axis was drawn on the back. The long-wave record and the wave envelope were then cut off as long narrow strips, using a sharp knife and a steel straight edge, and in this form were ready for use.

In conclusion the author wishes to thank Mr N. F. Barber for his stimulating interest and helpful advice during the progress of this work, and the Admiralty for permission to publish the work.

Reference

Munk, W. H. 1948 *Rev. Sci. Instrum.* **19**, 654.

Programme organization and initial orders for the EDSAC

By D. J. Wheeler

University Mathematical Laboratory, University of Cambridge

(*Communicated by D. R. Hartree, F.R.S.—Received* 4 *April* 1950)

Orders for the machine are presented to it in coded form on punched tape, and are translated by the machine itself into the form in which they are held in the store, in accordance with a standard set of 'initial orders'. In the course of this input of orders it is possible to use the machine to make systematic modifications to the orders as punched on the tape; the modifications required, if any, are indicated by a code letter included in each order as punched on the tape. This system allows the use of a more flexible system of representing orders than the binary form used inside the machine.

It also enables sub-routines to be drawn up in a form independent of the particular values of parameters in them, the values of these parameters in any particular application of the sub-routine being inserted by the machine in the course of the initial input of orders, or in the course of operation of the machine. This simplifies the formation of a library of sub-routines and its use in the construction of long programmes.

A complete example is worked.

1. Introduction

This paper is concerned with some aspects of the organization of work for one kind of automatic calculating machine, of which the EDSAC, installed at the Mathematical Laboratory of the University of Cambridge, is an example. A general description of this machine has been given by Wilkes & Renwick (1949, 1950).

The essential parts of any automatic digital calculating machine are a store for holding numbers, an arithmetical unit, and a control unit. The store holds numbers in storage locations which are numbered so that we can refer to a storage location by means of an integer, known as the 'address' of that location, or of its content.

The essential features of the EDSAC for the purpose of this paper are three. First, it uses a single-address form for orders, that is, each order refers to a single storage location; the order code is given in appendix I. Secondly, the orders are

expressed inside the machine in coded form as numbers, and are contained in the same store as numbers, so that it is possible to modify orders by means of the arithmetic unit. Thirdly, the orders are normally placed in the store in such a way that the sequence of their addresses is the same as the time sequence in which they are to be carried out.

Another feature of the EDSAC, which affects some of the details but not the principles of this paper, is provision for grouping two adjacent numbers together and operating directly on the long number so formed. The content of the short-storage location n will be written $C(n)$ and that of the long-storage location m as $C(m')$; the content of the accumulator will be written as $C\,(Acc)$.

Some elementary aspects of programming for a machine with these features have already been discussed in a paper by Wilkes (1949).

2. Initial orders

The orders which control the successive stages of the calculation, and the numerical data required in the course of it, are taken into the EDSAC from five-hole teleprinter tape. This tape is prepared using a keyboard perforator. This has been arranged so that each key corresponding to an integer between 0 and 9 punches a row of holes that represents the binary form of that number. The basic input operation is to read one row of holes on the tape and place the corresponding five-digit binary number in the store. This operation is controlled directly by orders held in the store, and, consequently, before the tape can be read it is necessary to place a group of orders in the store by some independent means. These orders are known as the initial orders and are permanently wired on a set of uniselectors. When the starting button is pressed, these orders are automatically transferred to the store and they then cause the input tape to be read. This means that the structure of the initial orders determines the way in which orders have to be punched on the tape. There are many forms the initial orders can take, and they should be chosen so that the work of preparing a tape is as simple as possible.

3. Representation of orders

Inside the machine each order is expressed as a seventeen-digit binary number according to the following code. The five most significant digits represent the function of the order—add, subtract, etc. The sixth digit is '0'. The next ten digits give the address, m, of the storage location to which the order refers. The seventeenth digit is '0' if the order refers to a short number, and '1' if it refers to a long number. Since the first digit is regarded as a sign digit and a 'binary point' supposed to exist before the second digit, the number equivalent to the address m of an order is $m \times 2^{-15}$.

It is desirable that orders should be punched on the tape in the same form as they are written, without any modification or transliteration being necessary. The form in which orders are written and punched can be, and should be, chosen primarily with a view to the convenience of the user; the necessary conversion to the form in

which they are held in the store, including conversion of the address to the binary form, can then be arranged by a suitably chosen set of initial orders.

Two ways of writing orders have been used in connexion with the EDSAC, corresponding to two sets of initial orders. The second set of initial orders was introduced because the first set was found to suffer from certain disadvantages. Both sets will be described, since the first form of initial orders illustrate in a simple form some features also incorporated in the second set.

4. Initial orders—first form

With the first form of initial orders each order is written and punched as follows: first, a letter representing the five function digits—*A* corresponding to add, *S* to subtract, etc.; secondly, the address in decimal form with non-significant zeroes omitted (e.g. 15, not 0015); and thirdly an *S* or *L* according as the order refers to a short or a long storage location. For example, *A* 97 *S* stands for an order which causes the contents of 97 to be added to the accumulator, and *S* 100 *L* stands for an order which causes the contents of long storage location 100 to be subtracted from the accumulator. The abbreviated form of words '*A* 97 *S* adds *C* (97) to the accumulator' and '*S* 100 *L* subtracts *C* (100′) from the accumulator' will be used for these operations. The characters *S* and *L* are punched by the keyboard perforator in the binary forms of the decimal numbers 12 and 25, and can therefore be distinguished from decimal digits which are all less than 10.

