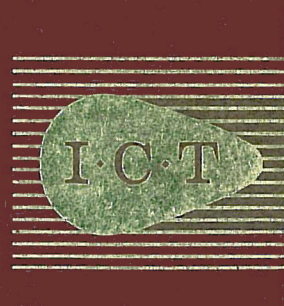
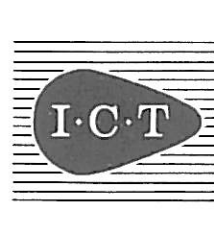


I.C.T. 1300 SERIES

**PROGRAMMERS
REFERENCE MANUAL**



COMPUTER
PUBLICATIONS



COMPUTER
PUBLICATIONS

THE GALDOR COMPANY (ELECTRONICS) LTD.

1-52 BINGHAM ROAD

SURBITON, SURREY, KT6 5PL 038

I.C.T. 1300 SERIES

PROGRAMMERS REFERENCE MANUAL

INTERNATIONAL COMPUTERS AND TABULATORS LIMITED

I.C.T. HOUSE · PUTNEY BRIDGE · LONDON S.W.6

© International Computers and Tabulators Limited 1964

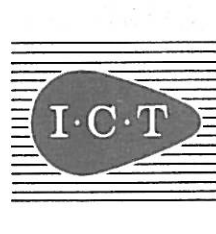
First Edition_____ March 1964

This manual is now the standard
Programmers Reference Manual
for the I.C.T. 1300-series Computers.

No further copies of existing
I.C.T. 1301 programming manuals will be available
but it is not necessary to replace them
with this new issue since they are not
thereby rendered out of date.

3165

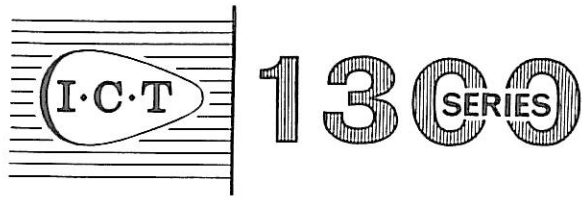
Printed in Great Britain by
International Computers and Tabulators Limited,
London



**AMENDMENT
RECORD**

TITLE

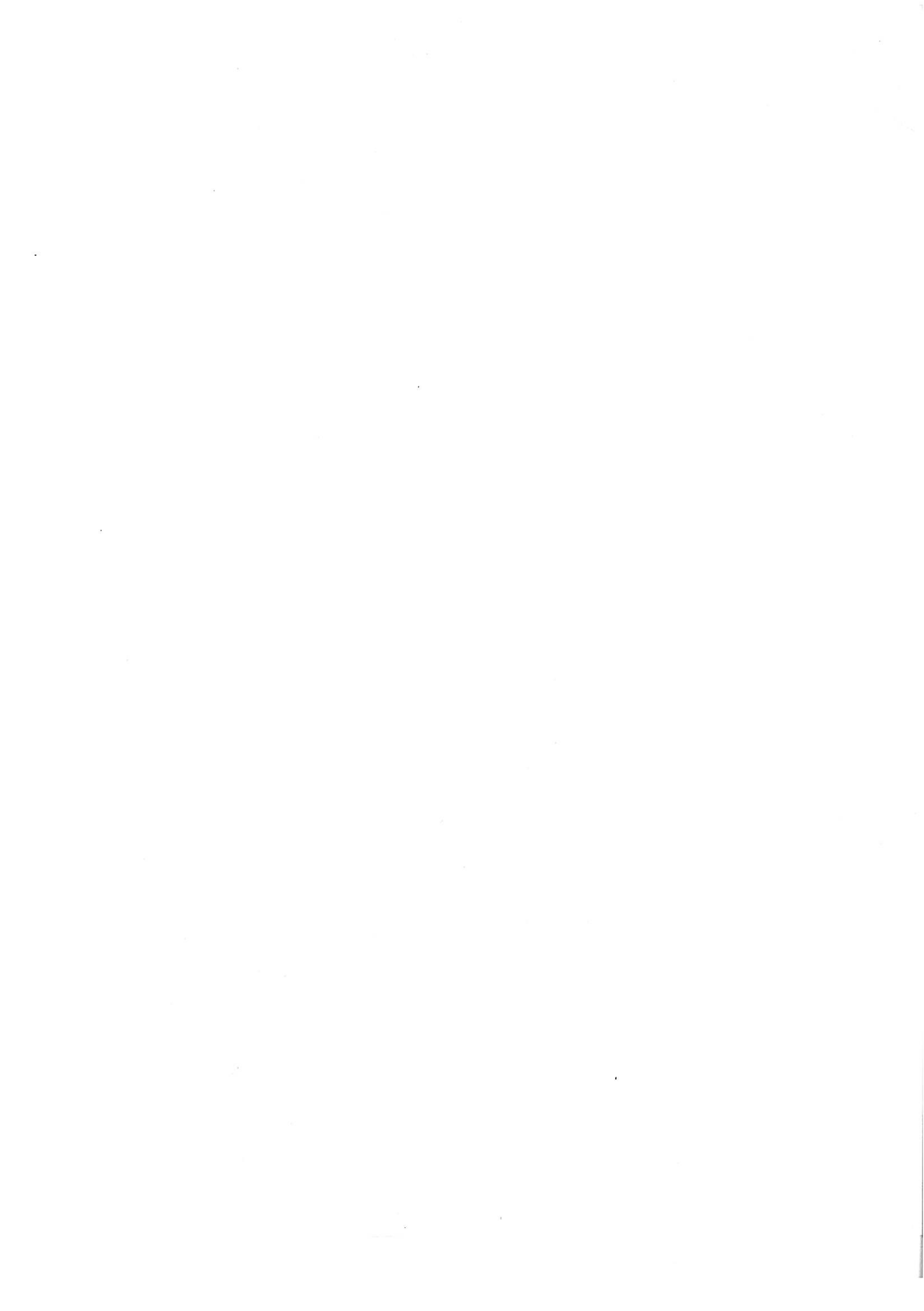
Amendment Number	Date	Chapter Page	Description

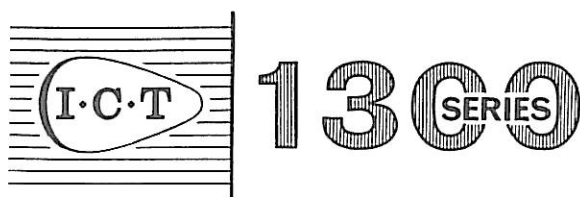


programmers reference manual

CONTENTS

- Part 1 Introduction
- Part 2 The Central Processor
- Part 3 Peripheral Equipment
- Part 4 Programming Techniques
- Part 5 Software Facilities
- Part 6 Reference Tables and Glossary





programmers reference manual

INTRODUCTION

Contents

	Page
1.1 The 1300-series Data Processing System... ..	1
1.2 Computer Storage	1
1.2.1 Immediate Access Storage (I.A.S.)... ..	1
1.2.2 Magnetic Drum Storage	2
1.3 The Arithmetic Unit	2
1.4 Indicators	4
1.4.1 Automatic Indicators	4
1.4.2 Programmed Indicators	4
1.4.3 Manual Indicators	5
1.5 Instructions	5
1.6 The Control Registers... ..	6
1.7 Magnetic Tape	6
1.7.1 One-inch Tape System	6
1.7.2 Half-inch Tape System... ..	7
1.7.3 Quarter-inch Tape System... ..	7
1.8 Input and Output Equipment	8
1.8.1 The Card Reader	8
1.8.2 The Card Punch	8
1.8.3 The Line Printer	8
1.8.4 The Paper-tape Reader	9
1.8.5 The Paper-tape Punch	9
1.8.6 The Interrogating Typewriter	10

Contents continued		Page
1.9	The Computer Console	10
1.10	Checking Facilities... ..	10
1.11	Addressing	11
1.12	Program Sheets	11
1.13	Relative Addressing	13
1.14	Program Cards	14
1.15	Instructions within the Computer	14
1.16	Initial Orders	15
1.17	Numbers within the Computer	16
1.18	Alphabetic Characters within the Computer ...	17
1.19	The Subroutine Library	18
1.20	1300-series System Configurations	19

Illustrations

Figure 1	The Arithmetic Unit	3
Figure 2	A Program Sheet	12
Figure 3	A Program Card	15

Part I

Introduction

THE 1300 - SERIES DATA - PROCESSING SYSTEM

1.1

The 1300-series data-processing system comprises a fast central processor incorporating magnetic core and magnetic drum storage, together with a wide range of peripheral equipment. Peripheral units provide punched-card and paper-tape input, punched-card, paper-tape and printed output, interrogating typewriter facilities and magnetic-tape storage. A summary of the main characteristics of the system is given in Part 6.

COMPUTER STORAGE

1.2

The store of the computer is divided into locations or words. Each word has a fixed size termed the word-length. The term 'word' usually refers to the contents of a location rather than to the location itself.

Each storage location can contain a number which may represent either program or data. There is a numerical code for each instruction which can be given to the computer, and a program consists of these instructions together with any constants which may be referred to by the program. Data are manipulated by the program and may consist of data numbers for calculations or alphabetic information held in coded form.

The word-length is twelve digits. Each digit is recorded in binary code and consists of four bits (i.e. *binary digits* with value 0 or 1) the '1' bits representing 1, 2, 4 and 8 respectively. By using a combination of bits, it is possible to record any number from 0 to 15 in one decimal digit position.

One word usually represents a 12-digit decimal number and in this instance a single digit lies in the range 0 to 9. The ability to record up to 15 in one position is, however, useful, particularly for holding 10 or 11 as one digit in the pence position when performing arithmetic operations in sterling.

Immediate Access Storage (I.A.S.)

1.2.1

The Immediate Access Storage (I.A.S.) is a magnetic-core store. Data held in I.A.S. are accessible instantly to the program, and program instructions are themselves obeyed from I.A.S.

The basic capacity of I.A.S. is 400 words. Each word is identified by an address which is a number in the range 0 to 399. The capacity of I.A.S. may be increased by units of 400 words up to a standard maximum of 2,000. Hence a particular computer may have 400, 800, 1200, 1600 or 2000 words of I.A.S. with addresses in the range 0 to 1999.

It is also possible for a computer to have, as an alternative unit, a 4,000-word I.A.S. with addresses in the range 0 to 3,999.

Magnetic Drum Storage

I.2.2

The capacity of I.A.S. is augmented by a second level of internal storage in the form of magnetic drums. Whereas any word in I.A.S. is available immediately, any word on the drum must first be transferred to I.A.S. before it is available to the program. The average access time to a word on the drum is in the order of a few milliseconds compared with an effectively zero access time to I.A.S.

A standard drum has 12,000 words of storage. A small computer may be fitted with a 3,000-word drum or a 6,000-word drum. Alternatively, a computer may have one or more standard drums up to a maximum of eight drums.

The drum is divided along its length into channels each containing 200 words. Each channel is further divided into 20 groups of 10 words each called decades. The decade is the smallest unit of transfer to or from the drum. The largest unit which can be transferred by one instruction is 20 decades or one channel. Data may be transferred to or from the drum either as a complete channel or as a number of consecutive decades. In the latter instance, provided that the decades are consecutive, they need not be confined to one channel or to one drum.

A series of read/write heads along the length of the drum transfers information to and from the drum as it rotates at a constant speed. Transfer is effected when the appropriate decade passes the corresponding read/write head.

Each drum has a further two channels (40 decades) of reserved storage. On the first drum on each machine these channels are used to retain permanently the Initial Orders program and engineers' test routines. Transfers can be made by the program from the reserved channels to I.A.S. but not vice versa.

THE ARITHMETIC UNIT

I.3

The arithmetic unit is that part of the computer where arithmetic calculations are performed. The unit consists basically of three registers A, B and C (Figure 1) that hold information and a Mill which carries out the actual additions and subtractions. (Multiplication is a built-in routine and is carried out in the Mill as a series of additions and subtractions.) Each register is one word length and can therefore hold a number consisting of twelve decimal or sterling digits. Program instructions cause numbers to be transferred from one register to another or to be added, subtracted or multiplied via the Mill.

Register A acts as the link with I.A.S. Every word entering or leaving I.A.S. is transferred via Register A. Hence, besides being used during arithmetical calculations, Register A is also used as follows:

- when transferring information between I.A.S. and Registers B or C,
- when transferring instructions from I.A.S. to be obeyed,
- when transferring information to or from magnetic tape,
- when transferring information to or from the magnetic drum,
- when performing logical functions involving a word in I.A.S.

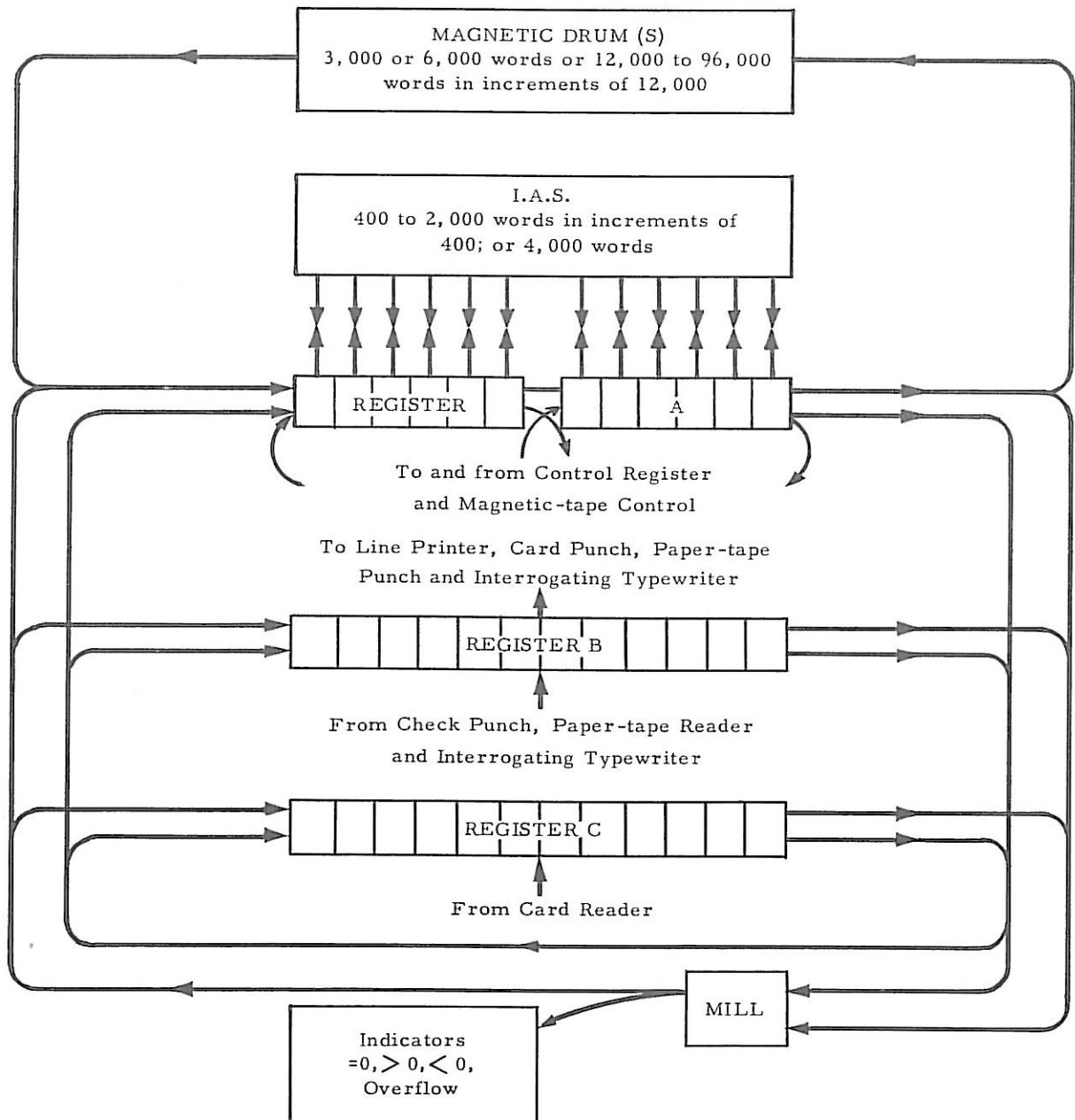


Figure 1: THE ARITHMETIC UNIT

Register B is involved in most arithmetical calculations and is the communicating link between the central processor and all peripheral units with the exception of magnetic-tape units and the card reader. It is also the only register in which numbers may be shifted i.e. displaced to the right or left within the register.

Register C is used during multiplication and for communication with the card reader.

As stated in 'Computer Storage', each digit of a number is represented by four bits. When information passes between a word of I.A.S. and Register A, the transfer takes place along 48 parallel paths, one for each data bit. When information is transferred between the registers and the Mill, the transfer takes place digit by digit, each digit being transferred along four parallel paths.

INDICATORS

1.4

Indicators are provided so that a program may determine whether or not a particular condition has been satisfied. An indicator is in one of two states, set or unset. When the state of an indicator is tested, there are two alternative pieces of program to be obeyed, one for a set condition and the other for an unset condition. If an indicator is tested and found to be set, then the normal sequence of instructions is interrupted and a jump takes place to another part of the program. If the indicator is unset, the next sequential instruction is obeyed.

Automatic Indicators

1.4.1

Automatic indicators serve as a communicating link between the computer and the program. These indicators are set, and in some cases unset, automatically by conditions arising in the computer or in the peripheral units. Automatic indicators associated with the Mill may be used to test whether a number is positive, negative or zero. There is also an overflow indicator that is set if an arithmetic operation gives a result which carries into the most-significant digit position. This is a warning to the programmer that subsequent arithmetic operations might produce a result that will be too large to be held in one location.

Indicators are automatically set if a transfer error occurs during transfer to or from I.A.S. or the drum, thus by programming an indicator test and an error routine the error can be detected by program and possibly corrected without manual intervention. The peripheral equipment is not synchronized with the computer and it is by testing the relevant indicators that a program determines when a unit is ready for use, before obeying instructions to control it.

There is also an automatic indicator that is permanently set enabling the program to jump unconditionally to another section of the program if this is necessary.

Programmed Indicators

1.4.2

Programmed indicators are entirely under program control and are set, unset, and tested by program. Where two pieces of program are largely similar, the programmer should be able

to combine them into one program using indicators to differentiate between the two cases where necessary.

Manual Indicators

I.4.3

Manual indicators enable operator action to affect the running of a program. These indicators are set and unset by manual controls (switches) on the console display panel. For example, a program might be written so that if a manual indicator is set, a complete table of results is printed, whereas if the indicator is unset, only a summary is printed. The choice is effected by the program testing an indicator to determine the switch setting.

INSTRUCTIONS

I.5

There is a range of permissible instructions which can be given to the computer. Each instruction has its own code number and the instructions are combined to form a coded program. Control may be transferred to a word of program stored anywhere in I.A.S.

A coded instruction is usually six digits in length and two instructions are normally held in one location. Some instructions (termed double-length instructions) are twelve digits in length and occupy a whole word.

The form and effects of the various instructions are described in Parts 2 and 3 and a summary is given in Part 6 of this manual. The following paragraphs form a guide to the different types of instruction available.

Transfer instructions enable information to be transferred between Registers A, B, C and I.A.S. or from Register C to Register B. When the transfer is to or from I.A.S., the particular I.A.S. word is specified as part of the instruction. There is also an instruction for transferring a group of consecutive words from one area of I.A.S. to another. The first words of the source and destination areas are specified in the instruction.

Arithmetic instructions cause decimal or sterling arithmetic to be performed, usually between Register B and a word of I.A.S. The I.A.S. word is specified in the instruction.

Logical instructions enable the programmer to store information in individual bits or digits of a word or to examine individual bits or digits of a word. Logical operations take place between Register B and a specified word of I.A.S.

Shift instructions cause the contents of Register B to be displaced in the register. The number of positions to be shifted is specified in the instruction.

Indicator-test instructions cause a jump to another part of the program if the indicator is set. If the indicator is unset, the next sequential instruction is obeyed. The indicator number and the I.A.S. word to which the jump is to be made if the test is successful are specified in the instruction. Programmed indicators are also set and unset by program instructions.

Magnetic drum instructions permit decades to be transferred from the drum to the I.A.S. and vice versa. The drum decade address and the I.A.S. location address at which the transfer is to commence are specified together with the number of decades to be transferred. All drum transfer instructions are double-length.

Peripheral instructions are used for controlling each of the peripheral units. Instructions to write to and read from magnetic tape are double-length. The I.A.S. word at which the transfer is to begin and the tape-unit address are both specified in the instruction.

THE CONTROL REGISTERS

1.6

There are three control registers: CR1, CR2 and CR3. Each is six digits in length and capable of holding one single-length instruction.

When a word of program is to be obeyed it is transferred from I.A.S. via Register A to CR1 and CR2. An instruction is actually obeyed from CR1. After the first instruction has been obeyed from CR1, the contents of the registers are shifted and the second instruction is moved from CR2 into CR1 to be obeyed.

The operation of the control registers is such that instructions are obeyed automatically in the same sequential order as they appear in I.A.S. When the two instructions in one word have been obeyed, the next word is automatically transferred through Register A to the control registers. The only instruction which causes this sequence to be broken is a successful indicator test.

The detailed operation of the control registers is described in Part 2.

MAGNETIC TAPE

1.7

Magnetic-tape units can be linked to the computer, and program instructions are available for reading from and writing to magnetic tape. By means of one instruction a block of consecutive words can be transferred from I.A.S. to tape or vice versa, the transfer being effected via Register A. There is no limit to the size of a block other than the size of I.A.S. and the remaining usable length of tape. Extra bits are automatically stored on the tape for checking purposes and transfer errors are detected during writing or reading. Magnetic tape has the advantage that reels of tape can be removed easily from the computer and the information on them preserved for future processing. Full details of the magnetic-tape systems are given in Part 3.

One - inch Tape System

1.7.1

This system comprises units recording information on one-inch wide magnetic tape. From one to eight units can be attached to the computer and each unit can be allocated an address in the range 1 to 8 by means of an eight-position switch. It is possible to read from one unit and write to another unit simultaneously.

Information is recorded on the tape along 16 tracks, the 16 bit-positions across the tape making one frame. The bits in one frame form two decimal digits of four bits each, the remaining eight bits being used for checking purposes. The checking system is such that single-bit errors are precisely detected and are automatically corrected on reading. One word is held on the tape in six frames.

The packing density on the tape is 300 frames (i.e. 600 digits) to the inch and the tape moves past the read/write heads at a speed of 150 inches a second. Information is thus transferred at 90,000 digits a second (90 kc/s). The maximum length of tape on each spool is 3,600 feet.

Half - inch Tape System

1.7.2

This system comprises units recording information on half-inch wide magnetic tape. From one to eight units may be attached to the computer and each unit can be allocated an address in the range 1 to 8 by means of an eight-position switch. It is possible to read from one unit and write to another unit simultaneously.

Information is recorded along the tape in ten tracks, the ten bit-positions across the tape making one frame. The bits in one frame form one decimal digit of four bits, the remaining six bits being used for checking purposes. The checking system is such that single-bit errors are precisely detected and are automatically corrected on reading. One word occupies twelve frames.

The packing density on the tape is 300 digits to the inch and the tape moves past the read/write heads at a speed of 75 inches a second. The transfer rate is therefore 22,500 digits a second ($22\frac{1}{2}$ kc/s). The maximum length of tape on each spool is 3,600 feet.

Quarter - inch Tape System

1.7.3

Four, six or eight units can be attached to the computer, each unit having a fixed address in the range 1 to 8. It is possible to read from one unit and write to another unit simultaneously.

The tape width is a quarter of an inch. Information is recorded by different digits being represented by different distances between signals on the tape. Information is recorded over the whole width of the tape and the value of a digit is determined by the time taken for a tape length to pass the read/write heads. For reading purposes the tape is divided into two tracks and reading may take place from either track. Extra information is held on the tape at the end of a block for checking purposes.

As the digit values are recorded by a length of tape, the packing density depends on the value of the data recorded. Assuming random distribution, the packing density is approximately 440 digits to the inch and the tape moves past the read/write heads at a speed of $37\frac{1}{2}$ inches a second. The transfer rate is therefore 16,500 digits a second (nominally 16 kc/s). The maximum length of tape on each spool is 1,800 feet.

INPUT AND OUTPUT EQUIPMENT

1.8

Input equipment provides the means of reading information into the computer and may consist of a card reader, a paper-tape reader and an interrogating typewriter, the last also being an output unit.

Information is presented to the card and paper-tape readers (in coded form) as holes punched in cards or paper tape and the readers convert this information into computer code.

The interrogating typewriter converts information typed on the keyboard directly into computer code. This information, together with the reply are then reproduced in plain language by the computer on the print unit of the typewriter.

Output equipment provides the means of obtaining the results of data processing and may consist of a card punch, a paper-tape punch and a line printer. The results may be printed in plain language or be produced (in coded form) as holes punched in cards or paper tape.

Full details of input and output equipment are given in Part 3.

The Card Reader

1.8.1

The card reader reads into the computer information punched on 80-column rectangular hole cards. The information is punched in code with one numeric or alphabetic character to a column. The characters are read into Register C, six columns at a time, and are then stored by program in I.A.S. Each character is represented in the computer by two digits termed the zone and numeric components.

The cards are read column by column, first by a reading station and then by a check-reading station. The two readings are automatically compared and an error indicator is set if they do not agree.

Two types of card reader are available, one reading up to 600 cards a minute and the other up to 300 cards a minute.

The Card Punch

1.8.2

The card punch punches information received from the computer into 80-column cards. The information is transmitted to the punch via Register B and the cards are punched row by row starting at the top of the card. When a card has been punched, the information is read back into the computer and compared by program with the original punch data.

The card punch operates at a maximum speed of 100 cards a minute.

The Line Printer

1.8.3

The line printer prints information received from the computer via Register B. The information is printed, one line at a time, on continuous stationery.

Printing is accomplished by activating print hammers which strike the paper and a carbon ribbon against a print barrel which rotates at a constant speed. All available characters (i.e. letters of the alphabet, numerals and special symbols) are embossed around the print barrel for each print position so that axial lines of similar characters are displayed on the surface of the barrel. Each character may be printed in any position along the line and spaces may be programmed.

As each line of characters on the barrel comes opposite the hammers the computer sends an impulse for each print position at which that character is to be printed. An indicator is set if a timing error arises.

Vertical line spaces can also be achieved by program instructions. The line printer may be one of the following types:

- (a) 120 print positions, with a maximum speed of 600 lines a minute,
- (b) 120 print positions, with a maximum speed of 300 lines a minute,
- (c) 80 print positions, with a maximum speed of 300 lines a minute.

The Paper - tape Reader

1.8.4

One or two paper-tape readers may be fitted to the computer. The paper-tape reader reads information punched in code on five, six, seven or eight tracks, the number of tracks being manually selected by a switch.

Information is read into Register B and then stored in I.A.S. by program. Each character is represented in Register B by two digits; the zone and numeric components.

In the case of 7- or 8-track paper tape, one track is reserved for a check bit. In these cases an automatic check is carried out on reading and an indicator is set if there is an error.

The feeding spool can hold up to 300 feet of paper tape and the reader operates at a maximum speed of 1,000 characters a second.

The Paper - tape Punch

1.8.5

The paper-tape punch punches information into 5-, 6-, 7- or 8-track paper tape, the number of tracks being selected by an engineer's adjustment. A character is punched from its zone and numeric components held in Register B. On 7- and 8-track tape, a check bit is automatically generated and punched in the track reserved for that purpose. The punched character is then check-read at a special sensing station and an indicator is set if an error is detected. There is no such check when punching takes place on 5- or 6-track tape.

The feeding spool can hold up to 800 feet of blank tape and up to 300 characters a second can be punched.

The Interrogating Typewriter

1.8.6

The interrogating typewriter can be used to request information which is readily accessible to the program but which is not normally required as printed output and is therefore not printed by the line printer.

The typewriter is used as an input and an output unit. A message entered on the keyboard is read by program into Register B and is typed (in red) on the typewriter. Acting on the message the program causes the reply to be transferred from Register B to the print unit of the typewriter. This information is typed in black.

All keyboard characters are available for both type-in and type-out. Each character is represented in Register B by its zone and numeric component.

The typewriter prints one character at a time and operates at up to ten characters a second.

THE COMPUTER CONSOLE

1.9

The computer console is a panel containing the switches necessary for operating the computer, and various lamps which indicate conditions within the computer. In addition, each peripheral unit has its own Switch and Indicator panel.

Some of the information which may be displayed on the console is as follows:

- the contents of Registers A, B and C,
- the contents of the control registers CR1, CR2, and CR3,
- the state of programmed indicators,
- the state of the transfer error indicators.

It is also possible to set numbers in Registers A, B and C and the control registers manually.

Instructions for operating the computer and the uses of the switches and indicators are contained in the Operators' Reference Manual. Charts of the various switch and indicator panels are shown in Part 6 of this manual.

CHECKING FACILITIES

1.10

The checking facilities associated with the input and output units and magnetic-tape units have already been mentioned. Facilities are also provided for detecting errors in transfers to and from I.A.S. and the magnetic drum.

A word in I.A.S. contains 48 data bits and two extra bits for checking purposes only. These bits are generated immediately before a transfer to I.A.S. and are used for checking purposes when the word is subsequently transferred from I.A.S. If the check fails an error indicator is set.

A word on the drum contains twelve data digits and an extra digit for checking purposes only. This digit is generated immediately before a transfer to the drum and is used for checking purposes when the word is subsequently transferred from the drum. If the check fails an error indicator is set.

The I.A.S. and drum checking systems are described in more detail in Part 2.

ADDRESSING

I.11

A single-length six-digit instruction is composed of two parts, a two-digit function code and a four-digit address part.

The function code number specifies a particular computer operation (e.g. multiply); the address part usually specifies the location of the information on which it is required to perform that operation. The address is usually the location of a word in I.A.S.

For example, one available function is "add the contents of Register B to a word of I.A.S. and store the result in the I.A.S. location from which the word originated". When it is desired to perform this operation the six-digit instruction contains the two-digit code number for the operation and the four-digit address of the I.A.S. word.

The I.A.S. word specified in the address part of an instruction may be program or data. In an indicator test instruction for example, the address is the program word to which the jump is to be made if the test is successful, whereas for an arithmetic instruction, the address is that of a data word.

When an indicator test instruction is used, a jump to another word of program causes the first instruction in that word to be obeyed. It is not possible to program a jump to a single-length instruction held as the second half of a word.

Drum transfer instructions are double-length and a drum decade address is specified as well as an I.A.S. address.

PROGRAM SHEETS

I.12

A program is written as a sequence of coded instructions, and this is done using the printed form shown in Figure 2.

This form is completed as follows:

- C** The number of the card within the block, numbering the first card, 1.
- I** The relative address of a word in the block, starting at zero. The word itself contains two single-length instructions (occupying columns D, F, A and R) separated on the program sheet by a broken line. A double-length instruction occupies both halves of a word and must not appear as the second half of one word and the first half of the next.

I.C.T COMPUTERS

1300 SERIES PROGRAM SHEET			JOB				BLOCK No.
			PROGRAMMER:-				SHEET No. /
							/ /
C	I	D	F	A	R	NARRATIVE	

Figure 2: A PROGRAM SHEET

- D** Designation column. Instructions to set, unset or test indicators contain a designation in this column as part of the instruction.
 - F** Contains the first part of the instruction, that is, the two-digit function-code number, or the indicator number in the case of an indicator instruction.
 - A** Contains the second part of the instruction, that is, the four-digit address.
 - R** Contains the relativizer reference number for the address. (If the address in Column A is that of a word in the same block then this column usually contains the letter 'B'.)
- Narrative** Used by the programmer for explanatory notes.

Constants are written on the program sheet as pseudo-instructions. A negative constant (not sterling) has 'M' written in Column D of the first half of the word and a positive constant may have 'P' written in this position though the 'P' is not essential.

RELATIVE ADDRESSING

I.13

When a program is being written it is convenient to split the program into sections which can be written more or less independently of one another, possibly by different programmers.

When a program is run, I.A.S. and drum are used to store the various sections of program and the necessary data. It is not usually possible to hold all of this information in I.A.S. at once and instructions must be included in the program to bring into I.A.S. the sections of program and data as they are required.

In writing a program, frequent references are made (in the address parts of instructions) to other words of I.A.S. and to words on the drum. These words may be data or program, in either the same section or another section of the program. During the early stages of writing a program it will not be possible to ascertain the exact I.A.S. location of each word because the lengths of the various sections of program are not known. To overcome this difficulty, a system of relative addressing is used.

The program and data are divided, as stated above, into sections, which are termed *blocks*. The words in each block are numbered in sequence, the first word being numbered zero. This sequence number is the relative address, i.e. it is not the actual address in storage but is the address *relative* to the first word of the block. (The actual address of a word in I.A.S. or on the drum is termed its absolute address.)

When a word of program (say, for example, word 3 of a block) refers in the address part of its instruction to another word (say, word 8) in the same block then the address part of word 3 will contain relative address 8 followed by the letter B denoting that word 8 is in the same block as word 3.

The blocks are given reference numbers termed *Relativizer Reference Numbers* (usually abbreviated to R.R.N.) and these numbers are used by the computer to convert relative addresses into absolute addresses.

For identification purposes, each program or data block is also given a block number and it

is usual to make this the same as the R.R.N. A word in a block would then be addressed by its relative address within the block and the block number, e.g. word 10 of block 15.

When the lengths of the various blocks are known it is possible to allocate storage in I.A.S. and on the drum. After this has been done, control words are written which set the I.A.S. and drum address for the first word of each block.

Each block is headed by a control word called its *block relativizer*. As the block is read into the computer, the block relativizer is used to convert into absolute addresses those addresses that refer to other words in the same block.

For each Relativizer Reference Number, there is another control word which sets the starting address for that relativizer in I.A.S. and on the drum. As the blocks are read into the computer the relativizer setting is used to replace by absolute addresses any address relative to that R.R.N.

Example If a program instruction has in its address part the I.A.S. address 'word 5 of block 16' and the R.R.N. for block 16 is set to a starting address of 150 then the absolute address referred to in the instruction is word 155.

The use of relativizers is fully described in the Initial Orders Manual.

The conversion of relative to absolute addresses takes place as the program is read into the computer. Before the program is obeyed, all addresses are absolute.

PROGRAM CARDS

I.14

When a program has been written on program sheets, the program is punched in code into program cards. The information in Columns I and Narrative is for the programmer's use only and is not punched. The block number and card number are punched into the cards so that a check can be carried out that the cards are in the correct sequence. The information in Columns D, F, A and R occupies nine columns on the card.

Three program words (i.e. six single-length instructions) are punched in each card and the columns are allocated as shown in Figure 3. Column 18 can be used as an extension to the card number when it is necessary to insert cards between sequentially numbered cards. Column 17 is used for a special marker on the last card of each block. This is used as an extra check that the cards are in correct sequence. Columns 1 to 16 are not allocated and are available for program reference purposes.

INSTRUCTIONS WITHIN THE COMPUTER

I.15

A single-length instruction occupies nine digit positions on a program sheet and nine corresponding columns of a program card. During read-in the instruction is condensed to six-digit form within the computer. This is achieved as follows:

Figure 3: A PROGRAM CARD

The R.R.N. is used to convert the instruction address to absolute form. Where the M control designation occurs the correct negative constant is formed. An indicator-test, set, or unset designation is recorded in the six-digit instruction by adding it to the most-significant position of the address.

Example An unconditional jump (test of indicator 00) to word 16 of block 32 is written on the program sheet as:

D	F	A	R
4	00	0016	32

If R.R.N. 32 has an I.A.S. setting of 23, the instruction is held in the computer as 004039.

Two instructions are held in the computer in one twelve-digit word as follows:

Digit Positions	1	2	3	4	5	6	7	8	9	10	11	12		
Instruction	FUNCTION		ADDRESS				FUNCTION		ADDRESS					
	First Instruction												Second Instruction	

INITIAL ORDERS

I.16

Initial Orders is a special program which enables other programs to be read into the computer. It reads the cards, stores the program instructions on the drum and then transfers a specified portion of the program to I.A.S.

Initial Orders is stored permanently on the reserved channel of the drum. A special button on the console causes Initial Orders to be transferred to I.A.S. ready to be obeyed when the Start button is pressed.

The input to Initial Orders is a pack of program cards. This pack contains program, constants and control words. The control words provide information for Initial Orders but are not stored in the computer program. A sequence check is carried out to ensure that the cards are in correct order.

Blocks of program are read by Initial Orders and within each block the words are stored in sequential locations on the drum. The drum address for the first word of each block is given by the block relativizer control word.

Initial Orders sets relativizers from control words read in for each R.R.N. and uses these relativizer settings to convert relative addresses into absolute addresses. Set, unset, or test designations are also added into the most-significant digit of the address.

If the card columns for a word are all blank, then that word is ignored. Provided that a zero constant has at least one zero punched, the P designation for a positive constant is not necessary. An M designation causes Initial Orders to store the constant in negative form.

It is permissible for addresses to be punched in absolute form so that Column R will be blank. Relative addresses should, however, be used since this saves programming and testing time. In particular, if relative addresses are used, storage re-allocation can be easily achieved if necessary by changing only the control words. Initial Orders and control words are described in more detail in Part 4.

As an instruction in absolute form is only six digits long, five words can be punched on a card instead of three. Cards punched in absolute form are called fast-read cards and are acceptable to Initial Orders. When a program has been fully tested the program pack can then be converted from ordinary program cards to fast-read cards, giving a smaller pack and faster read-in time. Full details of all Initial Orders facilities are given in the Initial Orders Manual.

NUMBERS WITHIN THE COMPUTER

I.17

One word of storage can hold a number consisting of twelve decimal or sterling digits. For arithmetic purposes, the most-significant digit is called the sign digit. Negative numbers are held as a complement of ten (or corresponding sterling complement) and the sign digit is therefore normally zero for a positive number and nine for a negative number.

The sterling positions in a word can be varied by program. Assuming that pence are in the least-significant digit position, the following examples show how numbers are represented in the computer.

Numbers Represented	Representation in Computer											
	1	2	3	4	5	6	7	8	9	10	11	12
365	0	0	0	0	0	0	0	0	0	3	6	5
£987.16.9	0	0	0	0	0	0	9	8	7	1	6	9
-6921	9	9	9	9	9	9	9	9	3	0	7	9
-£123456.19.2	9	9	9	8	7	6	5	4	3	0	0	10

Positive and negative constants entered by Initial Orders are held in the form shown. Initial Orders cannot accept sterling constants with an M designation. Such constants must be converted into sterling complements and then entered as if positive.

ALPHABETIC CHARACTERS WITHIN THE COMPUTER

1.18

Alphabetic characters are held in the computer in coded form. Each character is represented by two digits (called the zone and numeric components) and it is therefore possible to hold up to six alphabetic characters in one twelve-digit word of storage.

All characters punched on cards or paper tape are read into the computer with a zone and numeric component. For a numeric character, the zone component is always one, and the numeric component is equal to the number itself. The zone component is usually discarded (by program) for data numbers and the twelve digits of a word are used to hold only the numeric components of the data numbers. The zone component is re-generated before a data number is punched or printed.

The zone component is retained throughout all stages of data processing for alphabetic information or mixed alphabetic and numeric information.

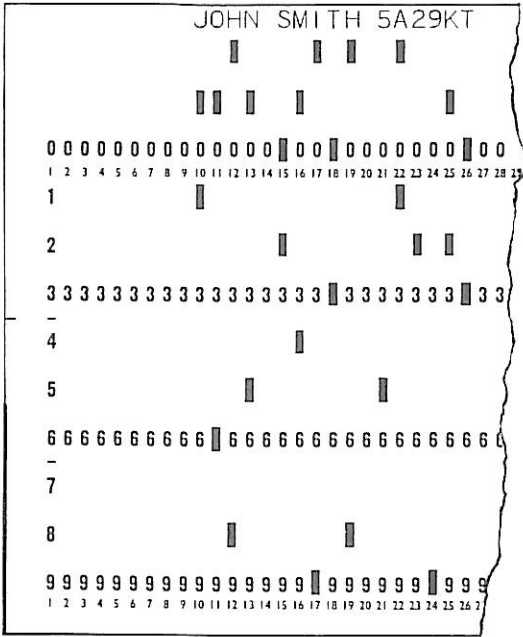
When information is read from cards, six characters are read at a time, the zone components being held in the six most-significant digits of the register and the numeric components being held in the six least-significant digits. The characters are then normally stored as consecutive words in I.A.S.

The standard card punching code together with the zone and numeric components recorded in the computer are given in the table below.

Card Numeric Punching and Computer Numeric Component	Over-Card punching				
	10	11	0	1	
	Computer Zone Component				
	1	2	3	4	5
10	10				
11	11				
0	0				
1	1	A	J	&	
2	2	B	K	S	%
3	3	C	L	T	$\frac{1}{4}$
4	4	D	M	U	-
5	5	E	N	V	/
6	6	F	O	W	$\frac{1}{2}$
7	7	G	P	X	.
8	8	H	Q	Y	@
9	9	I	R	Z	$\frac{3}{4}$

Example

On Card :-



Might be held in I.A.S. as :

Location 100	0	3	3	2	3	0	0	1	6	8	5	0	(JOHN)
Location 101	4	3	2	4	2	0	2	4	9	3	8	0	(SMITH)
Location 102	1	2	1	1	3	4	5	1	2	9	2	3	(5A29KT)
	Zone						Numeric						

THE SUBROUTINE LIBRARY

I.19

Certain sections of program are common to a variety of tasks. Economies may be effected in programming and testing time if these programs (subroutines) once written are preserved for future use. A library of these subroutines for the 1300-series is maintained by International Computers and Tabulators Limited.

The subroutines may be incorporated into a program using the relative addressing system. An index is maintained of all available routines and a specification sheet for each subroutine gives full instructions on how the subroutine can be used.

General purpose subroutines are termed 'software' since they can be used without programming effort but are not built into the 'hardware' of the computer. General subroutines are available for control of all input and output units and magnetic-tape units.

Besides subroutines which can be included in programs, certain complete routines are available for which the user need supply only data to provide the required result.

The available facilities are described in more detail in Part 5.

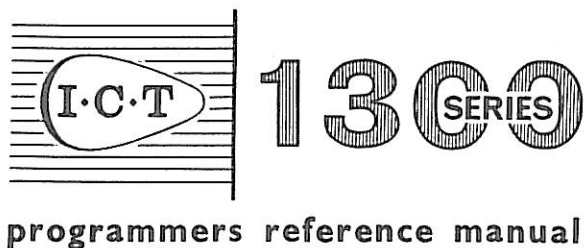
1300 - SERIES SYSTEM CONFIGURATIONS

I.20

The 1300-series Computer Systems can be considered in two sections: the Basic System, and those units which can be added to increase the capacity and scope of the system.

	1300	1301
Basic System		
Central Processor with		
Card Reader	80 column, 300 cards a minute	80 column, 600 cards a minute
Card Punch	80 column, 100 cards a minute	80 column, 100 cards a minute
Printer	80 position, 300 lines a minute	120 position, 600 lines a minute
I.A.S.	400 words	400 words
Drum	3,000 words	12,000 words
Additive Capacity		
Printer	Replacement by 120-position printer.	
I.A.S.	Up to four additional 400-word units	Up to four additional 400-word units or one replacement unit of 4,000 words
Drum	May be increased to 6,000 words or 12,000 words. Further 12,000-word drums can then be added up to a total of eight drums.	Additional 12,000-word drums up to a total of eight drums
Alternative Feature		
Card Reader	40 col., 600 cards a minute	40 col., 600 cards a minute
Additional Features (1300 and 1301)		
Magnetic-tape:	Quarter-inch (16 kc/s), 4, 6 or 8 units on a machine or half-inch (22½ kc/s), up to 8 units on a machine or one-inch (90 kc/s) up to 8 units on a machine	
Paper-tape Readers:	5-, 6-, 7- or 8-track, 1,000 characters a second, 1 or 2 a machine	
Paper-tape Punch:	5-, 6-, 7- or 8-track, 300 characters a second, 1 a machine	
Typewriter input/output	120 positions, 10 characters a second, 1 a machine	





THE
CENTRAL
PROCESSOR

Contents

	Page
2.1 GENERAL	1
2.1.1 The Arithmetic Unit	1
2.2 TRANSFER INSTRUCTIONS	2
2.2.1 Function 37	2
2.2.2 Function 40	3
2.2.3 Function 41	4
2.2.4 Function 42	5
2.2.5 Function 43	6
2.2.6 Function 44	6
2.2.7 Function 45	7
2.3 DECIMAL ADDITION AND SUBTRACTION	9
2.3.1 Function 60	10
2.3.2 Function 61	11
2.3.3 Function 62	12
2.3.4 Function 63	13
2.3.5 Function 64	14
2.3.6 Function 65	15
2.3.7 Function 66	16
2.3.8 Function 67	17
2.3.9 Function 68	18

Contents continued		Page
2.4	STERLING ADDITION AND SUBTRACTION	19
2.4.1	Function 22	19
2.4.2	Function 70	20
2.4.3	Functions 70 to 78	21
2.4.4	Function 71	21
2.4.5	Function 72	21
2.4.6	Function 73	21
2.4.7	Function 74	22
2.4.8	Function 75	22
2.4.9	Function 76	22
2.4.10	Function 77	22
2.4.11	Function 78	22
2.5	MULTIPLICATION AND THE DECIMAL POINT REGISTER	23
2.5.1	Decimal Multiplication	24
2.5.2	Function 69	24
2.5.3	Sterling Multiplication	24
2.5.4	Function 79	25
2.5.5.	The Decimal Point Register	25
2.5.6	Function 21	26
2.5.7	Decimal Multiplication Examples	27
2.5.8	Sterling Multiplication Example	29
2.5.9	Negative Factors in Multiplication	30
2.5.10	Multiplication Logic	32
2.6	LOGICAL FUNCTIONS	34
2.6.1	Function 35 (Logical AND)	34
2.6.2	Function 36 (Logical OR)	37
2.7	ROW BINARIZING	39
2.7.1	Function 30	40
2.7.2	Function 31	41
2.7.3	Function 32	43
2.7.4	Function 33	43
2.7.5	Function 34	43
2.7.6	Example of the Combined Use of Functions 30 to 35 ..	44
2.8	SHIFT INSTRUCTIONS	44
2.8.1	Function 54	45
2.8.2	Function 55	46
2.8.3	Function 56	46

Contents continued		Page
2.8.4	Function 57	48
2.8.5	Application of the Shift Functions	48
2.9	'DO NOTHING' AND 'STOP' FUNCTIONS	50
2.9.1	Function 00 (Do Nothing)	50
2.9.2	Function 11 (Stop)	51
2.10	INDICATORS	51
2.10.1	Representation of an Indicator Test Instruction	51
2.10.2	Incorrect Test Instructions	53
2.10.3	Automatic Indicators	53
2.10.4	Indicator 00	53
2.10.5	Mill Indicators	54
2.10.6	Indicator 01	55
2.10.7	Indicator 02	55
2.10.8	Indicator 03	55
2.10.9	Indicator 04	56
2.10.10	Indicator 06	56
2.10.11	Indicator 07 (Drum Check Error Indicator)... ..	57
2.10.12	Programmed Indicators	57
2.10.13	Indicators 10 to 19	57
2.10.14	Manual Indicators 20 to 29	58
2.11	THE MAGNETIC DRUM	59
2.11.1	Drum Instructions	61
2.11.2	Function 80	61
2.11.3	Function 81	62
2.11.4	Function 82	63
2.11.5	Function 83	63
2.11.6	Reserved Storage	64
2.11.7	Function 84	64
2.11.8	Function 85	64
2.11.9	Function 86	65
2.11.10	Function 87	65
2.11.11	Relative Addressing of Drum Instructions	65
2.12	THE CONTROL REGISTERS	66
2.13	PARITY CHECKING SYSTEMS	70
2.13.1	I.A.S. Parity Checking	70
2.13.2	Drum Parity Checking	71
2.14	TIMINGS	71

Illustrations

Figure 4	The Mill Indicators and Functions 60 to 68	9
Figure 5	Typical Portion of Program for Sterling Calculations	23
Figure 6	Examples of Indicator Test Instructions	53
Figure 7	Setting Indicators 01, 02, 03 and 04	55
Figure 8	Example Instructions Testing Indicators 01, 02, 03 and 04	56
Figure 9	Magnetic Drum	59
Figure 10	The Control Registers	67

GENERAL

2.1

The central processor consists of the units that are responsible for the overall organization of the routing and processing of data within the system, and comprises the arithmetic unit, the control registers and associated circuitry.

The Arithmetic Unit

2.1.1

The arithmetic unit adds, subtracts and multiplies; division is not a built-in routine but is accomplished by subroutines. The arithmetic unit also performs shift functions, transfer functions, row-binary functions and logical functions (AND, OR). The unit consists of:

- three twelve-digit registers - A, B and C,
- the Mill and associated indicators,
- a one-digit Sterling Position Register,
- a one-digit Decimal Point Register and
- a one-digit Row Binary Register.

Arithmetic operations are performed on numbers held in Registers A, B and C. Program instructions are available for transferring numbers between the registers and I.A.S. and for carrying out the arithmetic.

Register A is the link between I.A.S. and the arithmetic unit. On completion of a transfer to or from I.A.S., Register A contains the same number as the specified word of I.A.S. Register A is also used when instructions are transferred from I.A.S. to the control registers to be obeyed. It should therefore be noted that when control is transferred from one word of program to the next, the contents of Register A are destroyed.

Arithmetic operations are usually carried out between Register B and a word of I.A.S. For example, during addition, the sequence of operations is as follows:

The contents of the I.A.S. location (word) specified in the add instruction are transferred to the arithmetic unit via Register A and are added in the Mill to the contents of Register B. According to the add instruction given, the result is placed either in Register B or in the I.A.S.

location from which the word originated. In the latter instance, the result is also left in Register A and the contents of Register B are unaffected.

The only arithmetic operation in which Register C is used is multiplication and the process is explained under "Multiplication".

For the instructions described in this part of the manual, the contents of Registers A, B and C may be assumed to be unaltered unless otherwise specified. In particular, the contents of a register are unaltered when they are transferred to another register or to a word of I.A.S. A complete summary of instructions is given in Part 6 together with a chart showing which registers are involved when an instruction is obeyed.

The Mill accomplishes addition digit by digit, commencing with the least-significant digit. Decimal or sterling carries are delayed and are added in with the next (more significant) digit. Although a word may contain up to 15 in one digit position, it is not normally advisable to give instructions for such numbers (10 to 15) to pass through the Mill. Associated with the Mill are indicators that can be tested to ascertain whether the last number to pass through the Mill was positive, negative or zero.

The Sterling Position Register, Decimal Point Register and Row Binary Register are discussed in detail later in this part of the manual.

TRANSFER INSTRUCTIONS

2.2

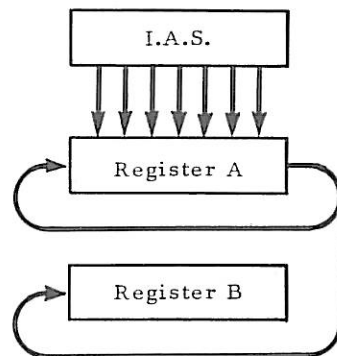
Transfer instructions move data between I.A.S. locations and registers or from one register to another without the data being processed arithmetically. As data transferred by-pass the Mill, the Mill Indicators are not affected.

Function 37

2.2.1

Effect Transfers the contents of the specified location of I.A.S. to Register B.

Operation The content of I.A.S. location overwrites the previous content of Register B. The data reach Register B by way of Register A and thus Register A also contains the contents now in Register B. The contents of I.A.S. remain unaltered.



Example Transfer word 42 of block 11 to Register B.

Instruction

D	F	A	R
	37	0042	11

Before

I.A.S.	0	0	0	0	0	0	0	5	7	9	4	1
Register A	0	0	0	0	0	0	1	4	7	6	8	2
Register B	0	0	0	0	0	0	0	0	5	6	3	

After

I.A.S.	0	0	0	0	0	0	0	5	7	9	4	1
Register A	0	0	0	0	0	0	0	5	7	9	4	1
Register B	0	0	0	0	0	0	0	5	7	9	4	1

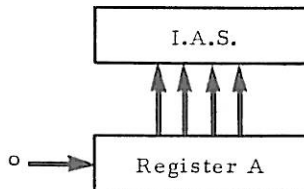
Notes As data being transferred by-pass the Mill, digits from 0 to 15 in any position of one I.A.S. word can be transferred without mutilation.

Function 40

2.2.2

Effect Transfers zeros to all positions of a specified word of I.A.S.

Operation The contents of the specified location in I.A.S. are overwritten with zeros. As zeros are formed in Register A and then transferred to I.A.S., the contents of Register A will also be zero.



Example Word 42 of block 11 is to be zeroized.

Instruction

D	F	A	R
	40	0042	11

Before

I.A.S.	0	0	0	0	0	0	5	8	9	4	3	0
Register A	0	0	0	0	0	0	6	5	8	7	9	2

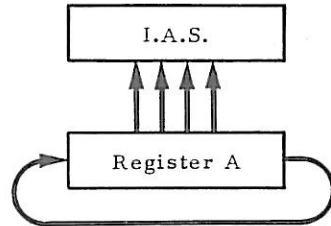
After

I.A.S.	0	0	0	0	0	0	0	0	0	0	0	0
Register A	0	0	0	0	0	0	0	0	0	0	0	0

Notes Zeroizing of words of I.A.S. will be necessary only in certain instances, such as where I.A.S. has been holding group accumulated totals and needs to be zeroized after each group.

Effect Transfers the contents of Register A to a specified word of I.A.S.

Operation The contents of Register A overwrite the previous contents in the specified location of I.A.S. The contents of Register A remain unaltered.



Example If it is necessary to transfer the contents of an I.A.S. location to some other location of I.A.S. while adding it to Register B, then a 41 order may be used as shown.

Program

	I	D	F	A	R
X					
			37	0049	17
X+1			62	0025	17
			41	0019	23
X+2					

	Before		After second instruction in word X																								
Word 49 Block 17	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>6</td><td>9</td><td>4</td><td>3</td><td>2</td></tr></table>	0	0	0	0	0	0	0	6	9	4	3	2	Word 49 Block 17	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>6</td><td>9</td><td>4</td><td>3</td><td>2</td></tr></table>	0	0	0	0	0	0	0	6	9	4	3	2
0	0	0	0	0	0	0	6	9	4	3	2																
0	0	0	0	0	0	0	6	9	4	3	2																
Register A	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>7</td><td>1</td><td>5</td><td>6</td><td>8</td></tr></table>	0	0	0	0	0	0	0	7	1	5	6	8	Register A	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>6</td><td>9</td><td>4</td><td>3</td><td>2</td></tr></table>	0	0	0	0	0	0	0	6	9	4	3	2
0	0	0	0	0	0	0	7	1	5	6	8																
0	0	0	0	0	0	0	6	9	4	3	2																
Register B	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>8</td><td>4</td><td>3</td><td>8</td><td>0</td></tr></table>	0	0	0	0	0	0	0	8	4	3	8	0	Register B	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>6</td><td>9</td><td>4</td><td>3</td><td>2</td></tr></table>	0	0	0	0	0	0	0	6	9	4	3	2
0	0	0	0	0	0	0	8	4	3	8	0																
0	0	0	0	0	0	0	6	9	4	3	2																

	After instructions in word X+1											
Word 25 Block 17	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>5</td><td>9</td><td>4</td></tr></table>	0	0	0	0	0	0	0	0	5	9	4
0	0	0	0	0	0	0	0	5	9	4		
Word 19 Block 23	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>5</td><td>9</td><td>4</td></tr></table>	0	0	0	0	0	0	0	0	5	9	4
0	0	0	0	0	0	0	0	5	9	4		
Register A	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>5</td><td>9</td><td>4</td></tr></table>	0	0	0	0	0	0	0	0	5	9	4
0	0	0	0	0	0	0	0	5	9	4		
Register B	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>7</td><td>0</td><td>0</td><td>2</td><td>6</td></tr></table>	0	0	0	0	0	0	7	0	0	2	6
0	0	0	0	0	0	7	0	0	2	6		

In the section of program above, word 49 of block 17 is transferred to Register B. Word 25 of block 17 is added to the contents of Register B and is also stored in word 19 of block 23. The 41 instruction effects the transfer of the contents of Register A to word 19 of block 23 thereby saving a special transfer instruction to Register B.

It must be noted that this will work only if the 41 instruction is the least-significant part of the instruction word. The contents of Register A would have been destroyed if control had been passed to the next word.

Notes The most common use of function 41 is to store the return jump instruction from a sub-routine. This is discussed fully in Part 3.

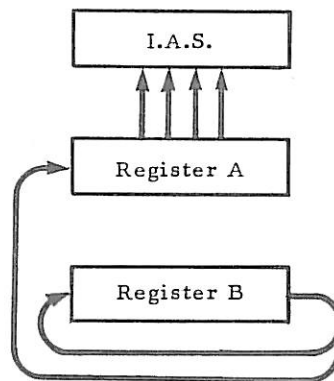
On completion of an instruction the contents of Register A are often useful and the use of function 41 should be kept in mind during programming.

Function 42

2.2.4

Effect Transfers the contents of Register B to a specified word of I.A.S.

Operation The contents of Register B are placed in Register A and thence overwrite the previous contents of the specified location in I.A.S. The contents of Register B are preserved.



Example It may be necessary to transfer Register B to I.A.S. temporarily while performing calculations on other factors, calling upon the original contents of Register B later. To perform the calculation $(a + b) \times (c + d) = e$, firstly $(a + b)$ will be calculated and the result temporarily stored using the 42 instruction. Secondly $(c + d)$ is calculated, and finally by bringing $(a + b)$ out of I.A.S. the multiplication can be performed.

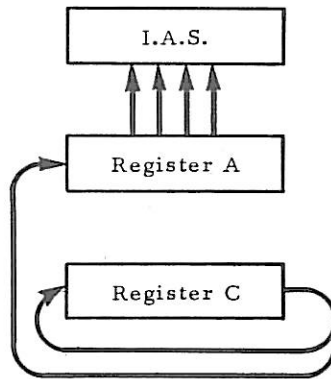
I	D	F	A	R	NARRATIVE
0		37	0012	13	a
		62	0014	13	(a+b)
1		42	0017	12	Temp Store (a+b)
		37	0013	13	c
2		62	0015	13	(c+d)
		69	0017	12	(a+b) x (c+d)
3		42	0016	13	e

Function 43

2.2.5

Effect Transfers the contents of Register C to a specified word of I.A.S.

Operation The contents of Register C are placed in Register A and thence overwrite the previous contents of the specified location in I.A.S. The contents of Register C are preserved.



Example Transfer the contents of Register C to word 42 of block 11.

Instruction

D	F	A	R
	43	0042	11

Before

I.A.S.	0	0	0	0	0	0	7	9	8	1	2
Register A	0	0	0	0	0	0	0	8	2	5	1
Register C	0	0	0	0	0	0	5	3	7	8	4

After

I.A.S.	0	0	0	0	0	0	5	3	7	8	4
Register A	0	0	0	0	0	0	5	3	7	8	4
Register C	0	0	0	0	0	0	5	3	7	8	4

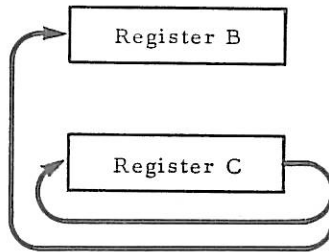
Notes The only occasion, other than on input, that Register C is used is when performing multiplication. The product is left in both Registers B and C at the end of a multiply function; it is possible to use function 43 as an alternative to function 42 when storing the product in I.A.S.

Function 44

2.2.6

Effect Transfers the contents of Register C to Register B.

Operation The contents of Register C are placed in, and overwrite, the contents of Register B. The contents of Register C are preserved.



Example

Instruction

D	F	A	R
	44	0000	00

Before

Register B	0	0	0	0	0	0	5	1	4	6	2
Register C	0	0	0	0	0	0	2	5	9	3	

After

Register B	0	0	0	0	0	0	0	2	5	9	3
Register C	0	0	0	0	0	0	0	2	5	9	3

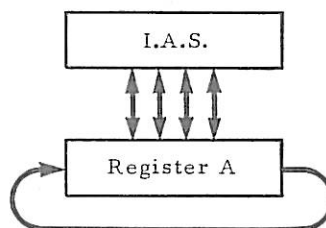
Note Register A is not used during this function.

Function 45

2.2.7

Effect Transfers the contents of one block of consecutively numbered locations in I.A.S. to another such block in I.A.S.

Operation A specified block of up to 20 words is transferred via Register A to overwrite another specified block of consecutive I.A.S. locations. The last word in the block transferred will be preserved in Register A but will always be overwritten during the change of control to the next word. Both I.A.S. blocks contain the same data when the function is completed (see Notes).



Instructions using function 45 are double-length and take the form:-

I	D	F	A	R
X	0	45	abcd	R ₁
	0	xy	pqrs	R ₂

where $abcd/R_1$ is the lowest numbered source address, $pqrs/R_2$ is the lowest numbered destination address and xy is the number of words in the block to be transferred, which must lie in the range 1 to 20.

Example

The instruction:

I	D	F	A	R
X	0	45	0010	12
	0	15	0100	16

transfers 15 words from addresses 0010, 0011, 0012..... 0024 of block 12 to addresses 0100, 0101, 0102 0114 of block 16.

Notes If the two relativizer reference numbers R_1 and R_2 are the same, they should be written in both places.

The previous contents of Register A are destroyed by this instruction.

If the two areas of storage between which the transfer is made do not overlap, then at the conclusion of the instruction both areas contain the information originally held in the source area. If the areas overlap, incorrect transfers can be obtained and to ensure correct transfers, they must be made from higher-numbered locations to lower-numbered ones, bearing in mind that absolute, not relative, addresses must be considered.

Example

INCORRECT TRANSFER

Order in which transfer is made	Source absolute address	Contents of the source address	Destination absolute address	Contents of destination address after transfer
1	0	115	1	115
2	1	125	2	115
3	2	135	3	115
4	3	145	4	115

This is obviously not a correct block transfer, although this property could be useful in zeroizing an area of I.A.S.

CORRECT TRANSFER

Order in which transfer is made	Source absolute address	Contents of the source address	Destination absolute address	Contents of destination address after transfer
1	101	321	100	321
2	102	16	101	16
3	103	143	102	143
4	104	92	103	92

This example demonstrates that the correct block transfer is achieved.

DECIMAL ADDITION AND SUBTRACTION

2.3

The decimal addition and subtraction instructions and the equivalent instructions described later under sterling addition and subtraction (Section 2.4) cause data to be processed in the Mill and therefore affect the Mill Indicators. Figure 4 is a summarized form of the schematic diagrams shown for each instruction under heading, Operation.

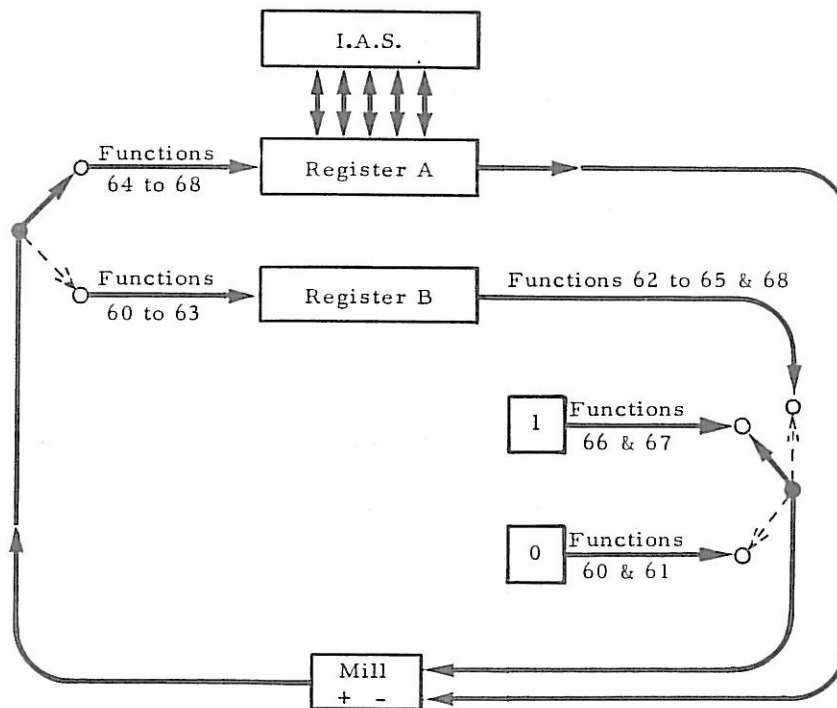


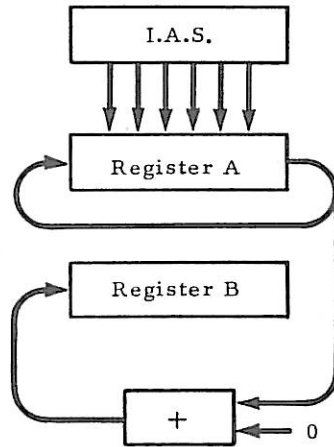
Figure 4: THE MILL INDICATORS AND FUNCTIONS 60 TO 68

Function 60

2.3.1

Effect This function CLEAR ADDS the contents of a specified location of I.A.S. into Register B.

