## 4. The Fixed Orders

### 4.1 Introduction

It was shown in 3.4.7 that when reading information from locations with addresses in the range 0 to 95 there are two sets of locations from which the information can come. It is important to understand that this duplication of the first 96 locations only concerns read operations. When writing information into the store, the normal order store is always used. The second set of 96 locations from which information can be read is the fixed order store. At any one time only one of the two possible sets of locations 0 to 95 are available for fetch operations and the execution of a 27 order removes the current set and makes the other set available. The computer is started by pressing the Initial Start Button. This causes the arithmetic and control unit registers to be cleared, with the exception of the 7-B-registers which are undisturbed. It also assigns the fixed order store to be available for fetching operations. On pressing the GO button, the computer will fetch and obey instructions in sequence starting from the location specified by the control counter. As one of the effects of the Initial Start Button was to set the control counter to zero, the first instruction to be obeyed will be taken from location 0 of the fixed order store. This is the first instruction of the fixed order programme which provides a means whereby instructions punched in decimal character form are read in, converted to binary instruction form, and stored in sequence starting from a specified load point.

Instructions are punched onto 5 channel paper tape by means of a Creed Teleprinter and have the following form

F.B.N

followed/

followed by either a space character or by carriage return + line feed characters. F is an unsigned integer in the range

$$0 \leq F \leq 63$$

representing the function number of the instruction. B is an unsigned integer in the range

$$0 \leq B \leq 7$$

representing a modifier address. If B is zero the instruction can be written as

F.N

N is a signed or unsigned integer as follows

signed $-1024 \leq N \leq 1023$

unsigned $0 \leq N \leq 2047$

During the formation of an instruction by the fixed orders, N is formed as a 20 digit integer and added to the partially formed instruction which is the binary equivalent of

F.B.0

Thus if N is negative it will have the effect of subtracting 1 from the bottom digit of the B part of the binary instruction. The following two examples demonstrate this

| On Tape In Decimal | In Solidac In Binary |
|---|---|
| 5.3. - 1 | 5.2. -1 |
| 7.0. - 3 | 6.7. -3 |

When punching instructions in the above forms any number of spaces are allowed between . and B and . and N. Spaces are not allowed between F and . and B and . . The initial orders will also read in data in the form of signed integers in the following range

$$-524288 \leq n \leq +524287$$

If an integer is positive the + sign may be omitted.

## 4.2 Directives and Items

In the following it will be useful to denote "carriage return", "line feed" and "space" characters by cr, lf and sp respectively:

Instructions formed from punched paper tape via the initial orders can either be placed in the store as part of a programme or they can be used as a means of providing control over the execution of the initial orders. This latter form of instruction is called a directive. A directive differs from a normal instruction in that it is immediately followed by an = character. Thus a directive is printed in the form

F. B. N =

Integers and instructions which are not directives, are called items. Items usually occur in strings and each item is terminated by cr, lf or sp. A string of items may be preceeded by cr, lf or sp. A directive must precede a group of items, and the directive instruction is obeyed each time an item has been read in and converted to binary and its terminator, i.e. sp or cr lf, is recognised. When the directive is being obeyed, the item which was last read is held in the M-register. Each time the directive is obeyed 1 is added to its address part and it is held ready to be obeyed when the next item has been converted and its terminator recognised. These facilities have been included so that the following instruction can be used as a directive for storing programme instructions in sequence from a given load point:-

40. n =

Following this directive are the instructions which form the programme being read. Each instruction is an item which must therefore be terminated by cr lf or sp. The instructions of the programme are stored in consecutive locations starting from n. When the last item of a programme has been read in it remains/

remains to enter the programme at the required point. This is usually done by the following directive and item

27: m = 0 cr lf

The instruction 27.m is a directive as it is followed by an = character. 0 is an item as it is followed by cr lf. On reading and recognising the terminator cr lf the initial orders would cause the instruction 27.m to by obeyed. The computer would then prepare to obey the instruction stored in location m while at the same time making the first 96 words of normal order store available for read operation. Example of a programme tape with directives:

| Instruction | Comment |
|---|---|
| 40. 0 = | Directive, store programme from 0 onwards. |
| 5.1.10 | Set B1 = 10 |
| 20.1.3 | Punch C(3 + B1) |
| 14.1.1 | Count and jump |
| 0.1 | Stop |
| 27.0 = 0 cr | Directive, enter programme at location 0 |

The programme would be enetered in the store as follows.

| Location | Contents |
|---|---|
| 0 | 5.1.10 |
| 1 | 20.1.3 |
| 2 | 14.1.1 |
| 3 | 0.1 |

## 4.3 Labels and Relative Addresses

### 4.3.1 Relative Addresses

Up to this point a programme once written can only be obeyed provided that it is entered in the correct part of the store. Hence a programme written to fit one part of the store would differ from the same programme written for another part as, for example, the addressed used in jump instructions within the programme would be different in each case. The initial orders provide a facility which allows the programmer to write his programmes using a relative addressing system, so that at writing time the absolute values of store locations are never used. When it is required to enter the programme in the store an absolute address is used in the store directive which preceeds the programme. On reading the instructions of the programme, the initial orders can then convert relative addresses to absolute addresses where necessary. The relative addressing system allows the programmer to refer to storage locations as being a certain number ahead of, or behind, the present instruction. The figure shift symbol v is used in the address part of an instruction when the relative addressing facility is required e.g. theinstruction.