The initial orders read the tape and assemble each order thereon in the following way:

The first row of holes—which corresponds to the function digits—is read, and the resulting binary digits are shifted to their correct digital position and placed temporarily in storage location 0. Subsequent rows of holes are then read. These will represent either decimal digits forming part of the address, or the characters *S* or *L* which terminates the address. The decimal digits are treated in sequence, a new partial address being formed each time, according to the following rules:

The first partial address is zero.

The $(n+1)$th partial address equals ten times the nth partial address, plus the nth decimal digit of the address. Since this is done inside the machine the address is formed as a binary number.

For example, suppose the address is 594. The numbers will be written in their decimal form, but it must be remembered that the machine deals with them in their binary form, the transformation of single decimal digits into binary form being done by the keyboard perforator. After the function digits are read the first partial address is 0. This is multiplied by 10 and 5 added, giving the second partial address, 5. This is multiplied by 10 and 9 added, giving the third partial address, 59. This is multiplied by 10 and 4 added, giving the fourth partial address, 594.

The end of the address is identified by the character *S* or *L*. The order is then assembled from the function digits and the address, the length discriminant digit being made '0' or '1' according as the order is terminated by *S* or *L*.

The assembled order is then placed in the store; the first order goes into location 31, the second into location 32, and so on. When the last order has been taken in

from the tape, the programme is started by transferring control to the order stored in location 31.

Since the number of orders is different for each programme, it is necessary to punch on to the tape some indication which will cause control to be switched to the beginning of the programme when all the orders have been taken in. With the set of initial orders now being discussed, this is done by making $T(n+1)\ S$ the first order on the tape, where n is the storage location into which the last order is to be placed. The purpose of this order is to serve as a basis of comparison with an order which is formed in the course of the input process so that it can be determined when the input of the orders from the tape is complete.

The initial orders are given in detail in appendix II, and their action can there be followed in more detail with the aid of the notes given.

5. Use of sub-routines

Programmes are usually constructed from sequences of orders called sub-routines, each of which performs one stage in the solution of the problem. As many programmes have similar sub-routines, it is convenient to have a 'library' containing sub-routines for performing such standard operations as the evaluation of a sine, or a scalar product. The preparation of long programmes can then be simplified by making use of sub-routines from the library. These sub-routines, having been previously tested, will be free from errors, so that the chance of an accidental error occurring in a long programme will be minimized. Usually it will still be necessary to construct sub-routines for performing operations which are special to the particular problem in hand.

It is not convenient, however, to use sub-routines in combination with the initial orders described above. The difficulty is best illustrated by means of an example, and we will consider a sub-routine for finding the modulus of a given number. Such a short sequence of orders scarcely warrants being called a sub-routine, and would usually be incorporated directly into a programme. It will serve, however, as an example:

Example. Place the modulus of the contents of storage location 1 in storage location 10.

location of order in store	order	interpretation of order	general sequence of events
100	*A* 1 *S*	add *C* (1) to the accumulator	add contents of 1 to the accumulator
101	*E* 104 *S*	test sign of the number in the accumulator; if positive, proceed with order 104; if negative, proceed with order 102	change sign of the number in accumulator if it is negative
102	*T* 10 *S*	transfer *C* (*Acc*) to 10 and clear accumulator	
103	*S* 10 *S*	subtract *C* (10) from the accumulator	
104	*T* 10 *S*	transfer *C* (*Acc*) to location 10 and clear the accumulator	transfer to location 10

It can easily be seen that if these orders had been placed in the store from location m onwards instead of from 100, the second order would have been $E\,(m+4)\,S$, i.e. the address specified in this order depends on the location in which the sub-routine is to be placed, whereas the addresses in the other orders do not. If the initial orders given above were used each time the sub-routine was used, it would be necessary before punching the tape to adjust appropriately all orders referring to addresses which are dependent on the location of the sub-routine.

The adjusting process follows fixed rules, and therefore the machine itself can be used to do it. The purpose of the second form of initial orders was to accomplish this in the most convenient way. As a result it is now possible to store sub-routines as strips of tape and to copy them mechanically on to any programme tape which requires them.

6. Parameters

By a parameter of a sub-routine is meant a number (often an address referred to in the course of the sub-routine) which may have different values in different applications of the sub-routine; for example, if we had a sub-routine for placing $|\,C\,(r)\,|$ in storage location s, for any values of r and s and not only for the values $r = 1$, $s = 10$ of the example already considered, r and s would be parameters of the sub-routine. It is clearly economical and convenient to be able to draft, and to punch, sub-routines in such a form that a single sub-routine can be used for different values of the parameters. For example, it is more economical to have a single sub-routine capable of finding the scalar product of n elements where n can be made to take any value from 1 to 20 as desired, rather than twenty different sub-routines.

It is desirable to distinguish between two types of parameters, pre-set and programme parameters. A programme parameter is one which is used where it is necessary to determine or to adjust its value *during the execution of a programme*; its value is inserted into the sub-routine by the machine in the course of the calculation. A pre-set parameter, which arises only from the use of a library, is one whose value is known and constant throughout the execution of any one programme, although it may vary from one programme to another. The second form of initial orders have been chosen so that the orders are adjusted according to the values of these pre-set parameters during the input of orders. The use of pre-set parameters has the advantage that the programme need not include provision for making these adjustments.

