each transfer of control, so that if control enters a new region during a consecutive sequence, the mode of operation does not change immediately.

5. Speed of operation is about 5 orders per second when printing function letters, 30 orders per second when suppressed.

6. C7 must be placed at the end of the orders on the tape. After being read it will direct control to itself and commence checking at order m.

7. See Part III for detailed program.

C8 Numerical check, with delayed start and printing from a restricted region of the store.
Special; *even; 41 + 32 storage locations; time = 200 msec per digit.

May be applied to a program in order to print C(Acc) before obeying T orders specifying addresses less than a certain number. The main program is obeyed at full speed until it encounters the order in m when checking commences. The value of m must be chosen so that C(Acc) = 0 before the order in m is obeyed. A new line of printing is started at each transfer of control. In general C8 should be used with programs in which all printing is done by a subroutine; printing is suppressed, using a dummy print routine, and part of C8 is written over the printing subroutine.

Preset parameters:				
	45	H	P h F	location of print subroutine
	46	N	P n F	location of store to hold part of C8 (n even)
	47	M	P m F	start address
	48	Δ	P d F	number of digits
	49	L	P s F	division of store.

Notes: *1. C8 has two parts, one of which, 41 orders long, is written over an existing print routine. The location of the first order of the other part (32 orders long) must be even.

2. C8 is to be copied onto the tape after the program to be checked. T 45 K is punched, followed by the parameters and C8, the orders of which are then placed in the specified positions. The order directing control to the main program must be placed immediately after C8.

3. C8 prints C(Acc) only if the address specified in the T order is less than s.

C9 Check function letters with delayed start and suppression of check during closed subroutines.
Special; 48 storage locations; time = about 1/4 sec per order printed.

Similar to C11 but has a delayed start and will cease checking during each closed subroutine. Cannot be applied to a program which contains a subroutine with more than one program parameter.

C10 Numerical check with delayed start and suppression of check during closed subroutines.
Special; even; 37 + 51 storage locations; time = 1/5 sec per digit printed.

May be applied to a routine in order to print C(Acc) before obeying T orders. It has a delayed start and will cease checking during each closed subroutine. It may be used only on programs containing subroutines with at most one program parameter. If the program has the order A n F in S(n) for a purpose other than entry to a closed subroutine, C10 will fail at that point.

Preset parameters:				
	45	H	P h F	see Note 1
	46	N	P n F	number of digits to be printed.
	47	M	P m F	address of order at which checking starts.

Notes: 1. Part of the subroutine, 51 orders long, is placed in locations h to (h+50) and may be written over a print routine in the master routine in which case printing from the master routine will be suppressed.

2. A new line of printing is started at each transfer of control.

3. A line feed occurs when a closed subroutine is encountered.

4. The address m of the order at which checking starts must be chosen as described in Note 2 of C5.

5. The first number printed by C10 is the numerical representation of the order at which checking starts.

6. C10 must be placed at the end of the tape and followed by E p K P F, directing control to the master routine.

7. A T order immediately following a closed subroutine with no program parameters will not cause C(Acc) to be printed.

8. See Part III for detailed program.

C11 Check function letters.
Special; 32 storage locations; time = about 1/5 sec per order.

Obeys a given program order by order and prints the function letter of each order. Cannot in general be used on a routine where printing occurs.

Notes: 1. A new line of printing is begun at each transfer of control.

2. If used on a print routine employing an F order C11 will behave as though the F order always reads from the teleprinter the symbol F. If the original program calls for a figure shift, the following printing will appear as figures. Otherwise symbols printed by the original program will appear as letters following the "O" which indicates the print order.

3. C11 should be placed at the end of the orders on the tape, (the final E m K P F being deleted) and followed by E m F. C11 will then begin to obey the original program at the order in location m.

C12 Check function letters with dummy print routine and delayed start.
Special; 40* storage locations.

The original program operates at full speed until the order h is obeyed for the nth time. Checking then commences, and function letters are printed. h must be so chosen that the accumulator is always empty before C(h) is obeyed and C(h) is not altered by the original program or taken into the arithmetical unit before checking starts. There is a dummy print routine at the head of C12 which returns control to the master routine without printing.

Notes: 1. C12 must not be fed into the machine before C(h).

2. Figure shift must not be called for by the original program, except in a print routine which is overwritten by C12. Print routines calling for figure shift during input must not be placed after C12 on the tape.

3. The dummy print routine destroys C(0). When checking, it appears as A T E
E

*4. If the original print routine had a normal program parameter, insert A 2 F in front of G K T 45 K. The whole routine now occupies 41 positions, and the dummy print routine appears when checking as A A T E
E

For two or more program parameters, insert A 2 F the corresponding number of times.

5. The program is started in the usual way by E m K P F.

Preset parameters:				
	45	H	P h F	position at which checking starts
	46	N	P n F	delay

D. Division subroutines.

D3 Division, small.
Closed; 20 storage locations; time = (22 m + 105) msecs, where $2^{-m-1} \leq |C(4D)| \leq 2^{-m}$.

Forms C(0D)/C(4D) and places the result in 0D. Slower than D7 but occupies less space.

Repetitive process:

$a_{n+1} = -a_n c_n + a_n$ $\qquad$ a_0 = dividend

$c_{n+1} = -c_n^2$ $\qquad$ $c_0 + 1$ = divisor

stop when $c_n = 0$.

D4 Division, small, positive divisor.
Open; 11 storage locations, working position 0D; time = (22m+108) msecs, where $2^{-m-1} \leq |C(Acc)| \leq 2^{-m}$.

Forms C(hD)/C(Acc), where $1 > C(Acc) > 0$, and places result in hD. A special case of D3.

Preset parameter: 45 | H | P h D

Notes: 1. The number of significant figures in the quotient is one less than the smaller of the numbers of significant figures in C(hD), C(Acc).
2. The left-hand half of the accumulator is clear at the end of the process, but the right-hand half is not. $0 < C(Acc) < 2^{-34}$.
3. See Part III for detailed program.

D6 Division, accurate, fast.
Closed; 36 storage locations; working positions 6D and 8D; time = (10m+120)msecs, where $2^{-m-1} \leq |C(4D)| < 2^{-m}$.

Forms C(0D)/C(4D) where $C(4D) \neq 0$ and $\neq -1$, and places result in 0D.

Accuracy: maximum error $\pm K \cdot 2^{-35} \pm 2^{-34}$, where K = quotient.

Note: See Part III for detailed program.

D7 Division, rapid.
Closed; 26 storage locations; time = (12m+105) msecs, where $2^{-m-1} \leq |C(4D)| < 2^{-m}$.

Forms C(0D)/C(4D) and places the result in 0D. Uses same repetitive process as D3.

Notes: 1. The right-hand side of the accumulator is not cleared at the end of the operation, i.e., $0 \leq C(Acc) < 2^{-34}$.
2. At the end of the process $-2^{-34} \geq C(4D) > -2^{-17}$.

E. Exponential subroutines.

E2 Exponential, slow.
Closed; 19 storage locations; working positions 0D and 6D; time = 930 msecs.

Forms exp [C(4D)] - 1 and places the result in 4D. $-1 \leq C(4D) < .693$.

Accuracy: probable error = 2^{-33}.

Note: See Part III for detailed program.

E3 Exponential, large range.
Closed; even; 56 storage locations; working position 4; time = (244+19p) msecs.

Forms exp $(2^p y)$, where $y = C(R) \leq 0$ and $p \geq 1$. Places result in 0D.

Preset parameter: 45 | H | P p F

Accuracy: the greatest error occurs when exp $(2^p y)$ is nearly equal to unity; the error is then less than $(2^{p-1} + 1) \cdot 2^{-34}$. The error diminishes rapidly as exp $(2^p y)$ decreases. For small values of exp $(2^p y)$ the error is less than $3 \cdot 2^{-35}$.

E4 Exponential, fast (used with R9).
Closed; even; 36 storage locations; time = 120 msecs.

Forms exp (x) where x = C(R) and $-1 \leq x \leq 0$. Places result in 0D. R9 must be in the store while E4 is being read.

Accuracy: maximum error is $2^{-34} + 2^{-35} \sum_{r=0}^{7} |x|^r$.

Note: See Part III for detailed program.

F. General routines relating to functions.

F1 Interpolation in a table of long numbers.
Closed; even; (68 + 2b) storage locations; working positions 0D, 10D, 12D, ..., (8 + 2b)D; time = $(60+24b^2)$ msecs.

Given in consecutive long storage locations, a table of a function at unit intervals of the argument, with the entry corresponding to zero argument specified by a program parameter, the subroutine calculates the value of the function for argument 2 C(4D) and places it in 10D. The number of entries over which the interpolation is made and their positioning with respect to the argument may be specified.

Preset parameters:	45	H	P 2b	D ⎤
	46	N	P a	F ⎦ see Note 1.
	47	M	P 2^{n-2}	F
	48	Δ	⎤	
	49	L	⎥	are also used by the subroutine
	50	X	⎦	
	p		A p F ⎤	
	p+1		G s F ⎦	Orders calling in F1
Program parameter	. . .p+2		A mD	where f(0) = C(mD)

Accuracy: Maximum error is $\pm 2^{-36} b(b+1)$ with probability $2^{-b(b+1)/2}$. R.M.S. value of error is $2^{-35}b$.

Notes: 1. The interpolation uses b values of the function of which there are a with arguments less than the integral part of the argument. $b \leq 11$.
a = 0, b = 2, corresponds to linear interpolation.
2. See Part III for detailed program.

F2 Solution of f(x) = 0, or inverse interpolation (second order process).
Closed; 58 storage locations; working positions 4D, (h+4)D, (h+6)D, (h+8)D.

Places a solution of f(x) = 0 in hD. f(x) must be defined by an auxiliary subroutine. Two trial values, x_1 and x_2, must be placed in hD and (h+2)D before F2 is called in. They must be such that $f(x_1)$ and $f(x_2)$ have opposite signs. The solution will lie between x_1 and x_2.

Preset parameters:	45	H	P h D
	46	N	P n F
	47	M	is also used by the subroutine.

Notes: 1. The auxiliary subroutine must be of the normal closed type commencing at n. It should place f [C(hD)] in 0D, leaving C(hD) unaltered.

2. If $f(x_1)$ and $f(x_2)$ have the same sign, F2 will place x_2 in hD and the accumulator will contain 2^{-35}.

3. 4D may be used by the auxiliary subroutine, but not (h+2)D to (h+8)D.

4. For inverse interpolation F1 can be used by the auxiliary.

5. If x is not required to an accuracy better than $\pm 2^{m-33}$, where $m \leq 10$, order 52 of F2 may be replaced by R 2^m F. This will save time, but the R.H. half of the accumulator may not be empty on exit.

6. See Part III for detailed program.

F3 Differencing and checking subroutine No. 1.
Closed; 26 storage locations + 12 for numbers; working position 4D; time = 36 msecs.

This subroutine calculates the first, second, third, and fourth differences of successive values of a function and checks that the magnitude of the fourth difference does not exceed a specified quantity. The current values of the function and the first 4 backward differences are held in mD, (m+2)D, (m+4)D, (m+6)D; each time the subroutine is called in C(0D) is taken as the new current value of the function and the differences are all advanced one step. If the fourth differences exceed C(m+10)D in magnitude a "?" is printed.

Preset parameter: 45 | H | P m D
or P m F

Notes: 1. The subroutine may be used to handle short numbers by punching P m F instead of P m D for the preset parameter. Differences of successive values of the function, the current value of which is placed in 0D, will then be computed and placed in (m+2), (m+4), etc.; a "?" will be printed if the fourth differences exceed C(m+10) in magnitude.

2. See Part III for detailed program.

F4 Differencing and checking subroutine No. 2.
Closed; 32 storage locations + 16 for numbers; working position 4D; time 42 units.

Similar to F3 but calculates the first, second, third, fourth, fifth, and sixth differences of successive values of a function and checks that the magnitude of the sixth difference does not exceed a specified quantity.

Preset parameter: 45 | H | P m D address of current value of function.

F5 The minimization of a positive function of n variables using a digital process.
Closed; 44 storage locations; working space, 0D, 0M, 2M; time = approx. 500n+50n·t msecs, where t = time for auxiliary subroutine.

Given a set of n variables in storage locations aD, (a+2)D, (a+4)D, ..., (a+2n-2)D the subroutine will continually adjust these variables, by the method described below, to seek a minimum of a positive function f of these variables. At each stage f is calculated and placed in 0D by an auxiliary closed subroutine whose first order is in 0Δ.

Method: 1. The first variable is successively decreased by an amount h until the calculated values of f begin to increase.

2. The smallest value of f is chosen and the process 1 repeated for each of the other variables in turn.

3. The processes 1 and 2 are repeated, using -h instead of h.

4. The processes 1, 2, and 3 are repeated, using -(-h)/8 as decrement.

5. When the decrement $h/8^m$ becomes less than 2^{-34}, the process terminates.

Accuracy: depends on the function and the starting value. For "well-conditioned" functions the subroutine will find a value such that if any one of the variables is adjusted by 2^{-34} the value of f will not decrease.

Note: 1. For maximum accuracy, h should be chosen so that the final decrement is -2^{-34}, that is, h should be of the form 2^{-1-3p}. The library tape uses $h = 2^{-4}$, specified by -h = V F, as the last order of the subroutine. If an approximate solution is known and used as starting values, h should preferably be of the order of the initial errors.

Preset parameters:						
45	H	P	a+2n-2	D	last variable
46	N	P	2n	F	number of variables
47	M	P	s	D	working space
48	Δ	P	t	F	location of first order of auxiliary subroutine

G. Subroutines for integration of ordinary differential equations

G1 Simultaneous first-order differential equations by modified Runge-Kutta process; single step, long numbers
Closed; even; 66 storage locations; working positions 0D, 4D, and 6D; time 0.21n seconds per step + time of auxiliary.

Each time this subroutine is called in it will advance the values of the variables by one step. It requires an auxiliary closed subroutine to calculate the first derivatives y' of all the variables from given values of the variables y. For detailed description of the process, see Part I, Sections 4-71, 4-72, and 4-73.

Preset parameters:					
45	H	P	a	D	
46	N	P	2n	F	
47	M	P	b-a	F	(if a<b)
		or			
		V(2048 - a+b)F			(if a>b)
48	Δ	P	c-b	F	(if b<c)
		or			
		V(2048 - b+c)F			(if b>c)
49	L	P	2^{m-2}	F	
50	X	P	d	F	

Accuracy: the truncation error in one step is of the order h^5. For a small set of well-behaved equations its magnitude is roughly $10^{-2}h^5$; for large sets or difficult equations it may be greater. Rounding-off errors accumulate at a rate corresponding to the keeping of (34+m) binary digits.

Notes: 1. The variables y are stored in n consecutive long storage locations, the last of which is aD.

2. The auxiliary subroutine is of the normal closed type and commences at d; it should place the quantities $2^m hy'$ in n consecutive long storage locations, the last of which is bD.

3. A further n long storage locations, the last of which is cD, are required by G1 to hold the quantities $2^m q$. At the beginning of a range these must be cleared.

4. If the independent variable is required it may be obtained by including an extra variable with the corresponding value of $2^m hy' = 2^m h$. The latter quantity may be set once and for all at the beginning of the range; it will not be disturbed.

5. m should be chosen so that the largest $2^m hy'$ is just within the capacity of the accumulator.

6. G1 uses 0D, 4D, and 6D, but these may also be used by the auxiliary subroutine.

7. See Part III for detailed program.

G2 Simultaneous first-order differential equations by Runge-Kutta process; single step, short numbers

Closed; even; 68 storage locations; working positions 0, 1, 4; time = 0·21n seconds per step + time of auxiliary.

Similar to G1, but works with short numbers.

Preset parameters:

45	H	P	a	F] note difference from G1
46	N	P	n	F]
47	M	P	b-a	F
	or	V	2048-a+b	F
48	Δ	P	c-b	F
	or	V	2048-b+c	F
49	L	P	2^{m-2}	F
50	X	P	d	F

Accuracy: The truncation error in one step is of the order h^5. For a small set of well-behaved equations its magnitude is roughly $10^{-2} h^5$; for large sets or difficult equations it may be greater. Rounding-off errors accumulate at a rate corresponding to the keeping of (16+m) binary digits.

G3 Integration of $y'' = f(x,y)$ by fifth-order process.

Closed; 45 storage locations; working positions 0D, 4D, 10D.

Each time this subroutine is called in it advances the integration by one step. A separate subroutine is needed to calculate f(x,y).

Preset parameters:

45	H	P	n	D	(n must be even)
46	N	P	m	F	

Accuracy: truncation error = $(y_0^{IV} - y_n^{IV})\, h^4/240$ over range y_0 to y_n.

Notes: 1. Apart from this subroutine, 11 long storage locations must be provided, beginning with nD.

2. An ordinary closed subroutine, starting in m must calculate f[C(n+10)D, C(n+18)D] and place it in (n+20)D.

3. The initial values must be placed in the following long storage locations at the beginning of the integration:

H	y_1	2H	$\delta y_{1/2}$	4H	y_1''	6H	$\delta y_{1/2}''$
10H	x_1	12H	h(integration step)			14H	h^2
		16H	1/12 = 0.083.				

4. After using the subroutine, corresponding values of x and y will be found in (n+10)D and (n+18)D respectively (i.e., in 10H and 18H).

5. Storage locations 0D, 4D, 10D are used. The auxiliary subroutine can use 0D and 4D but not 10D.

6. The time taken depends on the number of iterations necessary. It equals p(52 + auxiliary time) + 15 msecs, where p is the number of iterations, usually about 3.

7. See Part III for detailed program.

G4 Integration of $y'' = F(x,y)$ by sixth-order process
Closed; 47 storage locations.

Similar to G3 but uses a sixth-order process.

$$y_2 = y_1 + \delta y_{1/2} + (y_2'' + \delta^2 y_1''/12)h^2$$

$$y_2'' = f(x_2, y_2 - h^2\delta^2 y_1''/240)$$

J. Subroutines for calculating special functions

J1 Calculation of Legendre polynomials
Closed; even; 36+2q storage locations; working positions 0D, (4+2q)D, (6+2q)D; time = 52(q-1) msecs.

Calculates $\frac{1}{2}P_0(2x)$, $\frac{1}{2}P_1(2x)$, ... $\frac{1}{2}P_q(2x)$, where x = C(6D) and places them in 4D, 6D .... (4+2q)D respectively each time the subroutine is called in. $-\frac{1}{2} \leq x \leq \frac{1}{2}$, $q \leq 10$.

Preset parameters:

45	H	P	2q	D	the subroutine calculates P_0,
46	N	P	2q	θ	P_1, ... P_q.
47	M is also used by the subroutine.				

Accuracy: maximum error $2^{-34}[r + (4x)^{r-2}]$ approx.
rms value of error is $2^{r-35}(2x)^{r-2}$ approx.

Note: See Part III for detailed program.

K. Subroutines for the summation of power series

K1 Summation of a power series
Closed; 17 storage locations; time = (27+18n) msecs.

Calculates $F_n(x) = a_0x^n + a_1x^{n-1} + ... + a_n$ where x is C(R) and a_r = C(m+2r)D and places $F_n(x)$ in 0D.

Preset parameters:

45	H	P	2n	F
46	N	A	m+2n	D

(m must be even)

Accuracy: round-off error $\leq 2^{-35}[(1 - |x^n|)/(1 - |x|)]$

Note: See Part III for detailed program.

K2 Summation of a complex power series.
Closed; 30 storage locations; time = (57+48n) msecs.

Calculates $F_n(z) = \sum_{r=0}^{n} A_r z^{n-r} = X + iY$ where z = C(8D) + iC(10D) and A_r = C(m+4r)D + iC(m+4r+2)D, r = 0, 1, 2, ..., n. Places X in 4D and Y in 6D.

Preset parameters:

45	H	P	4n	F
46	N	P	m	H

(m must be even).

K3 Summation of a power series of even terms.
Closed; 27 storage locations; working position 6; time = (27+20r) msecs.

Calculates $F(x)$ = C(nD) + C[(n-2)D]·x^2 + ...C[(n-2r+2)D]·$x^{2(r-1)}$ where x = C(4D), and places $F(x)$ in 0D.

Program parameters:
Accuracy:
Notes:
] See K4.

K4 Summation of a power series.
Closed; 22 storage locations; working position 6; time = (15+20r) msecs.