26. -3v

will cause the computer to jump to the 3rd instruction behind this one, and the instruction

1. 1. 56v

will fetch the contents of the location 56 ahead of it to B1.
Using this facility it is seldom necessary to use absolute addresses when writing a programme.
The relative address facility is provided in the following way. As an instruction is being read from tape and converted to binary the fixed ord[illegible] programme/

programme has available the absolute address into which the instruction is to go. (This is supplied by adding 1 to the store directive address each time an instruction is stored); When a v is encountered this address is added to the bottom part of the instruction being formed. If the instruction being formed is

26. -3v

which is to be placed in location 624 say, then the instruction is first formed as

26. -3

and when the v is encountered, 624 is added to its address thus giving

26. -3 + 624 or 26. 621

as the final instruction to be stored.

4.3.2 Labels

It is possible to write instructions in the following general form

L) F. B. N

where L is an integer greater than zero. The label L may then be referred to in SUBSEQUENT instructions by using the figure shift symbol v followed by the integer L, e.g.

26. vl

will cause a jump to the instruction labelled 1.

The initial orders supply this facility as follows. On encountering the label L) preceeding an instruction the initial orders store the address of the instruction in label store L, whose actual address in the normal order store is 100 + L. When the label is used in an instruction as vL the initial orders then add the contents of 100 + L to the instruction. Labelling is therefore an extension of the relative addressing facility, and it is possible to refer to instructions relative to a labelled instruction, e.g.

26. -3v2

will cause a jump to the instruction 3 behind the instruction which is labelled 2.

4:4 · Using the Initial Orders Programme

The initial orders provide a convenient means of entering programme in the core.store. It is possible to include a title with a programme. A title is defined to be a series of characters beginning with TWO "letter shift" characters and ending with "line feed". Titling can appear in front of, or between, items and directives. A record of titles is punched onto paper tape when a programme is being read by the initial orders.

The following is an exception to the rules for punching instructions which were described in the previous sections:- It is not possible to use any of the following instructions

32. -N

32. -Nv

32. -NvM (N, M integers)

These will all cause permanent overflow to occur in the initial orders programme. It was shown in section 4.2 that the directive

40. N =

is useful for storing a programme from location N onwards. Due to the necessity of including this on the same tape as the programme, the load point N must be decided before run time. The following device allows the selection of a load point from the Hand Switches at run time:-

61. 99 = 0

40. -1v=

This works as follows. The normal order location 99 is used to hold the current directive and when this directive is obeyed 1 is added to its address part and the resulting address is copied into location 100. Location 100 therefore contains the address/

address into which the next item is to go when the directive in use is

40. .N=

If a v is encountered in an instruction the contents of location 100 are added to the address of the instruction as described in section 4.3.1. The pair of directives above make use of this facility. When the directive

61. 99

is obeyed the Hand Switch setting (N, say), is read into location 99, 1 is added to it and the result is copied into Location 100. When the v in the second directive is encountered the contents of location 100 are added to the instruction being formed. Hence the second directive is formed as

40. -1 + N + 1

or

40. N

This directive will then store the items which follow it, starting at location N. A further facility provided by the initial orders is that of clearing the core store locations prior to reading a programme. This is done by entering the initial orders programme at location 50.

A useful set of directives with which to start a programme is

| | | | |
|---|---|---|---|
| 26. | 50 | = 0 | clear store |
| 0. | 1 | = 0 | stop to allow H/S to be set |
| 61. | 99 | = 0 | Read H/S to directive store |
| 40. | -1V | = | Store as per H/S setting |

An extension to the relative addressing facilities of the initial orders is provided by the use of the "figure shift" symbol n, which can be used in any position which is legitimate for v. The effect of using n is to subtract the address/

address of the current location from the address part of the instruction being formed.

{manually transcribed replacement for illegible page}

4.5 The Use of Normal Store by the Initial Orders Programme.

The initial orders use certain normal store locations for holding data and intermediate results required in the formation of items and directives. They are as follows

| Store Location | Use |
|---|---|
| 96 | Character store |
| 97 | integer store |
| 98 | item store |
| 99 | directive store |
| 100 | location counter |
| 101 | store for address of label 1 |
| 102 | store for address of label 2 |
| .. | store for address of label .. |
| .. | store for address of label .. |
| 100 + L | store for address of label L |

It is most important to ensure that these locations are not overwritten by the items of the programme being read in.

