

# 1301 Programmers Manual

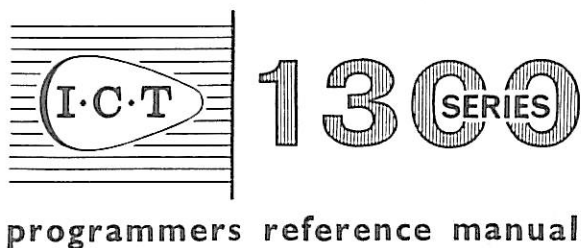
Section = Part Four

Original PDF by Roger Holmes  
Jan 2009

Sub Sections and HTML version  
by Rod Brown

Supplied by [ict1301.co.uk](http://ict1301.co.uk)  
from the pages of the

ICT 1301  
Resurrection Project



## PROGRAMMING TECHNIQUES

Contents	Page
4.1 FLOWCHARTING ... ..	1
4.1.1 Recommended Procedures ... ..	3
4.1.2 Recommended Symbols ... ..	3
4.2 WRITING THE PROGRAM ... ..	8
4.3 MODIFICATION ... ..	9
4.3.1 Useful Instructions for Modification ... ..	12
Functions 66 and 67 ... ..	12
Function 41 ... ..	13
4.3.2 Demodification ... ..	15
4.3.3 Modification and Relative Addresses ... ..	18
4.4. SUBROUTINES ... ..	18
4.4.1 Using the Relative Addressing System ... ..	22
4.4.2 Inclusion in the Main Program Block ... ..	22
4.4.3 Use of Indicators ... ..	23
4.4.4 Use of Several Entry Points ... ..	23
4.4.5 Standard Procedure ... ..	23
4.5 STORAGE ALLOCATION ... ..	24

Contents continued	Page
4.6 INITIAL ORDERS ... ..	25
4.6.1 Control Designation 'R' ... ..	25
4.6.2 Control Designation 'B' ... ..	27
4.6.3 Control Designation 'E' ... ..	29
4.6.4 Control Designation 'C' ... ..	30
4.6.5 Control Designation 'F' ... ..	31
4.6.6 Control Designations 'P' and 'M' ... ..	32
4.6.7 Initial Orders Sequence Check ... ..	34
4.7 RELATIVIZERS ... ..	35
4.7.1 Relative Addressing and Use of General Purpose Subroutines ... ..	35
4.8 RESTART PROCEDURES ... ..	37
4.8.1 Restarts following I.A.S. and Drum Parity Error Stops ... ..	38
4.9 PROGRAM TESTING ... ..	38
4.10 TIMING A PROGRAM ... ..	40
4.11 DOCUMENTATION ... ..	42
4.12 AN EXAMPLE OF A CODED PROGRAM. ... ..	43
4.12.1 Introduction ... ..	43
4.12.2 Factors ... ..	43
4.12.3 Results ... ..	44
4.12.4 Calculations... ..	44
4.12.5 P.A.Y.E. Calculation ... ..	44
4.12.6 Timing ... ..	46

## Illustrations

Figure 44 Systems Flowchart Symbols ... ..	2
Figure 45 Flowchart of P.A.Y.E. Program ... ..	47
Figure 46 Main Program ... ..	50
Figure 47 Constants ... ..	54
Figure 48 Temporary Storage ... ..	55
Figure 49 Input Information ... ..	56
Figure 50 Result of Calculations ... ..	57

### FLOWCHARTING

4.1

Programming consists of writing a series of instructions, each of which contributes minutely to the overall process, and it is essential that the programmer should have a clear representation of the logical processes involved in a job before commencing to write coded instructions. For this reason, the logic is expressed in diagrammatical form, in such a way that the flow of the logic can be easily followed. Such diagrams are called flowcharts.

I.C.T. has adopted certain conventions for drawing flowcharts and it is recommended that these should be followed.

When commencing a large job, particularly a commercial application, it is probable that the first flowchart will be a Systems Flowchart. This chart illustrates in broad outline the data to be supplied to the computer, the calculations to be performed and the form of the results, indicating which peripheral units are to be used. The systems flowchart is usually drawn up from a written specification of the job, the diagrammatic representation enabling any basic logical errors to be detected at an early stage. The systems flowchart should be expressed so that it can be readily understood both by programmers and by those who know the system under consideration but do not necessarily understand programming. The systems flowchart is very often the initial document received by the programmer and should give him all the information he requires to prepare the program.

Before coding is commenced, a programmers' flowchart should be prepared, either from the systems flowchart or from a written specification of the routine. This flowchart should cover in detail all calculations, input/output techniques and error procedures. In drawing up the programmers' flowchart, the program instructions available should constantly be borne in mind. The flowchart should be sufficiently detailed for the programmer to convert it directly into coded instructions. When the chart has been drawn, it should be carefully checked and tested for logical errors by mentally following the flow for sample data. Careful checking at this stage can save much wasted programming and machine time. As far as possible, programmers' flowcharts should be written in terms readily understood by other programmers not directly concerned with that particular job.



# COMPUTER EQUIPMENT

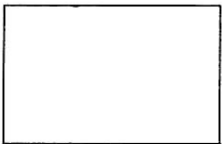
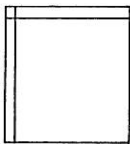

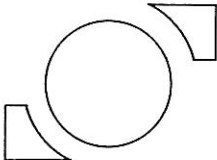

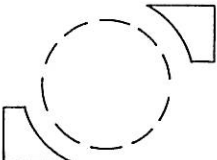
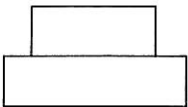
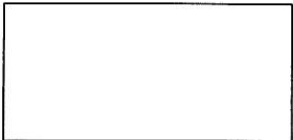
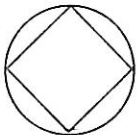
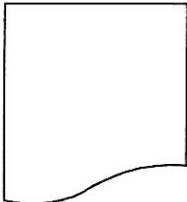
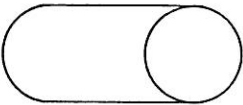
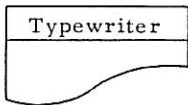
Description	I.C.T. Symbol	Description	I.C.T. Symbol
Source Document		Core Store	
Punched Card		Magnetic Tape	
Paper Tape		Work Tape	
Typewriter Input		General Operational Symbol	
Computer		Printed Output	
Magnetic Drum		Typewriter Output	

Figure 44: SYSTEMS FLOWCHART SYMBOLS

# NON-COMPUTER EQUIPMENT


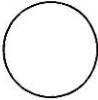
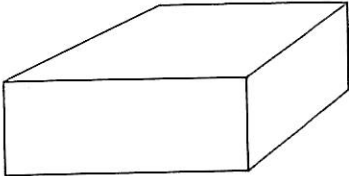
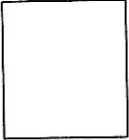
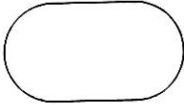
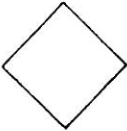
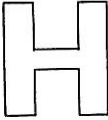
Description	I.C.T. Symbol	Description	I.C.T. Symbol
Punch and Verify		Sort	
File of Punched Cards		Tabulator	
Interpret, Match, Collate, Interpolate or Reproduce		Calculator	
		Manual Operations	

Figure 44 continued

## Recommended Procedures

4.1.1

Flowcharts should be drawn on small sheets of paper, being split into convenient sections to make this possible. Programmers' flowcharts should normally be drawn on foolscap sheets and systems flowcharts should not exceed double foolscap size. The use of small sheets makes paper-handling easier, makes charts less confusing to follow, and enables them to be more easily reproduced either by typing or by photo-copying. If black ink on tracing paper is used then dyeline copies are possible, which is more economical than photo-copying.

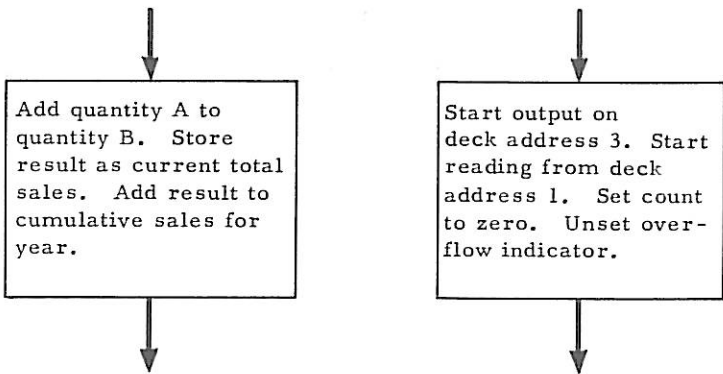
The flow of charting should be vertical, running from top to bottom of the sheet, the direction of flow being indicated by arrows, particularly when branches occur for alternative conditions. Recommended symbols should be used wherever possible. Templates are available which contain the standard symbols, and backing charts (Form No. 3203(3.60)) which have vertical and horizontal guiding lines heavily imprinted on them, so that they show through the flowchart paper, may also be obtained from I.C.T.