Operation The specified I.A.S. word is put in Register A, added to zero in the Mill and the result placed in Register B. At the conclusion of the instruction the contents of Register A will be the original I.A.S. word. The contents of the I.A.S. location are unaltered.



Example To convert pence held in one position of I.A.S. to pence in two positions, a digit can be mutilated to some purpose by the 60 function. Assume that position 12 of word 19 of block 25 holds 11d. After the instruction

D	F	A	R
	37	0019	25

Register B contains

0	0	0	0	0	0	0	0	0	0	0	0	11
---	---	---	---	---	---	---	---	---	---	---	---	----

If however the instruction

D	F	A	R
	60	0019	25

is used, then

Before

I.A.S.	0	0	0	0	0	0	0	0	0	0	0	11
Register A	0	0	0	0	0	0	0	2	4	5	9	
Register B	0	0	0	0	0	0	5	6	4	9	6	

After

I.A.S.	0	0	0	0	0	0	0	0	0	0	0	11
Register A	0	0	0	0	0	0	0	0	0	0	0	11
Register B	0	0	0	0	0	0	0	0	0	1	1	

Notes As shown the function is similar to function 37, the essential difference being that function 60 uses the Mill and therefore affects the Mill and Overflow Indicators. Because it uses the Mill the instruction mutilates any digit greater than 9 in any one digit position by subtracting 10 from the digit and carrying 1 to the next most-significant digit.

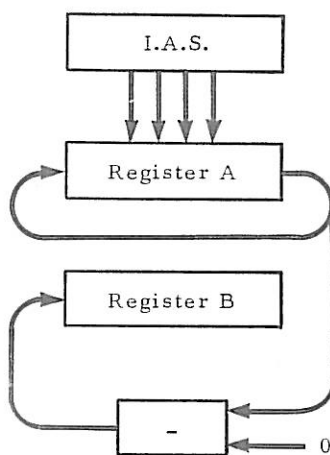
It is normally good practice to employ a 37 function when a transfer to Register B is required unless the special properties of a 60 function are needed.

Function 61

2.3.2

Effect This function CLEAR SUBTRACTS the contents of a specified location of I.A.S. into Register B.

Operation The specified I.A.S. word is put in Register A, subtracted from zero in the Mill and the result placed in Register B. At the conclusion of the instruction the contents of Register A will be the original I.A.S. word. The contents of the I.A.S. location are unaltered.



Example Clear subtract word 19 block 25.

Instruction

D	F	A	R
--	61	0019	25

Before

I.A.S.	0	0	0	0	0	0	0	0	0	8	2	4
Register A	0	0	0	0	0	0	6	4	8	9	2	
Register B	0	0	0	0	0	0	4	7	8	3	1	

After

I.A.S.	0	0	0	0	0	0	0	0	0	8	2	4
Register A	0	0	0	0	0	0	0	0	0	8	2	4
Register B	9	9	9	9	9	9	9	9	1	7	6	

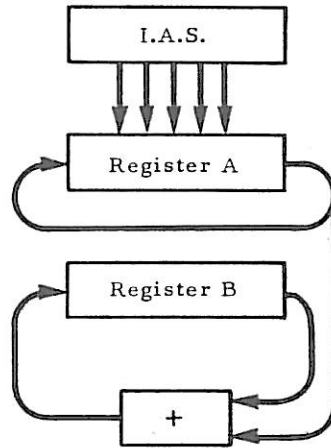
The result in Register B is the tens complement of 824.

Function 62

2.3.3

Effect ADDS a specified word of I.A.S. to the contents of Register B and places the result in Register B.

Operation The specified I.A.S. word is put in Register A, added in the Mill to the contents of Register B and the result placed in Register B. At the conclusion of the instruction the contents of Register A will be the original I.A.S. word. The I.A.S. word is unaltered.



Example Add 824 recorded as word 19 block 25 to 7453 contained in Register B.

Instruction

D	F	A	R
	62	0019	25

Before

I.A.S.	0	0	0	0	0	0	0	0	8	2	4
Register A	0	0	0	0	1	2	4	3	6	9	2
Register B	0	0	0	0	0	0	7	4	5	3	

After

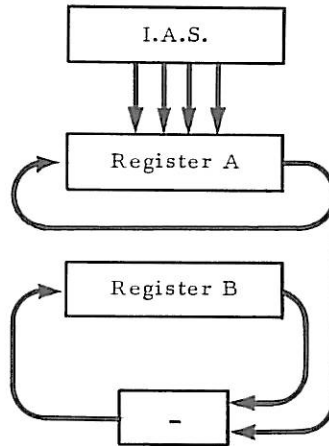
I.A.S.	0	0	0	0	0	0	0	0	8	2	4
Register A	0	0	0	0	0	0	0	0	8	2	4
Register B	0	0	0	0	0	0	8	2	7	7	

Function 63

2.3.4

Effect SUBTRACTS a specified word of I.A.S. from the contents of Register B and places the result in Register B.

Operation The specified I.A.S. word is put in Register A, subtracted in the Mill from the contents of Register B and the difference placed in Register B. At the conclusion of the instruction the contents of Register A will be the original I.A.S. word. The I.A.S. word is unaltered.



Example Subtract 421791 recorded as word 17 block 19 from 1924298 contained in Register B.

Instruction	D	F	A	R
		63	0017	19

Before

I.A.S.	0	0	0	0	0	0	4	2	1	7	9	1
Register A	0	0	0	0	0	0	0	0	0	0	0	0
Register B	0	0	0	0	0	1	9	2	4	2	9	8

After

I.A.S.	0	0	0	0	0	0	4	2	1	7	9	1
Register A	0	0	0	0	0	0	4	2	1	7	9	1
Register B	0	0	0	0	0	1	5	0	2	5	0	7

Alternatively if 1924298 is recorded as word 17 block 19 and is subtracted from 421791 contained in Register B the result is the complement or minus 1502507.

Before

I.A.S.	0	0	0	0	0	1	9	2	4	2	9	8
Register A	0	0	0	0	0	0	0	0	0	0	0	0
Register B	0	0	0	0	0	4	2	1	7	9	1	

After

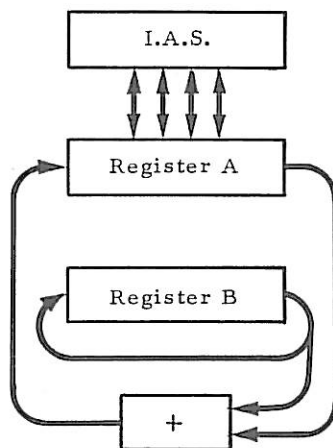
I.A.S.	0	0	0	0	0	1	9	2	4	2	9	8
Register A	0	0	0	0	0	1	9	2	4	2	9	8
Register B	9	9	9	9	8	4	9	7	4	9	3	

Function 64

2.3.5

Effect This function ADDS the contents of Register B to the contents of a specified location of I.A.S. and places the result in that location.

Operation The specified I.A.S. word is put in Register A, added in the Mill to the contents of Register B and the sum placed in the specified I.A.S. location by way of Register A. Thus on completion of this function Register A will also contain the result, which may be useful in certain cases. The original contents of Register B remain unaltered.



Example Add 6543 in Register B to 58216 recorded as word 17 block 19.

Instruction

D	F	A	R
	64	0017	19

Before

I.A.S.	0	0	0	0	0	0	0	5	8	2	1	6
Register A	0	0	0	0	0	0	0	0	0	0	0	0
Register B	0	0	0	0	0	0	0	6	5	4	3	

After

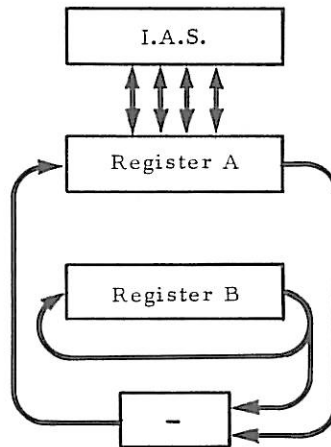
I.A.S.	0	0	0	0	0	0	0	6	4	7	5	9
Register A	0	0	0	0	0	0	0	6	4	7	5	9
Register B	0	0	0	0	0	0	0	6	5	4	3	

Function 65

2.3.6

Effect This function SUBTRACTS the contents of Register B from the contents of a specified location of I.A.S. and places the result in that location.

Operation The specified I.A.S. word is put in Register A, subtracted in the Mill from the contents of Register B and the difference placed in the specified I.A.S. location by way of Register A. Thus on completion of this function Register A will also contain the difference. The original contents of Register B remain unaltered.



Example Subtract 6543 in Register B from 58216 recorded as word 17 block 19.

Instruction

D	F	A	R
	65	0017	19

Before

I.A.S.	0	0	0	0	0	0	5	8	2	1	6
Register A	0	0	0	0	0	0	0	0	0	0	0
Register B	0	0	0	0	0	0	0	6	5	4	3

After

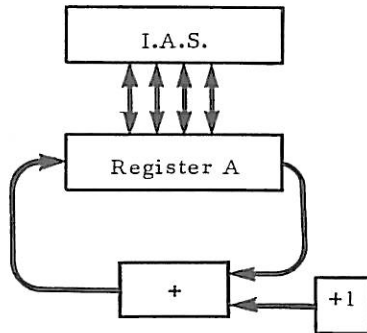
I.A.S.	0	0	0	0	0	0	5	1	6	7	3
Register A	0	0	0	0	0	0	5	1	6	7	3
Register B	0	0	0	0	0	0	0	6	5	4	3

Function 66

2.3.7

Effect This function ADDS 1 (one) to the least-significant position of the I.A.S. word specified in the instruction.

Operation The specified I.A.S. word is put in Register A, 1 is added in the Mill to the contents of Register A and the sum placed in the specified I.A.S. location by way of Register A. Thus on completion of this function Register A will also contain the sum. The contents of Register B are not affected by this instruction.



Example Add 1 to word 12 of block 24

Instruction

D	F	A	R
	66	0012	24
---	---	---	---
---	---	---	---

Before

I.A.S.	0	0	0	0	0	0	4	1	3	1	2	9
Register A	0	0	0	0	0	0	0	0	0	0	0	0
Register B	0	0	0	0	0	0	0	0	0	0	0	0

After

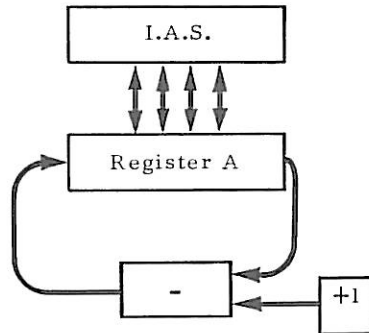
I.A.S.	0	0	0	0	0	0	4	1	3	1	3	0
Register A	0	0	0	0	0	0	4	1	3	1	3	0
Register B	0	0	0	0	0	0	0	0	0	0	0	0

Function 67

2.3.8

Effect This function SUBTRACTS 1 (one) from the least-significant position of the I.A.S. word specified in the instruction.

Operation The specified I.A.S. word is put in Register A, 1 is subtracted in the Mill from the contents of Register A and the difference placed in the specified I.A.S. location by way of Register A. Thus on completion of this function Register A will also contain the difference. The contents of Register B are not affected by this instruction.



Example Subtract 1 from word 17 of block 22.

Instruction

D	F	A	R
	67	0017	22

Before

I.A.S.	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Register A	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Register B	0	0	0	0	0	0	0	0	5	6	4	2		

After

I.A.S.	9	9	9	9	9	9	9	9	9	9	9	9	9	9
Register A	9	9	9	9	9	9	9	9	9	9	9	9	9	9
Register B	0	0	0	0	0	0	0	0	5	6	4	2		

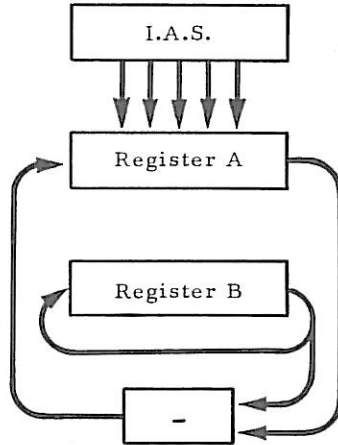
The result in word 17 block 22 is the equivalent of -1 as is also the content in Register A.

Function **68**

2.3.9

Effect COMPARES by subtracting the contents of Register B from the contents of the specified I.A.S. location, the result being placed in Register A.

Operation The contents of the specified location are placed in Register A, the contents of Register B are subtracted in the Mill from the contents of Register A, and the difference placed in Register A. The original contents of both Register B and the I.A.S. location are unaffected.



Example Compare the contents of Register B with word 17 of block 22.

Instruction	D	F	A	R
		68	0017	22

Before

I.A.S.	0	0	0	0	0	0	0	5	9	4	3	1
Register A	0	0	0	0	0	0	0	0	0	0	0	0
Register B	0	0	0	0	0	0	0	5	8	7	2	1

After

I.A.S.	0	0	0	0	0	0	0	5	9	4	3	1
Register A	0	0	0	0	0	0	0	0	7	1	0	
Register B	0	0	0	0	0	0	0	5	8	7	2	1

STERLING ADDITION AND SUBTRACTION

2.4

Special instructions enable sterling arithmetic to be performed. A Sterling Position Register is also provided which according to its setting ensures that sterling arithmetic is correctly carried out whatever the positions in a word of the pounds, shillings and pence columns. Thus the programmer can make best use of the 12-digit word, be it for several decimal places of pence or alternatively, when working with large values, as many places for pounds as possible.

When sterling arithmetic is performed the computer assumes that the sterling positions conform to the setting of the Sterling Position Register. It is therefore essential that, before an arithmetic instruction is given, the 10/- position in the operands concerned is as indicated by the Sterling Position Register. If the numbers are not correctly positioned they will be mutilated due to sterling carries being performed on the wrong digits.

The Sterling Position Register indicates the position of the tens of shillings (10/-) digit. Two digit positions, tens of shillings (10/-) and units of shillings (1/-) are allowed for shillings and one digit position for pence (excluding decimal positions). 10d and 11d are held in one digit position. The Sterling Position Register must be set before any sterling arithmetic is carried out and remains set until it is subsequently reset to another value.

Function 22

2.4.1

Effect Sets the Sterling Position Register according to the two least-significant digits of the address in the instruction.

Operation The position of the tens of shillings (10/-) digit is determined by using an address in the range 0002 to 0013. This determined position will remain set for all subsequent sterling arithmetic until reset by a further function 22.

Example To specify the pence in position 11 set the Sterling Position Register to 9, indicating the position of the tens of shillings (10/-) digit. Thus the instruction

D	F	A	R
--	22	0009	-

causes

0	0	0	2	9	4	6	2	1	9	10	0
---	---	---	---	---	---	---	---	---	---	----	---

to be operated on as

£29462..19..10.0d

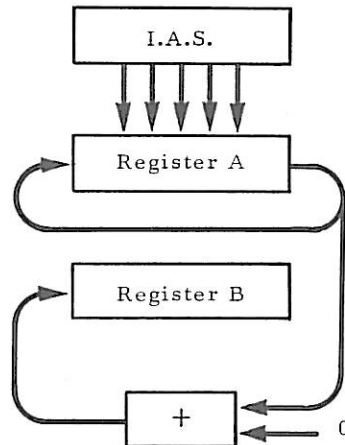
Notes It is *not permissible* to allocate position 1 for the tens of shillings (10/-) position. It is permissible to set the Sterling Position Register to 13 in which case sterling functions will be performed as if they were decimal functions. This is useful where a section of program may sometimes be using sterling factors and sometimes decimal factors, the digit entered by function 22 in the Sterling Position Register enabling the program to operate in either sterling or decimal.

Function 70

2.4.2

Effect CLEAR ADDS the sterling contents of a specified location of I.A.S. into Register B.

Operation The sterling contents of the specified I.A.S. location are put in Register A, added to zero in the Mill and the result placed in Register B. At the conclusion of the instruction the contents of Register A will be the original I.A.S. word. The I.A.S. word is unaltered.



Example Clear add £25..12..10d recorded as word 19 block 25 in Register B allowing two decimal places of pence. The Sterling Position Register is set to 8.

Instruction

D	F	A	R
	70	0019	25

Before

I.A.S.	0	0	0	0	0	2	5	1	2	10	0	0
Register A	0	0	0	0	0	0	0	0	0	0	0	0
Register B	0	0	0	0	0	0	5	0	8	11	3	5

After

I.A.S.	0	0	0	0	0	2	5	1	2	10	0	0
Register A	0	0	0	0	0	2	5	1	2	10	0	0
Register B	0	0	0	0	0	2	5	1	2	10	0	0

Notes Any decimal number being processed by this function will be mutilated during its progress through the Mill as illustrated on the next page.

Example Clear Add 45678 recorded as word 19 block 25 into Register B. The Sterling Position Register is set to position 10.

Instruction

D	F	A	R
	70	0019	25

Before

I.A.S.	0	0	0	0	0	0	0	4	5	6	7	8
Register A	0	0	0	0	0	0	0	0	0	0	0	0
Register B	0	0	0	0	0	0	0	5	9	2	3	

After

I.A.S.	0	0	0	0	0	0	0	4	5	6	7	8
Register A	0	0	0	0	0	0	0	4	5	6	7	8
Register B	0	0	0	0	0	0	4	6	4	7	8	

A 37 instruction is usually preferable to a 70 instruction, unless it is required to set the Mill and overflow indicators.

Functions 70 to 78

2.4.3

Functions 70 to 78 operate in sterling in the same manner that functions 60 to 68 do in decimal. Apart from function 70 just described only the function and brief operation details of each sterling function are given.

Function 71

2.4.4

Effect CLEAR SUBTRACTS the sterling contents of a specified location of I.A.S. into Register B.

Operation The sterling contents of the specified I.A.S. location are put in Register A, subtracted from zero in the Mill and the result placed in Register B. At the conclusion of the instruction the contents of Register A will be the original I.A.S. word. The I.A.S. word is unaltered.

Function 72

2.4.5

Effect ADDS the sterling contents of a specified location of I.A.S. to the sterling contents of Register B and places the result in Register B.

Operation The sterling contents of the specified I.A.S. location are put in Register A, added in the Mill to the sterling contents of Register B and the result placed in Register B. At the conclusion of the instruction the contents of Register A will be the original I.A.S. word. The I.A.S. word is unaltered.

Function 73

2.4.6

Effect SUBTRACTS the sterling contents of a specified location of I.A.S. from the sterling contents of Register B and places the result in Register B.

Operation The sterling contents of the specified I.A.S. location are put in Register A, subtracted in the Mill from the sterling content of Register B and the difference placed in Register B. At the conclusion of the instruction the contents of Register A will be the original I.A.S. word. The I.A.S. word is unaltered.

Function 74**2.4.7**

Effect This function ADDS the sterling contents of Register B to the sterling contents of a specified location of I.A.S. and places the result in that location.

Operation The sterling contents of the specified I.A.S. location are put in Register A, added in the Mill to the sterling contents of Register B and the sum placed in the specified I.A.S. location by way of Register A. Thus on completion of this function Register A will also contain the result, which may be useful in certain cases. The original contents of Register B remain unaltered.

Function 75**2.4.8**

Effect This function SUBTRACTS the sterling contents of Register B from the sterling contents of a specified location of I.A.S. and places the result in that location.

Operation The sterling contents of the specified I.A.S. location are put in Register A, subtracted in the Mill from the sterling contents of Register B and the difference placed in the specified I.A.S. location by way of Register A. Thus on completion of this function Register A will also contain the difference. The original contents of Register B remain unaltered.

Function 76**2.4.9**

Effect This function ADDS 1 (one) to the least-significant position of the sterling contents of the I.A.S. word specified in the instruction.

Operation The sterling contents of the specified I.A.S. location are put in Register A, 1 is added to the sterling contents of Register A in the Mill and the sum placed in the specified I.A.S. location by way of Register A. Thus on completion of this function Register A will also contain the sum. The contents of Register B are not affected by this instruction.

Function 77**2.4.10**

Effect This function SUBTRACTS 1 (one) from the least-significant position of the sterling contents of the I.A.S. word specified in the instruction.

Operation The sterling contents of the specified I.A.S. location are put in Register A, 1 is subtracted from the sterling contents of Register A in the Mill and the difference placed in the specified I.A.S. location by way of Register A. Thus on completion of this function Register A will also contain the difference. The contents of Register B are not affected by this instruction.

Function 78**2.4.11**

Effect COMPARES by subtracting the sterling contents of Register B from the sterling contents of a specified I.A.S. location, the result being placed in Register A.

Operation The sterling contents of the specified location are placed in Register A, the sterling contents of Register B are subtracted in the Mill from the sterling contents of Register A,

and the difference placed in Register A. The original contents of both Register B and the I.A.S. location are unaffected.

I	D	F	A	R	NARRATIVE
0		22	0008	-	10/- Position - Position 8
		70	0041	19	Clear Add I.A.S. 41 block 19 to Register B
1		72	0017	19	Add I.A.S. 17 block 19 to Register B
		74	0019	19	Add Register B to I.A.S. 19 block 19
2		22	0007	-	10/- Position - Position 7
		70	0049	16	Clear Add I.A.S. 49 block 16 to Register B
3		73	0046	16	Subtract I.A.S. 46 block 16 from Register B
		74	0051	16	Add Register B to I.A.S. 51 block 16
4		76	0051	16	Add 1 to I.A.S. 51 block 16

Figure 5: TYPICAL PORTION OF PROGRAM FOR STERLING CALCULATIONS

MULTIPLICATION AND THE DECIMAL POINT REGISTER

2.5

Multiplication is performed by a routine built into the computer and can be initiated by a single instruction. The sequence of operations requires the use of Registers A, B and C and the Mill to perform repeated additions and subtractions for each digit in the multiplier. The multiplier, which must be a decimal number, must be in Register B: the multiplicand, either a decimal or sterling number, must be in I.A.S. and the product will be placed in both Registers B and C. The logic for the multiplication routine is illustrated by the flowchart in Section.2.5.10.

Multiplication can be either

$$\begin{aligned} & \text{Decimal} \times \text{Decimal} = \text{Decimal (Function 69)} \\ \text{or} & \text{Sterling} \times \text{Decimal} = \text{Sterling (Function 79)}. \end{aligned}$$

Decimal places can be accommodated and multiplication of negative multipliers and multiplicands can be performed. Multipliers and multiplicands should not contain a digit other than 0 or 9 in the sign position (position 1) nor should they consist of a word of all zeros except for a nine in the most-significant position.

The digit positions of both multipliers and multiplicands should not contain values greater than 9 except for $\overline{10}$ or $\overline{11}$ held in the pence position of a sterling multiplicand.

Various examples are given after the functions for multiplication and setting of the Decimal Point Register have been described.

Decimal Multiplication

2.5.1

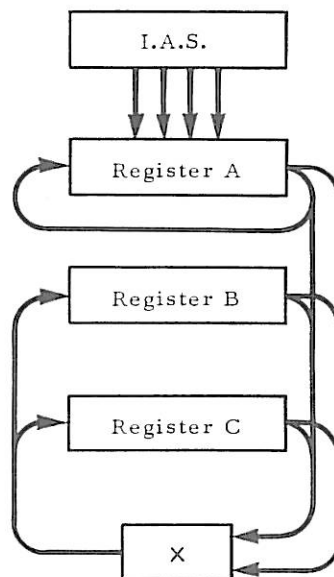
Although a 24-digit (double-length) product is theoretically produced within the computer, only twelve digits of the product are held at any one time, therefore twelve digits are available as the result, the selection of which twelve being determined by the setting of the Decimal Point Register (see 2.5.5).

Function 69

2.5.2

Effect The contents of a specified location of I.A.S. are multiplied by the contents of Register B. The twelve-digit product is placed in Registers B and C.

Operation The operation which is automatic is detailed in Section 2.5.10. The twelve-digit product formed in Register C is also placed in Register B. During multiplication and at the conclusion of the multiply instruction the specified word of I.A.S. (the multiplicand factor) remains unaltered. The previous contents of Register A are destroyed.



Notes This function causes data to be processed by the Mill and consequently, depending on the resultant product, the Mill Indicators and the Overflow Indicator may be set.

As the time taken to perform multiplication largely depends on the number of digits in the multiplier it is normally advantageous to position the smaller factor in Register B.

Sterling Multiplication

2.5.3

Before attempting to perform sterling multiplication ensure:

- that the sterling factor, the multiplicand, is in the specified location of I.A.S.,
- that the decimal factor, the multiplier, is in Register B,
- the Sterling Position Register is correctly set by using function 22.

Unless (a) and (b) are conformed to, the multiplication routine may enter a closed loop.

Effect The sterling contents of a specified location of I.A.S. are multiplied by the decimal contents of Register B. The twelve-digit product is placed in Registers B and C.

Operation The operation which is automatic is detailed in Section 2.5.10. The twelve-digit sterling product formed in Register C is also placed in Register B. During multiplication and at the conclusion of the multiply instruction the sterling contents of the specified location of I.A.S. (the multiplicand) remain unaltered. The previous contents of Register A are destroyed.

Notes As in decimal multiplication, data is processed by the Mill, thus the Mill Indicators and the Overflow Indicator are affected by the product.

The tens of shillings position of the resultant product will be the same as that of the multiplicand factor.

The Decimal Point Register

The Decimal Point Register is a one-digit subtraction counter which may be set by program to any value from 0 to 12. During multiplication, pulses are automatically fed to the counter and cause it to count down and feed a signal to control multiplication. Thus the setting of the Decimal Point Register is used for scaling during multiplication to produce the correct result.

As stated earlier, although the multiplication of two twelve-digit numbers results in a 24-digit product, only twelve digits can be formed in Register B. To determine which twelve of the 24 digits shall be retained divide the 24-digit product by 10^n where n is the number entered in the Decimal Point Register thus losing digits from the least-significant end. The twelve least-significant digits of the result are now taken as the final product.

When performing decimal arithmetic, n corresponds to the number of digits to be lost from the least-significant end of the 24-digit product. For example consider

$$\begin{aligned} &000000004321 \times 000000000005 \\ &= 00000000000000000021605 \end{aligned}$$

if the Decimal Point Register is set to 3 the product is divided by 10^3 (1,000) giving 000000000000000000021. As only the twelve least-significant digits remaining can be formed then the result registered is 000 000 000 021.

When decimal numbers are held in the computer any decimal points exist only in the mind of the programmer and are not physically recorded in the registers. If in the multiplication shown above, the factors comprised $.4321 \times .5$ the product would be $.21$ (605). The assumed decimal point after position 8 of the multiplicand is moved to after position 10 in the product. This could be a useful technique if the result is required to a specified number of decimal places.

If the Decimal Point Register is set to the number of decimal places in the multiplier word, then the decimal point is in the same position in the resulting product as it was in the multiplicand.

Consider again 000 000 004321 × 000000000005 representing .4321 × .5.

Set the Decimal Point Register to 1 as the number of decimal places in the multiplier is 1.

In the resulting product 000 000 002 160 there are effectively four decimal places with the decimal point after digit position 8 as it was in the multiplicand.

When sterling numbers are held in the computer, the positions of the sterling components are determined by the setting of the Sterling Position Register. In effect, this means that the decimal point is also fixed, being located after the pence position to denote decimals of pence.

When performing sterling multiplication, it is therefore apparent that the Decimal Point Register must be set to the number of decimal places in the multiplier to ensure that the decimal point is in the same position in the resulting product as it was in the multiplicand.

If £472.14.9 × 0.00429 is to result in £2.00.6 (to nearest penny below) with the Sterling Position Register set at 10 then the Decimal Point Register must be set to 5. Failure to set the Decimal Point Register will cause, in effect, £472.14.9 × 429 resulting in £202,804.07.9.

The rules for setting the Decimal Point Register may be summarized as follows:

Decimal × Decimal - Decimal Point Register setting = Number of digits to discard from least-significant end of product.

Sterling × Decimal - Decimal Point Register setting = Number of decimal places in multiplier word.

Function 21

2.5.6

Effect Sets the Decimal Point Register according to the two least-significant digits of the address in the instruction, which are in the range 00 to 12.

Operation The Decimal Point Register must be set by function 21 immediately prior to the multiply instruction, thus ensuring that during the automatic multiplication routine actuated by functions 69 and 79 the correct number of shifts take place. The Decimal Point Register is automatically set to zero when the multiplication has been completed.

If the Decimal Point Register is set and the multiply instruction does not occur until some time later, then the register will be reset to zero by the next instruction (see Example 2). An exception to this rule occurs if an instruction with designation 4, 8 or 9 or a function 00, 11, 22, 38, 39 and 80 to 87 is interposed between the 21 and 69 or 79 instructions.

Example 1 Set Decimal Point Register to 4.

Instruction	I	D	F	A	R
	x	--	21	0004	--

Example 2 Set the Decimal Point Register to 4 prior to performing decimal multiplication.

Correct Instructions					Incorrect Instructions				
I	D	F	A	R	I	D	F	A	R
x		37	0123	27	x		21	0004	-
		21	0004	-				37	0123
x+1		69	0124	27	x+1		69	0124	27

The incorrect set of instructions causes the Decimal Point Register to be reset to zero by the 37 instruction before multiplication takes place.

Note The programmer must ensure that the setting of the Decimal Point Register will not cause vital non-zero digits to be lost from the most-significant end.

Decimal Multiplication Examples

2.5.7

Example 1

Multiply the decimal contents of I.A.S. 14 block 23 by the decimal contents of
 I.A.S. 15 block 23 and store the result in
 I.A.S. 16 block 23

Instructions

I	D	F	A	R	NARRATIVE
x		37	0015	23	<i>Transfer I.A.S. to Register B</i>
		21	00??	-	
x+1		69	0014	23	<i>Multiply</i>
		42	0016	23	

Word 14 block 23 000, 000, 123, 456
 Word 15 block 23 × 000, 000, 999, 999
 = 000, 000, 000, 000, 123, 455, 876, 544

The digit in the Decimal Point Register determines the position and accuracy of the product in Registers B and C and consequently in I.A.S. 16 block 23.

Decimal Point Register	Product in Register B and Register C
0	123, 455, 876, 544 (sets Overflow Indicator)
1	012, 345, 587, 654 (4)
2	001, 234, 558, 765 (44)
3	000, 123, 455, 876 (544)
4	000, 012, 345, 587 (6544)
5	000, 001, 234, 558 (76544)
6	000, 000, 123, 455 (876544)
7	000, 000, 012, 345 (5876544)
8	000, 000, 001, 234 (55876544)
9	000, 000, 000, 123 (455876544)
10	000, 000, 000, 012 (3455876544)
11	000, 000, 000, 001 (23455876544)
12	000, 000, 000, 000 (123455876544)

The digits in brackets are those which are lost as a result of the content of the Decimal Point Register. There are no decimal places in the multiplier therefore a setting of 0 locates the decimal point in the same position as it is in the multiplicand i.e. after digit 12.

The original contents of I.A.S. 14 block 23 and I.A.S. 15 block 23 are unaltered at the completion of the instructions.

Example 2 Multiply the decimal contents of I.A.S. 92 block 41 by the decimal contents of I.A.S. 76 block 59 and store the result in I.A.S. 48 block 17.

Instructions

I	D	F	A	R	NARRATIVE
X		37	0076	59	Transfer I.A.S. to Register B
		21	00??	-	Set Decimal Point Register
X+1		69	0092	41	Multiply
		43	0048	17	Transfer (from Register C) to I.A.S.

Word 76 block 59 0.050, 000, 000, 00
Word 92 block 41 × 0.300, 000, 000, 00
 = 00.015, 000, 000, 000, 000, 000, 0

Decimal Point Register	Product in Register B and Register C
0 - 7	000, 000, 000, 000
8	500, 000, 000, 000
9	150, 000, 000, 000
10	015, 000, 000, 000
11	001, 500, 000, 000
12	000, 150, 000, 000

In this example any digit between 0 and 8 set in the Decimal Point Register will cause significant product digits to be lost. A setting of 11 locates the decimal point in the same position as it is in the multiplicand i.e. after position 1.

Example 3

$$\begin{array}{r}
 000, 000, 000, 123 \\
 \times 000, 000, 000, 005 \\
 = 000, 000, 000, 000, 000, 000, 000, 615
 \end{array}$$

Decimal Point Register	Product in Register B and Register C
0	000, 000, 000, 615
1	0, 000, 000, 000, 61
2	00, 000, 000, 000, 6
3 - 12	000, 000, 000, 000

Assuming that it is necessary to have the result with one decimal place, then 2 will be set in the Decimal Point Register.

Sterling Multiplication Example

2.5.8

Example Multiply the sterling contents of I.A.S. 19 block 24 by the decimal contents of I.A.S. 24 block 39 and store the sterling result in I.A.S. 92 block 17. The tens of shillings in the sterling factor is in position 6.

Instructions

I	D	F	A	R	NARRATIVE
X		37	0024	39	Transfer multiplier to Register B
		21	00??	-	Set Decimal Point Register
X+1		22	0006	-	Set Sterling Position Register
		79	0019	24	Multiply
X+2		42	0092	17	Store in I.A.S.

$$\begin{array}{r}
 05, 437.18.4.0000 \\
 \times 000, 000, 000, 008 \\
 = 00, 000, 000, 000, 043, 503.06.8.000, 0
 \end{array}$$

Depending on the digit entered in the Decimal Point Register before multiplication there would appear in Registers B and C, and consequently I.A.S. 92 block 17, the following:

Decimal Point Register	Product in Register B and Register C	
0	43503.06.8.0000	($\div 10^0$) (sets Overflow Indicator)
1	04350.06.8.0000	($\div 10^1$)
2	00435.00.8.0000	($\div 10^2$)
3	00043.10.0.8000	($\div 10^3$)
4	00004.07.0.0800	($\div 10^4$)
5	00000.08.8.4080	($\div 10^5$)
6	00000.00.10.4408	($\div 10^6$)
7	00000.00.1.0440 (8)	($\div 10^7$)
8	00000.00.0.1044 (08)	($\div 10^8$)
9	00000.00.0.0104 (408)	($\div 10^9$)
10	00000.00.0.0010 (4408)	($\div 10^{10}$)
11	00000.00.0.0001 (04408)	($\div 10^{11}$)
12	00000.00.0.0000 (104408)	($\div 10^{12}$)

If the multiplier had been 0.08 then a setting of 2 in the Decimal Point Register would have given the correct result.

The product (with 10/- in position 6) will be in both Registers B and C on completion of the multiply instruction and therefore can be stored in I.A.S. 92 block 17 with either a 42 or 43 instruction.

Negative Factors in Multiplication

2.5.9

Example Multiply the decimal contents of I.A.S. 196 block 37 by the negative decimal contents of I.A.S. 43 block 38 and store the negative result in I.A.S. 47 block 38.

Instructions	I	D	F	A	R
X			37	0043	38
			69	0196	37
X+1			42	0047	38

$$\begin{array}{r}
 000, 000, 000, 129 \\
 \times 999, 999, 924, 926 \quad \text{or effectively } \times (-75074) \\
 \hline
 = 999, 990, 315, 454 \quad \quad \quad = -9684546
 \end{array}$$

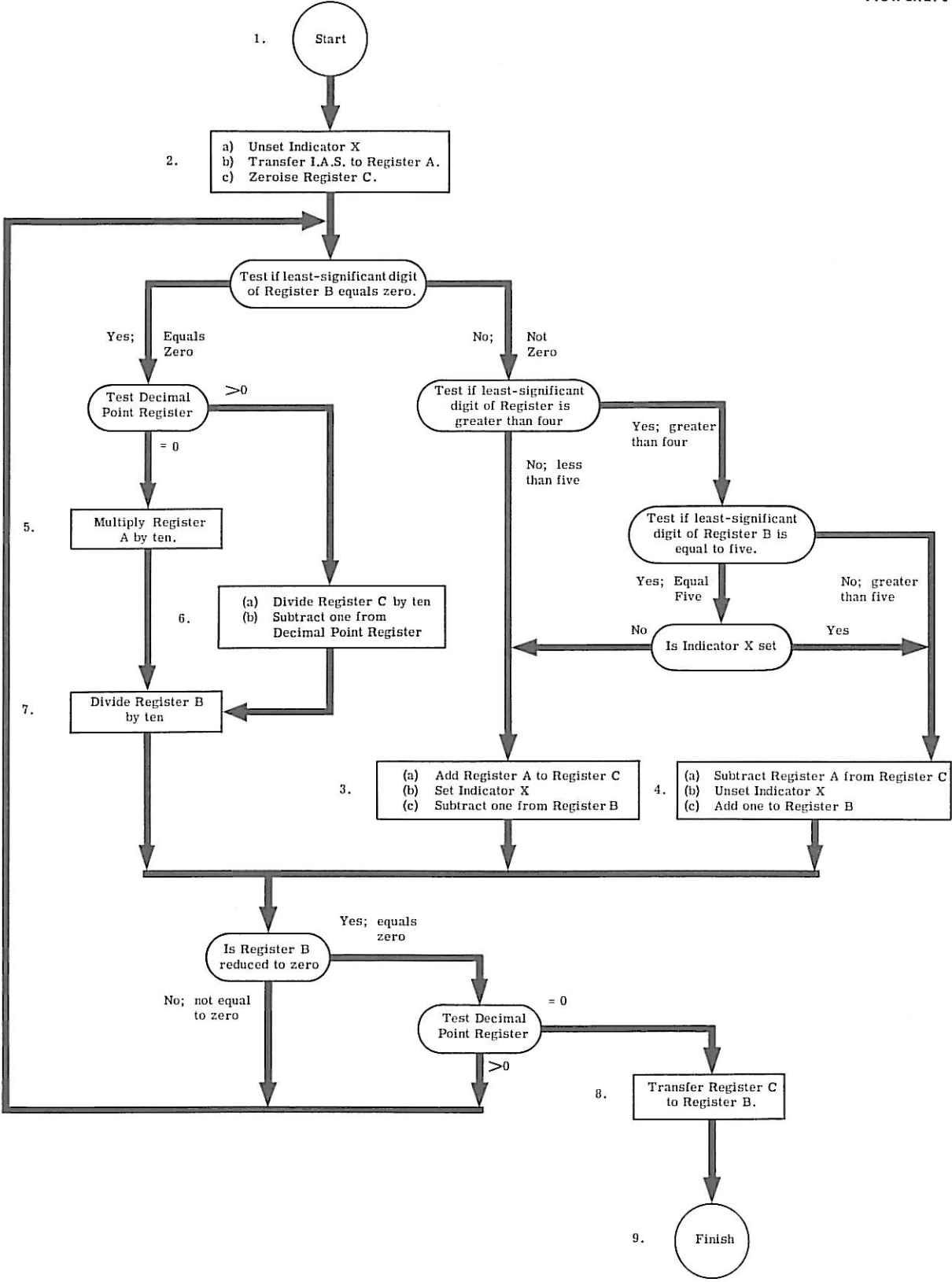
The Decimal Point Register is set to zero.

The reader may satisfy himself that similar negative multiplications can be performed when:

- (a) the content of the multiplier is negative, or
- (b) the content of the multiplicand is negative, either producing a negative result, or
- (c) both the contents of the multiplier and of the multiplicand are negative thereby producing a positive result, or
- (d) the above when the contents of the multiplicand are sterling.

Narrative

- 1 To commence the multiplicand is in I.A.S. and the multiplier is in Register B; this stage is achieved by the program.
- 2 This and following stages are all automatically achieved by the built-in routine.
 - (a) Unsetting indicator X.
 - (b) Multiplicand is put in Register A from the I.A.S.
 - (c) Put zeros in Register C which is used to accumulate the product.
- 3
 - (a) The content of Register A is added to that of Register C, the sum being placed in Register C.
 - (b) Indicator X is set.
 - (c) The content of Register B is reduced by one.
- 4
 - (a) The content of Register A is subtracted from Register C the difference being placed in Register C.
 - (b) Indicator X is unset.
 - (c) The content of Register B is increased by one.
- 5 The content of Register A is multiplied by ten.
- 6
 - (a) The content of Register C is divided by ten. Sign is propagated at the most-significant end.
 - (b) The Decimal Point Register is counted down by one.
- 7 The content of Register B is divided by ten. The sign is propagated at the most-significant end.
- 8 The accumulated product in Register C is transferred to Register B.
- 9 At the conclusion of the multiply function the multiplicand is still in I.A.S. The product is in Registers B and C. The original contents of Register A are mutilated.

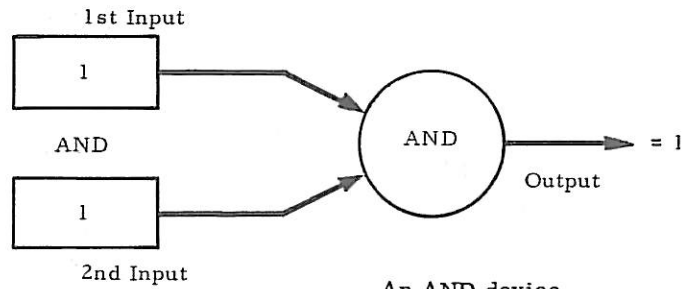


LOGICAL FUNCTIONS

2.6

The two functions 35 and 36 are termed Logical AND and Logical OR respectively. The logic of AND is:

- (a) $0 \text{ AND } 0 = 0$
- (b) $1 \text{ AND } 0 = 0$
- (c) $0 \text{ AND } 1 = 0$
- (d) $1 \text{ AND } 1 = 1$

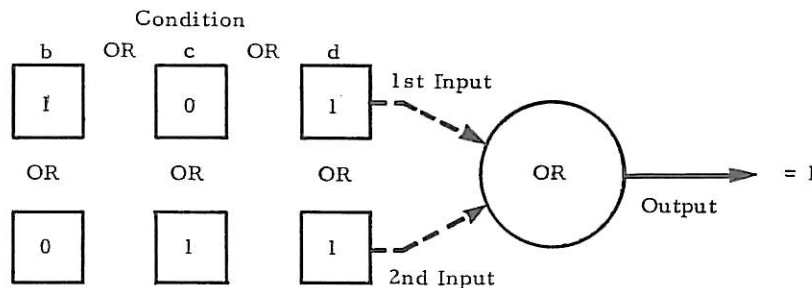


An AND device.

It is apparent that an AND device comprising two inputs must have a 1 at both inputs before a 1 will appear at its single outlet. The only time a 1 would be available at the output of the device illustrated above would be condition (d) $1 \text{ AND } 1 = 1$.

The logic of OR is:

- (a) $0 \text{ OR } 0 = 0$
- (b) $1 \text{ OR } 0 = 1$
- (c) $0 \text{ OR } 1 = 1$
- (d) $1 \text{ OR } 1 = 1$



An OR device.

As shown an OR device will have a 1 available at its output when a 1 is entered at either input or both inputs, i.e. conditions (b), (c) and (d) but not (a) cause a 1 to be available at the output.

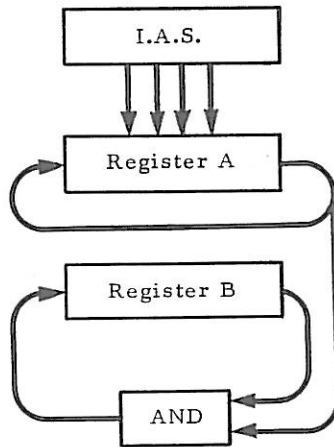
This concept can be applied to data in the computer by inspecting the binary coded representation or bits of the digits within a word. Thus those bits which match as shown in the logic above will be available in Register B at the conclusion of an instruction. Therefore logical functions are used when it is required to refer to parts of words rather than the complete word. A part may consist of several digits of a word or simply one or more bits which make up a single digit. When using logical functions it is important to remember the binary coded representation of the digits within a word.

Function 35

2.6.1

Effect Causes the operation of Logical AND to be carried out between the contents of Register B and the contents of a specified location of I.A.S. on a bit-for-bit basis.

Operation A number held in the specified I.A.S. location is put in Register A. Logical AND of the contents of Registers A and B then takes place in the Mill and the result is placed in Register B. At the conclusion of the instruction the contents of Register A will be the original I.A.S. word. The I.A.S. word is unaltered.



Examples If 2916359874 is contained in I.A.S. 15 block 12 and 4174322861 is contained in Register B, then at the completion of function 35 the registers and I.A.S. 15 block 12 will stand thus.

I.A.S.	0	0	2	9	1	6	3	5	9	8	7	4
Register A	0	0	2	9	1	6	3	5	9	8	7	4
Register B	0	0	0	1	1	4	3	0	0	8	6	0

A study of the binary representation will clarify this.

Word 15 Block 12

0	0	2	9	1	6	3	5	9	8	7	4
0	0	0	1	1	0	1	1	1	0	1	0
0	0	1	0	0	1	1	0	0	0	1	0
0	0	0	0	0	1	0	1	0	0	1	1
0	0	0	1	0	0	0	0	1	1	0	0

*particular location in IAS
As defined in address
of function*

Register B

0	0	4	1	7	4	3	2	2	8	6	1
0	0	0	1	1	0	1	0	0	0	0	1
0	0	0	0	1	0	1	1	1	0	1	0
0	0	1	0	1	1	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	1	0	0

Result in Register B

0	0	0	1	1	4	3	0	0	8	6	0
0	0	0	1	1	0	1	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	1	0	0

Three further examples of the use of this function are given although it has many varied one-off uses to which it may be put.

1 MASKING TO EXTRACT PART OF A WORD OF I.A.S.

Word 19 of block 14 contains unwanted data in positions 1 to 6, the data in position 7 to 12 is required in Register B.

Instruction

I	D	F	A	R
X	---	37	0019	14
		35	0021	16
X+1	---	---	---	---

data
- mask

Before

After

I.A.S. 21 Block 16

0	0	0	0	0	0	15	15	15	15	15	15
---	---	---	---	---	---	----	----	----	----	----	----

I.A.S. 21 Block 16

0	0	0	0	0	0	15	15	15	15	15	15
---	---	---	---	---	---	----	----	----	----	----	----

I.A.S. 19 Block 14

5	6	4	9	3	2	1	6	8	4	7	8
---	---	---	---	---	---	---	---	---	---	---	---

or

Register A

Register B

0	0	0	0	0	0	1	1	1	1	1	1	1
0	0	0	0	0	0	1	1	1	1	1	1	1
0	0	0	0	0	0	1	1	1	1	1	1	1
0	0	0	0	0	0	1	1	1	1	1	1	1

0	0	0	0	0	0	15	15	15	15	15	15
5	6	4	9	3	2	1	6	8	4	7	8
0	0	0	0	0	0	15	15	15	15	15	15
0	0	0	0	0	0	1	6	8	4	7	8

I.A.S. 19 Block 14

5	6	4	9	3	2	1	6	8	4	7	8
---	---	---	---	---	---	---	---	---	---	---	---

Only the matching binary digits for each position arrive in Register B.

2 MASKING TO EXTRACT AN ADDRESS

Mask constants can be used to extract the address from an instruction. An example of this use would be where the instruction tests an indicator, as in this case the address has 4000 added to it. The 4000 can be subsequently removed by using a mask as below

either

0	0	3	15	15	15	0	0	0	0	0	0
---	---	---	----	----	----	---	---	---	---	---	---

or

0	0	0	0	0	0	0	0	3	15	15	15
---	---	---	---	---	---	---	---	---	----	----	----

depending on whether the instruction to be masked is in the most-significant or least-significant half of the word.

3 ODD/EVEN NUMBER DISCRIMINATION

A mask can be used to determine whether a number is odd or even. Here word 17 block 19 is examined, the mask being word 14 block 13.

Instruction	I	D	F	A	R
X	---	---	60	0014	13
			35	0017	19
X+1	---	---	---	---	---

Before

After

I.A.S. 14 Block 13	0	0	0	0	0	0	0	0	0	0	1
I.A.S. 17 Block 19	0	0	0	0	0	6	5	2	3	8	3
Register A	0	0	0	0	0	0	0	0	0	0	0
Register B	0	0	0	0	0	0	8	9	2	3	4

I.A.S. 14 Block 13	0	0	0	0	0	0	0	0	0	0	1
I.A.S. 17 Block 19	0	0	0	0	0	6	5	2	3	8	3
Register A	0	0	0	0	0	6	5	2	3	8	3
Register B	0	0	0	0	0	0	0	0	0	0	1

At the conclusion of the 35 instruction, it will be possible to discriminate between the presence or absence of a 1-bit by means of Mill Indicators 01 and 02 (see page 54). In the example above indicator 02 is set, as Register B contains a 1-bit corresponding to the I.A.S. contents being odd. If the contents of Register B are zero corresponding to the I.A.S. contents being even, Mill Indicator 01 is set.

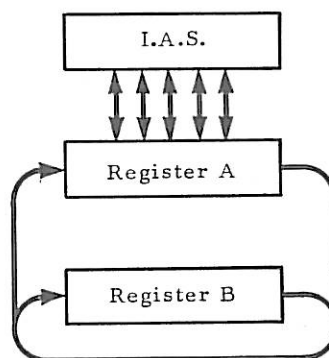
Notes During the execution of function 35 the overflow indicator is inhibited but the other Mill Indicators function normally.

Function 36

2.6.2

Effect Causes the operation of Logical OR to be carried out between the contents of Register B and the contents of a specified location of I.A.S. on a bit-for-bit basis.

Operation A number held in the specified I.A.S. location is put in Register A. Logical OR of the contents of Registers A and B then takes place and the result is placed in Registers A and B and in the original I.A.S. location specified in the instruction.



Examples If 123456123456 is contained in I.A.S. 15 block 12 and 999999666666 is contained in Register B, then at the completion of function 36 the Registers A, B and I.A.S. 15 block 12 will stand thus:

I.A.S.	9	11	11	13	13	15	7	6	7	6	7	6
Register A	9	11	11	13	13	15	7	6	7	6	7	6
Register B	9	11	11	13	13	15	7	6	7	6	7	6

A study of the 1, 2, 4 and 8 streams of the original contents of word 15 block 12 and Register B, together with the result obtained after Logical OR are shown below:-

Word 15 Block 12	1	2	3	4	5	6	1	2	3	4	5	6	
	1	0	1	0	1	0	1	0	1	0	1	0	1
	0	1	1	0	0	1	0	1	1	0	0	1	2
	0	0	0	1	1	1	0	0	0	1	1	1	4
	0	0	0	0	0	0	0	0	0	0	0	0	8

Register B	9	9	9	9	9	9	6	6	6	6	6	6	
	1	1	1	1	1	1	0	0	0	0	0	0	1
	0	0	0	0	0	0	1	1	1	1	1	1	2
	0	0	0	0	0	0	1	1	1	1	1	1	4
	1	1	1	1	1	1	0	0	0	0	0	0	8

Result in Word 15 Block 12 and Registers A and B	9	11	11	13	13	15	7	6	7	6	7	6	
	1	1	1	1	1	1	1	0	1	0	1	0	1
	0	1	1	0	0	1	1	1	1	1	1	1	2
	0	0	0	1	1	1	1	1	1	1	1	1	4
	1	1	1	1	1	1	0	0	0	0	0	0	8

The most common use for Logical OR is for modifying an instruction by the insertion of an address without using the Mill and in consequence not setting the overflow indicator. For example to modify the first instruction contained in word 12 of block 23 by inserting an address from word 29 of block 21.

Instructions	I	D	F	A	R
x			37	0029	21
			35	0011	23
x+1			36	0012	23

Before

I.A.S. 29 Block 21	0	0	1	2	4	8	0	0	0	4	1	9
I.A.S. 11 Block 23	0	0	3	15	15	15	0	0	0	0	0	0
I.A.S. 12 Block 23	6	0	0	0	0	0	6	3	1	1	9	2

Register A	0	0	0	0	0	0	0	0	0	0	0	0
Register B	0	0	0	0	0	0	0	0	0	0	0	0

After

I.A.S. 29 Block 21	0	0	1	2	4	8	0	0	0	4	1	9
I.A.S. 11 Block 23	0	0	3	15	15	15	0	0	0	0	0	0
I.A.S. 12 Block 23	6	0	1	2	4	8	6	3	1	1	9	2

Register A	6	0	1	2	4	8	6	3	1	1	9	2
Register B	6	0	1	2	4	8	6	3	1	1	9	2

The same result can be achieved without using Logical OR. Whereas there is little difference in the time, this second method affects the overflow indicator.

The instructions would be

I	D	F	A	R
X		37	0029	21
		35	0011	23
X+1		64	0012	23

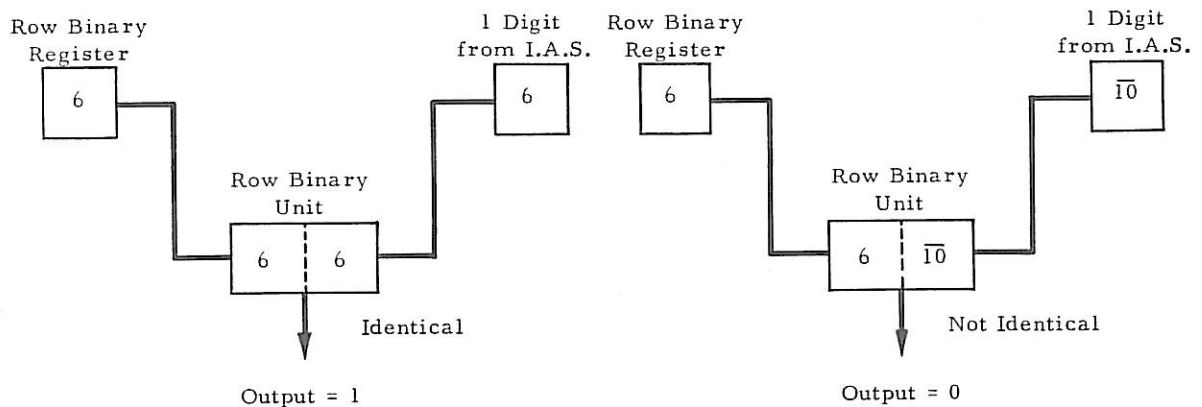
The uses of modifications are discussed in Part 4.

Notes As function 36 does not use the Mill, no Mill Indicators are set by this function.

ROW BINARIZING

2.7

Row-binary is normally created prior to print or punch output as described in Part 3. The components used to create row-binary are Register B, the specified word of I.A.S. and the Row Binary Register together with the Row Binary Unit. As illustrated schematically the Row Binary Register is a single-digit register capable of holding any number from 0 to 15, and the Row Binary Unit uses a technique similar to the Logical AND described in 2.6 but performing the logic on whole digits rather than bits. If a digit from a single position of an I.A.S. location is compared in the Row Binary Unit with the digit in the Row Binary Register, when the digits are identical a 1-bit will be emitted from the output of the device.

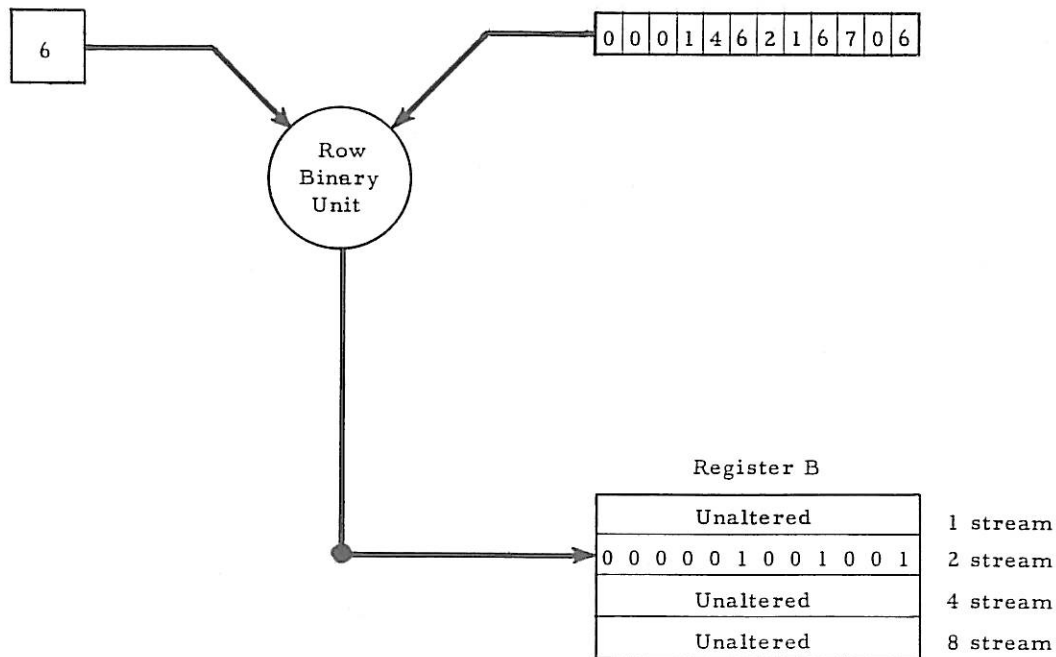


According to the function given, the 1s or 0s from the output of the Row Binary Register component can be directed to the appropriate stream of the B Register.

Thus in row binarizing each *digit* of the contents of the specified location of I.A.S. is compared against the digit set in the Row Binary Register. If the digits correspond then a 1-bit is emitted into the appropriate stream of Register B and if the digits fail to correspond a 0 is emitted into the stream of Register B e.g. the Row Binary Register is set to 6 and the instruction is given to create row-binary in stream 2 of Register B for I.A.S. 12 block 11 which contains 000046216706.

6 in the Row Binary Register

Word 12 Block 11



The Row Binary Register can be loaded in one of two ways:

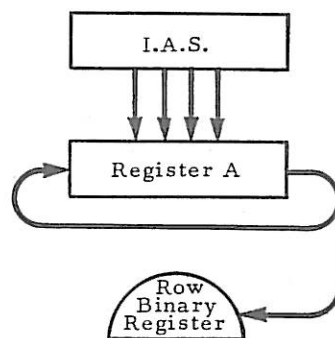
- (a) with the least-significant digit of the contents of a specified location of I.A.S. (see function 30 below).
- (b) with the contents of the index point (print) counter. This method is described in Part 3.

Function 30

2.7.1

Effect Causes the contents of the least-significant position of a specified location of I.A.S. to be transferred to the Row Binary Register.

Operation A number held in the I.A.S. location specified is put in Register A, the least-significant digit being placed in the Row Binary Register. The eleven most-significant figures do not affect the Row Binary Register. At the conclusion of the instruction the contents of Register A will be the original I.A.S. word. The I.A.S. word is unaltered.



Example Transfer 6 contained in the least-significant position of word 19 block 12 to the Row Binary Register.

Instruction

I	D	F	A	R
		30	0019	12

Before		After																									
I.A.S.	<table border="1"><tr><td>3</td><td>7</td><td>0</td><td>1</td><td>4</td><td>5</td><td>6</td><td>2</td><td>0</td><td>3</td><td>9</td><td>6</td></tr></table>	3	7	0	1	4	5	6	2	0	3	9	6	I.A.S.	<table border="1"><tr><td>3</td><td>7</td><td>0</td><td>1</td><td>4</td><td>5</td><td>6</td><td>2</td><td>0</td><td>3</td><td>9</td><td>6</td></tr></table>	3	7	0	1	4	5	6	2	0	3	9	6
3	7	0	1	4	5	6	2	0	3	9	6																
3	7	0	1	4	5	6	2	0	3	9	6																
Register A	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	0	0	0	0	0	0	0	0	0	0	0	0	Register A	<table border="1"><tr><td>3</td><td>7</td><td>0</td><td>1</td><td>4</td><td>5</td><td>6</td><td>2</td><td>0</td><td>3</td><td>9</td><td>6</td></tr></table>	3	7	0	1	4	5	6	2	0	3	9	6
0	0	0	0	0	0	0	0	0	0	0	0																
3	7	0	1	4	5	6	2	0	3	9	6																
Row Binary Register	8	Row Binary Register	6																								

Notes Sometimes a constant can be found in the program with a digit, in the least-significant position of the second address, suitable to load the Row Binary Register. If, however, none is available it is necessary to create a special constant.

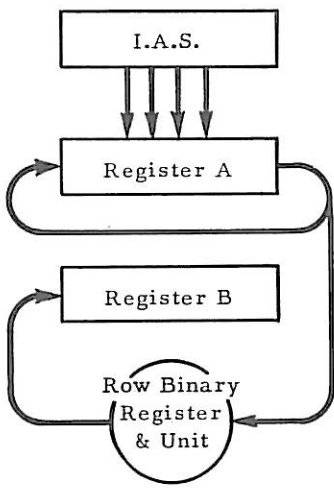
Care must be exercised while programming to ensure that the least-significant digit of the chosen constant is that of the *absolute* Address and not the Relative Address and also that the constant is in I.A.S. when function 30 is obeyed. It is safer to use either a constant or an instruction with no relativizer (e.g. those used to set the Decimal Point or Sterling Position Registers) as these will not change if the storage allocation is altered.

Function 31

2.7.2

Effect Creates row-binary into the 1 stream of Register B for the contents of a specified location of I.A.S.

Operation A number held in the specified I.A.S. location is put in Register A and each digit compared with the digit held in the Row Binary Register. When the digits correspond then a 1-bit is emitted into the 1 stream of Register B, when the digits do not correspond then a 0-bit is emitted into the 1 stream of Register B. At the conclusion of the instruction the contents of the 2, 4 and 8 streams of Register B and the original I.A.S. word are unaltered. Also the contents of Register A will be the original I.A.S. word.



Example 1 Create row-binary into the 1 stream of Register B for the contents of I.A.S. 42 block 15. The Row Binary Register contains a 6.

Instruction			
D	F	A	R
	31	0042	15

Before

After

I.A.S.	4	9	6	2	6	0	4	6	9	2	6	6
Register A	3	7	0	2	1	9	6	2	0	0	0	3

I.A.S.	4	9	6	2	6	0	4	6	9	2	6	6
Register A	4	9	6	2	6	0	4	6	9	2	6	6

Register B	0	1	0	0	1	1	0	0	0	1	0	0
	0	0	1	0	1	1	1	0	1	0	0	1
	0	1	1	0	0	0	0	1	0	1	1	0
	1	0	0	1	0	0	0	0	0	0	0	1
Row Binary Register												
	6											

Register B	0	0	1	0	1	0	0	1	0	0	1	1
	0	0	1	0	1	1	1	0	1	0	0	1
	0	1	1	0	0	0	0	1	0	1	1	0
	1	0	0	1	0	0	0	0	0	0	0	1
Row Binary Register												
	6											

Functions 30 to 34 will generally be used for output only, but row-binary techniques can occasionally be used as alternatives to conventional methods of programming in order to save space and/or time.

Example 2 Examine word 17 of block 15 for the presence of digit 3 in position 9.

Instruction					
	I	D	F	A	R
X			30	0041	19
			31	0017	15
X+1			35	0049	19

Before

After

I.A.S. 17 Block 15	0	0	0	3	8	3	2	4	3	9	2	6
I.A.S. 41 Block 19	0	0	0	0	8	3	2	6	2	4	0	3
I.A.S. 49 Block 19	0	0	0	0	0	0	0	0	1	0	0	0
Register A	0	0	0	0	0	0	0	0	0	0	0	0
Register B	0	0	0	0	0	0	0	0	0	0	0	0
Row Binary Register												
	0											

I.A.S. 17 Block 15	0	0	0	3	8	3	2	4	3	9	2	6
I.A.S. 41 Block 19	0	0	0	0	8	3	2	6	2	4	0	3
I.A.S. 49 Block 19	0	0	0	0	0	0	0	0	1	0	0	0
Register A	0	0	0	0	0	0	0	0	1	0	0	0
Register B	0	0	0	0	0	0	0	0	1	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0
Row Binary Register												
	3											

By examining Mill Indicators 01 and 02, as described under Section 2.6.1 Example 3 Odd/Even Discrimination, it is possible to jump to the section of the program relating only to digit 3 in position 9 of word 17 of block 15.

Example 3 Examine word 49 of block 22 for the presence of digit 6 in position 4, 7 or 11.