A block diagram, and the instructions of the initial orders programme are given in the Appendix.

## 4.6 Subroutines.

It is often required that the same group of instructions are to be obeyed at different points in the same programme. It would be wasteful of store to include these instructions at every point they are required; instead, the use of a sub-routine means that the set of instructions need only be included once.

A subroutine requires two types of data

a) The data upon which the instructions are to operate in order to produce the result.

b) The address to which control is to be transferred after the instructions in the subroutine have been obeyed.

This second type of information is called the Link. In the event that there are several possible exits from a subroutine more than one link may have to be supplied. The following example shows how entry may be made to a subroutine from a programme.

```
                    )
                    )  instructions of programme
                    )

  5.  1.  2V        store address of two locations ahead (link)
 26.      V4        Jump to subroutine labelled 4

                    )
                    )  instructions of programme
                    )
```

The subroutine would have to have the following form

```
  4)                )
                    )
                    )  instructions of subroutine
                    )
                    )

 26.  1.  0         Jump to location specified by B1 (link)
```

The following points must be noticed:-

a)/

a) The subroutine instructions would have to appear before the main programme so that the use of the label 4 in the instruction

```
26.  v4
```

would be valid.

b) The instructions of the subroutine must not destroy the contents of B1 which contains the address at which the main programme is to be resumed. If B1 were required as a working register by the subroutine then the subroutine could be modified as follows

```
4)  1.  1.  Nv      store link N instructions ahead
                  )
                  )
                  )  Instructions of subroutine (N-1 in number)
                  )
26.  0              Jump as per link.
```

Note that the instruction

```
1.  1.  Nv
```

stores the contents of B1 in the bottom 11 digits of the instruction

```
26.  0
```

and the top digits of this location are unaltered.

## 5. Operating Instructions

### 5.1 Description of Operator's Console

The appendix contains a diagram of the console. The following explains the symbols, indicators and controls which appear.

Supply Switch and Indicator - In the SUPPLY ON position the switch provides power to all computer units. On first switching on, both lights should be ON until the IS button is pressed. Thereafter the POWER FAILURE light should remain OFF. After IS has been pressed a power failure is indicated by the POWER FAILURE light coming ON.

Pulse Repitition Frequency Switch - This controls the master oscillators of the computer. In the NORM (normal) position the master oscillator produces 56000 pulses per second. In the SLOW position 2 pulses per second are produced. This enables the slowed-down operation of the computer to be seen by means of the monitor switches and indicator lights. In the single position pulses are produced by means of the SINGLE DIGIT button.

B-Register and Control Counter Monitoring - This is accomplished by means of a set of 11 indicators and a monitoring switch on the top right hand side of the centre panel.

Monitoring of Other Registers - This is done using the switch below the B-register monitor switch in conjunction with the 20 indicators across the centre of the console. As the L register has only 19 digits the twentieth (leftmost) digit is redundant when the switch is at L.

Overflow Monitoring - As there are only 20 indicators on the front panel the overflow digit of the M,S and D registers are not displayed. An overflow indicator is situated on the left of the dial which indicates the condition of overflow in the register/

register selected by the monitor switch. The OFF light is ON when the contents of the selected register are in the temporary overflow condition.

Store Indicators - These indicate that either the fixed or normal order store locations are available for read operations using addresses 0 to 95.

Hand Switch/Normal Control - This provides a means whereby instructions set up in binary on the hand switches may be obeyed. If instructions are obeyed from hand switches (switch in HS position) the normal process of incrementing the control counter by 1 at the end of an instruction is inhibited. Jump instructions are obeyed correctly from hand switches. Instructions are normally obeyed from hand switches with the computer in the "single-shot" mode.

IS Button and Indicator - By pressing the IS (initial start) button the M, L, D, S, C, V and C.C registers are set to zero and the fixed order store is made available. The computer is then ready to obey the initial orders programme starting at location 0 of the fixed order sotre. After the first IS has been given the POWER FAILURE light should be OFF. Notice that IS does not clear the B-register.

GO Button and Indicator - The GO button provides a means of starting the computer when it is stopped and no failure indication is present. Normal use of the GO button is encountered after depressing IS in which case GO causes the initial orders programme to be obeyed. With the computer in the "single-shot" mode the GO button may be used to obey instructions one at a time. If the computer has been stopped by an optional stop or normal stop instruction, pressing the GO button will cause the machine to be restarted.

Single Shot Switch and Indicator - It is possible to obey instructions one at a time by setting the SS/NORM switch to SS. This stops the computer at the end of the current instruction and it can then be restarted for one instruction by pressing the/

the GO button. By putting the SS/NORM switch back to NORM and pressing the GO button the computer will obey instructions continuously. If the computer has arrived at a stop of any kind the Control Register(C) will contain the last instruction to be obeyed and the Control Counter (CC) will contain the address of this instruction. When the GO button is pressed 1 is added to the Control Counter and so the next instruction is selected.