Calculates $F(x)$ = C(nD) + C[(n-2)D]·x + ... C[(n-2r+2)D]·x^{r-1} where x = C(4D), and places $F(x)$ in 0D.

Program parameters:

p	A	p	F	] Orders calling in K4
p+1	G	s	F	
p+2	A	n	D	
p+3	P	2r	F	

Accuracy: maximum error is $\pm 2^{-35}\,(1-|x|^r)/(1-|x|)$

Notes: 1. C(4D) remains unchanged.
2. Since C(R) = C(4D) at the end of the subroutine $x \cdot F(x)$ may be formed in the accumulator by using the order V D in the master routine after the subroutine.
3. See Part III for detailed program.

K5 Division of a polynomial by a linear factor giving the quotient polynomial and the remainder (complex numbers).
Closed; 37 storage locations; working positions 4D, 6D; time = 65+57n msecs.

Let $P(w) = \sum_{r=0}^{n} A_r w^{n-r}$ and $\frac{P(z)}{z-w} = \sum_{r=0}^{n-1} Z_r z^{n-1-r} + \frac{Z_n}{z-w}$. Given $A_0 \ldots A_n$ this subroutine calculates $Z_0 \ldots Z_n$ by means of the recurrence relation $Z_{r+1} = Z_r w + A_{r+1}$, where $Z_0 = A_0$. The real and imaginary parts of w are stored

in 8D and 10D respectively and the real and imaginary parts of A_r are stored in (4r+m)D and (4r+m+2)D respectively. The real and imaginary parts of Z_r are stored in (q+4r)D and (q+4r+2)D. In addition, the real and imaginary parts of Z_n are stored in 4D and 6D respectively.

Note: If desired the coefficients Z_r may be written over the coefficients A_r; that is, q may equal m.

Preset parameters:

45	H	P	4n F
46	N	P	m H
47	M	P	q-m F

K7 Shift of origin of a polynomial (real numbers).
Closed; 29 storage locations; working positions 0D, 4D; time = $11(n+1)^2$ msecs.

Given the coefficients of the polynomial $\sum_{r=1}^{n} a_r x^{n-r}$, the subroutine replaces these by the coefficients of the polynomial

$$\sum_{r=0}^{n} b_r x^{n-r} \equiv \sum_{r=0}^{n} a_r [x+C(R)]^{n-r},$$

using Horner's method. a_r is stored in (h+2r)D.

Note: The contents of the register are unaffected by the operation of the subroutine.

Parameters:

A	h	D
P	2n	F

punched at the end of the subroutine by the user.

K8 Shift of origin of a polynomial (numbers expressed in floating decimal form).
Closed; 34 storage locations; working positions 4D, 0D*, 0H*, 0N*; time = $n^2/8$ secs.

Given the coefficients of the polynomial $\sum_{r=0}^{n} a_r x^{n-r}$, the subroutine replaces these by the coefficients of the polynomial

$$\sum_{r=0}^{n} b_r x^{n-r} = \sum_{r=0}^{n} a_r (x+a)^{n-r},$$

using Horner's method. All numbers are expressed in floating decimal form as used in A11. a_r is stored in location (h+2r)D and a is stored in location 6D.

Notes: 1. A11 is used as an auxiliary subroutine.
*2. Working positions 0D, 0H, and 0N are those used by A11.
3. The floating decimal accumulator must be cleared before using K8 - it will be left clear.
4. Since K8 uses the same M and Δ parameters as A11, it may follow A11 on the input tape without these parameters being replaced.

Parameters:

A	h	D
P	2n	F

punched at the end of the subroutine by the user.

Preset parameters: M and Δ parameters as for A11

L. Subroutines for evaluating logarithms.

L1 Logarithm to base 2. Large range.
Closed; 38 storage locations; working positions 4D and 8; time = (13m+776) msecs where m = integral part of logarithm.

Calculates $(1/32)\log_2 [C(6D)]$ and places result in 0D.

Accuracy: 34 binary places, but not rounded off.

Notes: 1. If $C(6D)<2^{-32}$, accumulator capacity is exceeded.
2. -2^{-34} is left in 4D.
3. See Part III for detailed program.

L2 Logarithm to base 2, small range.
Closed; 31 storage locations; working position 4D; time = 950 msecs.

Calculates $\log_2 [2 \cdot C(6D)]$, where $1/4 \leq C(6D)<1$, and places result in 0D.

Accuracy: maximum error = $\pm 2^{-34}$

M. Miscellaneous subroutines.

M1 Assembly subroutine No. 1.
Special; 16 storage locations.

Facilitates the assembly of a master routine, number sequences, and closed subroutines to form a complete program. See Part I, Sections 4-61 and 4-62.

Note: See Part III for detailed program.

M2 Assembly subroutine No. 2.
Special; 16 storage locations.

Facilitates the assembly of a master routine and closed subroutines to form a complete program. Does not apply to number sequences. See Part I, Sections 4-63 and 4-64.

Note: See Part III for detailed program.

M3 Print heading.
Closed; 10 storage locations (temporarily); working position 0.

Copies information directly from the tape to tne teleprinter and may thus be used to print a heading at the top of a sheet.

Notes: 1. M3 is placed at the front of the program tape unless R9 is used, in which case M3 follows R9. No control combinations need precede M3.
2. M3 is immediately followed by the heading, which may include line feed, carriage return, etc., according to the teleprinter code.
3. The heading is followed by blank tape, and the succeeding orders should be prefaced by a control combination of the form P K T n K.

P. Print subroutines.

P1 Print a single positive number (without layout or round-off).
Closed; 21 storage locations; time = (171n+10) msecs.

Prints the positive number in 0D to n places of decimals, leaving $R \cdot 10^n$ in 0D, where R is the remainder.

Program parameter:

p	A	p	F
p+1	G	s	F
p+2	P	n	F

orders calling in P1

Notes: 1. Teleprinter must be on figure shift.
2. Layout must be separately controlled.
3. Round-off is not included.

P6 Print short positive integer.
Closed; 32 storage locations; working positions 1, 4, and 5; time = about 900 msecs.

Prints $2^{16} \cdot C(0)$ with suppression of nonsignificant zeros but without layout.

P7 Print positive integer up to 10 digits.
Closed; even; 35 storage locations; working position 4D; time = approx. 1.8 sec.

Prints $2^{34} \cdot C(0D)$ with zero suppression but without layout.

Notes: 1. Teleprinter must be on figure shift.
2. Layout must be separately controlled.
3. C(0D) must be positive and less than $10^{10} \cdot 2^{-34}$.
4. If the number to be printed is less than 10^9, the left-hand zeros are replaced by spaces. In any case, 10 positions on the paper are used.
5. See Part III for detailed program.

P8 Print table of positive integers in a special layout.
Closed; even; 62 storage locations; working position 4.

Prints $n = 2^{34} \cdot C(0D)$ in a special layout. $0 \leq n < 10^{10}$. Layout: first number in each row printed to full 10 decimal digits; of the remaining numbers, only the least significant d decimal digits are printed. c numbers in each row, one space between columns. 5 lines in each block.

P10 Print a short positive integer, with conversion check, error indication, and optional suppression of nonsignificant zeros.
Closed; 70 storage locations; working positions 0, 1, 4, 5, and 6; time = approx. 1.8 sec.

Prints C(4) as a short integer. Failure of F-check is indicated by "?" after incorrect digit. Failure of binary-decimal conversion check is indicated by "??" after incorrect number.

Program parameter: P F for no zero suppression
or I F for zero suppression

P11 Print signed decimals in a preset layout (with digit check).
Closed; 52+s* storage locations; working position 4; time = about 180 msecs per symbol.

Prints C(0D) rounded-off, preceded by negative sign if negative. Layout: n columns with s spaces between columns; preset digit layout (see note 8); blocks of 5 lines with one space between blocks.

Preset parameters:	45	H	round-off order
	46	N	P 5n F
	47	M	P (44+s) θ
	48	Δ	P x F

Notes: 1. Figure shift takes place during input of orders.
2. Negative numbers are preceded by -, positive numbers by a space.
3. Maximum width of layout = 70 symbols.
4. If the F order shows an error, a line feed occurs and the next digit printed may be in error.
*5. s can be 1, 2, 3, or 4.
6. Last order on library tape is P 5 F, giving number of lines in block. This may be altered if required, but the second preset parameter will then be n (block length).
7. If the subroutine starts in q, a new block will be started if (q+20) is cleared before the next number is printed.
8. The digit layout is determined by the fourth preset parameter P x F, where x may be obtained as follows. Imagine the printed characters, including digits and spaces (only single spaces are permissible) laid out in the form below, starting with the most significant digit at the left-hand end. Then add together the numbers below the spaces, and the number above the last digit; the sum is x.

8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
24576	12288	6144	3072	1536	768	384	192	96	48	24	12	6	

For example: (i) to print 10 digit numbers with spaces after the 3rd, 6th, and 9th digits, x = 6144 + 384 + 24 + 4 = 6556; (ii) to print 8 digit numbers with spaces after the 4th and 5th digits, x = 3072 + 768 + 32 = 3872.
9. See Part III for detailed program.

P12 Print signed integers in a standard layout (with digit check).
Closed; 57 storage locations; working position 4; time = about 300 msecs per symbol.

Prints 2^{34}·C(0D) preceded by negative sign if negative. Layout: 5 columns, 5 lines per block; numbers in subcolumns of 4, 3 and 3 digits with one space between numbers.

P13 Print single decimal (without layout or round-off) with digit check and variable digit layout.
Closed; 30 storage locations; time = (9+30n) msecs.

Prints the positive number in 0D to n places of decimals, leaving R·10 in 0D, where R is the remainder. The digit spacing and n are determined by a program parameter P x F, where x is calculated as in P11.

Program parameter:			
	p	A p F	orders calling in P13
	p+1	G s F	
	p+2	P x F	

Notes: 1. Teleprinter must be on figure shift.
2. Round-off is not included.
3. Failure of F check causes # to be printed.

P14 Print signed decimal with round-off and digit check. Layout controlled by program.
Closed; 46 storage locations.

Prints the decimal number in C(0D), rounded-off. Digit spacing, number of digits printed and layout are determined by a program parameter.

Preset parameter:	45	H	A m D — round-off order
Program parameter:	p	A p F	orders calling in P13
	p+1	G s F	
	p+2	P x F	
	or	K 4096+x F	Layout constant: see note 2.

Notes: 1. Figure shift is called during the input of orders.
2. The number of digits and their spacing is determined by the program parameter, which is calculated as in subroutine P11. Carriage return and line feed will occur before the number is printed if K 4096 F is added to this layout constant. Each number is followed by a space.
3. If the F order shows an error a line feed will occur and the next digit printed may be in error.
4. Negative numbers are preceded by a negative sign, positive numbers by a space.
5. See Part III for detailed program.

P15 Print positive number held in register (without digit check or layout).
Closed; 24 storage locations; working positions 0D, 4.

This subroutine will print the number held in the register to n decimals. Negative numbers are printed as complements. If P15 is entered at the first order a new line of printing is commenced. If it is entered at the third order the number is printed on the same line.

Accuracy: no round-off is incorporated.

Notes: 1. The F-check is not used.
2. Each number is followed by one space.

Parameters: P n F is punched at the end of the subroutine.

Q. Quadrature subroutines.

Q1 Evaluation of definite integral, using Simpson's rule.

Closed; 46 storage locations; working positions 0D and 4; time = 36+n(36+T) msecs, where n = (b-a+h)/h = number of ordinates, T = time for auxiliary subroutine.

Places $3\int_a^b f(x)dx$ in pD, where pD is specified by a preset parameter and a, b, and h (the interval of integration) are given by program parameters. f(x) is computed by an auxiliary closed subroutine placed with its first order in q and designed to put f(x) in 0D where x = C(0D).

Preset parameters:

45	H	P p	D
46	N	G q	F

Program parameters:

p	A	p	F	] orders calling in Q1
p+1	G	s	F	]
p+2	P	2^{15}a	F	
p+3	P	2^{15}h	F	
p+4	P	2^{15}b	F	

Accuracy: rounding-off error is $n \cdot 2^{-35}$ in the worst case.

Notes: 1. $-1 \leq a < b < 1$. h must be positive and such that (b-a)/h is an even integer.

2. The program parameters are shown above as pseudo-orders. They are really the values of a, b, and h expressed as short numbers.

3. Q1 uses 0D and 4 but these positions may also be used by the auxiliary subroutine.

4. If desired, the integration may be made to terminate when f(x) becomes less than a specified quantity by including a suitable test and conditional order in the auxiliary subroutine.

5. See Part III for detailed program.

Q2 Evaluation of a definite integral, using Gauss' 5-point formula.

Closed; even; 52 storage locations; working position (p+4)D; time = 206 msecs + 5(auxiliary time).

Places $\int_a^b f(x)dx$ in 0D where a = C(pD) and (b-a) = C[(p+2)D] and f(x) is computed by an auxiliary closed subroutine placed with its first order in n and designed to put f(x) in 0D where x = C(0D).

Preset parameters:

45	H	P p	D
46	N	G n	F

Accuracy: rounding-off error is $2^{-35}[1+5(b-a)]$ in the worst case.

Notes: 1. Remainder term of the formula used is $4.10^{-13} \cdot (b-a)^{11} f^{(10)}(x')$ where $a \leq x \leq b$.

2. At the end of the process $[1/(b-a)]\int_a^b f(x)dx$ is left in (p+4)D.

3. R9 must be in the store when Q2 is read.

Q3 Quadrature, using Gauss' 6-point formula.
Closed; even; 48 storage locations; working position (m+4)D;
time = 240 msecs + 6 times the time of the auxiliary subroutine.

Places $I = \int_{a-h}^{a+h} f(x)dx$ in 0D, where a = C(mD) and h = C[(m+2)D]; and f(x) is computed by an auxiliary closed subroutine whose first order is in n and which places f[C(0D)] in 0D.

Accuracy: truncation error of the formula used is

$$10^{-15} \times f^{12}(\theta) \cdot (2h)^{13} = 0.7 \times 10^{-7} \frac{f^{12}(\theta)}{(12)!} (2h)^{13}.$$

Rounding-off error in worst case is $2^{-35}[1 + 10h + 10h\ (\max f'(x))]$.
$(a+h) \geq \theta \geq (a-h)$

Notes: 1. R9 must be in the store when Q3 is read.
2. I/2h, that is the mean value, is placed in (m+4)D.
3. See Part III for detailed program.

Preset parameters:					
	45	H	P m D		location of parameters and working space
	46	N	P n F		location of auxiliary subroutine

R. Input subroutines.

R1 Input of a sequence of signed long decimal fractions.
Closed; 55 storage locations; working positions 0, 1, 4, 5, and 6.

Given a sequence of numbers punched as decimals followed by sign, this subroutine places these numbers in pD, (p+2)D, (p+4)D .... and returns control to the master routine when F appears on tape.

Preset parameters:			
	45	H	positions are used by subroutine
	46	N	

Program parameter:					
	m	A	m	F	orders calling in R1.
	m+1	G	s	F	
	m+2	T	p	D	

Notes: 1. Decimal point is immediately before first digit punched.
2. Any number of digits up to 10 may be punched; more will exceed the capacity of the accumulator.
3. Blank or erased tape is treated as F.
4. See Part III for detailed program.

R2 Input of positive integer during input of orders
Special; 15 storage locations (temporarily);

Reads the input tape and converts the decimal integers thereon to binary form multiplied by 2^{-34} and places these in sequence in storage locations mD, (m+2)D, (m+4)D, etc.

Parameter: T m D must follow the subroutine.

Notes: 1. After the subroutine T m D is punched, followed by the integers, each terminated by F with the exception of the last one which is terminated by π T Z.

2. After the integers have been read, π T Z returns control to the initial orders and subsequent orders read from the tape will be written over R2.

3. See Part III for detailed program.

R3 Input of one signed long decimal fraction.
Closed; even; 41 storage locations; working positions 4D and 6D.

Reads one fraction punched in decimal form followed by sign, and places it in 0D.

R4 Input of one signed integer.
Closed; 22 storage locations; working positions 4, 5, and 6.

Reads one integer y punched in decimal form followed by sign, and places $y \cdot 2^{-34}$ in 0D.

Notes: 1. $|y|<2^{-34}$

2. R4 is applicable to either long or short numbers; in the latter case $y \cdot 2^{-16}$ will be left in 0 provided that $-2^{16} \leq y < 2^{16}$.

R5 Input of a sequence of signed long decimal fractions during input of orders.
Special; even; 32 storage locations (temporarily); working position 0D.

The numbers are punched on a separate tape as sign followed by decimals, the first number being preceded by Z T X. After the subroutine, T m D is punched, followed by the sequence of numbers, which is copied in the reverse direction. The numbers are then placed in mD, (m-2)D, etc., so that mD is the location of the number originally punched last.

Parameter: T m D must follow the subroutine.

Notes: 1. Any number of digits may be punched.

2. After the decimals have been read control is returned to the initial orders and subsequent orders read from the tape will be written over R5.

R7 Input of a sequence of signed long decimal fractions during program.
Closed; even; 37 storage locations; working position 0D.

The numbers are punched on a separate tape as sign followed by decimals, each group being preceded by X. This tape is then copied on the main tape in the reverse direction. Each time the subroutine is used, it will read the numbers from the tape until X is reached. Control is then referred back to the main program, the numbers on the tape having been placed in storage locations mD, (m-2)D, etc., where mD is the storage location of the number originally punched last in that group.

Program parameter:

p	A p F	orders calling in R1
p+1	G s F	
p+2	T m D	

Notes: 1. Any number of digits may be punched.
2. In a decimal fraction the last significant digits of which are zero, these zeros may be omitted.
3. See Part III for detailed program.

R9 Input of positive integers during input of orders. Standard form for regular use.
Special: 15 storage locations.

The actual orders of this subroutine are identical with those of R2, but R9 is intended always to be placed in locations 56 to 70 inclusive, and to remain there throughout the input of a whole program, being used any number of times. Each time it is used it will read a sequence of positive decimal integers and place them in consecutive long storage locations.

Notes: 1. The subroutine tape commences with P K T 56 K, so that it may be copied immediately at the head of a tape. It does not have E 13 Z at the end, so that it is not automatically obeyed after being read.
2. R9 is called in by the control combination E 69 K T m D. This is followed by the integers each terminated by F except the last, which is terminated by π to return control to the initial orders. After this must be punched a control combination to restore the transfer order, e.g., T Z. The integers will be placed in mD, (m+2)D, (m+4)D, etc.
3. Negative integers may be read if 2^{35} is added to each before punching.

S. Subroutines for evaluation of fractional powers.

S1 Square root, slow.
Closed; 22 storage locations; working positions 4, 5, and 8; time = 825 msecs.

Forms $\sqrt{C(6D)}$ and places result in 0D.

Accuracy: 2^{-34}. Last digit is always 1.

Notes: 1. If $C(6D) \leq -2^{-32}$, accumulator capacity is exceeded. If $-2^{-32} < C(6D) < 0$ final $C(0D) = -(1-2^{-34})$.
2. C(6D) is left unchanged. C(4D) becomes -2^{-34}.

S2 Square root, fast.
Closed; 22 storage locations; working position 0D; time = approx. (36n+180) msecs, where $(2\ 1/4)^{-n-1} \leq C(4D) < (2\ 1/4)^{-n}$.

Forms $\sqrt{C(4D)}$ where $C(4D) > 0$ and places result in 4D.

Accuracy: Number of significant figures in result is two less than number of significant figures in argument.

Notes: 1. If C(4D) = 0, subroutine continues to cycle indefinitely.
2. See Part III for detailed program.

S3 Cube root.
Closed; 25 storage locations; working positions 4, 5, 8, and 9; time = approx. 1 sec.

Forms cube root of C(6D) and places result in 0D. C(6D) may be positive or negative and is left unchanged at the end.

Note: See Part III for detailed program.

S4 Reciprocal square root.
Closed; 22 storage locations; time = approx. (36n+180) msecs, where $(2.25)^{-n-1} \leq C(4D) < (2.25)^{-n}$.

Forms $C(0D)/\sqrt{C(4D)}$ and places the result in 0D. C(4D) must be >0.

Accuracy
Notes] See S2

T. Subroutines for calculating trigonometrical functions.

T1 Cosine, rapid.
Closed; even; 44 storage locations; working position 0D; time = .82 msecs.