7. Initial orders—second form

These orders are now wired on the uniselectors in place of the previous ones. Orders are punched and written in the following way: first a letter to represent the function digits; secondly, the address in decimal form; and thirdly, a code letter. The code letter does not strictly form part of the order but controls the action of the initial orders.

In addition to orders it is possible to punch 'control combinations' on the tape. These have a form similar to that of orders, but are used to control and direct the action of the initial orders. They are as follows:

$T\,m\,K$	causes the orders which follow to be placed in locations m, $m+1$, $m+2$, and so on. This can be used to place groups of orders where they are needed in the store.
$G\,K$	is punched before a sub-routine and causes the address of the storage location into which the first order of the sub-routine is put to be placed in storage location 42.
$T\,Z$	causes the order which follows to be placed in storage location p, and subsequent orders in $p+1$, $p+2$, etc., where p is the address held in 42.
$E\,n\,KPF$	causes control to be transferred to location n and so starts the programme.

The method of starting the programme used here is slightly more convenient in practice than the method used with the previous set of initial orders.

The function of the code letters is to specify the location of numbers which are to be added to the orders as punched on the tape before they are placed in the store. The code letters are as follows:

code letter	integer equivalent to code letter	storage location containing number specified	value of number
F	17	41	zero
θ	18	42	$m.\ 2^{-15}$. Address of first order of sub-routine
D	19	43	2^{-16}
ϕ	20	44	these can be set to any value from the tape using control combinations, e.g. $T\,45\,K$, $P\,10\,F$†, $P\,20\,F$ cause $P\,10\,F$ to be put in 45 and $P\,20\,F$ to be put in 46, etc.
H	21	45	
N	22	46	
M	23	47	
Δ	24	48	
L	25	49	
X	26	50	
G	27	51	
A	28	52	
B	29	53	
C	30	54	
V	31	55	

F and D can be considered as corresponding directly to the S and L of the previous initial orders. If an order of a sub-routine has the code letter θ, the address specified in it will be increased by the number held in location 42 which is the address of the first order of that sub-routine. If an order has code letter H, N, M, etc., the address specified in it will be increased by the value of the content of the storage location selected by that code letter, as given by the table.

There is one additional code letter, π. This is only used in combination with one of the other code letters and is always placed directly before that code letter. It causes the associated order to refer to a long number.

These initial orders operate in the following way:

(1) The function digits are read from the tape and shifted into their correct digital position and stored temporarily in storage location 40.

† The character P is equivalent to five zeroes, so that $P\,10\,F$ is numerically equal to 10×2^{-15}.

(2) The address is formed by the same method as that used with the first form of initial orders, the end of the address being indicated by a code letter which has a numerical value greater than ten.

(3*a*) The code letter is read and tested to see whether its numerical value is greater than 16; if so, it is then used to form an order which selects the appropriate number to be added to the address of the order that is being formed. The modified address and function digits are assembled and placed in the store.

(3*b*) If the code letter has a numerical value less than 17, control is switched to the appropriate sequence of operations defined by the code letter, e.g. *K*, *Z*, π.

This process continues until a control combination is encountered when the process is adjusted accordingly. The action of these initial input orders can be followed in greater detail in appendix III.

It will be seen that with this method of forming an order from the characters punched on the tape, non-significant zeroes in the address specified in an order do not have to be punched; and further that for this purpose the address of location 0 can be treated as a non-significant zero. It is therefore omitted in writing orders; for example, *T F* is used for *T* 0 *F*.

8. Use of sub-routines with the second form of initial orders

The sub-routine previously referred to for taking the modulus of a number is given below in the notation corresponding to the second form of initial orders. The action of the machine during the input of the orders is as follows; it is supposed that they are being placed in storage locations n, $n+1$, $n+2$, etc.

entry on the input tape	action during input
G K	this control combination puts the number $n \times 2^{-15}$ in storage location 42
A 1 *F*	this is placed in location n
E 4 θ	$n \times 2^{-15}$ is added and the result $E(n+4)$ *F* is placed in location $n+1$
T 10 *F* *S* 10 *F* *T* 10 *F*	these are not modified but are placed in locations $n+2$, $n+3$ and $n+4$

Before being placed in the store these orders are converted to their binary form. We represent the orders using the new notation, code letters *F* and *D* corresponding to the values 0 and 1 respectively of the length-discriminant digit:

location	orders
n	*A* 1 *F*
$n+1$	*E* $n+4$ *F*
$n+2$	*T* 10 *F*
$n+3$	*S* 10 *F*
$n+4$	*T* 10 *F*

It will be seen that these orders have now been adjusted with reference to the location in which they were to be placed, and have been placed there. By the use of a similar example we can show how we make use of the code letters *H*, *N*, ... *V*. Suppose we need to place the modulus of the number in long storage location 200 in long storage location 100. It is unlikely that we would have such a sub-routine, but we may have a general sub-routine in the library, 'place the modulus

of the contents of the location defined by the parameter in 45 (code letter H) in the location defined by the parameter in 46 (code letter N)'. Such a library tape is given below:

T	Z	control combination	library tape
A	H	orders	
E	4 θ		
T	N		
S	N		
T	N		

GK, T 45 K, P 200 D, P 100 D, is punched on the input tape before the library tape is mechanically copied on to it. Then the input tape becomes

G	K	control combination
T	45 K	
P	200 D	punched pre-set parameters
P	100 D	
T	Z	copied from library tape
A	H	
E	4 θ	
T	N	
S	N	
T	N	