Optional Stop Switch and Indicator - It was mentioned previously that instructions of the form

0. B. N

would only cause the computer to stop under certain conditions. If the switch is in the OST position, optional stops will be operative otherwise the optional stop instructions will have no effect. The optional stop facility is often of use in programme testing.

Stops - Eight indicator lights are grouped together as the STOPS indicators. If for any reason the computer should stop then one or more of the STOPS lights will be ON. The conditions under which each light comes on are as follows -

SS - Single Shot switch is up.

OST - Optional Stop switch is UP and an optional stop has been obeyed.

WAIT - After IS and Before GO has been pressed, or after obeying a normal stop instruction, or because any other stop has occurred.

ABS - After the instruction

0. 0. 0

has been obeyed, or after an instruction whose function number is unassigned has been obeyed, or POF, or OS, or TOF has occured.

IO - After an input/output failure has occurred.

OS/

OS - An instruction which does not use function number 24 follows overshift by a normalise instruction.

TOF - An illegitimate instruction has been obeyed when temporary overflow is indicated in M, S or D.

N.B. When temporary overflow exists the only legitimate instructions are those which use the following function numbers:- 35, 36, 37, 38, 39, 25, 16, 17.

POF - Permanent overflow (loss of top digits) has occured in M, S or D.

If the ABS light in ON the computer has stopped because of an error which cannot be rectified. It is possible to continue with the programme under these conditions by pressing the OR (over-ride) button. The OR light will then come ON to indicate that the computer has previously encountered a major error.

The Dial - This provides a means of entering information into the M-Register. The Dial is intended to work under programme control and it may only be used legitimately after a normal or optional stop has been obeyed. If this is so, then dialling the digit 5, for example, will add 5 in binary to the current contents of the M-register and provide a GO signal after this has been done. Dialling the digit 0 will cause 10 to be added to the M-register.

Input-Output Indicators - At present only input channel 1 and output channel 0 are used. The TAPE READY lights are ON when the input/output device is in a condition to accept an input/output transfer. The TAPE JAMMED light ON indicates that an input/output transfer has been initiated but after a reasonable time the input/output device has not performed the transfer. The NO TAPE indicator on the output channel will come ON when the amount of tape available is less than a certain minimum. The NO TAPE and TAPE JAMMED indicators all cause an IO stop on the computer. The computer may be restarted after a NO TAPE, I/O stop by pressing the NO TAPE OVER-RIDE button.