Forms $0.5 \cos[2 \cdot C(4D)]$ where $|2 \cdot C(4D)| \leq \pi/2$, and places result in 4D.

Accuracy: maximum error = 2^{-33}.

T3 General cosine (used with R9).
Closed; even; 59 storage locations; working position 0D; time = 105 msecs.

Forms $0.5 \cos[2^m \cdot C(4D)]$ and places result in 4D. R9 must be placed in the store before T3 is read.

Preset parameter: 45 | H | P 2^{m-3} F (or P D for m = 2)

Accuracy: maximum error has modulus $< 2^{-35+m}$.

Notes: 1. Applies to angles of any magnitude.
2. See Part III for detailed program.

T4 Inverse cosine.
Closed; even; 33 storage locations; working positions 0D and 6D; time = approx. 900 msecs.

Forms $0.5 \arccos [2 \cdot C(4D)]$ if $0 \leq C(4D) \leq 0.5$, or $0.5 \arccos [2|C(4D)|]$ if $-0.5 \leq C(4D) \leq 0$, and places result in 0D. $0 \leq \arccos [2 \cdot C(4D)] \leq \pi/2$.

Accuracy: maximum error has modulus less than 2^{-18}.

Note: See Part III for detailed program.

T5 0.5 cos x and 0.5 sin x at equal intervals of x. Version 1.
Open; even; 20 storage locations; time = 36 msecs.

Calculates 0.5 cos x and 0.5 sin x at equal intervals δx of x by use of the recurrence relation.

$0.5 \cos(x + \delta x) = (0.5 \cos x)\cos \delta x - (0.5 \sin x)\sin \delta x$,

$0.5 \sin(x + \delta x) = (0.5 \sin x)\cos \delta x + (0.5 \cos x)\sin \delta x$,

using long numbers.

Current value of 0.5 cos x in location 2D of subroutine. Current value of 0.5 sin x in location 4D of subroutine.

Cos δx and sin δx must be provided in mD, (m+2)D, respectively. T5 is fed into machine with 0.5 cos x = 1/2, 0.5 sin x = 0. Each entry advances x by δx.

Preset parameter: 45 | H | P m D

Notes: 1. Initial values may be reset by entering at order 6.

2. Other starting values may be set by direct planting. It is possible to change the scale factor by planting a cos x, a sin x.

3. See Part III for detailed program.

T6 0.5 cos x and 0.5 sin x at equal intervals of x. Version 2.
Open; even; 24 storage locations; time = 36 msecs.

Similar to T5, but with different starting condition: the first entry sets 0.5 cos x = 1/2, 0.5 sin x = 0, and each subsequent entry advances the value of x by δx.

Preset parameter: 45 | H | P m D

T7 Sine, rapid (used with R9).
Closed; even; 36 storage locations; working position 0D; time = 81 msecs.

Forms $0.5 \sin[2 \cdot C(4D)]$ where $|2 \cdot C(4D)| \leq \pi/2$ and places result in 4D. R9 must be in the store when T7 is read.

Accuracy: maximum error is $\pm 2^{-33}$.

Note: See Part III for detailed program.

T8 Inverse sine.
Closed; even; 37 storage locations; working positions 6D and 8D; time = approx. 1 sec.

Forms $0.5 \sin^{-1}[2 \cdot C(4D)]$ where $-1/2 \leq C(4D) \leq 1/2$ and places result in 0D.

Accuracy: probable error is 2^{-19} for the range $-15/32 \leq C(4D) \leq 15/32$.

T9 Tangent, rapid (used with R9).
Closed; even; 46 storage locations; working position 0D; time = 155 msecs.

Forms tan C(4D) where $-\pi/4 < C(4D) < \pi/4$ and places result in 4D. R9 must be in the store while T9 is being read.

Accuracy: maximum error is 2^{-33}

U. Subroutines for counting operations.

U1 Counting subroutine No. 1.
Closed; 33 storage locations; working position 4; time = 45 msecs per cycle + time for secondary subroutine.

Controls a secondary subroutine called in by a group of orders of the following form

p	A	p	F
p+1	G	m	F
p+2	A	q	F

The secondary subroutine is executed (t-s+r)/r times with q = s, s+r, s+2r, ..., t before control is returned to the master routine. (t-s+r)/r should be an integer. r, s, t, and m are specified by program parameters.

U2 Counting subroutine No. 2.
Open; 17 storage locations + 2 for each pair of parameters; time = 30 msecs (average).

This subroutine is incorporated in a program followed by pairs of parameters as follows: E a_1 F, P q_1 F; E a_2 F, P q_2 F, etc., (any number of pairs). Control is transferred at the end of the subroutine to a_1 if $q \leq q_1$ and to a_2 if $q_1 < q \leq q_2$, etc., where it is supposed that the subroutine has just been operated for the qth time.

U3 Counting subroutine No. 3.
Open; 17 storage locations + 2 for each pair of parameters; time = 30 msecs (average).

This subroutine is incorporated in a program followed by pairs of parameters as follows: E a_1 F, P q_1 F; E a_2 F, P q_2 F, etc., (any number of pairs). The first q_1 times the subroutine is operated, control is transferred to a_1, the next q_2 times to a_2, etc.

Program parameters:

E a_1 F	punched after the subroutine
P q_1 F	
E a_2 F	
P q_2 F	etc.

Notes: 1. A pair of parameters Z F, P 1 F will cause the machine to stop.
2. See Part III for detailed program.

U4 Counting subroutine No. 4.
Closed; 28 storage locations; time = 45 msecs per cycle + time for secondary subroutine.

Controls a secondary subroutine called in by a group of orders of the following form:

p	A	p	F
p+1	G	c	F
p+2	*A	q	F

The secondary subroutine is executed n times with q = s, s+r, s+2r, ..., s+(n-1)r before control is returned to the master routine.

m	A m F	
m+1	G e F	*A may be replaced by any other function letter according to the requirements of the secondary subroutine.
m+2	P r F	
m+3	*A s F	
m+4	A n F	
m+5	G c F	

Note: See Part III for detailed program.

U5 Counting subroutine No. 5.
Open; 21 storage locations + parameters; time = 33 msecs (average).

Similar to U3 but when control has been transferred q_r times to a_r, the subroutine is automatically reset and control is then transferred q_1 times to a_1, and the whole cycle is repeated.

Program parameters:

E a_1 F
P q_1 F
E a_2 F
P q_2 F
.
.
E a_r F
P q_r F
E 8 θ

Notes: 1. If E 7 θ is punched after the parameters instead of E 8 θ, the cycle will be repeated starting at the second exit, i.e., control will be transferred q_2 times to a_2, q_3 times to a_3, etc.

2. If the following orders are punched instead of E 8 θ the cycle will be repeated starting at the rth exit:

m	A m+2 F
m+1	E 7 θ
m+2	P 2r F

3. The subroutine may be made to repeat starting at any point in the cycle by means of orders in the master routine which place suitable quantities in 19 θ and 20 θ.

V1 Multiplication of vector by symmetric matrix.
Closed; even; 47 storage locations; working positions 0D, 4, 5, 6, and 7; time = (36n+18)n msecs.

Given a symmetric n-by-n matrix of which only 0.5 n(n+1) elements are stored starting in mD, and given an n-vector stored in cD, (c+2)D, ..., (c+2n-2)D, this routine will form their product and place it in sD, (s+2)D, ..., (s+2n-2)D.

Preset parameters:

45	H	P 2n F	
46	N	H c D	C(cD) = first element of vector.
47	M	V m D	C(mD) = first element of matrix.
48	Δ	T s D	C(aD) = first element of product.
49	L	Y F	included at head of library tape.
50	X	P 24 θ	

Notes: 1. The matrix elements must be placed in the store in the following order:

0 . . .
1 2 . .
3 4 5 .
. . . .

2. If it is desired to change the values of c, m, s in the course of the program, this can be done by changes in the following psuedo-orders (a) H c D in (p+26), (b) V m D in (p+27), and (c) T s D in (p+28), where p is the address of the first order of the subroutine.

3. If it is desired to incorporate a left-shift in the multiplication, this can be done by replacing the round-off order Y F, which is taken in in the form of a parameter, at the head of the tape.

4. See Part III for detailed program.

V2 Addition and subtraction of n dimensional vectors.
Closed; 25 storage locations; working position 1; time = (13+17.5n) msecs.

Adds (or subtracts) the vector with components in the n long storage locations ending in bD, to (from) the vector with components in the n long storage locations ending in aD, putting the components of the result into the n long storage locations ending in cD.

Preset parameter: 45 | H | P 2n F

Program parameters:

For addition

p+2	A a D
p+3	P b-a F
p+4	O c-b F

For subtraction

p+2	A a D
p+3	K 4096+b-a F
p+4	L c-b F

Note: See Part III for detailed program.

PART III

PROGRAMS OF SELECTED LIBRARY SUBROUTINES

The following notation is used on all library program sheets.

Entry points:	If control may arrive at an order by being transferred there by an E or G order the location of the latter (relative to the first order of the subroutine) is shown on the extreme left, with an arrow pointing to the address of the order to which control is transferred, e.g., 16→23 T 6 θ.
Unconditional transfers of control:	A horizontal line is drawn underneath every E or G order which is intended to produce a transfer of control each time it is encountered.
Variable orders:	Orders and pseudo-orders which are to be changed during the course of the calculation are shown in brackets.
Pseudo-orders:	A double vertical line is drawn on the left of the contents of all storage locations which are intended never to be obeyed as orders.
Use of J:	When reading the address part of an order the initial orders treat the letter J as a digit of value 10. Some subroutines therefore use J for the address 10, thus saving one row of holes on the tape.
Preset parameters:	C(45), C(46) .... when used as preset parameters are referred to as H parameter, N parameter ...
Control combinations:	Any "order" with code letter K or Z is a control combination. The more common ones are described in Part I, Section 2-5, and the less common ones in Appendix C.

A3 Special arithmetical operations on real numbers in floating decimal form.

See Part I, Section 4-82. In the store, x is represented by the number $2^{-12}x_0 + 2^{-9}p_x$. In the routine's "floating decimal accumulator" the number a is represented by $2^{-2}a_0$ in 10D and $2^{-15}p_a$ in 9. In the course of forming xy, its representation is adjusted by a factor of 10, and its exponent corrected accordingly. Thus $(xy)_0 = 10^{-1}x_0y_0$ and $p_{xy} = p_x + p_y + 1$. This is done to prevent overflow in the numerical part of the floating accumulator.

	Orders			Notes
Part 1:	R2 modified			order 3 in R2 is altered to S 40 D. Thus the integers are placed in the store negatively and become -10^{-n}, n = 0...7
	T	60πθ		
60πθ	17 179 869 184		F	
62πθ	1 717 986 918		F	
64πθ	171 798 692		F	
66πθ	17 179 869		F	
68πθ	1 717 987		F	
70πθ	171 799		F	
72πθ	17 180		F	
74πθ	1 718		π	
	T	56πZ		(clears sandwich digit between 56 and 57)
	P		F	
	T		Z	
0	A	3	F	plant link order
1	T	55	θ	
2	A		H	form and plant order to extract x
3	A	23	θ	
4	T	8	θ	
5	A		N	form and plant order to extract y
6	A	23	θ	
7	T	16	θ	
8	(A		D)	extract x from store — unpack x
9	U		D	copy in 0D
10	L	256	F	shift to remove exponent
11	U	4	D	$2^{-2}x_0$ to 4D
12	R	256	F	shift to form $2^{-12}x_0$
13	S		D	cancel x_0 leaving $-2^{-9}p_x$
14	R	16	F	$-2^{-15}p_x$ to 0
15	T		F	
16	(A		D)	unpack y similarly
17	U	12	D	
18	L	256	F	
19	U	6	D	
20	R	256	F	
21	S	12	D	
22	R	16	F	

23	A		F	$-2^{-15}(p_x+p_y+1)$ to 8
24	S	2	F	
25	T	8	F	
26	H	4	D	$2^{-4}x_0y_0$ to 0D
27	V	6	D	
28	Y		F	
29	T		D	
30	H		D	$2^{-2}10^{-1}x_0y_0$ to 4D and R
31	V	56$\pi\theta$		
32	Y		F	
33	T	4	D	
34	H	4	D	
35	A	9	F	form $2^{-15}(p_a-p_x-p_y-1)$
36	A	8	F	
37	E	45	θ	jump if xy is smaller than a (approx.)
38	T		F	clear accumulator
39	H	10	D	$2^{-2}a_0$ to R
40	A	4	D	$10^{-1}x_0y_0$ replaces a_0
41	T	10	D	
42	S	8	F	p_x+p_y+1 replaces p_a
43	T	9	F	
44	S		F	$2^{-15}(p_x+p_y+1-p_a)$ to Acc.
37→45	L		D	multiply by 2
46	S	59	θ	
47	E	54	θ	jump if smaller component is negligible
48	A	58	θ	form and plant N order referring to
49	T	50	θ	appropriate power of 10
50	(N		D)	multiply C(R) by $10^{-\lvert p_a-p_x-p_y-1\rvert}$
51	A	10	D	form sum and place in 10D
52	Y		F	
53	T	10	D	
47→54	T		F	clear accumulator
55	(E		F)	link order
56	L	1229	F	4/10
57	Y	819	F	
58	N	76$\pi\theta$		
59	P	16	F	

interchange xy and a if a is the smaller

(Storage locations $60\pi\theta$ to $74\pi\theta$ contain the constants -10^{-n}, n = 0 ... 7, which are read before the orders)

Part 2:	T	76	Z	
	G		K	(puts new reference address in 42)
	T	44πZ		(clears sandwich digit between 56 and 57)
	P		F	
	T		Z	
0	A	3	F	plant link order
1	T	43	θ	
2	A		M	form and plant transfer order
3	A	13	θ	
4	T	39	θ	

```
     5 | S  47  θ |  count number of
22 → 6 | A  47  θ |    adjustments
     7 | T   1  F |
     8 | H  10  D |
     9 | V  10  D |  jump to 23
    10 | S  44πθ  |    if |a0|>0.4
    11 | E  23  θ |
    12 | H  46  θ |  10/16 to R
    13 | T      F |  clear accumulator
    14 | V  10  D |
    15 | L   4  F |  multiply a0
    16 | T  10  D |    by 10
    17 | A   9  F |
    18 | S   2  F |  correct pa
    19 | T   9  F |    accordingly
    20 | A   1  F |
    21 | S  46  θ |  count number of
    22 | G   6  θ |    adjustments
11 → 23 | T      F |  clear accumulator
    24 | A   9  F |
    25 | S  48  θ |  jump if
    26 | G  28  θ |    pa<511
    27 | Z      F |  stop if pa≥511
26 → 28 | A  49  θ |  jump if pa≥-511
    29 | E  31  θ |
    30 | T      F |  clear acc. if pa<-511
29 → 31 | S  48  θ |
    32 | T   9  F |
    33 | A  10  D |
    34 | R 256  F |
    35 | Y      F |
    36 | R  16  F |
    37 | A   9  F |
    38 | L  16  F |  form 2^-9 p + 2^-12 a0
    39 | (T     D)|
    40 | T  10  D |
    41 | S  48  θ |  clear floating accumulator
    42 | T   9  F |
    43 | (E     F)|  link
    44 | M  81  D |  1/100 (rounded down)
    45 | P 327  D |
    46 | J      F |  10/16
    47 | π      F |  11/16
    48 | P 511  F |
    49 | P1022  F |
```

Adjust representation of a by multiplying a_0 by 10 and decreasing p_a by 1. This cycle is repeated until either $|a_0|>0.4$ or 10 adjustments have been made (to provide for $a_0 = 0$)

If $p_a \geq 511$, stop the machine. If $p_a < -511$, replace p_a by -511

Pack a and place in store

A5 Special arithmetical operations on complex numbers in floating decimal form.

See Part I, Section 4-82. In the store, z_1 is represented by $2^{-2}(x_1)_0$ in rD and $2^{-2}(y_1)_0$ in (r+2)D, each rounded off to 28 binary places. The last 6 digits of these locations contain the most and least significant halves respectively of the 12-digit integer p, the left-hand digit of which is treated as a sign digit. In the routine's "complex floating accumulator" the number z_a is represented by $2^{-2}(x_a)_0$ in 14D, $2^{-2}(y_a)_0$ in 16D, and $2^{-16}p_a$ in 18F. In the course of forming $z_1z_2 = z_3$, its representation is adjusted by a factor of 10, and its exponent corrected accordingly. Thus $(z_3)_0 = 10^{-1}(z_1)_0(z_2)_0$ and $p_3 = p_1 + p_2 + 1$. This is done to prevent overflow in the numerical part of the "accumulator."

Part 1:

	R2 modified				order 3 in R2 is altered to S 40 D. Thus the following integers are read negatively and become -10^{-n}, n = 0 ... 8.
	T	110πθ			
110πθ	17	179	869	184 F	
112πθ	1	717	986	918 F	
114πθ		171	798	692 F	
116πθ		17	179	869 F	
118πθ		1	717	987 F	
120πθ			171	799 F	
122πθ			17	180 F	
124πθ			1	718 F	
126πθ				172 π	
	T			Z	
0	A		3	F	plant link
1	T		104	θ	plant link
2	A			H	form and plant orders to extract z_1
3	A		105	θ	add C D
4	U		17	θ	form and plant orders to extract z_1
5	U		26	θ	form and plant orders to extract z_1
6	A		200	θ	add P 2 F
7	U		19	θ	form and plant orders to extract z_1
8	T		28	θ	form and plant orders to extract z_1
9	A			N	form and plant orders to extract z_2
10	A		105	θ	add C D
11	U		21	θ	form and plant orders to extract z_2
12	U		34	θ	form and plant orders to extract z_2
13	A		200	θ	add P 2 F
14	U		23	θ	form and plant orders to extract z_2
15	T		36	θ	form and plant orders to extract z_2
16	H		194πθ		prepare to collate numerical parts
17	(C			D)	$2^{-2}(x_1)_0$ to 4D
18	T		4	D	$2^{-2}(x_1)_0$ to 4D
19	(C			D)	$2^{-2}(y_1)_0$ to 6D
20	T		6	D	$2^{-2}(y_1)_0$ to 6D
21	(C			D)	$2^{-2}(x_2)_0$ to 10D
22	T		10	D	$2^{-2}(x_2)_0$ to 10D
23	(C			D)	$2^{-2}(y_2)_0$ to 12D
24	T		12	D	$2^{-2}(y_2)_0$ to 12D

unpack numerical parts

25	H	196πθ		prepare to collate exponents
26	(C		D)	most significant half of p_1
27	L	16	F	
28	(C		D)	least significant half of p_1
29	T	8	D	shift to top of Acc. to get top digit in sign position
30	A	8	F	
31	L	8	F	
32	R	8	F	shift to form $2^{-16}p_1$ and send to 0
33	T		F	
34	(C		D)	unpack p_2 similarly
35	L	16	F	
36	(C		D)	
37	T	8	D	
38	A	8	D	
39	L	8	F	
40	R	8	F	
41	A		F	
42	A	203	θ	add 2^{-16} ⎫ $2^{-16}(p_1+p_2+1) = 2^{-16}p_3$ to 8
43	A	8	F	
44	H	6	D	$2^{-4}\left[(x_1)_0(x_2)_0 - (y_1)_0(y_2)_0\right]$
45	N	12	D	$= 10\cdot 2^{-4}(x_3)_0$ to 4D
46	H	4	D	
47	V	10	D	
48	Y		F	
49	T	4	D	
50	V	12	D	$2^{-4}\left[(x_1)_0(y_2)_0 + (x_2)_0(y_1)_0\right]$
51	H	6	D	$= 10\cdot 2^{-4}(y_3)_0$ to 6D
52	V	10	D	
53	Y		F	
54	T	6	D	
55	H	106πθ		$2^{-2}(x_3)_0$ to 4D
56	V	4	D	
57	Y		F	
58	T	4	D	
59	V	6	D	$2^{-2}(y_3)_0$ to 6D
60	Y		F	
61	T	6	D	
62	A	8	F	form $2^{-16}p_m$ in Acc. where p_m = max(p_a,p_3)
63	S	18	F	
64	E	66	θ	
65	T		F	
64→ 66	A	18	F	$2^{-16}p_m$ to 19
67	U	19	F	
68	S	8	F	form $2^{-15}[2(p_m-p_3) - 18]$
69	L	1	F	
70	S	108	θ	jump if $p_3 \leq p_a - 9$
71	E	101	θ	
72	A	109	θ	form and plant order H [110 + $2(p_m-p_3)$]πθ
73	U	83	θ	
74	T	89	θ	

form numerical parts of product

Prepare to align numbers for addition. If one number is negligibly small compared with the other, jump to 101 or 96, by-passing the addition.