The action during the input of orders is as follows:

G	K	puts $P\,n\,F$ in 42, where n is the location of the first order of the sub-routine
T	45 K	arranges for parameters to be put in 45, 46, ...
P	200 D	is put in 45 | in the machine P corresponds to zero so that P 200 D can be con-
P	100 D	is put in 46 ∫ sidered as just an address to be inserted in an order
T	Z	arranges for the following orders to be put in n, $n+1$, etc. (n is in 42).
A	H	order becomes $(A\,F)+(P$ 200 $D)$, i.e. A 200 D, and is placed in n.
E	4 θ	order becomes $(E$ 4 $F)+(P\,n\,F)$, i.e. $E\,n+4\,F$, and is placed in $n+1$.
T	N	order becomes $(T\,F)+(P$ 100 $D)$, i.e. T 100 D, and is placed in $n+2$
		and so on.

The orders are thus put in the store in the desired form.

9. The use of sub-routines in constructing programmes

The simplest way of using sub-routines to form a programme is to arrange them in sequence in the store so that when one sub-routine is finished, the next one is commenced, and so on. Sub-routines used in this way will be called *open* sub-routines.

This method of arranging sub-routines becomes inconvenient if it is required to use the same sub-routine at more than one point of the programme; it is then necessary to switch control to and from the sub-routine at the appropriate time. In these circumstances it is more convenient to use a sub-routine into which the necessary switching order for transfer of control on completion of the sub-routine can be incorporated, so that it can be used from any point of the store, control being returned—when the calculation is completed—to a point immediately after the point from which the sub-routine was called in. This will be called a *closed*

sub-routine. There are various methods of arranging the operation of entering a closed sub-routine and returning to the main programme after the operation of the sub-routine has been completed. The method used with the EDSAC is given below; m is the address of the first order of the sub-routine.

location	order	notes
		orders calling in the sub-routine
		accumulator assumed empty
n	$A\ n\ F$	this order adds itself into the accumulator
$n+1$	$G\ m\ F$	control to m since C (Acc) is negative, as the numerical equivalent of the character A is negative
$n+2$		programme after sub-routine
		sub-routine
		accumulator contains $A\ n\ F$
m	$A\ 3\ F$	adds $U\ 2\ F$ to the accumulator forming $(U+A)(2+n)\ F$, numerically equal to the order $E\ n+2\ F$
$m+1$	$T\ m+r\ F$	transfers $E\ (n+2)\ F$ to the operational end of the sub-routine where it is used to return control to the main programme
...		accumulator empty
$m+r$	$(E\ \ F)$	becomes $E\ n+2\ F$ as a result of the operation of the order in $(m+1)$. When $(m+r)$ is reached in the sequence of operations of the sub-routine, control is switched to $(n+2)$

It should be noted that all orders in this example have been written with their true addresses, that is, in the form they take in the store, whereas on the tape they would have been punched in terms of θ, etc.

10. Use of parameters in closed sub-routines

As mentioned before, parameters associated with sub-routines greatly extend their scope. There are different places in which programme parameters can be stored, but one of the more convenient places in which to store them is in the main programme immediately following the orders calling in the sub-routine. Arranging for the parameters to be stored in the main programme leads to economy of thought, as the necessary values of the parameter to be used are considered with the 'calling-in' orders. Control is, of course, returned to the order in the main programme which follows the parameters. For example, a closed print sub-routine which prints $C(0')$ to m figures, where $P\ m\ F$ is a programme parameter specified in the main programme, would be 'called in' as follows:

location		
p	$A\ p\ F$	'calling-in' orders
$p+1$	$G\ s\ F$	
$p+2$	$P\ m\ F$	programme parameter

s is the address of the start of the print routine.

This sub-routine can find and use the parameter $P\ m\ F$ because the address specified in the order held in the accumulator on entering the sub-routine provides a reference number.

Suppose the sub-routine has been drawn up in such a way that the parameter $P\ m\ F$ is used by being added into the accumulator during the execution of the

sub-routine. Since the parameter is in location $(p+2)$, the order required for this is $A\ (p+2)\ F$. The 'link' order required to return control to the main programme is $E\ (p+3)\ F$. These orders can be formed as follows:

To $A\ p\ F$ contained in the accumulator on entering the sub-routine we add $A\ (p+2)\ F - A\ p\ F$ and so obtain $A\ (p+2)\ F$. This is placed in the appropriate position in the sub-routine.

To $A\ (p+2)\ F$ we add $E\ (p+3)\ F - A\ (p+2)\ F$ in order to obtain the link order $E\ (p+3)\ F$.

Now $A\ (p+2)\ F - A\ p\ F = (A-A)\ 2\ F$

$= P\ 2\ F$ as P corresponds to zero

and $E\ (p+3)\ F - A\ (p+2)\ F = (E-A)\ 1\ F$

$= U\ 1\ F$ (cf. §9).

These two constants are independent of p and would be included in the sub-routine.

11. Example of coding using library sub-routines

An example will now be given of the way in which a complete calculation could be coded. Suppose it is required to tabulate ψ, where

$$\cos\psi = \sin\omega\cos\delta + \cos\omega\sin\delta\sin\phi$$
$$= \sin(\omega+\delta) - (1-\sin\phi)\cos\omega\sin\delta;$$

δ varies from 0° to 23° in 1° steps; ϕ varies from 0° to 90° in 1° steps; ω is a given constant. ψ is to be tabulated in degrees and decimals.