## SOLIDAC INITIAL ORDERS

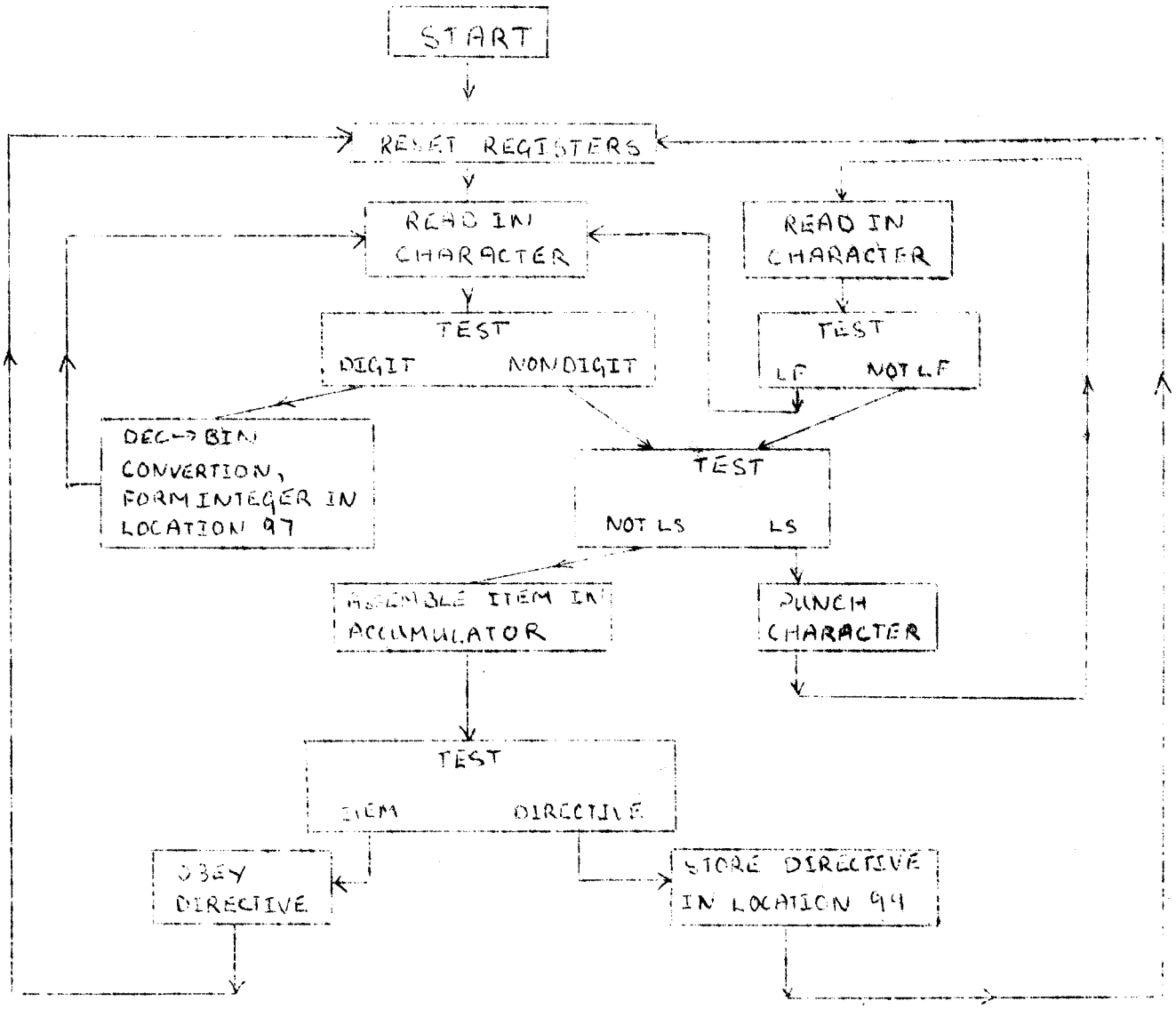
```
 0     55. 10     D : = 10;
 1      5.2.10    B2 : = 0;  Initial Shift;
 2      5.3.15    B3 : = 15;  FOR PLUS (and 1064);
 3      5.4. 0    B4 : = 0;  NON-ARITH, NON-REFERENCE;
 4     40. 98     Store or clear item store;
 5     31. 97     Store or clear integer store;
 6     10.1.96    READ to B1 and to character store;
 7     32. 35     Put part of digit pattern into L;
 8     43. 64     Add other part into M;
 9     38.1. 0    Shift the pattern as per B1;
10     22. 31     Jump if not a digit;
11     56. 97     A  : + 10 x previous integer;
12      8.1. 2    and out the parity bit:
13      1.1.96    character store : = true digit;
14     33. 96     Added to L;
15      6.4. 1    Add to ensure ARITH (B4 ≠ 0):
16     26.  5     Jump to store new integer;
17     12.1. 6    Jump to READ for Er = 31;
18      6.1.20    Add for   =   = 10;
19     13.1.22    Jump if not   =   ;
20     41. 99     directive store  : = A;
21     26. 47     Jump to collate directive address;
22     14.1.26    Jump if not   -   = 11;
23      5.3. 0    B3  : = 0 for PLUS;
24     13.4.42    Jump to stop if ARITH;
25     26.  4     Jump to store zero or re-store:
26     14.1.28    Jump if not   v   = 12;
27     26. 91     Jump to join   n   ;
28     15.1.32    Jump if not space = 14;
29     13.4.40    Jump if ARITH;
30     26.  6     Jump to READ;
31     13.1.54    Jump if B1 ≠ 0;
32     13.2.42    Jump to stop if SUBSEQUENT SHIFT;
33      6.1. 8    Add for ) = 6;
34     13.1.29    Jump if not );
35     12.4.37    Jump if not REFERENCE;
36      0.0. 0    STOP;
37      2.1.97    B1  : = 97;
38     12.1.46    Jump if integer is  0;
39      0.0. 0    STOP;
40     13.1.42    Jump to STOP unless CR or Sp;
41     17. 99     S99  : = next order;
42         0      Obey directive (STOP if jump entry);
43     42. 11     A : = -ve;
44     37. 17     A : = -1;
45     46. 99     S99 : = S99 - (-1);
46     42. 99     A : = directive;
47     48. 95     Collate for directive address;
```

```
48      40.1.100    S  100 + B1     : = Address;
49      26.    1    Jump to Resume;
50       5.1.1024   INTERLUDE
51      40.1.2047   TO CLEAR          Needs A = 0
52      14.1.51     MAIN              to clear
53      26.    0    STORE              S1023;
54      42.   97    A  : = 97;
55      12.4.58     Jump if NOT REFERENCE;
56      17.   97    Next order becomes  42.100 = S97  ;
57      42.  100    A : = S  100 + S97  ;
58      13.3.61     Jump if PLUS (B3 = 15);          ;
59      40.   97    integer store  : = A, clearing A;
60       ?.   97    A  : = 0 = S97;
61       7.1.26     B1 : = B1 - 26 = 0 if PLUS;
62      13.1.65     Jump if not PLUS;
63       5.3.15     B3 : = 15 for PLUS and 1064;
64      26.3.  9    Jump to join with MINUS at 1064;
65      14.1.81     Jump if 1 ≠ 27;
66      13.4.42     Jump to stop if ARITH;
67      13.2.42     Jump to stop if SUBSEQUENT SHIFT;
68      20.   96    TITLE OUTPUT;
69       6.1.18     Add for M / LF = 13;
70      13.1.72     Jump if not M / LF;
71      12.4.  6    Jump to READ if phi (LF);
72      10.1.96     TITLE READ TO B1 and S96;
73      13.1.75     Jump if ≠ 0;
74       5.4.  0    Set phi (FIGURE SHIFT);
75       7.1.27     Subtract for 1 = 27;
76      13.1.78     Jump if not 1;
77       5.4.1024   Set L (LETTER SHIFT);
78       7.1.  4    Subtract for ER = 31;
79      12.1.72     If ER jump to TITLE-READ;
80      26.   68    Jump to TITLE-OUTPUT;
81      14.1.88     Jump if not ( , ) = 28;
82      12.4.84     Jump if not a REFERENCE;
83       0.0.  0    STOP;
84      38.2.14     Logic shift (14 INITIALLY THEN 11);
85       5.2.2045   B2  : = -3 for SUBSEQUENT SHIFT;
86      43.   98    A  : = A + previous part item;
87      26.    2    Jump to SUBSEQUENT re-start;
88      43.   98    A  : = A + previous part item;
89      14.1.93     Jump if not   n   = 29;
90       5.3.  0    B3  : = 0 for MINUS;
91       5.4.1024   Set for REFERENCE and ARITH:
92      26.    4    Jump to store an item:
93      14.1.17     Jump if not CR (= 30):
94      26.   29    Jump to treat CR like Sp:
95            2047  Bit pattern for address collation:
```