	75	A	19	F	form $2^{-15}[2(p_m-p_a) - 18]$
	76	S	18	F	
	77	L	1	F	
	78	S	108	θ	
	79	E	96	θ	jump if $p_3 \geq p_a+9$
	80	A	109	θ	form and plant the order H $[110 + 2(p_m-p_a)]\pi\theta$
	81	U	85	θ	
	82	T	91	θ	
	83	(H		D)	align and add real parts
	84	N	4	D	
	85	(H		D)	
	86	N	14	D	
	87	Y		F	
	88	T	14	D	
	89	(H		D)	align and add imaginary parts
	90	N	6	D	
	91	(H		D)	
	92	N	16	D	
	93	Y		F	
	94	T	16	D	
	95	E	101	θ	
79→	96	T		F	clear accumulator
	97	A	4	D	replace $(z_a)_0$ by $(z_3)_0$
	98	T	14	D	
	99	A	6	D	
	100	T	16	D	
71, 95→	101	T		F	clear accumulator
	102	A	19	F	replace p_a by p_m
	103	T	18	F	
	104	(E		F)	link
	105	C		D	
		T	106π	Z	(clears sandwich digit between 106 and 107)
		P		F	
		T	106	Z	
	106	L	1229	F	4/10
	107	Y	819	F	
	108	P	18	F	
	109	H	128$\pi\theta$		

If $p_a > p_3$, $(z_a)_0$ is multiplied by 1 and $(z_3)_0$ by a negative power of 10, and vice versa

$p_3 \geq p_a+9$

(Storage locations $110\pi\theta$ to $126\pi\theta$ contain the constants -10^{-n}, n = 0 ... 8, which are read before the orders)

Part 2:	T	128	Z	
	G		K	(puts new reference address in 42)
	T	62π	Z	(clears sandwich digit between 62 and 63)
	P		F	
	T	64π	Z	(clears sandwich digit between 64 and 65)
	P		F	
	T		Z	

	0	A	3	F	plant link
	1	T	61	θ	
	2	A		M	form and plant transfer orders
	3	A	17	θ	
	4	U	56	θ	
	5	A	72	θ	
	6	T	50	θ	
	7	S	70	θ	count number of adjustments
29 →	8	A	70	θ	
	9	T	19	F	
	10	H	14	D	jump to 30 if $\lvert(z_a)_0\rvert > 0.4$
	11	V	14	D	
	12	H	16	D	
	13	V	16	D	
	14	S	62$\pi\theta$		
	15	E	30	θ	
	16	H	71	θ	10/16 to register
	17	T		D	clear accumulator
	18	V	14	D	multiply $(z_a)_0$ by 10
	19	L	4	F	
	20	T	14	D	
	21	V	16	D	
	22	L	4	F	
	23	T	16	D	
	24	A	18	F	correct p_a accordingly
	25	S	75	θ	
	26	T	18	F	
	27	A	19	F	count number of adjustments
	28	S	71	θ	
	29	G	8	θ	
15 →	30	T		F	clear accumulator
	31	A	18	F	jump if $p_a < 2047$
	32	S	73	θ	
	33	G	35	θ	
	34	Z		F	stop if $p_a \geq 2047$
33 →	35	A	74	θ	jump if $p_a \geq -2047$
	36	E	38	θ	
	37	T		F	clear Acc. if $p_a < -2047$
36 →	38	S	73	θ	
	39	T	18	F	
	40	A	14	D	round off $(z_a)_0$ to 28 binary places
	41	A	64$\pi\theta$		
	42	T	14	D	
	43	A	16	D	
	44	A	64$\pi\theta$		
	45	T	16	D	
	46	H	66$\pi\theta$		assemble $(y_a)_0$ with least significant half of p_a and transfer to store
	47	C	16	D	
	48	H	68$\pi\theta$		
	49	C	18	D	
	50	(T		D)	

Adjust representation of z_a by multiplying $(z_a)_0$ by 10 and decreasing p_a by 1. This cycle is repeated until either $\lvert(z_a)_0\rvert > 0.4$ or 10 adjustments have been made (to provide for $(z_a)_0 = 0$).

if $p_a \geq 2047$ stop machine. If $p_a < -2047$ replace p_a by -2047

	51	A	18	D	shift p_a so that its most significant half is at right-hand end of 18D
	52	R	16	F	
	53	T	18	D	
	54	C	18	D	assemble $(x_a)_0$ with most significant half of p_a and transfer to store
	55	H	66$\pi\theta$		
	56	C	14	D	
	57	(T		D)	
	58	T	14	D	clear "complex floating accumulator," i.e., put $z_a = 0.2^{-2047}$
	59	T	16	D	
	60	S	73	θ	
	61	T	18	F	
	62	(E		F)	link order
	63	P		F	dummy order
	64	P	16	F	
	65	P		F	
Temporary	66	S	2	F	form and plant C(66$\pi\theta$)
	67	R	1024	F	
	68	T	66$\pi\theta$		
	69	T	68$\pi\theta$		
	70	E	25	F	these orders are obeyed once during input of tape and then written over
		E	66	Z	final C(66$\pi\theta$) = -2^{-28}
		P		D	
		T	68	Z	
	68	P	31	D	
	69	P		F	
	70	π		F	11/16
	71	J		F	10/16
	72	P	2	F	
	73	P	1023	D	
	74	P	2047	F	
	75	P		D	
	76	M	81	D	1/100 (rounded down)
	77	P	327	D	

A10 Print single floating decimal number (used with A11).

		G		K	
enter	←0	O	55	θ	carriage return
	1	O	56	θ	line feed
	←2	A	3	F	plant link
	3	T	54	θ	
	4	S		D	separate numerical part x (negatively) and exponent p
	5	L	32	F	
	6	U	4	D	
	7	R	32	F	
	8	A		D	
	9	R		D	
	10	E	15	θ	← accumulator contains 2^{-7} p

11	O	2	θ	print "-"
12	T		D	modulus
13	S		D	modulus
14	E	16	θ	p<0
10→15	O	57	θ	print space (p≥0)
14,17-16	A	60	πθ	add $-10.2^{-7}+2^{-25}$
17	E	16	θ	divide by 10 (subtract and count)
18	S	60	θ	divide by 10 (subtract and count)
19	L	2	F	divide by 10 (subtract and count)
20	T		D	divide by 10 (subtract and count)
21	O		F	print tens digit of p
22	O	1	F	print units digit of p
23	O	57	θ	spaces
24	O	57	θ	spaces
25	S	4	D	
26	E	31	θ	accumulator contains $2^{-4}x$
27	T		D	modulus
28	S		D	modulus
29	O	2	θ	- sign (x<0)
30	E	32	θ	- sign (x<0)
26→31	O	57	θ	print space (x≥0)
30→32	A	58	πθ	round off and test for x = 10
33	G	35	θ	arrange to print # if
34	A	43	θ	accumulator is positive
33→35	S	59	θ	add 10.2^{-4}
36	T		D	
37	H	59	θ	-10/16 to register
38	A	62	θ	add digit layout parameter
50,53-39	L		D	digit cycle
40	T	4	F	digit cycle
41	O	1	F	digit cycle
42	A		D	digit cycle
43	F		F	digit cycle
44	S		F	digit cycle
45	L	4	F	digit cycle
46	T		D	digit cycle
47	N		D	digit cycle
48	T		D	digit cycle
49	A	4	F	digit cycle
50	E	39	θ	digit cycle
51	O	57	θ	digit layout
52	L		D	digit layout
53	G	39	θ	digit layout
54	(π		F)	figure shift or link
55	θ		F	carriage return
56	Δ		F	line feed
57	φ		F	space
	O	54	Z	figure shift, carriage return and
	P		F	figure shift, carriage return and
	P		F	figure shift, carriage return and line feed during input of tape
	A	1	D	figure shift, carriage return and line feed during input of tape

```
59 | ‖N     F |
   |  T  58 Z | = (-10 + 1/2 · 10^-6) · 2^-4
58 | ‖P 268 D |
   |  T  60 Z |
60 | ‖P 256 F |
61 | ‖C1536 F | = -10·2^-7 + 2^-25
62 | ‖S 128 F | digit layout constant
```

A11 Arithmetical operations on real numbers expressed in floating decimal form.

For representation of numbers see Part II. The number in the floating decimal accumulator (f.d.a.) is here referred to as $y \cdot 10^q$, and the operand as $x \cdot 10^p$.

Parameters:

Preset:

```
H | P s D |
N | P t F | location of numerical exponent part of f.d.a.
```

Preset by subroutine:

```
M | P 103 θ |
Δ | P   θ   |
L | E  56 θ | dynamic stop order
```

```
        | E  69 K |
        | T  9πM  |
  9πM   | 1 71798 69183 F | 1
 11πM   |   17179 86918 F | 10^-1
 13πM   |    1717 98692 F | 10^-2
 15πM   |     171 79869 F | 10^-3
 17πM   |      17 17987 F | 10^-4
 19πM   |       1 71799 F | 10^-5
 21πM   |         17180 F | 10^-6
 23πM   |          1718 π | 10^-7
        | T     Z |
      0 | A  46 θ | ]
94 ——→1 | A   2 F | ] prepare order 3
      2 | T   3 θ | ]
      3 | (H    F) | select parameter
      4 | C   9 M | entire parameter to accumulator
      5 | E  20 θ | jump for E or T parameters
      6 | R 256 F | 10 pl. ] Form switch order specifying
      7 | A     M |        ] address depending on parameter
      8 | T  19 θ |        ] function
      9 | C   1 M | ] form A order specifying ]
     10 | T  11 θ | ] same address as parameter ]
     11 | (A    F) | select operand ]
     12 | U  22 θ | store top of operand ] A, B, and V parameters: unpack operand
     13 | L  32 F | ] remove exponent p ]
     14 | R  32 F | ]                   ]
     15 | U     D | numerical part x·2^-10 to 0D ]
```

16	S	22	θ	-2^{-14} p to 22θ
17	R	64	F	
18	T	22	θ	
19	(E		F)	*
5→20	U	22	θ	parameter to 22θ and 64θ
21	T	64	θ	
22	(E		F)	**
23	A		H	
24	G	30	θ	if number in f.d.a. is positive, change sign and set order 63 to A H
25	T		H	
26	A	23	θ	
27	T	63	θ	
28	S		H	
29	E	61	θ	test if f.d.a. contains zero
24→30	A	2	M	form (10-y) · 2^{-11}
31	E	49	θ	
32	H	11πM		1/10 to register
39→33	T		H	(10-y) · 2^{-11}
34	A	5	M	adjust exponent
35	A		N	
36	T		N	
37	V		H	cycle to multiply y by negative power of 10 if necessary
38	A	4	M	add 9.2^{-11}
39	G	33	θ	
40	E	49	θ	
50→41	T		H	
42	S	5	M	adjust exponent
43	A		N	
44	T		N	
45	A		H	cycle to multiply y by positive power of 10 if necessary
46	L	1	F	x 10
47	A		H	
48	L		D	
31,40-49	S	4	M	sub. $9 \cdot 2^{-11}$
50	E	41	θ	
51	S	3	M	sub. 2^{-11}
52	Y		F	
53	T		H	final value of $-y \cdot 2^{-11}$ to 0H
54	A		N	$q \cdot 2^{-14}$
55	S	6	M	examine exponent
56	P		L	dynamic stop if q≥63
57	A	7	M	
58	E	61	θ	jump if q≥-63
59	T		H	set y = 0 if
60	T		H	q<-63
29,58-61	S	6	M	re-form $q \cdot 2^{-14}$
62	L	64	F	$q \cdot 2^{-6}$
63	(S		H)	add $y \cdot 2^{-11}$ †
64	(T		D)	to store ††

T parameters only

	65	A	28	θ	reset 63θ †
	66	T	63	θ	
	67	S	6	M	set q = -63
	68	T		N	
	69	E	91	θ	
19 →	70	S		D	change sign of x if to be added
	71	T		D	
19 →	72	H		D	subtrahend to register
	73	A		N	form 2^{-14} (q-p)
	74	A	22	θ	
	75	E	84	θ	jump if q≥p
19 →	76	E	95	θ	†††
	77	T	64	θ	-2^{-14} (q-p) to 64
	78	H		H	interchange numerical parts
	79	A		D	
	80	T		H	
	81	S	22	θ	larger exponent to 0N
	82	T		N	
	83	S	64	θ	2^{-14}(q-p) to Acc.
75 →	84	S	3	M	jump if smaller number is negligible
	85	E	92	θ	
	86	A	8	M	divide numerical part of smaller number by appropriate power of 10
	87	T	88	θ	
	88	(V		D)	
	89	A		H	combine numbers
101,102 →	90	Y		F	
69 →	91	T		H	sum of difference to 0H
85 →	92	T	64	θ	clear accumulator
	93	A	3	θ	prepare to change order 3
	94	G	1	θ	
76 →	95	S	22	θ	add exponents
	96	A		N	
	97	T		N	
	98	H		D	multiply numerical parts
	99	V		H	
	100	L	512	F	
	101	E	90	θ	
	102	G	90	θ	
M	0	E	78	θ	
	1	A	1023	D	
	2	P	160	F	$= 10.2^{-11}$
	3	P	16	F	$= 2^{-11}$
	4	P	144	F	$= 9.2^{-11}$
	5	P	2	F	$= 2^{-14}$
	6	P	126	F	$= 63.2^{-14}$
	7	P	252	F	$= 126.2^{-14}$
	8	V	25πM		
		T	128	Z	

A and B parameters only

larger number to f.d.a.

V parameters only

* Order 19 switches as follows:
if parameter function is A to 70 θ
" " " " B to 72 θ
" " " " V to 76 θ

** 22θ contains the parameter itself if the function is E or T (in the latter case the order plays no part in the calculation), otherwise 22θ is used to store $-p \cdot 2^{-14}$.

† Order 63 is always S H unless a T parameter is being obeyed and y is negative, in which case order 63 becomes A H.

†† 64θ holds the parameter itself if the function is E or T (in the former case 64θ is not encountered by control); when dealing with A and B parameters, 64θ is used to hold $-|q-p| \cdot 2^{-14}$. For all parameters (except E), 64θ is used as a "dump" by order 92 to clear the accumulator.

††† If control reaches order 76 from order 75, C(Acc.) must be <0 and control proceeds to 77. If 76 is reached from 19, C(Acc.) must be =0 and control is switched to 95.

B2 Complex operation No. 2.

Performs operations (including multiplication) on complex numbers. Uses as "multiplier register" H (real part) and 2H (imaginary part).

	46F	P	47 θ	N parameter
		T	50πZ	These orders do not go into the store but
		P	F	merely serve to clear 50$\pi\theta$ to ensure that the "sandwich digit" is zero when the constants PD and PF appearing at the end of the subroutine, are planted there.
		T	Z	
	0	A	N	forms A n+2 F
39→	1	T	2 θ	
	2	(A	F)	
	3	U	16 θ	plants order to be obeyed in 16 and in 26
	4	U	26 θ	
	5	A	N	increases address of order by 2
	6	U	30 θ	
	7	G	40 θ	
	8	T	F	clears accumulator
	9	A	17 θ	places ineffective order in 18, 26 for operations other than V or N
	10	U	26 θ	
46→	11	T	18 θ	
	12	A	4 H	multiply by 2^{-34}. "unpacks" real accumulator
	13	H	3πN	
	14	V	6 H	
	15	H	H	operation on real part
	16	(V	F)	
	17	H	2 H	
	18	(N	F)	

	19	U	4	H	"packs" real accumulator
	20	S	4	H	
	21	L		F	
	22	L		F	
	23	L	64	F	
	24	T	6	H	
	25	A	8	H	"unpacks" imaginary accumulator and performs operations
	26	(V		F)	
	27	H	3π	N	
	28	V	J	H	
	29	H		H	
	30	(V		F)	
	31	U	8	H	"packs" imaginary accumulator
	32	S	8	H	
	33	L		F	
	34	L		F	
	35	L	64	F	
	36	T	J	H	
	37	A	2	F	
	38	A	2	θ	
	39	G	1	θ	
T→	40	A	1	N	test for V
	41	E	45	θ	
	42	A	2	N	test for N
	43	E	8	θ	
	44	A	5	N	
41→	45	S	5	N	
	46	G	11	θ	
N	0	P	2	F	
	1	Q		F	
	2	I		F	
	3	P		D	
	4	P		F	
	5	J		F	

C7 Check function letters, with localized print suppression.

	T		Z
0	(Δ		F)
1	(P		F)
2	Q		F
3	A		F
4	θ		F
5	Δ		F
6	π		F
7	K	3000	F
8	P		H
9	P		N
10	P		M

33 →	11	U 26 θ	store address of C.O.	Transfer control
	12	S 8 θ	test for change of mode	
	13	E 15 θ		
	14	A 9 θ		
13 →	15	S J θ		
	16	(E 46 Δ)*		
	17	E 20 Δ		
58 →	18	O 4 θ	new line	
	19	O 5 θ		
17 →	20	U 37 θ	clear top of accumulator	
	21	S 37 θ		
	22	A 3 θ		
45 →	23	A 26 θ		Checking cycle, similar to that employed in C11
	24	U 26 θ		
	25	S 26 θ		
	26	(A F)	S.O.	
	27	U 37 θ		
Enter →	28	A θ		
	29	S 3 θ		
	30	(O 37 θ)	becomes E 34 θ for suppression	
	31	E 34 θ		
	32	A 2 θ		
30	33	E 11 θ		
31 →	34	U θ		
	35	S θ		
	36	A 1 θ		
	37	(K 3000 F)	C.O.	
	38	U 1 θ		
	39	G 41 θ		
	40	A 5 θ		
39 →	41	S 1 θ		
	42	U θ		
	43	S θ		
	44	A 2 F		
	45	E 23 θ		
16 →	46	O 6 θ	figure shift	Change of mode of operation from printing to suppressed or vice versa
	47	E 49 Δ		
16 →	48	O 7 θ	letter shift	
47 →	49	U 37 θ		
	50	S 37 θ		
	51	S 16 θ		
	52	A 59 θ		
	53	U 16 θ		
	54	S 16 θ		
	55	S 30 θ		
	56	A 60 θ		
	57	U 30 θ		
	58	E 18 θ		

	G Z	
59	C 35 θ	= C 94 θθ
60	S 12 θ	= S 71 θθ
	G K	
	W2015 Z	= E 28 Z: stops reading of tape and directs control to order 28 with E L in the accumulator.
	E L	

* Order 16 takes the following forms:

	Printing		Suppressed
Print low	E 46 θ	⟷	G 48 θ
Print high	G 46 θ	⟷	E 48 θ

C10 Numerical check, ignoring closed subroutines; will print C(Acc.) before obeying T orders.

Note: Code letter H refers to locations in the first part of the subroutine and θ to locations in the second part.

		E 25 K		
		T H		
H	0	A 3 F		
	1	T F	dummy print routine	
	2	E F		
10H →	3	O 2 θ	print +	
	4	E 14 H		
31θ →	5	S 6 θ	form A p F/D if order T p F/D is encountered	
	6	E 32 θ		
	7	S 2 H		
	8	T 9 H		
	9	(π F)	becomes A p F/D	
	10	E 3 H	test sign	
	11	T πθ	change sign	
	12	S πθ		
	13	O H	print -	
4H →	14	T πθ		
	15	S 33 H	set digit count in 9 H	
30H →	16	A 2 F		
	17	T 9 H		
	18	A πθ	multiply by 10/16	Print number transferred by T order
	19	R 1 F		
	20	S πθ		
	21	R D		
	22	A πθ		
	23	U θ	print	
	24	O θ		
	25	F θ		
	26	S θ		
	27	L 4 F		
	28	T πθ		

	29	A 9 H	digit count	Test for entry to closed subroutines and obey them directly
	30	G 16 H		
	31	T $\pi\theta$	clear accumulator	
	32	E 34 θ	to sequence control	
	33	‖P N	number of digits	
16θ →	34	A 2 H	test for order A n F in S(n), i.e., S.O. = C.O.	
	35	S 12 θ		
	36	G 19 θ		
	37	S 2 F		
	38	E 19 θ		
	39	O 3 θ	line feed	
	40	A 20 θ	form A n+2 F	
	41	A 12 θ		
	42	S 26 θ		
	43	U 12 θ		
	44	U 47 H		
	45	S 5 θ	form G n+1 F	
	46	T 50 H		
	47	(P F)	= C(Acc.) or A n+2 when subroutine is encountered	
	48	T 22 θ		
	49	A 40 H		
	50	(P F)	sign of C(Acc.) or G n+1 F when subroutine is encountered	

When an order A n F is encountered in n, the order in (n+2) is placed in the C.O. position and control is transferred to (n+1) with A 20 θ in the accumulator. Since there is a G order in (n+1) control is transferred to the subroutine and the link which is planted in the subroutine is E 22 θ (or E 23 θ if the subroutine has one program parameter). When the operation of the subroutine is finished control is transferred to order 22θ (or 23θ) of C10 and checking recommenced.