Instruction	I	D	F	A	R
X		--	30	0039	14
			31	0049	22
X+1		--	35	0040	14

Before

After

I.A.S. 39 Block 14	0	0	0	0	0	0	0	0	0	0	0	6
I.A.S. 40 Block 14	0	0	0	1	0	0	1	0	0	0	1	0
I.A.S. 49 Block 22	0	1	2	6	6	5	7	1	2	8	6	9

I.A.S. 39 Block 14	0	0	0	0	0	0	0	0	0	0	0	6
I.A.S. 40 Block 14	0	0	0	1	0	0	1	0	0	0	1	0
I.A.S. 49 Block 22	0	1	2	6	6	5	7	1	2	8	6	9

Register A	0	0	0	0	0	0	0	0	0	0	0	0
------------	---	---	---	---	---	---	---	---	---	---	---	---

Register A	0	0	0	1	0	0	1	0	0	0	1	0
------------	---	---	---	---	---	---	---	---	---	---	---	---

Register B	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	1	0	0	0	0	1	1	0	0	1
	0	0	0	0	0	1	0	0	1	0	1	1
	0	1	0	1	0	0	0	0	0	1	0	0
Row Binary Register												0

Register B	0	0	0	1	0	0	0	0	0	0	0	1	0
	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0
Row Binary Register												6	

By examining Mill Indicators 01 and 02, as described under Section 2.6.1 Example 3 Odd/Even Discrimination, it is possible to jump to the section of the program relating only to instances where a 6 appears in position 4, 7 or 11 of word 49 of block 22.

Note No Mill Indicators are set by function 31 as the Mill is not involved in the operation.

Function 32 **2.7.3**

Function 32 creates row-binary into the 2 stream of Register B for the contents of a specified word of I.A.S. in the manner described in 2.7.2.

Function 33 **2.7.4**

Function 33 creates row-binary into the 4 stream of Register B for the contents of a specified word of I.A.S. in the manner described in 2.7.2.

Function 34 **2.7.5**

Function 34 creates row-binary into the 8 stream of Register B for the contents of a specified word of I.A.S. in the manner described in 2.7.2.

Example of the combined use of Functions 30 to 35

2.7.6

Examine word 16 of block 21 for the presence of digit 3, 6, 7 or 8 in positions 11 or 12.

Instruction	I	D	F	A	R
x	--		30	0039	19
			31	0016	21
x+1	--		30	0040	19
			32	0016	21
x+2	--		30	0041	19
			33	0016	21
x+3	--		30	0042	19
			34	0016	21
x+4	--		35	0043	19

Before

After

I.A.S. 39 Block 19	8	4	0	4	2	7	9	1	5	8	6	3
I.A.S. 40 Block 19	5	7	9	1	3	6	2	8	0	5	4	6
I.A.S. 41 Block 19	3	7	7	7	5	1	9	6	1	2	4	7
I.A.S. 42 Block 19	3	2	4	8	4	6	4	2	7	1	1	8
I.A.S. 43 Block 19	0	0	0	0	0	0	0	0	0	15	15	
I.A.S. 16 Block 21	0	0	7	9	6	5	8	9	2	4	3	8

I.A.S. 39 Block 19	8	4	0	4	2	7	9	1	5	8	6	3
I.A.S. 40 Block 19	5	7	9	1	3	6	2	8	0	5	4	6
I.A.S. 41 Block 19	3	7	7	7	5	1	9	6	1	2	4	7
I.A.S. 42 Block 19	3	2	4	8	4	6	4	2	7	1	1	8
I.A.S. 43 Block 19	0	0	0	0	0	0	0	0	0	0	15	15
I.A.S. 16 Block 21	0	0	7	9	6	5	8	9	2	4	3	8

Register A

0	0	0	0	0	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---

Register A

0	0	0	0	0	0	0	0	0	0	15	15
---	---	---	---	---	---	---	---	---	---	----	----

Register B

0	0	0	0	0	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---

Register B

0	0	0	0	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	0	0	0	0	0	2
0	0	0	0	0	0	0	0	0	0	0	0	4
0	0	0	0	0	0	0	0	0	0	0	1	8

Row Binary Register

0

Row Binary Register

8

By testing Mill Indicators 01 and 02, as described under Section 2.6.1 Example 3 Odd/Even Discrimination, it is possible to jump to the section of the program relating only to instances where a 3, 6, 7 or 8 appears in positions 11 or 12 of word 16 of block 21.

SHIFT INSTRUCTIONS

2.8.

Data in Register B can be moved en bloc to right or left within the register. This technique known as shifting is used to move numbers beyond the capacity of Register B so that they are either eliminated or circulated, that is, they re-enter Register B at the opposite end.

Generally a simple problem of extraction of a part of a word can be solved by the shifting method, using less storage though taking slightly more time than the masking method using Logical AND (see 2.6.1). This is because the latter case requires storage for a special masking constant.

However, a more complicated problem of this type, for example the extraction of positions 1 to 3 and 6 to 9 of a word is more economically solved by the mask constant method both in time and storage required. The method to be preferred will depend on the relative necessity of economizing on time or storage in the rest of the program.

The number of positions shifted is determined by the number in the two least-significant positions of the address part of the instruction which may take any value from 00 to 12. As far as the shift instruction itself is concerned, it is immaterial whether the contents of Register B are in decimal or sterling form. However, when a sterling amount is shifted to a different position in Register B it is necessary to set the Sterling Position Register (see 2.4.1) to correspond to the new tens of shillings position before applying any of the sterling arithmetic functions (i.e. functions 70 to 79) to the contents of Register B.

The shift functions do not send the contents of Register B through the Mill, and therefore cannot set any of the Mill Indicators.

A relativizer should not be used with a shift instruction, otherwise the address part of the instruction which specifies the number of positions to be shifted will be mutilated.

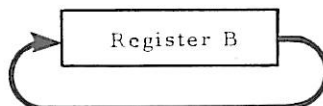
Function 54

2.8.1

Effect Causes the contents of Register B to be circulated to the left.

Operation A number held in Register B is effectively shifted *n* places to the left of its original position; *n* is determined by the number (between 00 and 12) in the two least-significant positions of the address part of the instruction. Those digits which move out of the most-significant position re-enter Register B at the least-significant position.

It is actually only possible to perform a right shift in Register B. Function 54 is therefore effected by a right shift of that number of places which achieves the same result as the specified left shift.



Example Circulate the contents of Register B five places to the left.

Instruction

D	F	A	R
	54	0005	-

Before

Register B

0	1	2	3	4	5	6	7	8	9	10	11
---	---	---	---	---	---	---	---	---	---	----	----

After

Register B

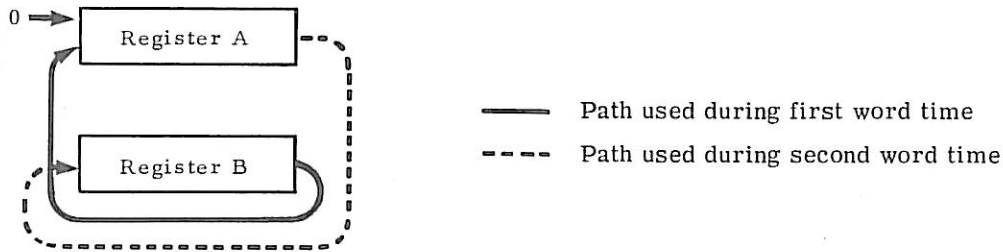
5	6	7	8	9	10	11	0	1	2	3	4
---	---	---	---	---	----	----	---	---	---	---	---

Function 55

2.8.2

Effect Causes the contents of Register B to be shifted to the left, entering zeros in the least-significant position.

Operation This function operates over two word times and unlike other shift functions utilizes Register A. This is necessary to achieve the specified left shift, which must in fact be performed as two right shifts. During the first word time Register A is zeroized and the digits which are to remain in the result are shifted from Register B into the most-significant end of Register A. During the second word time the contents of Register A are circulated into Register B to achieve the required result. Register A contains zeros on completion of the instruction.



Example Shift the contents of Register B six places to the left and fill the vacated positions with zeros.

Instruction

D	F	A	R
-	55	0006	-

Before

Register A	0	0	0	0	0	0	0	1	2	3	4	5
Register B	1	2	4	3	6	7	9	10	5	8	11	4

After

Register A	0	0	0	0	0	0	0	0	0	0	0	0
Register B	9	10	5	8	11	4	0	0	0	0	0	0

Note As this function takes longer to execute than function 54, wherever possible function 54 should be used in preference to function 55.

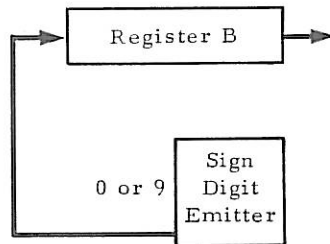
Function 56

2.8.3

Effect Causes the contents of Register B to be shifted to the right, entering either 0s or 9s in the most-significant position according to the value of the original digit in the most-significant position.

Operation A number held in Register B is shifted n places to the right of its original position: n is determined by the number (between 00 and 12) in the two least-significant positions of the address part of the instruction.

Those most-significant positions vacated are filled with either zeros if the original digit in the most-significant position was from 0 to 4, or nines if the original digit in the most-significant position was from 5 to 15. Thus the most-significant position is regarded as the sign position, 0 to 4 indicating a positive figure, 5 to 15 a negative figure.



Examples

- (a) Shift the contents of Register B five places to the right and fill the vacated positions with eight zeros if 0 to 4 is in position 1 of Register B or nines if 5 to 15 is in position 1 of Register B.

Instruction

D	F	A	R
	56	0005	-
---	---	---	---

Register B

Before

0	1	2	3	4	5	6	7	8	9	10	11
---	---	---	---	---	---	---	---	---	---	----	----

After

0	0	0	0	0	0	1	2	3	4	5	6
---	---	---	---	---	---	---	---	---	---	---	---

Register B

Before

9	9	1	2	3	4	5	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---

After

9	9	9	9	9	9	9	1	2	3	4	5
---	---	---	---	---	---	---	---	---	---	---	---

- (b) Shift the contents of Register B seven places to the right, propagating the sign.

Instruction

D	F	A	R
	56	0007	-
---	---	---	---

Register B

Before

3	4	5	6	7	8	3	4	5	6	7	8
---	---	---	---	---	---	---	---	---	---	---	---

After

0	0	0	0	0	0	0	3	4	5	6	7
---	---	---	---	---	---	---	---	---	---	---	---

Register B

Before

6	5	4	3	2	1	6	5	4	3	2	1
---	---	---	---	---	---	---	---	---	---	---	---

After

9	9	9	9	9	9	9	6	5	4	3	2
---	---	---	---	---	---	---	---	---	---	---	---

- (c) Shift the contents of Register B twelve places to the right, propagating the sign.

Instruction

D	F	A	R
	56	0012	-
---	---	---	---

Register B

Before

1	0	0	0	0	0	0	0	4	3	2	1
---	---	---	---	---	---	---	---	---	---	---	---

After

9	9	9	9	9	9	9	9	9	9	9	9
---	---	---	---	---	---	---	---	---	---	---	---

Register B

Before

4	6	0	0	0	0	1	2	3	4	5	6
---	---	---	---	---	---	---	---	---	---	---	---

After

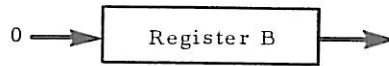
0	0	0	0	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---

Function 57

2.8.4

Effect Causes the contents of Register B to be shifted to the right, entering zeros in the most-significant position.

Operation A number held in Register B is shifted n places to the right of its original position; n is determined by the number (between 00 and 12) in the two least-significant positions of the address part of the instruction. The most-significant positions vacated are filled with zeros.



Examples

(a) Shift the contents of Register B five places to the right, do not propagate the sign.

D	F	A	R
	57	0005	-
---	---	---	---

Before

Register B											
0	1	2	3	4	5	6	7	8	9	10	11

After

0	0	0	0	0	0	1	2	3	4	5	6
---	---	---	---	---	---	---	---	---	---	---	---

(b) Shift the contents of Register B seven places to the right, do not propagate the sign.

D	F	A	R
	57	0007	-
---	---	---	---

Before

Register B											
6	5	4	3	2	1	6	5	4	3	2	1

After

0	0	0	0	0	0	0	6	5	4	3	2
---	---	---	---	---	---	---	---	---	---	---	---

Application of the Shift Functions

2.8.5

- (a) Zeroizing Register B. Instruction 570012 is the most economical method.
- (b) Extraction of part of a word: It is sometimes more economical in time or space or both to use shift functions in preference to Logical AND with a mask constant (described under 2.6.1). In the following examples, (i) the Shifting Method is more economical on storage, (ii) the Mask Constant Method is more economical on time.

Extract from word 25 of block 35 the quantity held in positions 9 to 12 and place it in word 26 of block 35.

(i) Shifting Method.

Instruction			
D	F	A	R
	37	0025	35
---	---	---	---
	54	0008	-
---	---	---	---
	57	0008	-
---	---	---	---
	42	0026	35
---	---	---	---

Before		After	
Word 25 Block 35	1 2 3 0 1 10 4 6 3 2 1 9	Word 25 Block 35	1 2 3 0 1 10 4 6 3 2 1 9
Word 26 Block 35	0 0 0 0 0 0 0 0 0 0 0 0	Word 26 Block 35	0 0 0 0 0 0 0 0 3 2 1 9
Register A	0 0 0 0 0 0 0 0 0 0 0 0	Register A	0 0 0 0 0 0 0 0 3 2 1 9
Register B	0 0 0 0 0 0 0 0 0 0 0 0	Register B	0 0 0 0 0 0 0 0 3 2 1 9

(ii) Masking Constant Method

Instruction	D	F	A	R
		37	0025	35
		35	0006	18
		42	0026	35

Before		After	
Word 6 Block 18	0 0 0 0 0 0 0 0 15 15 15 15	Word 6 Block 18	0 0 0 0 0 0 0 0 15 15 15 15
Word 25 Block 35	1 2 3 0 1 10 4 6 3 2 1 9	Word 25 Block 35	1 2 3 0 1 10 4 6 3 2 1 9
Word 26 Block 35	0 0 0 0 0 0 0 0 0 0 0 0	Word 26 Block 35	0 0 0 0 0 0 0 0 3 2 1 9
Register A	0 0 0 0 0 0 0 0 0 0 0 0	Register A	0 0 0 0 0 0 0 0 3 2 1 9
Register B	0 0 0 0 0 0 0 0 0 0 0 0	Register B	0 0 0 0 0 0 0 0 3 2 1 9

(c) Shifting the positions of quantities in a word, e.g. for rounding purposes.

Example 1 Round up quantity held in word 16 block 23.

Instruction	D	F	A	R
		60	0016	23
		62	0010	18
		56	0003	-
		42	0016	23

Before		After	
I.A.S. 10 Block 18	0 0 0 0 0 0 0 0 0 5 0 0	I.A.S. 10 Block 18	0 0 0 0 0 0 0 0 0 5 0 0
I.A.S. 16 Block 23	0 0 0 0 0 0 6 5 2 8 0 0	I.A.S. 16 Block 23	0 0 0 0 0 0 0 0 0 6 5 3
Register A	0 0 0 0 0 0 0 0 0 0 0 0	Register A	0 0 0 0 0 0 0 0 0 6 5 3
Register B	0 0 0 0 0 0 0 0 0 0 0 0	Register B	0 0 0 0 0 0 0 0 0 6 5 3

Example 2 Multiply word 129 of block 14 by word 28 block 13. Round up the product to the nearest penny and add the result to the quantity in word 39 block 15.

Instruction

I	D	F	A	R	NARRATIVE
x		37	0129	14	Transfer to Register B
		54	0003	-	Circulate left 3 positions
x+1		42	0129	14	Transfer to original locations
		37	0028	13	Transfer multiplier to Register B
x+2		22	0007	-	Set Sterling Position Register
		21	0003	-	Set Decimal Point Register
x+3		79	0129	14	Multiply
		72	0016	32	Round up
x+4		56	0003	-	Right Shift 3 positions
		22	0010	-	Reset Sterling Position Register
x+5		74	0039	15	Add to I.A.S.

Before

After

Word 28 Block 13	0 0 0, 0 0 0, 0 0 0, 9 9 9
Word 129 Block 14	0 0 0, 0 0 0, 0 0 9 1 9 1 1
Word 39 Block 15	0 0 0 0 0 0 0 0 0 0 0 0
Word 16 Block 32	0 0 0 0 0 0 0 0 0 5 0 0
Register B	0 0 0 0 0 0 0 0 0 0 0 0

Word 28 Block 13	0 0 0, 0 0 0, 0 0 0, 9 9 9
Word 129 Block 14	0 0 0 0 0 9 1 9 1 1 0 0 0
Word 39 Block 15	0 0 0 0 0 0 0 0 9 1 9 9
Word 16 Block 32	0 0 0 0 0 0 0 0 0 5 0 0
Register B	0 0 0 0 0 0 0 0 9 1 9 9

'DO NOTHING' AND 'STOP' FUNCTIONS

2.9

Function 00

2.9.1

Effect Causes the computer to do nothing and proceed to the next instruction in the normal manner.

Operation The computer obeys the instruction, therefore time is taken in operation but I.A.S. and registers concerning the programmer are in no way affected.

Notes This instruction may be used when it is necessary to modify the other instruction contained in the I.A.S. location. This may be more easily achieved by placing the instruction to be modified in the least-significant part of a location and having a 'do-nothing' instruction in the most-significant part e.g.

I	D	F	A	R
X	---	00	0012	00
		35	0009	12

The address will usually be zero, but as in the illustration, it may be any number between 0 and 3999.

Function ||

2.9.2

Effect Causes the computer to stop all further operations.

Operation The computer will stop all operations when this instruction is obeyed. If the address positions are used to contain a code number in the range 0 to 3999, after the stop the code number will be displayed on the console in CR3 with one added to it. Normal computer operation will be resumed when the Start button is operated.

Note The code numbers used in the address positions for this instruction can be used to indicate particular reasons for the stop e.g. when a sequence error occurs or an abnormal arithmetic condition is reached.

INDICATORS

2.10

Indicators are devices which may be in either one of two states, set or unset, and are used to direct the program into one of two branches according to the state of the indicator. This is achieved by writing an indicator test instruction. If the indicator is unset, the program proceeds to the normal next instruction. If the indicator is set, the program branches to the instruction pair in the I.A.S. location specified in the address part of the test instruction.

A summary of the characteristics of indicators numbers 00 to 29 is given in tabular form in the table overleaf. All other indicators (numbers 30 to 99) are dealt with in Part 3.

A complete summary of indicators is given in Part 6.

Representation of an Indicator Test Instruction

2.10.1

Some examples of indicator test instructions are given in Figure 6 (page 53). The first and second positions of the instruction contain the indicator number. The third position must contain a decimal digit whose binary representation contains a 1 in the 4 stream and a 0 in the 8 stream i.e. the number must be 4, 5, 6 or 7. This indicates to the computer that the instruction is a test instruction. The 4-bit is entered into the third position of the instruction during the Initial Orders program. On both the program sheet and the program card, the 4-bit occupies a separate column known as the designation column.

The third, fourth, fifth and sixth positions contain the I.A.S. address of the instruction to which the program will jump if the tested indicator is set. Thus in the case of a computer fitted with

more than 1,000 words of I.A.S., following completion of the Initial Orders program, position 3 of the instruction may contain both a 1 and a 4 in binary representation. For a machine with 4,000 words position 3 may also contain a 2-bit.

Indicator No.	Class of Indicator	Effect of Designation Indicator			Set by:-	Unset by:-
		4	8	9		
00	Unconditional Jump Indicator	Test	-	-	Permanently set	-
01	Mill Indicator	Test	-	-	Last no. through Mill zero.	Last no. through Mill not zero.
02	Mill Indicator	Test	-	-	Last no. through Mill > 0 (sign digit 0-4)	Last no. through Mill ≤ 0
03	Mill Indicator	Test	-	-	Last no. through Mill < 0 (sign digit 5-9)	Last no. through Mill ≥ 0
04	Error Indicator	Test & Unset	-	-	Overflow (sign digit 1-8 or sign digit 9 & all other digits zero)	Program when tested
06	Error Indicator	Test & Unset	-	-	I.A.S. Parity Check Error	Program when tested
07	Error Indicator	Test & Unset	-	-	Drum Parity Check Error	Program when tested
10 ' ' ' 19	Program Indicators	Test	Set	Unset	Instruction (Des.8)	Instruction (Des.9)
20 ' ' ' 29	Manual Indicators	Test	-	-	Manual Control on Console (Switch on)	Manual Control on Console (Switch off)

TABLE OF INDICATORS 00 TO 29

An indicator test instruction may form either the first or second half of an instruction word i.e. either positions 1 to 6 or 7 to 12.

Test instruction as written on program sheet	Form of instruction in computer following Initial Orders program	Significance of instruction										
<table border="1"> <thead> <tr> <th>I</th> <th>D</th> <th>F</th> <th>A</th> <th>R</th> </tr> </thead> <tbody> <tr> <td>21</td> <td>4</td> <td>03</td> <td>0037</td> <td>-</td> </tr> </tbody> </table>	I	D	F	A	R	21	4	03	0037	-	034037	Test Indicator 03; if set, jump to I.A.S. 37 (absolute address); if unset proceed to instruction in second half of I.A.S.21.
I	D	F	A	R								
21	4	03	0037	-								
<table border="1"> <thead> <tr> <th>I</th> <th>D</th> <th>F</th> <th>A</th> <th>R</th> </tr> </thead> <tbody> <tr> <td>36</td> <td>4</td> <td>15</td> <td>0106</td> <td>B</td> </tr> </tbody> </table> <p>Where block relativizer is set to 350</p>	I	D	F	A	R	36	4	15	0106	B	154456	Test Indicator 15; if set, jump to I.A.S. 106 of same block; if unset, proceed to instruction in second half of I.A.S. 36 of the same block.
I	D	F	A	R								
36	4	15	0106	B								
<table border="1"> <thead> <tr> <th>I</th> <th>D</th> <th>F</th> <th>A</th> <th>R</th> </tr> </thead> <tbody> <tr> <td>53</td> <td>4</td> <td>27</td> <td>0123</td> <td>72</td> </tr> </tbody> </table> <p>Relativizer for block 72 is set to 1860</p>	I	D	F	A	R	53	4	27	0123	72	275983	Test Indicator 27; if set, jump to I.A.S. 123 block 72 if unset, proceed to instruction in first half of I.A.S. 54 of the same block.
I	D	F	A	R								
53	4	27	0123	72								

Figure 6: EXAMPLES OF INDICATOR TEST INSTRUCTIONS

Incorrect Test Instructions

2.10.2

In the case of incorrect test instructions being given to the computer, the sequence of operations is as follows:

If the specified indicator exists, but the I.A.S. address is outside the specifiable range, then; if the indicator is unset, the computer will proceed in the normal manner to the next instruction; if the indicator is set, the computer will stop with zeros in control registers 1 and 2, and the "I.A.S. Check Error on transfer to control register" visual indicator will be set.

If the specified indicator does not exist the computer will proceed in the normal manner to the next instruction.

Automatic Indicators

2.10.3

Automatic indicators are set, and in some instances unset, by conditions arising in the central processor or in the peripheral units. The conditions arising in the peripheral units are described in Part 3. The other automatic indicators are numbers 00 to 07.

Indicator 00

2.10.4

Purpose Testing indicator 00 causes an unconditional jump to a specified word of I.A.S.

Operation This indicator is permanently set. When tested, the next program instruction will not be taken from the next word of I.A.S., but due to the jump, will be taken from the word of I.A.S. specified in the address part of the indicator test instruction.

Example It is required to transfer the contents of I.A.S. location 12 into I.A.S. location 19 if the contents are positive to zero, but to leave the contents of I.A.S. 19 unaltered if the contents of I.A.S. 12 are negative, I.A.S. 12 and 19 being relative addresses in block 17.

I	D	F	A	R	NARRATIVE
125	-	60	0012	17	Contents of I.A.S. 12, block 17, into Register B via Mill Test indicator 03, if set (set on < 0) jump to I.A.S. 127 of same block
	4	03	0127	B	
126	-	42	0019	17	If indicator 03 is not set, write contents of Register B in I.A.S. 19, block 17, then jump to I.A.S. 127 of same block
	4	00	0127	B	

Notes An unconditional jump may be usefully employed when it is not possible to use the second half of an I.A.S. location for an instruction. The use of an unconditional jump under these circumstances is preferable to the use of function 00 (Do-nothing instruction) because the former takes less time.

Mill Indicators

2.10.5

The Mill Indicators are indicators 01, 02, 03 and 04. These indicators may be set by any function which involves the transfer of words through the Mill i.e. functions 35, 60 to 69, 70 to 79, except that indicator 04 (the overflow indicator) cannot be set by function 35. A word in this case is regarded as a number up to eleven digits together with a sign digit in the first position.

The Mill Indicators are set by the number which leaves the Mill after completion of the appropriate arithmetical operations within the Mill and not by the numbers entering the Mill.

Indicators 01, 02 and 03 remain set until a number with a different sign results in the Mill. Indicator 04 remains set until it is tested; testing automatically unsets it.

The setting of indicators 01, 02, 03 and 04 is shown diagrammatically in Figure 7.

Figure 8 (page 56) depicts a portion of program in which indicators 01, 02, 03, and 04 are tested. The example assumes that according to the nature of the sum of the contents of I.A.S. locations 20 and 31 (relative addresses in block 10) the following action is required:

- (a) if an overflow occurs, the program must enter an error routine starting at I.A.S. location 80 of the same block of program;
- (b) if the sum of the contents is zero, positive, or negative, a jump is to be made to I.A.S. location 61, 65 or 73 respectively, also in the same block.

The last instruction could equally well be an unconditional jump because if indicators 02 and 03 are both unset, indicator 01 must be set.

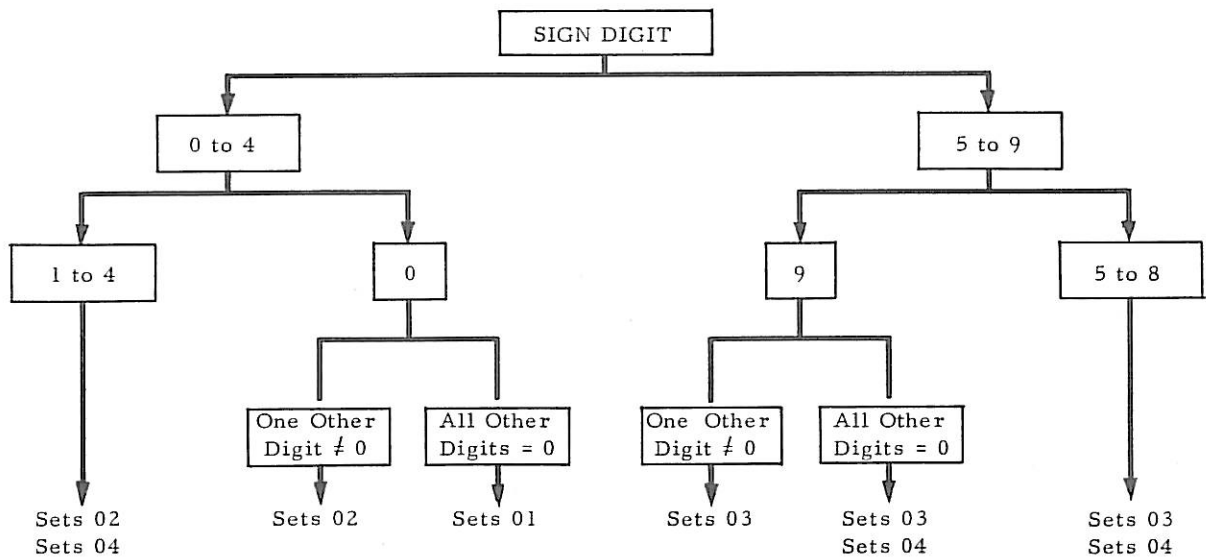


Figure 7: SETTING INDICATORS 01, 02, 03, AND 04

Indicator 01

2.10.6

Purpose May be used to test whether the last number resulting in the Mill was zero.

Operation Indicator 01 is set if the entire word is zero, i.e. if the sign digit and all other positions are zero (Figure 7). When indicator 01 is tested and found to be set, a jump is made to the I.A.S. word specified in the address part of the indicator test instruction.

The indicator remains set until a non-zero number results in the Mill.

Example See Figure 8.

Indicator 02

2.10.7

Purpose May be used to test whether the last number resulting in the Mill was positive.

Operation Indicator 02 is set if the word is positive, i.e. either if the sign digit is zero but at least one other digit is greater than zero, or, if the sign digit lies between 1 and 4 inclusive regardless of the state of the rest of the word (Figure 7). When indicator 02 is tested and is found to be set, a jump is made to the I.A.S. word specified in the address part of the indicator test instruction.

The indicator remains set until a negative or zero number results in the Mill.

Example See Figure 8.

Indicator 03

2.10.8

Purpose May be used to test whether the last number resulting in the Mill was negative.

Operation Indicator 03 is set if the word is negative, i.e. if the sign digit lies between 5 and 9 inclusive regardless of the state of the rest of the word (Figure 7). When indicator 03 is tested

and is found to be set, a jump will be made to the I.A.S. word specified in the address part of the indicator test instruction.

The indicator remains set until a positive or zero number results in the Mill.

Example See Figure 8.

Indicator 04

2.10.9

Purpose May be used to test whether an overflow has resulted in the Mill since the indicator was last tested.

Operation Indicator 04 (overflow indicator) is set if any of the arithmetical functions (functions 60 to 69, 70 to 79) causes either of the following two conditions to result in the Mill:

- (a) a sign digit with a value 1 to 8 inclusive, or
- (b) a 9 in the sign position with zeros in all other positions.

The overflow indicator remains set until it is tested, testing automatically unsets the indicator. Thus by testing the overflow indicator it is possible to detect whether or not an overflow has occurred into the sign position at any time since the indicator was last tested. If two numbers are added or subtracted and the overflow indicator is set, no information will have been lost. If two numbers are multiplied and the overflow indicator is set, information may have been lost.

When indicator 04 is tested and is found to be set, a jump will be made to the I.A.S. word specified in the address part of the indicator test instruction.

Example

I	D	F	A	R	NARRATIVE
57	4	04	0058	B	Test for overflow, to unset indicator
58	60	0020	10		Clear Add I.A.S. 20, block 10 to Register B.
	62	0031	10		Add I.A.S. 31, block 10 to Register B
59	4	04	0080	B	If overflow ind set, jump to I.A.S. 80, same block
	4	02	0065	B	If ind. 02 set (No. > 0), jump to I.A.S. 65, same block
60	4	03	0073	B	If ind. 03 set (No. < 0), jump to I.A.S. 73, same block
	4	01	0061	B	If ind. 01 set (No. = 0), jump to I.A.S. 61, same block

Figure 8: EXAMPLE INSTRUCTIONS TESTING INDICATORS 01, 02, 03, AND 04

Indicator 06 (I.A.S. Check Error Indicator)

2.10.10

Purpose Indicator 06 is used to test whether an I.A.S. transfer parity error has been detected since the indicator was last tested.

Operation An I.A.S. parity check is carried out when a word is transferred from I.A.S. to Register A.

Indicator 06 is automatically set if an I.A.S. parity error is detected on a transfer other than a transfer to the control registers.

Indicator 06 is unset when tested by program.

When the indicator is set the I.A.S. Error lamp on the console glows. If the Optional Stop switch is on, then the computer stops as soon as the parity error is detected. In this case the I.A.S. Error light is extinguished when the computer is restarted but indicator 06 remains set until tested by program.

Notes The computer always stops automatically if an I.A.S. parity error is detected on a transfer to the control registers. The I.A.S. parity checking system is described in more detail in 2.13.1.

The technique for restarting after I.A.S. parity failure is discussed in Part 4.

Indicator 07 (Drum Check Error Indicator) 2.10.11

Purpose Indicator 07 is used to test whether a drum transfer parity error has been detected since the indicator was last tested.

Operation A drum parity check is carried out when words are transferred from the drum.

Indicator 07 is automatically set if a drum parity error is detected.

Indicator 07 is unset when tested by program.

When the indicator is set the Drum Error lamp on the console glows. If the Optional Stop switch is on, then the computer stops as soon as the parity error is detected. In this case the Drum Error light is extinguished when the computer is restarted but indicator 07 remains set until tested by program.

Notes The drum parity checking system is described in more detail in 2.13.2.

The technique for restarting after drum parity failure is discussed in Part 4.

Programmed Indicators 2.10.12

The ten programmed indicators are indicators 10 to 19. These indicators are identical in operation and are set, unset and tested by means of instructions. The testing will not cause the indicators to be set or unset as a result of the test.

There are ten visual indicators (lamps) on the console, one for each indicator, which show the state of the indicators.

Indicators 10 to 19 2.10.13

Purpose These indicators are program controlled and are provided so that conditions can be designated and then later differentiated.

Operation Indicators 10 to 19 are set by an instruction containing an 8 in the designation position and are unset by a 9 in the designation position. Either of these two designations must be combined with the number of the indicator (10 to 19) in the first and second positions of the instruction. During Initial Orders, the 8 or 9 designation will be placed in the third position of the instruction.

The lamp associated with a particular indicator glows when that indicator is set. The light is extinguished when the indicator is unset.

Example It is required to set indicator 13 if the class of card designation held in position 12 of I.A.S. location 32 (a relative address in block 10) is a 5; (positions 0 to 11 are zeros); the indicator is to be unset later in the program.

Instruction

I	D	F	A	R	NARRATIVE
35	-	60	0032	10	<i>Class of card designation transferred to Reg. B</i>
	-	63	0006	15	
36	4	01	0037	B	<i>Test if ind. 01 set, i.e. if card designation is 5, if so jump to I.A.S. location 37 of this block</i>
	4	00	0038	B	
37	8	13	0000	-	<i>If indicator 01 set, set indicator 13</i>
	4	00	0038	B	
65	9	13	0000	-	<i>Unset indicator 13</i>

An example of a test instruction for indicator 15 is shown in Figure 6.

Manual Indicators 20 to 29

2.10.14

There are ten manual indicators, i.e. 20 to 29, which are all identical in operation. These indicators are set and unset by means of manual controls (switches) on the console display panel, the operation of one of these switches causing its corresponding indicator to be set or unset. The indicator will remain set until it is unset by operation of the appropriate switch, i.e. its state is unaffected by test instructions.

The manual indicators are intended for use in applications where it is desired to make minor alterations to a program without recourse to reading in a different program.

Example In a payroll application a manual indicator could effect a change-over from weekly to monthly payroll, the indicator being used to modify the program in such a way that the appropriate monthly tax constants would be used instead of weekly constants during the P.A.Y.E. calculations.

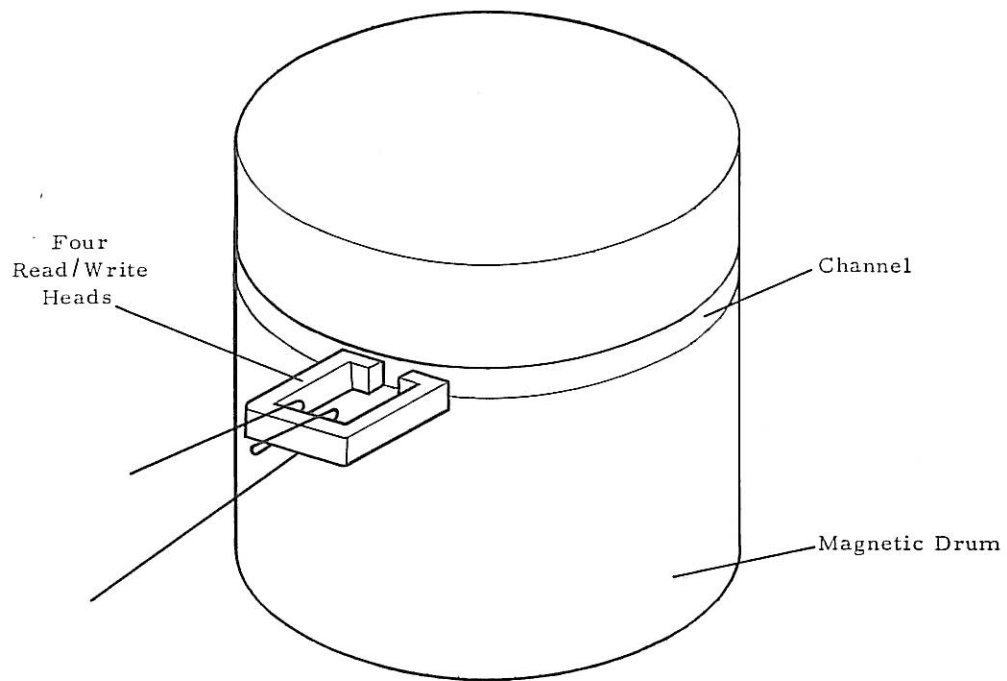


Figure 9: MAGNETIC DRUM

THE MAGNETIC DRUM

2.11

The magnetic drum is a backing store for the computer. Program instructions cannot be obeyed from the drum and the drum cannot be addressed directly other than to effect a transfer to or from I.A.S. Program instructions are provided for executing drum transfers.

The drum store is much larger than I.A.S. and it is therefore usual to hold the bulk of information on the drum and to transfer the information to I.A.S. as it is required. However, access to the drum is time consuming and the program should be arranged so that drum transfers are kept to a minimum.

A standard drum has 12,000 storage locations. A computer may be fitted with a quarter drum of 3,000 words, a half drum of 6,000 words, or one or more standard drums up to a maximum of eight drums.

Each 12,000 word drum is divided into two sections of 6,000 words and for timing purposes is regarded as being two separate drums. There are no such divisions on the 3,000 word drums or the 6,000 word drums.

The drum (Figure 9) is divided along its length into channels. A series of read/write heads are positioned along the length of the drum for transferring information to and from each channel. Each channel contains 200 words divided into groups of 10 words each called decades. The drum rotates at a constant speed, the locations moving in sequence past the read/write heads.

The decade is the smallest unit of transfer possible. In a drum transfer instruction, the drum address specified is a decade address. The largest unit which can be transferred by one instruction is 20 decades. It is possible to effect drum transfers either as a specified number of decades or as a complete channel. If the transfer is more than one decade, then it is possible for the decades to be taken from two separate channels provided that the decade numbering sequence is not broken.

The decade addresses of locations on the drum are shown in the table below.

Words	Drum Number	Decade Addresses
3,000	1	0000 to 0299
6,000	1	0000 to 0599
12,000	1	0000 to 1199
12,000	2	1200 to 2399
12,000	3	2400 to 3599
12,000	4	3600 to 4799
12,000	5	4800 to 5999
12,000	6	6000 to 7199
12,000	7	7200 to 8399
12,000	8	8400 to 9599

DECADE ADDRESSES OF 3,000, 6,000 AND 12,000 WORD DRUMS

If an address outside the range for any machine is specified in an instruction, the machine will stop due to failure to synchronize the clock source and the Slip Pulse lamp will glow.

The gap between decades is one word length and channel switching (i.e. when transferring decades from the end of one channel and the beginning of the next) occurs while this gap is passing the read/write heads.

Example Decades 59 and 60 (i.e. the last decade of one channel and the first decade of the next) are to be transferred. The transfer will be completed 21 word times after the start of the transfer of the first word of decade 59.

It is apparent that when transferring a specified number of decades the transfer time is the same irrespective of whether the transfer extends over one or two channels. This rule does not apply however when switching from the last decade of the first drum section to the first decade of the second drum section or when channel switching between drums.

Switching between drums or switching between drum sections takes up to just over one drum revolution.

Example 1 If decades 1199 and 1200 (i.e. the last decade of drum number 1 and the first decade of drum number 2) are transferred, then, on completion of the transfer of decade 1199, the transfer stops until the central processor is clocked by the second drum. This may take up to just over one drum revolution time. When the computer is clocked, decade 1200 is positioned at the read/write heads and the second transfer is effected.

Example 2 If decades 599 and 600 (i.e. the last decade of the first drum section and the first decade of the second drum section) are transferred, then, on completion of the transfer of decade 599, the transfer stops until the central processor is clocked by the second section. This may take up to just over one drum revolution time. When the computer is clocked, decade 600 is positioned at the read/write heads and the second transfer is effected.

In the two examples given, the maximum time taken for transfer of the two decades is a normal access time, plus a little over one drum revolution clocking time, plus the normal transfer time for two decades. The clock generator of the central processor is synchronized either by the drum or drum section currently in use, or by the drum or drum section to which reference was last made.

The arrangements referred to in the preceding paragraphs are dealt with completely automatically within the computer. It is only necessary to take note of these times under special circumstances which involve time-sharing of the central processor between drum references and input or output equipment.

In addition to the storage quoted for each drum, there are a further two channels (i.e. 400 locations) of reserved storage. On the first drum of each machine the reserved storage holds the Initial Orders program and engineer's Test Routines. The reserved storage is not accessible to the programmer for transfers from I.A.S. though transfers to I.A.S. from the reserved channels are permissible.

Drum Instructions

2.11.1

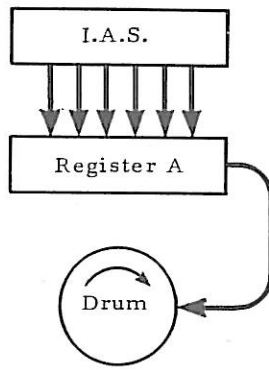
All transfer instructions from and to the drum are double-length and occupy all twelve digits of a storage location. There are two types of transfers; decade transfers and channel transfers. The functions are 80 to 87; functions 84 to 87 involve the use of the reserved store.

Function 80

2.11.2

Effect Transfers the contents of a specified number of decades of I.A.S. to the drum.

Operation The first location of I.A.S. from which the transfer is made is that specified in the instruction, successive transfers being from successively higher numbered locations of I.A.S. The information is taken from I.A.S. and is placed in Register A before being written on the drum. The first location of the drum to which the transfer is to be made is the first location of the drum decade specified in the instruction, successive locations being transferred to the second, third ... tenth locations of that decade followed by the first, second, third ...tenth locations of successively higher numbered decades. The contents of I.A.S. remain unaltered.



Example The contents of I.A.S. locations 249 to 388 (i.e. 14 decades starting from location 249) are to be transferred to decades 0805 to 0818 on the drum.

Instructions

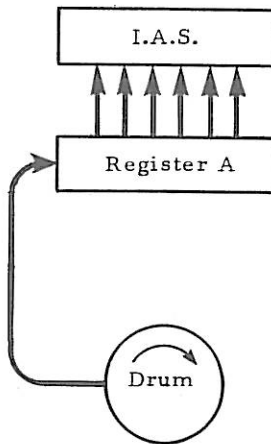
I	D	F	A	R
X	---	90	0249	-
	---	14	0805	-

Function 81

2.11.3

Effect Transfers the contents of a specified number of drum decades to I.A.S.

Operation The first location of the drum from which the transfer is made is that specified in the instruction, successive transfers being from the second, third.....tenth location of that decade followed by the first, second, third.....tenth locations of successively higher numbered decades. The information is taken from the drum and is placed in Register A before being written into I.A.S. The first location of I.A.S. to which the transfer is to be made is the first location of I.A.S. specified in the instruction, successive transfers being to successively higher numbered location of I.A.S. The contents of the drum remain unaltered.



Example The contents of drum decades 0805 to 0818 (i.e. 140 words starting from decade 0805) are to be transferred to I.A.S. locations 249 to 388.

Instruction	I	D	F	A	R
X	-	-	81	0249	-
			14	0805	-

Function 82

2.11.4

Effect Transfers the contents of 200 locations of I.A.S. to a specified channel on the drum.

Operation The first location of I.A.S. from which the transfer is made must be a multiple of 200, i.e. 0000, 0200, 0400..... or 3800. Successive words are transferred from successively higher numbered locations of I.A.S. The information is routed through Register A. The drum address specified in the instruction is the lowest numbered decade of the channel to which the transfer is to be made. The transfer begins at the start of the first decade to reach the read/write heads. This means that if I.A.S. locations 0 to 199 are to be transferred to drum decades 60 to 79 and drum decade 74 is the first decade to reach the read/write heads, then; I.A.S. word 140 is transferred to the first location of drum decade 74 followed in sequence by words 141, 142199, 0, 1, 2..... to 139. The drum decades will be filled in the sequence 74, 75 79, 60, 61 to 73.

Example The contents of I.A.S. locations 200 to 399 are to be transferred to the 32nd channel on the drum (i.e. decades 0620 to 0639).

Instruction	I	D	F	A	R
X	-	-	82	0200	-
			20	0620	-

Function 83

2.11.5

Effect Transfers the contents of a specified channel of the drum to 200 locations of I.A.S.

Operation As with function 82, the I.A.S. address must be 0000, 0200, 0400, ...or 3800. Also, the drum address specified in the instruction is the lowest numbered decade of the channel from which the transfer is made. The transfer begins with the first location of the first decade to reach the read/write heads.

Example The contents of the 50th channel of the drum (i.e. decades 0980 to 0999) are to be transferred to locations 0800 to 0999 of I.A.S.

Instruction	I	D	F	A	R
X	-	-	83	0800	-
			20	0980	-

Note on Functions 82 and 83 If two channel transfers are succeeding instructions, then the access time for the second channel transfer will be zero provided that both the transfers are to or from the same drum or to or from the same drum section on 12,000 word drums.

Reserved Storage

2.11.6

As stated in 2.11, there are two additional channels of reserved storage on each drum from which the programmer may transfer information or instructions to I.A.S., but to which the programmer can only transfer from I.A.S. if an engineer's adjustment is made to the computer.

On the first drum, part of this storage is used to hold the Initial Orders program of instructions, which could be mutilated accidentally unless there was some restriction placed on its use by the programmer.

This storage is decade addressed as follows:-

- (a) 0000 to 0019 plus 1200 (n - 1) } where n is the number of the drum and
- (b) 0100 to 0119 plus 1200 (n - 1) } n = 1 for 3,000 word and 6,000 word drums.

It will be appreciated that these addresses are also decade addresses of the 'free' storage on each drum, the distinction being drawn by the computer according to the function codes used. Functions 84, 85, 86 and 87 are used when transferring information or instructions to or from the reserved channels.

If any of these functions is used in such a way that the drum address in the instruction is outside the specifiable address range for any of the reserved channels then the transfer will be made to or from the address specified in free storage on the drums.

Function 84

2.11.7

Effect Transfers the contents of a specified number of decades of I.A.S. to the reserved channels of the drum.

Operation Similar to function 80, but can only be accomplished with the assistance of the engineer and has restricted use.

Example The contents of locations 129 to 248 of I.A.S. are to be transferred to decades 0102 to 0113 of reserved storage on the first drum.

Instruction	I	D	F	A	R
X			84	0129	-
			12	0102	-

Function 85

2.11.8

Effect Transfers the contents of a specified number of decades of drum reserved storage to I.A.S.

Operation Similar to function 81.

Example The contents of decades 0004 to 0012 of reserved storage on the first drum are to be transferred to locations 0302 to 0391 of I.A.S.

Instruction	I	D	F	A	R
X	--	--	85	0302	--
			09	0004	

Function 86

2.11.9

Effect Transfers the contents of 200 locations of I.A.S. to one of the reserved channels of the drum.

Operation Similar to function 82 but can only be accomplished with the assistance of the engineer and has restricted use.

Example The contents of location 1200 to 1399 of I.A.S. are to be transferred to the second reserved channel of the third drum.

Instruction	I	D	F	A	R
X	--	--	86	1200	-
			20	2500	-

Function 87

2.11.10

Effect Transfers the contents of a reserved channel of the drum to 200 locations of I.A.S.

Operation Similar to function 83.

Example The contents of the first reserved channel of the fifth drum are to be transferred to locations 1800 to 1999 of I.A.S.

Instruction	I	D	F	A	R
X	--	--	87	1800	-
			20	4800	-

Relative Addressing of Drum Instructions

2.11.11

The drum instructions have been explained using absolute addresses. Relative addresses can be used, in which case the I.A.S. address and the drum decade address can refer to the same, or to different relativizers. The appropriate relativizer reference number must be placed in each half of the double-length instruction.

Example If R.R.N. 15 has an I.A.S. address of 800 and a drum location address of 9800, the instruction

I	D	F	A	R
X	--	83	0	15
		20	0	15

will be obeyed in the same way as

I	D	F	A	R
X	--	83	0800	-
		20	0980	-

and the contents of the 50th channel of the drum (decades 980 to 999) will be transferred to locations 800 to 999 of I.A.S.

Example If R.R.N. 17 has an I.A.S. address of 240 and a drum location address of 5000, and R.R.N. 18 has an I.A.S. address of 360 and a drum location address of 8000 the instruction

I	D	F	A	R
x	-	81	9	17
		14	5	18

will be obeyed in the same way as

I	D	F	A	R
x	-	81	0249	-
		14	0805	-

and the contents of drum decades 0805 to 0818 will be transferred to I.A.S. locations 249 to 388.

THE CONTROL REGISTERS

2.12

Instructions are normally obeyed in the sequential order in which they are stored in I.A.S. Instructions are transferred from I.A.S. by way of Register A to the control registers before being obeyed.

There are three control registers CR1, CR2 and CR3, each six digits in length and therefore capable of holding one single-length instruction apiece. The circuitry of the control registers is so arranged that after the second instruction in a word has been obeyed, control is transferred automatically to the first instruction of the next word. This sequence is broken only when a programmed jump occurs.

An instruction is actually obeyed when the instruction is in CR1. Figure 10 shows two transfer paths, one for an instruction other than a jump instruction and the other for a jump instruction. The sequence of operations for each instruction is as follows:

- (a) An instruction other than an indicator test entering CR1 is obeyed. The contents of CR3 move into CR2, the previous contents of CR2 move into CR1 and the previous contents of CR1 have one added in the least-significant position before being moved into CR3.
- (b) When an indicator test enters CR1 and is unsuccessful, then the transfer sequence is as follows:
 - (i) The contents of the first half of the I.A.S. location specified in the instruction are transferred to the second half of Register A; the first half of Register A will contain zeros in all six positions.
 - (ii) The contents of CR3 move into CR2, the previous contents of CR2 move into CR1 and the previous contents of CR1 have one added in the least-significant position before being moved into CR3.
- (c) When an indicator test instruction enters CR1 and is successful, a change of control takes place and the transfer sequence is as follows:
 - (i) The contents of the I.A.S. location in the address part of the instruction are moved from I.A.S. into Register A.

- (ii) The contents of Register A are moved into CR1 and CR2. The previous contents of CR2 and CR3 are moved into Register A and the previous contents of CR1 have one added in the least-significant position before being moved into CR3.

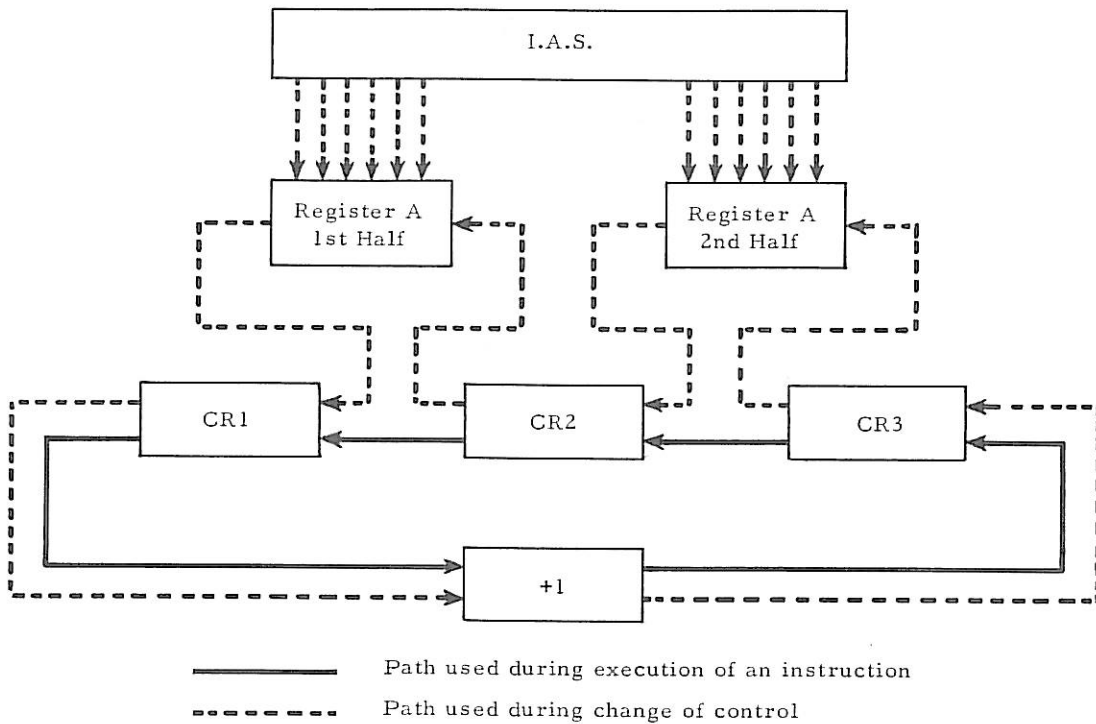


Figure 10: THE CONTROL REGISTERS

An indicator test instruction counts as a jump instruction only if it is successful.

When an indicator test instruction is transferred from CR1 to CR3 (with one added) the function digits are zeroized i.e. the instruction is converted into an unconditional jump instruction.

The following example shows the contents of the different registers for the given portion of program.

I	D	F	A	R	NARRATIVE
20		37	120	-	Transfer word 120 of I.A.S. to Register B
		68	121	-	Compare Register B with I.A.S. 121
21	4	02	22	-	Test indicator 02 If difference > 0
	8	11			Jump to word 22
22					otherwise set indicator 11

Assume location 120 contains 00000000615 and that location 121 contains 00000000600 thus making indicator 02 unset.

	CR1	Register A First Half	CR2	Register A Second Half	CR3
Instruction in CR1 obeyed	370120	000000	680121	000615	004021
Contents of control registers shifted - Normal path.	680121	000000	004021	000615	370121
Instruction in CR1 obeyed	680121	999999	004021	999985	370121
Contents of control registers shifted - Normal path.	004021	999999	370121	999985	680122
Change of control. Contents of location 21 enter Register A.	004021	024022	370121	118000	680122
Contents of control registers shifted - Change of control path.	024022	370121	118000	680122	004022
Instruction in CR1 obeyed - unsuccessful indicator test.	024022	000000	118000	Word 22a	004022
Contents of control registers shifted - Normal path, function digits zeroized.	118000	000000	004022	Word 22a	004023
Instruction in CR1 obeyed	118000	000000	004022	Word 22a	004023
Contents of control registers shifted - Normal path.	004022	000000	004023	Word 22a	118001
Change of control. Contents of location 22 enter Register A.	004022	Word 22a	004023	Word 22b	118001
Contents of control registers shifted - change of control path.	Word 22a	004023	Word 22b	118001	004023

It is apparent that adding one to the contents of CR1 before storing in CR3 causes the words to be obeyed in sequential order unless there is a programmed jump to another part of the program.

For explanation purposes the contents of the registers have been shown as though the control register shift does not take place until the instruction in CR1 has been obeyed. In practice the shift takes place as soon as the instruction in CR1 has been initiated and while it is still in the process of being obeyed. For a control change the control register shift is also an integral part of the change of control process. In either case, therefore, the shifts will always be carried out before the computer stops. The quantities in heavy outline (e.g. 680121 000000 004021 000615 370121) are the numbers which will be displayed in the registers if the computer is stopped manually after each instruction.

It is apparent that:

- (a) CR3 contains the instruction which has just been obeyed with one added to its address.
- (b) CR1 contains the next instruction to be obeyed if the computer is restarted.
- (c) When there has been a change of control between consecutive words of program, Register A contains the previous two instructions which have been obeyed, each with one added to its address (and indicator number zeroized if the instruction was an indicator test).

The following example shows the contents of the different registers for the same portion of program used in the preceding example with the assumption that:

Location 120 contains 00000000500 and Location 121 contains 00000000600 thus causing indicator 02 to be set.

	CR1	Register A First Half	CR2	Register A Second Half	CR3
Instruction in CR1 obeyed	370120	000000	680121	000500	004021
Contents of control registers shifted - Normal path.	680121	000000	004021	000500	370121
Instruction in CR1 obeyed	680121	000000	004021	000100	370121
Contents of control registers shifted - Normal path.	004021	000000	370121	000100	680122
Change of control. Contents of location 21 enter Register A.	004021	024022	370121	118000	680122
Contents of control registers shifted - Change of control path.	024022	370121	118000	680122	004022
Programmed change of control. Contents of location 22 enter Register A.	024022	Word 22a	118000	Word 22b	004022
Contents of control registers shifted - Change of control path. Function digits zeroized.	Word 22a	118000	Word 22b	004022	004023

Note that zeroizing the function digits on transfer to CR3 puts an 'unconditional jump to word 23' into CR3. This will ensure that when the instructions in word 22 have been obeyed, control will be transferred to word 23.

The contents of Register A, following a programmed change of control, provide a convenient method for a general purpose routine to restore control to the main program. The contents of Register A following a control change can also be useful for modification purposes. These techniques are discussed in Part 4.

When the first half of a double-length instruction enters CR1 this is detected by the machine and the instruction is obeyed from both CR1 and CR2. During the execution of double-length instructions, the instructions are modified in the control registers. When the instruction has been obeyed it is transferred to CR2 and CR3 but the addresses of both halves may have been modified (other than by having one added). At the change of control which immediately follows a double-length instruction, the contents of CR2 and CR3 are transferred to Register A. These contents can be useful for modification purposes and the relevant details are given in Part 4.

PARITY CHECKING SYSTEMS

2.13

Facilities are provided for checking errors which might occur when information is being transferred either between I.A.S. and the registers, or, between the drum and I.A.S. A word contains 48 data bits representing the 1, 2, 4 and 8 streams for each digit position. In addition to the data bits a word contains extra bits which are used for checking purposes only.

I.A.S. Parity Checking

2.13.1

A word in I.A.S. consists of 48 data bits and two additional bits termed check bits. The two check bits are generated every time a word enters Register A prior to being stored in I.A.S. One check bit is associated with the first half of the word and the other with the second half of the word. The first half of the word consists of the six most-significant digits and the second half of the word consists of the six least-significant digits.

In Register A the word is examined to ascertain whether the sum of the 1-bits in each half of the word is odd or even. If the sum of the 1-bits in the first half of the word is an odd number, a 1-bit is generated to make the total even. This ensures that the first half of every word entering I.A.S. from Register A is of even parity. If the sum of the 1-bits in the second half of the word is an even number, a 1-bit is generated to make the total odd. This ensures that the second half of every word entering I.A.S. from Register A is of odd parity.

Example The word 012345678901 is to be examined to ascertain the parity of each half of the word.

This is recorded in storage as:-

	0	1	2	3	4	5	6	7	8	9	0	1
1	0	1	0	1	0	1	0	1	0	1	0	1
2	0	0	1	1	0	0	1	1	0	0	0	0
4	0	0	0	0	1	1	1	1	0	0	0	0
8	0	0	0	0	0	0	0	0	1	1	0	0

Number of 1-bits in first half (012345) = 7.

This is an odd number, therefore a 1-bit is generated to make the total even (8).

Number of 1-bits in second half (678901) = 9.

This is an odd number therefore no 1-bit is generated and the total remains odd (9).

Whenever a word enters Register A from I.A.S. it is checked for even parity in the first half and for odd parity in the second half. Then:

- (a) If the parity check fails during a transfer to the control register the computer stops automatically and the I.A.S. to Control Register lamp glows.
- (b) If the parity check fails during any other transfer from I.A.S., indicator 06 is set, the I.A.S. Error lamp glows and if the Optional Stop switch is set, the computer stops.

A parity check failure means that a transfer error has occurred either during the current transfer from I.A.S., or during the previous transfer to I.A.S.

When a transfer is made from I.A.S. to Register A the parity check is carried out and then the I.A.S. parity bits are regenerated and the word is re-stored in the location from which it originated. The programming implications of this are discussed in Part 4.

Drum Parity Checking

2.13.2

A word on the drum consists of 12 data digits and one extra digit for checking purposes. The check digit is generated immediately before a word is transferred to the drum. The check digit is generated as follows:

- (a) The sum of the 1-bits in the data word is subtracted from 14.
- (b) The number of times 14 can be subtracted from the sum of the 1-bits is multiplied by 16.
- (c) The product obtained in the second operation is added to the result obtained in the first operation to produce the check digit.

Example

Word	No. of 1-bits	Step 1	Step 2	Step 3	Check Digit
001234432100	10	$14 - 10 = +4$	$10 \div 14 = 0 + \text{remainder}, 0 \times 16 = 0$	$+4 + 0 =$	4
123456789012	17	$14 - 17 = -3$	$17 \div 14 = 1 + \text{remainder}, 1 \times 16 = 16$	$-3 + 16 =$	13
777777777777	36	$14 - 36 = -22$	$36 \div 14 = 2 + \text{remainder}, 2 \times 16 = 32$	$-22 + 32 =$	10
$\overline{15} \overline{15} \overline{15} \overline{15} \overline{15} \overline{15} \overline{15} \overline{15} \overline{15} \overline{15}$	48	$14 - 48 = -34$	$48 \div 14 = 3 + \text{remainder}, 3 \times 16 = 48$	$-34 + 48 =$	14

When a word is transferred from the drum, the check digit is calculated from the transferred data bits. This check digit is compared with the check digit stored with the word. If these are not the same then the parity check fails, indicator 07 is set and the Drum Error lamp glows. If the Optional Stop switch is set then the computer stops.

A parity check failure means that a transfer error has occurred either during the current transfer from the drum or when the information was previously transferred to the drum. The programming implications of this are discussed in Part 4.

Timings

2.14

The following table shows the speeds and timings associated with the central processor. Notes on using these to time a complete program are given in Part 4.

	Function	Time taken to execute function
Transfer Instructions	37	21 μ s
	40	21 μ s
	41	21 μ s
	42	21 μ s
	43	21 μ s
	44	17 μ s
	45	26 μ s per word transferred
Decimal and Sterling Addition and Subtraction	60, 70	21 μ s
	61, 71	21 μ s
	62, 72	21 μ s
	63, 73	21 μ s
	64, 74	25 μ s
	65, 75	25 μ s
	66, 76	25 μ s
	67, 77	25 μ s
	68, 78	26 μ s
	22	12 μ s
21	12 μ s	
Multiplication	69, 79	Maximum $44(6n + 2 + m)$ μ s Minimum $44(n + 1 + m)$ μ s where n = number of digits in the multiplier (excluding non- significant zeros) and $m = P - n$ if positive and 0 otherwise, where P is the number entered in the Decimal Point Register.
Logical Functions	35	21 μ s
	36	21 μ s
Row-binarizing	30	21 μ s
	31	21 μ s
	32	21 μ s
	33	21 μ s
	34	21 μ s
Shift Instructions	54	17 μ s
	55	34 μ s
	56	17 μ s
	57	17 μ s

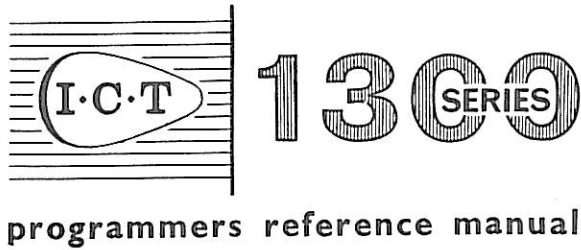
	Time taken to execute function
Indicators	A program instruction to test, set or unset any indicator takes 12 μ s. (See Note.)
Do Nothing	The 'Do Nothing' instruction takes 12 μ s.
The Magnetic Drum	The magnetic drum revolves at 5240 r.p.m. i.e. one drum revolution takes 11.45 ms.
	For a decade transfer the access time varies between 0 and 1 drum revolution. Therefore average access time = 5.7 ms. Transfer time = .57 ms per decade. Hence for functions 80, 81, 84, 85, average time = (5.7 + .57n) ms. Maximum time = (11.4 + .57n) ms. Where n is the number of decades transferred. If there is a change of drum or a change to the other half of a 12,000 word drum (this may be before or during a transfer) a maximum of 12 ms must be added.
	For a channel transfer the access time varies between 0 and the time for the decade to pass the read/write heads. Transfer time = 1 drum revolution. Therefore for functions 82, 83, 86, 87, average time = 11.7 ms; maximum time = 12ms. If there is a change of drum or a change to the other half of a 12,000 word drum before the transfer a maximum of 12 ms must be added.
The Control Registers	The change of control between two consecutive words of program takes 12 μ s. (See Note.)

Note An indicator test instruction takes 12 microseconds whether or not it is successful. If it is successful then this time represents the change of control, hence, when jumping to a word of program, this takes the place of the normal 12 microseconds control change from the previous word. It follows that a jump instruction effectively takes no time at all and for this reason, if the second half of a word has to be wasted, it is preferable to use an unconditional jump rather than a 'Do Nothing'.