# PAPER TAPE CODE → SYMBOL AND SYMBOL → PAPER TAPE CODE INDEX

| CODE | L.S. | F.S. | L.S. | CODE | F.S. | CODE |
|---|---|---|---|---|---|---|
| 00.000 | F.S. | BLANK | A | 10.000 | 1 | 10.000 |
| 10.000 | A | 1 | B | 01.000 | 2 | 01.000 |
| 01.000 | B | 2 | C | 11.000 | 3 | 11.001 |
| 11.000 | C | 3 | D | 00.100 | 4 | 00.100 |
| 00.100 | D | 4 | E | 10.100 | 5 | 10.101 |
| 10.100 | E | ( | F | 01.100 | 6 | 01.101 |
| 01.100 | F | ) | G | 11.100 | 7 | 11.100 |
| 11.100 | G | 7 | H | 00.010 | 8 | 00.010 |
| 00.010 | H | 8 | I | 10.010 | 9 | 10.011 |
| 10.010 | I | ≠ | J | 01.010 | 0 | 00.001 |
| 01.010 | J | = | K | 11.010 | + | 01.011 |
| 11.010 | K | - | L | 00.110 | - | 11.010 |
| 00.110 | L | 0 | M | 10.110 | . | 00.111 |
| 10.110 | M | L.F. | N | 01.110 | / | 11.101 |
| 01.110 | N | SPACE | O | 11.110 | X | 00.011 |
| 11.110 | O | , | P | 00.001 | = | 01.010 |
| 00.001 | P | 0 | Q | 10.001 | ≠ | 10.010 |
| 10.001 | Q | > | R | 01.001 | ≥ | 01.001 |
| 01.001 | R | ≥ | S | 11.001 | > | 10.001 |
| 11.001 | S | 5 | T | 00.101 | A | 10.111 |
| 00.101 | T | → | U | 10.101 | ( | 10.100 |
| 10.101 | U | 5 | V | 01.101 | ) | 01.100 |
| 01.101 | V | 6 | W | 11.101 | , | 11.110 |
| 11.101 | W | / | X | 00.011 | B | 11.000 |
| 00.011 | X | X | Y | 10.011 | → | 00.101 |
| 10.011 | Y | 9 | Z | 01.011 | D | 00.110 |
| 01.011 | Z | + | L.S. | 11.011 | SPACE | 01.110 |
| 11.011 | L.S. | | . | 00.111 | E | 11.111 |
| 00.111 | . | . | ? | 10.111 | L.F. | 10.110 |
| 10.111 | ? | A | C | 01.111 | C.R. | 01.111 |
| 01.111 | C | C.R. | F.S. | 00.000 | BLANK | 00.000 |
| 11.111 | | | | | | |