		T Z		
θ	0	‖(P F)	working space for print cycle	
	1	(P F)		
	2	Z F		
	3	Δ F		
	4	θ F		
	5	Q 1 F		
	6	Q F		
	7	A M	extracts order at which checking starts and replaces it by order directing control to C10 (order 21θ)	
	8	T 47 H		
	9	A 15 θ		
	10	T M		
	11	O 4 θ	carriage return	
	12	O 3 θ	line feed	
	13	O 9 H	figure shift	
	14	E 25 F		
	15	‖E 21 θ		
		E 7 Z		
		P F		

The orders 7 to 14 are executed once during input, and then written over by:

		T	7	Z	
18θ →	7	O	4	θ	carriage return
	8	O	3	θ	line feed
	9	S	2	H	form A n F when control is transferred
36θ →	10	U	12	θ	
	11	S	12	θ	
	12	(G	2047	M)	= A -1 M, becomes select order (S.O.)
	13	U	22	θ	
	14	A	50	H	test for transfer of control
	15	S	2	H	
	16	G	34	H	
	17	S	6	θ	
	18	G	7	θ	
36H, 38H →	19	U	50	H	
	20	S	50	H	
Enter →	21	A	47	H	Add "C(Acc.)"
	22	(T		M)	current order (C.O.)
	23	U	47	H	transfer "C(Acc.)"
	24	E	26	θ	test C(Acc.) for sign, if - send 1/2 to 50H
	25	S	3	θ	
24θ →	26	S	47	H	
	27	U	50	H	
	28	S	50	H	
	29	A	22	θ	examine C.O. and test for T order
	30	S	1	H	
	31	E	5	H	
6H →	32	U	22	θ	
	33	S	22	θ	
32H →	34	A	12	θ	sequence control
	35	A	2	F	
	36	G	10	θ	

Checking cycle similar to that employed in C11.

During the course of this subroutine the 17 most significant digits of C(Acc.) are stored in 47 H and are restored when an order from the original program is executed.

C11 Check function letters.

		G		K	
	0	(P		F)	sign of C(A)*
	1	(P		F)	C(A) stored
	2	θ		F	
	3	Δ		F	
	4	A		F	
	5	Q		F	
18 →	6	A	4	θ	form new S.O.
	7	O	2	θ	new line
	8	O	3	θ	
31 →	9	U	11	θ	plant S.O.
	10	S	11	θ	

transfer control

11	(Z		F)	select order (S.O.)
12	U	22	θ	plant C.O.
Enter → 13	O	22	θ	print function***
14	S		θ	subtract ΔF if C(A)<0*
15	A	4	θ	⎫
16	E	19	θ	⎬ test for transfer of control**
17	A	5	θ	⎪
18	E	6	θ	⎭
16 → 19	U		θ	⎫ clear top of accumulator
20	S		θ	⎭
21	A	1	θ	restore C(A)
22	(K	3000	F)	current order (C.O.)
23	U	1	θ	store C(A)
24	E	26	θ	⎫
25	A	3	θ	⎬ store sign of C(A)* and clear
24 → 26	S	1	θ	⎪ top of accumulator
27	U		θ	⎪
28	S		θ	⎭
29	A	11	θ	⎫
30	A	2	F	⎬ advance address specified
31	G	9	θ	⎭ in S.O.
	E	13	Z	

Followed on tape by

E	m	F	punched by user. Hence control enters at order 13, with E m F in accumulator

Notes: C(A) refers to the 17 most significant digits which would be in the accumulator if the original program were operating directly.

C.O. = current order, the order in the original program which is being dealt with.

S.O. = select order, the A order which selects the current order from the original program.

* sign of C(A) is stored in θ coded thus: ΔF for negative
PF for positive or zero

** after executing order 14θ, the function digits in the accumulator represent E if and only if a transfer of control is to occur.

*** on entry, order 13θ causes a letter shift.

C12 Check function letters, with dummy print routine and delayed start.

Places "blocking order" in h and commences checking when blocking order is obeyed for the nth time.

	T		Z	
0	A	3	F	⎫
1	T		F	⎬ dummy print routine
2	E		F	⎭
3	(E	6	θ)	(1) blocking order (2) counter (3) -2^{-15}
4	θ		F	

	5	‖Δ		F	
H →	6	(A	3	θ)	(1) A 3 θ (2) sign of C(Acc.)*
	7	A	2	F	count
	8	E	28	θ	
	9	T	3	θ	
	10	(K	3000	F)	(1) letter shift (2) order from H (3) store C(Acc.)
	11	E	1	H	return to 1H
	12	G	1	H	
25 →	13	S	2	θ	
	14	O	4	θ	
	15	O	5	θ	
	16	O	J	θ	temporary, during input
	17	A		H	
	18	T	J	θ	
	19	A	3	θ	
	20	T		H	
	21	S	46	F	
	22	T	3	θ	
	23	E	34	F	
		E	14	Z	
		P		F	
		T	16	Z	
38 →	16	U	18	θ	checking cycle as in C11
	17	S	18	θ	
	18	(G	2047	H)	S.O.
	19	U	29	θ	
	20	S	6	θ	
	21	S	2	θ	
	22	O	29	θ	
	23	G	26	θ	
	24	S	39	θ	
	25	G	13	θ	
23 →	26	U	6	θ	
	27	S	6	θ	
8 →	28	A	J	θ	
	29	(T		H)	C.O.
	30	U	J	θ	
	31	E	33	θ	
	32	A	5	θ	
31 →	33	S	J	θ	
	34	U	6	θ	
	35	S	6	θ	
	36	A	18	θ	
	37	S	3	θ	
	38	G	16	θ	
	39	‖Q		F	

* sign of C(Acc.) stored thus: PF = positive or zero
ΔF = negative

D4 Division, small; positive divisor.

Repetitive process: $a_{n+1} = -a_n c_n + a_n$ a_0 = dividend
$c_{n+1} = -c_n^2$ $c_0 + 1$ = divisor

Stop when $c_n = 0$.

	T		Z	
0	S	2	θ	subtract 1
1	A	3	θ	subtract 1
10 → 2	T		D	c_n
3	H		D	a_{n+1}
4	N		H	a_{n+1}
5	A		H	a_{n+1}
6	Y		F	a_{n+1}
7	T		H	a_{n+1}
8	N		D	c_{n+1}
9	Y		F	c_{n+1}
10	G	2	θ	test for $c_{n+1} = 0$

repetitive cycle

D6 Division, accurate, fast.

C(0D)/C(4D) to 0D.

$a_{n+1} = a_n - c_{n+1}a_n + c_{n+1}$

$c_{n+1} = -a_n b + (b-1)$, where b is the shifted divisor

$i - a_n \rightarrow 1/b$

$c_n \rightarrow 0$ a_n and c_n are negative

$a_0 = 2b - 2\sqrt{2} + 1$; therefore c_n is negative until process is completed

	G		K	
0	A	3	F	plant link
1	T	34	θ	plant link
7 → 2	S	4	D	make divisor positive and change sign of quotient
3	E	13	θ	make divisor positive and change sign of quotient
4	T	4	D	make divisor positive and change sign of quotient
5	S		D	make divisor positive and change sign of quotient
6	T		D	make divisor positive and change sign of quotient
7	E	2	θ	make divisor positive and change sign of quotient
14 → 8	T	4	D	shift divisor and dividend until divisor exceeds capacity
9	A		D	shift divisor and dividend until divisor exceeds capacity
10	L		D	shift divisor and dividend until divisor exceeds capacity
11	T		D	shift divisor and dividend until divisor exceeds capacity
12	A	4	D	shift divisor and dividend until divisor exceeds capacity
3 → 13	L		D	shift divisor and dividend until divisor exceeds capacity
14	E	8	θ	shift divisor and dividend until divisor exceeds capacity
15	R		D	
16	U	4	D	b-1 to 4D
17	L		D	
18	A	35	θ	
19	T	6	D	a_0 to 6D
20	E	25	θ	

30 →	21	U	8	D	c_{n+1} to 8D
	22	N	8	D	$-c_{n+1} \cdot a_n$
	23	A	6	D	$+a_n$
	24	T	6	D	$+a_{n+1}$ to 6D
20 →	25	H	6	D	a_{n+1} to multiplier register
	26	S	6	D	a_n
	27	N	4	D	$-(b-1) \cdot a_n$
	28	A	4	D	$+(b-1)$
	29	Y		F	
	30	G	21	θ	test ← accumulator contains 2^{-34}
	31	S		D	] form quotient
	32	V		D	]
	33	T		D	
	34	(E		F)	link
	35	‖W	1526	D	$3 - 2\sqrt{2}$

E2 Exponential (slow).

(e^x-1) to 4D, where x = C(4D)

Uses a recurrence relation $z_{n-1} = z_n + \frac{z_n^2}{2^{n+1}}$ starting with $z_{33} = x$ and ending with $z_0 = (e^x-1)$

		G		K	
	0	A	3	F	] plant link
	1	T	18	θ	]
	2	Y		F	]
	3	L		D	] 2^{-n} to 6D.
16 →	4	T	6	D	]
	5	H	4	D	]
	6	V	4	D	] form z_n^2
	7	T		D	]
	8	H	6	D	
	9	V		D	$z_n^2/2^n$
	10	R		D	$z^2/2^{n+1}$
	11	A	4	D	] $z_n + z_n^2/2^{n+1}$
	12	Y		F	]
	13	T	4	D	z_{n-1} to 4D
	14	A	6	D	shift strobe
	15	L		D	
	16	E	4	θ	test strobe for end of cycle
	17	T		D	
	18	(E		F)	link

E4 Exponential, fast. Exp C(R) to 0D

G		K
E	69	K
T	$18\pi\theta$	

calls in R9

```
18πθ        2 60054 F ┐
20πθ       31 23906 F │
22πθ      235 85378 F │
24πθ     1430 07273 F │
26πθ     7157 73946 F │  coefficients in power series
28πθ    28633 01149 F │
30πθ    85899 33588 F │
32πθ  1 71798 69147 F │
34πθ  1 71798 69184 π ┘
        T     Z
   0    A   3 F   ┐
   1    T  14 θ   ┘ plant link
   2    V  18πθ   ┐
   3    A  20πθ   ┘ form a8 + a9x
   4    T     D
   5    S  17 θ     set power series counter
13→6    A  16 θ   ┐
   7    T   9 θ   │
   8    V     D   │
   9   (A     D)  ┘ form a1 + a2x + ...
  10    T     D
  11    A   9 θ
  12    S  15 θ
  13    G   6 θ
  14   (E     F)    link
  15    A  34πθ
  16    A  36πθ
  17    P  14 F
        T  36 Z
```

F1 Interpolation.

The subroutine places f $[2^n C(4D)]$ in 10D. The process used is that described in Milne's Numerical Calculus, p. 72, and known as Neville's method.

```
48F  (P     F)   (becomes P 2b πθ)   Δ parameter
49F  (P     F)   (becomes P 2b D)     L parameter
50F   P  61 θ                        X parameter

        T     Z
   0    A  45 F
   1    U  49 F   send P 2b D to 49
   2    A  42 F
   3    T  48 F   send P 2b πθ to 48
   4    S   2 F
   5    R     M
   6    L  64 F
   7    L  32 F
   8    T  45 F   send -2^-n to 45
   9    A  46 F
```

10	R		M	
11	L	64	F	
12	L	32	F	
13	T	46	F	send $a2^{-n}$ to 46
14	E	25	F	
	E		Z	
	P		F	

This sequence of orders calculates the parameters required by the subroutine: these orders are written over by the orders following:

		T		Z	
9 ⟶	0	T	20	F	] partial sum times 10
	1	V		D	
	2	L	8	F	
	3	S	40	D	subtract new digit
14 ⟶	4	U		D	
	5	(T		F)	= T 66 Δ
	6	I	40	F	] read next symbol
	7	S	40	F	
	8	A	39	F	
	9	E		θ	test for F
	10	A	2	F	
	11	E	25	F	test for Z
	12	A	5	θ	change destination order
Enter ⟶	13	T	5	θ	
	14	E	4	θ	
		E	13	Z	
		T	66	Δ	

66Δ	1	71798	69184 F
64Δ		85899	34592 F
62Δ		57266	23061 F
60Δ		42949	67296 F
58Δ		34359	73837 F
56Δ		28633	11531 F
54Δ		24542	67026 F
52Δ		21474	83648 F
50Δ		19088	74354 F
48Δ		17179	86918 Z

These orders place -1, - 1/2, ... - 1/10 in positions 66Δ, 64Δ, ..., etc., and are then written over by what follows

	T		Z	
0	A	3	X	
1	U	11	θ	form A p+2 F
2	A	7	θ	
3	T	60	θ	form E p+3 F
4	H	1	X	] collate "integral" part of C(4D)
5	C	4	D	

	6	S	2	X	subtract a
	7	U	1	F	
	8	R	32	F	form address of first entry
	9	R	32	F	of the table used.
	10	L		M	
	11	(A		F)	= A p+2 F
	12	T	16	θ	
	13	A	4	D	
	14	S	1	F	
	15	T	4	D	
	16	(A		F)	add first entry used
	17	T	10	D	transfer to 10D
	18	A	16$\pi\theta$		
	19	E	21	θ	
58 →	20	A		X	
19 →	21	A	3π	X	
	22	T	26$\pi\theta$		
	23	A	4	D	modify argument
	24	A	1	X	
	25	T	4	D	
	26	(A		D)	add next entry of table required
	27	(T		D)	
	28	A	27	θ	form orders required in locations 40θ, 41θ, 49θ, 50θ, and 52θ for successive linear interpolation.
	29	S	59	θ	
55 →	30	A	17	θ	
	31	U	52	θ	
	32	A	6	X	
	33	U	41	θ	
	34	A	7	X	
	35	U	40	θ	
	36	U	50	θ	
	37	S	27	θ	
	38	A	8	X	
	39	T	49	θ	
	40	(A		D)	linear interpolation
	41	(S		D)	
	42	T		D	
	43	H		D	
	44	V	4	D	
	45	L		M	
	46	Y		F	
	47	T		D	
	48	H		D	
	49	(N		D)	
	50	(A		D)	
	51	Y		F	
	52	(T		D)	

```
        53 |  A  52 θ |
        54 |  S  59 θ |
        55 |  E  30 θ |   count number of interpolations
        56 |  A 26πθ  |
        57 |  S   5 X |
        58 |  G  20 θ |
        59 |  T  12 D |   clear accumulator
        60 | (E    F) |   link
X       0  |‖T   8 L |
        1  |‖P     H |   = -2^-n
        2  |‖P     N |   = a2^-n
           |  T 64πZ  |
        4  |‖P   2 F  |
           |  T  64 Z |
        3  |‖P   2 F  |
           |  T  66 Z |
        5  |‖T   6 L  |
        6  |‖U     F  |
        7  |‖K4098 F  |
        8  |‖V  66 Δ  |
           |  E  25 K |
           |  T 67πΔ  |
```

The subroutine is of variable length. The number of reciprocals required depends on the number of entries used in the interpolation. Their position is so arranged that those not required are written over by the orders of the subroutine.

F2 Solution of f(x) = 0, or inverse interpolation (second-order process).

Working space is allocated thus:

hD	x_c
(h+2)D	x_a
(h+4)D	x_b
(h+6)D	$-f(x_a) = -f_a$ (say)
(h+8)D	$-f(x_b)$ or $-2^{-m}f(x_b) = -F_b$ (say)

x_a and x_b are two values of x such that f_a and f_b have opposite signs. x_c is a value obtained by linear inverse interpolation between x_a and x_b. The auxiliary routine places f_c in 0D. If the sign of f_c is opposite to that of f_a, then F_b is replaced by f_a and f_a by f_c, also x_b by x_a and x_a by x_c. If the sign of f_c is the same as that of f_a, then f_a is replaced by f_c and x_a by x_c, and also F_b is halved. This latter operation prevents x_b remaining unaltered for many cycles, as this would cause the process to become a first-order one, or fail to converge altogether. At the start x_a and x_c are the given values x_1 and x_2, and the f_a position is cleared. This ensures that initially f_a and f_c are treated as of opposite sign, and the first two function-values to be calculated are $f(x_1)$ and $f(x_2)$. The process terminates either when $f_c = 0$ or when $|x_a - x_b| \leq 2^{-34}$

	47F	Δ		F	M parameter
		T		Z	
	0	A	3	F	plant link
	1	T	7	θ	
	2	T	6	H	clear (h+6)D
56 →	3	A	3	θ	call in auxiliary subroutine
	4	G		N	
Aux →	5	H		D	test whether $f_c = 0$
	6	N		D	
54 →	7	(Z		F)	link
	8	T	4	D	clear accumulator
	9	V	6	H	test relative sign of f_a and f_c
	10	G	17	θ	
	11	T	4	D	clear accumulator
	12	A	2	H	x_a to x_b
	13	T	4	H	
	14	A	6	H	f_a to F_b
	15	T	8	H	
	16	E	21	θ	opposite signs
10 →	17	T	4	D	clear accumulator
	18	A	8	H	halve F_b
	19	R		D	
	20	T	8	H	same sign
16 →	21	S		D	f_c to f_a
	22	U	6	H	
	23	S	8	H	form $\frac{f_c}{f_a - f_c}$ in 0D by division process similar to that used in D7
	24	E	29	θ	
	25	T	4	D	
	26	S		D	
	27	T		D	
	28	S	4	D	
24 →	29	A	57	θ	
38 →	30	T	4	D	
	31	Y		F	
	32	H	4	D	
	33	N		D	
	34	A		D	
	35	T		D	
	36	N	4	D	
	37	Y		F	
	38	G	30	θ	
	39	H		D	
	40	A		H	plant new x_a
	41	U	2	H	
	42	S	4	H	$(x_a - x_b)$ to 0D
	43	T		D	
	44	A		H	new x_c
	45	V		D	
	46	Y		F	
	47	T		H	

	48	A		D	test for $\lvert x_a - x_b \rvert \leq 2^{-34}$
	49	G	52	θ	
	50	T		D	
	51	S		D	
49 →	52	R		D	
	53	Y		F	
	54	E	7	θ	
	55	T		D	clear accumulator
	56	E	3	θ	repeat
	57	‖Δ		M	= -1

F3 Differencing and checking subroutine No. 1.

	T		Z	
0	A	3	F	plant link
1	T	24	θ	
2	A		D	form fourth difference
3	S		H	
4	S	2	H	
5	S	4	H	
6	S	6	H	
7	U	8	H	plant differences and new function value
8	A	6	H	
9	U	6	H	
10	A	4	H	
11	U	4	H	
12	A	2	H	
13	U	2	H	
14	A		H	
15	T		H	
16	A	8	H	test fourth difference
17	G	20	θ	
18	T	4	D	
19	S	4	D	
17 → 20	A	10	H	
21	E	23	θ	
22	O	25	θ	
21 → 23	T	4	D	
24	(Z		F)	link
25	‖B		F	

G1 Simultaneous first-order differential equations by modified Runge-Kutta process; single step.