## Recommended Symbols

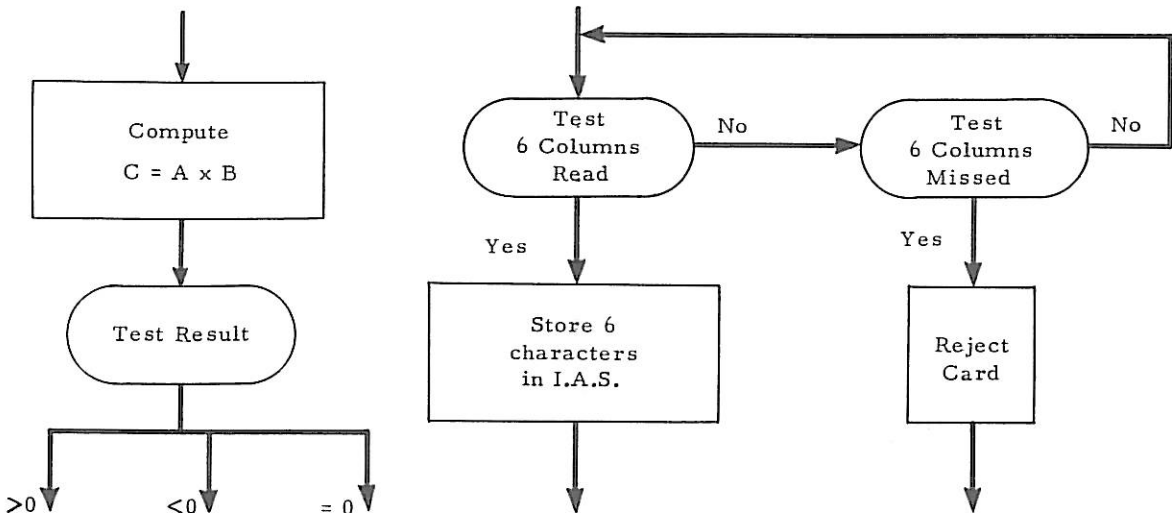
4.1.2

The I.C.T. recommended symbols for systems flowcharts are shown in Figure 44. The following symbols are recommended for programmers' flowcharts:

**Boxes** Statements of procedure should be enclosed in boxes. Where consecutive statements are uninterrupted by entries from other parts of the chart, several statements should be enclosed in one box. The statements should be clear and expressed as short sentences.



**Branches** Where different procedures are followed corresponding to different conditions holding, the necessary test to distinguish the various conditions is enclosed by an oval symbol. The branches stemming from the oval should each be marked to indicate which condition they denote.

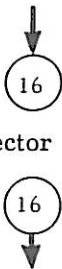


**Connectors** A connector symbol is provided so that the flow can proceed from one sheet to another. Connector symbols may also be used to proceed from one point to another on one sheet, particularly when a line joining the points would confuse rather than clarify the chart.

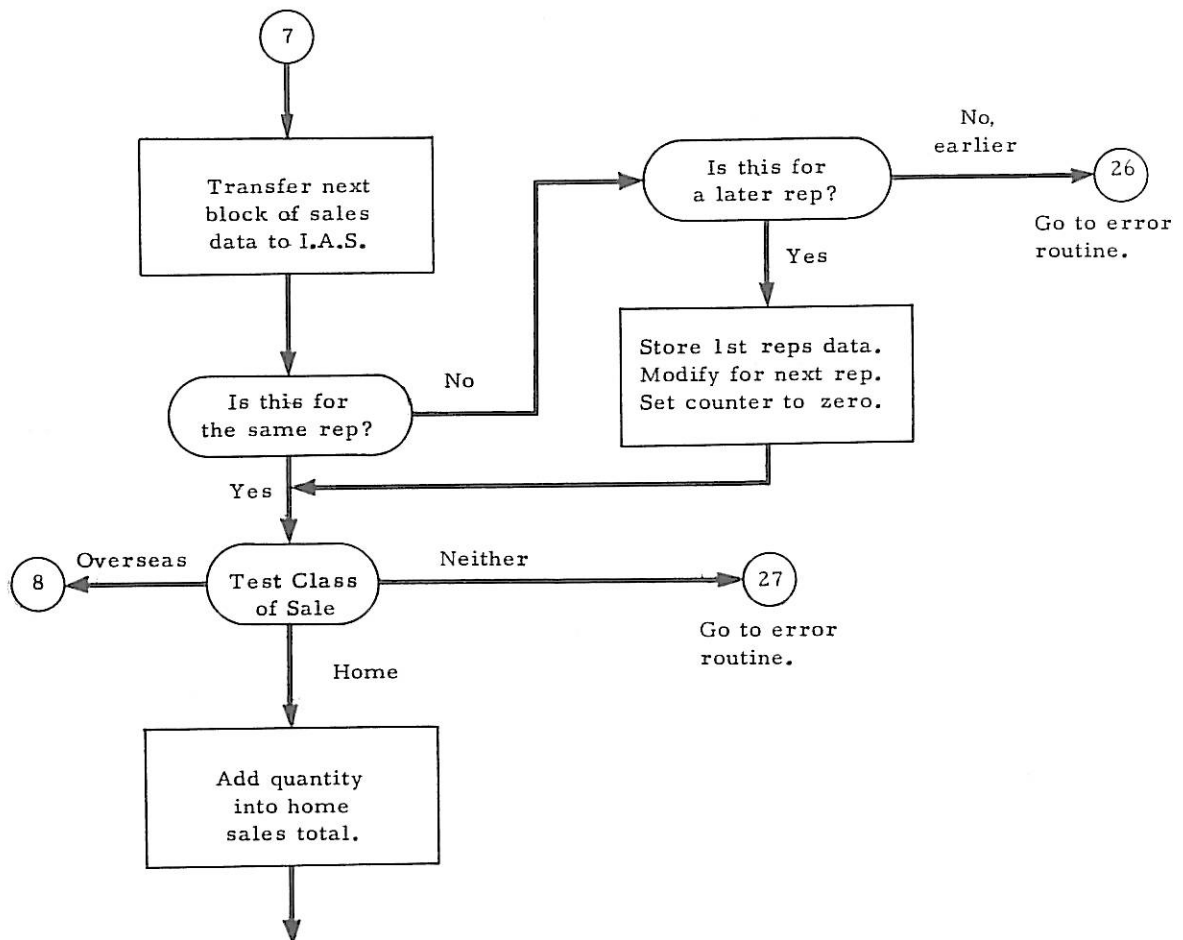
The connector symbol consists of a small circle enclosing an identifying number.

Thus,

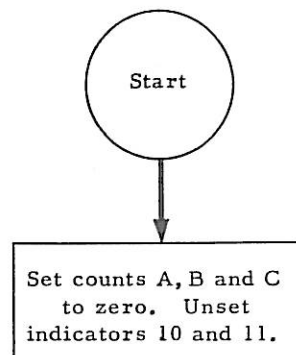
indicates that the procedure continues at connector 16 with an arrow leading from it, i.e.



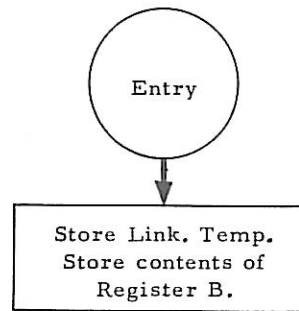
When subsequently converting a flowchart to program instructions, the presence of a connector with an arrow leading from it indicates a junction of several pieces of program. This means that a programmed jump is going to occur to the next instruction and it should therefore be in the first half of a word.