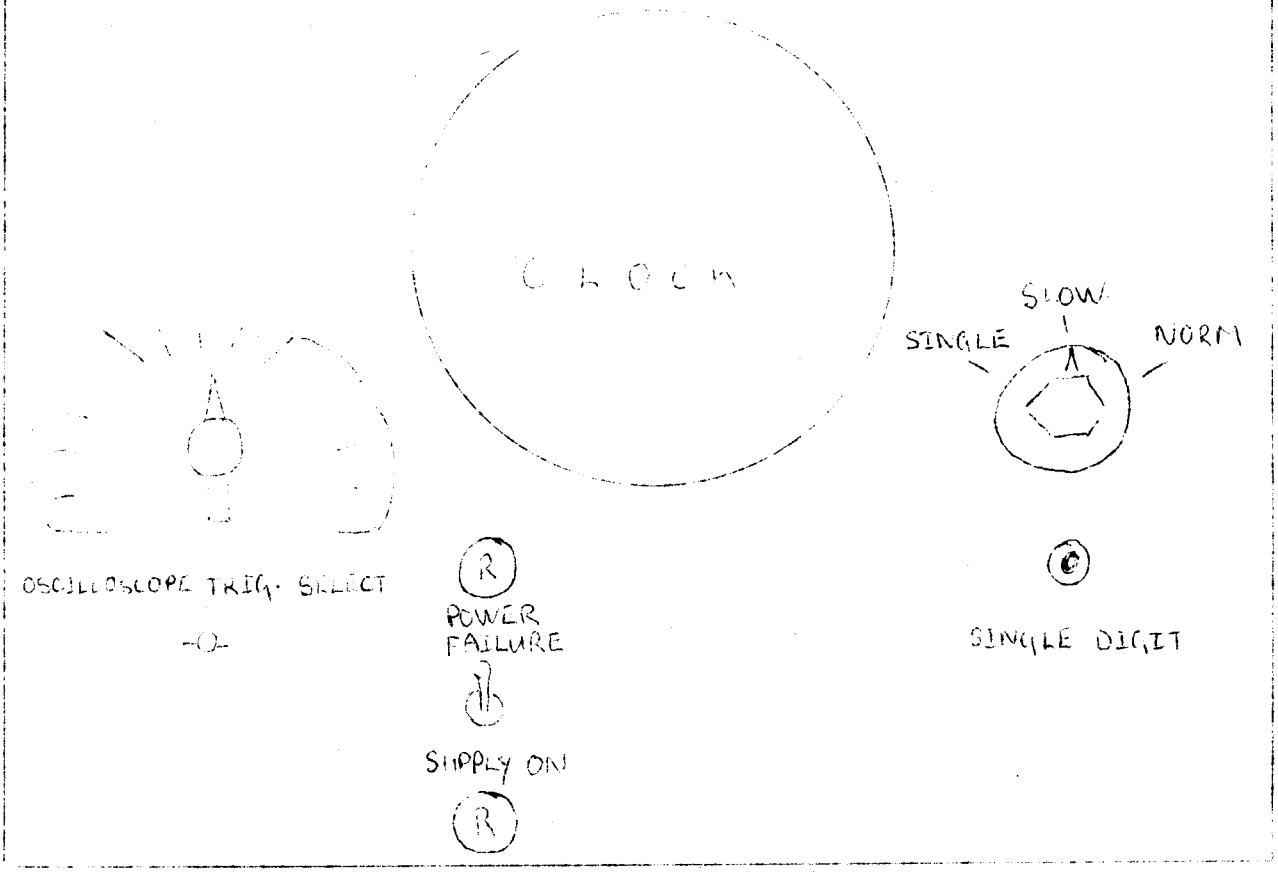
# INITIAL ORDERS PROGRAMME FLOW DIAGRAM



Instruction Execution Times

| Instruction | Unmodified | Modified |
|---|---|---|
| 0.0.0 | $\frac{1}{4}$ msec | - |
| 0.0.N | $\frac{1}{4}$ msec | - |
| 0.B.N | $\frac{1}{2}$ msec | - |
| B-Register Operations | $\frac{1}{2}$ msec | - |
| Input | 4 msec | - |
| B-Register Jumps | $\frac{1}{4}$ msec | - |
| 16 | $\frac{1}{2}$ msec | $\frac{2}{3}$ msec |
| 17 | $\frac{2}{3}$ | 1 msec |
| Output | 40 msec | 40 msec |
| Modifible Jumps | $\frac{1}{4}$ | $\frac{1}{2}$ msec |
| 28,29 | $\frac{1}{4}$ msec | $\frac{1}{2}$ msec |
| 31 | $\frac{2}{3}$ | 1 msec |
| 32,33,34 | 1 msec | $1\frac{1}{3}$ msec |
| Shifts | $\frac{1}{4}$msec + 18 μsec/shift | $\frac{1}{2}$msec + 18μsec/shift |
| M-Register Operations | $\frac{2}{3}$ msec | 1 msec |
| D-Register Operations | $\frac{2}{3}$ msec | 1 msec |
| Multiply | From 1 to 5 msec | From $1\frac{1}{3}$ to $5\frac{1}{3}$msec |
| Divide | $\doteq$ 10 msec | $\doteq$ 10 msec |
| 59,60,61 | $\frac{2}{3}$ msec | 1 msec |