	T		4πZ
5	‖H	682	D
	T		6πZ
7	‖P		N
	T		12πZ
13	‖ϕ	1405	D

```
                T  14πZ
          15    T     H
                T  16πZ
          17    T   2 H
                T     Z
           0    A   3 F     ] plant link
           1    T  61 θ     ]
           2    A  31 θ     set count = A 8 θ
           3    G  63 θ
           4    Δ     F     = -1/2 ]  = -2/3
                T   6 Z           ]
           6    P     N
                T   8 Z
           8    M     M
           9    O     Δ
Aux ──→   10    H   4 θ     enter for first stage
          11    A  20 θ
          12    (E 23 θ)    ] ±√(1/2)
                T  14 Z     ]
Aux ──→   14    A     H     enter for second stage
                T  16 Z
          16    A   2 H
                T  18 Z
Aux ──→   18    H  12πθ     enter for third stage
          19    S  12πθ
          20    T  12πθ
          21    E  28 θ
Aux ──→   22    H   4πθ     enter for fourth stage
12  ──→   23    T   4 D     clear 4D or accumulator
          24    U     F
          25    S  38 θ     ]
          26    A  25 θ     ] switch order 38*
          27    T  38 θ     ]
21  ──→   28    S   6πθ
58  ──→   29    A  16πθ     ]
          30    U  46πθ     ]
          31    A   8 θ     ]
          32    U  37 θ     ] plant variable orders
          33    A   9 θ     ]
          34    U  55 θ     ]
          35    A  24 θ     ]
          36    T  39 θ     ]
          37    (Z     F)   add 2^m hy'
          38    (R1057πθ)   *
          39    (Z     F)   subtract 2^m q
          40    Y     F     ]
          41    U   6 D     ]
          42    V   6 D     ] form r
          43    R     L     ]
          44    Y     F     ]
```

cycle dealing with each variable in turn

	45	U	D	store r
	46	(Z	F)	add old y
	47	(Z	F)	plant new y
	48	A	D	
	49	L	D	
	50	A	D	
	51	L	L	form new $2^m q$
	52	S 6	D	
	53	N 4	D	
	54	Y	F	
	55	(Z	F)	plant $2^m q$
	56	A	$46\pi\theta$	
	57	S	$14\pi\theta$	test for last variable
	58	G	29 θ	
	59	A	65 θ	
	60	S	11 θ	
	61	(Z	F)	link: tests for end of step
	62	A	35 θ	
3 →	63	U	65 θ	
	64	G	X	to auxiliary subroutine
	65	‖(Z	F)	count

* Order 38 is switched to U 1028 D (equivalent to U 4 D) for stages 1, 2, and 3 and back to R $1057\pi\theta$ (equivalent to R D) for stage 4. During stage 4, 4D remains empty.

The register contains $-1/2$ during stage 1, $-\sqrt{1/2}$ during stage 2, $+\sqrt{1/2}$ during stage 3, and $-2/3$ during stage 4. At each stage, the cycle of orders 29 to 58 is performed n times.

G3 Integration of $y'' = f(x,y)$ by 5th order process.

$$y_2 = y_1 + \delta y_{1/2} + (y_2'' + 1/12\,\delta^2 y_1')h^2 \qquad (1)$$

$$y_2'' = f(x_2, y_2) \qquad (2)$$

First, (1) is used with y_1'' in place of y_2'' and $\delta^2 y_0''$ in place of $\delta^2 y_1''$. The value of y_2 ($= [y_2]_a$, say) obtained is used in (2) to get $[y_2'']_a$, from which a value of $\delta^2 y_1'' = [\delta^2 y_1'']_a$ can be obtained. $[y_2'']_a$ and $[\delta^2 y_1'']_a$ are then used in (1) to get a new value of $y_2 = [y_2]_b$ and so on. The process is continued until two consecutive values of $\delta^2 y_1''$ differ by 2^{-31} or less.

		T	Z	
	0	A 3	F	plant link
	1	T 44	θ	
		G	K	
	0	A 12	H	
	1	A J	H	increase x_1 to x_2
	2	T J	H	

31 →	3	H	16	H	$(y''_0 + 1/12\ \delta^2 y''_1)$ to 0D
	4	V	8	H	
	5	A	4	H	
	6	Y		F	
	7	T		D	
	8	H		D	
	9	V	14	H	
	10	Y		F	
	11	A	2	H	
	12	U	J	D	provisional value of $\delta y_{1\frac{1}{2}}$ to 10D
	13	A		H	
	14	T	18	H	y_2 to 18H
	15	A	15	θ	call in auxiliary subroutine
	16	G		N	
Aux →	17	A	20	H	
	18	S	4	H	
	19	S	6	H	
	20	U	4	D	trial value of $\delta^2 y''_1$ to 4D
	21	S	8	H	
	22	G	25	θ	
	23	T		D	
	24	S		D	
22 →	25	R	4	F	
	26	Y		F	
	27	E	32	θ	test difference in consecutive values of $\delta^2 y''_1$
	28	T		D	
	29	A	4	D	
	30	T	8	H	repeat cycle with new value of $\delta^2 y''_1$
	31	E	3	θ	
27 →	32	A	4	D	
	33	U	8	H	
	34	A	6	H	
	35	U	6	H	
	36	A	4	H	
	37	T	4	H	set new values of operands
	38	A	J	D	
	39	U	2	H	
	40	A		H	
	41	T		H	
	42	(Z		F)	link

J1 Calculation of Legendre polynomials.

Uses a recurrence relation giving 0.5 P_n in terms of 0.5 P_{n-1} and 0.5 P_{n-2} which are stored in 4H and 6H respectively. (4H and 6H are used as working space.)

	47F	P	68	N	M parameter
		T		Z	
9 →	0	T	20	F	
	1	V		D	

```
          2 |  L   8  F |
          3 |  S  40  D |
14 ------>4 |  U      D |
          5 |  (T     F) |   This section reads the numbers following
          6 |  I  40  F |     as in the first section of subroutine F1
          7 |  S  40  F |
          8 |  A  39  F |
          9 |  E      θ |
         10 |  A   2  F |
         11 |  E  25  F |
         12 |  A   5  θ |
         13 |  T   5  θ |
         14 |  E   4  θ |_____
            |  E  13  Z |
            |  T      M |
          1 | 71798 69184 F
          1 | 14532 46123 F
            | 85899 39592 F
            | 68719 47674 F
            | 57266 23061 F
            | 49085 34053 F
            | 42949 67296 F
            | 38177 48708 F
            | 34359 73837 Z
```

These orders place -1, -2/3, -2/4, ..., -2/10 in M, M-2, etc., and are then overwritten.

```
            |  T      Z |
          0 |  A   3  F |  ]
          1 |  T  35  θ |  ] plant link
          2 |  A  38  θ |  ]
          3 |  U   4  D |  ] plant 0.5 P0(2x) in 4D and 6H
          4 |  T   6  H |  ]
          5 |  A   6  D |    put 0.5 P1(2x) in 4H
          6 |  T   4  H |
          7 |  A  39  θ |  ]
          8 |  T  20  θ |  ] form multiplier order
          9 |  A  40  θ |  ]
37 ------>10 |  T  24  θ |  ] set transfer order
         11 |  H   6  D |  ]
         12 |  A   6  H |  ]
         13 |  R      D |  ]
         14 |  N   4  H |  ]
         15 |  Y      F |  ]  form 0.5 Pn(2x) from 0.5 Pn-1 and 0.5 Pn-2
         16 |  T      D |  ]    by recurrence relation
         17 |  V   4  H |  ]    0.5 Pn = 4x(0.5 Pn-1) - 0.5 Pn-2
         18 |  L   1  F |  ]           - 2/n · [x · 0.5 Pn-1 - 0.25 Pn-2]
         19 |  S   6  H |  ]
         20 |  (H     D) |  ]
```

```
21 |  N     D  |
22 |  Y     F  |
23 |  U     D  |
24 |  (T    D) |  transfer order
25 |  A  4  H  |
26 |  T  6  H  |  set P n-1, P n-2 in preparation
27 |  A     D  |    for the next cycle
28 |  T  4  H  |
29 |  A 20  θ  |
30 |  S 41  θ  |  modify H order
31 |  T 20  θ  |
32 |  A 24  θ  |
33 |  S  6  θ  |  count and modify transfer order
34 |  G 36  θ  |
35 |  (Z    F) |  link
36 |  A  4  θ  |  count and modify transfer order
37 |  E  J  θ  |
38 |  I     F  |  = 1/2
39 |  H     M  |
40 |  T  8  D  |
41 |  P  2  F  |
   |  E 25  K  |
   |  T 1πM    |
```

K1 Summation of power series.

```
   |  T     Z  |
 0 |  A  3  F  |  plant link
 1 |  T 13  θ  |
 2 |  T     D  |  clear 0D
 3 |  S 14  θ  |  form A m D
12 |  A 16  θ  |
 5 |  T  7  θ  |
 6 |  V     D  |  multiply partially formed polynomial by x
 7 |  (A    D) |  add in next coefficient
 8 |  Y     F  |  round off
 9 |  T     D  |  transfer to 0D
10 |  A  7  θ  |  start to form next value of order 7
11 |  S 15  θ  |
12 |  G  4  θ  |
13 |  (E    F) |  link
14 |  P  2  H  |
15 |  P     N  |
16 |  P  2  N  |
```

K4 Summation of power series.

```
   |  G     K  |
 0 |  A 21  θ  |  form A m+2 F
 1 |  U 11  θ  |
```

```
     2   A   3  F   ]
     3   U  19  θ   ] form E m+4 F
     4   S  20  θ   ]
     5   T   8  θ   ] form S m+3 F
     6   H   4  D
     7   T      D     clear 0D
     8   (S     F)   ]
18 → 9   A  21  θ    ]
    10   U   6  F    ] form A n-2p D
    11   (A     F)   ]
    12   T  14  θ    ]
    13   V      D    ]
    14   (A     F)   ] form polynomial
    15   Y      F    ]
    16   T      D    ]
    17   A   6  F   ]
    18   G   9  θ   ] test for completion
    19   (E     F)    link
    20 ‖ M   1  F
    21 ‖ P   2  F
```

K8 Shift of origin of a polynomial (numbers expressed in floating decimal form).
Uses library subroutine A11, whose first order is in Δ.

```
       G      K
     0 A   3  F   ]
     1 T  29  θ   ] plant link
     2 S  33  θ   ]
28 → 3 T   4  F   ] plant -(P 2s F); initially -(P 2n F)
     4 A  30  θ   ]
     5 T      D   ] place 0.10^-60 in 0D
     6 E   9  θ    jump
25 → 7 S   4  F   ]
     8 A   5  M    add P 2 F ] modify counter
     9 U   5  F   ]
    10 A  32  θ   ]
    11 U  18  θ   ] form and plant A h+2t D
    12 A  31  θ   ]
    13 T  22  θ   ] form and plant T h+2t D
    14 A  14  θ   ]
    15 G      Δ   ] call in A11
    16 ‖ A    D   ]
    17 ‖ V   6 D    X_{n+1} = X_n · x+a_r+1   ] parameters interpreted by A11
    18 ‖ (A   D)  ]
    19 ‖ T    D   ]
    20 ‖ E  21 θ   ]
A11→21 A      D   ]
    22 (T     D)  ] copy X_{r+1}
```

23	A	5	F	⎫	test for end of each synthetic division
24	A	4	F	⎬	
25	G	7	θ	⎭	
26	A	4	F	⎫	test for end of last division
27	A	5	M	⎬	add P 2 F
28	G	3	θ	⎭	
29	(E		F)		link
30	F		F		0.10^{-60}
31	O		F		(T - A)
32	A	h	D	⎫	punched by user
33	P	2n	F	⎭	

L1 Logarithm to base 2, large range.

$\frac{1}{32}\left[\log_2 C(6D)\right]$ to 0D

Fractional part of logarithm is formed digit by digit, using a shifting (negative) strobe.

	G		K		
0	A	3	F	⎫	plant link
1	T	33	θ	⎭	
2	E	11	θ		
3	I		F		= 1/2
4	P	1024	F		= 1/32
5	P	512	F		= 1/64
14 → 6	A	3	θ	⎫	integral part of logarithm: shift to left, counting in 0D
7	L		D	⎬	
8	T	6	D	⎬	
9	A		D	⎬	
10	S	4	θ	⎬	
2 → 11	T		D	⎬	stop when C(6D) ≥ 1/2
12	S	3	θ	⎬	
13	A	6	D	⎬	
14	G	6	θ	⎭	
15	T	8	F		clear accumulator
16	S	5	θ	⎫	plant strobe
32 → 17	T	4	D	⎭	
18	H	6	D	⎫	square C(6D) and test whether ≥1/2 or <1/2
19	V	6	D	⎬	
20	S	3	θ	⎬	
21	E	34	θ	⎭	
22	A	3	θ		
23	L		D	⎫	shift left
24	Y		F	⎬	
25	T	6	D	⎭	
26	A	4	D	⎫	enter digit in logarithm
27	A		D	⎬	
28	T		D	⎭	

<1/2

Digit cycle for fractional part of logarithm

```
37 ——→29 |  A   4  D |
         30 |  R      D |  shift strobe
         31 |  Y      F |
         32 |  G  17  θ |  test for last digit
         33 | (E      F) |  link
21 ——→34 |  A   3  θ |
         35 |  Y      F |
         36 |  T   6  D |  ≥1/2
         37 |  E  29  θ |
```

M1 Assembly subroutine No. 1.
For details see Part I, Section 4-6.

```
    |  G      K |
  0 |  H   2  F |  P 1 F to multiplier register
  1 |  T      F |  store TF, IF or PF
  2 |  A  42  F |
  3 |  C  42  F |  increase C(42) by 1 if odd
  4 |  U  42  F |
  5 |  A   8  F |  add TF
  6 |  U  22  F |  revised transfer order
  7 |  A  15  θ |  form reference order*
  8 |  A      F |
  9 |  H   8  F |  restore multiplier register contents
 10 | (T  45  F) |  plant reference order
 11 |  A  10  θ |
 12 |  A   2  F |  advance location of reference order
 13 |  T  10  θ |
 14 |  E  34  F |  return to initial orders
 15 | ‖N      F |
    |  T  44  K |
```

*C(Acc.) upon entry: T, I, or P. Corresponding reference order: P, E, or G.

M2 Assembly subroutine No. 2.
For details see Part I, Section 4-6.

```
    |  G      K |
Enter on-0 |  T      F |  address (0 or 1) to 0
reading S 1 |  H      F |
         2 |  C  22  F |
         3 |  A  22  F |  adjust address of transfer order if necessary
         4 |  U  22  F |
         5 |  S   8  F |
         6 |  U  42  F |  set C(42)
         7 |  A  40  F |  add function letter to form reference order
         8 | (T  16  θ) |  plant reference order
         9 |  A   2  F |
        10 |  A   8  θ |  advance location of reference order
        11 |  T   8  θ |
```

12	H	8	F	restore C(R)
13	E	34	F	return to initial orders
Enter on-14	A	43	F	these orders perform the role of the
reading π15	E	8	F	original orders in 27 and 28
	T	27	K	
27F	E	14	θ	these orders replace orders 27 and 28 so
28F	E		θ	that π switches to 14θ and S to θ
	T	44	K	
44F	P	16	θ	

P7 Print positive integer up to 10 digits.

Prints C(0D) · 2^{34} with suppression of nonsignificant zeros but without layout.

	G		K	
0	A	3	F	plant link
1	T	26	θ	
2	H	28π	θ	multiply by $2^{34}/10^{10}$ and add 2^{-34}
3	N		D	
4	Y		F	
5	L		D	
6	T	4	D	
7	S	27	θ	- 1/32 to 0
8	T		F	
9	H	8	θ	set multiplier
10	S	8	θ	set digit count
25 → 11	T	1	F	digit count
12	V	4	D	multiply
13	A		F	test for first nonzero digit*
14	G	31	θ	
15	S		F	
16	L		D	shift
17	U		F	print
18	O		F	
19	F		F	check and remove
20	S		F	
21	L	4	F	shift
34 → 22	T	4	D	
23	A	1	F	count digits
24	A	27	θ	
25	G	11	θ	
26	(		)	link
	T	28π	Z	**
	P		F	
	T	27	Z	
27	P	1024	F	$= -2^{33}/10^{10}$
28	P	610	D	
29	θ	524	D	
30	ϕ		F	

digit cycle

14 → 31	O	30	θ	space	⎤	
32	S		F	add 1/32		suppress zero
33	L	8	F	shift		
34	E	22	θ		⎦	

* C(0) = - 1/32 until first nonzero digit is printed, when C(0) becomes positive, thus preventing the suppression of later zeros.

** These symbols appear on the tape and serve merely to clear 28D, thus ensuring that the sandwich digit between 28 and 29 is zero, before further orders are read.

P11 Print signed decimals in preset layout.

	T		Z		
0	A	3	F	⎤	plant link
1	T		M	⎦	
2	A	20	θ	⎤	test for end of line
3	G	11	θ	⎦	
4	O	1	M		carriage return
5	O	2	M		line feed
6	S	2	F	⎤	
7	E	J	θ		
8	O	2	M		
9	A	7	M		line and column count
7 → 10	S	19	θ		
3 → 11	A	7	M		
12	T	20	θ	⎦	
13	A		D		
14	E	21	θ		test sign
15	T		D		
16	S		D		change sign
17	O		θ		print -
18	E	22	θ		
19	P		N		
20	P		F		
14 → 21	O	5	M		space
18 → 22	P		H		round-off
23	T		D		
24	A	6	M	⎤	
25	H	4	M		
	G		K		set multiplier, digit count
41					
44 (0) 26	T	4	F	⎦	
(1) 27	V		D		multiply by 10/16
(2) 28	U		F	⎤	print
(3) 29	O		F	⎦	
(4) 30	F		F	⎤	check and remove
(5) 31	S		F	⎦	
(6) 32	G	9	θ	⎤	
(7) 33	S	3	M		
(8) 34	G	J	θ		test for correct print
(6) → (9) 35	O	2	M		
(8) → (10) 36	A	3	M	⎦	

	37	L	4	F		shift
	38	T		D		
	39	A	4	F	⎫	
	40	L		D	⎪	
	41	E		θ	⎪	
	42	O	5	M	⎪	digit count
	43	L		D	⎪	
	44	G		θ	⎭	
	45	O	5	M	⎫	
	46	O	5	M	⎬	spacing
	47	O	5	M	⎭	
		O	40	K	⎫	figure shift performed during input
		π		F	⎭	of orders
		T	22	K		
		T		M		
M	0					link
	1	θ		F		carriage return
	2	Δ		F		line feed
	3	Q		F		1/16
	4	J		F		10/16
	5	φ		F		space
	6	P		Δ		digit layout constant
	7	P	5	F		block constant

P14 Print signed decimal with round-off and digit check.

		O	40	K	⎫	figure shift during input
		π		F	⎭	
		T		Z		
	0	A	45	θ		
	1	U	4	θ		form A n+2 F
	2	A	22	θ		
	3	T	39	θ		form link
	4	(A		F)		= A n+2 F or layout count
	5	E	8	θ		
	6	O	40	θ		carriage return
	7	O	41	θ		line feed
5 →	8	T	4	θ		layout count in 4θ
	9	A		D		
	10	E	15	θ		test sign of C(0D)
	11	T		D	⎫	
	12	S		D	⎭	reverse sign
	13	O		θ		print -
	14	E	16	θ		
10 →	15	O	42	θ		print space
14 →	16	P		H		round-off order
	17	T		D		
	18	H	44	θ		
	19	A	4	θ		

35, 38 →	20	T	4	θ	
	21	V		D	multiply by 10/16
	22	U	1	F	print digit and check
	23	O	1	F	
	24	F		F	
	25	S		F	
	26	G	29	θ	
	27	S	43	θ	
	28	G	30	θ	
26 →	29	O	41	θ	
28 →	30	A	43	θ	
	31	L	4	F	
	32	T		D	
	33	A	4	θ	layout count
	34	L		D	
	35	E	20	θ	
	36	O	42	θ	
	37	L		D	
	38	G	20	θ	
	39	(E		F)	link
	40	θ		F	carriage return
	41	Δ		F	line feed
	42	φ		F	space
	43	Q		F	
	44	J		F	
	45	P	2	F	

Q1 Quadrature, using Simpson's rule.