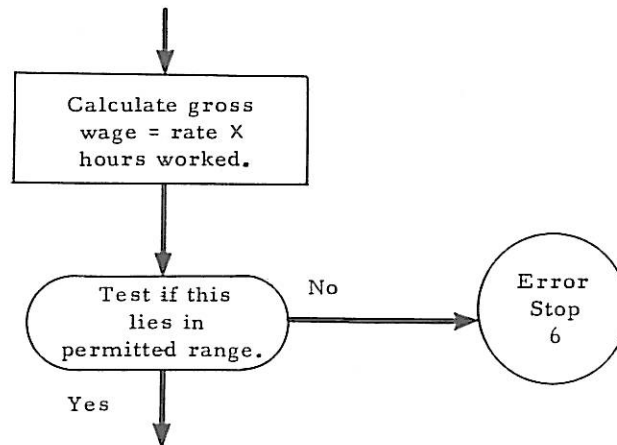
**Starts, Stops and Subroutines** It is usual to indicate the start of a routine by enclosing the word Start in a circle rather larger than the connector symbol.



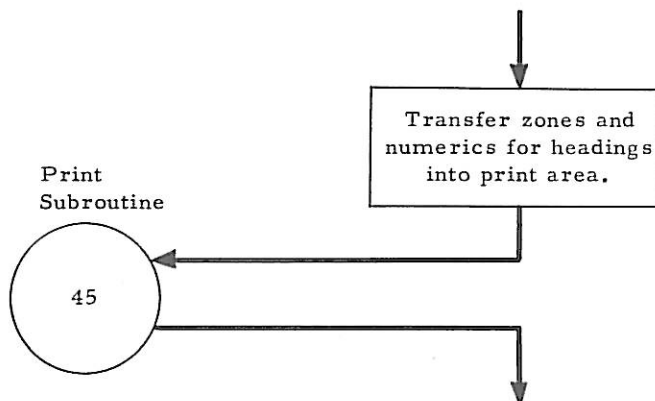
When flowcharting a subroutine the word Entry is used instead of Start.



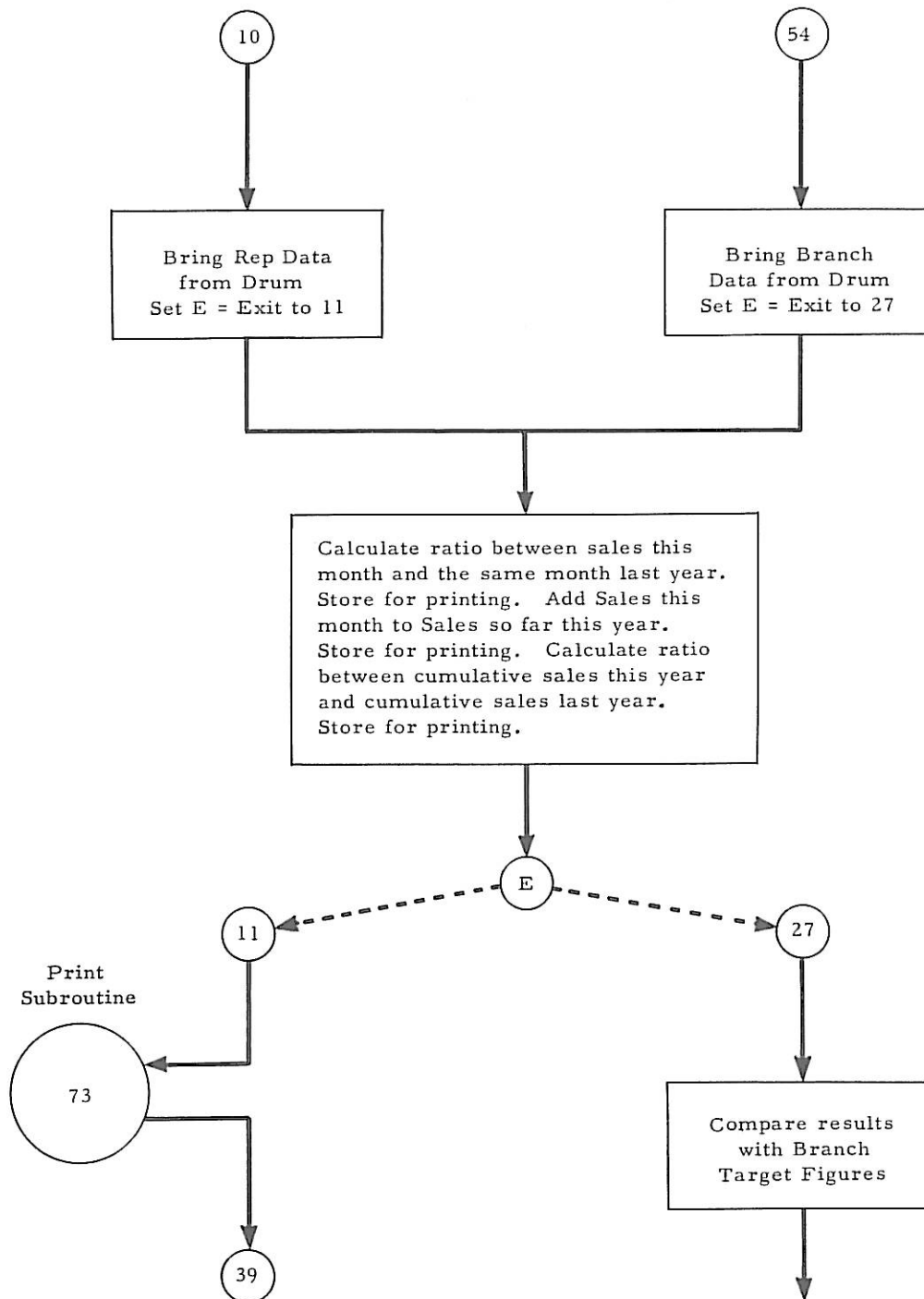
When the procedure involves stopping the computer, this is indicated by the word Stop enclosed in a circle. A number may also be included to identify the stop.



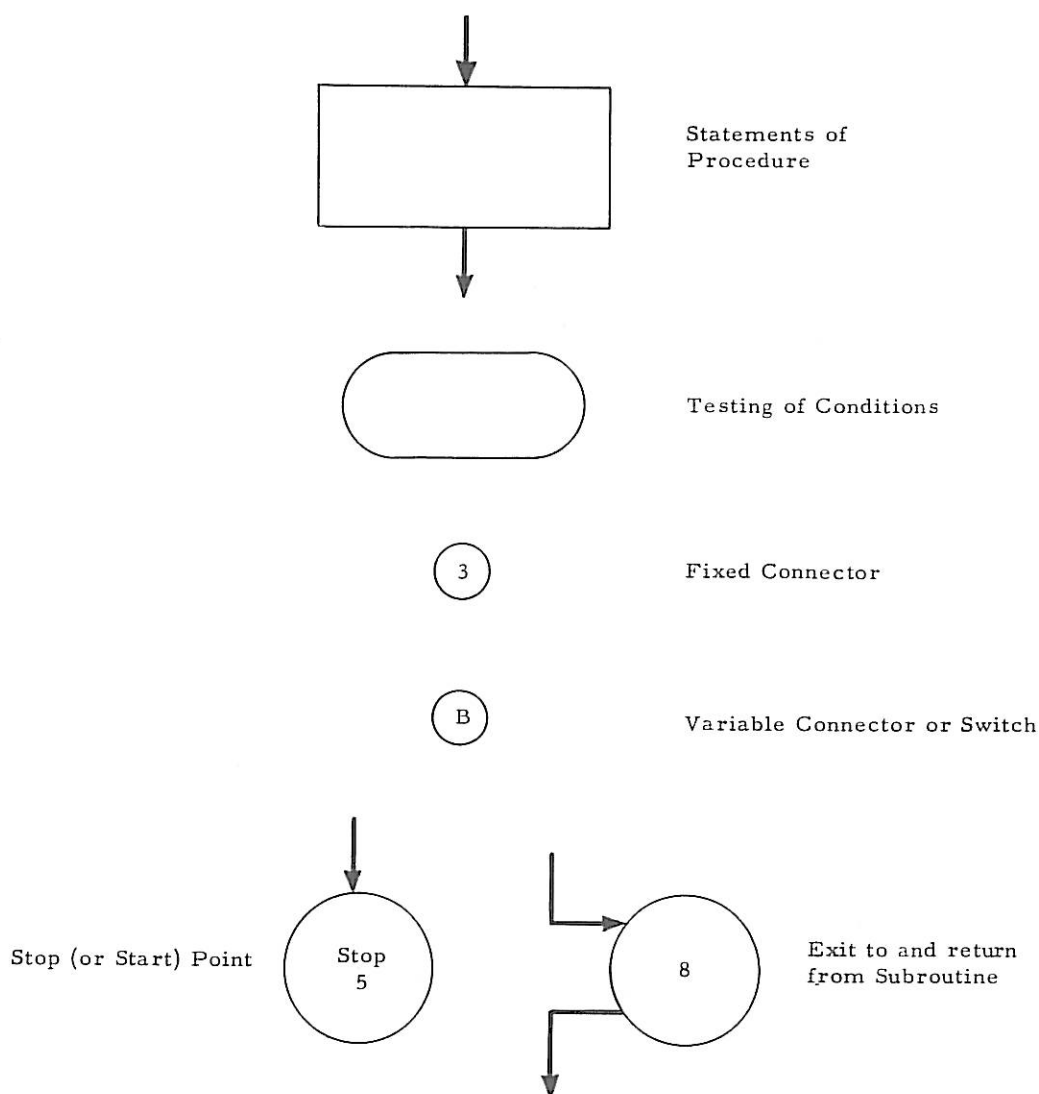
When a subroutine is used as part of the procedure, the subroutine logic is not flowcharted as part of the main routine. The entry to and exit from the subroutine are indicated on the main flowchart by lines leading to and from a circle. The circle contains either the name of the subroutine or an identifying number. In the latter case the name of the subroutine should also be written beside the circle. If the subroutine has been specially written for the job concerned, then it is flowcharted separately. If the subroutine is a library routine, its flowchart need not be included.