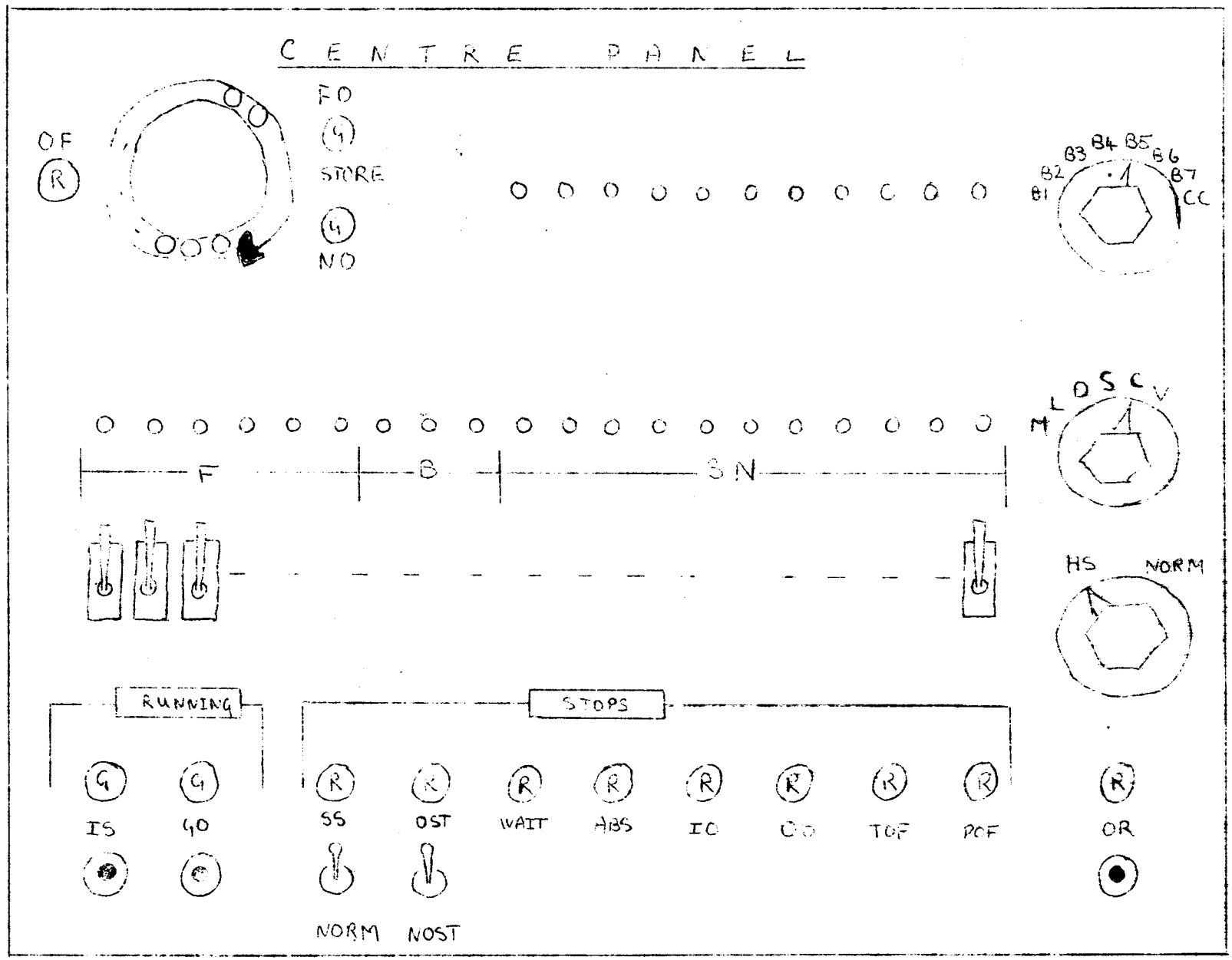
## LEFT PANEL




## RIGHT PANEL

| INPUT | 1 | 3 | 5 |
|---|---|---|---|
| TAPE READY | G | G | G |
| TAPE JAMMED | R | R | R |

| OUTPUT | 0 | 2 | 4 |
|---|---|---|---|
| TAPE READY | G | G | G |
| TAPE JAMMED | R | R | R |
| NO TAPE | R | R | R |
| NO TAPE OVERRIDE | O | O | O |


CENTRE PANEL
OF
R
FO
G
STORE
H
NO
B1
B2
B3
B4
B5
B6
B7
CC
M
L
D
S
C
V
F
B
SN
HS
NORM
RUNNING
STOPS
G
G
IS
GO
R
R
R
R
R
R
R
R
SS
OST
WAIT
ABS
IC
CO
TOF
POF
OR
NORM
NOST