Forms and places in pD the sum:

$$h\left[f_0 + 4f_1 + 2f_2 + 4f_3 + \ldots + 4f_{n-1} + f_n\right] \approx 3\int_a^b f(x)dx$$

where $f_0 = f(a)$, ... $f_r = f(a+rh)$, ... $f_n = f(b)$

		T		Z	
	0	A	41	θ	plant A m+2 F
	1	U	8	θ	
	2	A	2	F	plant A m+3 F
	3	U	11	θ	
	4	A	3	F	plant link
	5	U	40	θ	
	6	S	42	θ	plant S m+4 F
	7	T	36	θ	
	8	(P		F)	becomes A m+2 F; later $x_r = a + rh$
	9	U	8	θ	set $x_0 = a$
	10	T		F	
	11	(P		F)	becomes A m+3 F; later h
	12	T	11	θ	plant h
	13	T		H	clear pD

```
      14 |  T  43  θ |  clear 43
39 ---> 15 |  T   4  F |  clear accumulator ]
      16 |  A  21  θ |                    ] for first and
      17 |  T  23  θ |  set "shift" order* ] last ordinates
37 ---> 18 |  T   4  F |  clear accumulator
      19 |  A  19  θ |  call in auxiliary sub-
      20 |  P      N |    routine to compute f(x_r)
Aux --> 21 |  H  11  θ |  multiply ordinate by h
      22 |  V      D |
      23 | (P      F) |  "shift" order*
      24 |  Y      F |
      25 |  A      H |  add to partial sum
      26 |  T      H |
      27 |  A  45  θ |
      28 |  S  43  θ |
      29 |  U  43  θ |  set "shift" order*
      30 |  A  44  θ |
      31 |  T  23  θ |
      32 |  A   8  θ |
      33 |  A  11  θ |  add h to a + rh
      34 |  U   8  θ |
      35 |  U      F |
      36 | (P      F) |  becomes S m+4 F
      37 |  G  18  θ |  jump when r≤n-1
      38 |  S  11  θ |
      39 |  G  15  θ |  jump when r = n
      40 | (P      F) |  link
      41 |  P   2  F |
      42 |  M   1  F |
      43 | (P      F) |  P F ←→ P D
      44 |  L      D |
      45 |  P      D |
```

* Order 23 is L 1 F for odd numbered ordinates, and L D for even ordinates, except the first and last, for which it is H 11 θ (no effect).

Q3 Quadrature using Gauss' six-point formula

Computes $\int_{a-h}^{a+h} f(x)dx$ by the approximation

$$2h \sum_{i=1}^{3} d_i[f(a+b_i h) + f(a-b_i h)],$$

where d_i and b_i are constants. This is equivalent to fitting a curve of the eleventh degree.

$a = C(mD)$, $h = C[(m+2)D]$

```
      |  T      Z |
 0    |  A   3  F |  ]
 1    |  T  30  θ |  ] plant link
 2    |  T   4  H |    clear 4H
```

```
          3 |  S   31  θ |
24 ——→    4 |  A   32  θ |  plant orders
          5 |  U   16  θ |
          6 |  A   33  θ |
22 ——→    7 |  T    J  θ |
          8 |  A        H |  +a
          9 |  H    2   H |
         10 | (V        F) |  a ± b_i·h
         11 |  Y        F |
         12 |  T        D |  gives x
         13 |  A   13  θ  |  calculate f(x)
         14 |  G        N |
Aux ——→  15 |  H        D |
         16 | (V        F) |  d_i·f(x)
         17 |  Y        F |
         18 |  A    4   H |
         19 |  T    4   H |  forms Σ_{i=1}^{3} d_i[f(a+b_i h) + f(a-b_i h)]
         20 |  A    J  θ  |
         21 |  A   34  θ  |  test if V or N in Jθ
         22 |  G    7  θ  |
         23 |  S   35  θ  |
         24 |  G    4  θ  |
         25 |  H    4   H |
         26 |  V    2   H |
         27 |  L        D |
         28 |  Y        F |
         29 |  T        D |
         30 | (E        F) |  link order
         31 ||  P    6   F |
         32 ||  V   42πθ   |
         33 ||  M    6   F |
         34 ||  O        F |
         35 ||  I   46πθ   |
            |  E   69   K |
            |  T   36πθ   |
36πθ        |    14716 66184 F |  d_1
38πθ        |    30989 18315 F |  d_2
40πθ        |    40193 50093 F |  d_3
42πθ        | 1  60197 04270 F |  b_1
44πθ        | 1  13594 90762 F |  b_2
46πθ        |    40994 46400 π |  b_3
            |  T   48   Z |
```

R1 Input of a sequence of signed, long, decimal fractions.

```
            |  G        K |
            |  T   45   K |
45F         |  P   32  θ  |  H parameter
46F         |  P   47  θ  |  N parameter
            |  T        Z |
```

	0	A	3	N	plant A m+2 F
	1	U	4	θ	
	2	A	4	N	plant link
	3	T	9	H	
	4	(A		F)	(i) A m+2 F (ii) digit count
35 →	5	T		H	plant transfer order
	6	T		D	clear 0D and 4D
	7	T	4	D	
	8	A	5	N	reset switch
	9	T	13	θ	
	10	H	2	N	set multiplier
	11	S	6	N	set digit count**
24 →	12	T	4	θ	digit count
	13	(I		F)	or T F when switched*
	14	A		F	
	15	S	6	N	test symbol for
	16	E	4	H	+, -, or F
	17	T	6	F	clear accumulator
	18	V	4	D	multiply previous digits
	19	L	8	F	shift
	20	A		D	add new digit
	21	T	4	D	
46 →	22	A	4	θ	
	23	A	7	N	count digits
	24	G	12	θ	
	25	H	4	D	
	26	N		N	
	27	R	128	F	
	28	R	128	F	multiply by $2^{34}/10^{10}$
	29	V	1	N	
	30	L		D	
	31	Y		F	
H	32	(T		F)	transfer to store
1	33	A		H	change transfer order
2	34	A	3	N	
3	35	E	5	θ	
16 → 4	36	S	6	N	
5	37	E	42	θ	test for -
6	38	A	7	N	
7	39	E	44	θ	test for +
8	40	T	6	F	F: clear accumulator
9	41	(E		F)	link
37 →	42	S	4	D	negative:
	43	T	4	D	change sign
39 →	44	A	2	N	
	45	T	13	θ	set switch to TF
	46	E	22	θ	
N	47	P	610	D	
1	48	Z	1523	D	
2	49	T		F	= 10/32

digit cycle

+, - and F

+ and -

3	50	P	2	F
4	51	U	1	F
5	52	I		F
6	53	P	5	D
7	54	P		D

* Order 13 is I F during input of punched digits, T F for dummy zeros which make up remainder of 10 digits.

** Digit count is actually set to 11 because + or - sign is counted as a digit.

R2 Positive integer input during input of orders.

		G		K	
9 →	0	T	20	F	
	1	V		D	] 10 · (partial sum)*
	2	L	8	F	]
	3	A	40	D	add new digit
14 →	4	U		D	] New partial sum to 0D** and to
	5	(T		F)	] final destination of number
	6	I	40	F	] read next symbol
	7	A	40	F	]
	8	S	39	F	subtract 11.2^{-16}
	9	G		θ	test for F
	10	S	2	F	subtract 2.2^{-16}
	11	G	23	F	test for π (if π return to initial orders)
	12	A	5	θ	] change destination of integer
Enter →	13	T	5	θ	]
	14	E	4	θ	

digit cycle

number cycle

Followed on tape by:

E	13	Z	on subroutine tape
T	m	D	punched by user

Hence control enters subroutine at order No. 13, with T m D in the accumulator.

* The multiplier register contains 10/32 throughout input of orders and operation of this subroutine.

** When obeyed for the first time in each number cycle, this order clears 0D.

R7 Input of a sequence of signed long decimal fractions during program.
For details of punching see specification in Part II.

	G		K	
0	A	6	θ	] forms A n+2 F
1	U	4	θ	]
2	A	7	θ	
3	T	31	θ	plants link

```
 4 |  (A      F)
 5 |   E  18  θ
 6 |   P   2  F
 7 |   U   1  F
   |   T  8πZ
   |   C 819  F      places -1/10 in 8πθ
   |   T   8  Z
   |   L1229  F
   |   T  J πZ
   |   D 409  D      places -8/10 in 10πθ
   |   T  J   Z
   |   S1638  D
   |   T  12  Z
33 → 12 V  J πθ      multiply by -8/10
36 → 13 L     D
   14   Y     F
   15  (T     D)     transfer to store
   16   A  15  θ
   17   S   6  θ     decrease transfer order by 2
 5 → 18   T  15  θ
   19   E  22  θ
29 → 20   N  8πθ     multiply by 1/10
   21   Y     F
19 → 22   T     D
   23   H     D
   24   I     F      read digit                    digit cycle
   25   A     F
   26   R  16 F      shift to most significant
   27   T     D       position
   28   A     F
   29   E  20  θ     test for + or -
   30   A  21  θ
   31  (E     F)     test for x: also link
   31   S  25  θ
   33   E  12  θ     test for +
   34   T     F
   35   N  J πθ      multiply by 8/10
   36   E  13  θ
```

S2 Square root, fast $\sqrt{C(4D)}$ to 4D.

Repetitive process

$$a_{n+1} = a_n - 0.5\, a_n c_n \qquad a_0 = C(4D) \qquad a_n \to \sqrt{C(4D)}$$

$$c_{n+1} = c_n^2 (0.25\, c_n - 0.75) \qquad c_0 = C(4D) - 1 \qquad c_n \to 0$$

```
   |  G     K
 0 |  A   3 F    plant link
 1 |  T  20 θ
 2 |  A   4 D    form c0
 3 |  S   9 θ
 4 |  A   6 θ
```

19 →	5	U		D	c_n to R	repetitive cycle
	6	H		D		
	7	R	1	F	$(0.25\,c_n - 0.75)$ to 0D	
	8	S	21	θ		
	9	T		D		
	10	N	4	D	a_{n+1} to 4D	
	11	R		D		
	12	A	4	D		
	13	Y		F		
	14	T	4	D		
	15	V		D	form c_{n+1}	
	16	T		D		
	17	V		D		
	18	Y		F		
	19	G	5	θ	test for $c_{n+1} = 0$	
	20	(E		F)	link	
	21	‖S		F	= 3/4	

S3 Cube root of C(6D) to 0D.

Root is formed digit by digit, using a shifting (negative) strobe.

		G		K	
	0	A	3	F	plant link
	1	T	20	θ	
	2	T		D	set first trial (i.e., zero)
	3	S	24	θ	set strobe
19 →	4	T	4	D	
	5	H		D	form $(\text{trial})^3$ - C(6D)
	6	V		D	
	7	Y		F	
	8	T	8	D	
	9	V	8	D	
	10	S	6	D	
	11	E	21	θ	
	12	T	8	D	increase trial
	13	S	4	D	
23 →	14	A		D	
	15	T		D	
	16	A	4	D	shift strobe
	17	R		D	
	18	Y		F	
	19	G	4	θ	
	20	(E		F)	link
11 →	21	T	8	D	decrease trial
	22	A	4	D	
	23	G	14	θ	
	24	‖I		F	= 1/2

T3 General cosine (used with R9).

0.5 cos $[2^m C(4D)]$ to 4D. The argument is formed modulo 2π by multiplication by $2/\pi$, and a suitable left shift to cast off the integral part. This yields θ/π where $\theta = 2^m C(4D)$ modulo 2π, and $|\theta| \leq \pi$. A power series is then used to form 0.5 sin 2x, where $x = 0.5\ (|\theta| - \pi/2)$.

```
        46F |  P  42πθ |  N parameter
            |  E  69  K |
            |  T     N |
N       0   | 11,453,246,123 F
        2   |  2,290,649,225 F
        4   |    218,157,069 F
        6   |     12,119,837 F
        8   |        440,721 F
       10   |         11,144 F
       12   | 13,493,037,705 F
       14   | 10,937,044,409 π
            |  T     Z |
        0   |  A   3  F |
        1   |  T  41  θ |  plant link
        2   |  H   4  D |
        3   |  V  14  N |  multiply by 2/π
        4   |  (L     H) |  multiply by 2^(m-1)
        5   |  E   8  θ |  ┐
        6   |  T      D |  ├ take modulus
        7   |  S      D |  ┘
5 ──→   8   |  S  58  θ |  - 1/2
        9   |  T      D |
       10   |  H      D |
       11   |  N  12  N |  multiply by π/4
       12   |  L      D |
       13   |  T   4  D |
       14   |  H   4  D |  ┐
       15   |  V   4  D |  ├ x² to 0D
       16   |  Y      F |  │
       17   |  T      D |  ┘
       18   |  H      D |  ┐
       19   |  N  J   N |  ├ a11 - a13x²
       20   |  A   8  N |  │
       21   |  T      D |  ┘
       22   |  N      D |  ┐
       23   |  A   6  N |  ├ a9 - a11x² +
       24   |  T      D |  ┘
       25   |  N      D |  ┐
       26   |  A   4  N |  ├ a7 - a9x² +
       27   |  T      D |  ┘
       28   |  N      D |  ┐
       29   |  A   2  N |  ├ a5 - a7x² +
       30   |  T      D |  ┘
```

31	N		D	$a_3 - a_5x^2 +$
32	A		N	
33	T		D	
34	N		D	$-a_3x^2 + a_5x^4 -$
35	T		D	
36	H		D	$-a_3x^3 + a_5x^5$
37	V	4	D	
38	Y		F	
39	A	4	D	$x - a_3x^3 + a_5x^5 \ldots$
40	T	4	D	
41	(E		F)	
	T	58	Z	
58	‖I		F	$= 1/2$

T4 Inverse cosine.

0.5 arc cos 2 C(4D) to 0D where $0 \leq C(4D) \leq 1/2$. Proceeds by finding successively the sign of 0.5 cos 2^nC(4D) formed from 0.5 cos 2^{n-1}C(4D) by $x_n = 4x_{n-1}^2 - 1/2$. The required result is built up digit by digit, using a negative strobe.

		G		K	
	0	A	3	F	
	1	T	28	θ	plant link
	2	T		D	
	3	A	32	θ	
20 →	4	T	6	D	strobe in 6D
	5	H	4	D	form $x_n = 4x_{n-1}^2 - 1/2$
	6	V	4	D	
	7	L	1	F	
	8	S	29	θ	
	9	Y		F	
	10	E	16	θ	test sign of x_n
	11	T	4	D	
	12	S		D	form partial sum
	13	A	6	D	
	14	T		D	
	15	S	4	D	
10 →	16	T	4	D	
	17	A	6	D	shift strobe
	18	R		D	
	19	Y		F	test for end of cycle
	20	G	4	θ	
	21	H		D	
	22	N	30	$\pi\theta$	multiply by $\pi/4$
	23	Y		F	
	24	E	27	θ	take modulus
	25	T	4	D	
	26	S	4	D	
24 →	27	T		D	
	28	(E		F)	

29	‖I		F	= 1/2
	T	30πZ		
31	‖D	888	F	places $-\pi/4$ in $30\pi\theta$
	T	30	Z	
30	‖O	699	D	
	T	32	Z	
32	‖K	4096	F	= -1

T5 0.5 cos x and 0.5 sin x at equal intervals of x. Version 1.

	T		Z	
0	E	9	θ	
1	‖I		F	= 1/2
	T	2πZ		
2πθ	‖(I		F)	0.5 cos x
	T	4πZ		
4πθ	‖(P		F)	0.5 sin x
	T	6	Z	
Reset → 6	A	1	θ	
7	T	2πθ		reset to x = 0
8	E	19	θ	
0 → 9	H	4πθ		
10	N	2	H	
11	H	2πθ		
12	V		H	new value of 0.5 cos x
13	Y		F	
14	T	2πθ		
15	V	2	H	
16	H	4πθ		
17	V		H	new value of 0.5 sin x
18	Y		F	
8 → 19	T	4πθ		

T7 Sine, rapid (used with R9).

1/2 sin [2 C(4D)] to 4D

	G		K	
	E	69	K	calls in R9
	T	26πθ		
26πθ	11	453	246 086	F
28πθ	2	290	648 539	F
30πθ		218	152 390	F coefficients of power series
32πθ		12	105 378	F
34πθ			419 996	π
	T		Z	
0	A	3	F	
1	T	25	θ	link
2	H	4	D	
3	V	4	D	
4	Y		F	form $[C(4D)]^2$
5	T		D	

6	H		D	summation of power series
7	N	$34\pi\theta$		
8	A	$32\pi\theta$		
9	T		D	
10	N		D	
11	A	$30\pi\theta$		
12	T		D	
13	N		D	
14	A	$28\pi\theta$		
15	T		D	
16	N		D	
17	A	$26\pi\theta$		
18	T		D	
19	N		D	
20	T		D	
21	H		D	
22	V	4	D	
23	A	4	D	
24	T	4	D	
25	(E		F)	link
	T	36	Z	

U3 Counting subroutine No. 3 (open).

	G		K	
0	A	16	θ	test and jump if $q = q_r$
1	S	18	θ	
2	E	6	θ	
3	T		F	
4	A	16	θ	
5	E	12	θ	
2 → 6	A	1	θ	modify C(1) and C(14)
7	A	15	θ	
8	T	1	θ	
9	A	14	θ	
10	A	15	θ	
11	T	14	θ	
5 → 12	A	2	F	replace q-1 by q
13	T	16	θ	
14	E	17	θ	
15	P	2	F	q
16	(P		F)	

U4 Counting subroutine No. 4 (closed).

	G		K	
0	A	2	F	A m+2 F to 22 θ
1	A	2	F	
2	U	22	θ	
3	A	2	F	

	4	U	12	θ	A m+3 F to 12 θ
	5	A	2	F	
	6	U	14	θ	A m+4 F to 14 θ
	7	A	28	θ	
	8	U	19	θ	G m+5 F to 19 θ
	9	S	28	θ	
	10	A	3	F	
	11	T	27	θ	E m+6 F to 27 θ
	12	(P		F)	becomes A m+3 F
	13	T	20	θ	*A s F to 20 θ
	14	(P		F)	becomes A m+4 F
	15	T	14	θ	
	16	S	14	θ	
26 →	17	T	14	θ	
	18	A	18	θ	
	19	(P		F)	becomes G m+5 F
	20	(P		F)	becomes A s+xr F
	21	A	20	θ	
	22	(P		F)	becomes A m+2 F
	23	T	20	θ	
	24	A	14	θ	
	25	A	2	F	
	26	G	17	θ	
	27	(P		F)	becomes E m+6 F (link)
	28	‖V	1	F	

* may be replaced by any other function letter.

V1 Multiplication of vector by contracted symmetric matrix.

	49F	Y		F	L parameter
	50F	P	24	θ	X parameter
		T		Z	
	0	A	3	F	] plant link
	1	T	20	X	]
	2	T	7	F	clear 7
46 →	3	U	6	F	] set C(6), C(7) and initial values of C(4), C(5) for next scalar product
	4	A	7	F	
	5	U	7	F	
	6	T	4	D	]
	7	A	6	F	] set transfer order
	8	A	4	X	
	9	T	17	X	]
	10	T		D	clear 0D
39 →	11	A	2πX		]
	12	A	4	D	
	13	T	14πθ		
	14	(H		F)	] form single product
	15	(V		F)	]
	16	P		L	round-off or shift]]

	17	A		D	add to partial sum		
	18	T		D			
	19	A	4	F			
	20	S	6	F			
	21	G	6	X	to 6X if diagonal not yet reached		
	22	A	6	F	on or after diagonal		
	23	E	7	X			
X	24	V	2046	H	= P 2(n-1) F		
1	25	P		H	= P 2n F		
2	26	P		N	= H c D		
3	27	P		M	= V m D	single	
4	28	P		Δ	= T s D	product	
5	29	P	2	F		cycle	
21→ 6	30	T	14	θ	clear accumulator		scalar
23→ 7	31	A	5	X			product
8	32	A	4	D	advance C(5)		cycle
9	33	T	4	D			
10	34	A	4	F			
11	35	S		X	advance C(4) and		
12	36	E	16	X	test for end of		
13	37	A	1	X	scalar product		
14	38	T	4	F			
15	39	E	11	θ			
36→16	40	A		D	plant scalar product		
17	41	(T		F)			
18	42	A	6	F			
19	43	S		X	link	advance C(6)	
20	44	(E		F)		and test for	
	45	A	1	X		end of process	
	46	E	3	θ			

Note: When forming the pth scalar product, C(6) = P 2(p-1) F and C(7) = P p(p-1) F. When forming the qth term of the pth element, C(4) = P 2(q-1) F and C(5) is the address of the matrix element relative to m. If

$q \leq p$, C(5) = P[p(p-1) + 2(q-1)] F,

$q > p$, C(5) = P[q(q-1) + 2(p-1)] F.

V2 Addition and subtraction of n dimensional vectors.