**Switches** Where different cases share a common piece of procedure, this may be conveniently represented on a flowchart using switches or variable connectors. The variable connector differs from the fixed connector in that it contains a letter instead of a number. The letter in the variable connector is set to different values, the value indicating at which connector the procedure is to continue.



### Summary of programmers' flowcharting symbols



## WRITING THE PROGRAM

### 4.2

When the flowcharts have been drawn and checked, the actual programming can begin. If the program is large, it should be divided into sections and written in blocks. Each block should be given a relativizer reference number (R.R.N.) which should also be used as the block number. It is recommended that blocks of program should not exceed about 50 words as this makes alteration and rewriting easier if they should become necessary during testing. Relativizer reference numbers should also be allocated to data blocks and relative addressing should be used throughout. Where there are several blocks in a program, constants should be held under a separate R.R.N., so that their addresses remain unchanged if the other blocks are rewritten. An area of temporary storage is normally required for counts and intermediate results. This too should be given its own R.R.N. There are recommended uses for certain relativizer numbers, which are described later.

Throughout program writing, certain information should be recorded as running documentation, even though this may later have to be altered. Any program indicators having common significance to several blocks should be allocated initially. Indicators should normally be used starting at 10 and working upwards, to avoid possible clashes with those used by library routines which start at 19 and work downwards. Card and sheet layouts should be recorded on the charts available for this purpose. Any stops incorporated into the program should be given an identifying number, and a note should be kept of the stops used together with their significance. It should be remembered that in order to produce the identifying number in control register 3 when the computer stops, it must be written in the address part of the stop instruction with one subtracted from it. Possible stop numbers lie in the range 1 to 4,000. Numbers 1 to 2,000 are reserved as standard stops for use in general purpose routines and numbers in the range 2,001 to 4,000 should therefore be used by the programmer. Any allocations of I.A.S. and drum space should be recorded as they are allocated, and updated when necessary. An up-to-date record of any allocations which have already been made assists greatly in the final storage allocations.

The program is written on program sheets and should be accompanied by narrative. The narrative should not necessarily cite exactly what each instruction does, but should give a clear overall picture of what the program is accomplishing. The narrative column should be filled in as the program is written, as this helps when checking the program and prevents the tedious job of working through sheets of program filling in the narrative. It is helpful if there is a system of cross-referencing between the program sheets and the flowcharts. The block numbers should be written on the flowchart by the appropriate sections. As an added help, a number, which is also written at the appropriate place on the program sheet, can be given to each step on the flowchart. Any such numbers on the program sheet should be written where they cannot be confused with information which is to be punched in the program cards. Any information under the columns 'D', 'F', 'A', 'R' which is not to be punched should be enclosed in brackets. It is also helpful if a note is made on the program sheet alongside any words which are referred to by instructions in other blocks. This ensures that if the block has to be altered and its addresses changed, the necessary alterations are made to the instructions referring to them.

## MODIFICATION

### 4.3

The address part of an arithmetic instruction may be that of any word in I.A.S. and may therefore refer to either data or program. For purposes of calculation, the addresses would, of course, refer to data numbers. Arithmetic operations on program words can, however, be used to good effect. The alteration of instructions by arithmetic processes is called modification, and is a very useful technique for reducing the number of words in a program.

Suppose that a particular calculation involves operations on five data numbers in I.A.S., and that the calculation has to be repeated for ten such sets of numbers. The 50 numbers are held consecutively in 50 words of I.A.S.



There are three possible alternatives:

- (a) The sequence of instructions may be written out for each of the ten sets in turn, and obeyed in this sequence.
- (b) The sequence of instructions may be written out for the first set. Instructions may then be included to modify the address parts of the instructions so that they refer to the second set. The calculation and modification instructions are repeated until the tenth set of data has been used.
- (c) Instructions may be written to transfer the first set of numbers to a working area. The instructions to perform the calculation refer to the numbers in this working area. After the calculation, instructions are written to modify the transfer instructions so that it transfers the second set of numbers to the working area.

The transfer, calculation and modification instructions are repeated until the calculation has been performed on all ten sets of data.

Methods (b) and (c) show two methods using modification, and it is evident that the written program using either of these methods is considerably shorter than that using method (a). The relative merits of methods (b) and (c) depend on the particular calculation involved. If the calculation is small and there are few instructions to be modified, then method (b) would be used. If, however, the calculation is extensive, the program is shorter and fewer mistakes are made if method (c) is used. To economize on storage space in method (c) it may be possible to use the space occupied by the first set of data as the working area, obeying the calculations on the first set and then successively transferring the other sets into that area.

*Example 1 - The use of method (b)*

I	D	F	A	R	NARRATIVE
9	8	10			Set indicator 10
		37	0000	4	Perform
10		57	0001		Calculation
		62	0003	4	
11		64	0000	6	Accumulate result
		67	0000	5	Subtract 1 from counter
12	4	01	0015	B	Test if calculation has been performed 10 times
		37	0001	5	Pick up modifying constant
13		64	0009	B	Modify instructions
		64	0010	B	
14	4	00	0009	B	Repeat calculation for next set of data

The data is held under R.R.N. 4. The result is accumulated in word 0 block 6, which is initially zero. Word 0 block 5 contains 000 000 000 010. This is used as a counter, allowing the calculation to be performed on ten sets of data.

Word 1 block 5 contains 000 000 000 005. This is a constant used to modify the addresses by five so that they refer to the next set of data.

*Example 2 - The use of method (c)*

I	D	F	A	R	NARRATIVE
12	45	0000	4		Transfer set of data
	5	0000	10		to working area
13	37	0000	10		
	62	0001	10		
14	54	0001			Perform calculation
	62	0001	10		and accumulate
15	64	0000	6		result
	37	0002	10		
16	69	0003	10		
	69	0004	10		
17	64	0000	6		
	67	0000	5		Subtract 1 from counter
18	4 01	0020	B		Test if calculation has been
	37	0001	5		performed 10 times
					Modify transfer
19	64	0012	B		instruction
	4 00	0012	B		Repeat calculation for
					next set of data

The data is held under R.R.N. 4. R.R.N. 10 is used as a working area. The result is accumulated in word 0 block 6, which is initially zero.

Word 0 block 5 contains 000 000 000 010. This is used as a counter, allowing the calculation to be performed on ten sets of data.

Word 1 block 5 contains 000 005 000 000. This is a constant used to modify the transfer instruction, so that it transfers the next set of data to the working area.

It is sometimes convenient to modify the function part of an instruction.

For example if the instruction pair

D	F	A	R
62	0100	9	
42	0000	8	

has the constant 010 000 000 100 added to it, it becomes:

D	F	A	R
63	0100	9	
42	0100	8	

**Example** The function digits may be modified in a tape program when the deck address is given as a parameter. The following program would ensure that the correct Deck Address Ready indicator is tested. The deck address is in the least-significant digit position of word 0 block 9.