Example

C		D	F	A	R	NARRATIVE
	125	-	60	0012	17	<i>Contents of I.A.S. 12, block 17, into Register B via Mill</i> <hr/> <i>Test indicator 03; if set (set on < 0) Jump to word 127 of block</i>
		4	03	0127	B	
	126	-	42	0019	17	<hr/> <i>If ind. 03 not set, write contents of Reg. B in I.A.S. 19 of block 17 then jump to word 127 of block</i>
		4	00	0127	B	

Here the unconditional jump instruction brings about an immediate control change and the contents of location 127 are transferred to the control registers. A 'Do Nothing' would have taken 12 microseconds and would then have been followed by a control change.



PERIPHERAL EQUIPMENT

Contents

	Page
3.1 INTRODUCTION	1
3.2 THE CARD READER	2
3.2.1 Time Available for Programming	4
3.2.2 Card Read Instructions	5
Instruction 380002	5
Instruction 380007	5
3.2.3 Card Read Indicators	5
Indicator 35 Card Read Ready	5
Indicator 36 6 Columns Read	6
Indicator 37 6 Columns Missed	6
Indicator 38 Mismatch	6
3.2.4 A Card Read Program	7
3.2.5 Timings	11
3.3 THE CARD PUNCH	13
3.3.1 Checking of Punching	13
3.3.2 Punch Instructions	15
Instruction 380042	15
Instruction 380043	15
Instruction 380044	15
Instruction 380045	16
Instruction 380046	16
Instruction 380047	16
3.3.3 Card Punch Indicators	17
Indicator 54 Punch Ready	17
Indicator 55 Punch Index Point Time	17
Indicator 56 Check Index Point Time	17

Contents continued		Page
	Indicator 57 Punch Index Point Time Missed	17
	Indicator 58 Check Index Point Time Missed	18
3.3.4	Distribution of Punch Data	18
3.3.5	Flowchart of a Punch Program	19
3.3.6	Timings	23
3.4	THE LINE PRINTER	25
3.4.1	The Print Unit	25
	Print Barrel	25
	Print Characters	25
	Print Hammers	26
	Printing	27
3.4.2	Print Instructions	28
	Instruction 380013	28
	Instruction 380014	28
	Instruction 380015	29
	Instruction 380016	29
3.4.3	Print Indicators	29
	Indicator 42 Printer Ready	29
	Indicator 43 Print Index Point Time	30
	Indicator 44 Print Character Time	30
	Indicator 49 Print Count Error	30
3.4.4	Distribution of Print Data	31
	A Printing Program	31
3.4.5	Flowchart of a Print Program	33
3.4.6	Paper Movement	38
3.4.7	Paper Throw Instructions	40
	Sprag Instructions 380020 to 380026	40
3.4.8	Paper Throw Indicators	40
	Indicator 45 Line Space Time	40
	Indicator 47 Paper Trolley Empty	40
3.4.9	Flowchart for a Paper Throw Program	41
3.4.10	Timings	43
3.5	THE PAPER-TAPE READER	45
3.5.1	Interpretation within the Computer	45
	Valid Tape Characters	45
3.5.2	Parity Checking	47
3.5.3	Tape Read Instructions	47
	Instruction 380050	47
	Instructions 380051 and 380052	47
3.5.4	Tape Read Indicators	48
	Indicator 60 Tape Reader Ready	48
	Indicator 61 Parity Error	48
3.5.5	Tape Read Program	49
3.5.6	Timings	49

Contents continued		Page
3.6	THE PAPER-TAPE PUNCH	50
3.6.1	Output from the Computer	50
3.6.2	Checking Facilities	51
3.6.3	Tape Punch Instructions	51
	Instruction 380076	51
3.6.4	Tape Punch Indicators	51
	Indicator 65 Tape Supply Low	51
	Indicator 66 Tape Punch Ready	52
	Indicator 67 Tape Punch Error	52
3.6.5	Tape Punch Programs	53
3.6.6	Timings	53
3.7	THE INTERROGATING TYPEWRITER	54
	Keyboard	54
	Print Unit	54
3.7.1	Representation within the Computer	56
3.7.2	Typewriter Instructions	57
	Instruction 380070	57
	Instruction 380071	57
	Instruction 380072	58
3.7.3	Typewriter Indicators	59
	Indicator 50 Paper Supply Low	59
	Indicator 51 Typewriter Ready	59
	Indicator 52 Request Type-in	60
	Indicator 53 Carriage at End	60
	Indicator 59 Typewriter Mechanical Failure	61
3.7.4	Interrogating Typewriter Programs	61
3.7.5	Timings	61
3.8	THE ONE-INCH (90 kc/s) AND HALF-INCH (22½ kc/s) MAGNETIC-TAPE SYSTEMS	62
3.8.1	Tape Layout	63
3.8.2	Tape Organization	64
3.8.3	Checking Facilities	64
3.8.4	Tape Deck Addresses and Queueing	65
3.8.5	Magnetic-tape Instructions	66
	Tape Write Instruction	66
	Tape Read Instruction	66
	Backspace Instruction	67
	Cancel Instruction	68
	Rewind Instruction	68
	Unload Instruction	69
3.8.6	Magnetic-tape Indicators	71
	Indicators 81 to 88 Deck Address Ready	71
	Indicator 89 Transport Mechanically Ready	71
	Indicator 80 Tape Order Error	72
	Indicator 70 Write Unit Ready	73
	Indicator 72 Read Unit Ready	73
	Indicator 74 Any Errors	74
	Indicator 75 Multiple Errors	74

Contents continued		Page
	Indicator 76 End of Tape	74
	Indicator 77 Early End of Tape and Short Block	75
	Indicator 71 Write Master	75
	Indicator 73 Read Master	76
	Indicator 79 Writing Ring Present	77
3.8.7	Program Interrupt Facility	77
	Reading Tape	77
	Writing Tape	78
3.8.8	Flowcharts for Reading and Writing Tape	81
	Deck Testing Subroutine	81
	Tape Write Subroutine	83
	Write Exceptions Routine	85
	Tape Read Subroutine	87
	Read Exceptions Routine	89
3.9	THE ONE-INCH (90 kc/s) MAGNETIC-TAPE SYSTEM	90
3.9.1	Representation of Data on Tape	90
3.9.2	Timings and Statistics	93
3.9.3	Timing	94
3.10	THE HALF-INCH (22½ kc/s) MAGNETIC-TAPE SYSTEM	95
3.10.1	Representation of Data on Tape	95
3.10.2	Timings and Statistics	97
3.10.3	Timings	97
3.11	THE QUARTER-INCH (16 kc/s) MAGNETIC-TAPE SYSTEM	99
3.11.1	Tape Layout	99
3.11.2	Tape Organization	100
3.11.3	Checking Facilities	101
	Writing Tape	101
	Reading Tape	102
3.11.4	Tape Deck Addresses	102
3.11.5	Magnetic-tape Instructions	103
	Tape Write Instruction	103
	Tape Read Instructions	103
	Backspace Instruction	104
	Cancel Instruction	105
	Rewind Instruction	105
	Unload Instruction	106
3.11.6	Magnetic-tape Indicators	107
	Indicators 81 to 88 Deck Address Ready	107
	Indicator 89 Transport Mechanically Ready	107
	Indicator 80 Tape Order Error	108
	Indicator 70 Write Unit Ready	109
	Indicator 72 Read Unit Ready... ..	109
	Indicator 74 Write Errors	109
	Indicator 75 Read Errors	109
	Indicator 76 End of Tape, Writing	110
	Indicator 77 Short Block, Reading	110
	Indicator 71 Write Master	110
	Indicator 73 Read Master	110
	Indicator 79 Writing Ring Present	111

Contents continued		Page
3.11.7	Program Interrupt Facility	112
	Reading Tape	112
	Writing Tape	113
3.11.8	Method of Recording on Tape	114
3.11.9	Parity Checking System	116
3.11.10	Flowcharts for Reading and Writing Tape ...	119
	Deck Testing Subroutine	119
	Tape Write Subroutine	121
	Write Exceptions Routine	123
	Tape Read Subroutine	125
	Read Exceptions Routine	127
3.11.11	Tape Statistics	128
3.11.12	Timing	129

Illustrations

Figure 11	The Card Reader	2
Figure 12	Contents of Register C during Card Reading ...	3
Figure 13	Card Punching Codes	3
Figure 14	An Example of a Card Read Subroutine	8
Figure 15	The Card Punch	14
Figure 16	Flowchart of a Punch Program	20
Figure 17	Table of Codes and Coded Zones for Printing ...	26
Figure 18	Character Arrangement on Print Barrel	26
Figure 19	Diagrammatic Representation of Relation between Register B and Print Barrel Assembly ...	27
Figure 20	Example of Print Data Distribution	31
Figure 21	Example of Sterling Printing	32
Figure 22	Flowchart of Program to Print One Line	34
Figure 23	Location of Row-binary for Printing in I.A.S. ...	37
Figure 24	Sprag Mechanism	39
Figure 25	Spacing Routine for Printer	41
Figure 26	Computer Interpretation of Paper-tape Code Punchings	46
Figure 27	Flowchart for Reading Paper Tape	49

Illustrations continued	Page
Figure 28 Computer Interpretation of Output to Paper Tape	50
Figure 29 Flowchart for Punching Paper Tape	53
Figure 30 Interrogating Typewriter: Keyboard Layout ...	55
Figure 31 Representation of Typewriter Codes in Computer	56
Figure 32 Operation of Function Keys	58
Figure 33 Operation of Function Keys	59
Figure 34 Read/Write Heads	62
Figure 35 Summary of Magnetic-tape Instructions, as held in the Computer	70
Figure 36 Summary of Magnetic-tape Indicators	76
Figure 37 Tape Read	78
Figure 38 Tape Write	78
Figure 39 Summary of Magnetic-tape Indicators	111
Figure 40 Tape Read	112
Figure 41 Tape Write	113
Figure 42 Quaternary Equivalents of Digits 0 to 15 showing Time Intervals between Flux Reversals..	115
Figure 43 Parity Checking	117

INTRODUCTION

3.1

The term 'Peripheral Equipment' is used here to define those machines which can be used with 1300-series Computers to facilitate the input or output of data.

All such equipment associated with the 1300-series Computer System is controlled by program. Any program instruction to control a unit of peripheral equipment (other than magnetic tape) is of the form:

F	A
38	00xy

where 38 is the function component that directs that the instruction applies to an input/output unit; and x and y are two numbers, each having a value from 0 to 9, that comprise the address component that specifies the operation to be executed by that unit. The various program instructions for the utilization of this equipment are explained in this section and indications are given of how to combine them into programs. There are, however, standard subroutines available for all units and it is recommended that these should be used. There is also a Print, Punch and Feed (P.P.F.) Program available that controls the programming for the printer, card punch and card reader on a time-sharing basis.

The various units of peripheral equipment are considered separately in sections. Each section is then further broken down into the following parts:

- (a) A general description of the unit,
- (b) instructions that can be given to the unit,
- (c) indicators associated with the unit,
- (d) programming,
- (e) timings.

With regard to item (d), it is impossible to give a set method of programming because of the variety of requirements that can arise. It is only possible to give some typical examples. Here again it is relevant to make mention of the existence of established subroutines, which are recommended for use. In the examples given, standard flowcharting symbols are used and reference is made to subroutines and modifications. These techniques are discussed in Part 4.

THE CARD READER

3.2

The card reader reads information punched in 80-column cards and stores it in the computer in a coded form. The cards are fed face down, column 80 leading and are read column by column by means of photo-electric reading stations (see Figure 11). The cards are read first by a reading station, and then by a check-reading station (which is situated one column behind the reading station) and the two readings compared automatically for checking purposes. The checked information is read into Register C from where it is stored by program in sets of six column readings.

Each character is represented in the computer by a zone component and a numeric component. A blank column is entered with zone and numeric components both zero. The zone and numeric components for each set of six columns are read successively into Register C, and are stored in I.A.S. by program before being overwritten in Register C by the contents of the next six columns. Figure 12 shows the successive sets of six card columns to occupy Register C, and the disposition of the zone component and the numeric component for each card column.

For the standard punching code, the coded zone component lies in the range 1 to 5, and the numeric component is the same as the numeric punching in the card. The standard code for punching is shown in Figure 13, together with the computer zone component. For alphabetic characters, which are represented by two holes in one column, the upper hole is translated as a number in the range 1 to 5 and the lower hole by its number.

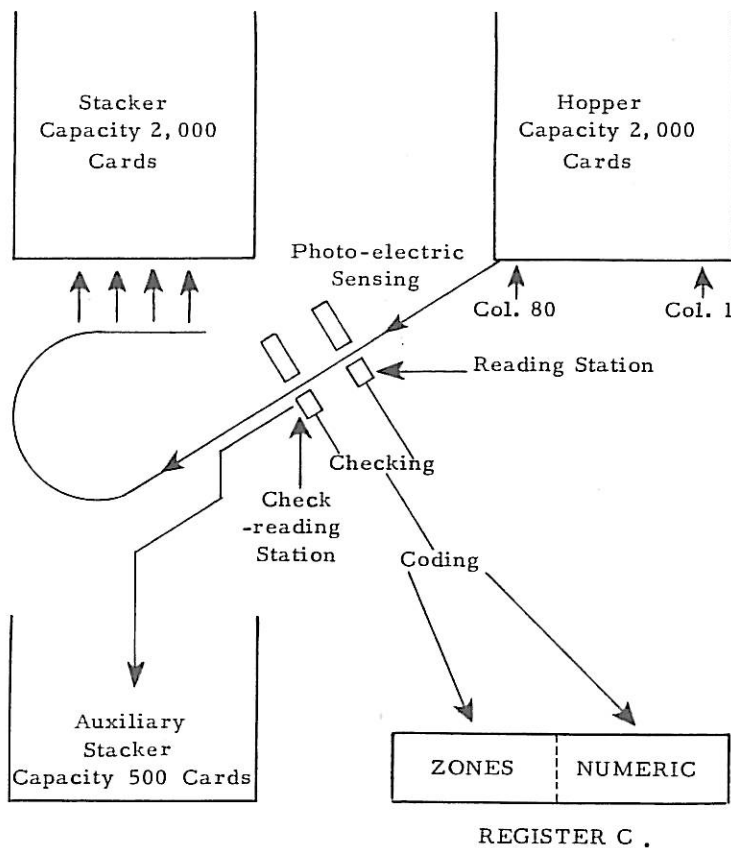


Figure 11: THE CARD READER

REGISTER C POSITIONS

1	2	3	4	5	6	7	8	9	10	11	12
75	76	77	78	79	80	75	76	77	78	79	80
69	70	71	72	73	74	69	70	71	72	73	74
63	64	65	66	67	68	63	64	65	66	67	68
57	58	59	60	61	62	57	58	59	60	61	62
51	52	53	54	55	56	51	52	53	54	55	56
45	46	47	48	49	50	45	46	47	48	49	50
39	40	41	42	43	44	39	40	41	42	43	44
33	34	35	36	37	38	33	34	35	36	37	38
27	28	29	30	31	32	27	28	29	30	31	32
21	22	23	24	25	26	21	22	23	24	25	26
15	16	17	18	19	20	15	16	17	18	19	20
9	10	11	12	13	14	9	10	11	12	13	14
3	4	5	6	7	8	3	4	5	6	7	8
*	*	*	*	1	2	*	*	*	*	1	2
Zone Component						Numeric Component					

Successive content of Register C during card reading, showing disposition of card columns. No assumption can be made about the contents of positions marked with an asterisk.

Figure 12: CONTENTS OF REGISTER C DURING CARD READING

Card Punching	Numeric No Overpunch	Numeric + 10 Overpunch	Numeric + 11 Overpunch	Numeric + 0 Overpunch	Numeric + 1 Overpunch
10	10				
11	11				
0	0				
1	1	A	J	&	
2	2	B	K	S	%
3	3	C	L	T	$\frac{1}{4}$
4	4	D	M	U	-
5	5	E	N	V	/
6	6	F	O	W	$\frac{1}{2}$
7	7	G	P	X	.
8	8	H	Q	Y	@
9	9	I	R	Z	$\frac{3}{4}$
Coded Zone Component	1	2	3	4	5

Figure 13: CARD PUNCHING CODES

Example Suppose that columns 75 to 79 of a card are punched with the characters 10, $\frac{1}{2}$, A, N and Z respectively and that column 80 is blank. The columns will be punched as follows:

Column	75	76	77	78	79	80
Punch Position(s)	10	1 & 6	10 & 1	11 & 5	0 & 9	Blank

Immediately after columns 75 to 80 have been read at the reading station, Register C will be as follows:

Card Column	Zone Components						Numeric Components					
	75	76	77	78	79	80	75	76	77	78	79	80
Contents of Register C.	1	5	2	3	4	0	10	6	1	5	9	0
Digit Positions	1	2	3	4	5	6	7	8	9	10	11	12

Time available for Programming

3.2.1

The card reader is cyclic in operation, one cycle being the time taken to feed and read one card. If the instruction to feed a card is not received within a certain time after the last six columns of the previous card have been read, then a cycle is missed. The card reader unlatches and a full card cycle elapses before the next card can be fed. It follows that the time between successive cards being read is a multiple of the cycle time and that if alternate cycles are consistently missed, the card reader is reduced to half its maximum speed.

Once the instruction has been given for a card to be fed, it is automatically moved through the card reader and the information from the card transferred into Register C. The only program instructions required are those to store Register C at six-column intervals. During the card cycle, therefore, there is a considerable amount of time for which the reader requires no program control and it is possible to use this time to obey other program instructions. This process is known as time-sharing and can considerably speed up the overall time required for processing a job.

Time is available for programming between giving the instruction to feed a card and reading the first six columns; and between reading successive sets of six columns. The actual cycle times and time available for programming are given later in this section. Care should be taken to ensure that the program does not take longer than the time available.

Card Read Instructions

3.2.2

Instruction 380002

Effect This instruction will initiate the feeding of a single card. After the card has been read, card feed automatically stops.

Operation When a 380002 instruction is given, a card is fed into the reader and moved through it. If the instruction does not follow within a given time from the last six columns of the previous card having been read, then the card reader unlatches and a full card cycle elapses before the next card is fed.

Instruction 380007

Effect This instruction causes a card to be ejected into the reject stacker.

Notes Instruction 380007 is provided to enable the program to cause a card to be ejected to the reject stacker if an error arises. For example, a card may be rejected if that card slips during feeding, thus causing a card read error; or if a programmed sequence check fails, thus indicating that the cards are out of order. A 380007 instruction cannot be given until after the first six columns are read, and there is a time limit after which it cannot be given.

Card Read Indicators

3.2.3

Indicator 35 Card Reader Ready

Purpose This indicator is set when the card reader is in a ready condition.

Operation Indicator 35 is unset when a 380002 instruction has been given, but the card has not yet moved far enough for reading to commence. It is also unset when any of the following external conditions arise to cause the reader to be unready (i.e. the Card Reader Interlock is set):

- (a) The Emergency Stop button has been operated,
- (b) the feed hopper is empty,
- (c) there are less than 350 cards in the hopper and the card weight has been removed,
- (d) the main stacker is full,
- (e) the reject stacker is full,
- (f) the power supply to the card reader has been cut off,
- (g) a card jam or mis-feed has occurred.

This indicator remains unset until the condition has been remedied and the Start and Reset manual control has been operated.

Notes Indicator 35 should be tested by program prior to giving a 380002 instruction to ensure that none of the above conditions has arisen.

The 380002 instruction unsets indicator 35 and it remains unset until shortly before the first six columns are read, when it is set to indicate to the program that reading is about to commence.

At this stage the indicator is set even if external conditions have arisen which would normally cause it to be unset.

At any stage during the reading of a card another card may be called; indicator 35 will then be unset and remain so until reading is about to commence on the second card.

It is recommended that indicator 35 should be tested again before preparing to read a called card. This will ensure that the card is in the correct position for reading and also that indicators 37 (6 Columns Missed), and 38 (Mischeck) have been unset.

Indicator 36 6 Columns Read

Purpose This indicator is set when six columns have been read into Register C, and indicates to the program that information is ready to be stored.

Operation After the data contained in a set of six card columns has been read into Register C, the register is full. It is required that the contents of this register be transferred to I.A.S. before being overwritten by the data from the succeeding six card columns. When indicator 36 is set, after each six columns is read, it shows that Register C is full and ready for its contents to be transferred to store.

Indicator 36 is unset when tested by program. If the indicator is not tested (and therefore not unset) by program it is unset automatically after a fixed time interval.

Indicator 37 6 Columns Missed

Purpose This indicator is set when the contents of Register C are overwritten without having been transferred to store.

Operation Indicator 37 is unset for each card cycle just before indicator 35 is reset after a 380002 instruction. If indicator 36 is unset automatically, then indicator 37 is set. It follows that if indicator 37 is set, then the program did not test indicator 36 before it was automatically unset; therefore the contents of Register C were not stored before they were overwritten, i.e. six columns have been missed. Indicator 37 is unset when tested by program.

Indicator 38 Mischeck

Purpose This indicator is set if the reading station and the check-reading station make an unsuccessful comparison of data read from one card column.

Operation Indicator 38 is unset for each card cycle just before indicator 35 is reset after a 380002 instruction. Indicator 38 is set if the reading station and the check-reading station have made an unsuccessful comparison; it remains set until tested by program, when it is unset.

Notes Indicator 38 should be tested when the last six columns of a card have been read, to check that a card has been read correctly. Thus, this indicator must be tested before it is unset at the next card cycle and the mischeck indication lost.

If a programmer is using information on the last columns of a card only, it is permissible to start the main program when the required information has been stored, even though the columns will still have to be read into Register C. If this is done, however, it is advisable to wait until the 6 Columns Read indicator is set for the next six columns and then test indicator 38. If this is not done an error may not be detected. However, this technique is not advised as mistakes are often made and the saving in time is small. In particular, care must be taken to ensure that multiplication does not take place before card reading has finished, since both operations make use of Register C.

A Card Read Program

3.2.4

Figure 14 overleaf shows a subroutine to read one card and demonstrates the sequence in which the card read instructions and indicators are used.

This subroutine allows time-sharing between calling a card and reading the first six columns.

On the first entry indicator 19 must be unset. After the card has been called, control is restored to the main program.

On the second entry the card is read and its contents stored in I.A.S. under relativizer 3 as follows:

I.A.S. Word Relativizer 3	1	2	3	4	5	6	7	8	9	10	11	12
0	*	*	*	*	1	2	*	*	*	*	1	2
1	3	4	5	6	7	8	3	4	5	6	7	8
2	9	10	11	12	13	14	9	10	11	12	13	14
3	15	16	17	18	19	20	15	16	17	18	19	20
4	21	22	23	24	25	26	21	22	23	24	25	26
5	27	28	29	30	31	32	27	28	29	30	31	32
6	33	34	35	36	37	38	33	34	35	36	37	38
7	39	40	41	42	43	44	39	40	41	42	43	44
8	45	46	47	48	49	50	45	46	47	48	49	50
9	51	52	53	54	55	56	51	52	53	54	55	56
10	57	58	59	60	61	62	57	58	59	60	61	62
11	63	64	65	66	67	68	63	64	65	66	67	68
12	69	70	71	72	73	74	69	70	71	72	73	74
13	75	76	77	78	79	80	75	76	77	78	79	80
	Zone Component						Numeric Component					

1300 SERIES PROGRAM SHEET		JOB <i>SUBROUTINE TO READ 1 CARD - PARTIAL TIME-SHARING</i>				BLOCK No. —	
PROGRAMMER:-					SHEET No 1/2		
C	I	D	F	A	R	NARRATIVE	
1		B					
2	0		41 00	11	B	Store Link	
	1	4	35	2	B	Test Card Reader Ready	
		4	00	1	B		
2	2	4	19	4	B	1st Entry - Unset; 2nd Entry - Set	
		38	2			Call Card Feed	
3	3	8	19			Set indicator 19	
		4	00	11	B	Jump to Link	
	4		45	15	B	Transfer basic instruction	
02			7	B	and counter		
5	5	4	36	7	B	Test 6 Columns Read	
		4	37	12	B	Test 6 Columns Missed	
4	6	4	00	5	B		
		7	67	7	B	Store Register C	
			43	13	3		in I.A.S.
5	8	00				Counter	
		00	14				
		9	67	8	B	Subtract 1 from counter	
5	9	4	02	5	B		
		10	4	38	12	B	Test Mismatch indicator
			9	19			Unset indicator 19 for next entry
6	11			0		Link	
		12	38	7		Reject card if error	
			11	332		Stop	
6	13	4	35	14	B	Restart: Test card reader ready	
		4	00	13	B		
	14	38	2		Call Card Feed		
		4	00	1	B		

Figure 14: AN EXAMPLE OF A CARD READ SUBROUTINE

1300 SERIES PROGRAM SHEET			JOB <i>SUBROUTINE TO READ 1 CARD - PARTIAL TIME-SHARING</i>				BLOCK No <i>-</i>
			PROGRAMMER: <i>/ /</i>				SHEET No <i>2/2</i>
C	I	D	F	A	R	NARRATIVE	
<i>7 (y)</i>	<i>15</i>		<i>67 43</i>	<i>7 13</i>	<i>B 3</i>	<i>Basic transfer instruction</i>	
	<i>16</i>			<i>14</i>		<i>Constant for setting counter</i>	
	<i>17</i>						

Figure 14 continued

If six columns are missed or if an error occurs on reading, the card is rejected and the computer stops with 110333 displayed in CR3. A re-start program is included after the error stop. The rejected card should be replaced at the bottom of the pack in the hopper. When the Start button is pressed the card will be re-read.

11-11-11

11-11-11

Timings

3.2.5

Two types of card reader are available with the 1300-series Systems. These are identical in specification except that they operate at different speeds. The speeds and timings associated with the card reader are shown for both types in the following table.

Maximum speed at which cards can be read	600 cards a minute	300 cards a minute
	(minimum times)	(minimum times)
Time for one complete card cycle	100 ms	200 ms
Time after 14th <i>6 Columns Read</i> in which a Call Card Instruction may be given to maintain continuous running of the reader.	2.44 ms	7.3 ms
Time between calling a card and the first <i>6 Columns Read</i> if the card reader has been unlatched. Register C comes into use for card reading before the first <i>6 Columns Read</i> .	36.1 ms	70.4 ms
Therefore if multiplication is taking place this should be reduced to:	32.4 ms	60.75 ms
An instruction to call the next card may be given any time during the reading of the current card. When the reader is running continuously the time between the 14th <i>6 Columns Read</i> and the 1st <i>6 Columns Read</i> of the next card is:	40 ms	80 ms
If multiplication is taking place this should be reduced to:	35.4 ms	70.8 ms
Interval between successive <i>6 Columns Read</i> . Multiplication cannot be carried out during this time.	3.18 ms	6.36 ms
Time after 14th <i>6 Columns Read</i> during which a Reject Card instruction can be given.	25.4 ms	50.8 ms
Time after which <i>6 Columns Read</i> indicator (36) is automatically unset if not previously unset by program.	538 μ s	1076 μ s



THE CARD PUNCH

3.3

The card punch receives information from the computer, and punches it in 80-column cards. The cards are fed face down, 10-edge leading. Punching is effected by a single row of 80 magnetically-actuated punch knives, punching in each card row in sequence; 10, 11, 0, 1 9. As each row comes under the knives, the computer sends an impulse to the punch magnets for each column required to be punched with that digit value. The punch magnets are in two groups having separate controls: the Left group effecting punching in columns 1 to 40, and the Right group effecting punching in columns 41 to 80.

Information is sent to the punch from Register B in two transfers of 40 bits (row-binary); 0 signifying *do not punch* and 1 signifying *punch*. Synchronization between the computer and the punch is achieved by means of timing signals from the punch, and their effect on certain indicators (54 to 58) within the computer. This is explained under 3.3.3 Card Punch Indicators.

Checking of Punching

3.3.1

While one card is being punched the previous card is check-read at a brush sensing station. The checking of punching is effected by reading the data in the card just punched, entering it back into the computer and comparing it by program with the I.A.S. information from which the card was punched. The data is read back from the card by a set of 80 brushes, one card cycle behind the punch magnets. The data is read for each row in turn, and is entered into Register B in row-binary form immediately after the corresponding row of the following card has been punched. The contents of Register B are then compared by program with the row-binary information, stored in I.A.S., from which the punching was derived. The punch can be instructed so that in the event of an error being detected it will offset the error card about half an inch lengthways as it is fed to the stacker.

The normal error procedure is as follows: both the cards being processed when the check fails, i.e. the one passing the punch brushes, and the next one, i.e. the one being punched, will be offset in the stacker by program. An attempt will then be made to repunch the same data in the next two cards, and check the punching. If the check fails again this process is repeated. The reason for offsetting two cards and then repunching both, when an error is detected in the first card, is that this method enables the offset cards to be removed without affecting the order of the cards. If only the error card were to be offset and repunched after the card passed the punch magnets, the order of the error card and the following card would have to be reversed on extraction from the stacker.

The relation between the punch magnets, punch brushes, and cards is shown diagrammatically in Figure 15.

It should be noted that when an unsuccessful comparison is made, an error may have occurred either in the punching or in the reading back into the computer; a card which has been offset will not necessarily be wrongly punched.

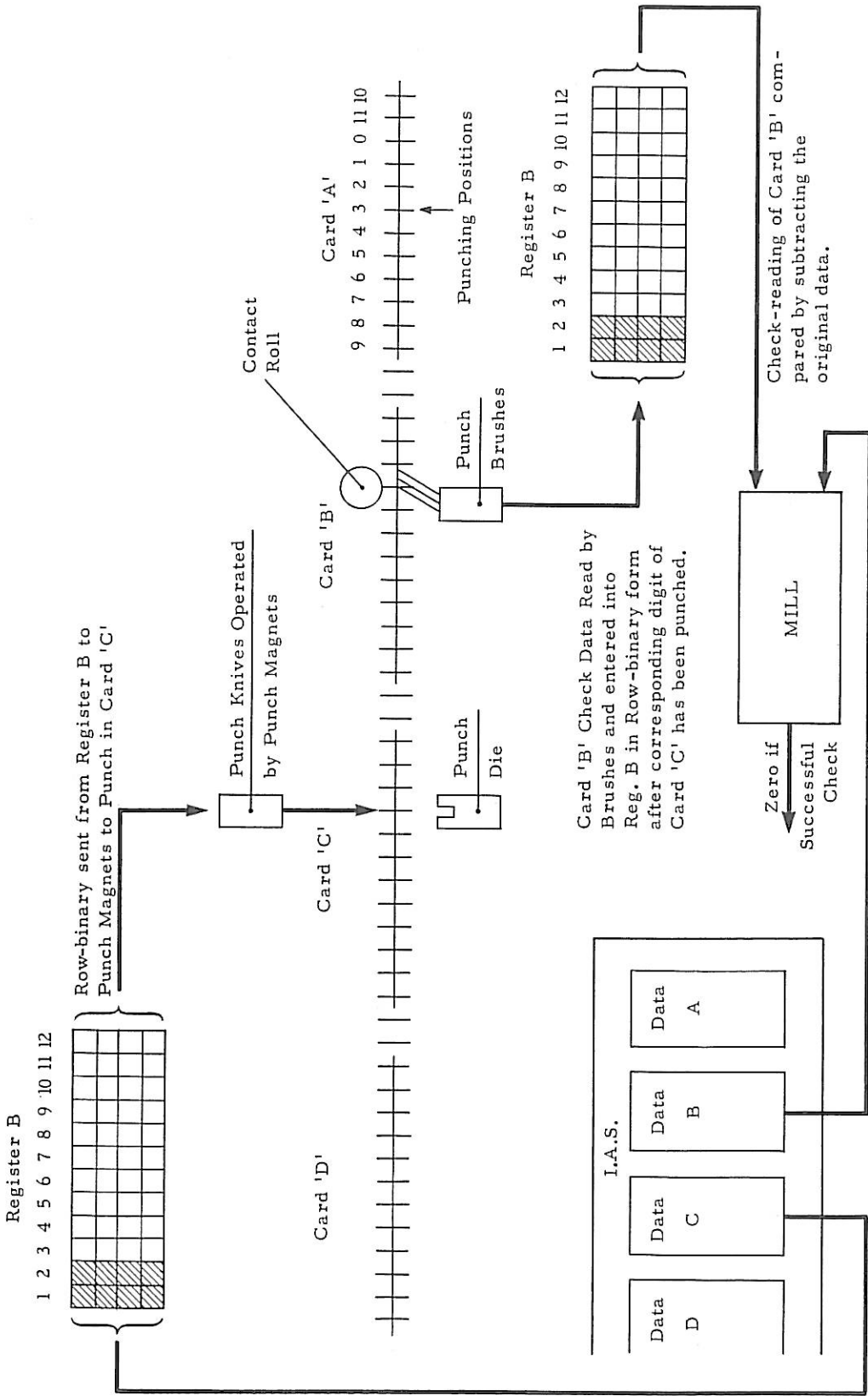


Figure 15: THE CARD PUNCH

Instruction 380042

Effect A 380042 instruction will cause the card feed to operate for one card cycle.

Operation On receipt of the instruction one card will be fed into position under the punch knives, and each card in the card track will move one card cycle towards the stacker. If a 380042 instruction is not given within a certain time after the previous one, the punch motor is automatically switched off. The motor is switched on again automatically when the next 380042 instruction is given.

Instruction 380043

Effect Instruction 380043 causes punching to take place on the Left group of punch knives (columns 1 to 40).

Operation When a 380043 instruction is given the contents of Register B will cause punching to take place in columns 1 to 40. The presence of a 1-bit in any position of Register B will cause the associated punch magnet to be energized. The bit positions of Register B will be associated with punch magnets as shown below.

1	2	3	4	5	6	7	8	9	10	11	12	
		1	2	3	4	5	6	7	8	9	10	1
		11	12	13	14	15	16	17	18	19	20	2
		21	22	23	24	25	26	27	28	29	30	4
		31	32	33	34	35	36	37	38	39	40	8

Instruction 380044

Effect Instruction 380044 causes punching to take place on the Right group of punch knives (columns 41 to 80).

Operation When a 380044 instruction is given, the contents of Register B will cause punching to take place in columns 41 to 80. The presence of a 1-bit in any position of Register B will cause the associated punch magnet to be energized. The bit positions of Register B will be associated with the punch magnets as shown as follows:

1	2	3	4	5	6	7	8	9	10	11	12	
		41	42	43	44	45	46	47	48	49	50	1
		51	52	53	54	55	56	57	58	59	60	2
		61	62	63	64	65	66	67	68	69	70	4
		71	72	73	74	75	76	77	78	79	80	8

Instruction 380045

Effect Instruction 380045 causes one row of data punched in columns 1 to 40 of the card passing the punch brushes to enter Register B for comparison with the originating data.

Operation When a 380045 instruction is given the data holes punched in one row, of columns 1 to 40, are sensed by the punch brushes, and the information is entered into Register B in row-binary form and with the same column-to-bit position pattern as that for instruction 380043.

Notes It is essential to clear Register B before using this instruction.

Instruction 380046

Effect Instruction 380046 causes one row of data punched in columns 41 to 80 of the card passing the punch brushes to enter Register B for comparison with the originating data.

Operation When a 380046 instruction is given, the data holes punched in one row of columns 41 to 80, are sensed by the punch brushes, and the information is entered into Register B in row-binary form and with the same column-to-bit position pattern as that for instruction 380044.

Notes It is essential to clear Register B before using this instruction.

Instruction 380047

Effect Instruction 380047 causes a selected card to be offset lengthways in the stacker by about half an inch.

Operation When a 380047 instruction is given, the card that has just left the punch brushes will be offset when it is fed to the stacker. This instruction must be given before the next 380042 instruction otherwise the wrong card will be offset. The instruction is normally given immediately after the checking of the punch data.

There is a visual indicator above the stacker which glows when one or more cards in the stacker have been offset.

Notes When, during checking of punching, the check-read data and the original data do not correspond, the card at fault can be offset in the stacker by giving a 380047 instruction. Alternatively, the instruction can be used to offset a card if indicator 57 or 58 is tested and found to be set (see Section 3.3.3).

Indicator 54 Punch Ready

Purpose Indicator 54 is set when the punch is available for use.

Operation Indicator 54 is unset by the Card Punch Interlock, and automatically by a 380042 instruction. The Card Punch Interlock will cause the indicator to be unset when any of the following conditions arise:

- (a) The hopper is empty,
- (b) the stacker is full,
- (c) a card jam has occurred,
- (d) a misfeed has occurred,
- (e) the Emergency Stop manual control has been operated,
- (f) the power supply fails,
- (g) the die is removed,
- (h) the knock-off bar is removed.

The indicator is automatically set at the end of each card cycle.

Indicator 55 Punch Index Point Time

Purpose Indicator 55 is automatically set as each row of punching positions of a card is positioned under the punch knives ready for punching.

Operation As each row is positioned under the punch knives, indicator 55 is automatically set to indicate that the punch magnets are ready to receive the information in Register B in row-binary form. Indicator 55 is unset when tested by program. If the indicator is not unset (by testing) within a certain time, it is unset automatically.

Indicator 56 Check Index Point Time

Purpose Indicator 56 is automatically set as each row of punching positions is positioned over the brushes ready for checking.

Operation As each row is positioned over the brushes, indicator 56 is automatically set to indicate that the checking station is ready to enter data in row-binary form into Register B for checking purposes. Indicator 56 is automatically set a short time after indicator 55 is unset. It is unset when tested by program, or automatically after a certain time if not tested.

Indicator 57 Punch Index Point Time Missed

Purpose Indicator 57 is set in the event of indicator 55 being unset automatically (i.e. it has not been tested by program) indicating that it is too late to punch the associated row.

Operation If within a certain time of being set, indicator 55 is not tested and unset by program, it is unset automatically and indicator 57 is set. This will normally only happen if the program has consumed so much time since indicator 55 was last tested and found to be set, that indicator 55 has been set and unset automatically so that it is too late to punch the associated row. Indicator 57 is unset when tested by program.

Notes The testing of indicator 57 will permit the appropriate error routine to be brought into operation if the index point time is missed. If this indicator is not tested, and an index point time has been missed, the computer will go into a continuous loop after punching the last row. This is because the index point counter has not been reduced to zero by the time the last row has been punched but holds the number of index point rows missed. Since the counter does not show zero the computer goes into a continuous loop, testing indicator 55 which remains unset.

Indicator 58 Check Index Point Time Missed

Purpose Indicator 58 is set in the event of indicator 56 being unset automatically (i.e. it has not been tested by program) indicating that it is too late to check the associated row of punching positions for correct punching.

Operation Except that it is associated with indicator 56, and is concerned with checking instead of punching, indicator 58 corresponds to indicator 57 in operation and use.

Distribution of Punch Data

3.3.4

Before punching information in a card it must first be distributed into words of ten digits, each occupying digit positions 3 to 12 of an I.A.S. location. Successive I.A.S. words represent columns 1 to 10, 11 to 20, 21 to 30 and so on for both zone and numeric components. For each group of 40 punch magnets the data occupy four words with zone components and another four with numeric components. Thus if both groups are to be used, 16 words are required for each card to be punched.

Example

Zone components for card columns	Stored in positions 3 to 12 of I.A.S. locations	Numeric components for card columns	Stored in positions 3 to 12 of I.A.S. locations
1 to 10	200	1 to 10	208
11 to 20	201	11 to 20	209
21 to 30	202	21 to 30	210
'	'	'	'
'	'	'	'
71 to 80	207	71 to 80	215

In this example, position 3 of I.A.S. word 200 and position 3 of I.A.S. word 208 contain the zone and numeric component respectively for data to be punched in column 1.

The suppression of any non-significant zeros should be arranged at this stage. If the punching of non-significant zeros to the left of a number or value is to be suppressed, the position of the most-significant digit must be determined to permit the insertion of the necessary zone components in the words containing the zoning for output, i.e. zeros which are to be punched should be given a zone component of 1 and zeros which are to be suppressed should be given a zone component of 0.

The next step is to convert the information into row-binary form for each of the 12 punching positions. It must be remembered that punching positions 10 will be required to be punched, not only as a numeric digit of 10 (as in 10d.) but also as a zone component of 2 where alpha contents of this zone are concerned. Hence, it is necessary during row-binarizing to combine data for numeric 10 with that for zone 2 into the same row-binary word by use of Logical OR. Similar treatment is required for combining numeric 11 with zone 3, numeric 0 with zone 4 and numeric 1 with zone 5. On completion of the row-binary routine, the data, both zone and numeric combined, comprises 12 words in punching position order for each punch magnet group.

Flowchart of a Punch Program

3.3.5

The flowchart in Figure 16 represents a program to punch and check data which have already been distributed and row-binarized. The flowchart is included here to illustrate the use of the instructions and indicators associated with punching and the action to be taken if an error is detected.

Each card requires 24 words of row-binary, the first 12 containing data to be punched in columns 1 to 40 in index point order 10, 11, 0 and 1 8, 9, the second 12 containing data for columns 41 to 80.

A further 24 words are required which contain data for checking the last card punched in the previous entry to the program. For the first entry into this routine at the beginning of a job run, these 24 words should contain all $\bar{1}$ s since with no card at the check station, check-reading will give a word of $\bar{1}$ s. In order that the last card should be checked, an extra entry should be made to the subroutine to punch a blank card after the last data card.

Programmed indicators 11 and 12 are used with the program only; their state on entering the program is immaterial.

The following error actions are taken:-

- (a) On failure of the programmed check, punch or check index point missed, the card being checked and the subsequent card are both offset in the stacker, the second card not being checked. Both cards are then repunched in the correct order.
- (b) On a second check failure occurring for the card which previously failed the check, the machine will stop. Programmed indicator 12 will then remain set and the corresponding visual indicator will be lit.

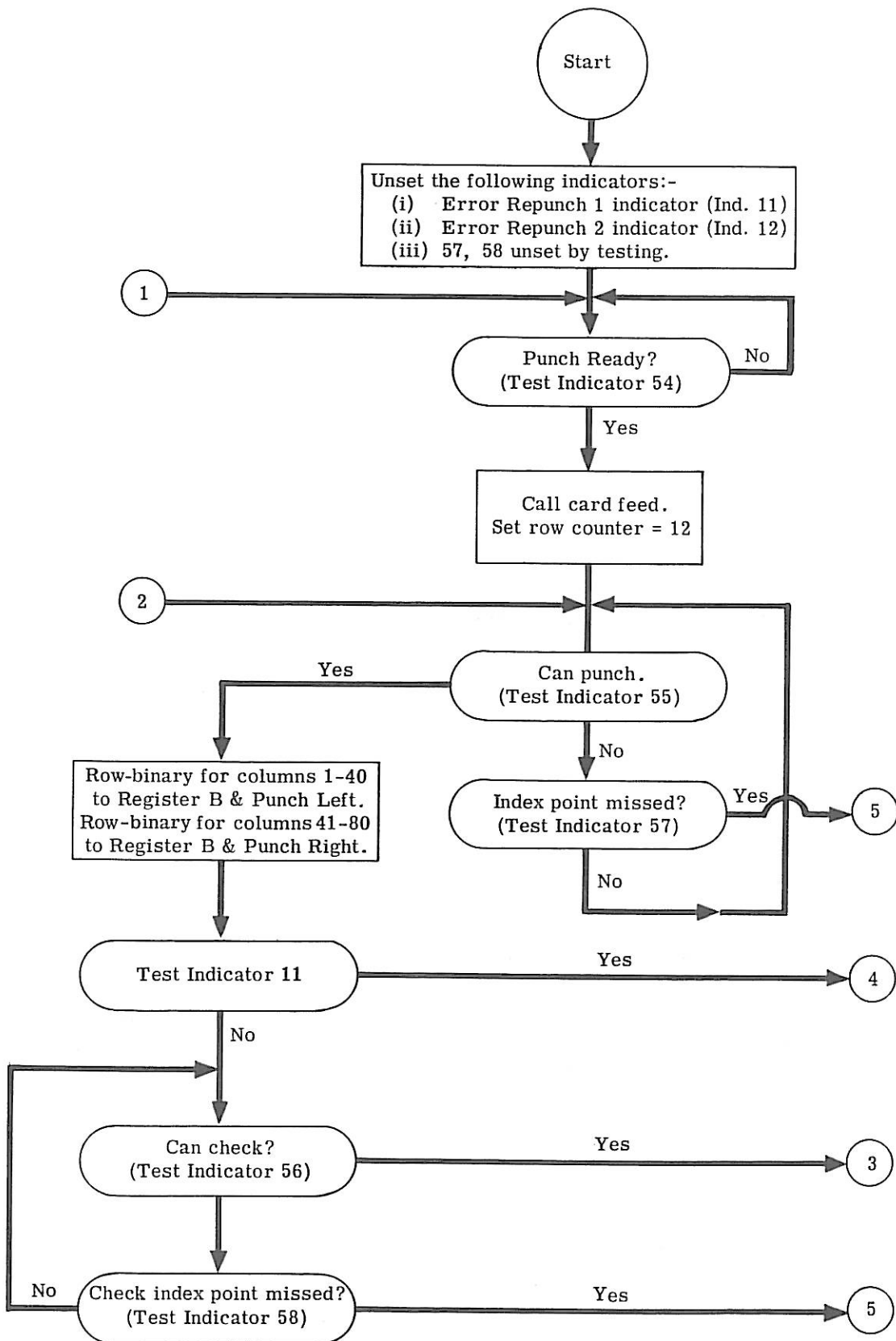


Figure 16: FLOWCHART OF A PUNCH PROGRAM

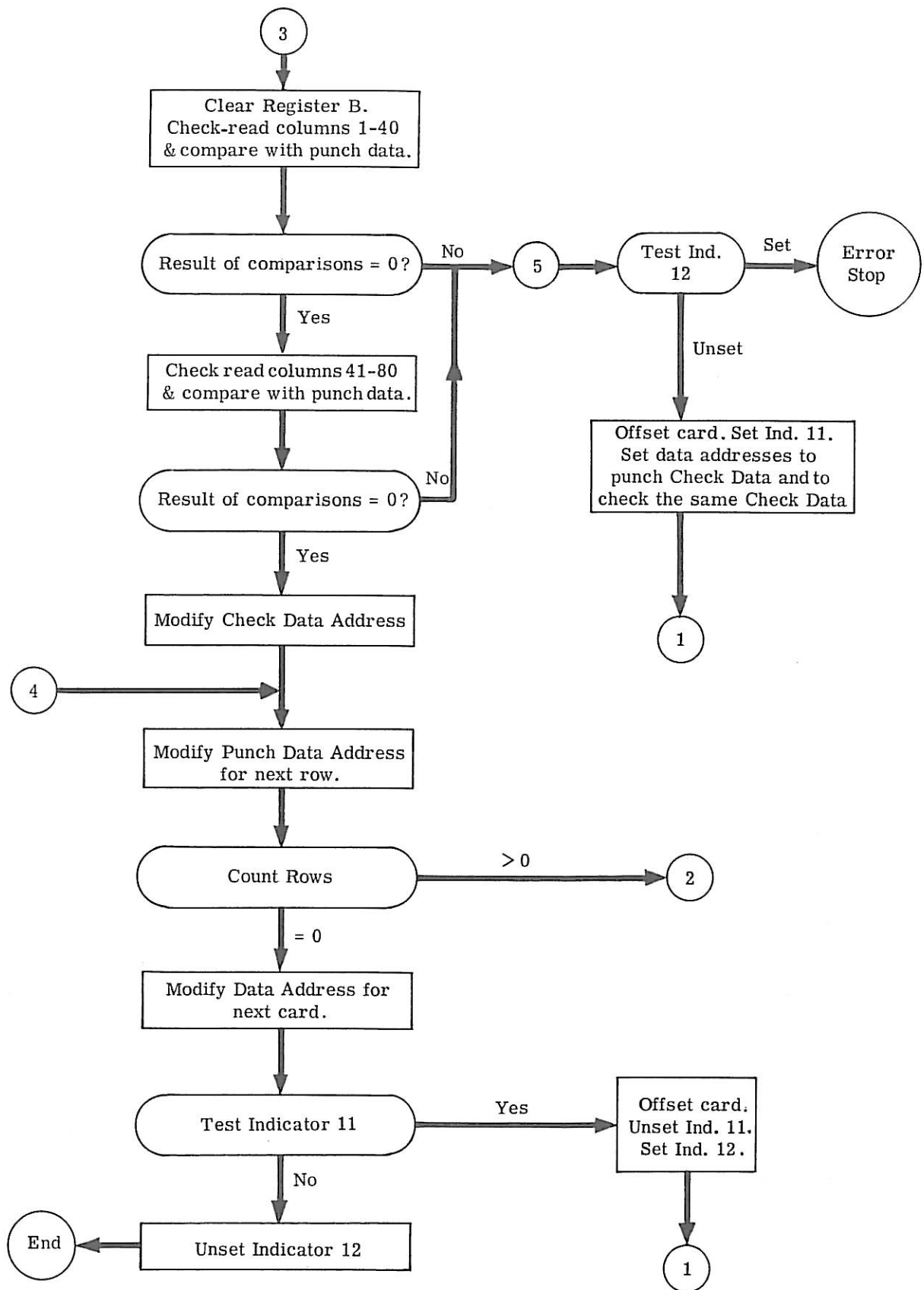


Figure 16: continued



OPERATION	MINIMUM TIME OF OPERATION
Card Punching rate	100 cards a minute
Card Cycle	600 ms
Calling Punch from rest, with motor running, to 1st Punch Index Point Time on.	35.83 ms
Index Point Interval	40.7 ms
Punch Index Point Time indicator (55) duration	6.52 ms
Check Index Point Time indicator (56) duration	14.66 ms
Interval between last setting of indicator 56 to last time to give 380042 instruction for continuous running.	58.22 ms
Interval between last setting of indicator 56 and 1st setting of indicator 55 when feeding continuously	102.2 ms
Time between last setting of indicator 56 and last time to give 380047 instruction	41.94 ms



THE LINE PRINTER

3.4

The line printer basically comprises a print unit, stationery feed mechanism and control logic.

The print unit consists of a print barrel and a number of associated print hammers. The print barrel, with the type characters embossed on it, rotates at high speed. The print hammers, under the control of the computer, strike the stationery momentarily against the print barrel through an inked ribbon located between the stationery and the barrel.

The stationery feed mechanism comprises a tractor-driven, continuous-stationery feeding device which can feed up to four parts of stationery. A reversible carbon ribbon is fitted for the top copy. It must be noted that the stationery and carbon paper employed must conform to specifications that have been laid down. The paper movement (i.e. spacing and throwing) is controlled by a sprag and ratchet wheel mechanism which is operated by control signals generated by the computer program.

The Print Unit

3.4.1

Print Barrel The print barrel has 120 positions and each print position comprises 50 type characters embossed round the barrel. The lateral spacing of printing is ten characters to the inch and the vertical spacing is six characters to the inch.

At every character position in the barrel's position, the computer emits an impulse to the printer for each print position at which a character must be printed.

A line printer is available which has only 80 print positions. Throughout this section however, reference is made only to 120 print positions with the appropriate short qualifying note where necessary.

Print Characters The 50 print characters comprising one print position are arranged round the print barrel in ten sectors with five characters in each sector. The ten sectors and the arrangement of characters within these sectors correspond to the I.C.T. 5-zone card code. This code is shown in Figure 17 and the arrangement of the characters on the barrel is shown in Figure 18. Comparing these two diagrams, it can be seen that the ten sectors correspond to the ten numeric index positions (0 to 9) in the card code and the five characters within these sectors correspond to the five zone positions. For example, $\frac{3}{4}$, Z, R, I, 9 in the sector for index point 9 correspond to numeric 9 zone 5, numeric 9 zone 4, numeric 9 zone 3.....numeric 9 zone 1 respectively. It should be noted that there are only ten index positions and positions 10 and 11 are represented as zone 3-numeric 0 and zone 2-numeric 0 respectively.

Card Code		Zone				
		Numeric	10	11	0	1
Computer Code		1	2	3	4	5
Index Point	0	0	11	10	*	£
	1	1	A	J	&	¢
	2	2	B	K	S	%
	3	3	C	L	T	$\frac{1}{4}$
	4	4	D	M	U	-
	5	5	E	N	V	/
	6	6	F	O	W	$\frac{1}{2}$
	7	7	G	P	X	.
	8	8	H	Q	Y	@
	9	9	I	R	Z	$\frac{3}{4}$

Figure 17: TABLE OF CODES AND CODED ZONES FOR PRINTING

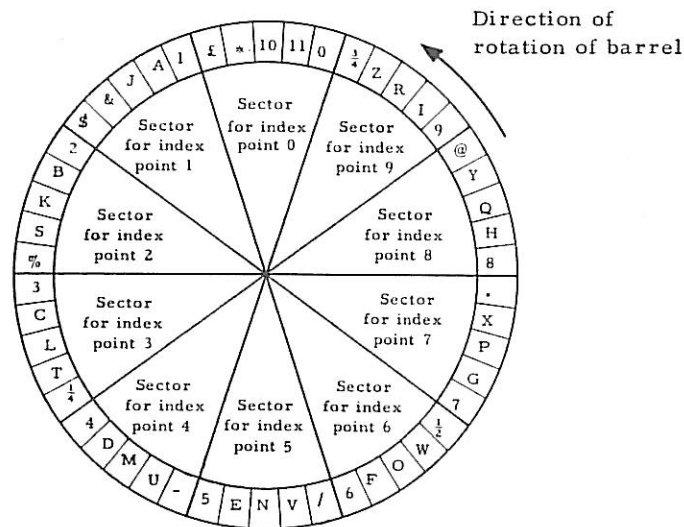


Figure 18: CHARACTER ARRANGEMENT ON PRINT BARREL

Print Hammers There are 120 print hammers fitted, i.e. one for each print position. The 120 print hammers are grouped into three banks of 40 hammers in each bank. The print hammers strike the stationary against the print barrel when the character to be printed is opposite the print hammers. Printing is achieved on each of the three banks of 40 print hammers in turn. This entails up to three transfers (one per print bank) of 40 bits from Register B to activate the print hammers. Each bit will either be 0 or 1; 0 means *do not print* (i.e. do not activate print hammer) and 1 means *print* (i.e. activate print hammer). The 80 print position printer has 80 print hammers arranged in two banks of 40.

Printing Figure 19 shows the relationship between the print unit and Register B.

Data to be printed, that is the zone and numeric components of each character, must be row-binarized (see 2.7) so that it is in the 0 and 1-bit form described above. During printing, positions 3 to 12 of Register B must hold data in row-binarized form. The contents of Register B are transferred to one of the print banks according to the program instruction. Thus the essential

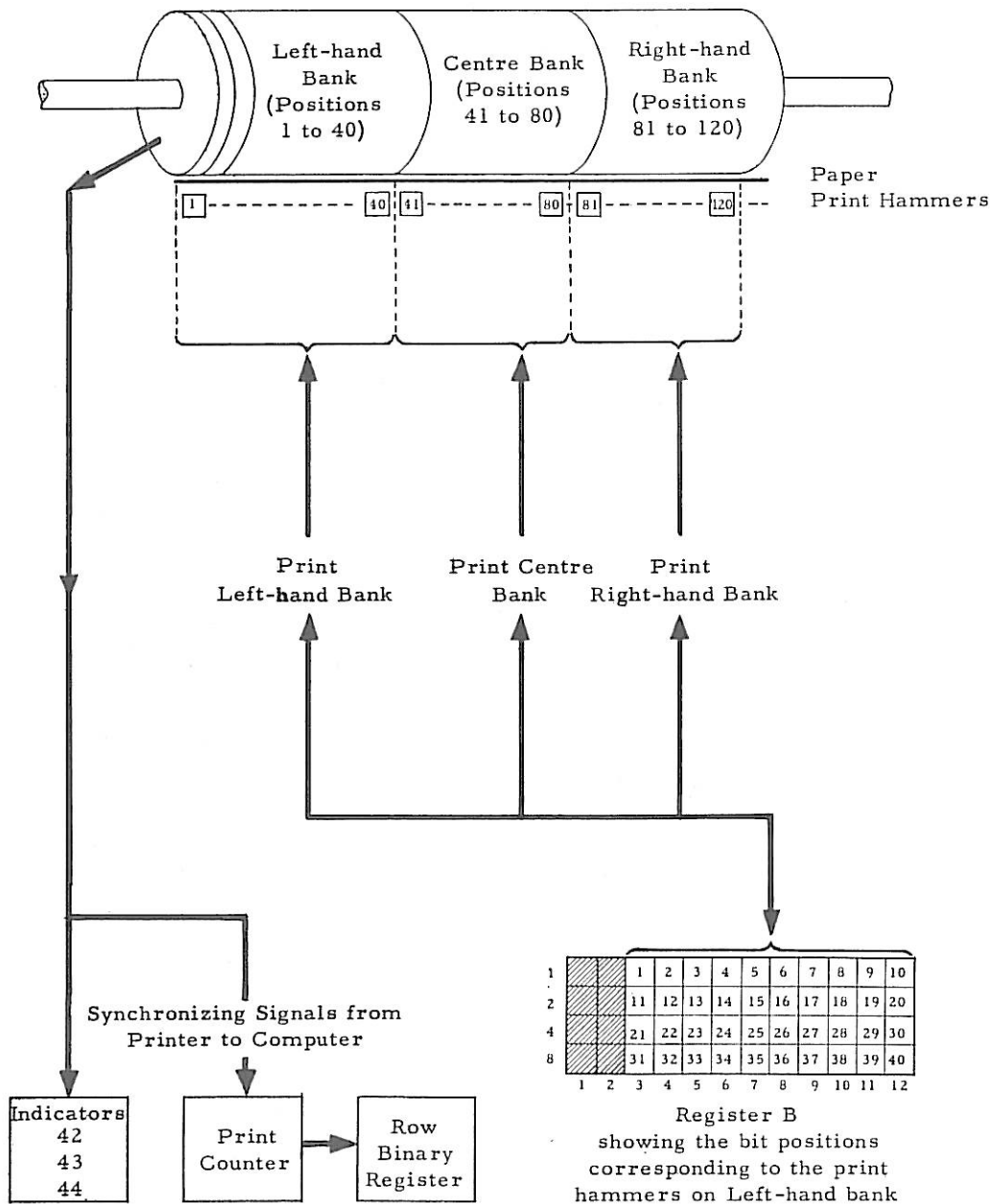


Figure 19: DIAGRAMMATIC REPRESENTATION OF RELATION BETWEEN REGISTER B AND PRINT BARREL ASSEMBLY

part of a printing program is the creation of row-binary, which takes place in two stages: row-binarizing the zone components of print characters and row-binarizing the numeric components of print characters. The zones and numerics are then combined for each character by the Logical AND instruction.

Synchronizing signals are emitted from the printer to the computer and have two main functions: they set various indicators and they set the print counter. The print counter registers which sector of the print barrel is opposite the print hammers ready for printing. This counter is a subtraction counter which counts down from 9 to 0 in steps of 1 and is then reset to 9 at each complete revolution of the print barrel; thus the figure held in this counter is equal to the sector (index point) which is at that moment ready for printing. When a sector is available for printing, the numeric components required for this sector must be row-binarized. To enable this to be done, an instruction can be given which will cause the contents of the print counter to be transferred to the Row Binary Register.

Print Instructions

3.4.2

Instruction 380013

Effect A 380013 instruction will cause the contents of the print counter to set the Row Binary Register.

Operation The Print Counter will register which sector is available for printing. Thus row-binary may be formed for the numeric components prior to printing.

Instruction 380014

Effect A 380014 instruction will cause printing to take place on the left-hand print bank.

Operation When a 380014 instruction is given, the contents of Register B will cause printing to take place on print positions 1 to 40. The presence of a 1-bit in any position of Register B will cause the associated print hammer to be actuated. The bit positions of Register B will be associated with print hammers 1 to 40 as shown below:

1	2	3	4	5	6	7	8	9	10	11	12	
		1	2	3	4	5	6	7	8	9	10	1
		11	12	13	14	15	16	17	18	19	20	2
		21	22	23	24	25	26	27	28	29	30	4
		31	32	33	34	35	36	37	38	39	40	8

Instruction 380015

Effect A 380015 instruction will cause printing to take place on the centre print bank.

Operation When a 380015 instruction is given the contents of Register B will cause printing to take place on print positions 41 to 80. The bit positions in Register B will be associated with print hammers 41 to 80 as shown below:

1	2	3	4	5	6	7	8	9	10	11	12	
///	///	41	42	43	44	45	46	47	48	49	50	1
///	///	51	52	53	54	55	56	57	58	59	60	2
///	///	61	62	63	64	65	66	67	68	69	70	4
///	///	71	72	73	74	75	76	77	78	79	80	8

Note On the 80 print-position printer only the 380014 and 380015 instructions are applicable.

Instruction 380016

Effect A 380016 instruction will cause printing to take place on the right-hand print bank.

Operation When a 380016 instruction is given, the contents of Register B will cause printing to take place on print positions 81 to 120. The bit positions of Register B will be associated with print hammers 81 to 120 as shown below:

1	2	3	4	5	6	7	8	9	10	11	12	
///	///	81	82	83	84	85	86	87	88	89	90	1
///	///	91	92	93	94	95	96	97	98	99	100	2
///	///	101	102	103	104	105	106	107	108	109	110	4
///	///	111	112	113	114	115	116	117	118	119	120	8

Print Indicators

3.4.3

A print program requires the use of six indicators, two of which - indicators 45 and 47 - are used only for spacing and are described in 3.4.8. The remaining four are as follows:-

Indicator 42 Printer Ready

Purpose Indicator 42 is automatically set when the printer is available for printing.

Operation Indicator 42 is unset by the printer interlock if the printer is not available for printing. The printer is not available for printing when one of the following seven conditions arises:

- (a) The stationery feed mechanism is moving, i.e. the printer is spacing or throwing.

- (b) The automatic paper-feed stop device has operated. This occurs when the paper has been thrown excessively, this being regarded as an error condition. The autostop occurs when the paper throw is about twenty inches.
- (c) The print barrel mechanism has been opened.
- (d) The Stop Printer button has been pressed.
- (e) The Emergency Stop button has been pressed.
- (f) No paper i.e. the paper has passed the tensioning device.
- (g) The power supply has been cut off.

It must be noted that indicator 42 is *not* unset when tested.

Indicator 43 Print Index Point Time

Purpose Indicator 43 is set when the start of a sector is being positioned for printing.

Operation At every fifth character time, an impulse is emitted from the printer to the computer to denote the start of a new sector. This impulse will cause indicator 43 to be set and the print counter will be stepped down by 1. Indicator 43 is unset when tested by program. If the indicator is not unset (by testing) within a certain time from the start of a sector time, it is unset automatically.

Indicator 44 Print Character Time

Purpose Indicator 44 is set when a character is about to be presented for printing.

Operation In order that the computer program may be aware when a character is about to be presented for printing, impulses are emitted from the line printer to the computer. One impulse is emitted at the start of each character time and this impulse causes indicator 44 to be set. Indicator 44 will be unset when tested by program.

Indicator 49 Print Count Error

Purpose Indicator 49 will be set if the print counter is not in its correct condition when a master synchronizing pulse is emitted.

Operation At each complete revolution of the print barrel, a master (datum) pulse is emitted. If the print counter is not in its correct condition when this pulse is emitted, indicator 49 will be set. This indicator is unset when tested.

Indicator 49 should be tested at regular intervals in the program, for example when a document has been printed.

Whenever a fault occurs that sets this indicator it should be noted and an engineer informed.

A Printing Program

Essentially the process of printing covers the following:

- (a) Distributing print data in a correct sequence in I.A.S.
- (b) Converting the binary-coded information to the printing code i.e. 4-bit information converted to single-bit information (by creating row-binary) to operate the print hammer triggers.
- (c) Presenting the single-bit information to the appropriate print hammer at the correct time.
- (d) Above must also include provision for zero suppression and the creation of spaces between print characters where necessary.

Before entering the print program the output information for printing must be arranged in a sequence consistent with the print positions. In addition, the zone digits must be generated for numeric information brought to output from processing where the zone digits will have been discarded.

If data is to be printed on all three print banks (i.e. on all 120 print positions) then the print data should be stored in 24 successive locations of I.A.S. 24 locations are required because each location holds the zone or numeric components of ten print characters in digit positions 3 to 12. 12 successive locations of I.A.S. contain the zone components of the print data and the next 12 successive locations of I.A.S. contain the numeric components of the print data. Thus print data may be stored as shown in Figure 20. Thus, for example, position 3 of I.A.S. word 200 and position 3 of I.A.S. word 212 contain the zone and numeric components for data to be printed on print position 1.

Zone Components for print positions:	Stored in positions 3 to 12 of I.A.S. locations:	Numeric components for print positions:	Stored in positions 3 to 12 of I.A.S. locations:
1 to 10	200	1 to 10	212
11 to 20	201	11 to 20	213
21 to 30	202	21 to 30	214
31 to 40	203	31 to 40	215
'	'	'	'
'	'	'	'
'	'	'	'
111 to 120	211	111 to 120	223

Figure 20: EXAMPLE OF PRINT DATA DISTRIBUTION

If the printing of non-significant zeros to the left of a number or value is to be prevented, the position of the most-significant digit must be determined in order to insert the appropriate zone components into the words containing the zoning for output. For example, suppose that it is required to print a sterling value which may range from zero to £99999-19-11, and that single spaces are required to separate the £, s, and d, then the zone and numeric components for three possible values will be as shown in Figure 21.