	T		Z	
0	A	23	θ	
1	U	11	θ	
2	A	2	F	
3	U	13	θ	
4	A	2	F	
5	U	15	θ	
6	A	10	θ	
7	T	22	θ	plant link
8	S	24	θ	

21 →	9	A	23	θ		⎫
	10	U	1	F		⎪
	11	(P		F)	A m+2 F	⎪
	12	U	17	θ		⎪ plant arithmetic orders
	13	(P		F)	A m+3 F	⎪
	14	U	18	θ		⎪
	15	(P		F)	A m+4 F	⎪
	16	T	19	θ		⎭
	17	(P		F)		⎫
	18	(P		F)		⎬ arithmetic operation
	19	(P		F)		⎭
	20	A	1	F		
	21	G	9	θ		test for end of operation
	22	(P		F)		link
	23	P	2	F		
	24	P		H		

APPENDIX A Keyboard perforator code, etc.

Perforator character		Teleprinter character		Code as punched on tape	Binary equivalent	Decimal equivalent	
P	0	P	0	● .	00000		0
Q	1	Q	1	● . ●	00001		1
W	2	W	2	● . ●	00010		2
E	3	E	3	● . ● ●	00011		3
R	4	R	4	● . ●	00100		4
T	5	T	5	● . ● ●	00101		5
Y	6	Y	6	● . ● ●	00110		6
U	7	U	7	● . ● ● ●	00111		7
I	8	I	8	● ● .	01000		8
O	9	O	9	● ● . ●	01001		9
J	Bell	J		● ● . ●	01010		10
π		Figures		● ● . ● ●	01011		11
S	'	S	"	● ● . ●	01100		12
Z	§	Z	+	● ● . ● ●	01101		13
K	(	K	(	● ● . ● ●	01110		14
Erase		Letters		● ● . ● ● ●	01111		15
Blank tape		(no effect)		.	10000	-16	16
F		F	$	. ●	10001	-15	17
θ		Carriage return		. ●	10010	-14	18
D	,	D	;	. ● ●	10011	-13	19
ϕ		Space		. ●	10100	-12	20
H	+	H	£	. ● ●	10101	-11	21
N	-	N	,	. ● ●	10110	-10	22
M	'	M	.	. ● ● ●	10111	-9	23
Δ		Line feed		● .	11000	-8	24
L	%	L	)	● . ●	11001	-7	25
X		X	/	● . ●	11010	-6	26
G		G	#	● . ● ●	11011	-5	27
A	:	A	-	● . ●	11100	-4	28
B	?	B	?	● . ●	11101	-3	29
C	(	C	:	● . ● ●	11110	-2	30
V	)	V	=	● . ● ● ●	11111	-1	31

Notes: 1. Positive and negative decimal equivalents are given for the last sixteen codes above. The negative equivalent applies when the symbol occurs as the five most significant digits of an order. The extreme left-hand digit is then a "1" and, for numerical purposes, acts as a sign digit, thus indicating a negative number.

2. It will be seen that the secondary characters on keyboard perforator and teleprinter do not agree in every case. It is intended that they should all eventually be brought into line.

APPENDIX B The initial orders

Location	Order	Notes
0	(T F)	These orders cause control to be transferred to 20. They are not used after the start, but their locations are used as working space.
1	(E 20 F)	
2	P 1 F	These are constants which are intended to be left here unaltered in any program.
3	U 2 F	
12→ 4	A 39 F	Input of address. This group of orders is entered at 8 with the accumulator empty, so that 0 is cleared. The next digit on the tape is taken in and tested to see if it is less than eleven; if so it is doubled and added to ten times the content of 0, the sum being sent back to 0. The next digit is read, tested, etc., and this is continued until the whole address has been formed; the next digit read, x, is greater than ten and so corresponds to a code letter.
5	R 4 F	
6	V F	
28 7	L 8 F	
38→ 8	T F	
9	I 1 F	
10	A 1 F	
11	S 39 F	
12	G 4 F	
13	L D	These test to see if x is greater than sixteen. If it is, the order A(24+x)F is formed and planted in 20. If x is sixteen or less a switch order E(16+x)F is formed and planted in 20.
14	S 39 F	
15	E 17 F	
16	S 7 F	
15→17	A 35 F	
18	T 20 F	
19	A F	This adds the address, which is always positive, into the accumulator.
20	(H 8 F)	This order places 10/32 in the multiplier register during the start and is later replaced by a manufactured one which either adds to the accumulator the number determined by x, or switches control to an address determined by x.
21	A 40 F	This adds in the function digits of the order so the accumulator now contains the order from the tape plus the number selected by x.
22	(T 43 F)	This (the transfer order) transfers the assembled order to its final place in the store.
23	A 22 F	These orders increase the address specified in the transfer order by unity.
24	A 2 F	
31→25	T 22 F	
26	E 34 F	Transfers control to 34.
20→27	A 43 F	Control is switched to these orders by 20 when π has been read from the tape. They add 2^{-16} to the address (which is in the accumulator)
28	E 8 F	

			and transfer control to 8. The address now refers to a long storage location.
20→ 29	A	42 F	This adds the address in 42 to the accumulator.
20→ 30	A	40 F	This adds the function digits of the order to the accumulator. The result is that the number in the accumulator is positive if the order has function digits represented by T or E, while it is negative in the case of G.
31	E	25 F	If the accumulator is positive, the order in the accumulator replaces the order in 22; if negative the accumulator contains the address specified in order 22 which is then put in 42 (the storage location corresponding to θ).
20→ 32	A	22 F	
33	T	42 F	
26→ 34	I	40 D	These take in the function digits, shift them to their correct place and transfer them to 40. The order in 35 is also used as a constant.
35	A	40 D	
36	R	16 F	
37	T	40 D	
38	E	8 F	
39	P	5 D	A constant used in the input of the address. It equals 11.2^{-16}
40	(P	D)	A constant used during the start. It equals 2^{-16}.

When the starting button is pressed, the initial orders are placed in storage locations 0 - 40 and control transferred to 0. The first orders to be executed are the following:

0	T	F	clears accumulator
1	E	20 F	transfers control to 20
20	H	8 F	places 10/32 in multiplier register
21	A	40 F	adds 2^{-16} to accumulator
22	T	43 F	transfers 2^{-16} to 43 (the storage location corresponding to D).
23	A	22 F	increase order 22 to T 44 F
24	A	2 F	
25	T	22 F	

The initial input is now ready to take in orders; the first part of the input tape is blank so that the first code letter is a space which corresponds to 16; control is therefore switched from 20 to 32, and the contents of 22 are transferred to 42. This action will continue, the spaces being treated alternately as function digits and code letters. The first symbols encountered will be P and F. There are two possibilities, either

(1) the last space has been treated as a function digit in which case the psuedo-order "space F" = 1000000000000... is transferred to 44, or

(2) the last space was treated as a code letter, in which case PF is transferred to 44.

In both cases the following orders will be put in sequence starting at 45, unless a control combination comes first.

APPENDIX C Control combinations

The operation and use of the more important control combinations are described in Part I, Chapter 2. Details will now be given of some less common control combinations which are sometimes used and which may be encountered in certain library subroutines.

It may be noted that the operation of code letter Z is always equivalent to that of K and θ combined.

Throughout this appendix storage location 42 is assumed to contain P n F, placed there by the preceding G K.

1.	(a) E Z P F	Transfers control to the first order of the last subroutine to be read, leaving the accumulator clear, i.e., control is transferred to n.
	(b) E m Z P F	Transfers control to the mth order of the last subroutine to be read, leaving the accumulator clear, i.e., control is transferred to (m+n).
2.	(a) E Z (b) E m Z (c) E m K — followed by any positive* order	These control combinations transfer control to (a) the first order of the last subroutine, (b) the mth order of the last subroutine, (c) the order in storage location m. The accumulator in all three cases is <u>not</u> left clear but contains the positive order which follows the control combination.
3.	T m Z	Replaces the transfer order by T (m+n) F, i.e., causes the orders following on the tape to be placed in storage locations (m+n), (m+n+1), etc.
4.	(a) T m π K (b) T m π Z	Replaces the transfer order by (a) T m D, (b) T (m+n) D, i.e., the next order, or pseudo-order, to be read from the tape is placed in the most significant half (the odd-numbered half) of the long storage location m, or (m+n), the least significant half, including the sandwich digit (see Part I, Section 4-2) being cleared. If the control combination is followed by P F, the whole long storage location is cleared.
5.	E 25 K followed by any positive* order	Transfers control to order 25 of the initial orders, which causes the transfer order to be replaced by the <u>positive</u> order following the E 25 K.
6.	T 22 K	Causes the transfer order to be replaced by the next order on the tape regardless of whether this is positive or negative. The address specified in this order is <u>immediately increased by unity</u>. For example, T 22 K, T m F will cause the orders following to be placed in storage locations (m+1), (m+2), etc.

7. (a) α m K (b) α m Z	where α is a function letter and α m F$\geq$0	Causes the transfer order to be replaced by (a) α m F, (b) α (m+n) F. If the accumulator is not cleared by this order further operation of the initial orders will not be possible unless the transfer order is restored by a suitable control combination.
(c) 0 40 K α F		This is a particular case of 7(a) and causes the character α to be printed during input without occupying any storage space. T m K must follow on the tape where the following order is to be placed in m.
8. G m K		Places a reference address in 42 equal to m plus the current address specified in the transfer order.
9. G Z		Adds the current address specified in the transfer order to C(42).

The above list explains the means whereby most simple operations can be carried out during the input of orders. More elaborate operations may be carried out by temporarily interrupting the action of the initial orders and transferring control to a suitable sequence of orders which have been placed in the store. The last of these orders should return control to order 25 of the initial orders. Care must be taken to ensure that none of the initial orders is disturbed and that the content of the multiplier register is restored if necessary.

* By "any positive order" is meant any order or pseudo-order whose numerical representation in the machine is positive. In general this means that the function letter on the tape must be positive, but there may be exceptions. For example, if the H parameter is P (n+1) F, then a pseudo-order punched as V 2047 H will appear in the machine as P n F.

APPENDIX D Interpretive subroutines: example of packing of orders

Consider the evaluation of the sum of the squares of the residuals of a set of nonlinear algebraic equations, that is, the evaluation of

$$\sum_{i=0}^{m} f_i(x_i.....x_m)^2,$$

where $f_i(x_i.....x_m) = 0$ is a typical equation, f_i being a function of its arguments which can be evaluated by a finite number of additions, subtractions and multiplications only.

If there is no uniformity in the algebraic forms of the function f_i, direct programming of their evaluation takes a large number of orders. However, if the number of variables is not too large, a considerable saving can be effected in the space occupied by orders by using a special "order" code, of which each "order" specifies a sequence of machine orders, and by packing two such "orders" into a single storage location. This also simplifies the task

of the programmer, for, since each special "order" specifies a sequence of operations, these do not have to be programmed individually. An interpretive subroutine is required to interpret "orders" expressed and packed in this form.

A possible code of special "orders" is given in the table. Six "orders" suffice to carry out the operations required in the process of calculating the residuals and summing their squares. One further "order" is required to return control to the main program. Thus the "orders" can be specified by three binary digits.

Moreover, for the operation of the code, three storage locations are used. These storage locations, numbered from an arbitrary zero, are indicated by [0], [1], [2]. [0] is used as a multiplier register, [1] to accumulate the sum of the squares of the residuals, and [2] as a working position used in the evaluation of each function value f_i in turn. The interpretation of these numbers in terms of long or short storage locations in the machine is carried out by the interpretive subroutine.

Thus, if provision is to be made for reference to not more than 29 other locations [n] for variables, constants, and intermediate quantities that must be stored in the course of the calculation, five digits will be required to specify the "address" of the number to be operated on. This brings the total number of digits necessary to specify an order to 3 + 5 = 8.

This leads to the possibility of packing two "orders" into one short storage location. Thus, the two "orders"

p	8	i.e.,	0 0 0 | 0 1 0 0 0
t	17		1 0 1 | 1 0 0 0 1

would appear in one short storage location as

Normal position of binary point

	p	8	t	17
0	0 0 0	0 1 0 0 0	1 0 1	1 0 0 0 1

Unpacking is performed by suitable collating and shifting orders, and packing by a small subroutine which can subsequently be written over, since it is only required during input. In the above example the packing subroutine used takes 38 orders, and the unpacking part of the interpretive subroutine 19 orders only.

As an example of programming using these special "orders" suppose the evaluation of

$$(x+y-2)^2 + (xy-1)^2$$

is required. If x is stored in location 4 (related to an arbitrary zero), y in location 5, 2 (suitably scaled down) in location 6, and 1 in location 7, the "coding" would appear as

w 4	(cont'd.)
w 5	w 0
e 6	e 7
r	r
p 4	y
q 5	

This program will thus occupy only 5 short storage locations.

The disadvantage of using such subroutines is the time involved. Here, the factor over direct coding is about 7, depending on the proportion of the different "orders" used. However, against this it might be pointed out that with a particular set of eight equations the over-all space saving was 70 short storage locations in 200.

A further possibility which arises is the packing of the interpretive section of the subroutine itself so that the same unpacking procedure applies to the routine being interpreted and to the routine doing the interpretation. If this is done, however, the time factor increases considerably (by about 40:1 in one program investigated) and it would appear, at least until faster high-speed stores become available, that such a procedure is of restricted utility.

"Order" code

Binary equivalent	Code letter	Symbolic description	Verbal description
0	p	C [n]→[0]	This is the first "order" in the formation of a sequence of continued products and puts C(n) in the storage location used as a multiplier register.
1	q	C[n] C[0]→[0]	This executes multiplication and stores the product ready as multiplier for the next multiplication.
2	w	C[n]+ C[2]→[2]	Accumulation of sums and differences.
3	e	-C[n]+ C[2]→[2]	
4	r	$C[2]^2 + C[1] \rightarrow [1]$ 0→[2]	Accumulation of squares of residuals. The interpretive subroutine must put C[1] = 0 at beginning of operation.
5	t	C[0]→n	Transfer. Many intermediate products may be repeated and should be stored for re-use.
6	y	Switch "order"	Return to machine order beyond this "order," i.e., return control from interpretive subroutine.
7	u	Blank	

APPENDIX E Methods of counting in a simple cycle

In programming, one of the most common problems is the coding of a simple cycle of orders in such a way that it is performed a certain number of times, n say, before the machine proceeds to the next part of the problem. In the absence of any special considerations, this is best done as follows. Assume that P n F, or $n \cdot 2^{-15}$, is stored in a, that the cycle begins at the order stored in c, and that b is used for the counting operation.

```
(c-1) | S a F |
  c   | T b F |
 c+1  |       |  ] required orders        ] cycle
  .   |       |  ]=accumulator clear      ]
  .   | A b F |                           ]
  .   | A 2 F |                           ]
  .   | G c F |                           ]
```

As the cycle is entered for the first time, $-n \cdot 2^{-15}$ is sent to b; thereafter it is increased by 2^{-15} each time the cycle is performed. On encountering the order G c F, C(Acc.) is negative each time until the end of the nth repetition, when it is zero.

Two advantages of this method should be noted. First, it is self-resetting, that is, it may be used several times in succession, without anything having to be restored. Second, when control finally leaves the cycle to obey the order following G c F, the accumulator is empty (as it is usually required to be). This method will not necessarily be the best if, for example, the accumulator is not required to be empty afterwards or if resetting is not required. There are many other possibilities. The counting may be done in steps of any size, positively or negatively, and the orders may be rearranged to suit special cases. When using a novel method, care must be taken to see that exactly the right number of repetitions will be obtained.

One common variation occurs when one or more orders within the cycle have to be changed each time the cycle is performed. To take a simple example, suppose that the long number in each location from 100 D to 298 D inclusive is to be increased by x. The orders to be changed have to be increased by P 2 F each time, so it is convenient to count in steps of P 2 F. Assume as before that P 200 F is stored in a, that the cycle begins at c, and that b is used for counting. In addition, the following constants are required:

Address	Constant
d	A 300 D
e	O F
f	P 2 F
gD	x

Then the previous example could be modified thus:

```
c-1  |  S  a  F  |  subtract P 200 F
c    |  U  b  F  |                          ]
c+1  |  A  d  F  |  add A 300 D             ]
c+2  |  U (c+6) F|                          ]
c+3  |  A  e  F  |  add O  F                ]
c+4  |  T (c+7) F|                          ]  cycle
c+5  |  A  g  D  |  add x                   ]  m = 100, 102, ..., 298
c+6  |  (Z    F) |  becomes A m D           ]
c+7  |  (Z    F) |  becomes T m D           ]
c+8  |  A  b  F  |                          ]
c+9  |  A  f  F  |  add P 2 F               ]
c+10 |  G  c  F  |                          ]
```

The variable orders are formed from the count-number by orders (c+1) to (c+4). Note that since the variable orders and the count number always change by the same amount, their differences are constant. The variable orders may thus be formed in succession from the count-number by adding the differences, without clearing the accumulator.

The cycle may be shortened by one order by using one of the variable orders itself as the counter. f is now used to store A 298 D instead of P 2 F, and b is no longer required.

c-1	S a F	subtract P 200 F	
c	A d F	add A 300 D	⌐
c+1	U (c+5) F		
c+2	A e F	add O F	
c+3	T (c+6) F		
c+4	A g D	add x	cycle
c+5	(Z F)	becomes A m D	
c+6	(Z F)	becomes T m D	
c+7	A (c+5) F		
c+8	S f F	subtract A 298 D	
c+9	G c F		⌙

Here, the order A m D itself is used as the counter. When it has reached A 298 D, C(Acc.) is no longer negative after obeying the order in (c+8), so the cycle is no longer repeated. It will be seen that the process is self-resetting. Examples of such cycles will be found in library subroutines E4 (orders 6 to 13), G1 (29 to 58), and K1 (4 to 12).

Counting operations are not restricted to addition and subtraction; it is sometimes convenient to count by shifting. In subroutine E2, for example, the number 2^{-34} is first placed in 6D. This number consists of a single digit at the right-hand end and this digit, or "strobe," is moved one place to the left at each repetition of the cycle. When it reaches the sign position it appears negative and repetition ceases. In L1, S3, and T4, a negative strobe moving to the right is used. The end of the process is detected by rounding off. When the strobe reaches -2^{-35}, the rounding-off brings it to zero and the sign digit changes. In all these examples, the shifting method is adopted because the strobe is also used in the calculation.

A more elaborate form of counting by shifting is employed in print subroutines P11 (orders 39 to 44) and P14 (orders 33 to 38) to count the characters printed in a number. A single counting operation controls not only the total number of decimal digits printed, but also the layout of subcolumns. Briefly, a certain psuedo-order is shifted one place to the left each time a character is printed, the sign digit is examined, and appropriate action taken. By suitably arranging the 0's and 1's in the pseudo-order a great variety of results may be obtained, thus in this instance, a pair of 1's terminates a subcolumn and a single 1 terminates the number.

Use of "tags"

It is sometimes possible to do away with the need for counting by arranging that the numbers operated upon give an indication when the last repetition is reached. If this can be done, it often reduces the number of orders required in the cycle, and increases the speed of working.

For example, if an operation is being carried out on a series of positive numbers, a negative number can easily be detected and if inserted deliberately will cause repetition to cease. Such numbers, with distinctive properties used to control the program, are called "tags." Further examples are the numbers 0 and -1. 0 can be distinguished because when squared negatively it gives a non-negative result, and -1 because its square and its complement appear negative to the machine. Tags can be used in a great variety of ways, apart from the control of a simple cycle. Thus numbers at one end of a permitted range can be detected by adding a constant and testing the sign, and then the result of the discrimination may be used to operate a multiway switch (see below).

Multiway switches

It is often convenient to pursue any one of a number of routes after a certain point in a program. These routes are usually defined by a discriminating number used to fabricate an E or G order. This order then transfers control to any of a number of E or G orders placed consecutively in the store, which in turn switch control to the desired address.

Thus, if a number $a \cdot 2^{-1b}$ is in the accumulator at a certain point and it is desired to switch control to one of a set of storage locations x_1, x_2, x_3, ..., x_n thereby, it is possible to proceed as follows:

s θ	A (b-1) θ	] adds a to E b θ forming E b+a θ, which is placed in storage location (s+2) θ.
s+1	T (s+2) θ	
s+2	(Z F)	
.	.	
.	.	
.	.	
.	.	
(b-1) θ	E b θ	
b	E x_1 θ	
b+1	E x_2 θ	
.	.	
.	.	
.	.	

INDEX