There are various ways in which this calculation could be done; the one which is used here as an illustration is as follows:

$\sin\delta$ is calculated;
$\sin\delta\cos\omega$ is calculated;
$\sin\phi$ is calculated;
$\sin\delta\cos\omega\sin\phi - \sin\delta\cos\omega$ is calculated;
$\sin(\delta+\omega)$ is calculated;
$\cos\psi$ is calculated;
ψ is calculated in radians;
ψ is multiplied by a factor to convert it into degrees and decimals, and printed;
the current value of δ is tested to find out if it has reached 23; if it has not, the current value of δ is increased by 1° and the calculation repeated for this new value of δ; if it has, the current value of ϕ is tested. If this has reached 90° then the calculation is finished; if it has not then δ is made zero and the current value of ϕ is increased by 1° and the calculation repeated.

A list of the sub-routines required is drawn up. They are

(*a*) decimal input sub-routine; this is required to convert the decimals to binary form and place them where they are required in the store;

(*b*) sine sub-routine;

(*c*) inverse cosine sub-routine;

(*d*) print sub-routine; this converts the results to decimal form and prints them in a layout specified by values of pre-set parameters.

The index of the library is consulted and the following sub-routines selected:

(*a*) Sub-routine $R\ 5$. This is a sub-routine of special form, not conforming to either the open or the closed form. It converts decimal numbers on the input

tape into the binary form required for the machine. It is used before the main programme has been taken into the store so that the space occupied by it may be used again for other sub-routines. It is followed on the tape first by a parameter $T\ m\ D$, which specifies where the numbers are to be placed in the store, next by the numbers themselves punched in decimal form, and lastly by a control combination $X\ T\ Z$. This causes control to be returned to the initial orders so that more orders may be taken into the store.

(*b*) Sub-routine T 2. This is a closed sub-routine. It causes $\frac{1}{2}\sin[2\,C\,(4')]$ to be placed in long storage location 4, $C\,(4')$ being an angular measure in radians. It occupies 42 storage locations, and its first order must be placed in an even location.

(*c*) Sub-routine T 5. This is a closed sub-routine. It causes $\frac{1}{2}\cos^{-1}[2\,C\,(4')]$ (radians) to be placed in long storage location 4. It occupies 58 storage locations and its first order must be placed in an even storage location.

(*d*) Sub-routine P 4. This is a closed sub-routine. It causes the positive number (which must be less than 1) held in long storage location 0 to be printed to r decimals, the numbers being arranged in c columns with s spaces between them; r, c, and s are pre-set parameters of the sub-routine and the values to be given to them in any particular application of this sub-routine are specified as follows. Suppose 12 columns, two spaces between columns, and 4-figure entries are required. Then

$$\begin{array}{ll} G & K \\ T\ 45 & K \\ P\ 12 & F \\ P\ 2 & F \\ P\ 4 & F \end{array}$$

is punched on the tape directly in front of this sub-routine. It occupies 55 storage locations.

12. The coding of the main programme

The next step is to plan and code the main programme. In order to do this, an indication is required for the locations to be used for numbers and constants. It is not convenient to fix the actual locations until later, and for this reason the code letter H will be used in orders referring to these numbers and constants. Then the actual locations may be fixed later by assigning the appropriate numerical value to the parameter m to be used in instructions with code letter H. The initial values of all these numbers are placed in their respective storage locations by the decimal input sub-routine, as follows:

location	number	
$m+0$	current value of $\frac{1}{2}\delta$	specified in radians
2	current value of $\frac{1}{2}\phi$	
4	end value of $\frac{1}{2}\delta$, $11\frac{1}{2}°$*	
6	end value of $\frac{1}{2}\phi$, $45°$*	
8	$\cos\omega$	
10	$\frac{1}{2}\omega$	
12	$\frac{180}{\pi}\times\frac{1}{100}$	a constant which converts radians to hundreds of degrees

* In practice the values stored here would be $11\frac{3}{4}°$ and $45\frac{1}{4}°$ to ensure the correct number of repetitions of the calculation independently of rounding errors.

Suppose that the starting location of the sine sub-routine is indicated by the code letter N, that of the inverse cosine sub-routine by M, and that of the print sub-routine by Δ. These values can also be fixed later. The location of the main programme in the store is also undetermined for the present; it is therefore coded just like a sub-routine, with the code letter θ for those orders in which the address has to be adjusted in accordance with the address into which the first order of the programme is placed. In the explanation, 'to 10'' will mean 'transfers to long storage location 10'.