	£	£	£	£	£		s	s		d
Numeric -	1	2	3	4	5	0	1	0	0	0
Zone -	1	1	1	1	1	0	1	1	0	2
Printed result:	1	2	3	4	5		1	0		11

(i)

	£	£	£	£	£		s	s		d
Numeric -	0	0	2	0	3	0	0	0	0	0
Zone -	0	0	1	1	1	0	0	1	0	3
Printed Result:			2	0	3			0		10

(ii)

	£	£	£	£	£		s	s		d
Numeric -	0	0	0	0	0	0	0	5	0	2
Zone -	0	0	0	0	0	0	0	1	0	1
Printed result:								5		2

(iii)

Figure 21: EXAMPLE OF STERLING PRINTING

It can be seen from the above examples that several points must be borne in mind when printing sterling values:

- (a) The pence position must be tested for 10d. or 11d., the zone components for which are 3 and 2 respectively; for 0-9, the zone component will be 1.
- (b) If spaces are required between £, s, and d, the corresponding spaces must be present in the output data (positions 7 and 11 in the example).
- (c) The most-significant £'s position must be determined in order to prevent the printing of non-significant zeros, the zone component being 1 for all £'s positions which are to be printed.
- (d) Assuming that the units of shillings position must always be printed, the zone component will be 1. The figure in the 10/- position can be either 1 or 0. If it is required to print in both instances the zone component will always be 1. If it is required to print 1 and suppress 0, the 10/- position must be tested to detect which number is present; a zone component of 1 then being inserted for the number to be printed and a zone component of 0 for the number to be suppressed.

Subroutines are available for the zero suppression of decimal or sterling amounts which achieve maximum efficiency in terms of time and storage capacity requirements.

Flowchart of a Print Program

3.4.5

Figure 22 is a flowchart of a program for printing one line. The following three assumptions are made:

- (a) Row-binarizing has not been completed before entering the program.
- (b) It is required to print on all three print banks.
- (c) All data to be printed has been distributed, as described under "Distribution of Print Data"

The letters in brackets beside the paragraphs describing the program relate to the letters shown on the flowchart.

- (a) Indicator 42 is tested.

If set proceed to (b).

If unset the printer is not yet ready (probably spacing). Loop back and continue testing until printer is ready.

- (b) Enter a 5 into Row Binary Register by means of function 30 and also into an I.A.S. location used as the Row Binary Counter. Set another I.A.S. location, used as the Index Point Counter, to 10.

- (c) The instructions to transfer the row-binarized zoning from Register B to I.A.S. (see (d) page 34) are formed by adding the contents of the Row Binary Counter to the basic 42 instructions; e.g. if the row-binarized information is to be stored in words 0 to 17 of block 10, the instructions would be as follows where x is the number in the Row Binary Counter.

D	F	A	R
-	42	0000+x	10
-			

Instruction to store row-binarized zoning for printing on left-hand bank

D	F	A	R
-	42	0006+x	10
-			

Instruction to store row-binarized zoning for printing on centre bank

D	F	A	R
-	42	0012+x	10
-			

Instruction to store row-binarized zoning for printing on right-hand bank

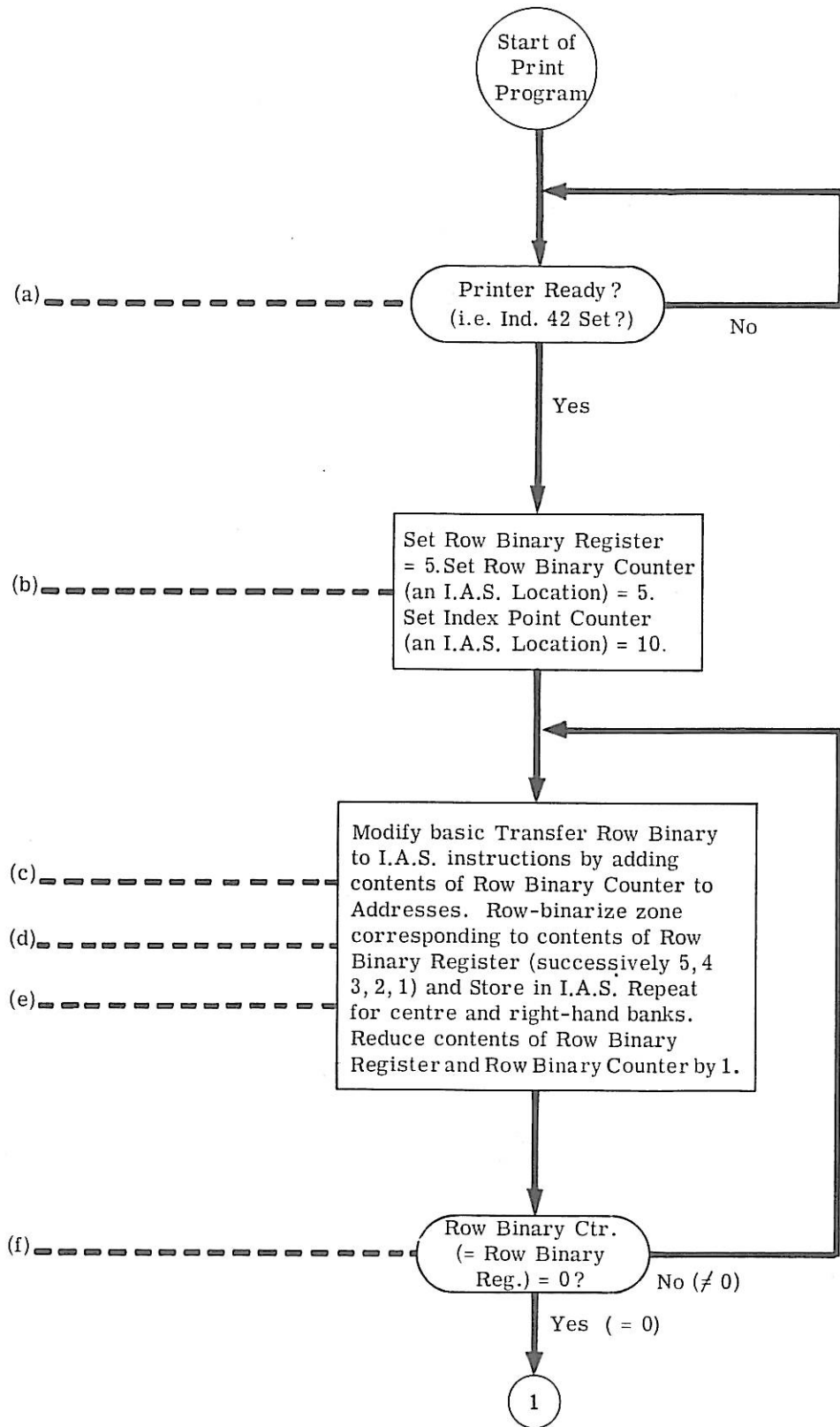


Figure 22: FLOWCHART OF PROGRAM TO PRINT ONE LINE

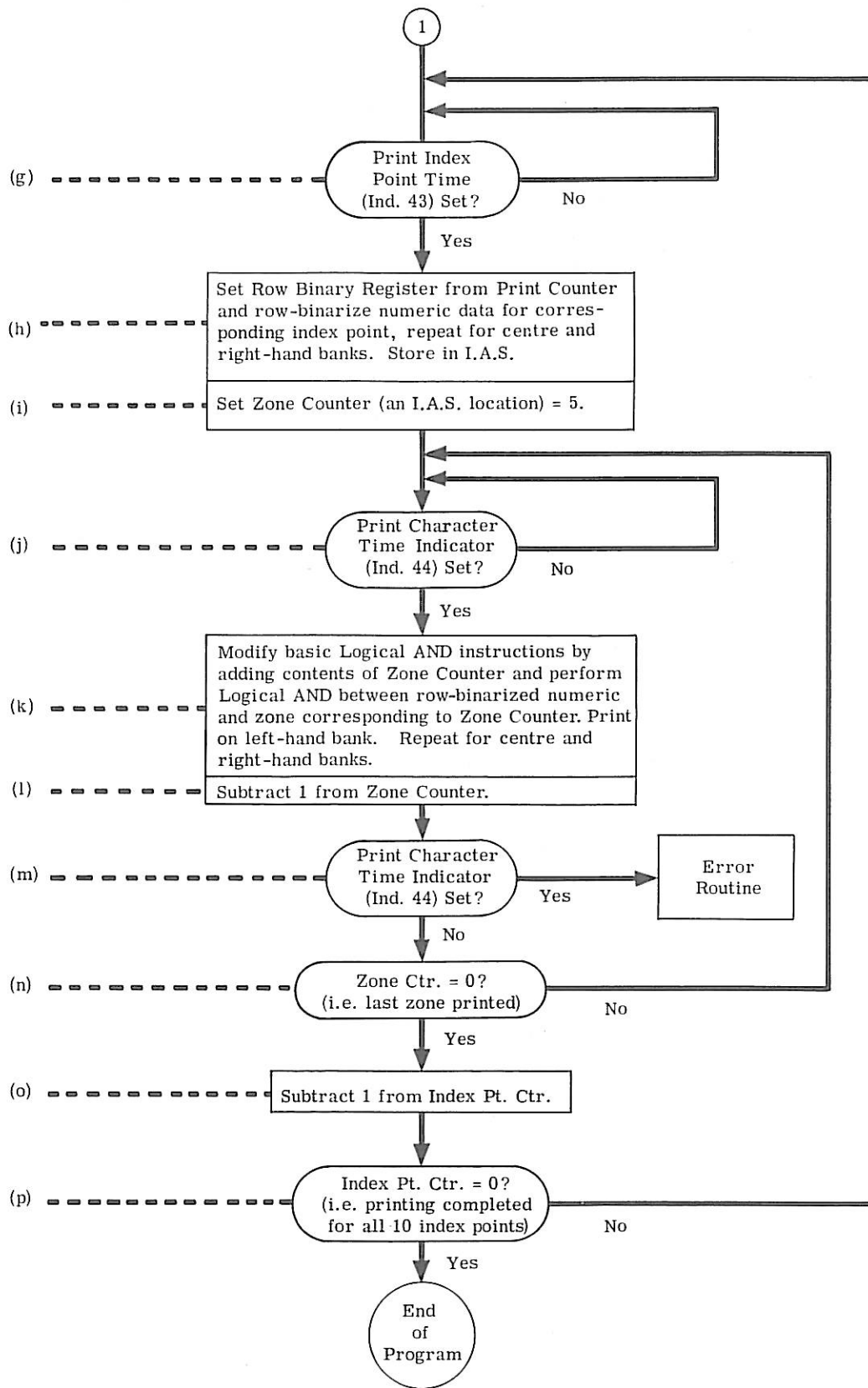


Figure 22: continued

- (d) The twelve words of zoning are row-binarized, the actual zone corresponding to the contents of the Row Binary Register. Zone 5 is binarized first, starting with the first four words of data; words 1 to 4 being binarized into the 1, 2, 4 and 8 streams respectively of Register B. The resulting row-binary is then stored in I.A.S. using the instruction created in (c) for storing away row-binary corresponding to the left-hand print bank. This is then repeated for words 5 to 8 (centre bank) and 9 to 12 (right-hand bank).
- (e) The contents of the Row Binary Counter are reduced by 1 (causing either indicator 01 or indicator 02 to be set) and the resulting figure is entered into the Row Binary Register.
- (f) Indicator 01 is tested to determine whether or not the contents of the Row Binary Register and Counter have now been reduced to zero. If not zero the program loops back to the beginning of (c). In (d) the next zone is row-binarized, the order being 5, 4, 3, 2, 1. After zone 1 has been binarized, the content of the Row Binary Counter is reduced to 0 and the program proceeds to (g).
- (g) Indicator 43 is tested to determine whether or not the next sector is ready for printing. If the indicator is not set the program continues to test it until it is, when the program proceeds to (h).
- (h) By means of the instruction:

D	F	A	R
-	38	00 13	-
---	---	---	---

the contents of the print counter, which corresponds to the number of the sector which is currently available for printing, is entered into the Row Binary Register. It will not be known what this number is since the position of the print barrel is also unknown. This will be of no consequence, provided the printing routine is repeated for each of the ten index points; it is not necessary to start with any particular sector. Following the 380013 instruction, the whole of the numeric data is row-binarized. The first four words are binarized into the 1, 2, 4 and 8 streams of Register B respectively, then stored in I.A.S. by means of a basic transfer instruction as used in (c). This is then repeated for words 5 to 8 and words 9 to 12. Row-binarizing is thus completed for printing all five characters of the sector concerned, the location of row-binary in I.A.S. being as shown in Figure 23.









- (i) Enter a 5 into an I.A.S. location which is used as the Zone Counter.
- (j) Test indicator 44 to determine whether or not a character of the sector is in position for printing. The program repetitively tests the indicator until it is set, when the program proceeds to (k).
- (k) Every character to be printed is represented within the computer by numeric and zone components. Consider the letter H (numeric 8, zone 2) about to be printed. Assuming

Row-binary for printing on left-hand bank		Row-binary for printing on centre bank		Row-binary for printing on right-hand bank	
Word No.	Zone/Numeric	Word No.	Zone/Numeric	Word No.	Zone/Numeric
0	Numeric	6	Numeric	12	Numeric
1	Zone 1	7	Zone 1	13	Zone 1
2	Zone 2	8	Zone 2	14	Zone 2
3	Zone 3	9	Zone 3	15	Zone 3
4	Zone 4	10	Zone 4	16	Zone 4
5	Zone 5	11	Zone 5	17	Zone 5





Figure 23: LOCATION OF ROW - BINARY FOR PRINTING IN I.A.S.

the row-binary to be stored in the locations shown in Figure 23, then the binary for the numeric component is in words 0, 6 and 12, and that for the zone component (zone 2) in words 2, 8 and 14. The row-binary for character H is produced by means of Logical AND between words 0 and 2 (left-hand bank), 6 and 8 (centre bank) and 12 and 14 (right-hand bank).

Example: Assume words 0 and 2 contain the following row-binary

Word 0		0	0	1	0	0	0	0	0	1	0	1
		1	0	0	1	0	1	0	0	0	0	2
		0	1	0	0	0	0	0	0	0	0	4
		0	0	0	1	0	0	1	0	0	0	8
Word 2		0	1	1	0	1	0	0	1	0	1	1
		1	0	0	1	1	0	0	0	1	0	2
		0	1	0	0	1	1	1	0	0	0	4
		0	0	1	0	0	1	0	0	0	0	8

The contents of word 0 indicate that a character with a numeric component of 8 (i.e. 8, H, Q, Y or @) is to be printed on each of positions: 3, 9, 11, 14, 16, 22, 34, 37 and 40 of the left-hand bank. The contents of word 2 indicate that a character with zone component 2 (i.e. 11 or any letter from A to I) is to be printed on each of positions: 2, 3, 5, 8, 10, 11, 14, 15, 19, 22, 25, 26, 27, 33, 36 and 40 of the left-hand bank. The contents of word 0 are transferred to Register B and Logical AND is performed between Register B and word 2, resulting in a 1 in any bit position where there was a 1 in the corresponding bit position of both words, thus:

	0	0	1	0	0	0	0	0	0	0	0	1
	1	0	0	1	0	0	0	0	0	0	0	2
	0	1	0	0	0	0	0	0	0	0	0	4
	0	0	0	0	0	0	0	0	0	0	0	8

This indicates that character H is to be printed on each of positions: 3, 11, 14, 22 and 40 of the left-hand bank.

Immediately following the Logical AND a 380014 instruction is given, and the H's are printed on the left-hand bank. Then the row-binary for the centre bank is produced by Logical AND between words 6 and 8, and character H printed on the centre bank; and similarly for the right-hand bank.

For each sector, the characters with zone component 5 are printed first, followed by those with components 4, 3, 2 and 1.

- (l) The zone counter is stepped down by 1.
- (m) Indicator 44 is tested.
If unset: continue to (n).
If set: too much time has been consumed in printing character and the following character time has been encroached upon. Program enters the appropriate error routine.
- (n) (i) If zone counter positive after (l), program loops back to (j).
(ii) If zone counter contains 0, all five characters on the sector have been printed and the program proceeds to (o).
- (o) The index point counter is stepped down by 1.
- (p) (i) If the index point counter positive, the program loops back to (g) to print the characters on the next of the ten sectors.
(ii) If index point counter contains 0, printing has been completed for all ten sectors, and therefore the complete line has been printed.

Paper Movement

3.4.6

Continuous stationery employed on the printer is punched with marginal sprocket holes and is tractor driven. Up to 4-part stationery may be used and the stationery and carbon paper must conform to the 1300-series stationery specifications.

Paper movement is controlled by the computer and either paper spacing or paper throwing (i.e. more than five spaces) is possible. The marginal sprocket holes in the stationery are engaged on the upper and lower sets of tractor wheels. These motor driven tractors rotate, thus moving the stationery. The rotating action of the tractors is controlled by a sprag and ratchet wheel mechanism; this mechanism is shown diagrammatically in Figure 24. It can be seen that the mechanism consists of six sprags engaging into a ratchet wheel which has five faces; the ratchet wheel controls the rotation of the tractor wheels and therefore rotates with paper movement. The rotating action of the ratchet wheel is in turn controlled by the lifting and dropping of the six sprags according to the computer program instructions.

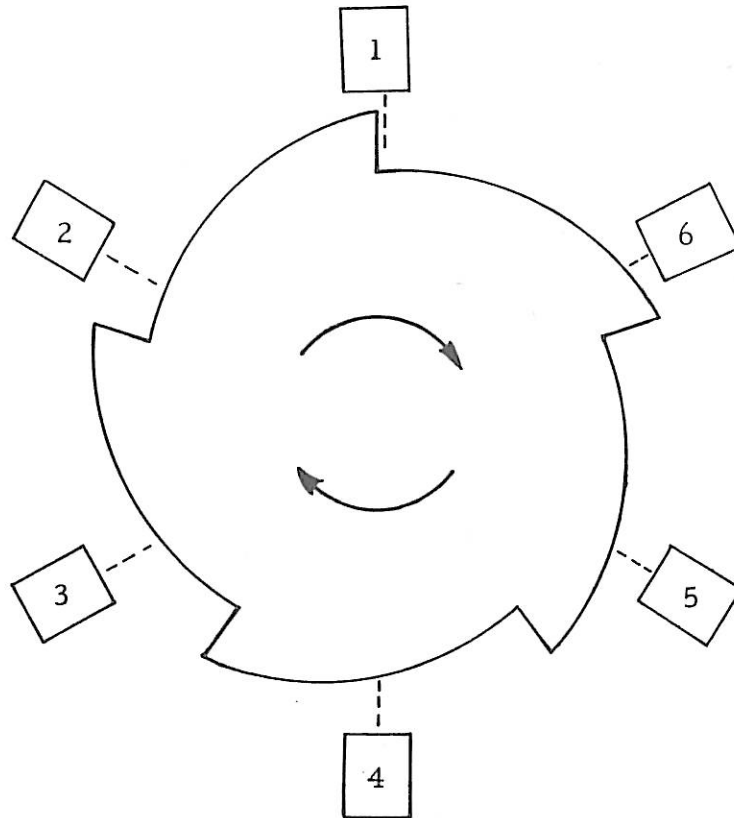


Figure 24: SPRAG MECHANISM

If a sprag is dropped, the rotation of the ratchet wheel is arrested when that sprag makes contact with one of the five faces; thus, paper movement is arrested. If a sprag is lifted and another sprag is dropped, the ratchet wheel will rotate until it is arrested by the dropped sprag; thus, a certain movement of the paper will take place before the paper is again stopped. When it is known which sprag is currently engaged, a single space (a vertical movement of $\frac{1}{6}$ inch) may be achieved by raising that sprag (and all other sprags) and allowing the ratchet wheel to rotate. The next sprag in counterclockwise succession is dropped thus arresting the movement. Similarly, two spaces may be achieved by lifting the engaged sprag and dropping the second sprag in counterclockwise succession and so on; up to five spaces may be achieved by this method.

An interlock is set up by any paper movement order and will remain set until the paper comes to rest. An automatic stop device is incorporated in the printer which will operate when the paper has been thrown about twenty inches. The device will cause:

- (a) The paper feed mechanism to stop, and
- (b) an interlock between printer and computer to be set up, and
- (c) a visual indicator on the printer to be illuminated and remain so until it is switched off by a manual control on the printer switch panel.

Paper Throw Instructions

3.4.7

Sprag Instructions 380020 to 380026

All paper movement instructions are sprag instructions; that is, they control the lifting and dropping of the sprags. These instructions are shown in the table below.

Function	Address	Description
38	20	Lift all sprags
38	21	Drop Sprag 1 : lift all others
38	22	Drop Sprag 2 : lift all others
38	23	Drop Sprag 3 : lift all others
38	24	Drop Sprag 4 : lift all others
38	25	Drop Sprag 5 : lift all others
38	26	Drop Sprag 6 : lift all others

These instructions are used to control the sprags as explained in the preceding description of the operation of the sprags.

Paper Throw Indicators

3.4.8

Indicator 45 Line Space Time

Purpose Indicator 45 is set when each line space has been completed.

Operation When a line space has been achieved indicator 45 will be set. It will be unset when tested by program. If indicator 45 is not tested (and therefore not unset) within a certain time from it becoming set, then it will be unset automatically.

Notes Indicator 45 will require testing when more than five line spaces are required on a single throw; a further description of the use of this indicator is given under 3.4.9. It is possible to time-share paper throwing with another program by using the time between successive line space indications.

Indicator 47 Paper Trolley Empty

Purpose Indicator 47 is set automatically when the supply of paper in the printer is almost exhausted.

Operation The paper stacking trolley is fitted with a Paper Low microswitch. When the last sheet of paper leaves the trolley then indicator 47 is set. It will remain set until a new supply of paper is inserted.

Notes This indicator should be tested regularly throughout a print program e.g. after each document is printed.

Flowchart for a Paper Throw Program

3.4.9

A sample flowchart of a spacing routine is shown in Figure 25. Two examples are given overleaf of the selection of the appropriate sprag when it is known which sprag is engaged and how many spaces are required.

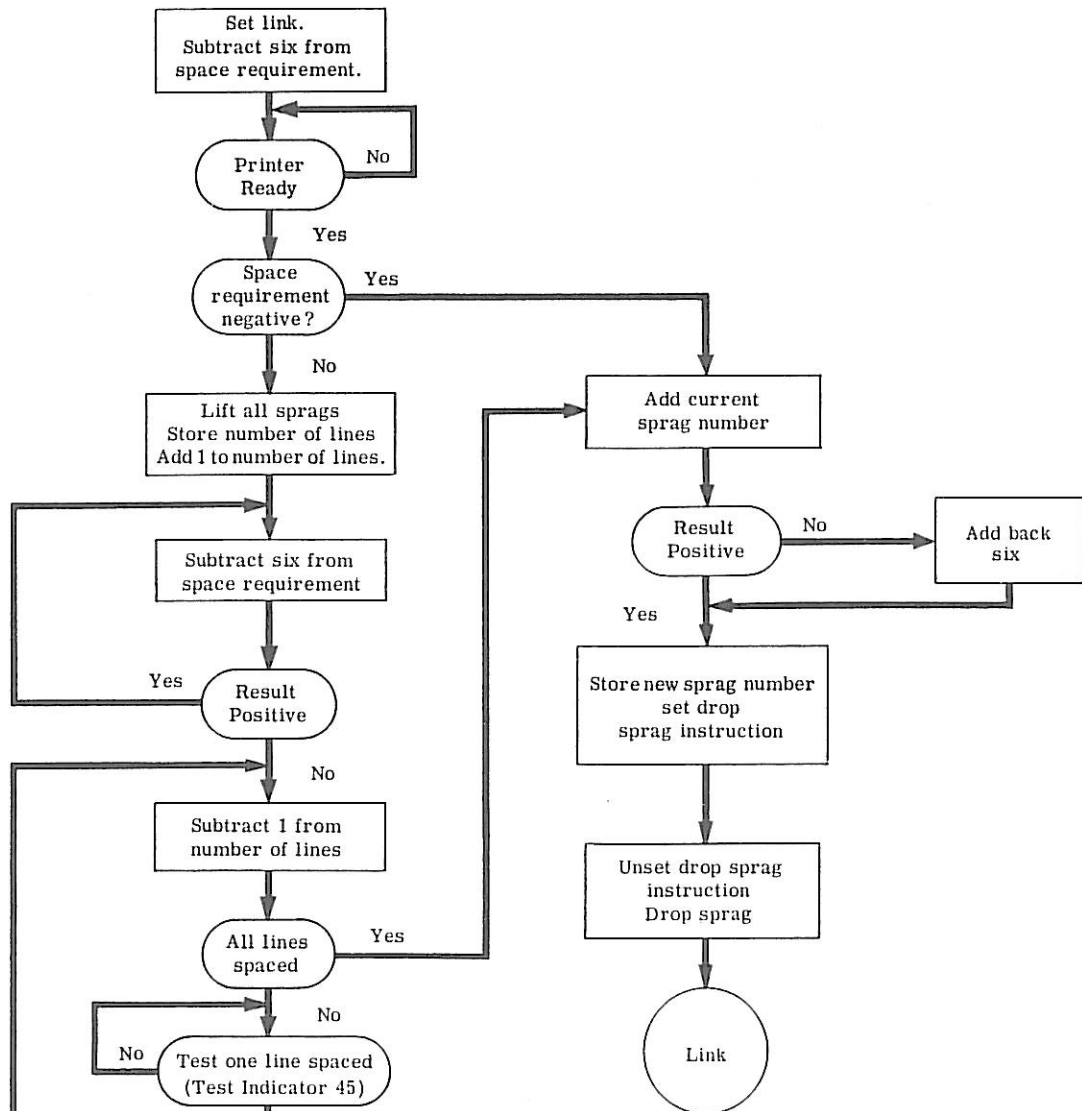


Figure 25: SPACING ROUTINE FOR PRINTER

Example 1

Sprag 5 engaged; space 3 lines

Number of spaces required = 3

Subtract 6 (constant) = $3 - 6 = -3$

Result negative: add the sprag engaged number (5) = $-3 + 5 = +2$ (Sprag No.)

Result positive: drop sprag number 2 immediately.

Example 2

Sprag 3 engaged; throw 7 lines

Number of spaces required = 7

Subtract 6 (constant) = $7 - 6 = +1$

Result positive: lift all sprags;

Subtract 6 again = $1 - 6 = -5$

Result negative: count lines spaced.

All lines spaced: add the sprag engaged number (3) = $-5 + 3 = -2$.

Result negative : add back 6 (constant) = $-2 + 6 = +4$: drop sprag number 4.

Timings

3.4.10

The line printer may be one of the following types:

- (a) 120 print positions operating at a maximum speed of 600 lines a minute, or
- (b) 120 print positions operating at a maximum speed of 300 lines a minute, or
- (c) 80 print positions operating at a maximum speed of 300 lines a minute.

The following table lists the timings associated with the 600 lines a minute and 300 lines a minute printers respectively. The timings are not affected by the number of print positions.

	600 lines a minute Minimum Timings	300 lines a minute Minimum Timings
Interval between successive character times	1.2 ms	2.6 ms
Interval between successive line space indications at speed	6.9 ms	6.9 ms
Duration for which Indicator 45 ("Line Space Time") is set if not tested	3.1 ms	3.1 ms
Duration for which Indicator 43 ("Print Index Point Time") is set if not tested	550 to 800 μ s	1150 to 1700 μ s
Time to space first line (i.e. from time instruction given until the paper comes to rest)	32 ms	32 ms
Time each additional line will take after first line space	7.56 ms	7.56 ms



THE PAPER - TAPE READER

3.5

One or two paper-tape readers may be linked to the computer as additional units. If two paper-tape readers are specified they are incorporated in one free-standing unit. The paper-tape reader is capable of accepting 5-, 6-, 7- or 8-track tape, a Track-Select switch being set for the appropriate number of tracks on the tape to be read. Tape reading is achieved by means of photo-electric cells.

Interpretation within the Computer

3.5.1

It is recommended that British Standards Institute tape codes should be used although it is possible to use most standard paper-tape codes. British Standards Institute recommended tape codes are given in Part 6 of this manual.

A paper-tape character consists of a zone and numeric component; these components are represented on the tape tracks as shown in Figure 26. There are four numeric components (N1, N2, N4 and N8) on 5-, 6-, 7- and 8-track tape represented by tape tracks 1, 2, 3 and 4 respectively: 5-track tape has 1 zone (Z1) represented by track 5; 6-track tape has 2 zones (Z1 and Z2) represented by tracks 5 and 6. Both 7- and 8-track tape have zones represented by tracks 6 and 7 and 6, 7 and 8 respectively; track 5 is employed for parity purposes and is not read in the interpretation of a character.

When a valid character is read, it is entered into an 8-bit buffer store so that the zone and numeric components are each stored in 4 bits of the buffer. The contents of this buffer store are transferred to Register B as two digits, one representing the zone and one representing the numeric. The zone component will be read into position 1 of Register B and the numeric component will be read into position 7 of Register B.

The previous contents of positions 1 to 5 and 7 to 11 of Register B will be shifted one position to 2 to 6 and 8 to 12 respectively. The previous contents of positions 6 and 12 of Register B will be lost. In position 1 of Register B, those bits which are not employed in storing the zone components are automatically zeroized. Thus, when 8-track tape is being read, the 8-bit of position 1 is zeroized automatically as zone 7 is the maximum value possible in position 1, and when 6- or 7-track tape is being used, the 4 and 8-bits are zeroized as zone 3 is the maximum value possible in position 1. Similarly, 5-track tape has only one zone so the 2, 4 and 8 bits are zeroized automatically.

Valid Tape Characters

On 5-track paper tape all codes including those with parity errors will be valid characters with the exception of blank tape. Blank tape is that tape which has a row punched with a feed hole only, i.e. no data holes or parity hole punched.

On 6-, 7- and 8-track paper tape neither the blank tape code nor the erase (all data holes punched) code are recognized as valid characters.

No. of Tracks	Track No.	Computer Interpretation
5	1	N1
	2	N2
	3	N4
	4	N8
	5	Z1
6	1	N1
	2	N2
	3	N4
	4	N8
	5	Z1
	6	Z2
7	1	N1
	2	N2
	3	N4
	4	N8
	5	Parity
	6	Z1
	7	Z2
8	1	N1
	2	N2
	3	N4
	4	N8
	5	Parity
	6	Z1
	7	Z2
	8	Z4

(N = Numeric Z = Zone)

Figure 26: COMPUTER INTERPRETATION OF PAPER-TAPE CODE PUNCHINGS

Invalid characters are ignored and are not read into the computer. The detection of an invalid character does not stop the computer but the tape is fed until a valid character is detected.

Parity Checking

3.5.2

As stated above, a parity check is made only when 7- or 8-track tape is being read and tape track 5 is allocated to punching a parity bit.

When 7-track tape is read, odd parity will be checked automatically when each character is read and the parity bit will not be read into the computer.

When 8-track tape is being read, even parity will be checked when each character is read and the parity bit will not be read into the computer.

If a parity error is detected (i.e. even parity on 7-track tape or odd parity on 8-track tape) an indicator (61) will be set within the computer; this indicator may be tested by program. The indicator will be permanently unset when 5- or 6-track tape is being read, i.e. no parity check is being made.

Tape Read Instructions

3.5.3

Instruction 380050

Effect This instruction will cause a character to be read and the tape reader will then step to the next valid tape code.

Operation When a 380050 instruction is given, a valid character is read into the buffer store. The previous contents of the buffer (zone and numeric components of the last character read) will be transferred to positions 1 and 7 of Register B.

Notes When setting up the selected tape reader it is necessary to give two extra 380050 instructions. The first 380050 instruction will cause the first valid tape character to be positioned beneath the reading photo-electric cells. The second 380050 instruction will cause the character to be read into the reading buffer. The next 380050 instruction (given in the reading program) will cause the contents of the buffer to be transferred to Register B. It will be remembered that only positions 1 and 7 of Register B are affected when reading in data from the buffer store, thus the remaining ten positions of the register may contain rubbish.

Instructions 380051 and 380052

Effect Instruction 380051 selects paper-tape reader 1 for operation. Instruction 380052 selects paper-tape reader 2 for operation.

Operation A 380051 or 380052 instruction will select the appropriate paper-tape reader for operation and a subsequent 380050 instruction will cause the reading of tape to commence on the selected reader.

Indicator 60 Tape Reader Ready

Purpose This indicator is set when the selected paper-tape reader is ready to read a valid character.

Operation A 380051 or 380052 instruction will cause this indicator to be associated with paper-tape reader 1 or 2 respectively. Indicator 60 will be set automatically when the following three conditions are satisfied:

- (a) The tape reader motor is running.
- (b) The photo-electric cell lamp is on.
- (c) A valid character is positioned beneath the photo-electric cell unit.

Indicator 60 will be unset if any of the above three conditions is not satisfied.

Notes Indicator 60 should be tested during a paper-tape read-in program before each tape code is read thus:

I	D	F	A	R	NARRATIVE
13	4	60	0014	B	Selected reader ready
	4	00	0013	B	
14		57	0012		Clear Register B
		38	0050		Read 1 tape character

Indicator 61 Parity Error

Purpose This indicator is set when a parity error is detected while reading 7- or 8-track tape.

Operation The execution of instruction 380051 or 380052 will cause this indicator to be associated with paper-tape reader 1 or 2 respectively.

Indicator 61 will be set when:

- (a) the number of holes (data plus parity hole) punched in one row of a valid character in 7-track tape is found to be even, or
- (b) the number of holes (data plus parity hole) punched in one row of a valid character in 8-track tape is found to be odd.

Indicator 61 is unset when tested by program.

Notes Indicator 61 should be tested during the read-in program and a section of the program should be devoted to parity errors. When setting up the selected tape reader it is advisable that indicator 61 should be unset (by testing) before the first tape character is read thus:

I	D	F	A	R	NARRATIVE
2		38	0050		Move to 1st Valid Tape Character
		38	0050		Read 1st Valid Tape Character
3	4	61	0004	B	Unset Parity Error Indicator

When 5- or 6-track tape is being read, no parity check is made and therefore indicator 61 is permanently unset.

Tape Read Programs

3.5.5

A flowchart for reading one character is shown in Figure 27. It is however strongly recommended that the standard I.C.T. subroutines for the paper-tape reader should be used. These subroutines, which enable a block of several characters to be read and stored, facilitate the following:

- (a) Reading each valid paper-tape character and storing it in the correct position within the selected output word in I.A.S.
- (b) An exit to the main program when:
 - (i) the appropriate number of characters have been read and stored, and/or
 - (ii) an end of data block code, specified by the programmer, is read from the tape.

A section of the subroutine effects the selection of the required paper-tape reader and a section for parity errors is also included.

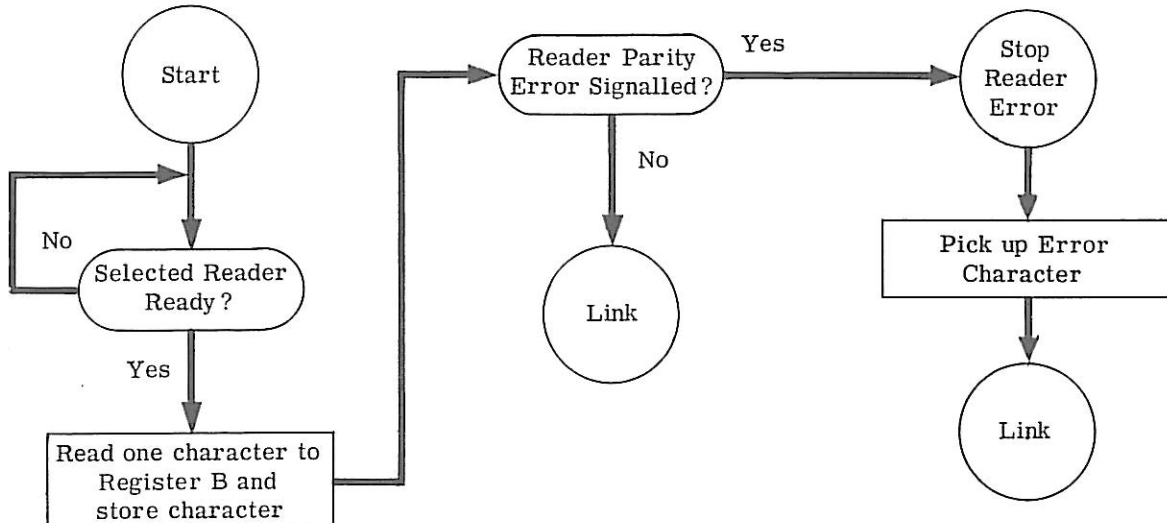


Figure 27: FLOWCHART FOR READING PAPER TAPE

Timings

3.5.6

The only timing relevant to the programmer is that tape is read at a maximum speed of 1000 characters a second; thus the average time required to read one tape character is 1 millisecond.

THE PAPER - TAPE PUNCH

3.6

One paper-tape punch may be fitted as an additional unit to the computer. The punch is available for 5-, 6-, 7- or 8-track tape and the number of tracks to be punched has to be specified when ordering.

Tape is punched at a punching station fitted with one punch knife for each tape track. The punched tape is read for checking purposes by photo-electric cells at a reading station located three characters beyond the punching station.

Output from the Computer

3.6.1

The zone and numeric components of the character to be punched must be located in positions 1 and 7 of Register B. When a punch instruction is given, the zone and numeric components are transferred to an 8-bit buffer and punched. The tape will be punched with the zone and numeric components as shown in Figure 28. Those bits of position 1 in Register B for which there is no corresponding paper-tape track must be zero. When a character is read from Register B to the buffer, the contents of Register B will be unchanged, i.e. Register B will not be shifted nor will its contents be lost.

	Zone Component Position 1 of Register B		Numeric Component Position 7 of Register B	
	Bit Number	Track Number	Bit Number	Track Number
5-track Tape	1	5	1	1
			2	2
			4	3
			8	4
6-track Tape	1	5	1	1
			2	2
			4	3
			8	4
7-track Tape	1	6	1	1
			2	2
			4	3
			8	4
8-track Tape	1	6	1	1
			2	2
			4	3
			8	4

Figure 28: COMPUTER INTERPRETATION OF OUTPUT TO PAPER TAPE

Checking Facilities

3.6.2

In 7- and 8-track paper tape, track 5 is allocated for parity purposes. When a character is punched, a parity bit is automatically generated to maintain odd parity in 7-track tape or even parity in 8-track tape. The parity bit is punched in track 5. The punched-tape codes are then read at the check-reading station and a parity check is made on the tape codes. If the parity check fails an indicator (67) is set within the computer.

No parity check is made on either 5- or 6-track tape and the Parity Check indicator remains unset.

Tape Punch Instructions

3.6.3

Instruction 380076

Effect This instruction will cause one character to be punched and one character to be read for parity checking.

Operation A 380076 instruction will cause one character (zone and numeric components in positions 1 and 7 of Register B) to be read into the 8-bit tape-punch buffer and thence to be punched.

A 380076 instruction will also cause one character to be read at the reading station for parity checking.

Notes A 380076 instruction must only be given when indicator 66 (Tape Punch Ready) has been tested and the punch found to be ready.

Tape Punch Indicators

3.6.4

Indicator 65 Tape Supply Low

Purpose This indicator is set when there is less than twenty feet of tape on the feed spool.

Operation Indicator 65 will be set as described above and unset automatically when there is more than twenty feet of tape on the feed spool. This indicator may be tested by program.

Notes Indicator 65 must be tested at convenient places in the program, e.g. at the beginning of every data block to be punched.

Indicator 66 Tape Punch Ready

Purpose Indicator 66 will be set automatically when the tape punch is ready to punch one character.

Operation Indicator 66 will be set when the following conditions are satisfied:

- (a) The power supply to the tape punch is on.
- (b) The 8-bit punch buffer is clear to accept a character from Register B.

The indicator will be unset if either of the above two conditions is not satisfied.

Notes Indicator 66 must be tested and found to be set before a 380076 instruction is given.

Indicator 67 Tape Punch Error

Purpose This indicator will be set when the parity check at the punch reading station fails.

Operation Indicator 67 will be set automatically if the parity check fails thus:

- (a) The number of code holes read (data plus parity holes) in one row of 7-track tape is found to be even (i.e. in error).
- (b) The number of code holes read (data plus parity holes) in one row of 8-track tape is found to be odd (i.e. in error).

Indicator 67 will be unset when tested by program.

Notes Indicator 67 should be tested after each punch instruction when 7- or 8-track tape is being punched. When indicator 67 is tested and found to be set (i.e. a parity error has occurred), the character which is in error is not the last character punched but a previously punched character because the reading station is located three tape codes beyond the punching station.

On the initial paper-tape loading condition (e.g. a new spool of tape) blank tape (tape punched with feed holes only) and perhaps spurious punching will be read at the reading station before a valid punched code is read. It is necessary therefore that parity error conditions which occur during tape run-in are ignored. Therefore indicator 67 must be tested before a character is punched to ensure that it is unset.

If blank tape is produced during a 7-track paper-tape punching program, it is necessary to suppress the punching of the parity bit which would normally be punched to give the correct odd parity condition. An additional bit must therefore be read into the punch buffer (via Register B) to simulate an odd parity condition and this additional bit must be positioned so that it will not be punched into the tape. For example, when blank tape is to be produced, the buffer store could hold a zone component of 4 and a numeric component of 0; thus odd parity would be detected and no parity hole would be punched. As the 4-bit of the zone component is not punched into 7-track tape, both zone and numeric are interpreted as zero and no code is punched.

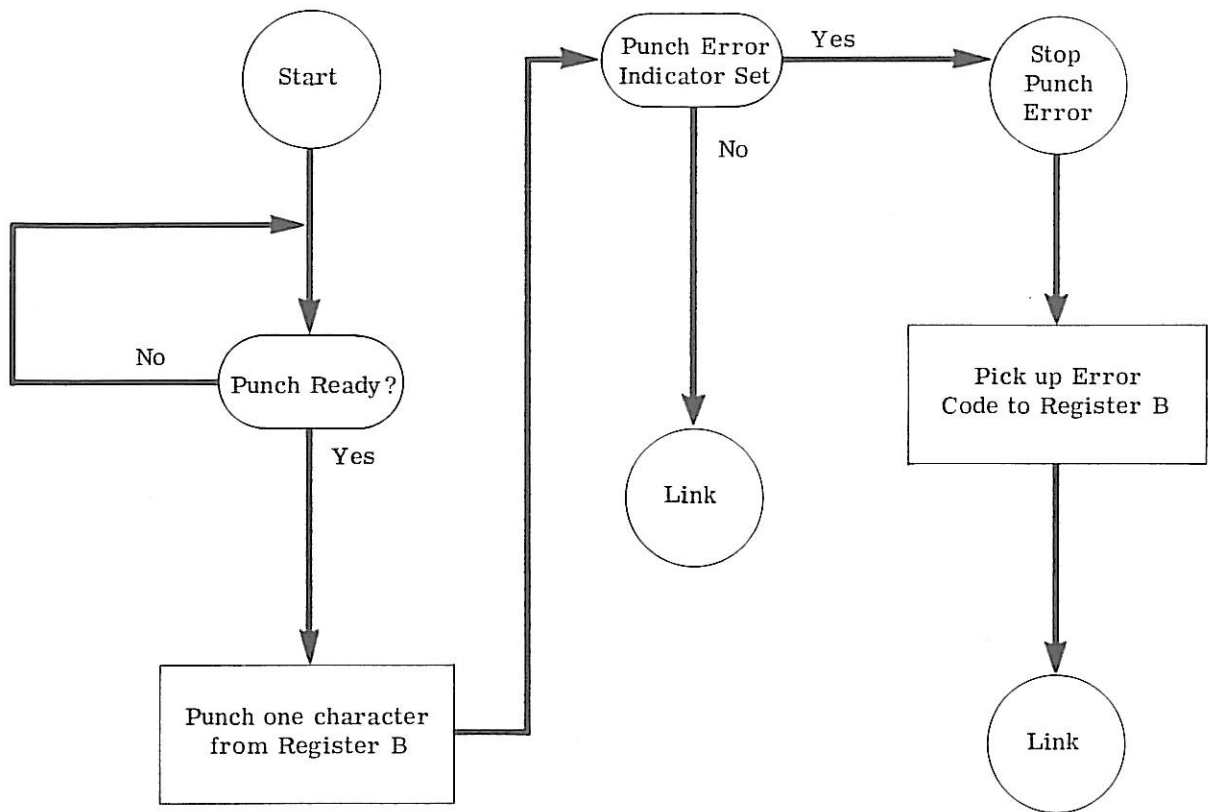


Figure 29: FLOWCHART FOR PUNCHING PAPER TAPE

Tape Punch Programs

3.6.5

A flowchart to punch one character is shown in Figure 29. It is strongly recommended that the standard I.C.T. subroutines for the paper-tape punch should be used. These routines allow for punching a block of several characters and provide the following facilities:

- (a) Transferring the data to be punched from the appropriate successive locations in I.A.S. to Register B, positioning the individual zone and numeric component within positions 1 and 7 of this Register and punching the data.
- (b) Provision for an exit to the main program when:
 - (i) an end of data block code, specified by the programmer, is detected, and/or
 - (ii) the appropriate number of characters have been punched.

Timings

3.6.6

The only timing relevant to the programmer is that tape is punched at a maximum speed of 300 characters a second; thus the average time required to punch one tape character is 3.33 milliseconds.

One interrogating typewriter may be fitted to a 1300-series machine as an additional unit. The typewriter can operate in a type-in or type-out mode and the selection of the mode of operation is by program. Thus the typewriter may be used for keying in data while in the type-in mode and printing out data while in the type-out mode.

The typewriter consists basically of a keyboard and print unit. There is no mechanical link between the keyboard and the print unit except when the typewriter is being used in the type-in mode. Thus except for engineering maintenance the typewriter cannot be used off-line from the computer.

Keyboard

The layout of the typewriter keyboard is shown in Figure 30. It can be seen that a repertoire of 74 characters is available and this repertoire comprises 16 numeric characters 0 to 15, 26 alphabetic characters A to Z and 32 symbols; five function bars and five function keys are also fitted on the keyboard.

When the typewriter is in the type-in mode and the computer program is ready to accept data typed in, the Type indicator lamp will glow and the operator may commence typing. When the typewriter is in the type-out mode, the keyboard is locked and all keys are inoperative except for the Request Type-in key (labelled Type In) and the shift keys.

When a character key or function key is pressed (i.e. all function keys except the type-in key and shift keys), a code is automatically generated and a subsequent Type instruction enters this code into Register B.

Print Unit

The typewriter print unit operates under manual control during the type-in mode and under computer control during the type-out mode. It is possible to print out the same characters as arranged on the typewriter keyboard, i.e. the 16 numeric characters 0 to 15, the 26 alphabetic characters A to Z and the 32 symbols.

There are 44 type bars but the full repertoire of 74 characters and symbols is available as each type bar has an upper and lower shift component. (N.B. numeric characters 0 to 11 and the symbols Full Stop and Comma are arranged in both upper and lower case positions.)

A dual-coloured inked ribbon, red and black, is fitted between the print hammers and the stationery. When the typewriter is in the type-in mode the red ribbon is automatically set for printing and when the typewriter is in the type-out mode the black ribbon is automatically positioned for printing.

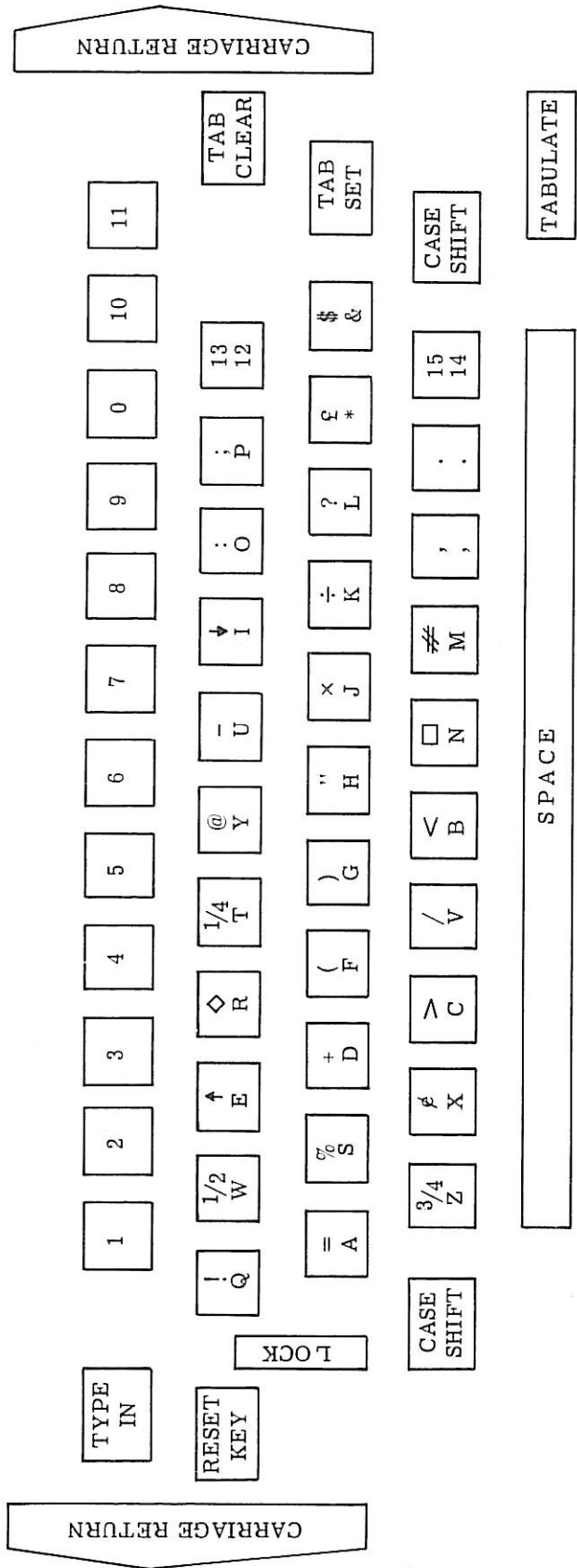


Figure 30: INTERROGATING TYPEWRITER:KEYBOARD LAYOUT

As in a normal typewriter, tabs are fitted and the tabs may be set to the required positions or cleared either manually or by computer program.

In the type-out mode, the typewriter carriage operates under computer control, i.e. it is operated by a space, line feed, tabulate or carriage return code from the computer.

A continuous stationery feeding device is fitted to the typewriter print unit. The maximum length of a continuous roll of stationery (single part) is 250 feet. The device is fitted with a Paper Present micro switch which will anticipate the running out of paper by approximately eight inches.

Representation within the Computer

3.7.1

All codes which are entered into the computer from the typewriter or transferred from the computer to the typewriter consist of a zone and numeric component. The zone and numeric components of each typewriter code for the 74 characters and symbols available are shown in Figure 31. The components of each code are represented as two digits. When a typewriter key has been pressed, a subsequent type instruction causes the zone and numeric components of the character concerned to be entered into positions 1 and 7 of Register B respectively. Similarly in the type-out mode, the zone and numeric components of a character to be printed must be positioned in positions 1 and 7 of Register B. When a type instruction is given, the zone and numeric components in Register B are transferred to a 7-bit buffer store.

It will be noted that the typewriter code is compatible with the computer code used in the punching of cards and printing on the line printer, the 10 and 11 representation being as in the card code. Zones 0, 6 and 7 are not of course included in the standard codes for the card punch and line printer.

It can be seen in Figure 31 that 'Do Nothing' is represented by zone 0 (as are all control codes) and numeric 1; if unallocated combinations of components are used for output for the typewriter, they will *not* produce 'Do Nothing' but a character may be erroneously typed for this code.

		ZONE						
Numeric	0	1	2	3	4	5	6	7
0	Space	0			*	£	,	¢
1	Do Nothing	1	A	J	&	\$	=	x
2	Tabulate	2	B	K	S	%	<	÷
3	Set Tab	3	C	L	T	$\frac{1}{4}$	>	?
4	Clear Tab	4	D	M	U	-	+	#
5	Carr. Return, L/F.	5	E	N	V	/	↑	□
6		6	F	O	W	$\frac{1}{2}$	(:
7		7	G	P	X	.)	;
8		8	H	Q	Y	@	"	!
9		9	I	R	Z	$\frac{3}{4}$	↓	◇
10		10						
14		14						
15		15						

Figure 31: REPRESENTATION OF TYPEWRITER CODES IN COMPUTER

Instruction 380070

Effect A 380070 instruction will cause the typewriter to be set to the type-in mode.

Operation A 380070 instruction will cause the following:

- (a) The typewriter keyboard will become released (if it had been previously locked by a 380071 instruction; see below).
- (b) The Type indicator lamp on the typewriter will glow.
- (c) The inked ribbon will be positioned so that data will be printed in red.

Notes When the typewriter is in the type-out mode and while the typewriter subroutine is not required, the typewriter keyboard will be locked by a 380071 instruction. Thus a safeguard is provided against the accidental operation of the keyboard.

As previously described, the Type indicator lamp will glow thus showing the operator that the typewriter is in the type-in mode. This provides a further safeguard to ensure that the type-in subroutine is in I.A.S. before data are typed in. A 380070 instruction must only be given after indicator 51 has been tested and found to be set.

Instruction 380071

Effect A 380071 instruction will cause the typewriter to be set to the type-out mode.

Operation A 380071 instruction will cause the following:

- (a) The typewriter keyboard will be locked.
- (b) The inked ribbon on the print unit will be positioned so that data are printed in black.
- (c) The Type indicator lamp will be extinguished.

Notes The operation of any key on the keyboard other than the Request Type-in key will be ignored. This provides a safeguard against the accidental operation of the keyboard during the type-out mode.

It is advised that the keyboard should be locked (by a 380071 instruction) and released only when the keyboard is about to be operated; i.e. the keyboard should be locked when a type-out subroutine link is stored.

I	D	F	A	R
0	---	41	0000	01
		38	0071	-

A 380071 instruction may be given at any time, but it will only become effective when any operation in progress has been completed.

Instruction 380072

Effect When the typewriter is in the type-in mode, a 380072 instruction will cause the character being typed in to be read into Register B.

When the typewriter is in the type-out mode, a 380072 instruction will cause a character to be read from Register B to the 7-bit typewriter buffer, and printed.

Operation A 380072 instruction is given after the typewriter has been set in the type-in or type-out mode, i.e. after a 380070 or 380071 instruction.

TYPE-IN MODE: A 380072 instruction will cause the zone and numeric components of the character being keyed in to be entered into positions 1 and 7 of Register B respectively. The previous contents of positions 1 to 5 and 7 to 11 of Register B will be shifted into positions 2 to 6 and 8 to 12 respectively. The previous contents of positions 6 and 12 of Register B will be lost.

A 380072 instruction will also cause the function keys to operate as shown in Figure 32.

TYPE-IN MODE: after instruction 380072 given;

Function Key	Code Entering Register B		Action
	Position 1 (Zone)	Position 7 (Numeric)	
Space	0	0	Typewriter carriage spaced one character position to the left automatically.
Carriage Return	0	5	Typewriter carriage will be shifted automatically to right until left-hand margin stop is engaged; paper is simultaneously spaced vertically the amount set by manual control on print unit.
Clear Tab.	0	4	The tabulating setting will be cleared automatically at the position of the carriage when the key is pressed.
Set Tab	0	3	The tabulating setting will be set up automatically at the position of the carriage when the key is pressed.
Tab	0	2	Typewriter carriage will be moved automatically to the left to the next Tab Stop position.
Shift Key			When an additional key is simultaneously operated, the upper shift character for that key will be transferred to Register B.
Shift Lock			Causes Shift Key to be locked so that upper shift codes are transferred to Register B for any keys operated whilst lock is applied.

Figure 32: OPERATION OF FUNCTION KEYS

TYPE-OUT MODE: after instruction 380072 given;

Function Key	Code Entering Register B		Action
	Position 1 (Zone)	Position 7 (Numeric)	
Space	0	0	Typewriter carriage spaced one character position to the left.
Carriage Return	0	5	Typewriter carriage will be shifted to right until left-hand margin stop is engaged; paper is simultaneously spaced vertically the amount set by manual control on print unit.
Clear Tab	0	4	The tabulating setting will be cleared at the position of the carriage when code typed out.
Set Tab	0	3	The tabulating setting will be set up at the position of the carriage when the code is typed out.
Tab	0	2	Typewriter carriage will move to the left to the next Tab Stop position.

Figure 33: OPERATION OF FUNCTION KEYS

TYPE-OUT MODE: A 380072 instruction will cause the zone and numeric component of a character in positions 1 and 7 of Register B to be transferred to the 7-bit typewriter buffer and printed if it is a code to be typed. The buffer may however hold a code which operates the typewriter; the codes will operate the typewriter as shown in Figure 33.

Notes A 380072 instruction must only be given after the typewriter has been set to the type-in or type-out mode and after indicator 51 (Typewriter Ready, see below) has been tested and found to be set.

Typewriter Indicators

3.7.3

Indicator 50 Paper Supply Low

Purpose Indicator 50 will be set when the supply of paper in the continuous stationery feed device is low.

Operation This indicator will be set when the Paper Present micro switch has detected the absence of paper. The micro switch will be actuated when there is less than about eight inches of paper left on the platen. Indicator 50 will be unset when a new supply of paper is inserted.

Notes Indicator 50 should be tested at a convenient point in a typewriter program.

Indicator 51 Typewriter Ready

Purpose When the typewriter is in the type-in mode, indicator 51 will be set when a code is keyed in.

When the typewriter is in the type-out mode, indicator 51 will be set when the print unit is ready for operation.

Operation

TYPE-IN MODE: When the typewriter is in this mode of operation, indicator 51 will be set when a key is pressed with the exception of the Request Type-in key and the shift keys. This indicator is unset when a 380072 instruction is given. Indicator 51 will also be unset if the power supply fails.

TYPE-OUT MODE: When the typewriter is in this mode of operation, indicator 51 will be set when the print unit is ready to accept a 380072 instruction. This indicator will be unset when a 380072 instruction is given and accepted and remains unset until the action initiated by the 380072 instruction is complete. Indicator 51 will also be unset if the power supply fails.

Notes Indicator 51 must be tested and found to be set before a 380070 or 380072 instruction is given.

Indicator 52 Request Type-in

Purpose Indicator 52 will be set when the Request Type-in key is pressed.

Operation As stated above indicator 52 will be set when the Request Type-in key on the typewriter keyboard is pressed to signal that the operator is ready to use the typewriter for input purposes. Indicator 52 is unset only when tested by program.

Notes Indicator 52 can be tested at convenient points in a program to ascertain whether the typewriter is required for input purposes.

Indicator 53 Carriage at End

Purpose Indicator 53 will be set when the typewriter carriage passes the last typeable position during its movement from left to right.

Operation When the typewriter carriage reaches the right-hand margin stop, indicator 53 will be set. This indicator is unset when the typewriter carriage leaves the left-hand margin stop.

Notes Indicator 53 can be tested between the testing of indicator 51 and finding it set and giving a 380072 instruction. If it is found to be set the only acceptable orders are 'Do Nothing' or 'Line Feed'.

A counter is usually set to a figure which is equal to the number of characters or spaces for which the typewriter carriage is to be moved and a Line Feed instruction is given when this figure is reached. Thus if indicator 53 is tested and found to be set before this figure is reached, an error has occurred.

When the length of typed data required per printed line is variable, indicator 53 can be tested between the testing of indicator 51 and a 380072 instruction being given.

Indicator 59 Typewriter Mechanical Failure

Purpose Indicator 59 is set when a type bar fails to operate after a 380072 instruction.

Operation As previously described a 380072 instruction will cause the contents of positions 1 and 7 of Register B to be read out to activate a specific type bar. Should the type bar not be activated, then indicator 59 will be set. Indicator 59 will be unset when tested by program.

Notes The programming implications (which are somewhat complex) need not concern the programmer, as the testing of indicator 59 is included, as appropriate, in the various subroutines available from the Subroutine Library.

Interrogating Typewriter Programs

3.7.4

Programs for the Interrogating Typewriter are complex and somewhat involved. It is recommended that the proven subroutines be used whenever possible. Careful study of these subroutines will enable a programmer to become conversant with the programming requirements for the typewriter.

Timings

3.7.5

The *maximum* rate of operation is ten characters a second, exclusive of line feed and tabulating times etc.; a change of shift is approximately equivalent to one character time.

**THE ONE - INCH (90kc/s)
AND HALF - INCH (22½kc/s) MAGNETIC - TAPE SYSTEMS**

3.8

Magnetic-tape units can be linked to the computer and used as extra storage for program and for data words. Magnetic tape has the advantage that its storage capacity is virtually unlimited and also that reels of tape can be used for permanent storage. Unlike information recorded on the magnetic drum, information recorded on magnetic tape can be removed from the computer and preserved for use on a future run. Information cannot be written onto a tape unless the spool has a writing ring fitted onto it by the operator. This prevents the accidental overwriting of master information.

Each tape unit is capable of handling one reel of tape at a time. When the tape is in motion it passes from one tape spool past the read/write heads where information is read from or recorded on the tape, and onto another spool. From one to eight magnetic-tape units may be linked to the computer and it is possible to read from one unit and write to another unit simultaneously.

A tape control unit acts as a link between the central processor and the individual units. The tape control unit consists of a read unit and a write unit each containing a buffer store for use when reading or writing tape. Program instructions are available for controlling the tapes. Words may be transferred from magnetic tape to I.A.S. and vice versa, the transfers taking place by way of the appropriate buffer and Register A.

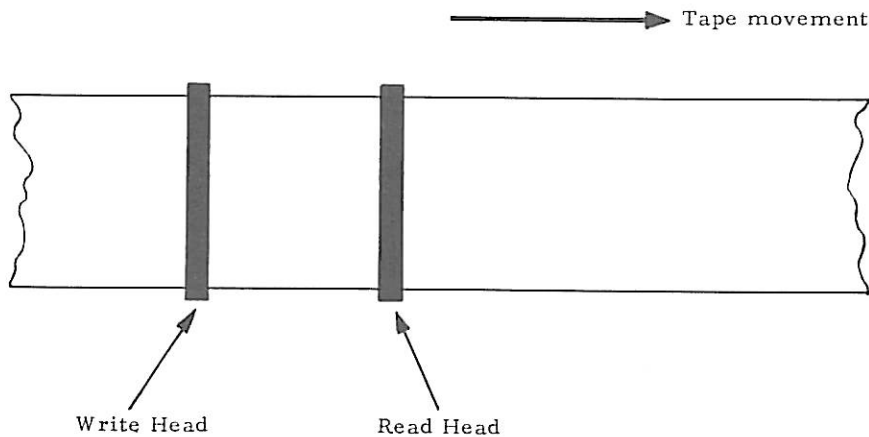


Figure 34: READ/WRITE HEADS

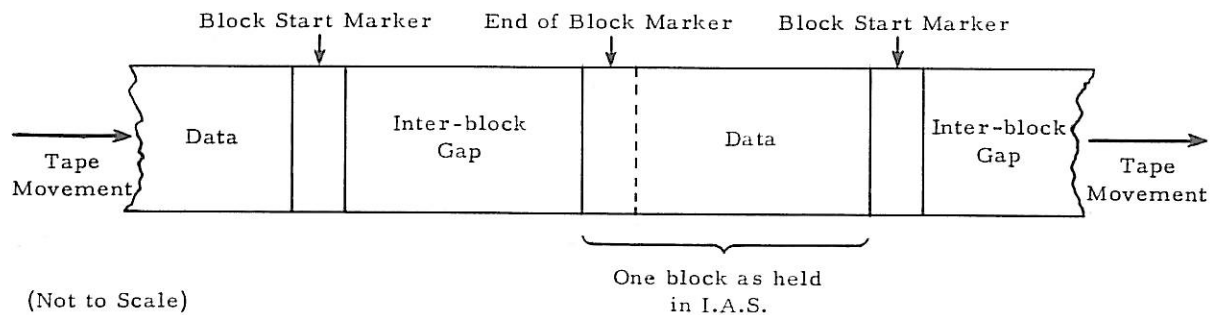
Single instructions initiate the transfer of consecutive words between tape and I.A.S., the unit of transfer being called a block. There is no restriction on the number of words in a block other than the size of I.A.S. and the remaining length of the tape. The average length of blocks should be not less than ten words.