I	D	F	A	R	NARRATIVE
8		37	0000	9	Enter deck address
		54	0010		Shift to required position in word
9		64	0030	B	Modify indicator test instruction
30	4	80	0032	B	Test Deck Address Ready

It is also possible to subtract the modifier instead of adding it, and to use negative modifiers. As there are two instructions to a word, care must be taken to ensure that the modifying constant is correctly positioned for modifying the required instructions. When arithmetic is performed, a word of program is treated as a twelve-digit number. This means that carries may occur between the two instructions in a word. Because of this, particularly when using negative modifiers, the modification should be carefully checked to ensure that the required result will be achieved.

#### Examples

To modify instruction pair

D	F	A	R
	37	0024	8
	42	0009	10

to

D	F	A	R
	37	0044	8
	42	0029	10

Add a constant of 000 020 000 020, or subtract a constant of 999 979 999 980.

To modify instruction pair

D	F	A	R
	37	0024	8
	42	0029	10

to

D	F	A	R
	37	0044	8
	42	0009	10

Add a constant of 000 019 999 980 or subtract a constant of 999 980 000 020.

## Useful Instructions for Modification

4.3.1

### Functions 66 and 67

These instructions are often used for modification, since they provide a convenient means of modifying by one the address of an instruction in the least-significant half of a word.

### Example

I	D	F	A	R	NARRATIVE
5	4	36	0007	B	Test 6 Columns Read
	4	37	0012	B	Test 6 Columns Missed
6	4	00	0005	B	
7		67	0007	B	
		43	0013	3	Store Register C in I.A.S.
8					

Notice that in this case the 67 instruction is modifying the instruction contained in the second half of its own word. The two instructions are transferred to CR1 and CR2 as soon as control is transferred to word 7. This means that the second instruction is obeyed the first time in its unmodified form, since it is already in CR2 when the 67 instruction is obeyed. The contents of Register C are therefore stored in word 13 block 3 the first time, followed by words 12, 11, 10..... when the instructions are subsequently obeyed.

### Function 41

The use of the 41 instruction to store the contents of Register A following a control change gives an extremely useful form of modification.

As was explained in Part 2 under Control Registers, when two single-length instructions have been obeyed and control is transferred to the following word, Register A contains the last two instructions to have been obeyed each with one added to them.

Consider the program

I	D	F	A	R
10	--	37	0024	13
		62	0019	23
11	--	41	0010	B

The 41 instruction effectively causes both instructions in word 10 to have 1 added to their addresses.

If it is required to modify by one the address in the first instruction only, this can be achieved as follows:

I	D	F	A	R
10	--	37	24	13
		62	19	23
11	--	41	10	B
		67	10	B

Modification can also be effected by using a 41 instruction after certain double-length instructions.

The following table shows the contents of Register A following the control change after these instructions have been obeyed.

Function	Contents of Register A
45	The 45 instruction with the addresses in both halves increased by the number of words transferred. If the number of words is specified as zero, the addresses are each increased by 20.
80 or 84	The 80 or 84 instruction with the I.A.S. address increased by the number of words transferred and the drum address increased by the number of decades transferred.
81 or 85	The 81 or 85 instruction with the I.A.S. address increased by the number of words transferred less one, and the drum address increased by the number of decades transferred.

#### Example

I	D	F	A	R
8	--	45	0000	8
		15	0000	9
9	--	41	0008	B
	--			

The 41 instruction causes the addresses in both halves of the 45 instruction to have 15 added to them.

## Demodification

### 4.3.2

If a section of program which has been modified is to be used again later in the program, then it is necessary that it should be reset so that its effect is the same as the first time it was obeyed, before any modification took place. For example, a card-read program is modified so that successive sets of six columns are stored in different words of I.A.S. When the next card is to be read, the program must be reset so that the first (and subsequent) six columns are stored in the correct words in I.A.S.

The process of resetting the program is termed demodification and can be achieved as follows:

- (a) Instructions can be included before modification takes place which set the instructions to their initial values, thus eliminating any previous modifications.
- (b) Instructions can be included after modification which reset the instructions in readiness for the next time that they are obeyed.
- (c) The form of the modification can be such that the newly formed instructions overwrite the previous instructions, thus eliminating the need for special resetting instructions.
- (d) The section of program can be transferred from the drum, where it is stored in an unmodified form, before being obeyed.

*Method (a)* Constants are held which consist of the instructions held in their unmodified form.

*Example*

I	D	F	A	R
4		45	0014	B
		2	0007	B
5	4	36	0007	B
	4	37	0012	B
6	4	00	0005	B
7		67	0007	B
		43	0013	3
8				
			0014	
9		67	0008	B
	4	02	0005	B
14		67	0007	B
		43	0013	3
15				
			0014	

The instruction to store the contents of Register C, and the counter indicating the number of times Register C is to be stored, are both set to their unmodified forms using a 45 instruction.

**Method (b)** This may consist of a technique similar to that used in method (a). Alternatively, demodification may be achieved by subtracting (or adding if modification consisted of subtraction) the modifier or its multiple if the instruction has been modified several times.

**Example 1**

I	D	F	A	R
17	---	37	0000	3
		64	0018	B
18	---	65	0018	B
		37	0000	25

The 37 instruction is obeyed in its modified form since it is already in the control registers when demodification takes place.

**Example 2**

I	D	F	A	R	NARRATIVE
8	---	57	0012		Zeroise Register B
9	---	45	0000	19	Set counter of 10
		1	0000	1	
10	---	66	0010	B	Accumulate result in Register B
		62	0000	5	
11	---	67	0000	1	Return to add in next quantity
		4	02	B	
12	---	42	0010	5	Accumulation complete. Store result
		37	0000	19	
13	---	65	0010	B	Demodify

where word 0 block 19 contains a constant of 000 000 000 010. Words 0, 1, 2 ... 9 of block 5 are summed in Register B and the result stored in word 10 block 5.

**Method (c)** Constants are held of the instructions in their unmodified form. The modifiers are added to these constants in Register B, and the modified instruction is stored in the appropriate place in the program.

### Example

I	D	F	A	R
9	---	37	0000	5
		54	0006	
10	---	62	0031	B
		42	0011	B
11	---	37		
		62	0000	8

31	---	37		
		62	0000	8

where the required address for the 37 instruction is held in the least-significant half of word 0 block 5.

There is an extension to this technique which is known as chain modification. Chain modification can sometimes be used where several instructions need to be modified by the same number.

### Example

Suppose that  $x$  is the modifier and that it is required to perform the following program:

I	D	F	A	R
	8	10		
		37	( $x$ )	
		54	0001	
		62	( $10+x$ )	

The required modification could be achieved as follows:

I	D	F	A	R

6	---	37	0016	B
7	---	62	0000	5
		42	0009	B
8	---	62	0017	B
		42	0010	B
9	8	10	---	---
		37		
10	---	54	0001	
		62		

16	8	10	---	---
		37		
17	---	43	2001	---
		25	0010	



where the modifier x is stored in word 0 block 5. When the first instruction has been modified it is still contained in Register B. The addition of a specially created constant produces the required second instruction. This method reduces the number of program instructions, since there is no need for a 37 instruction to enter a constant to form the second (and subsequent) instructions.

**Note** When arithmetic is performed on instructions it is likely that the result (an instruction pair) will satisfy overflow conditions. It should be noted, therefore, that modification of instructions may cause the overflow indicator to be set.

## Modification and Relative Addresses

4.3.3

Modification takes place when the program is obeyed and it is therefore the absolute address which is modified.

When it is required to modify the address of an instruction to another under the same R.R.N., however, the absolute address need not be considered.

Consider

I	D	F	A	R
5	--	66	0019	23
	4	00	0006	B

where word 19 block 23 is

D	F	A	R
--	--	--	--
		0000	5

If R.R.N. 5 is set, say to an I.A.S. address of 224, then when the 66 instruction is obeyed this will be modified to 225. When programming, this may be considered as modifying the address word 0 block 5 to word 1 block 5, and the modification will remain correct whatever the setting of R.R.N.5.

If it is required to modify the address of an instruction to another under a different R.R.N., the modifier cannot be assessed until the relativizer settings have been allotted.