number of order in main programme	order	action of orders	general sequence of events
	$T\ \ Z$	control combination on the tape	
0	$T\ \ F$	clears the accumulator	
1	$A\ \ H$	$\frac{1}{2}\delta$ to 4′	calculates $\frac{1}{2}\sin\delta$
2	$T\ 4\,D$		
3	$A\ 3\,\theta$	calls in the sine routine; results in $\frac{1}{2}\sin\delta$ to 4′	
4	$G\ \ N$		
5	$H\ 4\,D$	$(\frac{1}{2}\sin\delta)\cos\theta$ to 10′	calculates $\frac{1}{2}\cos\theta\sin\delta$
6	$V\ 8\,H$		
7	$Y\ \ F$		
8	$T\ 10\,D$		
9	$A\ 2\,H$	$\frac{1}{2}\phi$ to 4′	calculates $\frac{1}{2}\sin\phi$
10	$T\ 4\,D$		
11	$A\ 11\,\theta$	$\frac{1}{2}\sin\phi$ to 4′	
12	$G\ \ N$		
13	$H\ 4\,D$	$2\,(\frac{1}{2}\sin\phi)\,(\frac{1}{2}\cos\theta\sin\delta) - \frac{1}{2}\cos\theta\sin\delta$ to 10′	calculates $\frac{1}{2}\cos\psi$
14	$V\ 10\,D$		
15	$L\ \ D$		
16	$S\ 10\,D$		
17	$Y\ \ F$		
18	$T\ 10\,D$		
19	$A\ 10\,H$	$\frac{1}{2}(\omega+\delta)$ to 4′	
20	$A\ \ H$		
21	$T\ 4\,D$		
22	$A\ 22\,\theta$	$\frac{1}{2}\sin(\omega+\delta)$	
23	$G\ \ N$		
24	$A\ 4\,D$	$\frac{1}{2}\cos\psi$ to 4′	calculates $\frac{1}{2}\psi$
25	$A\ 10\,D$		
26	$T\ 4\,D$		
27	$A\ 27\,\theta$	calls in inverse cosine routine and results in $\frac{1}{2}\psi$ to 4′	
28	$G\ \ M$		
29	$H\ 4\,D$	expresses the angle in units of 100°, so that its measure is less than 1. The first two digits of the number printed will represent the integral number of degrees in the angle, the other digits represent the fraction of a degree contained in that angle	
30	$V\ 12\,H$		
31	$L\ \ D$		
32	$Y\ \ F$		
33	$T\ \ D$		
34	$A\ 34\,\theta$	calls in the print sub-routine and prints the number in 0′	
35	$G\ \ \Delta$		
36	$A\ \ H$	tests to see if the current value of δ has reached $11\frac{1}{2}°$. If it has, switches control to 44θ	
37	$S\ \ 4\,H$		
38	$E\ 44\,\theta$		
39	$T\ \ F$	clears the accumulator	
40	$A\ \ H$	increases $\frac{1}{2}\delta$ by $\frac{1}{2}°$	
41	$A\ 14\,H$		
42	$T\ 14\,H$		

number of order in main programme	order	action of orders
43	*E* *θ*	starts calculation again
44	*T* *F*	reached from 38 if current value of δ has reached $11\frac{1}{2}°$. Clears accumulator
45	*A* 2 *H*	tests if current value of ϕ has reached 90°; if it has, switch control to 54
46	*S* 6 *H*	
47	*E* 54 *θ*	
48	*T* *F*	clears the accumulator
49	*A* 2 *H*	increases $\frac{1}{2}\phi$ by $\frac{1}{2}°$
50	*A* 14 *H*	
51	*T* 2 *H*	
52	*T* *H*	makes (current value of δ) = 0
53	*E* *θ*	re-starts calculation for next value of ϕ
54	*Z* *F*	stop order; reached from 47 if current value of ϕ is 90°

The next step is to decide on which positions the various sub-routines and numbers will occupy in the store. When this has been done the programme tape can be constructed.

Let the locations of the sub-routines be chosen as follows: Input sub-routine *R* 5 starts at location 60. Sine sub-routine *T* 2 starts at location 60 (this goes into the storage locations previously occupied by *R* 5). Since this occupies 42 storage locations it ends at 101. The inverse cosine sub-routine must start at an even location; let it start at 102; it then ends at 159. Print sub-routine *P* 4 starts at storage location 160 and ends at 214. The main programme starts at storage location 215 and ends at 269, and the numbers start at long storage location 270. Thus the parameter for the input routine must be *T* 270 *D* and the main programme must have the following symbols punched in front of it on the input tape:

G *K*	
T 45 *K*	
P 270 *D*	used by code letter *H*
P 60 *F*	used by code letter *N*
P 102 *F*	used by code letter *M*
P 160 *F*	used by code letter Δ

At the end of the main programme the control combination *E* 215 *K*, *P F* is required so that after taking in the whole programme the machine will return to the start of the main programme at address 215 to start the calculation itself.

The programme tape is then punched as follows:

storage locations to be used	tape entries	notes
	T 60 *K*	control combination to ensure that the first order of the first sub-routine is placed in 60
60*	input sub-routine *R* 5	copied from library tape
	T 270 *D*	parameter used by *R* 5
	numbers and constants	placed by *R* 5 in 270′, 272′, etc.
60 – 101	sine sub-routine *T* 2	copied from library tape
102 – 156	inverse cosine sub-routine *T* 5	copied from library tape

* Temporarily, while input of numbers is taking place.

storage locations to be used	tape entries	notes
	G K *T* 45 *K* *P* 270 *D* *P* 60 *F* *P* 102 *F* *P* 160 *F*	sets parameters for main programme
215 269	main programme	
	E 215 *K* *P F*	control combination to start the programme

Appendix I. EDSAC order code

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	Transfer 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 into the accumulator.
N n	Multiply the number in storage location *n* by the number in the multiplier register and subtract 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.
R $2^{n-2}F$*	Shift the number in the accumulator *n* places to the right; i.e. multiply it by 2^{-n}.
L $2^{n-2}F$*	Shift the number in the accumulator *n* places to the left; i.e. multiply it by 2^{n}.
E n	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	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 tape and place the resulting 5 digits in the least significant places of 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 next to be printed by the teleprinter in the five most significant places in storage location *n*.
Y	Round off the number in the accumulator to 34 binary digits.
Z	Stop the machine and ring the warning bell.