The last word of each block is the end of block marker. This consists of a word containing data bits in all positions, i.e. a word with $\bar{15}$ in every digit position. When writing to tape the programmer must position the end of block marker in the I.A.S. word immediately following the last data word of the block. A write instruction causes writing to commence at a specified word of I.A.S. Consecutive words are written to tape up to and including the end of block marker. A read instruction causes a block to be read into I.A.S., starting at a specified word. The end of block marker is read and stored in I.A.S. as the last word of the block.

When reading or writing is taking place, the tape moves past the read/write heads at a constant speed. When a write order is obeyed, there is a slight delay before the first word of the block is written to tape. If the tape is starting from rest, then the tape is moving at full speed by the time the first word is written. When the end of block marker has been written to tape there is a delay, while the tape is still moving, during which the end of block marker is check-read. These delays during writing to tape cause gaps on the tape between consecutive blocks. These gaps are called 'inter-block gaps'. They are erased by the write head and therefore contain no information.

Immediately before the first data word of a block is written to tape, a 'block start marker' is automatically written which consists of three words of special code which are distinct from any data configuration. On subsequent reading, the block start marker indicates to the read unit that the first word of the block is about to reach the read head.

The tape layout is summarized below:



If a write instruction is given within a short period of time, called the 'fixed delay', of the previous end of block marker being checked, then writing takes place immediately without the tape stopping. In this case the inter-block gap is termed a 'short gap'.

If a write instruction is not given within the fixed delay period then writing cannot take place until the tape has stopped and re-started, resulting in a further delay while the tape comes to a stop and then accelerates again to full speed. In this case the inter-block gap is termed a 'long gap'.

During tape reading, the fixed delay is a short period after the reading of a previous end of block marker. If a read instruction is given within the fixed delay, reading takes place without the tape stopping. If a read instruction is not given in the fixed delay period, reading cannot take place until the tape has stopped and re-started. The stopping and starting takes place in the inter-block gap, the short gap being sufficiently long for this to occur.

Just before the end of a tape there is an end of tape marker. An indicator is set if this marker is detected on reading or writing tape.

There is an early end of tape marker which can be inserted in the tape at a convenient position before the end of tape marker. Thus the program has warning that the end of tape is approaching and this allows the necessary action to be taken.

Tape Organization

3.8.2

It is recommended that certain blocks should be written on the tape for organizational purposes only. It is important that the correct tape reels should be loaded for the job to be run and for this reason the first block on tape is made an identification block. This is called the 'beginning of tape label', and should be recorded on the tape when the tape is written and checked whenever it is read. There is a recommended layout for this block and also for the last block on a particular tape and for the final block of a tape file. General routines are available for reading and writing tape and for creating and checking label blocks. These 'housekeeping' routines are described in Part 5 and full details are given in the tape manual (Magnetic-tape Housekeeping Routines and Conventions).

Checking Facilities

3.8.3

Each digit is represented on the tape by four data bits; in addition to the four data bits, additional bits are generated and held on the tape for checking purposes. The arrangement of the check bits is such that the vast majority of multiple transfer errors can be detected and single-bit transfer errors can be detected precisely. Indicators are set if any errors are detected. If a single-bit error occurs on reading tape, then the error is not only detected but automatically corrected in the computer.

When information has been written to tape, the tape passes to the read head and the information read back into the computer, where it is automatically checked for transfer errors which may have occurred during writing. If an error occurs on writing tape, the tape should be backspaced and the block re-written. If the error is persistent, then that section of tape can be cancelled and the block written on the next section. Cancelled tape is ignored when it is subsequently read. Data is transferred from I.A.S. to the tape control unit from whence it is written to tape. When the end

of block marker is detected in the tape control unit, the transfer from I.A.S. ceases and the end of block marker is written to tape. If an error should occur in the writing of the end of block marker, the check-reading system will not detect the end of the block but will read the inter-block gap; when this is detected the check-reading system causes the tape to stop.

When tape is being read, single-bit errors are detected precisely and automatically corrected. If multiple-bit errors occur then the tape should be backspaced and the block re-read. Data is read from tape to the tape control unit from whence it is transferred to I.A.S. When the end of block marker is detected in the tape control unit, tape reading ceases and the end of block marker is stored in I.A.S. If a multiple error should occur in reading the end of block marker, the end of the block will not be detected. Therefore the incorrect end of block marker will be transferred to I.A.S. as a normal data word and the inter-block gap will be read. The inter-block gap is detected on reading and this will cause the tape to stop. If a single-bit error occurs on reading the end of block marker, this is automatically corrected and the end of block is detected in the normal manner.

Tape Deck Addresses and Queueing

3.8.4

Each tape unit has an eight-point rotary switch which enables the unit to be allocated an address in the range 1 to 8. This applies to all tape units irrespective of the number of units linked to the computer. The tape deck address is specified in magnetic-tape instructions.

When the loading spool is correctly fitted, the operation of the Allocate button on the tape unit causes the address set on the eight-point switch to be seized by that unit, provided the address is not already allocated to another unit. When the tape unit has seized an address it is under computer control and is governed by instructions referring to that deck address.

If a second unit is allocated an address which has already been seized, then it queues and waits until an unload instruction is given to that address. The first unit then ceases to be under computer control and the second unit automatically seizes the address and comes under computer control.

A tape unit may be queued at any time provided that there is no other tape unit already in the queue.

Any attempt to allocate the same deck address to a third unit will have no effect.

Magnetic - tape Instructions

3.8.5

All magnetic-tape instructions have function 39, the various instructions being distinguished by their address digits. A summary of the instructions is given in Figure 35.

Tape Write Instruction

Effect This instruction causes one block to be written from I.A.S. to magnetic tape.

Operation Starting at a specified word of I.A.S., successive words are written on tape, up to and including the end of block marker.

Notes The instruction is double-length and is made up as follows:

First half	Function digits	-	39
	First two address digits	-	00
	Third address digit	-	1
	Fourth address digit	-	Tape deck address - i.e., a number in the range 1 to 8
Second half	Function digits	-	00
	Address digits	-	Address of first I.A.S. word to be written to tape.

Thus the instruction

D	F	A	R
	39	0012	
	00	0035	10

causes a block to be written to deck address 2 starting at word 35 block 10.

During the execution of this instruction the write unit is occupied. Further instructions to the specified tape deck or write or cancel instructions to any tape deck cause the Tape Order Error indicator to be set.

Tape Read Instruction

Effect This instruction causes one block to be read into I.A.S. from magnetic tape.

Operation Information from tape is stored in successive words of I.A.S. up to and including the end of block marker.

Notes The instruction is double-length and is made up as follows:

- | | | | |
|-------------|--------------------------|---|---|
| First half | Functions digits | - | 39 |
| | First two address digits | - | 00 |
| | Third address digit | - | 2 |
| | Fourth address digit | - | Tape deck address -
i.e., a number in the range 1 to 8 |
| Second half | Function digits | - | 00 |
| | Address digits | - | Address of first I.A.S. word in
which information is to be stored. |

Thus the instruction

D	F	A	R
	39	0021	
	00	0050	37

causes a block to be read from deck address 1 and stored in I.A.S. starting at word 50 block 37.

During the execution of this instruction the read unit is occupied. Further instructions to the specified tape deck or read or backspace instructions to any tape deck cause the Tape Order Error indicator to be set.

Backspace Instruction

Effect This instruction causes the tape to be backspaced one block.

Operation The tape on a specified deck is rewound one block. When it has stopped the tape is positioned so that the block can be read, written or cancelled. The previous block is unaltered.

Notes The instruction is a single-length instruction as follows:

- | | | |
|--------------------------|---|-------------------|
| Function digits | - | 39 |
| First two address digits | - | 00 |
| Third address digit | - | 3 |
| Fourth address digit | - | Tape deck address |

Thus the instruction

D	F	A	R
	39	0034	

causes the tape on deck address 4 to be rewound one block.

During the execution of this instruction the read unit is occupied. Further instructions to the specified tape deck or read or backspace instructions to any tape deck cause the Tape Order Error indicator to be set.

After backspacing the read unit becomes ready when the block start marker is detected. The Deck Address Ready indicator does not become set until the tape movement is stopped. Therefore the Deck Address Ready indicator should be tested after a backspace instruction rather than the Read Unit Ready indicator. When the Deck Address Ready indicator is set, the tape deck is ready to accept another instruction.

Cancel Instruction

Effect This instruction causes a block of tape to be cancelled.

Operation The tape is cancelled by writing zeros onto certain of the data and check bits, forming a pattern which is distinct from any possible data configuration. One block is cancelled, the end of the block being detected by the check-reading of the inter-block gap. The latter contains a zero in all bit positions and is therefore distinct from cancelled tape.

Cancelled tape is ignored by subsequent read or backspace instructions and no error indicators are set.

Notes The instruction is a single-length instruction as follows:

- Function digits - 39
- First two address digits - 00
- Third address digit - 4
- Fourth address digit - Tape deck address

Thus the instruction

D	F	A	R
	39	0041	

causes a block of tape to be cancelled on tape deck address 1.

A cancel instruction should always be preceded by a backspace or rewind instruction.

During the execution of the cancel instruction the write unit is occupied. Further instructions to the specified tape deck or write or cancel instructions to any tape deck cause the Tape Order Error indicator to be set.

Rewind Instruction

Effect This instruction causes the tape on a specified deck to be rewound to the beginning of the tape.

Operation The tape on the specified deck is rewound to the beginning of the tape. A subsequent read or write instruction causes the first block on tape to be read or a new block to be written.

Notes The instruction is a single-length instruction as follows:

Function digits - 39
First two address digits - 00
Third address digit - 5
Fourth address digit - Tape deck address

Thus the instruction

D	F	A	R
	39	0056	

causes the tape on tape deck address 6 to be rewound to the beginning of the tape.

While the tape is being rewound, any further instruction to the same tape deck causes the Tape Order Error indicator to be set. Any tape instructions can be carried out on other decks.

Unload Instruction

Effect This instruction causes the tape on a specified tape deck to be rewound ready for removal from the unit.

Operation The tape on the specified deck is rewound on the loading spool and the spool made ready for removal. When a tape has been unloaded the tape deck is no longer under computer control.

Notes The instruction is a single-length instruction as follows:

Function digits - 39
First two address digits - 00
Third address digit - 6
Fourth address digit - Tape deck address

Thus the instruction

D	F	A	R
	39	0062	

causes the tape on tape deck address 2 to be rewound on the loading spool ready for removal.

Any instruction to a deck for which the unload instruction has been accepted causes the Tape Order Error indicator to be set. Any tape instructions can be carried out on other decks. If the deck is queued by another deck, then the unload instruction causes the deck address to be seized by the second deck. The second deck is then controlled by instructions specifying that deck address.

The following notes deal with consecutive instructions to the same tape deck:

If a read instruction follows a write or cancel instruction no assumption can be made about the information which is read. This is because of certain tolerances on the lengths of the inter-block

Digit Positions	1	2	3	4	5	6	7	8	9	10	11	12
Instruction	Function		0	0	Command	Deck Address	0	0	I.A.S.			
Write	3	9	0	0	1	1-8	0	0	I.A.S. Address			
Read	3	9	0	0	2	1-8	0	0	I.A.S. Address			
Backspace	3	9	0	0	3	1-8						
Cancel	3	9	0	0	4	1-8						
Rewind (Back to start of spool)	3	9	0	0	5	1-8						
Unload (Tape rewound on spool)	3	9	0	0	6	1-8						

} Double Length Instructions
 } Single Length Instructions
 (May be in either half of word)

Figure 35: SUMMARY OF MAGNETIC-TAPE INSTRUCTIONS, AS HELD IN THE COMPUTER

gaps. For example, if a block is overwritten by another block of the same data length then the second block may occupy slightly more tape than the original, thus overwriting the beginning of the next block.

Errors may also arise if a write instruction follows a read instruction to the same unit. This is because of the distance between the write head and the read head. When a block has been read the tape stops with the read head positioned in the inter-block gap. Since the read head follows the write head, however, some of the tape following the inter-block gap may already have passed the write head. This means that part of a previously recorded block may be left on the tape between the block just read and the block being written, thus causing errors when the tape is subsequently read. To avoid this, a write or cancel instruction should not follow a read instruction to the same deck, unless the relevant part of the tape has been previously cancelled, or the whole tape has been previously erased (by an engineer).

A cancel instruction should normally be used only if a section of tape is found to be persistently faulty. When the errors are discovered, the relevant block has passed the read/write heads and the tape must be backspaced so that the correct section is cancelled. A cancel instruction should therefore always be preceded by a backspace instruction.

The following table summarizes the instructions which may follow each other:

First instruction	Next instruction to the same tape deck			
	Write	Read	Backspace	Cancel
Write	Yes	No	Yes	No
Read	Yes if tape previously cancelled	Yes	Yes	No
Backspace	Yes	Yes	Yes	Yes
Cancel	Yes	No	Yes	No

Indicators 81 to 88 Deck Address Ready

Purpose These indicators are used to test if a tape deck is ready for use, before giving instructions to operate it. Indicators 81 to 88 may also be used to test whether a read/write operation has been completed prior to checking it.

Operation There are eight of these indicators, one for each possible deck address. The indicators are numbered 81 to 88, and are associated with deck addresses 1 to 8 respectively.

The Deck Address Ready indicator is automatically set if:

- (a) The appropriate deck address has been seized by a tape unit, and
- (b) the tape deck is not busy, i.e., no tape instruction is currently being executed on that deck, and
- (c) the tape deck is mechanically ready.

The indicator is unset if any of the above conditions are not satisfied, indicating that the deck is not ready to receive an instruction.

Notes For example the instruction

D	F	A	R
4	83	0012	B

tests whether deck address 3 is ready for use, if it is ready a jump is made to word 12 of the current block.

The appropriate Deck Address Ready indicator should be tested before giving a tape instruction. This prevents a tape order error due to an instruction being given to a deck which is not ready.

Indicator 89 Transport Mechanically Ready

Purpose This indicator is used in conjunction with the Deck Address Ready indicators. If one of the indicators 81 to 88 is unset, this indicator can be used to discover which of the possible conditions caused it to be unset.

Operation Indicator 89 is automatically set if:

- (a) the last Deck Address Ready indicator to be tested was found to be set, or
- (b) the last Deck Address Ready indicator was unset because the tape deck was busy.

The indicator is automatically unset if the last Deck Address Ready indicator was unset:

- (a) because the address had not been seized by a tape deck, or
- (b) because the tape deck was not mechanically ready.

Notes Indicator 89 should be tested when one of the indicators 81 to 88 is found to be unset.

If indicator 89 is set the tape deck is busy and the program must wait until the deck is not busy before giving it an instruction.

If indicator 89 is unset, operator action is required. The address has not been correctly allocated or there is a mechanical failure. Thus the instructions:

I	D	F	A	R
5	4	81	0007	B
	4	89	0005	B
6		11	2002	

will cause the following:

If deck address 1 is ready for use a jump is made to word 7 of the block which continues with the tape program.

If deck address 1 is not ready indicator 89 is tested. If indicator 89 is set the tape deck is busy, so a jump is made to re-test indicator 81 to see if the deck is no longer busy. If indicator 89 is unset the computer is stopped with 2003 displayed in CR3. This indicates to the operator that action must be taken.

If a tape deck becomes mechanically unready, indicator 89 will become unset after a delay of up to 500 milliseconds.

Indicator 80 Tape Order Error

Purpose This indicator is set if a tape instruction is given which cannot be accepted.

Operation Indicator 80 is automatically set if:

- (a) A tape instruction is given to a deck address which has not been seized by a tape unit.
- (b) An instruction is given to a tape deck which is not mechanically ready.
- (c) An instruction is given to a tape deck which is busy.
- (d) A write or cancel instruction is given to any tape deck while either instruction is already being executed.
- (e) A read or backspace instruction is given to any tape deck while either instruction is already being executed.
- (f) A write or cancel instruction is given to a deck which has not had a writing ring fitted to the tape spool.

The indicator is unset by program test.

Notes When indicator 80 is set the Tape Order Error light on the console is lit. If the Optional Stop switch is on, the computer stops automatically when there is a tape order error, CR3 containing the faulty instruction with 1 added to it. In the latter case starting the computer causes the Tape Order Error light to go out, but the indicator remains set until tested by program.

The setting of the indicator normally indicates that a program error has occurred. The correct testing of the other tape indicators should ensure that indicator 80 is never set.

Indicator 70 Write Unit Ready

Purpose This indicator is used to test whether the write unit is busy.

Operation Indicator 70 is set automatically when a write or cancel instruction has been completed, indicating that the write unit is ready to accept another instruction.

It is unset automatically when a write or cancel instruction is accepted.

Notes Indicator 70 should be tested before a write or cancel instruction to ensure that the write unit is ready and to prevent a possible tape order error. For example instructions:

I	D	F	A	R
10	4	70	11	B
	4	00	10	B
11		39	0013	
		00	0005	15

will cause the following:

Indicator 70 is repeatedly tested until the write unit is ready. When the write unit is ready the instruction to write to deck address 3 is obeyed.

Indicator 72 Read Unit Ready

Purpose This indicator is used to test whether the read unit is busy.

Operation Indicator 72 is set automatically when a read or backspace instruction has been completed, indicating that the read unit is ready to accept another instruction.

It is unset automatically when a read or backspace instruction is accepted.

Notes Indicator 72 should be tested before a read or backspace instruction to ensure that the read unit is ready and to prevent a possible tape order error.

Indicator 74 Any Errors

Purpose This indicator is used to detect whether any transfer errors have occurred during writing or reading tape.

Operation Indicator 74 can be associated with writing or reading. When the Write Unit Ready or Write Master indicators are tested, indicator 74 is associated with writing. When the corresponding read indicators are tested, it is associated with reading. For explanation purposes indicator 74 is referred to as W74 when writing and R74 when reading and this convention also applies to indicators 75, 76 and 77.

W74 is set if any errors are detected when information is written to tape.

W74 is automatically unset when a write or cancel instruction is accepted.

R74 is set if any error is detected on reading. R74 is set even if the error is a single-bit error which has been automatically corrected.

R74 is unset automatically when a read instruction is accepted.

Notes R74 is unaffected when a section of tape is read which has been previously cancelled. Indicator 74 is associated with writing or reading according to the last Unit Ready or Master indicator to be tested. A test of indicator 74 should therefore be immediately preceded by a test of one of the indicators 70, 71, 72 or 73.

Indicator 75 Multiple Errors

Purpose This indicator is used to detect any multiple-bit transfer errors which occur during writing or reading tape.

Operation Indicator 75 can be associated with writing or reading. When the Write Unit Ready or Write Master indicators are tested, indicator 75 is associated with writing. When the corresponding read indicators are tested, it is associated with reading. Indicator 75 is referred to as W75 when writing and R75 when reading.

W75 is set if any errors other than single-bit errors occur on writing.

W75 is automatically unset when a write or cancel instruction is accepted.

R75 is set if any errors other than single-bit errors occur on reading.

R75 is unset when a read instruction is accepted.

Notes R75 is unaffected when a section of tape is read that has previously been cancelled. Indicator 75 is associated with writing or reading according to the last Unit Ready or Master indicator to be tested. A test of indicator 75 should therefore be immediately preceded by a test of one of the indicators 70, 71, 72 or 73.

Indicator 76 End of Tape

Purpose This indicator is used to detect the end of tape marker.

Operation Indicator 76 can be associated with writing or reading. When the Write Unit Ready or Write Master indicators are tested, indicator 76 is associated with writing. When the corresponding read indicators are tested, it is associated with reading. Indicator 76 is referred to as W76 when writing and R76 when reading.

W76 is set when the end of tape marker is detected during writing or cancelling.

W76 is unset by program test.

R76 is set when the end of tape marker is detected during reading.

R76 is unset by program test.

Notes Indicator 76 is associated with writing or reading according to the last Unit Ready or Master indicator to be tested. A test of indicator 76 should therefore be immediately preceded by a test of one of the indicators 70, 71, 72 or 73.

Indicator 77 Early End of Tape and Short Block

Purpose This indicator is used to detect the early end of tape marker or to detect a short block consisting of four words.

Operation Indicator 77 can be associated with writing or reading. When the Write Unit Ready or Write Master indicators are tested, indicator 77 is associated with writing. When the corresponding read indicators are tested, it is associated with reading. When writing, indicator 77 is called the Early End of Tape indicator and is referred to as W77. When reading, indicator 77 is called the Short Block indicator and is referred to as R77.

W77 is set when the early end of tape marker is detected during writing or cancelling.

W77 is unset by program test.

R77 is set when a short block, consisting of exactly four words including the end of block marker, is detected on reading. Short blocks may be used for identification purposes and are used as label blocks by the housekeeping routines.

R77 is unset by program test.

Notes Indicator 77 is associated with writing or reading according to the last Unit Ready or Master indicator to be tested. A test of indicator 77 should therefore be immediately preceded by a test of one of the indicators 70, 71, 72 or 73.

Indicator 71 Write Master

Purpose This indicator is used to test if writing has been successfully completed.

Operation Indicator 71 is set if any of the indicators W74, W76 or W77 is set, i.e., if:

- (a) any errors have been detected on writing, or
- (b) if the end of tape marker has been detected during writing, or
- (c) if the early end of tape marker has been detected during writing.

Indicator 71 is unset when W74, W76 and W77 are all unset.

Notes If indicator 71 is unset it follows that writing has been successful and no special action need be taken. Only if indicator 71 is set is it necessary to test W74, W76 and W77 to establish which exceptions condition has arisen.

Indicator 73 Read Master

Purpose This indicator is used to test whether reading has been successfully completed.

Operation Indicator 73 is set if any of the indicators R75, R76 or R77 is set, i.e., if:

- (a) Multiple errors have been detected while reading.
- (b) The end of tape marker is detected on reading.
- (c) A short block has been read.

Indicator 73 is unset when R75, R76 and R77 are all unset.

Notes If indicator 73 is unset it follows that reading has been successful (single-bit errors being corrected automatically) and no special action need be taken. Only if indicator 73 is set is it necessary to test R75, R76 and R77 to establish which exceptions condition has arisen.

INDICATOR	TITLE	SET BY	UNSET BY
70	Write Unit Ready	Write Unit not busy	Write or cancel instruction
71	Write Master	W74, W76, W77 becoming set	Unsetting of all three indicators
72	Read Unit Ready	Read Unit not busy	Read or backspace instruction
73	Read Master	R75, R76, R77 being set	Unsetting of all three indicators
74	W74 Write Any Errors	Any bit errors written	Write or cancel instruction
	R74 Read Any Errors	Any bit errors read	Read instruction
75	W75 Write Multiple Errors	Multiple bit errors written	Write or cancel instruction
	R75 Read Multiple Errors	Multiple bit errors read	Read instruction
76	W76 Write Final End of Tape	Final end of tape marker	Program test
	R76 Read Final End of Tape	Final end of tape marker	Program test
77	W77 Write Early End of Tape	Early end of tape marker	Program test
	R77 Read Short Block	Short block read	Program test
79	Writing Ring Present	Writing ring present on spool and transport mechanically ready and address seized on deck last tested.	Writing ring not present on spool transport not mechanically ready or address not seized on deck last tested
80	Tape Order Error	Unacceptable instruction	Program test
81 to 88	Deck Address (1-8) Ready	Address seized and tape deck not busy and mechanically ready	Address not seized or tape deck busy or not mechanically ready
89	Transport mechanically ready	Transport mechanically ready and address seized on deck last tested.	Transport not mechanically ready or address not seized on deck last tested.

Figure 36: SUMMARY OF MAGNETIC-TAPE INDICATORS

Indicator 79 Writing Ring Present

Purpose This indicator is used to ensure that a writing ring has been fitted before a write instruction is given.

Operation This indicator is associated with the last Deck Address Ready indicator to be tested.

Indicator 79 is set if a writing ring is present on the tape deck for which the last Deck Address Ready indicator was tested and if indicator 89 (Transport Mechanically Ready) is set.

It is unset if the writing ring is absent, or if indicator 89 (Transport Mechanically Ready) is unset. If a tape deck becomes mechanically unready, indicator 79 will become unset after a delay of up to 500 milliseconds.

Notes Indicator 79 should be tested before writing to tape. This ensures that a writing ring has been fitted and prevents a possible tape order error.

The Secure lamp on the tape unit is lit when no writing ring is present on the loaded reel.

Program Interrupt Facility

3.8.7

Once a tape instruction has been initiated it is carried out automatically and does not require constant program control. This means that other programs can be obeyed while the tape is in motion. An automatic interrupt facility is incorporated to cater for the transfer of words between I.A.S. and the tape control unit. Thus the need for any program control during the reading or writing of tape is eliminated. This enables the programmer to make maximum use of the time available while the tape is in motion.

When a rewind or unload instruction is being obeyed, there is no transfer between I.A.S. and magnetic tape and thus any instruction may be obeyed other than a tape instruction to the same deck.

During a cancel or backspace instruction there is no transfer between I.A.S. and magnetic tape and any instructions can be obeyed other than drum transfers or tape instructions to the same deck. Drum transfers cannot be obeyed since they use the same clocking system as the tape decks.

Reading Tape

Information read from tape is transferred into a special register, Register G. Register G is a register used as a buffer store when tape reading is taking place, and can be considered as being in two halves, each consisting of two 6-digit registers and a small buffer. The layout of the registers is shown diagrammatically in Figure 37.

Information is read from tape into the small buffers and thence it is transferred into the two 6-digit registers.

When a complete word is contained in the 6-digit registers the tape control unit indicates to the central processor that it is ready to interrupt the main program in order to transfer the word to I.A.S.

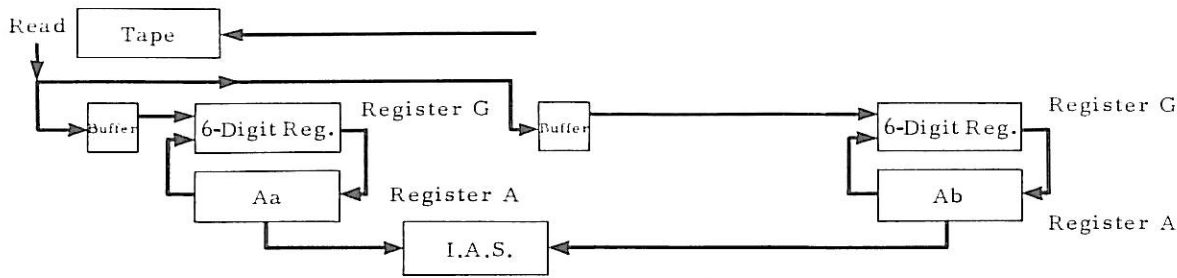


Figure 37: TAPE READ

When the current program instruction has been completed the tape control unit breaks in. The contents of the registers are circulated so that they are interchanged with the contents of Register A. The data from tape is now contained in Register A and is transferred to I.A.S. When the transfer has been completed, the contents of the registers are again interchanged with those of Register A. The original contents of Register A have thus been restored and the main program continues until the next word has been read into the registers, when it is again interrupted.

When the tape control unit is ready to break in, the program is interrupted at the end of the instruction it is obeying. If it is obeying a multibeat instruction, e.g. multiplication, the break-in takes place at the end of the micro-instruction currently taking place. Drum transfers cannot be obeyed while reading is taking place since they use the same clocking system as the tape decks and also take longer than the frequency at which the break-in occurs, and cannot be interrupted in the middle of execution. Great care should be taken if other peripheral units are used while the tape is moving since the interruptions from the tape control unit upset the timings. In particular the full P.P.F. cannot be used during tape-reading and printing should not take place while reading one-inch (90 kc/s) tape. Processing should not take place on the current block while it is still being read since, if any transfer error occurs, the whole block is re-read.

Writing Tape

Information to be written to tape passes through a register, Register F. Register F is used as a buffer store and can be considered as being in two halves, each consisting of a 6-digit register and a small buffer. The layout of the registers is shown diagrammatically in Figure 38.

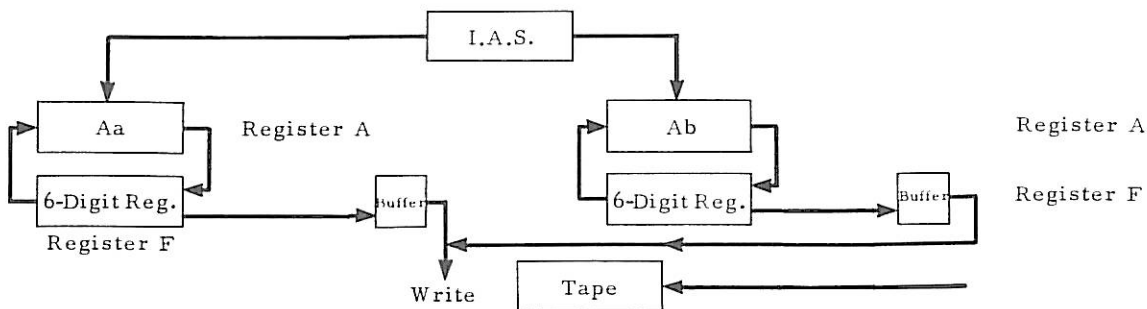


Figure 38: TAPE WRITE

When a word is to be written to tape the contents of the 6-digit registers are interchanged with those of Register A. A word is then written from I.A.S. to Register A and the contents of the registers again interchanged. The original contents of Register A have thus been restored and the word to be written to tape is held in the two 6-digit registers. The word is written to tape via the small buffers. When the 6-digit registers have become empty the tape control unit is ready to receive another word from I.A.S. and a request to break in is made to the central processor.

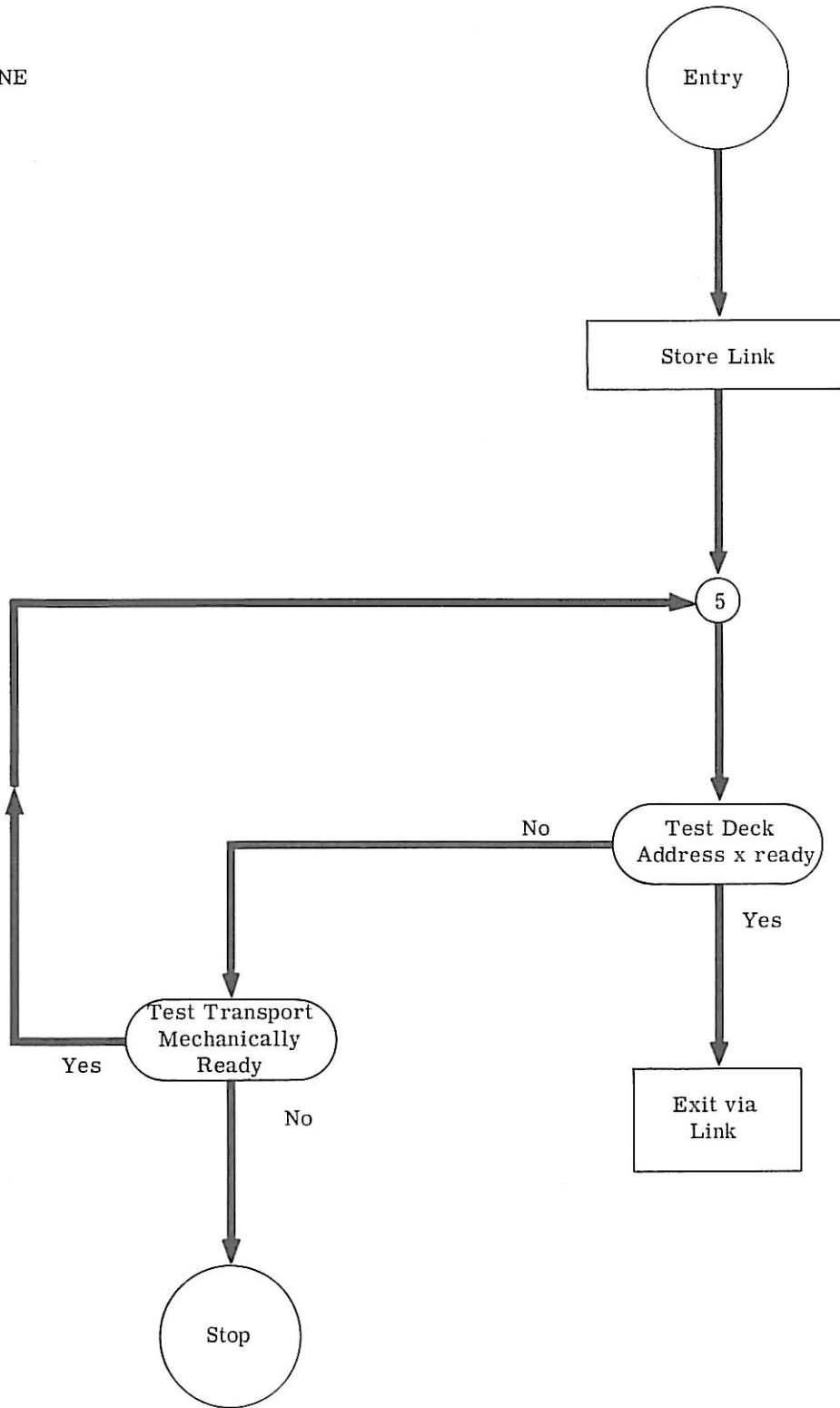
The program is interrupted at the end of the instruction it is obeying, or at the end of the micro instruction if it is obeying a multibeat instruction. Drum transfers cannot be obeyed while writing is taking place as they use the same clocking system as the tape decks and also take longer than the frequency at which the break-in occurs and cannot be interrupted in the middle of execution. Great care should be taken if other peripheral units are used while the tape is moving since the interruptions from the tape control unit upset the timings. In particular the full P.P.F. cannot be used during tape writing and printing should not take place while writing one-inch (90 kc/s) tape. Processing should not take place on the block which is being written to tape since if any transfer error occurs, the whole block is re-written.

It is possible to read from one tape deck and write to another tape deck simultaneously. In this case the main program is interrupted by both the read unit and the write unit, the read unit taking preference if they should both require to break in at the same time.

The time-sharing with a tape read or write program is effected as follows:

When the read or write instruction has been given, a jump is made from the tape program to the section of program which is to be time-shared with the reading or writing. This program is then executed with occasional automatic interruptions from the tape control unit when a transfer involving I.A.S. occurs. When the program has been completed a return is made to the tape program which, when the transfer has been completed, tests the necessary indicators to ensure that the block has been correctly read or written. If a section of the main program takes longer to complete than the reading or writing of the block, the tape will have stopped before control is returned to the tape program; this however does not matter. If drum transfers occur within the time-sharing program they should be preceded by tests of Read and Write Units Ready indicators (72 and 70) to check that the tape transfer has been completed. Facilities for time-sharing when reading or writing tape are provided with the tape housekeeping routines.

DECK
TESTING
SUBROUTINE



Deck not correctly allocated
or there is a mechanical failure.
Correct fault and restart from beginning.

Flowcharts For Reading and Writing Tape

3.8.8

Flowcharts for reading and writing magnetic tape are included as an illustration of the use of the various indicators and instructions. These flowcharts do not cover all the housekeeping procedures in the standard routines and are intended as an example only. It is assumed that beginning of tape labels have already been written or checked. Reading and writing takes place on tape deck address x . The value of x would be entered as a parameter at the start of the routine.

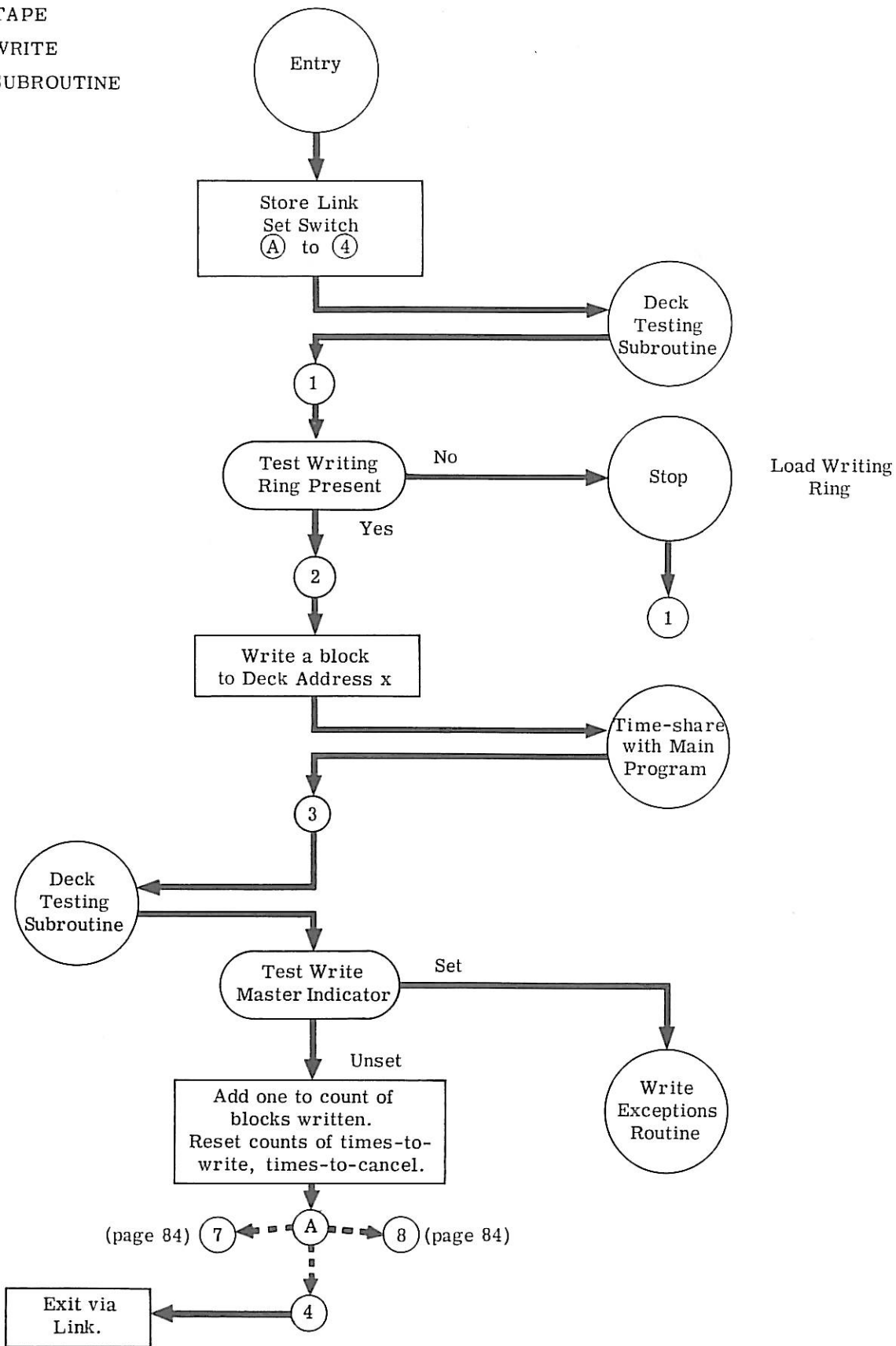
Deck Testing Subroutine

This routine determines if deck address x is busy. It may be used to ensure that a tape deck has finished obeying the current instruction and is ready to receive a further instruction. The use of this routine prevents any tape order errors arising due to instructions being given to decks which are busy.

To ensure that tape deck address x is mechanically ready the appropriate Deck Address Ready indicator and indicator 89 (Transport Mechanically Ready) are repeatedly tested until the deck ceases to be busy and is ready to receive another instruction. If the tape deck is not mechanically ready, then the deck has not been correctly allocated or there is a mechanical failure. The fault should be corrected and the job restarted from the beginning.

The Deck Testing routine has been made into a subroutine since it is used in several places by the other routines.

TAPE
WRITE
SUBROUTINE



Tape Write Subroutine

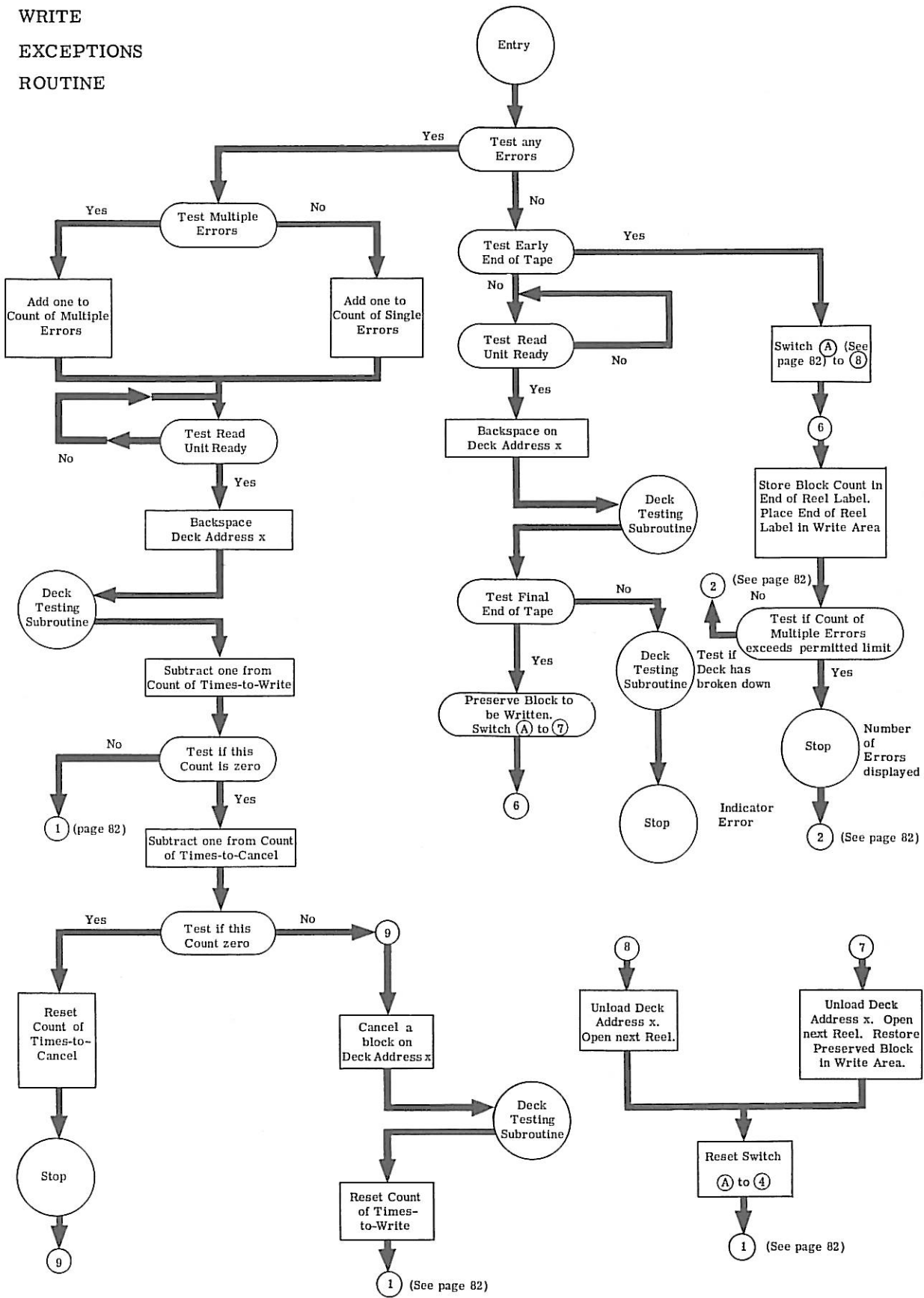
This routine writes a block from I.A.S. to magnetic tape on deck address *x*. A test is made to ensure that writing has been successfully completed and a count (initially zero) of the number of blocks which have been written is updated.

A test is made to ensure that a writing ring has been fitted before an attempt is made to write. The Deck Testing routine is entered before this test since it tests the Deck Address Ready indicator and thus associates the Writing Ring Present indicator with the correct deck address.

When the write instruction has been given, an exit is made so that the writing routine can be time-shared with the main program. On return to the write routine, the Deck Testing routine is entered to test whether writing has been completed. When the block has been written, indicator 71 (Write Master) is tested. If it is unset then there have been no writing errors and no special conditions have arisen, and the block count is updated. The times-to-write and times-to-cancel counts are used when transfer errors occur.

The times-to-write count corresponds to the number of attempts which must be made to write on a particular section of tape before cancelling that section. The times-to-cancel count corresponds to the number of sections of tape upon which attempts are made to write before giving an indication that the tape is faulty. If indicator 71 (Write Master) is set, then an error (or exceptional case) has arisen and the Write Exceptions routine is entered.

WRITE
EXCEPTIONS
ROUTINE



Write Exceptions Routine

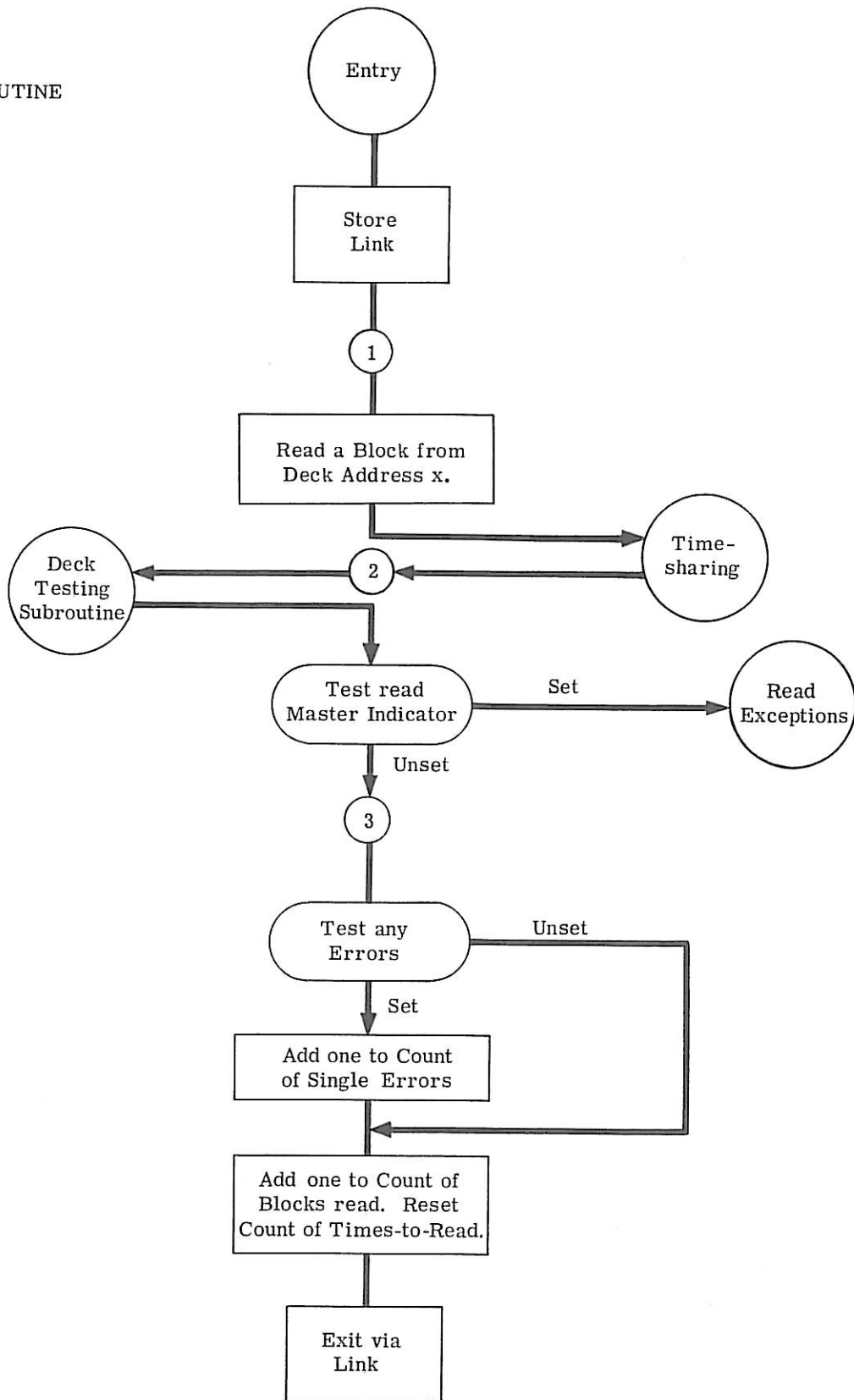
The Write Exceptions routine discovers which exceptions condition has arisen and deals with it appropriately.

A count is kept (initially zero) of any single- or multiple-bit errors which occur. If an error occurs, the tape is backspaced and another attempt made to write on the same section of tape. If necessary several attempts are made until the times-to-write count is reduced to zero. When the count is reduced to zero the block is cancelled and writing is attempted on the next section. If the times-to-write count is reduced to zero on the next section then this too is cancelled. Attempts are made, if necessary, on successive sections until the times-to-cancel count has been reduced to zero. When the count is reduced to zero the computer stops with an indication that there are repeated errors. Restart causes further attempts to be made on following sections of tape.

If the early end of tape marker is detected then a short block is written to tape as the last block on the reel. This is called the end of reel label and has a special format to distinguish it from other short blocks. The block count is stored in the end of reel label. A test is made to discover whether the multiple errors have exceeded the permitted number. If the permitted number is exceeded the computer stops with the number of single and multiple errors displayed. When the end of reel label has been correctly written, deck address x is unloaded and a new reel is opened.

The final end of tape should never be detected, since the end of reel label is written immediately after the block in which the early end of tape marker was read. If, however, the final end of tape is detected, the block which has just been written is preserved. The tape is backspaced and the end of reel label is written in place of the last block. When the new reel has been opened the preserved block is written as the first data block of the new spool.

TAPE
 READ
 SUBROUTINE

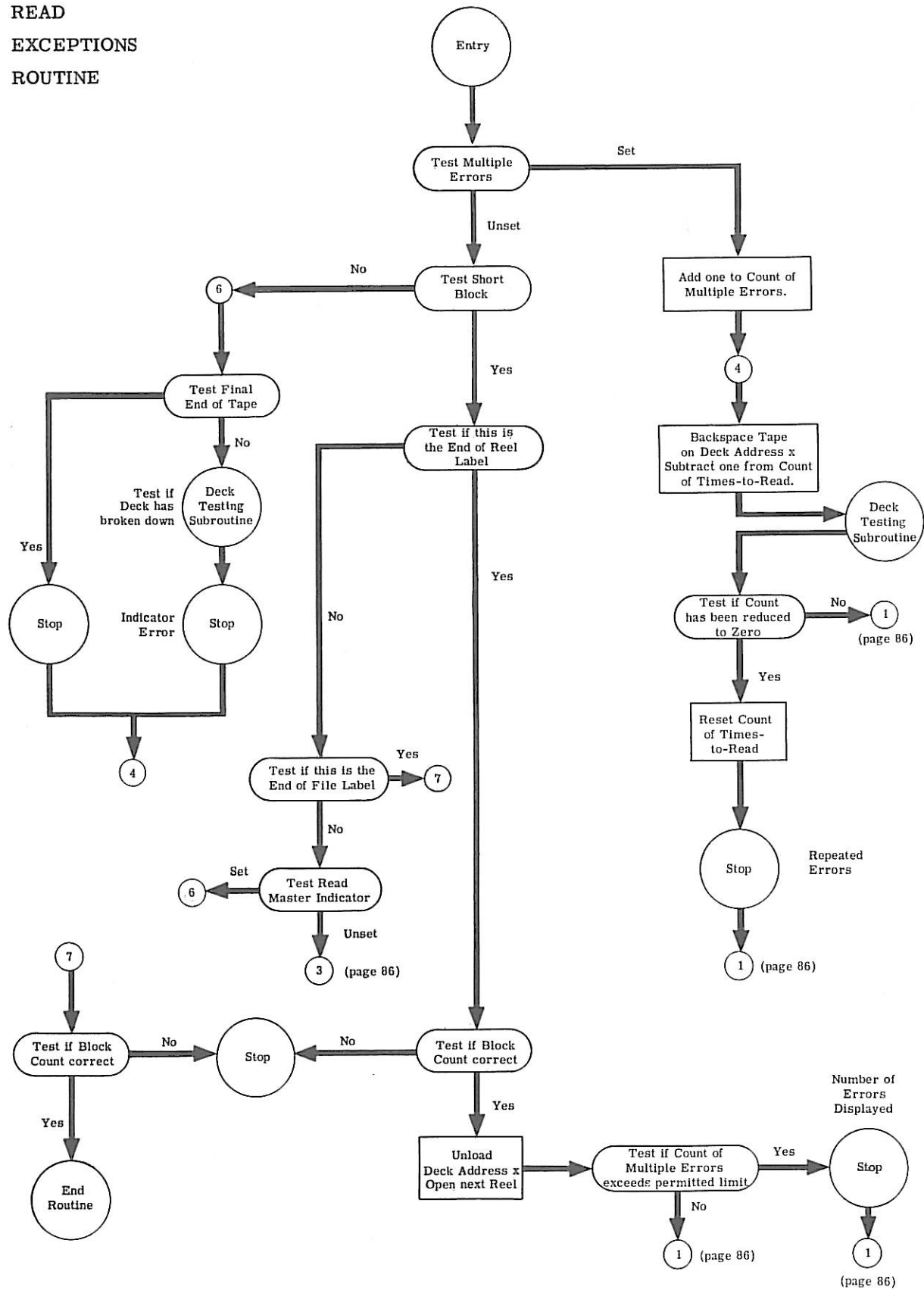


Tape Read Subroutine

This routine reads a block from deck address *x* and stores it in I.A.S. A test is made to ensure that reading has been successfully completed and a count (initially zero) of the number of blocks which have been read, is updated.

When the read instruction has been given, an exit is made so that the read routine can be time-shared with the main program. When the block has been completely read, indicator 73 (Read Master) is tested. If this indicator is unset no special conditions have occurred. Indicator R74 (Any Errors) is then tested to determine if any single-bit errors have occurred, these having been automatically corrected and requiring no special action. The count of single-bit errors is updated if necessary and the number of blocks read count is also updated. A times-to-read count is used when errors occur. This count corresponds to the number of attempts which are to be made to read a block before the attempt is abandoned and the computer stopped. If indicator 73 is set, then a multiple error (or exceptional case) has arisen and the Read Exceptions routine is entered.

READ
EXCEPTIONS
ROUTINE



Read Exceptions Routine

The Read Exceptions routine discovers which exceptions condition has arisen and deals with it appropriately. A count is kept of any multiple errors which occur.

If a multiple error occurs the tape is backspaced and another attempt made to read the block. If necessary, several attempts are made until the times-to-read count is reduced to zero. The computer is then stopped as an indication that the tape deck may be faulty. Restart causes further attempts to be made.

If a short block is detected, it is examined to see if it is the end of reel or end of file label. The end of file label is written after the last data block. It acts as an end of reel label and in addition indicates that there is no further reel to follow. When the end of reel or end of file label is detected a check is made to ensure that the number of blocks read corresponds to the block count stored in the label. If a short block is not an end of reel or end of file label, then it is treated as an ordinary data block.

If the final end of tape label is detected the computer is stopped. This condition should never arise since the end of reel label should always be read before the final end of tape label.

THE ONE-INCH (90 kc/s) MAGNETIC-TAPE SYSTEM

3.9

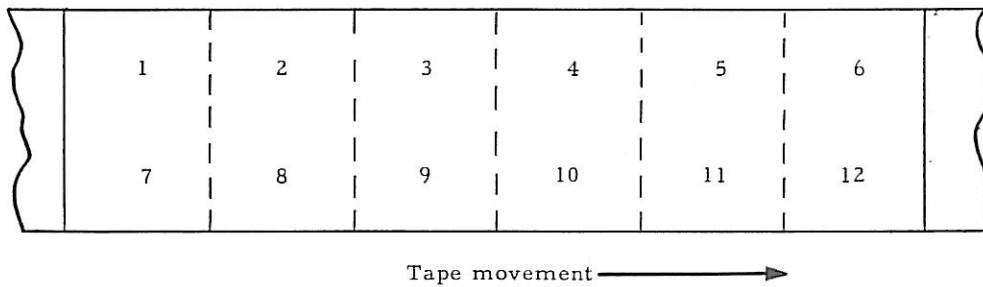
The magnetic tape is one inch wide and is divided into 16 longitudinal tracks. Data is recorded on the tape in the form of bits. Thus a cross-section of the tape consists of 16 bit positions and is called a frame. The tape moves at a speed of 150 inches a second. The packing density is 600 digits (50 words) to the inch and the tape digit transfer speed is therefore 90,000 digits a second (90 kc/s). The time for one word to pass the read/write heads is 133 microseconds.

The maximum length of tape on a spool is 3,600 feet. The length of tape between the end of tape marker and the end of the tape is a minimum of 15 inches. The length of tape between the early end of tape marker and the end of tape marker is a minimum of 15 feet.

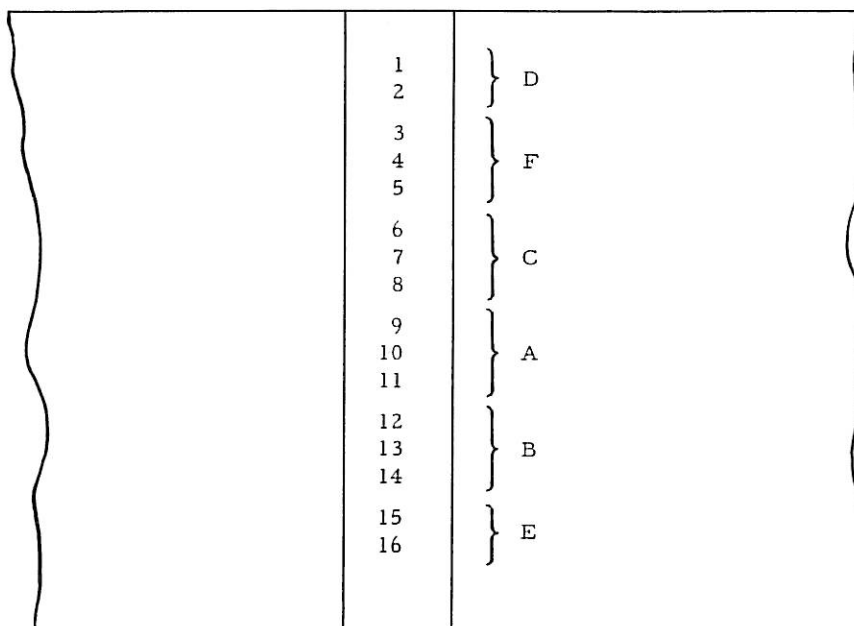
Representation of Data on Tape

3.9.1

One tape frame contains eight data bits and eight check bits. The eight data bits comprise two four-bit digits and it follows that one word of data occupies six tape frames. The digits of a word are written on the tape as follows:



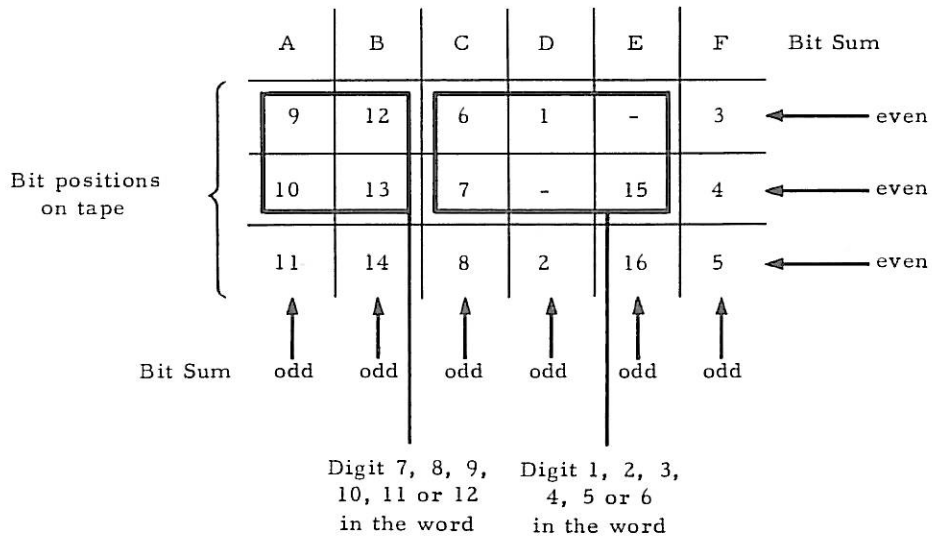
The bit positions in one tape frame are grouped as follows:



The groups are lettered for explanation purposes only.

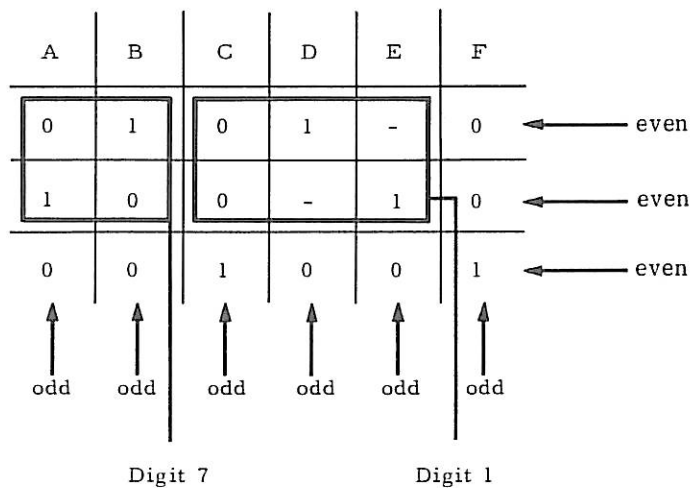
The 1, 2, 4 and 8 bits of digit 1, 2, 3, 4, 5 or 6 in the word are recorded in bit positions 1, 6, 7 and 15 respectively. The 1, 2, 4 and 8 bits of digit 7, 8, 9, 10, 11 or 12 in the word are recorded in bit positions 9, 10, 12 and 13 respectively. Bit positions 2, 3, 4, 5, 8, 11, 14 and 16 are reserved for check bits.

The check bits are generated so as to make unique combinations of odd and even parities for the digits represented. The way in which the parity bits are generated can be illustrated by the following diagram:



The check bits are generated so that the sum of the bits held in the bit positions in each column is odd and that the sum in each row is even.

The data and parity bits for a 9 in digit position 1 and a 6 in digit position 7 of a word are as follows:



This is recorded on a tape frame as:

	1	}	D
	0		
	0	}	F
	0		
	1		
	0	}	C
	0		
	1		
	0	}	A
	1		
	0		
	1	}	B
	0		
	0		
	1	}	E
	0		

Example

Consider a word in I.A.S. with the value 000042972481.

0	0	0	0	4	2
9	7	2	4	8	1

When transferred to tape this is recorded in six frames thus:

Tape movement



The data and check bits are recorded on tape as follows:

Tape movement



0	0	0	0	0	0
1	1	1	1	1	1
1	0	0	1	0	0
1	1	1	0	0	0
1	0	0	0	1	1
0	0	0	0	0	1
0	0	0	0	1	0
1	1	1	1	0	0
1	1	0	0	0	1
0	1	1	0	0	0
0	1	0	1	1	0
0	1	0	1	0	0
1	0	0	0	1	0
0	0	1	0	0	1
0	0	0	0	0	0
1	1	1	1	1	1

The check bits are generated when information is written to tape. When a word is read from tape the odd and even parity configuration is checked for each pair of digits. The system is such that a single-bit gain or loss can be detected precisely.

Time for a queued tape unit to come under computer control.	800 milliseconds after the unload instruction has been accepted
Fixed Delay Period on reading or writing	0.48 ms
Delay before writing first block	2 seconds
Rewinding or unloading speed	Two speeds available, choice depending on engineers adjustment. (a) 150 inches a second. (b) 300 inches a second, subject to last 600 feet (approx.) of tape being rewound at 150 inches a second; previous 2,400 feet (approx.), or remainder of tape if less than 3,000 feet, is rewound at the full speed of 300 inches a second; remainder, up to 600 feet, is rewound at 150 inches a second.
Time to rewind a 3,600 foot reel.	At 150 inches a second, 4 minutes 48 seconds At 300 inches a second, 3 minutes 12 seconds
Length of long gap.	1.35 inches
Length of short gap.	1.12 inches
Time to create long gap on writing	11.2 ms
Time to create short gap on writing	7.5 ms
Time to traverse long gap on reading - with stop/start	11.2 ms
Time to traverse long gap on reading - without stop/start	9.0 ms
Time to traverse short gap on reading - with stop/start	9.7 ms
Time to traverse short gap on reading - without stop/start	7.5 ms
Time for a word to be transferred from Register G to I.A.S. during reading	15 μ s
Time for a word to be transferred from I.A.S. to Register F during writing	15 μ s
Minimum distance between final end of tape marker and actual end of tape	15 inches
Minimum distance between early end of tape marker and final end of tape marker	15 feet

Timing

3.9.3

When the tape is being read or written, one word passes the read/write heads every 133 microseconds. Of these 133 microseconds, only 15 microseconds are used for transferring information between tape and I.A.S. and the remaining 118 microseconds can be used for other processing.