For example to modify

D	F	A	R
--	37	0014	15

to

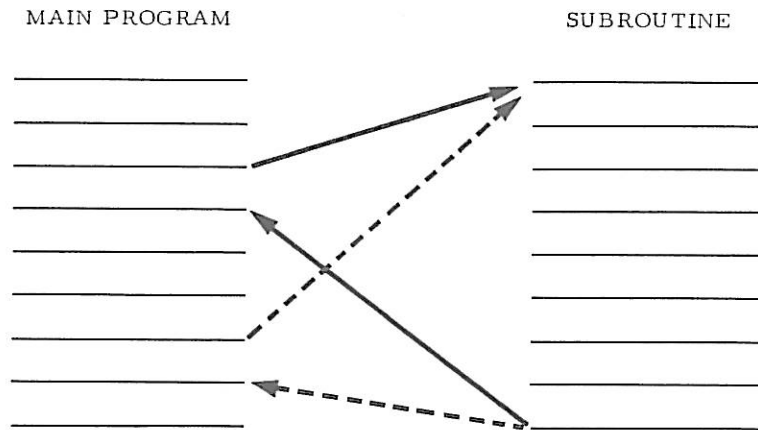
D	F	A	R
--	37	0012	16

and R.R.N. 15 is set to an I.A.S. address of 200, R.R.N. 16 is set to an I.A.S. address of 300, then it is required to modify absolute address 214 to absolute address 312 i.e. the required modifier is 98. This type of modification should be avoided, since the modifier becomes incorrect if the storage is re-allocated.

## SUBROUTINES

4.4

A subroutine is a self-contained section of program which can be incorporated into a complete program. A subroutine can be entered from any point in the main program and is so constructed that, when the subroutine has been obeyed, a return jump is automatically made to the instruction immediately following the jump which entered the subroutine.



There are two main reasons for using subroutines:

- (a) Certain routines are of a general nature and are common to many programs. These routines are made generally available via the I.C.T. Subroutine Library. Examples of such routines are sines, cosines, square roots etc. for scientific applications and P.A.Y.E. calculation for commercial applications. Details of the types of routines available are given in Part 5. The use of general purpose routines can save much programming and testing time.
- (b) Certain sections of program may be required at several different points in the main program. Storage space can be saved by making these sections into subroutines, thus storing them only once instead of storing them separately each time they are required.

The return jump from the subroutine to the main program is effected by making use of the contents of Register A following the programmed jump to the subroutine.

Suppose that the subroutine starts at word 0 block 17, R.R.N. 3 is set to an I.A.S. address of 100, R.R.N. 17 is set to an I.A.S. address of 150, and that the block in which the program is written starts at word 0 in I.A.S.

Consider the program:

I	D	F	A	R
10	--	37	0010	3
		62	0011	3
11	4	00	0000	17
		42	0015	3

The successive contents of the control registers and Register A are as follows:

CR1	Register A First Half	CR2	Register A Second Half	CR3
370110		620111		004011
620111	Word 110a	004011	Word 110b	370111
004011	Word 111a	370111	Word 111b	620112
004150	370111	420115	620112	004012
Word 150a	420115	Word 150b	004012	004151

Before the first instruction of the subroutine (word 150a) is obeyed, the contents of Register A are 420115004012. If these contents are preserved at the beginning of the subroutine, and obeyed when the subroutine has been completed, control is effectively transferred to the instruction in the main program immediately following the jump to the subroutine.

Consider now the case where the jump to the subroutine is in the second half of a word:

I	D	F	A	R
10		37	0010	3
		62	0011	3
11		63	0013	3
	4	00	0000	17

The successive contents of the control registers and Register A are as follows:

CR1	Register A First Half	CR2	Register A Second Half	CR3
370110		620111		004011
620111	Word 110a	004011	Word 110b	370111
004011	Word 111a	370111	Word 111b	620112
630113	370111	004150	620112	004012
004150	Word 113a	004012	Word 113b	630114
Word 150a	004012	Word 150b	630114	004151

Before the first instruction of the subroutine (word 150a) is obeyed, the contents of Register A are 004012630114. If these contents are preserved at the beginning of the subroutine, and obeyed when the subroutine has been completed, this gives the required return jump to word 12 of the main program.

It can be seen from the two examples just given that, whether the jump to a subroutine is in the first or second half of a word, Register A contains the necessary information for the return to the main program. The first instruction to be obeyed in any subroutine is therefore a 41 instruction. The 41 instruction stores the contents of Register A in a word specially allocated for the purpose. This word is known as the Link since it provides communication between the subroutine and the main program. When the instructions forming the subroutine have been completed, control is transferred to the link, which restores control to the correct place in the main program.

#### Example

MAIN PROGRAM

I	D	F	A	R
13	--	62	0049	17
		64	0094	17
14	4	64	0087	17
		00	0000	13
15	--	60	0191	18
		56	0004	

SUBROUTINE (BLOCK 13)

I	D	F	A	R
	B	--	----	----
0	--	41	0012	B
		37	0004	3

12	P	--	[LINK]	----
----	---	----	--------	------

The system of storing a link for a return jump makes it possible for subroutines to themselves make use of other subroutines, there being no theoretical limit to the number of subroutines in operation at one time.

#### Example

MAIN PROGRAM

I	D	F	A	R
29	---	62	0014	13
		64	0015	13
30	4	00	0005	10
31	---	---	---	---

BLOCK 10

I	D	F	A	R
5	---	41	0011	B
		37	0010	B
6	4	00	0016	9
		42	0013	1
7	---	37	0002	3

BLOCK 9

I	D	F	A	R
16	---	41	0030	B
		37	0001	3
17	---	---	---	---
18	---	---	---	---

11

I	D	F	A	R
11	P	[LINK]	---	---

30

I	D	F	A	R
30	P	[LINK]	---	---

It may be necessary for the main program to provide data on which the subroutine can operate, or for it to provide other necessary information. For example, for a general subroutine to print one line, the information to be printed must be provided, and also an indication must be given of the number of spaces required after the line has been printed. Information, other than data, which is provided by the main program is termed a parameter or key. This communication between main program and subroutine can be achieved by the following methods.

Using the Relative Addressing System

4.4.1

The subroutine is written assuming that the data or parameters are held in specified words under a specified R.R.N. The main program ensures that the information is correctly positioned before the subroutine is entered. The subroutine does not, of course, impose any storage allocation restrictions on main program since, although it makes reference to a given R.R.N., the main program is responsible for setting the value of the relativizer concerned.

Inclusion in the Main Program Block

4.4.2

Where there are only one or two data or parameter words, these can be placed in the program block after the jump to the subroutine. The parameters are preceded by a jump instruction causing the parameter words to be by-passed and not obeyed as program instructions.

**Example** Consider a subroutine which performs the necessary drum parity error procedure following a drum transfer. The drum transfer to be obeyed must be provided to the subroutine as a parameter.

MAIN PROGRAM					SUBROUTINE (BLOCK 29)				
I	D	F	A	R	I	D	F	A	R
12	4	00	0000	29		B			
	4	00	0014	B					
13		81	0000	23	0		41	0006	B
		10	0000	23					
14		37	0019	23					
		62	0016	23	6	P	[LINK]		

The link will contain the absolute form of

D	F	A	R
4	00	0014	B
4	00	0013	B

The second half contains the address of the parameter, which can be easily extracted by program.

#### Use of Indicators

4.4.3

A subroutine can be written so that if an indicator is set it operates slightly differently from its operation when the indicator is unset. The main program sets or unsets the indicator to achieve the required result. For example the programs to evaluate the sine and cosine of an angle are largely similar and are therefore combined into one subroutine. According to the state of an indicator, the sine or cosine is evaluated.

#### Use of Several Entry Points

4.4.4

As an alternative to using indicators it is possible for a subroutine to have several entry points, corresponding to alternative requirements. Thus, in the first example, it would be possible to have an entry point at word 0 of the subroutine to evaluate the sine, and an entry point at word 1 of the subroutine to evaluate the cosine.

Each entry word has a 41 instruction in the first half to store the return to the main program. The entry points can use a common link since only one entry point is used in any particular case.

#### Example

I	D	F	A	R	
	B				
0	41	0030	B		1st entry point.
	4 00	0010	B		
1	41	0030	B		2nd entry point.
	4 00	0015	B		
2	41	0030	B		3rd entry point.
	4 00	0021	B		

#### Standard Procedure

4.4.5

The following procedure is used by library routines, and is recommended for all subroutines.

The subroutine is, if possible, written in one block and is headed by a blank block relativizer word. The block relativizer setting is inserted here by the main programmer. The subroutine is written without a block number. The user incorporates the subroutine into his program in the same way as he would a block of his own program, allocating a block number for it and setting the relativizer for the corresponding R.R.N. Where a subroutine extends over more than one block, R.R.Ns 99, 98, 97 ... are used, the appropriate R.R.Ns being stated on the specification sheet.

Recommended R.R.Ns are used for temporary storage, data, parameters and results. These are described later (4.7).