* For a shift of one place right or left the forms of the order are *RD*, *LD* respectively (with first form of initial orders, *F* and *D* are replaced by *S* and *L*).

Appendix II. Initial orders 1

When the starting button is pressed these orders are placed in the store in locations 0 to 30 and control is set so that the first order obeyed is in location 0.

location of order	order	action of order	general sequence of events
0	*T S*	clears accumulator after transferring *C* (*Acc*) to location 0	
1	*H* 2 *S*	places *C* (2) in multiplier register. This *T S* is 10/32 numerically	clears accumulator and puts 10/32 in multiplier register
2	*T S*	clears accumulator	
3	*E* 6 *S*	examines *C* (*Acc*). As this is zero, control is transferred to the order in location 6.	control switched to 6. These locations 0–3 are then used as 'working space'
4	*P* 1 *S*	2^{-15}	constants
5	*P* 5 *S*	10×2^{-16}	
6	*T S*	clears accumulator	input of function digits to their correct digital location in 0
7	*I S*	input of function digits to location 0	
8	*A S*	adds function digits into accumulator	
9	*R* 16 *S*	shifts *C* (*Acc*) six places to the right so that the function digits are in the five most significant places of the least significant half of the accumulator	
10	*T L*	transfers *C* (*Acc*) to *long* location 0 so the function digits are in their correct positions in short storage location 0	
11	*I* 2 *S*	input of next character on the tape to storage location 2	reads character on the tape and tests whether it is numerically less than 10
12	*A* 2 *S*	adds *C* (2) to *C* (*Acc*)	
13	*S* 5 *S*	subtracts *C* (5) from *C* (*Acc*)	
14	*E* 21 *S*	if *C* (*Acc*) $\geqslant 0$, control is transferred to order in location 21	
15	*T* 3 *S*	clears the accumulator using 3 as a rubbish dump	
16	*V* 1 *S*	*C* (1), multiplied by 10/32 which is held in multiplier register, is added to the accumulator	one stage of the binary decimal conversion. New partial address is obtained by taking 10 times old partial address and adding the next digit
17	*L* 8 *S*	shifts *C* (*Acc*) five places to the left, i.e. *C* (*Acc*) is multiplied by 32	
18	*A* 2 *S*	adds *C* (2) to *C* (*Acc*)	
19	*T* 1 *S*	transfers *C* (*Acc*) to storage location 1	
20	*E* 11 *S*	As *C* (*Acc*) $= 0$, control is transferred to the order in location 11	
21	*R* 4 *S*	shifts *C* (*Acc*) four places to the right	control is transferred to 21 from the order 14 when character read is *S* or *L*. When *L* has been read the 17th digit of the accumulator is a 1, when *S* has been read it is a 0
22	*A* 1 *S*	adds *C* (1) to *C* (*Acc*)	the address has been formed $\times 2^{-16}$ and so needs doubling
23	*L L*	shifts *C* (*Acc*) one place to the left	
24	*A S*	adds *C* (0) to *C* (*Acc*)	forms the complete order in accumulator
25	*T* 31 *S*	transfers *C* (*Acc*) to 31. The address specified in this order is increased by 1 each time the order is used and it places the constructed orders in sequence in the store	transfers the order to its final position in the store
26	*A* 25 *S*	adds *C* (25) to *C* (*Acc*)	increases the address specified in order 25 by 1; e.g. *T* 31 *S* is replaced by *T* 32 *S* and so on
27	*A* 4 *S*	adds *C* (4) to *C* (*Acc*)	
28	*U* 25 *S*	transfers *C* (*Acc*) to 25 without clearing accumulator	
29	*S* 31 *S*	subtracts *C* (31) from *C* (*Acc*)	tests whether all orders have been taken in. Location 31 contains order *T* (*n*+1) *S*, the first order to be placed in the store: and so *C* (*Acc*) will be positive only when all orders have been taken into the store
30	*G* 6 *S*	if *C* (*Acc*) $\leqslant 0$, control is transferred to the order in location 6	
		end of initial orders	
31	*T* (*n*+1) *S*		the first order to be placed in the store

Appendix III. Initial orders—second form

When the starting button is pressed these orders are placed in the store locations 0 to 40 and control is set so that the first order obeyed is in location 0. In the notes b stands for the numerical equivalent of the character last read from the tape.