It follows that if tape reading or writing is being time-shared with the main program, then for every 118 microseconds of program there is a delay of 15 microseconds to allow the tape transfer to take place. This corresponds to an increase in program time of approximately 12.7 per cent.

If both reading and writing are taking place then of every 133 microseconds, 30 microseconds are used for transfers between tape and I.A.S., leaving 103 microseconds available for other processing.

It follows that if reading and writing are being simultaneously time-shared with the main program, then for every 103 microseconds of program there is a delay of 30 microseconds to allow the actual transfers to take place. This corresponds to an increase in program time of approximately 29.1 per cent.

The actual time for a block to be read or written on tape can be calculated by allowing 133 microseconds for each word in the block.

In calculating the total time for reading or writing tape, allowance should be made for the time to traverse the inter-block gaps. It is advisable to allow for long gaps when estimating times.

THE HALF - INCH ($22\frac{1}{2}$ kc/s) MAGNETIC - TAPE SYSTEM

3.10

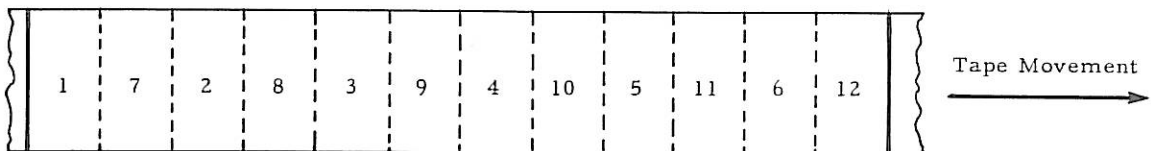
The tape is half an inch wide and is divided into ten longitudinal tracks. Data is recorded on the tape in the form of bits. Thus a cross-section of the tape consists of ten bit positions and is called a frame. The tape moves at a speed of 75 inches a second. The packing density is 300 digits (25 words) to the inch and the tape digit transfer speed is therefore 22,500 digits a second ($22\frac{1}{2}$ kc/s). The time for one word to pass the read/write heads is 533 microseconds.

The maximum length of tape on a spool is 3,600 feet. The length of tape between the end of tape marker and the end of tape is a minimum of 15 inches. The length of tape between the early end of tape marker and the end of tape marker is a minimum of 20 feet.

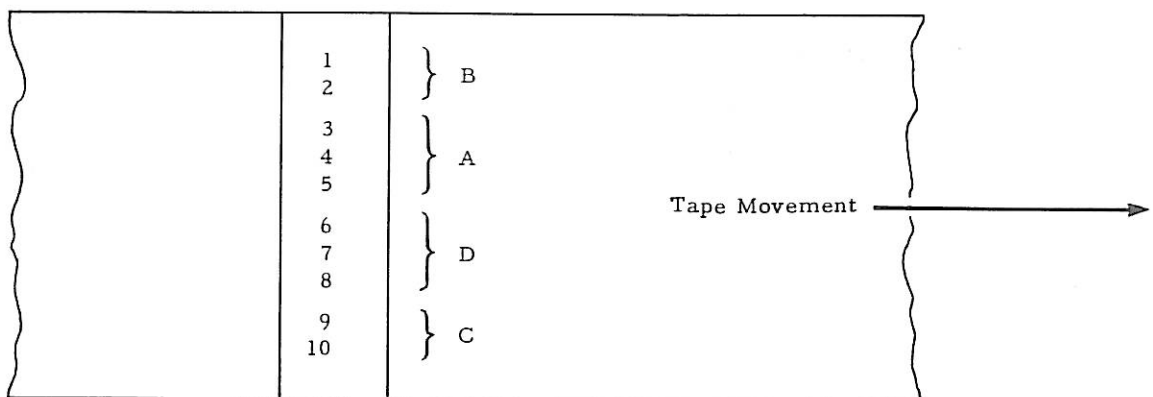
Representation of Data on Tape

3.10.1

One tape frame contains four data bits and six check bits. The four data bits represent one digit, and it follows that one word of data occupies twelve tape frames. The digits of a word are written on tape as follows:



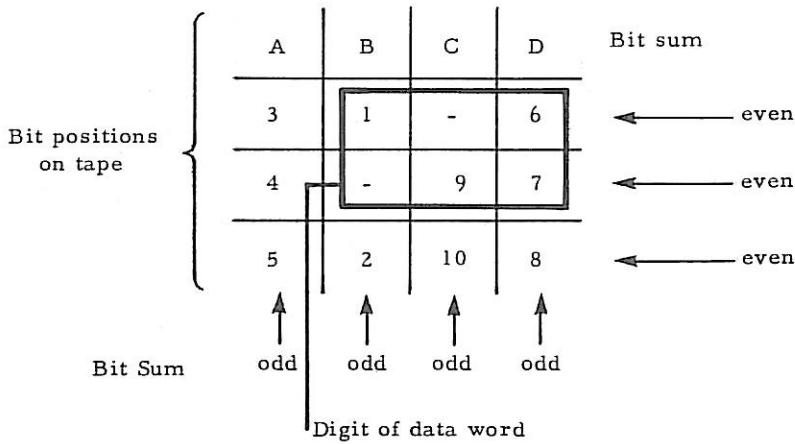
The bit positions in one tape frame are grouped as follows:



The groups are lettered for explanation purposes only.

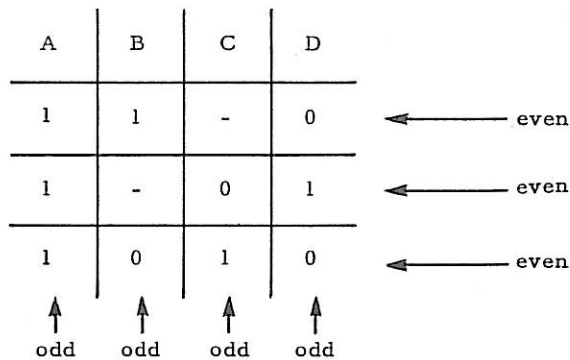
The 1, 2, 4 and 8 bits of a digit are recorded in bit positions 1, 6, 7 and 9 respectively. Bit positions 2, 3, 4, 5, 8 and 10 are reserved for check bits.

The check bits are generated so as to make unique combinations of odd and even parities for the digit represented. The way in which the parity bits are generated can be illustrated by the following diagram:

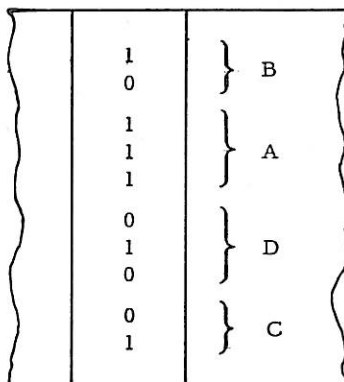


The check bits are generated so that the sum of the bits held in the bit positions in each column is odd, and that the sum in each row is even.

The data and parity bits for a digit with value 5 are as follows:



This is recorded in a tape frame as:



The check bits are generated when information is written to tape. When a word is read from tape the odd and even parity configuration is checked. The system is such that a single bit which has been gained or lost can be detected precisely.

Timings and Statistics**3.10.2**

Time for a queued tape unit to come under computer control	800 ms after the unload instruction has been accepted.
Fixed delay period on reading or writing	2.04 ms
Delay before writing first block	2 seconds
Rewinding or unloading speed	The rewinding speed is not constant but decreases as the length of tape on the loading spool increases. The time to rewind half a reel is therefore more than half the complete rewind time.
Time to rewind a 3,600 foot reel	Under 4 minutes
Length of long gap	1.24 inches
Length of short gap	1 inch
Time to create long gap on writing	18.8 ms
Time to create a short gap on writing	13.4 ms
Time to traverse long gap on reading - with stop/start	18.8 ms
Time to traverse long gap on reading - without stop/start	16.6 ms
Time to traverse short gap on reading - with stop/start	15.6 ms
Time to traverse short gap on reading - without stop/start	13.4 ms
Time for a word to be transferred from Register G to I.A.S. during reading	15 μ s
Time for a word to be transferred from I.A.S. to Register F during writing	15 μ s
Minimum distance between final end of tape marker and actual end of tape	15 inches
Minimum distance between early end of tape marker and final end of tape marker	15 feet

Timings**3.10.3**

When the tape is being read or written, one word passes the read/write heads every 533 microseconds; only 15 microseconds are used for transferring information between tape and I.A.S., and the remaining 518 microseconds can be used for other processing.

It follows that if tape reading or writing is being time-shared with the main program, then for every 518 microseconds of program there is a delay of 15 microseconds to allow the tape transfer to take place. This corresponds to an increase in program time of approximately 2.9 per cent.

If both reading and writing are taking place then of every 533 microseconds, 30 microseconds are used for transfers between tape and I.A.S. leaving 503 microseconds available for other processing.

It follows that if reading and writing are being simultaneously time-shared with the main program, then for every 503 microseconds of program there is a delay of 30 microseconds to allow the actual transfers to take place. This corresponds to an increase in program time of about 6 per cent.

The actual time for a block to be read or written to tape can be calculated by allowing 533 microseconds for each word in the block. In calculating the total times for reading and writing to tape, allowance should be made for the time to traverse the inter-block gaps. It is advisable to allow for long gaps when estimating times.

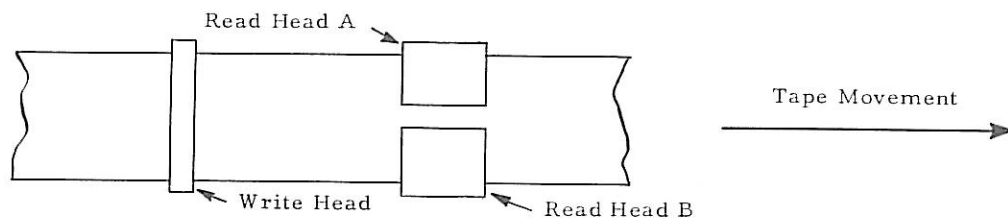
Four, six or eight tape decks can be linked to the computer and used as extra storage for program and/or data words. Each tape deck is capable of handling one reel of tape at a time. When the tape is in motion it passes from one tape spool, past the read/write heads, where information is read or recorded on the tape, and onto another spool. Information cannot be written to tape unless the spool has a writing ring fitted to it by the operator. This prevents the accidental overwriting of master information. It is possible to read from one deck and write to another deck simultaneously.

There is a tape control unit which acts as a link between the central processor and the individual tape decks. The tape control unit consists of a read unit and a write unit each containing a buffer store for use when reading or writing tape. Program instructions are available for controlling the tapes. Words may be transferred from magnetic tape to I.A.S. and vice versa, the transfer taking place via the appropriate buffer and Register A.

Single instructions initiate the transfer of consecutive words between tape and I.A.S., the unit of transfer being called a block. There is no restriction on the number of words in a block other than the size of I.A.S. and the remaining length of the tape.

Information is recorded on tape by varying the length of the signals on the tape to represent different digital values. This means that information is recorded across the full width of the tape.

The tape is divided into two tracks which run longitudinally along the tape. There is one write head which writes information across the full width of the tape, and two read heads (designated A and B), one for each track. Program instructions are available for reading from either track.



Tape Layout

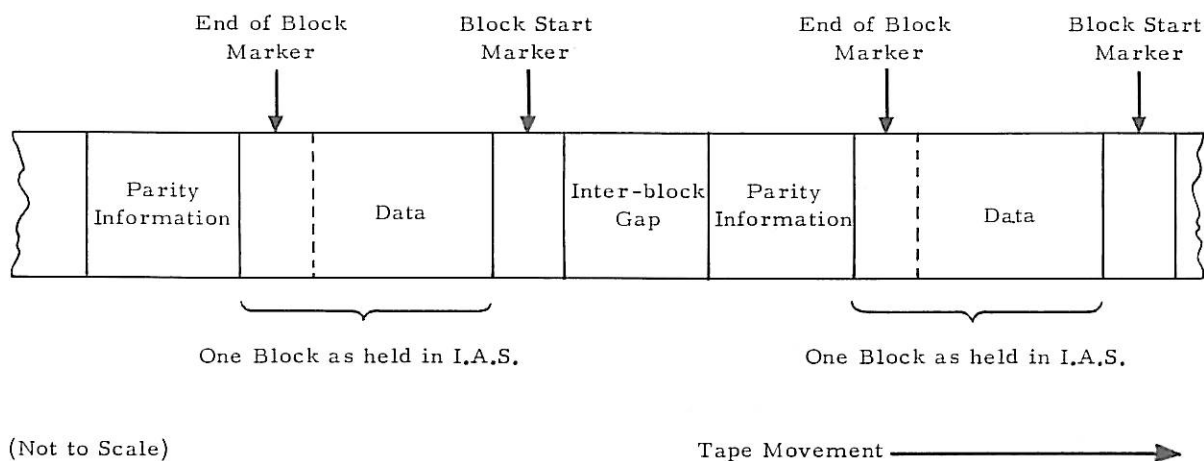
3.11.1

The last word of each block is the end of block marker which consists of a word with 15 in every digit position. When writing to tape the programmer must position the end of block marker in the I.A.S. word immediately following the last data word of the block. A write instruction causes writing to commence at a specified word of I.A.S. Consecutive words are written to tape up to and including the end of block marker. A read instruction causes a block to be read into I.A.S. starting at a word specified in the instruction. The end of block marker is read and stored in I.A.S. as the last word of the block.

While a block is being written to tape, parity information is accumulated and stored on the tape at the end of the block, immediately following the end of block marker.

After the parity information has been recorded on tape there is a delay while it is check-read after which the tape is brought to rest. A subsequent write instruction causes the tape to start moving again, and writing to take place when the tape has reached the necessary speed. These delays during writing give rise to gaps between consecutive blocks of data. They are called inter-block gaps. They are erased by the write head and therefore contain no information.

Immediately before writing commences a block start marker is written to tape consisting of six words of zeros. On subsequent reading this indicates that the data block is about to reach the reading heads. The tape layout is summarized below.



When reading, the tape is brought to rest between consecutive read instructions. The stopping and starting takes place in the inter-block gap.

Soon after the beginning of the tape there is a beginning of tape marker. This is read photo-electrically and prevents the first block being written on the leading end of the tape which may easily be damaged.

Just before the end of tape there is an end of tape marker. An indicator is set if this is detected on writing to tape.

Tape Organization

3.11.2

It is recommended that certain blocks should be written on the tape for organizational purposes only. It is important that the correct tape reels should be loaded for the job to be run and for this reason the first block on tape is made a label block; and should be recorded on the tape when the tape is written, and checked whenever it is read. There is a recommended layout for this block and also for the last block on a particular tape and for the final block of a tape file. General routines are available for reading and writing tape and for creating and checking label blocks. These 'housekeeping' routines are described in Part 5 and full details are given in the tape manual (Magnetic Tape Housekeeping Routines and Conventions).

Data values are recorded on tape by recording in one direction for a given time interval. When a new value is to be recorded, the direction of recording is changed. A change in recording direction is known as a flux reversal. On reading back, the value of data is assessed by time intervals elapsing between successive flux reversals. There are four basic time intervals and each data digit is recorded uniquely on the tape as a combination of two of these intervals. With this method of recording the following transfer errors may occur:

- (a) A reversal may be missed or an extra reversal generated.
- (b) The time interval may have deviated from its correct value sufficiently for it to be interpreted as an adjacent time interval.

Checks are made to detect both these types of error. Error indicators are set when errors are detected.

Writing Tape

As a block is being written to tape, parity bits are accumulated for the data contained in the block. These bits are recorded on the tape at the end of the block as a series of time intervals and are used for checking purposes. When information has been written to tape it is passed to the read heads where it is automatically checked on read head A. During check-reading a check is made that the time interval between any two consecutive flux reversals does not exceed a certain fixed limit. A check is also made that the total reversals read in a block represent an integral number of words. These two checks detect errors due to reversals being missed or extra reversals being generated. During check-reading the parity bits are re-accumulated as the data is read. When the complete block has been check-read these bits are compared with the parity bits recorded on the tape. Any discrepancy indicates that a write transfer error has occurred and an appropriate error indicator is set. The parity checking system detects errors which occur due to the time interval extending into an adjacent range. When an error occurs during writing tape, the tape should be backspaced, a section of tape should be cancelled and the block re-written on the next section. If necessary this procedure should be repeated several times before stopping the computer and indicating that there is a persistent error. Cancelled tape is ignored when subsequently read.

Data are transferred from I.A.S. to the tape control unit from whence they are written to tape. When the end of the block marker is detected in the control unit, the transfer from I.A.S. ceases and the end of block marker is written to tape. As each complete word is check-read, it is tested to see if it is the end of block marker. If reversals are gained or lost during writing the check-reading becomes out of phase and parts of two separate words are interpreted as one word. The data may be such that two part words form a word of fifteens and are interpreted as the end of block marker. This condition is detected and an error indicator set, when the following characters are read. This is because the first part of the parity information, which immediately follows the

end of block marker, should contain zeros. If reversals are gained or missed, or if the reversals extend to an adjacent range, the end of block marker may not be detected at all. In this case check -reading continues until the inter-block gap is detected when the tape is automatically stopped.

Reading Tape

When tape is read, a check is made that the time interval between consecutive reversals does not exceed a fixed limit, and also that the reversals in a block represent an integral number of words. The parity bits are regenerated and compared with those recorded on the tape. Indicators are set if any transfer errors are detected by these means. If errors occur during reading tape, the tape should be backspaced and the block re-read.

Data are read from tape to tape control unit from whence they are transferred to I.A.S. When the end of block marker is detected in the tape control unit, reading ceases and the end of block marker is stored in I.A.S. If reversals are gained or missed, the reading becomes out of phase, and parts of two separate words are interpreted as one word. The data may be such that two part words form a word of fifteens and are interpreted as the end of block marker. This condition is detected, and an error indicator set, when the following characters are read. This is because the first part of the parity information, which immediately follows the end of block marker, should contain zeros. If reversals are gained or missed, or if the data is misread due to the time interval extending to an adjacent range, the end of block marker may not be detected at all. In this case reading continues until the inter-block gap is detected when the tape is automatically stopped.

Tape Deck Addresses

3.11.4

Each tape deck has a fixed address in the range 1 - 8, which is specified in magnetic-tape instructions.

The pressing of the Allocate button on the tape deck causes the deck to come under computer control. It is then operated by any instructions referring to its address.

The housekeeping routines provide facilities for overcoming the difficulties of fixed addresses. This is done by allowing the programmer to specify a programmed address which can be made, by operator action, to correspond to an actual deck address. This means that the operator can allocate the decks as he wishes and prevents program running being held up because a particular deck is not available.

The housekeeping routines also provide facilities for queueing. When an unload instruction has been given to a tape deck, it is possible for its programmed address to be allocated to another deck. Details of these facilities are given in the Tape Housekeeping Manual.

Tape Write Instruction

Effect This instruction causes one block to be written from I.A.S. to magnetic tape.

Operation Starting at a specified word of I.A.S., successive words are written on both tracks of the tape, up to and including the end of block marker.

Notes The instruction is double-length and is made up as follows:

First half	Function digits	- 39
	First two address digits	- 00
	Third address digit	- 1
	Fourth address digit	- Tape deck address i.e., a number in the range 1 to 8
Second half	Function digits	- 00
	Address digits	- Address of first I.A.S. word to be written to tape

Thus the instruction

D	F	A	R
	39	0012	
	00	0035	10

causes a block to be written to deck address 2 starting at word 35 block 10.

During the execution of this instruction the write unit is occupied. Further instructions to the specified tape deck or write or cancel instructions to any tape deck cause the Tape Order Error indicator to be set.

Tape Read Instructions

Effect These instructions cause a block to be read into I.A.S. from magnetic tape. Reading may take place either on track A or on track B.

Operation Information from tape is stored in successive words of I.A.S. up to and including the end of block marker. Two instructions are available, one using the read head on track A and the other using that on track B.

Notes The instructions are double-length and are made up as follows:

- First Half Function digits - 39
- First two address digits - 00
- Third address digit - 2 if reading on track A
 7 if reading on track B
- Fourth address digit - Tape deck address

- Second half Function digits - 00
- Address digits - Address of first I.A.S. word in which information
 is to be stored

For example the instruction

D	F	A	R
	39	0073	
	00	0009	16

causes a block to be read, using track B, from deck address 3 and to be stored in I.A.S. starting at word 9 block 16.

During the execution of these instructions the read unit is occupied. Further instructions to the specified tape deck or read or backspace instructions to any other tape deck will cause the Tape Order Error indicator to be set.

Backspace Instruction

Effect This instruction causes the tape to be backspaced one block.

Operation The tape on a specified unit is rewound one block. When it has stopped the tape is positioned so that the block can be read, written or cancelled. The previous block is unaltered.

Notes The instruction is a single-length instruction as follows:

- Function digits - 39
- First two address digits - 00
- Third address digit - 3
- Fourth address digit - Tape deck address

Thus the instruction

D	F	A	R
	39	0034	

causes the tape on tape deck address 4 to be rewound one block.

During the execution of this instruction the read unit is occupied. Further instructions to the specified tape deck or read or backspace instructions to any tape deck cause the Tape Order Error indicators to be set.

Cancel Instruction

Effect This instruction causes a section of tape to be cancelled so that it is ignored when subsequently read.

Operation The tape is erased while it moves forward for a fixed length of time. A cancelled section of tape contains no information and is therefore ignored by subsequent read or backspace instructions and no error indicators are set.

Notes The instruction is a single-length instruction as follows:

Function digits - 39
First two address digits - 00
Third address digit - 4
Fourth address digit - Tape deck address

Thus the instruction:

D	F	A	R
	39	0041	

causes a section of tape to be cancelled on tape deck address 1.

During the execution of the cancel instruction the write unit is occupied. Further instructions to the specified tape deck or write or cancel instructions to any tape deck cause the Tape Order Error indicator to be set.

Rewind Instruction

Effect This instruction causes the tape on a specified deck to be rewound to the beginning of the tape.

Operation The tape on the specified deck is rewound to the beginning of the tape. A subsequent read or write instruction causes the first block on tape to be read or a new block to be written.

Notes The instruction is a single-length instruction as follows:

Function digits - 39
First two address digits - 00
Third address digit - 5
Fourth address digit - Tape deck address

Thus the instruction

D	F	A	R
	39	0056	

causes the tape on tape deck address 6 to be rewound to the start of the tape. While the tape is being rewound, any further instructions to the same tape deck cause the Tape Order Error indicator to be set. Any tape instructions can be carried out on other decks. When the tape has been rewound there is a delay before the first block is read or written.

Unload Instruction

Effect This instruction causes the tape on a specified tape deck to be rewound ready for removal from the deck.

Operation The tape on the specified deck is rewound on the loading spool and the spool made ready for removal. When an unload instruction has been accepted the tape deck is no longer under computer control.

Notes The instruction is a single-length instruction made up as follows:

- Function digits - 39
- First two address digits - 00
- Third address digit - 6
- Fourth address digit - Tape deck address

Thus the instruction

D	F	A	R
	39	0062	

causes the tape on deck address 2 to be rewound on the loading spool ready for removal.

Any instruction to a deck for which the unload instruction has been accepted causes the Tape Order Error indicator to be set. Any tape instructions can be carried out on other decks.

Digit Position	1	2	3	4	5	6	7	8	9	10	11	12
	Function	0	0	Command	Deck Address	0	0	I.A.S.	Effect			
Double-length Instructions	39	0	0	1	1-8	0	0	I.A.S. Address	Write			
	39	0	0	2	1-8	0	0	I.A.S. Address	Read Track A			
	39	0	0	7	1-8	0	0	I.A.S. Address	Read Track B			
Single-length Instructions (may begin either half of word)	39	0	0	3	1-8				Backspace			
	39	0	0	4	1-8				Cancel			
	39	0	0	5	1-8				Rewind (Back to Start of Spool)			
	39	0	0	6	1-8				Unload (Tape Rewound on Spool)			

SUMMARY OF MAGNETIC-TAPE INSTRUCTIONS, AS HELD IN THE COMPUTER.

Indicators 81 to 88 Deck Address Ready

Purpose There are eight of these indicators, one for each possible deck address. They are used to test whether a tape deck is ready for use before giving instructions to use it.

Operation The indicators are numbered 81 to 88, and are associated with deck addresses 1 to 8 respectively. The Deck Address Ready indicator is automatically set if:

- (a) The appropriate tape deck is not busy, i.e. no tape instruction is currently being executed on that deck, and
- (b) the tape deck is mechanically ready.

The indicator is unset if either of the above conditions are not satisfied, indicating that the deck is not ready to receive an instruction.

Notes For example the instruction

D	F	A	R
4	85	0009	B

tests whether deck address 5 is ready for use, if it is ready a jump is made to word 9 of the current block.

The appropriate Deck Address Ready indicator should be tested before giving a tape instruction. This prevents a tape order error due to an instruction being given to a deck which is not ready.

Indicator 89 Transport Mechanically Ready

Purpose This indicator is associated with the Deck Address Ready indicators. If one of the indicators 81 to 88 is unset, this indicator can be used to discover which condition caused it to be unset.

Operation Indicator 89 is automatically set if:

- (a) The last Deck Address Ready indicator to be tested was found to be set, or
- (b) the last Deck Address Ready indicator was unset because the tape deck was busy.

The indicator is automatically unset if the last Deck Address Ready indicator was unset because the tape deck was mechanically unready. If a tape deck becomes mechanically unready, indicator 89 will become unset after a delay of up to 500 milliseconds.

Notes Indicator 89 should be tested when one of the indicators 81 to 88 is found to be unset.

If indicator 89 is set the tape deck is busy, the program must wait until the deck is not busy before giving it an instruction.

If indicator 89 is unset operator action is required because the deck is mechanically unready.

For example

I	D	F	A	R
5	4	81	0007	B
	4	89	0005	B
6		11	2002	

will cause the following:

If deck address 1 is ready for use a jump is made to word 7 of the block which continues with the tape program.

If deck address 1 is not ready for use, indicator 89 is tested. If indicator 89 is set the tape deck is busy and a jump is made to re-test indicator 81 to see if the deck is no longer busy. If indicator 89 is unset the computer is stopped with 2003 displayed in CR3. This indicates to the operator that action must be taken.

Indicator 80 Tape Order Error

Purpose This indicator is set whenever a tape instruction is given which cannot be accepted.

Operation Indicator 80 is automatically set if:

- (a) A tape instruction is given to a tape deck which has not been allocated.
- (b) An instruction is given to a tape deck which is busy.
- (c) A write or cancel instruction is given to any tape deck while either type of instruction is being executed.
- (d) A read or backspace instruction is given to any tape deck while either type of instruction is being executed.
- (e) A write or cancel instruction is given to a deck which has not had a writing ring fitted to below the tape spool.
- (f) A tape instruction is given without specifying a deck address.

Notes When indicator 80 is set the Tape Order Error light on the console is lit. If the Optional Stop switch is on, the computer stops automatically when there is a tape order error, CR3 containing the faulty instruction with 1 added to it. In this case, restarting the computer causes the light to go out, but the indicator remains set until tested by program.

Indicator 70 Write Unit Ready

Purpose This indicator is used to test whether the write unit is busy.

Operation Indicator 70 is set automatically when a write or cancel instruction has been completed, indicating that the write unit is ready to accept another instruction.

It is unset automatically when a write or cancel instruction is accepted.

Notes Indicator 70 should be tested before a write or cancel instruction to ensure that the write unit is ready and to prevent a possible tape order error. For example instruction:

I	D	F	A	R
10	4	70	11	B
	4	00	10	B
11		39	0013	
		00	0005	15

will cause the following:

Indicator 70 is repeatedly tested until the write unit is ready. When the write unit is ready the instruction to write to deck address 3 is obeyed.

Indicator 72 Read Unit Ready

Purpose This indicator is used to test whether the read unit is busy.

Operation Indicator 72 is set automatically when a read or backspace instruction has been completed, indicating that the read unit is ready to accept another instruction.

It is unset automatically when a read or backspace instruction is accepted.

Notes Indicator 72 should be tested before a read or backspace instruction to ensure that the read unit is ready and to prevent a possible tape order error.

Indicator 74 Write Errors

Purpose This indicator is used to detect any transfer errors which occur during writing tape.

Operation Indicator 74 is automatically set if any errors are detected during the check-reading which follows writing.

Indicator 74 is unset when a write or cancel instruction is accepted.

Indicator 75 Read Errors

Purpose This indicator is used to detect any transfer errors which occur during reading tape.

Operation Indicator 75 is automatically set if any errors are detected during reading.

Indicator 75 is unset when a read instruction is accepted.

Notes Indicator 75 is unaffected when a section of tape is read which has been previously cancelled.

Indicator 76 End of Tape, Writing

Purpose This indicator is used to detect the end of tape marker when writing tape.

Operation Indicator 76 is automatically set if the end of tape marker is detected when writing or cancelling tape.

Indicator 76 is unset by program test.

Notes The end of tape marker is read photo-electrically at a station situated before the write head. It is not possible to write over the end of tape marker and therefore writing is automatically inhibited when the end of tape marker is detected. When the next instruction is given, writing is automatically suspended until the end of tape marker has passed the write head.

When writing is inhibited, a block may have been only partly written. This condition is detected at the check-reading station and the Write Errors indicator is set. The normal error procedure of backspacing, cancelling and re-writing the block ensures correct recording.

Indicator 77 Short Block, Reading

Purpose This indicator is used to detect a short block of four words.

Operation Indicator 77 is set automatically when a short block, consisting of exactly four words (including the end of block marker) is detected on reading. Short blocks may be used for identification purposes and are used as label blocks by the housekeeping routines.

Indicator 77 is unset by program test.

Indicator 71 Write Master

Purpose This indicator is used to test whether writing has been successfully completed.

Operation Indicator 71 is set if indicator 74 or indicator 76 is set, i.e. if:

- (a) Any errors have been detected on writing, or
- (b) if the end of tape marker has been detected on writing.

Indicator 71 is unset if both indicator 74 and indicator 76 are unset.

Notes If indicator 71 is unset it follows that writing has been successful. Only if indicator 71 is set is it necessary to test indicators 74 and 76 to discover which exceptions condition has arisen.

Indicator 73 Read Master

Purpose This indicator is used to test whether reading has been successfully completed.

Operation Indicator 73 is set if either indicator 75 or indicator 77 is set, i.e. if:

- (a) Any errors have been detected while reading, or
- (b) if a short block has been read.

Indicator 73 is unset when indicators 75 and 77 are both unset.

Notes If indicator 73 is unset it follows that reading has been successful. Only if indicator 73 is set is it necessary to test indicators 75 and 77 to discover which exceptions condition has arisen.

Indicator 79 Writing Ring Present

Purpose This indicator is used to ensure that a writing ring has been fitted before a write instruction is given.

Operation This indicator is associated with the last Deck Address Ready indicator to be tested.

Indicator 79 is set if a writing ring is present on the tape deck for which the last Deck Address Ready indicator was tested and if indicator 89 (Transport Mechanically Ready) is set.

It is unset if the writing ring is absent, or if indicator 89 (Transport Mechanically Ready) is unset. If a tape deck becomes mechanically unready, indicator 79 will become unset after a delay of up to 500 milliseconds.

Notes Indicator 79 should be tested before writing to tape. This ensures that a writing ring has been fitted and prevents a possible tape order error.

The Secure lamp on the tape deck is lit when no writing ring is present on the load reel.

INDICATOR	TITLE	SET BY	UNSET BY
70	Write Unit Ready	Write Unit not busy	Write or Cancel Instruction
71	Write Master	Indicator 74 or 76 becoming set	Unsetting of both Indicators 74 and 76
72	Read Unit Ready	Read Unit not busy	Read or Backspace Instruction
73	Read Master	Indicators 75 or 77 becoming set	Unsetting of both Indicators 75 and 77
74	Write Errors	Any Errors during Writing	Write or Cancel Instruction
75	Read Errors	Any Errors during Reading	Read or Backspace Instruction
76	End of Tape	End of Tape Marker during Writing	Program Test
77	Short Block	Short Block Read	Program Test
79	Writing Ring Present	Writing Ring Present on Spool and transport mechanically ready and address seized on deck last tested	Writing Ring not Present on Spool or Transport Mechanically unready or address not seized on deck last tested.
80	Tape Order Error	Unacceptable Instruction	Program Test
81 to 88	Deck Address (1 to 8) Ready	Tape Deck not busy and Mechanically Ready	Tape Deck busy or not Mechanically Ready
89	Transport Mechanically Ready	Transport Mechanically Ready and address seized on deck last tested	Transport not Mechanically ready or address not seized on deck last tested

Figure 39: SUMMARY OF MAGNETIC-TAPE INDICATORS

Once a tape instruction has been initiated it is carried out automatically and does not require constant program control. This means that other programs can be obeyed while the tape is in motion. An automatic interrupt facility is incorporated to cater for the transfer of words between I.A.S. and the tape control unit. Thus the need for any program control during reading or writing tape is eliminated. This enables the programmer to make maximum use of the time available while the tape is in motion.

When a rewind or unload instruction is being obeyed, there is no transfer between I.A.S. and magnetic tape and thus any instruction may be obeyed other than a tape instruction to the same deck.

During a cancel or backspace instruction there is no transfer between I.A.S. and magnetic tape and any instructions can be obeyed other than tape instructions to the same deck.

Reading Tape

Information from tape is read into a special register, Register G. Register G is a register used as a buffer store when tape reading is taking place. Register G can be considered as being in two halves, each consisting of a 6-digit register and a small buffer. The layout of the registers is shown diagrammatically in Figure 40.

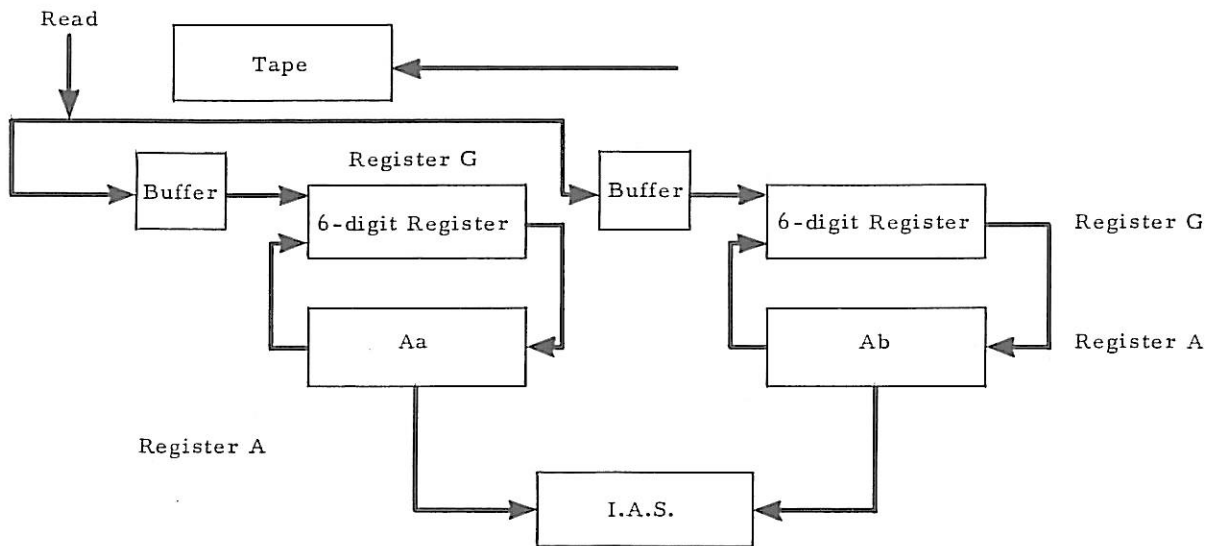


Figure 40: TAPE READ

Information is read from tape into the small buffers and thence transferred into the two 6-digit registers.

When a complete word is contained in the 6-digit registers, the tape control unit indicates to the central processor that it is ready to interrupt the main program in order to transfer the word to I.A.S.

When the current program instruction has been completed the tape control unit breaks in. The contents of the registers are circulated so that they are interchanged with the contents of Register A. The data read from tape is now contained in Register A and is transferred to I.A.S. When the transfer has been completed, the contents of the registers are again interchanged with those of Register A. The original contents of Register A have thus been restored and the main program continues until the next word has been read into the registers, when it is again interrupted.

When the tape control unit is ready to break in, the program is interrupted at the end of the instruction it is obeying. If it is obeying a multibeat instruction, e.g. multiplication, the break-in takes place at the end of the micro instruction currently taking place. Drum transfers cannot be obeyed while reading is taking place since they take longer than the frequency at which the break-in occurs, and cannot be interrupted in the middle of execution. Great care should be taken if other peripheral units are used while the tape is moving since the interruptions from the tape unit upset the timings. In particular the full P.P.F. cannot be used during tape reading. Processing should not take place on the current block while it is still being read since, if any transfer error occurs, the whole block is re-read.

Writing Tape

Information to be written to tape passes through a special register, Register F, which is used as a buffer store. Register F can be considered as being in two halves, each consisting of a 6-digit register and a small buffer. The layout of the registers is shown diagrammatically in Figure 41.

When a word is to be written to tape the contents of the 6-digit registers are interchanged with those of Register A. A word is then written from I.A.S. to Register A and the contents of the registers again interchanged. The original contents of Register A have thus been restored and the word to be written to tape is held in the two 6-digit registers. The word is written to tape via

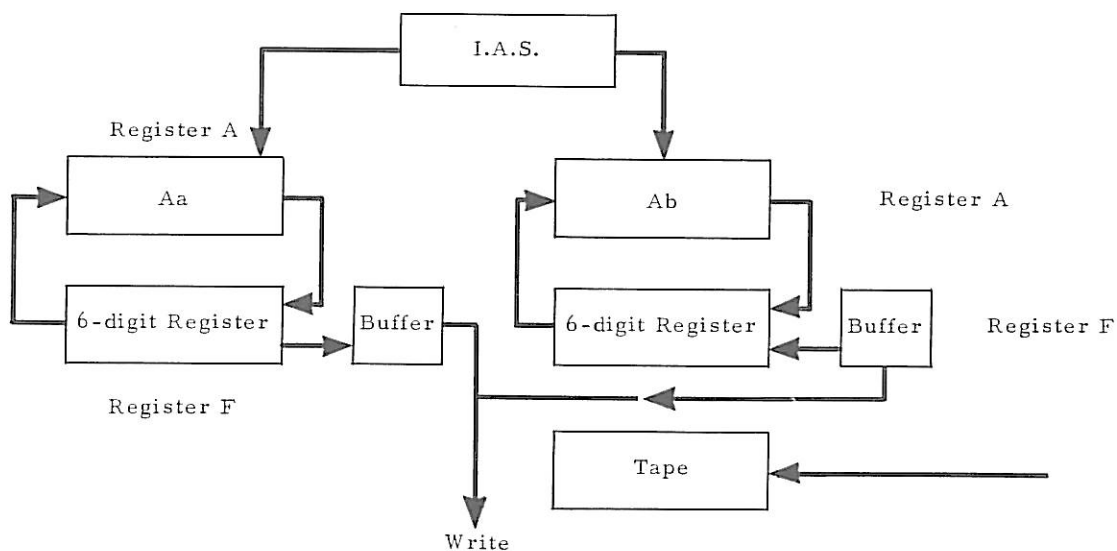


Figure 41: TAPE WRITE

the small buffers. When the 6-digit registers have become empty the tape control unit is ready to receive another word from I.A.S. and a request to break in is made to the central processor.

The program is interrupted at the end of the instruction it is obeying, or at the end of the micro instruction if it is obeying a multi-beat instruction. Drum transfers cannot be obeyed while writing is taking place as they take longer than the frequency at which the break-in occurs, and cannot be interrupted in the middle of execution. Great care should be taken if other peripheral units are used while the tape is moving since the interruptions from the tape control unit upset the timings. In particular the full P.P.F. cannot be used during tape writing. Processing should not take place on the block which is being written to tape since, if any transfer error occurs, the whole block is re-written.

It is possible to read from one tape deck and write to another tape deck simultaneously. In this case the main program is interrupted by both the read unit and the write unit, the read unit taking preference if they should both require to break in at the same time.

The time-sharing with a tape read or write program is effected as follows:

When the read or write instruction has been given, a jump is made from the tape program to the section of program which is to be time-shared with the reading or writing. This program is then executed with occasional automatic interruptions from the tape control unit when it requires to communicate with I.A.S. When the program has been completed a return is made to the tape program which when the transfer has been completed, tests the necessary indicators to ensure that the block has been correctly read or written. If the section of main program takes longer than the reading or writing of the block, causing the tape to have stopped before control is returned to the tape program, this does not matter. If drum transfers occur within the time-sharing program they should be preceded by tests of Read and Write Unit Ready indicators (72 and 70) to ensure that the tape transfer has been completed. Facilities for time-sharing when reading or writing tape are provided with the tape housekeeping routines.

Method of Recording on Tape

3.11.8

Information is recorded by lengths of tape which correspond to time intervals between successive flux reversals. There are four basic time intervals referred to as intervals t_0 , t_1 , t_2 , and t_3 respectively. The table below shows the actual value given to each interval when writing tape, together with the range of timings associated with each interval when reading.

Time Interval	Duration of Recording during Writing (microseconds)	Permitted Range on Reading (microseconds)
t_0	16	10 to 21
t_1	28	22 to 34
t_2	43	35 to 51
t_3	62	52 to 72

If time intervals are read which are less than 10 microseconds or greater than 72 microseconds, the error indicators are set.

When a block of words is written to tape, each digit is recorded uniquely as a combination of two of the basic time intervals. In order to achieve this, the 4-bit digit is first converted into two digits. These digits are known as quaternary digits since they may assume the values 0, 1, 2 or 3, each one corresponding to a basic time interval. The conversion is achieved as follows:

- (a) The 4-bit and the 1-bit of the data digit are combined to give the first quaternary digit. A quaternary digit of value 0, 1, 2 or 3 is achieved by giving a value of 2 to the original 4-bit and a value of 1 to the original 1-bit.
- (b) The 8-bit and the 2-bit of the data digit are combined to give the second quaternary digit. A quaternary digit of value 0, 1, 2 or 3 is achieved by giving a value of 2 to the original 8-bit and a value of 1 to the original 2-bit.

Example

The digit 14 is recorded in the computer as an 8-bit, a 4-bit and a 2-bit.

The first quaternary digit is formed by assigning a value 2 to the original 4-bit. This gives a quaternary digit of 2, which is recorded on tape by time interval t_2 elapsing between successive flux reversals.

Digit Value	Binary Coding in Computer				Quaternary Equivalent		Time Intervals between Flux Reversals		
	Bit Value				First Digit	Second Digit	First Digit (microseconds)	Second Digit (microseconds)	Digit Total (microseconds)
	8	4	2	1	4 and 1 Bits	8 and 2 Bits			
0	0	0	0	0	00 or 0	00 or 0	16	16	32
1	0	0	0	1	01 or 1	00 or 0	28	16	44
2	0	0	1	0	00 or 0	01 or 1	16	28	44
3	0	0	1	1	01 or 1	01 or 1	28	28	56
4	0	1	0	0	10 or 2	00 or 0	43	16	59
5	0	1	0	1	11 or 3	00 or 0	62	16	78
6	0	1	1	0	10 or 2	01 or 1	43	28	71
7	0	1	1	1	11 or 3	01 or 1	62	28	90
8	1	0	0	0	00 or 0	10 or 2	16	43	59
9	1	0	0	1	01 or 1	10 or 2	28	43	71
10	1	0	1	0	00 or 0	11 or 3	16	62	78
11	1	0	1	1	01 or 1	11 or 3	28	62	90
12	1	1	0	0	10 or 2	10 or 2	43	43	86
13	1	1	0	1	11 or 3	10 or 2	62	43	105
14	1	1	1	0	10 or 2	11 or 3	43	62	105
15	1	1	1	1	11 or 3	11 or 3	62	62	124

Figure 42: QUATERNARY EQUIVALENTS OF DIGITS 0 TO 15 SHOWING TIME INTERVALS BETWEEN FLUX REVERSALS

The second quaternary digit is formed by assigning a value 2 to the original 8-bit and a value 1 to the original 2-bit. This gives a quaternary digit of 3, which is recorded on tape by time interval t_3 elapsing between successive flux reversals.

The table in Figure 42 shows the quaternary digits for each digit value 0 to 15 together with the time intervals associated with them.

Parity Checking System

3.11.9

While information is being written to tape, parity bits (dependent on the data value) are accumulated in two special parity registers. After the end of block marker has been written, these parity bits are converted into eight quaternary digits. Eight reversals, with time interval t_0 , are written immediately after the end of block marker and these are followed by eight reversals representing the parity information. When reading or check-reading takes place, the parity bits are regenerated in the parity registers in the reading or check-reading unit. At the end of the block the parity bits are converted to quaternary digits and are automatically compared with the parity digits recorded on tape.

The generation of the parity bits in the parity registers is achieved as follows:

For each quaternary digit written to tape there is a corresponding parity digit. The parity digit has the value 0, 1, 2 or 3 and is expressed as 2 bits, a 2-bit and a 1-bit. The quaternary digits and the corresponding parity digits are shown below:

Quaternary Digit	0	1	2	3
Parity Digit	0	3	2	1
Parity Digit Bit 2	0	1	1	0
Parity Digit Bit 1	0	1	0	1

There are six possible transitions for which a time interval may be interpreted as an adjacent interval, i.e. the transitions t_0 to t_1 , t_1 to t_0 , t_1 to t_2 , t_2 to t_1 , t_2 to t_3 and t_3 to t_2 . It can be seen that four of these transitions change both the 2-bit and the 1-bit of the parity digit, and the other two change the 1-bit of the parity digit only.

Two groups of parity bits are accumulated as follows:

One group is formed by accumulating the 2-bits of the parity digits in a 5-bit shifting register.

A second group is formed by accumulating the 1-bits of the parity digits in a 7-bit shifting register.

The arrangement is shown diagrammatically in Figure 43.

The accumulation consists of summing the bit which has been shifted out of the register with the appropriate bit of the parity digit in a modulo 2 adder. The result is then shifted into the

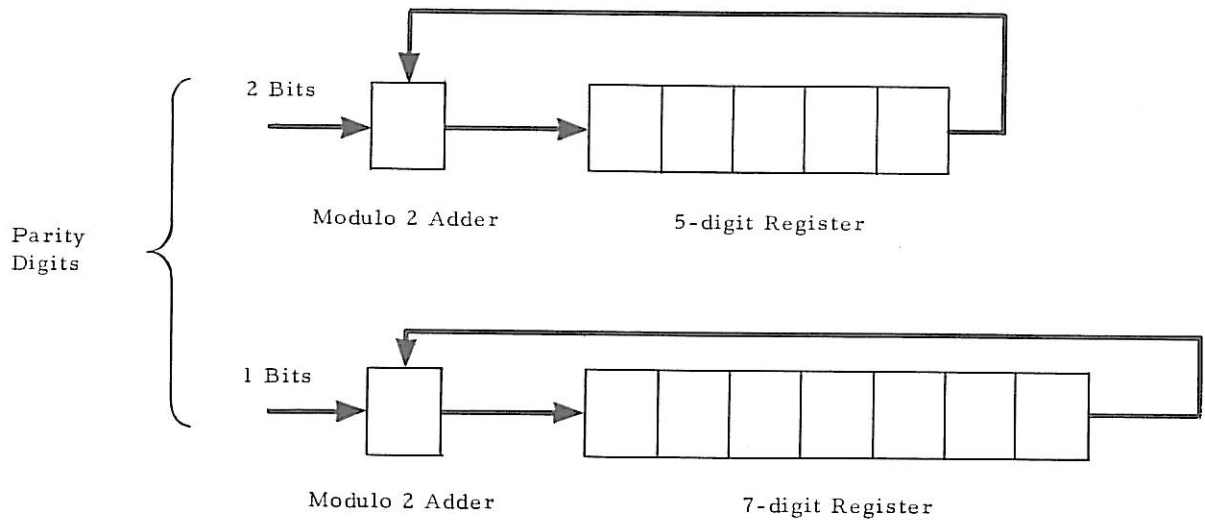
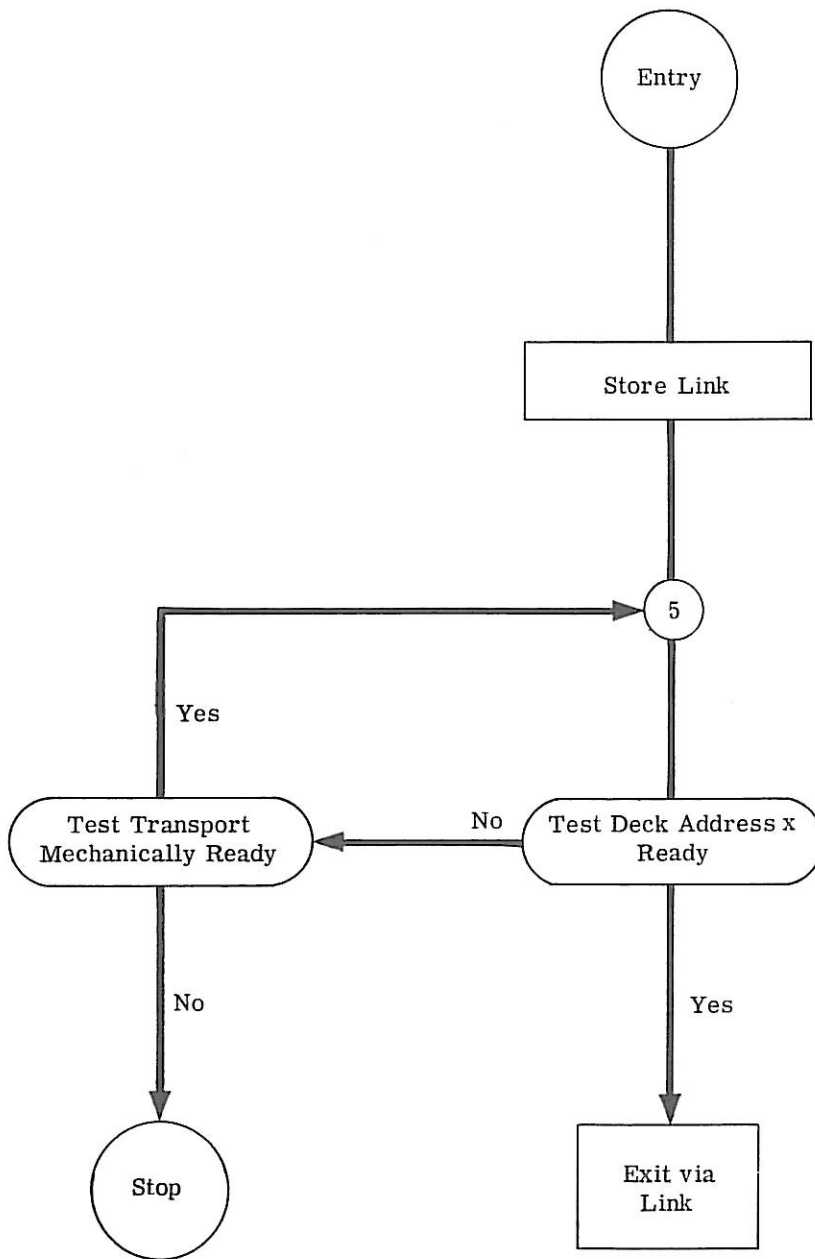


Figure 43: PARITY CHECKING

register. The summation modulo 2 in effect means that if the output from the register and the parity bit are the same, then a zero is entered into the register; if they are different, then a one is entered into the register.

DECK
TESTING
SUBROUTINE



Deck not correctly allocated
or there is a Mechanical
Failure. Correct Fault and
restart from beginning.

Flowcharts for reading and writing magnetic tape are included as an illustration of the use of the various indicators and instructions. These flowcharts do not cover all the housekeeping procedures in the standard routines and are intended as an example only. It is assumed that beginning of tape labels have already been written or checked. Reading and writing takes place on tape deck address x. The value of x would be entered as a parameter at the start of the routine.

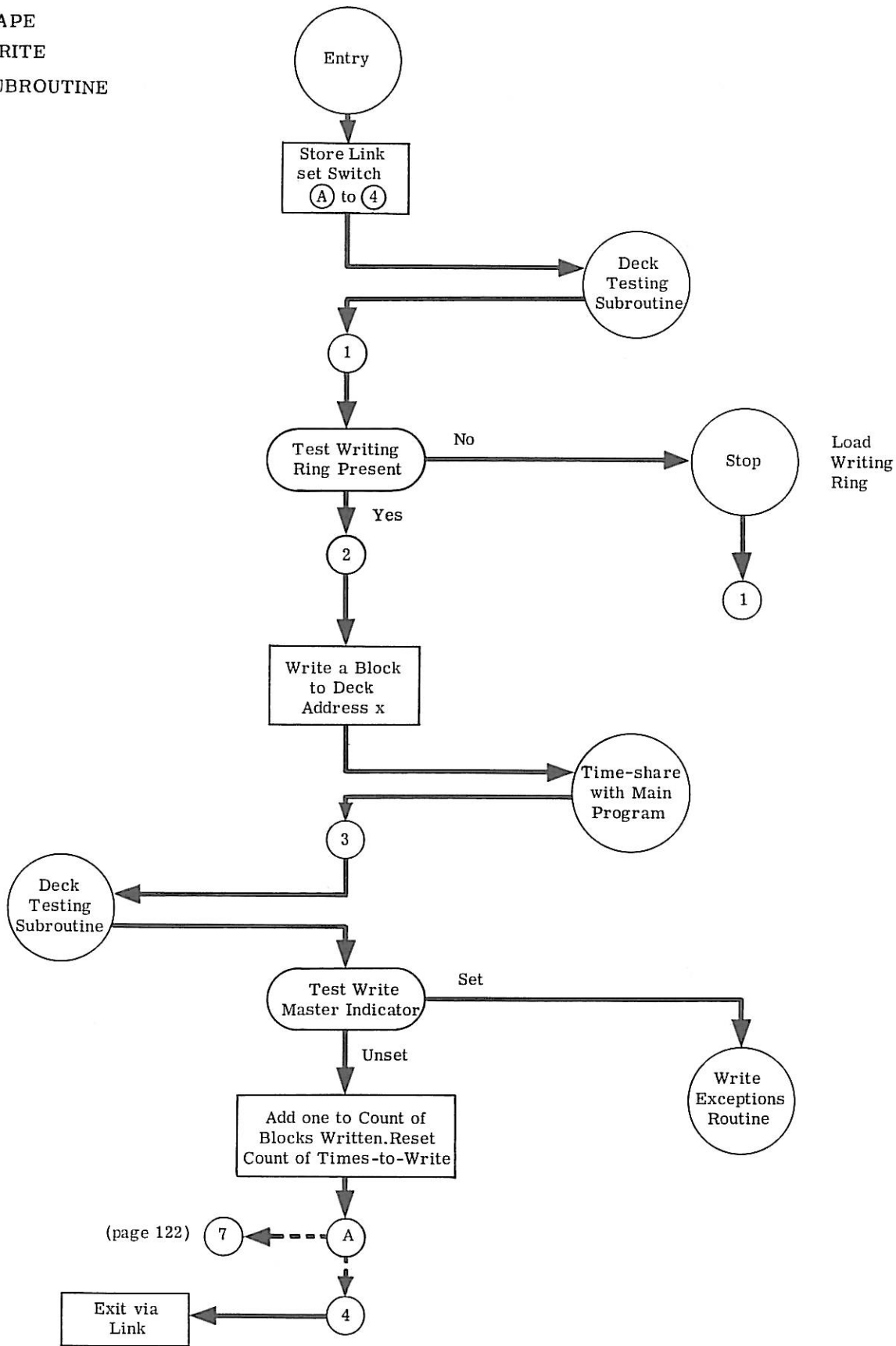
Deck Testing Subroutine

This routine determines if deck address x is busy. It may be used to ensure that a tape deck has finished obeying the current instruction and is ready to receive a further instruction. The use of this routine prevents any tape order errors arising due to instructions being given to decks which are busy.

To ensure that tape deck address x is mechanically ready the appropriate Deck Address Ready indicator and indicator 89 Transport Mechanically Ready are repeatedly tested until the deck ceases to be busy and is ready to receive another instruction. If the tape deck is not mechanically ready, then the deck has not been correctly allocated or there is a mechanical failure. The fault should be corrected and the job restarted from the beginning.

The Deck Testing routine has been made into a subroutine since it is used in several places by the other routines.

TAPE
WRITE
SUBROUTINE



Tape Write Subroutine

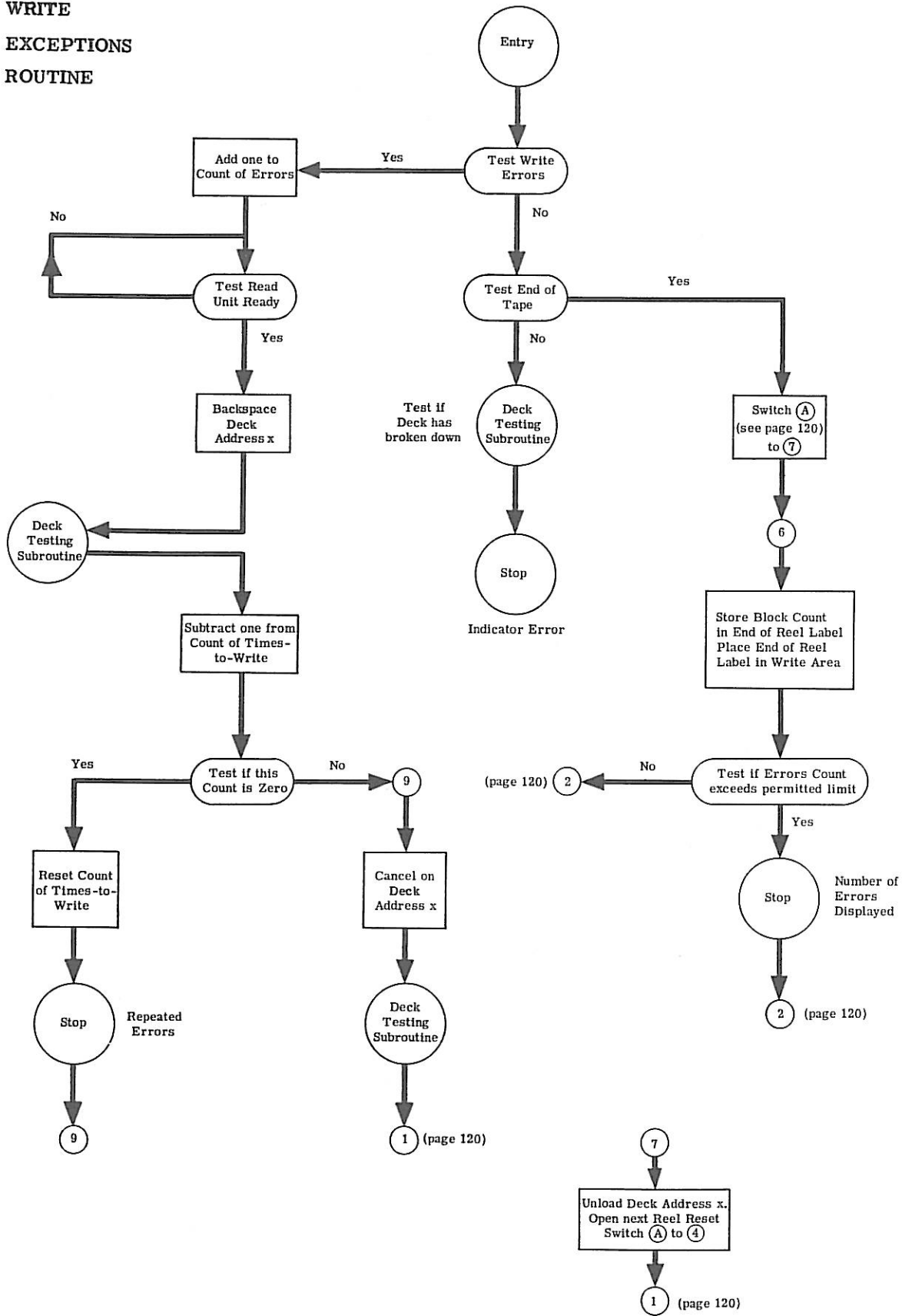
This routine writes a block from I.A.S. to magnetic tape on deck address x. A test is made to ensure that writing has been successfully completed and a count (initially zero) of the number of blocks which have been written is updated.

A test is made to ensure that a writing ring has been fitted before an attempt is made to write. The Deck Testing routine is entered before this test since it tests the Deck Address Ready indicator and thus associates the Writing Ring Present indicator with the correct deck address.

When the write instruction has been given, an exit is made so that the writing routine can be time-shared with the main program. On return to the write routine the Deck Testing routine is entered to test whether writing has been completed. When the block has been written indicator 71 (Write Master) is tested. If it is unset then there have been no writing errors and no special conditions have arisen, and the block count is updated. The times-to-write count is used when transfer errors occur.

The times-to-write count corresponds to the number of attempts to be made to write a block on successive sections of tape before giving an indication that there are persistent errors. If indicator 71 (Write Master) is set, then an error (or exceptional case) has arisen and the Write Exceptions routine is entered.

**WRITE
EXCEPTIONS
ROUTINE**



Write Exceptions Routine

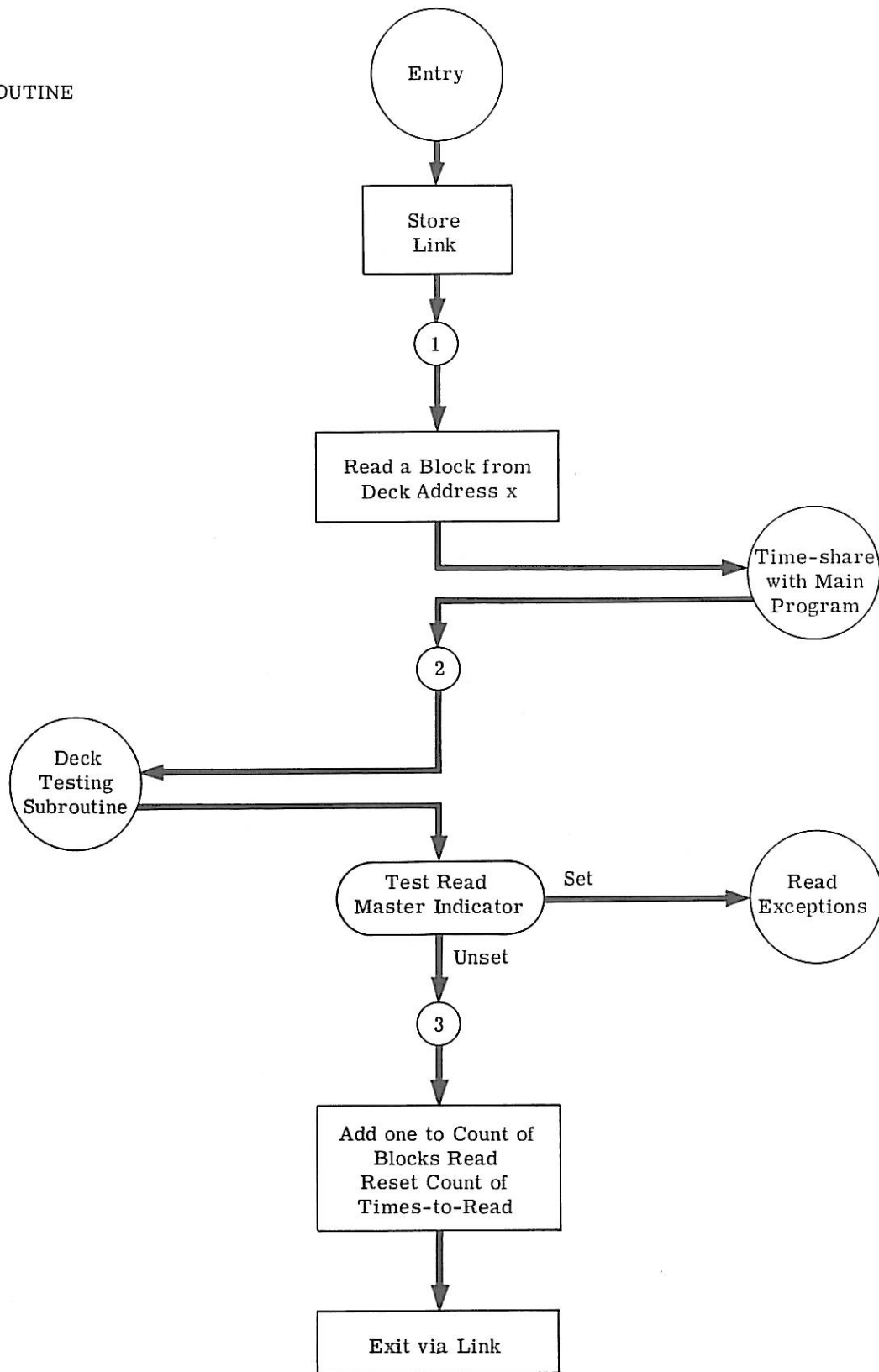
The Write Exceptions routine discovers which exceptions condition has arisen and deals with it appropriately.