Indicators are used in the order 19, 18, 17 .... Unless the state of an indicator is an entry condition, no assumption is made about the initial states of indicators. Indicators used by the subroutine may be in either condition on completion of the subroutine.

Each subroutine is described by a specification sheet for which there are standard forms. The specification should contain all the necessary information for using the subroutine. The specification sheet contains the following information:

Title:	A brief title, giving an indication of what the routine does.
Description:	A concise description of what the routine does.
Entry Points:	The word at which entry is to be made. If there are several entry points, an indication of their differences.
Entry Conditions:	Relative addresses of any data or parameters used by the routine. Required state of any indicators used as entry conditions.
Results:	Statement of what results are produced and their relative addresses.
Storage:	The number of words occupied by the subroutine for program and constants. The number of words used as temporary storage, the contents on entry being immaterial.
Time:	If possible an exact time for execution of the subroutine or a formula from which the time can be calculated.
Limitations:	Any limitations of the routine, e.g. cases which are not catered for, or limitations on accuracy of results.
Program	
Indicators used:	A list of any indicators used by the subroutine.
Error Conditions:	Details of error conditions that may arise and of the appropriate action to be taken.
Notes:	Any further description required for using the routine.

## STORAGE ALLOCATION

## 4.5

When a program is ready to be tested, its storage position on the drum and in I.A.S. must be allocated. The program is stored on the drum when it is read by Initial Orders, and each block must therefore be given a drum allocation which does not overlap with that of another block. The I.A.S. allocation is that occupied by the block when it is transferred to I.A.S. Since it is possible that the whole program cannot be concurrently held in I.A.S., it is possible that several blocks may share the same I.A.S. allocation. Storage should also be allocated for any data areas required in I.A.S. and on the drum. It should be remembered that any block which is to be specified for the start of a drum transfer must be stored on the drum starting at the beginning of a decade.

The storage allocation should be recorded, and it is recommended that the I.C.T. storage charts are used for this purpose. It is likely that, during testing, some of the blocks may have to be increased and the storage allocation modified. Any such changes should be recorded as they are made.

## INITIAL ORDERS CONTROL WORDS

4.6

The required storage allocation is communicated to the Initial Orders program by means of control words. The control words are written on program sheets, punched in program cards and read by Initial Orders as part of the program pack. The control words include special control designations which are recognized by Initial Orders. They provide information for Initial Orders and are not stored in the machine as program.

The most common forms of the control words are described below. Full details of all possible uses of the control words, together with a full description of the operation of Initial Orders and the possible error conditions which can arise are given in the Initial Orders Manual.

### Control Designation 'R'

4.6.1

The control word having control designation 'R' is termed a relativizer word. Its purpose is to provide Initial Orders with the I.A.S. and drum starting addresses for a particular R.R.N. These relativizer settings are recorded in the machine and used during program reading to convert addresses referring to that R.R.N. from relative to absolute form. The relativizer control word must be read before any instruction referring to that R.R.N. The relativizer control words are normally punched three to a card and read in at the start of the program pack before the program instructions. During program testing, it may be preferable to punch only one relativizer control word to a card.

The form of the relativizer control word is as follows:

D	F	A	R
<i>R</i>	---	<i>I.A.S. Starting Address</i>	---
		<i>Drum Starting Address</i>	<i>R.R.N</i>

Hence the control word:

D	F	A	R
<i>R</i>	---	<i>0134</i>	---
	<i>01</i>	<i>1315</i>	<i>23</i>

sets relativizer 23 to an I.A.S. starting address of word 134 and a drum starting address of word 11315. A block may be stored starting at any word on the drum and therefore it is necessary to specify a drum word (not decade) address in the relativizer word. Note that the drum word address may exceed four digits and that it can therefore overflow into the function digits.



Instead of writing the absolute I.A.S. and drum addresses in a relativizer word, a relativizer can be set by making it relative to another relativizer which has previously been set. To achieve this the previously set relativizer is specified in the relativizer column of the first half of the control word.

Thus the control words:

D	F	A	R
R		0200	
		3500	10
R		0100	10
		0026	32

- (a) Set relativizer 10 to an I.A.S. word address of 200 and a drum word address of 3500.
- (b) Set relativizer 32 to an I.A.S. address 100 more than that of relativizer 10 and a drum address 26 more than that of relativizer 10. Hence relativizer 32 is set to an I.A.S. word address of 300 and a drum word address of 3526.

This technique is known as the use of relative relativizers. The use of relative relativizers is strongly recommended particularly when several blocks are stored consecutively on the drum. Their use can save many tedious alterations to the relativizer words if a block has to be extended during testing.

**Example** Consider the control words:

D	F	A	R
R		0000	
		0000	5
R		0023	5
		0023	16
R		0044	16
		0044	19

which achieve the following settings:

Relativizer	I.A.S. Address	Drum Address
5	0	0
16	23	23
19	67	67

Suppose now that an alteration is made to block 5 which makes it two words longer.

This means that the relativizer settings must be changed to:

Relativizer	I.A.S. Address	Drum Address
5	0	0
16	25	25
19	69	69

assuming that the blocks are still to be held consecutively on the drum and in I.A.S.

This can be achieved as follows:

D	F	A	R
R		0000	
		0000	5
R		0025	5
		0025	16
R		0044	16
		0044	19

Note that only the control word immediately following that for R.R.N. 5 has been altered.

#### Control Designation 'B'

#### 4.6.2

The control word having control designation 'B' is termed the block relativizer word. A block relativizer word should appear at the head of every block of program. It is standard practice to reserve the first card of the block (the B-card) for the block relativizer control word only.

The block relativizer word serves two main purposes:

- It indicates to Initial Orders the drum word address for storing the first word of the following block of program. The remainder of the block is stored in consecutive words on the drum following the first word.
- It specifies the I.A.S. and drum addresses for the current block relativizer. These are used during reading the block to convert any instructions with 'B' in the relativizer column from relative to absolute form. The block relativizer settings are recorded in the machine as those for R.R.N.2. R.R.N.2 must therefore not be used as an ordinary relativizer.

The form of the block relativizer control word is as follows:

D	F	A	R
B		I.A.S. Starting Address	
		Drum Starting Address	

Hence the control word:

D	F	A	R
B		0100	
		3240	

- Sets the starting address on the drum for the program block at word 3240.
- Sets relativizer 2 (i.e. relativizer B) to an I.A.S. address of 100 and a drum address of 3240.

A block may be stored starting at any word on the drum and therefore it is necessary to specify a drum word (not decade) address in the block relativizer word. Note that the drum word address may exceed four digits and it can therefore overflow into the function digits.

Instead of writing the absolute I.A.S. and drum addresses in a block relativizer word, the block relativizer can be set by making it relative to a previously set relativizer. To achieve this the previously set relativizer is specified in the relativizer column of the first half of the block relativizer control word.

Thus the control words:

D	F	A	R
<i>R</i>		0032	
		0160	14

D	F	A	R
<i>B</i>		0020	14
		0030	

- (a) Set relativizer 14 to an I.A.S. address of 32 and a drum address of 160.
- (b) Indicate that the following block of program is to be stored starting at word 190 on the drum.
- (c) Set the block relativizer to an I.A.S. address of 52 and a drum address of 190.

It is strongly recommended that block relativizers should be set relative to the relativizer corresponding to their own R.R.N.

**Example** Suppose that relativizer 10 has been set by the control word

D	F	A	R
<i>R</i>		0380	
		5010	10

Then consider the following in block 10

D	F	A	R
<i>B</i>		0000	10
		0000	
	37	0019	<i>B</i>
	35	0012	<i>B</i>

The first word of program is stored in word 5010 of the drum.

The block relativizer is set to the same I.A.S. and drum addresses as relativizer 10. Since the B in the relativizer column refers to block 10 this is evidently what is required.

The first word of program is stored in word 5010 of the drum as 370399350392.

This technique eliminates the need for altering the block relativizer control words if the drum allocation of the blocks is altered.

### Control Designation 'E'

4.6.3

The control word with designation 'E' is termed the entry word. The entry word is read after the last program word has been read. It is standard practice to reserve the last card of the program pack (the E-card) for the entry word only.

The purpose of the entry word is to inform Initial Orders which part of the program is to be transferred to I.A.S. and which word of the program is to be obeyed first.

The form of the entry word is as follows:

D	F	A	R
E		I.A.S. Address	
	No. of Decades	Drum Decade Address	

The effect of the E-word is as follows:

- The specified number of decades are transferred from the specified drum *decade* address into I.A.S. *starting at word 0*.
- A jump instruction is set up ready for control to be transferred to the specified I.A.S. word when the Start button is pressed.