location of order	order	action of order	general sequence of events
0	*T F*	clears accumulator using 0 as a 'rubbish bin'	the accumulator is cleared and control is switched to 20; thereafter storage locations 0, 1 are used as working space
1	*E* 20 *F*	$C(Acc)=0$ so that control is switched to order 20	
2	*P* 1 *F*	2^{-15}	constants intended to be left here unaltered for use in any programme
3	*U* 2 *F*	not used by initial orders	
4	*A* 39 *F*	adds 11×2^{-16} to $C(Acc)$	construction of the address. These orders are entered at 8 with $C(Acc)=0$. The next character, represented by b on the input tape is read and tested to see if it is less than 11; if so it is doubled and added to ten times $C(0)$, the sum being sent back to 0. This cycle will be repeated until a code letter is read from the tape, when the sequence of orders starting at 13 is carried out
5	*R* 4 *F*	shifts $C(Acc)$ four places to the right, i.e. divides $C(Acc)$ by 16	
6	*V F*	multiplies $C(0)$ by contents of multiplier register which is holding 10/32	
7	*L* 8 *F*	shifts $C(Acc)$ five places to the left, i.e. multiplies $C(Acc)$ by 32	
8	*T F*	transfers $C(Acc)$ to 0 and clears Acc	
9	*I* 1 *F*	input of next character (b) to storage location 1	
10	*A* 1 *F*	adds $b\times2^{-16}$ to $C(Acc)$	
11	*S* 39 *F*	subtracts 11×2^{-16} from $C(Acc)$	
12	*G* 4 *F*	control is transferred to order 4 if $b<11$	
13	*L D*	multiplies $C(Acc)$ by two $C(Acc)=(2b-33)\,2^{-16}$	an order $A\,(24+b)\,F$ is formed if $b>16$. An order $E\,(16+b)\,F$ is formed if $b<17$. This formed order is placed in 20
14	*S* 39 *F*	subtracts 11×2^{-16} from $C(Acc)$ $C(Acc)=(2b-33)\,2^{-16}$	
15	*E* 17 *F*	control is transferred to order 17 if the $C(Acc)\geqslant0$	
16	*S* 7 *F*	subtracts $C(7)$ from $C(Acc)$	
17	*A* 35 *F*	adds $C(35)$ to $C(Acc)$ $C(Acc)=E\,(16+b)\,F$ if $2b<33$ $=A\,(24+b)\,F$ if $2b>33$	
18	*T* 20 *F*	transfers $C(Acc)$ to 20	
19	*A F*	adds the address constructed in storage location 0 to $C(Acc)$	adds address of order to $C(Acc)$
20	*H* 8 *F*	puts $C(8)$ equal numerically to 10/32, in multiplier register. This order only used during start it later becomes	
	A (24+*b*) *F*	adds $C(24+b)$ to $C(Acc)$	adds selected number to $C(Acc)$
or	*E*(16+*b*) *F*	transfers control to $(16+b)$ as $C(Acc)\geqslant0$	*or* switches control to order $16+b$
21	*A* 40 *F*	adds $C(40)$ to $C(Acc)$	adds function digits of order to $C(Acc)$ *or* during the start adds *PD* from 40
22	*T* 43 *F*	transfers the order in the accumulator to its place in the store. This order later becomes	assembled order to its location in the store
	T n F		
or	*E m F*	switches control to begin obeying the programme—after EmKPF	
23	*A* 22 *F*	adds $C(22)$ to $C(Acc)$	increases the address specified in order 22 by unity
24	*A* 2 *F*	adds $C(2)$ to $C(Acc)$	
25	*T* 22 *F*	transfers $C(Acc)$ to 22	
26	*E* 34 *F*	since $C(Acc)=0$ control is transferred to order 34	control to 34
27	*A* 43 *F*	adds 2^{-16} to $C(Acc)$	control is switched to order 27 by the order in 20 when the code letter π has been read from the input tape. (π corresponds to 11.) These add 2^{-16} to the address which is in the accumulator and return control to order 8. This order puts back the address in 0, which has thus been increased by 2^{-16}
28	*E* 8 *F*	since $C(Acc)$ is positive, control is transferred to order 8	

location of order	order	action of order	general sequence of events
29	*A* 42 *F*	adds *C* (42) to *C* (*Acc*)	control switched here by the code letter *Z*. This adds *C* (42) to the address in accumulator
30	*A* 40 *F*	adds *C* (40) to *C* (*Acc*)	control is switched here by code letter *K*. The function digits from 40 are now added to *C* (*Acc*)
31	*E* 25 *F*	if *C* (*Acc*) is positive, switch control to order 25; if negative proceed with next order	the accumulator is negative if the code letter *K* was preceded by *G*. It is positive if *K* was preceded by a *T* or an *E*, and control is transferred to 25 when the order held in the accumulator is placed in location 22
32	*A* 22 *F*	adds *C* (22) to *C* (*Acc*)	the accumulator holds *GF* at this point and order 22 (*TnF*) is added to it; as *G* is the complement of *T*, *PnF* is transferred to 42. Note *P* = 0 and so just the address is put in 42
33	*T* 42 *F*	transfers *C* (*Acc*) to 42, clearing the accumulator	
34	*I* 40 *D*	reads next row of holes from the input tape and places them in long location 40	these orders place the function digits in their correct position in 40: control is switched to 8 (see initial orders I). C (35) also used as a constant
35	*A* 40 *D*	adds *C* (40′) to *C* (*Acc*)	
36	*R* 16 *F*	shifts *C* (*Acc*) six places to the right	
37	*T* 40 *D*	transfers *C* (*Acc*) to 40′ and clears accumulator	location 40 now contains the function digits, and 41 contains 0
38	*E* 8 *F*	since *C* (*Acc*) is zero, control is transferred to order 8	
39	*P* 5 *D*	11×2^{-16}	constants. *PD* is transferred to 43 during the start and location 40 thereafter used as working space
40	*P* *D*	2^{-16}	

The author wishes to thank Dr M. V. Wilkes and Professor D. R. Hartree for their many helpful suggestions, and also the Department of Scientific and Industrial Research for a maintenance grant.

References

Wilkes, M. V. 1949 *J. Sci. Instrum.* **26**, 217.
Wilkes, M. V. & Renwick, W. 1949 *J. Sci. Instrum.* **26**, 385.
Wilkes, M. V. & Renwick, W. 1950 *M.T.A.C.* **4**, 61.