Hence the control word:

D	F	A	R
E		0020	
	12	0290	

- Transfers 120 words of program from word 2900 on the drum, and stores them in words 0-119 in I.A.S.
- Prepares to transfer control to word 20 in I.A.S.

Once the instruction has been set up for control to be transferred to the specified word of I.A.S. the Initial Orders routine has been completed. After the Start button is pressed the computer is controlled by the user's program. It is possible to transfer a maximum of 200 words of program to I.A.S. using the E-word transfer. Any further transfers must be included in the program.

It is permissible to use relative addresses in the entry word although this is not generally recommended.

Thus the control words:

D	F	A	R
<i>R</i>	---	0000	---
		6000	10
<i>R</i>	---	0031	10
		0031	15
<hr/>			
	---	---	---
<i>E</i>	---	0000	15
	20	0000	10

- Set relativizer 10 to an I.A.S. address of 0 and a drum word address of 6000.
- Set relativizer 15 to an I.A.S. address of 31 and a drum word address of 6031.
- Transfer 200 words of program from drum words 6000-6199 to I.A.S. words 0-199.
- Prepare to enter the program at word 31 in I.A.S. (i.e. at word 0 block 15).

If a program has already been stored (and not overwritten) on the drum, then it is possible to enter it by reading the E-word under Initial Orders. In order to make use of this facility it is recommended that:

- The E-word should be punched on a separate card from the last program words.
- It should have a card number 1 so that it will satisfy the sequence check (see later) when not preceded by other cards.
- Absolute addresses should be used in the E-word. If relative addresses are used then the relativizer cards must be read first to set the necessary relativizers.

The above facility can be particularly useful during program testing when computer time can be saved by reading the program once and then entering it several times to test various conditions.

#### Control Designation 'C'

#### 4.6.4

The control word with control designation 'C' indicates to Initial Orders that it is required to zeroize a specified number of words on the drum. Its effect is exactly similar to writing a number of consecutive zero constants in a block of program, the use of the C-word being more convenient, however, particularly if there are a large number of words to be zeroized.

The form of the C-word is as follows:

D	F	A	R
C	Number of words to be zeroized		

the contents of the second half of the C-word are ignored by Initial Orders.

*Example*

I	D	F	A	R
	B			18
0-23	C		0024	
24	8	10		
		37	0030	4
25		62	0016	3
		57	0001	

The C control word causes the first 24 words of block 18 to be zeroized on the drum.

#### Control Designation 'F'

4.6.5

When the program instructions are all written in absolute form they can be punched with five words to a card instead of three. When the program words are punched five to a card they are said to be punched in fast-read form. The control word with designation 'F' indicates to Initial Orders that a specified number of words are punched in fast-read form. The remainder of the card following the F control word should be left blank and the words in fast-read form should start on the following card. When the fast-read words have been punched the remainder (if any) of the last card should be left blank and normal punching of three words to a card should start on the following card. When program is punched in fast-read form each word occupies only 12 columns on a card. The words must be punched in the form in which they are held in the machine, i.e. addresses must be absolute, the designation should be added in to the most-significant digit of the address and negative constants should be punched in complementary form. During fast read, a blank word of 12 columns is stored as a zero word on the drum.

The form of the F control word is as follows:

D	F	A	R
F	Number of words punched in fast-read form		

the second half of the F control word is blank.

**Example** The control words

D	F	A	R
B			
		2000	
F		0190	

cause the following 190 fast-read program words to be stored on the drum starting at word 2000.

**Note** It is normally recommended that fast-read form should be used only when a program has been fully tested. There is a library program available for punching program stored on the drum into fast-read cards. There is a standard format for fast-read cards and this is described in the Initial Orders Manual.

#### Control Designations 'P' and 'M'

4. 6. 6

Control designation 'P' indicates to Initial Orders that a positive constant is to be stored in the program. This constant is normally the actual number written in the Function and Address columns on the program sheet.

**Example**

BLOCK 30

I	D	F	A	R
10	P		26	
		02	1964	

causes a constant of 000 026 021 964 to be stored in word 10 of block 30.

However relativizers may be included in either or both halves of the word, Initial Orders treating the constant as an instruction-pair. The two halves are therefore normally relativized by the I.A.S. settings of the specified R.R.Ns. However if digit position 1 is an 8, the constant is treated as a drum transfer instruction and the second half is relativized by the drum setting of the specified R.R.N.

It should be noted that there is no carry between the two halves of an instruction-pair.

**Examples** Suppose that R.R.Ns 19 and 20 are set by the following control words:

D	F	A	R
R		230	
		430	19
R		117	
	01	1350	20

(a)

BLOCK 16				
I	D	F	A	R
5	P	---	10	19

causes a constant of 000000000240 to be stored in word 5 of block 16  
and b)

BLOCK 40				
I	D	F	A	R
26	P	12	4610	19
			23	20

causes a constant of 124840000140 to be stored in word 26 of block 40  
and c)

BLOCK 32				
I	D	F	A	R
19	P	89	6543	20
			0	19

causes a constant of 896660000043 to be stored in word 19 of block 32.

**Note** If the constant requires a drum relativizer setting, then digit position 1 *must* be an 8.

Control designation 'M' indicates to Initial Orders that a negative constant is to be created and stored in the program.

**Example**

BLOCK 10				
I	D	F	A	R
30	M	---	12	---

causes a constant of 999999999988 to be stored in word 30 of block 10.

Relativizers may be used in either half of the word as for the P designation.

**Example**

BLOCK 48				
I	D	F	A	R
25	M	---	30	34

where R.R.N.34 has an I.A.S. setting of 60 causes a constant of 999999999910 to be stored in word 25 of block 48.

**Note** When a constant with designation M has relativizers specified, it is converted to absolute form *before* negating.



## Initial Orders Sequence Check

4. 6. 7

While reading the program pack, Initial Orders carries out a sequence check to ensure that the cards are in the correct order. It is for this purpose that the block number and the card numbers within the block are punched on the program cards. In order for the sequence check to be passed an end of block marker should also be punched on the last card of each block. The end of block marker takes the form of a punching in column 17 which has a non-zero numeric component. The following conventions are recommended for the end of block marker:

Last card of a subroutine - Y

Last card of a complete program - Z

Last card of a block other than either of above cases - X.

The end of block marker should be written on the program sheet beneath the card number for the last block.

### Example

C	I	D	F	A	R
1	0	B	---	---	17
2	1	---	45 2	0002 0018	3 B

11 (x)	28	---	37 4 00	0000 0000	3 21
-----------	----	-----	------------	--------------	---------

It should be noted that all cards must have a card number, including those cards which contain control words only.

The following conventions are recommended:

- (a) Relativizer cards at the head of the program pack are numbered as if they were a block of program cards, the last relativizer card being punched with an end of block marker. It is usual to regard the relativizer cards as block 0.
- (b) The E-card should be regarded as a one word block and should not be numbered as the last card of the last program block. It is usual to make the E-card card 1 of block 999.

It is possible to include cards between consecutively numbered cards by including a suffix in a card column allocated for the purpose. Details of how to do this are given in the Initial Orders Manual.

Suffixed cards should only be included during program testing, the cards being renumbered when the program is proved.

## RELATIVIZERS

4.7

There are 99 relativizers with R.R.Ns 1 to 99, which may each be set to an I.A.S. and drum starting address by means of relativizer control words.

Certain standards have been adopted concerning the use of R.R.Ns in the general purpose routines. These are as follows:

- R.R.N.1 - This is used as a block for temporary storage, i.e. storage for counters and intermediate results.
- R.R.N.2 - This is used by Initial Orders for storing the block-relativizer settings. R.R.N.2 should not therefore be used as a relativizer number.
- R.R.N.3 - Input data to a subroutine.
- R.R.Ns 4 to 9 - Any other data areas required, input or output.
- R.R.Ns 99, 98..... - R.R.Ns given to subroutine block if these are necessary.

### Relative Addressing and the use of General Purpose Subroutines

4.7.1

It is evident that, although the allocation of a block number for the subroutine can normally be left to the user, the subroutine itself often needs to refer to data for which an R.R.N. must be specified. When this is necessary the above conventions are adopted.

When several general purpose routines are being incorporated into a program it is possible that different routines may make different uses of the same R.R.N. This is a problem which can be easily overcome by resetting the relativizers using relative relativizers.