A count is kept (initially zero) of any transfer errors which occur.

If an error occurs, the tape is backspaced, a section of tape is cancelled and an attempt is made to write the block on the next section. If necessary this is repeated several times until the times-to-write count is reduced to zero. If this happens the computer stops with an indication that there are repeated errors. Restart causes further attempts to be made on following sections of tape.

If the end of tape marker is detected then a short block is written to tape as the last block on the reel. This is called the end of reel label and has a special format to distinguish it from other short blocks. The block count is stored in the end of reel label. A test is made to discover whether the transfer errors have exceeded the permitted number. If this happens the computer stops with the number of errors displayed. When the end of reel label has been correctly written, deck address *x* is unloaded and a new reel is opened.

TAPE
READ
SUBROUTINE

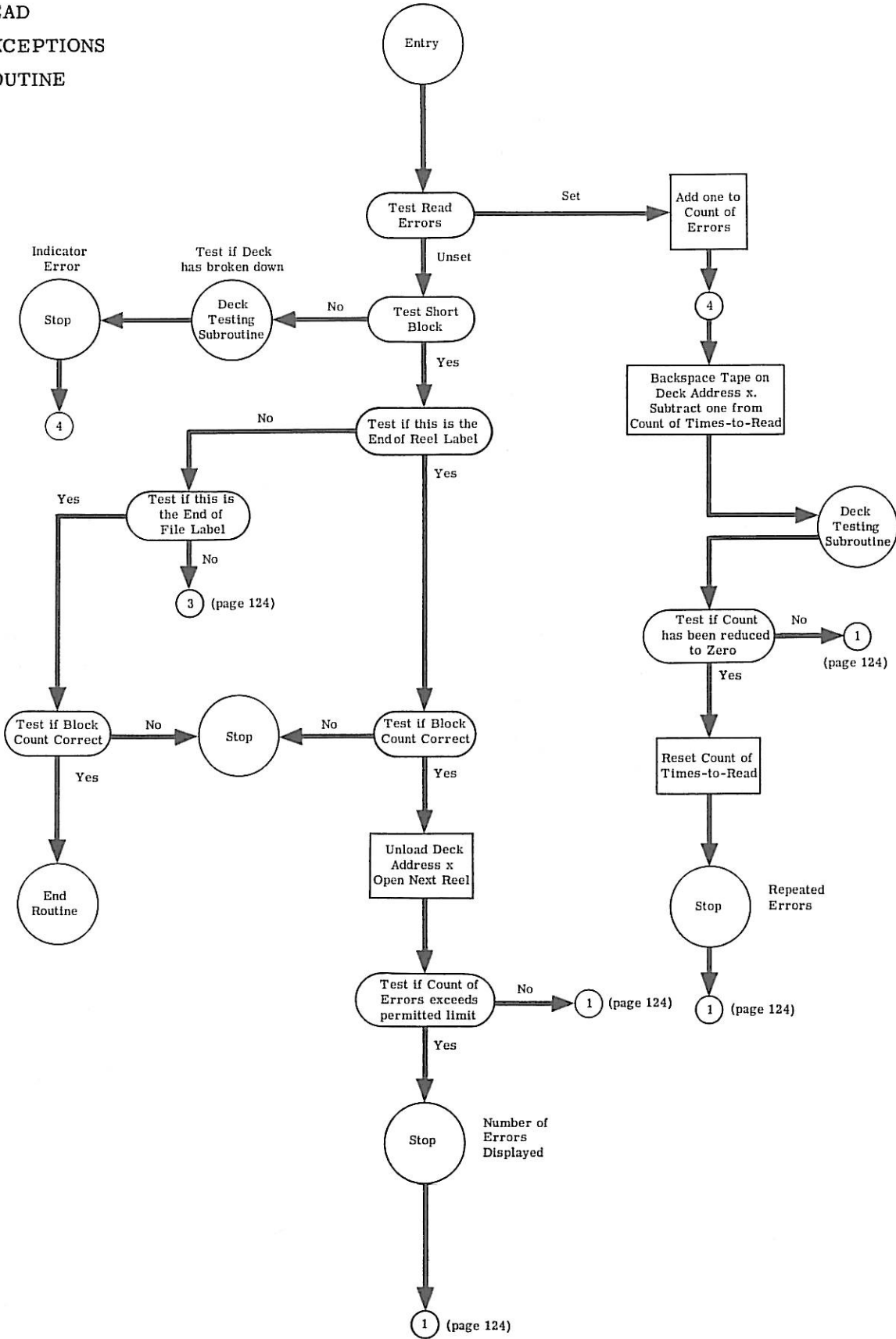


Tape Read Subroutine

This routine reads a block from deck address x and stores it in I.A.S. It tests that reading has been successfully completed and updates a count (initially zero) of the number of blocks which have been read.

When the read instruction has been given, an exit is made so that the read routine can be time-shared with the main program. When the block has been completely read indicator 73 (Read Master) is tested. If it is unset no special conditions have arisen and the block count is updated. The times-to-read count is used when errors occur and corresponds to the number of attempts to be made to read a block before stopping the computer. If the Read Master indicator is set, then a transfer error or exceptional condition has arisen and the Read Exceptions routine is entered.

READ
EXCEPTIONS
ROUTINE



Read Exceptions Routine

The Read Exceptions routine discovers which exceptions condition has arisen and deals with it appropriately. A count is kept of any errors which occur.

If an error occurs, the tape is backspaced and another attempt made to read the block. If necessary several attempts are made until the times-to-read count is reduced to zero. The computer is then stopped as an indication that the tape deck may be faulty. Restart causes further attempts to be made.

If a short block is detected, it is examined to see if it is the end of reel or end of file label. The end of file label is written after the last data block. It acts as an end of reel label and in addition indicates that there is no further reel to follow. When the end of reel or end of file label is detected a check is made to ensure that the number of blocks read corresponds to the block count stored in the label. If a short block is not an end of reel or end of file label, then it is treated as an ordinary data block.

Maximum length of tape on spool	1,800 feet
Minimum distance between end of tape marker and actual end of tape	25 feet
Distance between beginning of tape and beginning of tape marker	7 to 10 feet
Tape Speed	37½ inches a second
Delay before writing first block	5 seconds
Rewind time for a complete reel	Under 3 minutes
Packing density - assuming random distribution of digits	440 digits an inch
Packing density - assuming 25% of digits are zero - otherwise random distribution	480 digits an inch
Digit rate - assuming random distribution	16,500 digits a second
Digit rate - assuming 25% of digits zero otherwise random distribution	18,000 digits a second
Digit rate for all zeros	32,000 digits a second
Digit rate for all fifteens	8,000 digits a second
Average time for one word to pass the read/write heads - assuming random distribution	727 μs
Length of inter-block gap	Approx. 0.8 inches
Time to traverse inter-block gap on writing and write block start marker - excluding time when tape is awaiting an instruction	20 to 25 ms
Time to traverse inter-block gap and block start marker on reading - excluding time when tape is awaiting an instruction	20 to 25 ms
Break-in time when transferring a word to or from I.A.S.	16 μs
Time tape is erased during Cancel Instruction	50 ms (equivalent to a block of about 60 words)

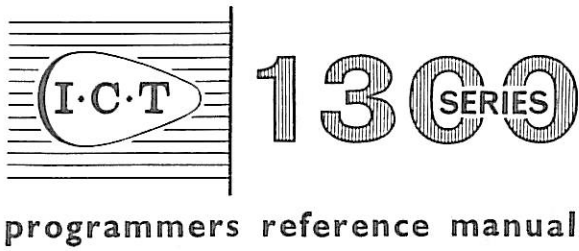
Timing

3.11.12

Assuming random distribution, one word takes 727 microseconds to pass the read/write heads. When reading or writing is taking place, the break-in time for transferring a word to or from I.A.S. takes 16 microseconds, thus leaving 711 microseconds available for other programs. It follows that when a program is being time-shared with tape reading or writing, the program time is effectively increased by 16 microseconds for every 711 microseconds. This corresponds to an effective increase of 2.2 per cent. When reading and writing are simultaneously being time-shared with the main program, the break-in time amounts to 32 microseconds leaving 695 microseconds available for program. This corresponds to an effective increase in program time of 4.6 per cent.

The actual time to write or read a block can be calculated by allowing 727 microseconds for each word in the block. In calculating the total time for reading or writing tape, allowance should be made for the time to traverse the inter-block gaps.





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


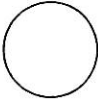
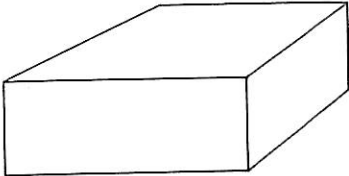
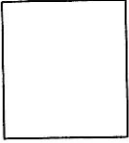
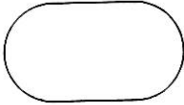
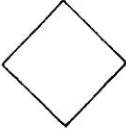
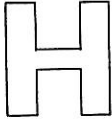
Description	I.C.T. Symbol	Description	I.C.T. Symbol
Punch and Verify		Sort	
File of Punched Cards		Tabulator	
Interpret, Match, Collate, Interpolate or Reproduce		Calculator	
		Manual Operations	

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.

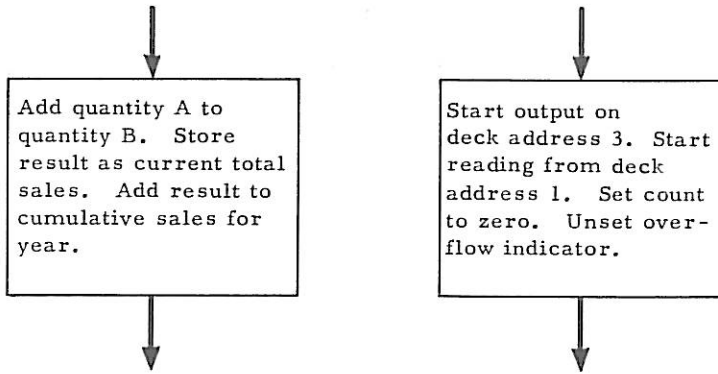
The flow of charting should be vertical, running from top to bottom of the sheet, the direction of flow being indicated by arrows, particularly when branches occur for alternative conditions. Recommended symbols should be used wherever possible. Templates are available which contain the standard symbols, 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

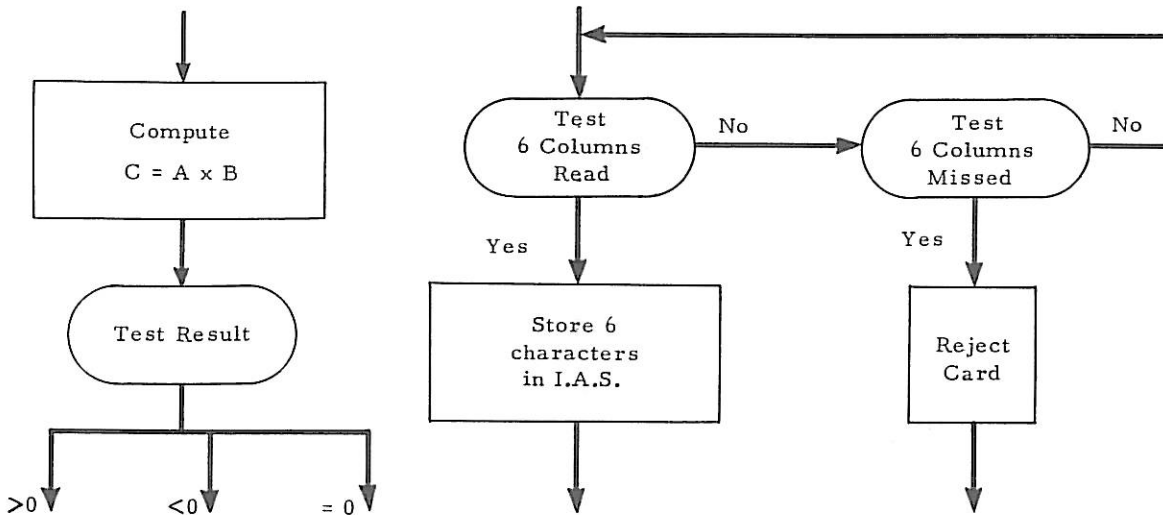
4.1.2

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.

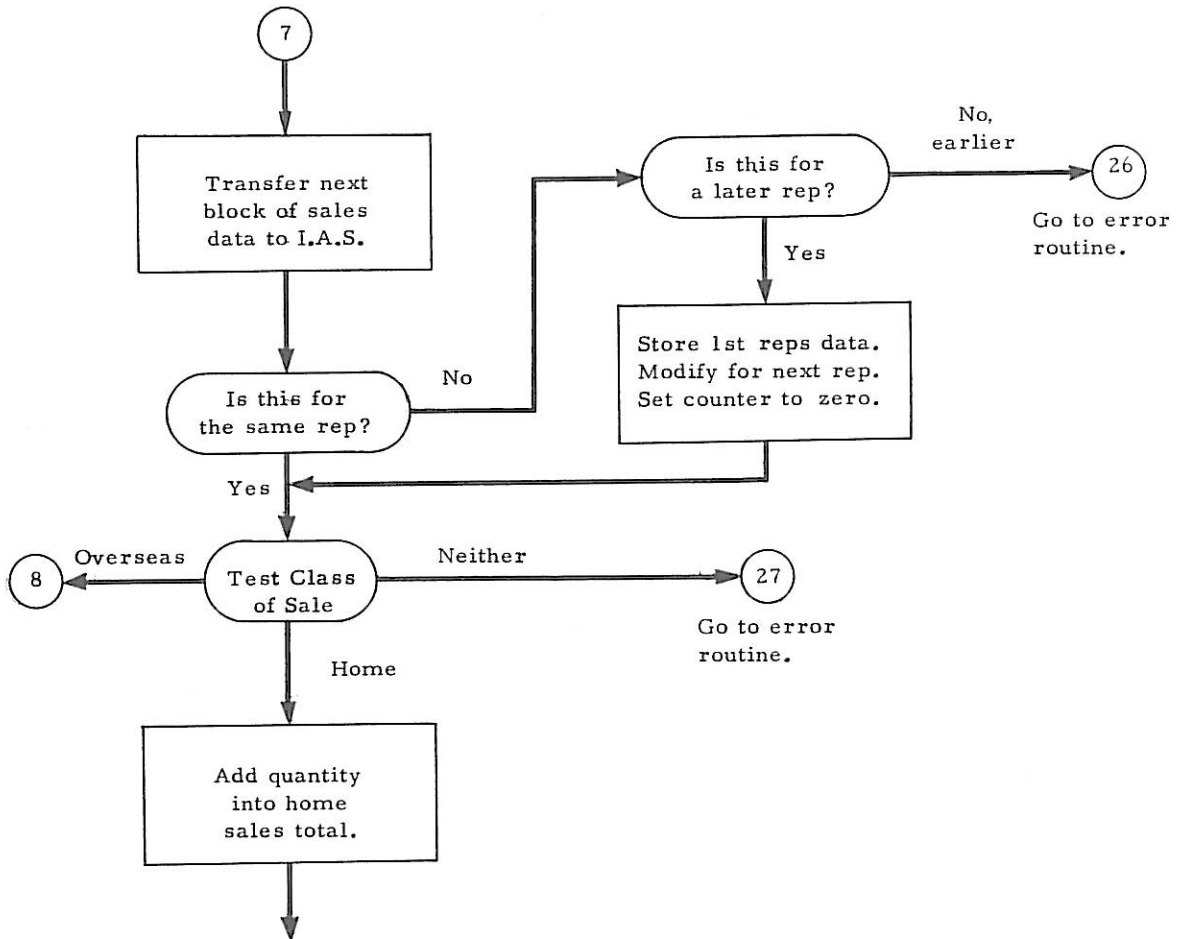
Thus,



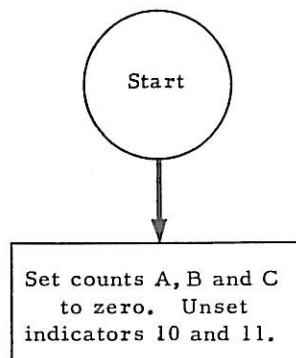
indicates that the procedure continues at connector 16 with an arrow leading from it, i.e.



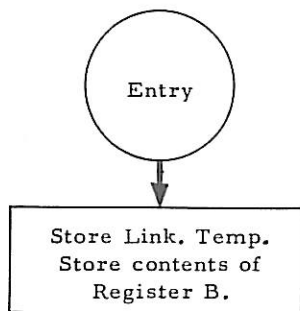
When subsequently converting a flowchart to program instructions, the presence of a connector with an arrow leading from it indicates a junction of several pieces of program. This means that a programmed jump is going to occur to the next instruction and it should therefore be in the first half of a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



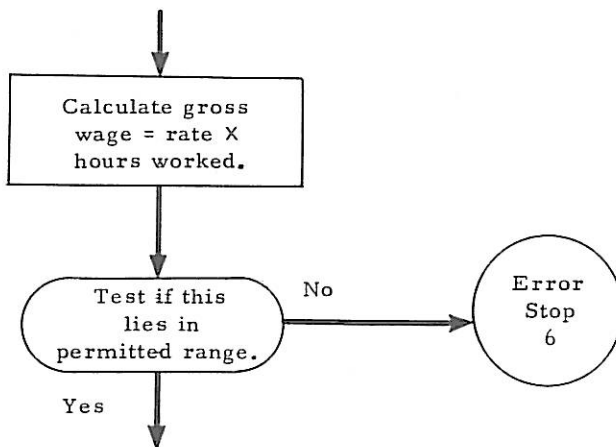
Starts, Stops and Subroutines It is usual to indicate the start of a routine by enclosing the word Start in a circle rather larger than the connector symbol.



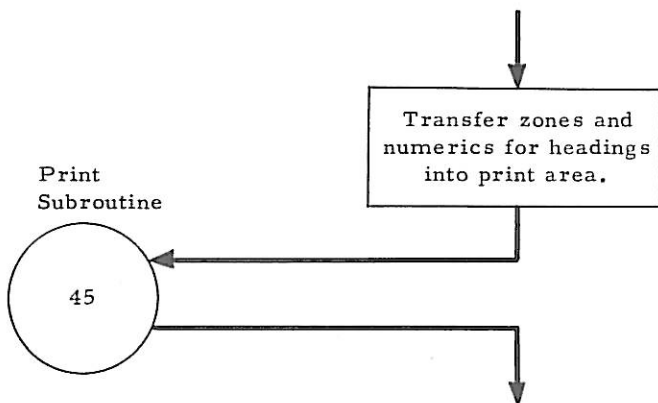
When flowcharting a subroutine the word Entry is used instead of Start.



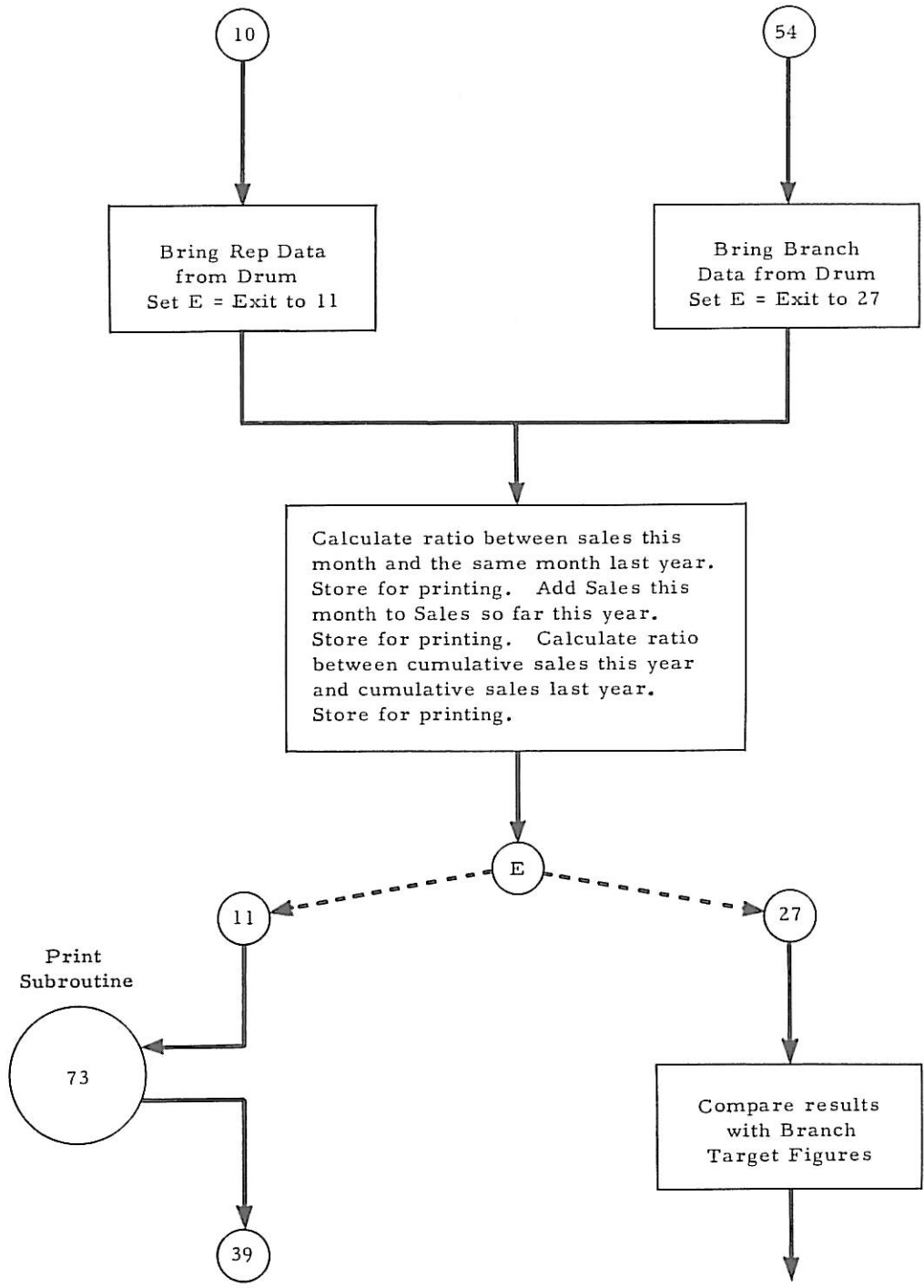
When the procedure involves stopping the computer, this is indicated by the word Stop enclosed in a circle. A number may also be included to identify the stop.



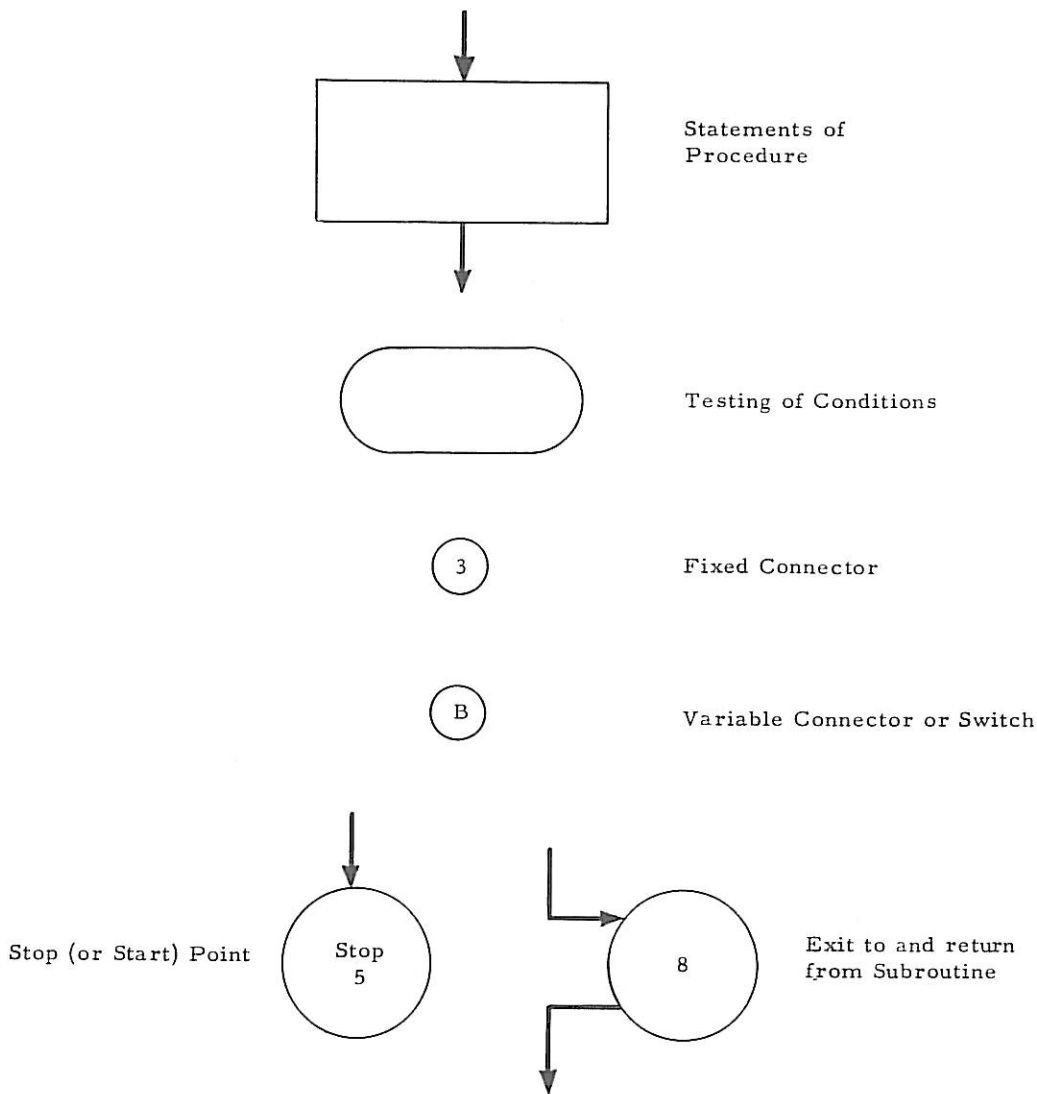
When a subroutine is used as part of the procedure, the subroutine logic is not flowcharted as part of the main routine. The entry to and exit from the subroutine are indicated on the main flowchart by lines leading to and from a circle. The circle contains either the name of the subroutine or an identifying 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	
10		57	0001		Calculation
		62	0003	4	
11		64	0000	6	Accumulate result
		67	0000	5	
12	4	01	0015	B	Test if calculation has been performed 10 times
		37	0001	5	
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	<i>Transfer set of data</i>
		5	0000	10	
13		37	0000	10	
		62	0001	10	
14		54	0001		<i>Perform calculation</i>
		62	0001	10	
15		64	0000	6	<i>result</i>
		37	0002	10	
16		69	0003	10	
		69	0004	10	
17		64	0000	6	
		67	0000	5	
18	4	01	0020	B	<i>Test if calculation has been</i>
		37	0001	5	<i>performed 10 times</i>
19		64	0012	B	<i>Modify transfer</i>
	4	00	0012	B	
					<i>Repeat calculation for</i>
					<i>next set of data</i>

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	
		62	0000	5	Accumulate result in Register B
11		67	0000	1	
	4	02	0010	B	Return to add in next quantity
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
		62	0000	5
7		42	0009	B
		62	0017	B
8		42	0010	B
		8	10	
9		37		
		54	0001	
10		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

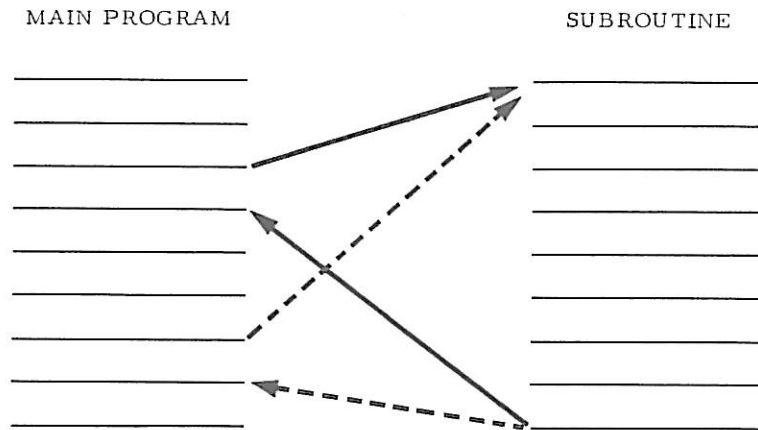
D	F	A	R
	37	0012	16

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

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

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

I	D	F	A	R
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

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

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

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

I	D	F	A	R
11	P		[LINK]	

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.

I	D	F	A	R
12	4	00	0000	29
	4	00	0014	B
13		81	0000	23
		10	0000	23
14		37	0019	23
		62	0016	23

I	D	F	A	R
	B			
0		41	0006	B
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
R		I.A.S. Starting Address	
		Drum Starting Address	R.R.N.

Hence the control word:

D	F	A	R
R		0134	
	01	1315	23

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
~~~~~			
<i>E</i>		0000	15
	20	0000	10

- (a) Set relativizer 10 to an I.A.S. address of 0 and a drum word address of 6000.
- (b) Set relativizer 15 to an I.A.S. address of 31 and a drum word address of 6031.
- (c) Transfer 200 words of program from drum words 6000-6199 to I.A.S. words 0-199.
- (d) 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:

- (a) The E-word should be punched on a separate card from the last program words.
- (b) It should have a card number 1 so that it will satisfy the sequence check (see later) when not preceded by other cards.
- (c) 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	0002	3
			2	0018	B

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



**Example** Consider a program which requires the use of the following general purpose routines:

- (a) A card read subroutine which stores the card image (i.e. successive sets of 6 columns in consecutive words of I.A.S.) under R.R.N.3.
- (b) A distribution routine which distributes a card image held under R.R.N.3 into a form suitable for processing, the distributed information being stored under R.R.N.4.
- (c) A print routine which prints information held under R.R.N.3 and takes the current sprag number from word 0 block 4.

Confusion can be avoided if the user allocates his own block numbers for the various sets of information. Suppose that the program is written and relativizer control words set at the head of the program assuming the following:

- (a) The information from cards is stored as a card image in R.R.N.10.
- (b) This information is distributed from R.R.N.10 into R.R.N.11 ready for processing.
- (c) The information is processed in R.R.N.11 and is distributed into R.R.N.12 (commencing at word 1) ready for printing.
- (d) The current sprag number is placed in word 0 block 12. The print routine prints the results from words 1, 2.....block 12 taking the sprag number from word 0 block 12.

In order that the general purpose routines should operate on the correct data it is necessary to include relativizer control words immediately before the cards for these routines.

Thus, the control word

D	F	A	R
R		0000	10
		0000	3

should be included in the program pack immediately before the cards for the card read subroutine, the control words

D	F	A	R
R		0000	11
		0000	3
R		0000	12
		0000	4

should be included in the program pack immediately before the cards for the distribution routine,

and the control words

D	F	A	R
R	---	0001	12
		0001	3
R	---	0000	12
		0000	4
---	---	---	---

should be included in the program pack immediately before the cards for the print subroutine.

Relativizer cards included in this way must, of course, be numbered so that they pass the Initial Orders sequence check. Subroutines available from the I.C.T. Library have relativizer control words included at their head for all relativizers used by the routine. These relativizer cards have card numbers already punched but the relativizer setting columns are left blank for the user to include his own settings. The I.A.S. and drum address columns of the block relativizer word(s) should also be punched by the user.

To avoid possible errors, general purpose routines should be included in the program pack after the other program blocks. Hence, in the example given, if block 4 were being used by the main program, relativizer 4 should not be set to the same addresses as relativizer 10 until all main program references to R.R.N.4 have been read and converted to absolute form.

All library routines use R.R.N.1 as a temporary storage area. When using several such routines, therefore, it is sufficient to allow space under R.R.N.1 for the largest temporary storage requirement of the routines being used. Care should be taken, however, that intermediate results are not stored in R.R.N.1 by the main program in words where they will be destroyed by one of the subroutines.

## RESTART PROCEDURES

4.8

When a program makes a test and discovers that an error condition has arisen, the computer should be stopped with an identifying number in the address part of the stop instruction which is displayed in CR3. By means of this information the operator can diagnose the type of error condition which has arisen, and take the necessary action to restart the program.

It should normally be possible to continue with the program once the necessary error procedure has been carried out. For example, if an error is found in the data it might be possible, when the computer is restarted, to enter an error routine which prints out a statement that the set of data is not correct. When this has been done the program can continue to read and process the next set of data.

There are some errors which may arise after which it is not possible to continue with the program because information recorded in the machine may be incorrect. Before continuing the processing, therefore, it is necessary to return to an earlier stage in the program and re-create this information. For a small program this would mean completely re-running the program. For programs which run for a long time, however, it is necessary to have restart points to which a return can be made if necessary. Restart points should be included at least once every half hour in the program running time. The program must be written so that when a return is made to the previous restart point any accumulations are set to their original value at that point in the program.

#### **Restarts following I.A.S. and Drum Parity Error Stops**

**4.8.1**

If an I.A.S. parity error occurs, a return should always be made to the previous restart point. When a word is transferred from I.A.S. to Register A an automatic parity check is carried out and indicator 06 is set if an error is detected. Parity bits are then regenerated and the word is stored back in its original location in I.A.S. This means that, if a further attempt is made to effect the transfer from I.A.S., the parity will be correct although the word itself is incorrect. To avoid errors, therefore, a return should be made to the previous restart point.

There are standard routines available for testing drum parity and it is recommended that these should be used. By retaining such routines permanently in I.A.S. and using them for all drum transfers certain programming difficulties can be overcome. The drum transfer is supplied to the routine as a parameter and it is obeyed as part of the subroutine. This makes it possible for a drum transfer instruction to cause itself to be overwritten, the instruction being preserved in the subroutine if it becomes necessary to repeat the transfer.

If a drum parity error occurs then the computer is brought to a stop; restart causes another transfer attempt to be made. When a drum parity error is detected this implies that a transfer error has occurred either in the current transfer or when the information was previously recorded on the drum. In the former case, unless there is a serious mechanical fault, the error should be corrected when another attempt is made. If the error is persistent then, either there is a mechanical fault or the transfer error occurred on writing to the drum. In either case a return should be made to the previous restart point.

#### **PROGRAM TESTING**

**4.9**

When a program has been written and thoroughly checked it is necessary to test it by running it on the computer. In order to test a program it is necessary to create (or obtain) sample data. In addition, particularly when testing a subroutine or a section of a large program, it may be necessary to write a special test program to act as a control routine for the program under test. The control routine would, for example, position the input data correctly and ensure that the programmed indicators were in the required states. The test data and programs should be such that every possible sequence of executing the instructions is tested including the error procedures.

For a large program it is advisable to carry out testing in stages. This enables errors to be more quickly located since they are confined to certain sections of the calculation. The testing of a large program can be achieved by one of the following methods:

- (a) Divide the program into sections and test each section individually. When the sections have all been tested combine them and re-test as a complete program.
- (b) Divide the program into sections. Test the first section and then combine it with the second section. Test these sections and then combine with the third section. Continue until the complete program has been assembled and tested.
- (c) Assemble the whole program initially but start by using data which test only part of the program. Gradually include in the tests sufficient data to cover all conditions.

When a program is ready to be tested it should be punched into program cards. The punched cards should be interpreted so that they can be quickly checked for punching errors. The pack should then be run with the Validity Check subroutine to detect invalid instructions. Data cards should also be punched and checked and placed at the end of the program pack. The assembled pack should be accompanied by an operators' instruction sheet and sent to the machine room. If any print-outs are required of the I.A.S., drum or magnetic-tape storage, these should be specified on the operators' instruction sheet. The program pack will be returned with an indication of whether or not the final stop was reached. If the final stop was not reached then a console log is returned with the pack which contains details of the error and the contents of the registers and states of indicators as displayed on the console when the computer stopped. The states of the Mill, overflow and program indicators and the contents of Registers B and C are not included on the console log since they are given at the head of the I.A.S. or drum print-out.

The information provided on the console log should, together with the print-outs of storage, normally be sufficient for the programmer to locate his error. When one error has been found it is a good practice to examine the print-outs to determine whether, discounting that error, the program has operated correctly. Machine time can be saved and the program proved more quickly if several errors can be corrected for one run on the machine. There are several general purpose routines available which can be used as aids to program testing. These are described in Part 5 under Diagnostic Routines.

During program testing it is very often necessary to alter or re-write sections of the program. It should be noted that if a block is lengthened some relativizer settings may need to be altered. Any alterations should be recorded on the program sheets so that the latter provide an up-to-date copy of the program. Care should be taken during re-writing to ensure that altering the positions of instructions does not lead to further errors. Particular care should be taken where:

- (a) There is a jump to a program word from another part of the program.
- (b) Reference is made to a word by an instruction in another block; for example where a constant held in one block is used by another.
- (c) Where instructions are used whose effect depends on their positions in the first or second half of a word. This is particularly true of a 41 instruction.

## TIMING A PROGRAM

4.10

It is often necessary to calculate the exact time which a piece of program is going to take when it is run on the computer. This is particularly necessary for a general purpose routine or a piece of program which is to be time-shared with the card reader or card punch. It is evident that such timings cannot be observed on the computer, since the times involved are much too small to be accurately measured. The times must therefore be estimated by a summation of the times involved in the execution of the instructions which constitute the program, together with the time for the change of control between program words.

The execution time for a section of program consisting of a straightforward sequence of instructions can be easily assessed as follows:

- (a) Sum the instruction times for the instructions which constitute the program. Allow 12 microseconds for an unsuccessful indicator test but add nothing for a successful indicator test.
- (b) Add  $12n$  microseconds where  $n$  is the number of words constituting the second of program. This in effect allows 12 microseconds for each control change, whether it is programmed or occurs automatically between consecutive program words.

*Example*

I	D	F	A	R
11		45	0000	3
		3	0000	5
12		37	0000	5
		62	0001	5
13		57	0001	
		62	0002	5
14		42	0000	1
	4	18	0016	B
15		61	0000	1
		42	0000	1

Suppose first that indicator 18 is set. The time can be assessed as follows:

45 instruction (3 words transferred)	78 $\mu$ s
37 instruction	21 $\mu$ s
62 instruction	21 $\mu$ s
57 instruction	17 $\mu$ s
62 instruction	21 $\mu$ s
42 instruction	21 $\mu$ s
Successful indicator test	-
12 x number of words obeyed	48 $\mu$ s
TOTAL TIME	227 $\mu$ s

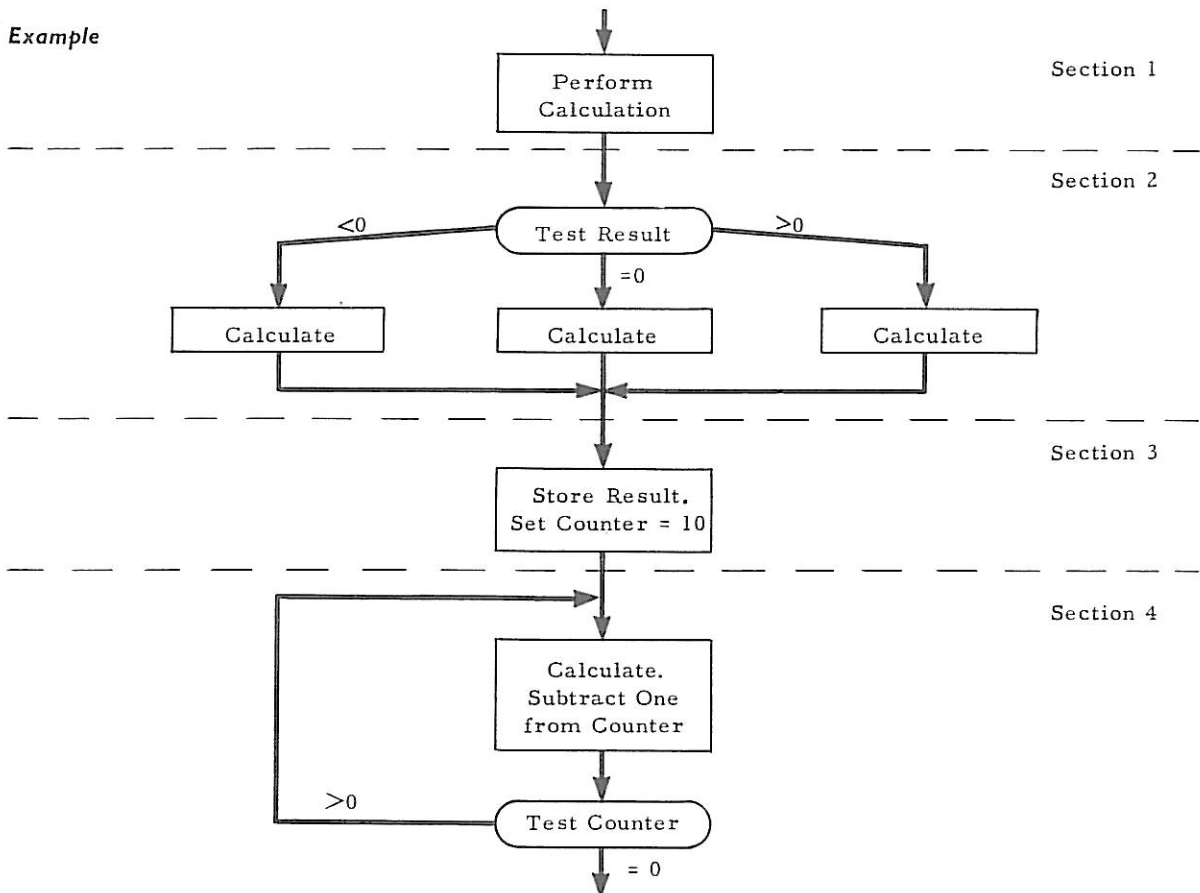
If indicator 18 is unset the times are as follows:

45 instruction (3 words transferred)	78 $\mu$ s
37 instruction	21 $\mu$ s
62 instruction	21 $\mu$ s
57 instruction	17 $\mu$ s
62 instruction	21 $\mu$ s
42 instruction	21 $\mu$ s
Unsuccessful indicator test	12 $\mu$ s
61 instruction	21 $\mu$ s
42 instruction	21 $\mu$ s
12 $\times$ number of words obeyed	60 $\mu$ s
TOTAL TIME	293 $\mu$ s

It should be noted that these times include the control jump to the next word of program. When timing a subroutine the link should be counted as one of the words constituting the program.

Timing becomes slightly more difficult when the program branches into several alternative paths or when a section of program forms a loop and is obeyed several times. The best method is to use the flowchart as a guide and to divide the flowchart into sections for timing purposes.

*Example*



Suppose that it is known that the entire calculation is to be performed 18 times and that of these, eight correspond to the case  $<0$ , seven to the case  $= 0$  and three to the case  $>0$  in section two. The complete time can then be assessed as follows:

- (a) The time for the program corresponding to section 1 can be assessed. This is multiplied by 18.
- (b) The time for each of the three branches can be calculated. The 12 microseconds for an unsuccessful test instruction should be included where appropriate. The times for the three branches are multiplied by 8, 7 and 3 respectively.
- (c) The time for the program corresponding to section 3 is assessed. This is multiplied by 18.
- (d) The time for the program corresponding to section 4 is assessed and is multiplied by 10 because the loop is obeyed 10 times. This time is then multiplied by 18.
- (e) All the times are summed.

It is not always possible to give an exact time for a piece of program. The program paths followed may depend on the data values and the calculation may contain instructions such as multiplication or drum transfers which do not take a fixed time. In such cases average and/or maximum times should be estimated. For a general purpose routine the average and maximum timings should be quoted. When a program is to be time-shared with a print or punch program the maximum timings should be calculated since it is necessary that in all cases the program should be within the permitted time available.

For a general purpose routine, it is often helpful to express the timing as a formula when exact times cannot be given. The variables in the formula may be dependent on the data or on parameters which are given to the routine. For example, the timing for a division routine might depend on the number of decimal places required in the result. This might be given to the routine as a parameter. A typical example of timings given by formulae are those for the sorting routines. These are expressed in terms of two variables, the number of words in a record and the number of records to be sorted.

When timing a large program it is not normally necessary to consider the time for each instruction. The operational speed is normally governed by the peripheral units and a time assessment can be made by considering the speed at which these will be working. Alternatively a time can be obtained by timing the program on the computer for a small amount of data.

## DOCUMENTATION

### 4.11

It is essential that, once a program has been written and tested, it is preserved not merely as a pack of cards but as a completely documented piece of work. The documentation should be such that it contains complete instructions for using the routine. Sufficient information should be included for a different programmer to be able to understand the program if it becomes necessary to amend it at a later date. The documentation should include the following details where appropriate:

Specification - A short description giving details of what the routine does and how to use it. In the case of a general purpose subroutine, this should conform to the standard layout.

Description - Any further description not included in the specification.

Flowcharts - Complete flowcharts using standard symbols. For large jobs these should be cross-referenced with the program sheets.

Input and Output Layouts - Planning charts showing the input and output card layouts and the printed output layout.

Storage Charts - Charts showing the I.A.S. and drum storage allocation.

Program Sheets - Program sheets for the entire program complete with narrative. Program sheets for the test program should also be included.

Program Cards - Complete pack of program cards correctly numbered for Initial Orders sequence check.

Test Data and Test Program - Cards for test program and data.

Operating Instructions - Complete instructions including action on error stops.

Sample Print- and Punch-outs - Sample results for the program when run with the test program.

## AN EXAMPLE OF A CODED PROGRAM

4.12

### Introduction

4.12.1

The accompanying program is an example of a weekly P.A.Y.E. calculation and should be regarded purely as a specimen piece of program and not as the standard method of P.A.Y.E. calculation which will normally be used. For the sake of simplicity, no subroutines or modification techniques have been used; therefore, this program requires considerably more than the minimum possible number of instructions. The program is only intended to demonstrate the use of the functions and indicators described earlier in this manual.

The following is a summary of the calculations involved.

Factors	Maximum Value	4.12.2
Week No.	52	
Week 1 designation	(1 = Week 1 case; 0 = normal)	
Number of holiday weeks	9	
Week 1 Free Pay	£ 99. 19. 0.	
Gross Wage Brought Forward	£9,999. 19. 11.	
Gross Wage this week	£ 99. 19. 11.	
Tax brought forward	£ 999. 19. 0.	
Total Deductions	£ 99. 19. 11.	



Results	Maximum Value	4.12.3
Gross Wage carried forward	£9,999. 19. 11	
Tax carried forward	£ 999. 19. 0	
Tax this week	£ 99. 19. 0	
Tax Deduction/Refund designation	(1 = Refund, 0 = Deduction)	
Net Wage	£ 99. 19. 11.	

**Calculations** 4.12.4

- (a) Gross Wage + Gross brought forward = Gross carried forward.
- (b) Calculation of Tax carried forward as follows:-
  - If Week 1 case:- 1 + number of holiday weeks = 'Tax Week Number'.
  - If not Week 1 case:- Week No. + number of holiday weeks = 'Tax Week No.' Set Indicator 10.
  - Week 1 Free Pay × 'Tax Week Number' = Tax Free pay to date.
  - If Week 1 case:- Gross Wage - Tax free pay to date = Net Taxable Income.
  - If not Week 1 case:- Gross carried forward - Tax free pay to date = Net Taxable Income.
  - From Net Taxable Income calculate Tax to date carried forward (for explanation of P.A.Y.E. calculation, see section 4.12.5).
- (c) Tax carried forward - Tax brought forward = Tax this week (Deducted or Refund).
- (d) Gross Wage ± Tax this week - Total Deduction = Net Wage.

**P.A.Y.E. Calculation** 4.12.5

The method of deriving Tax carried forward following the calculation of the Net Taxable Income is explained in detail below. For the sake of simplicity the 'Tax Week Number' of Section 4.12.4 will be referred to as week number, and in all references to tax it should be noted that this will actually be 'tax this week' in the case of employees being taxed on a Week 1 basis, and 'tax carried forward' in all other cases. Net Taxable Income is abbreviated to N.T.I.

Certain assumptions are made regarding Earned Income Allowance, rates of tax etc., as follows:-

Earned Income Allowance is assumed to be at the rate of  $\frac{2}{9}$ ths of the N.T.I., and in the specimen program it is assumed that no employee has a N.T.I. of over £4,005 so that it is unnecessary to allow for the reduction of the Earned Income Allowance to  $\frac{1}{9}$ th of the N.T.I. when the latter exceeds £4,005.

Over the whole year, it is assumed that, after deduction of Earned Income Allowance, the first £60 of the N.T.I. is taxed at 1/9d in the £, the next £150 at 4/3d in the £, the next £150 at 6/3d in the £, and the remainder at 7/9d in the £. At any given week, the £60 and the two £150 ranges will be the proportionate amounts up to and including that week, so that, at any week x, the ranges will be:-

$$\frac{£60x}{52} \quad ; \quad \frac{£150x}{52} \quad ; \quad \frac{£150x}{52}$$

The following is a summary of the sequence of operations involved in the P.A.Y.E. calculation in the specimen program:-

- (a) If the N.T.I. is negative or zero, the computer is programmed to ignore the whole of the tax calculation so that the result will be zero.

If the N.T.I. is positive, then:-

- (b) Tax payable at 1/9d in the £ is calculated by multiplying the whole of the N.T.I. by 0.06806, which is derived as follows:-

$$\frac{1\frac{3}{4}}{20} \times \frac{7}{9} = 0.06806 \text{ approximately.}$$

Thus the  $\frac{2}{9}$ ths Earned Income Allowance is automatically taken into account.

- (c) The £60 range factor is deducted from the N.T.I. This is calculated as follows: at Week 1 the amount to be deducted will be

$$\frac{\text{£}60}{52} \times \frac{9}{7} = \text{£}1-9-8.044 \text{ approximately.}$$

The reason for increasing the proportionate £60 range, i.e.  $\frac{\text{£}60}{52}$  by  $\frac{9}{7}$ ths is the fact that  $\frac{2}{9}$ ths Earned Income Allowance has not been deducted from the N.T.I. and therefore the factor deducted must be increased proportionately while the tax-rate decimal (e.g. 0.06806 in (b) above) is correspondingly reduced by multiplying by  $\frac{7}{9}$ . Therefore, at Week x, the factor to be deducted from the N.T.I. is found by multiplying £1-9-8.044 by the Week Number x.

- (d) Following subtraction of the appropriate £60 range factor, the computer tests whether or not the result is negative; if it is negative, no tax is due at the higher rates and there is an automatic jump to the end of the tax calculations. If the result is positive (or zero), the remainder of the N.T.I. is taxed at a further 2/6d in the £ by multiplying by .09722, which is derived as follows:-

$$\frac{2\frac{1}{2}}{20} \times \frac{7}{9} = 0.09722 \text{ approximately.}$$

Again the  $\frac{2}{9}$ ths Earned Income Allowance is automatically allowed for, and the resulting amount of tax is added on to the tax calculated in (b) above.

- (e) The first £150 range factor is deducted from the N.T.I. This is calculated in a similar manner to the £60 range factor in (c) above, the amount at Week 1 being:-

$$\frac{\text{£}150}{52} \times \frac{9}{7} = \text{£}3-14-2.11 \text{ approximately.}$$

This will be multiplied by the Week Number to give the appropriate factor to be deducted at the given week.

- (f) Following subtraction of the first £150 range factor from the N.T.I. the computer goes into the same routine as that described in (d) at a further 2/- in the £ by multiplying by 0.07778, which is derived as follows:-

$$\frac{2}{20} \times \frac{7}{9} = 0.07778 \text{ approximately.}$$

- (g) The second £150 range factor is then deducted from the N.T.I., the amount being the same as in (e) on the previous page.

- (h) Following subtraction of the second £150 range factor from the N.T.I. the computer goes into the same routine as in (d) above, except that the remainder of the N.T.I. if any, is now taxed at a further 1/6d in the £ by multiplying by 0.05833, which is derived as follows:-

$$\frac{1\frac{1}{2}}{20} \times \frac{7}{9} = 0.05833 \text{ approximately.}$$

In effect, therefore, the remainder of the N.T.I., if any, has been taxed at 7/9d in the £, being taxed at 1/9d in the £ in (b) above; a further 2/6d in the £ in (d); a further 2/- in (f), and a further 1/6d in (h).

- (i) The application to the N.T.I. of the various tax rates has now been completed, the resulting amount of tax representing the Tax to date carried forward in normal cases, i.e. those being taxed on an accumulative basis, and Tax this week in the case of employees being taxed on a Week 1 basis. Therefore, the computer now tests whether or not indicator 10 is set; if it is unset, indicating a Week 1 case, Tax brought forward is added to Tax this week to give Tax carried forward for all cases, Week 1 or normal, and Tax this week can then be calculated (see Section 4.12.4).

## Timing

### 4.12.6

The total time required for the program will be very variable, depending on the size of the Net Taxable Income, the minimum being the case where the Net Taxable Income is zero or negative, in which case the computer skips over instructions 11-37 inclusive (see Figure 46), and the time required will be about 1 millisecond. On the other hand, when the Net Taxable Income is sufficiently large for tax to be paid at the highest rate, i.e. 7/9d in the £, the time required will be about 6 milliseconds.

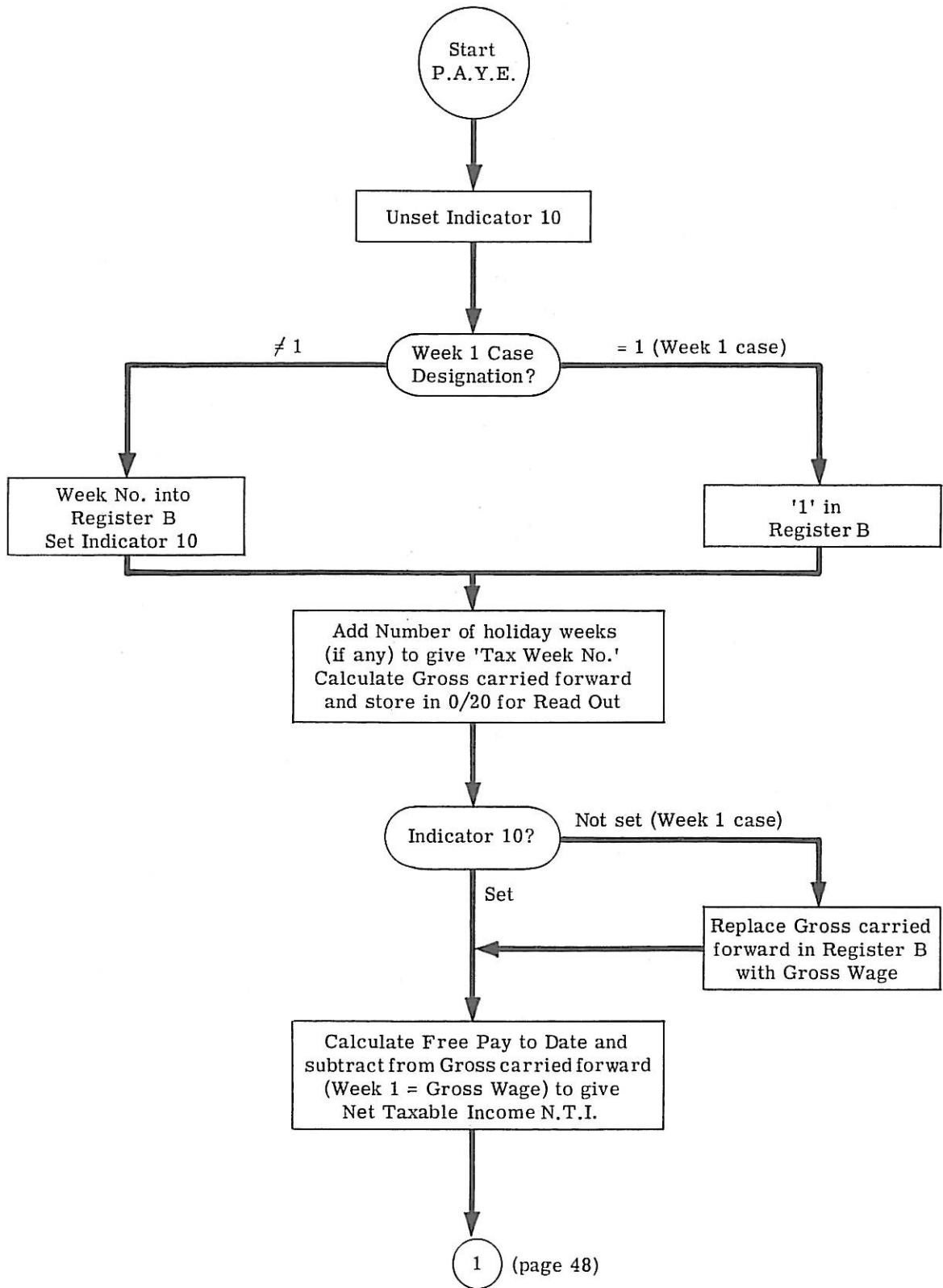


Figure 45: FLOWCHART OF P.A.Y.E. PROGRAM

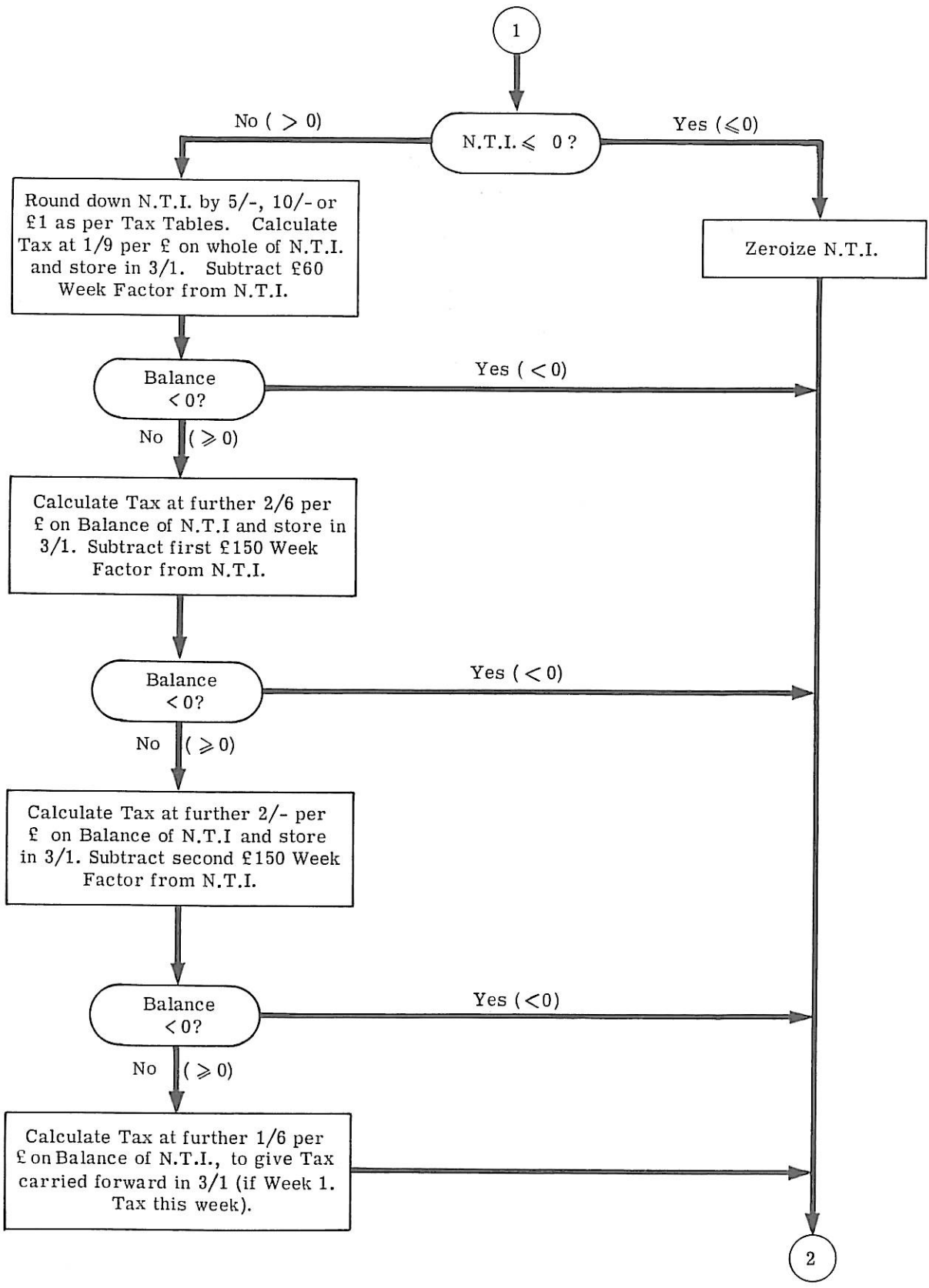


Figure 45: continued

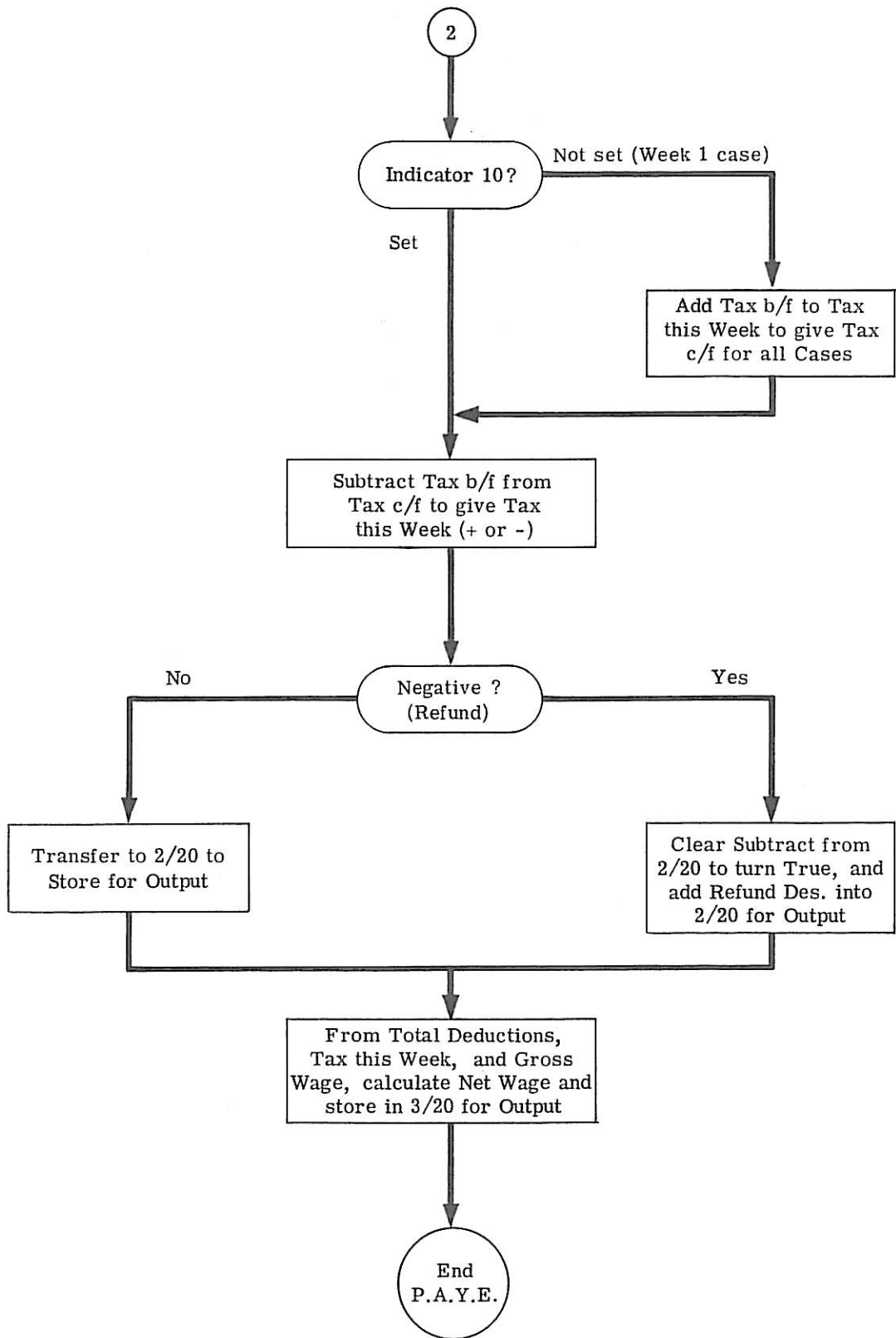


Figure 45: continued

I.C.T COMPUTERS

1300 SERIES PROGRAM SHEET		JOB <i>P.A.Y.E. (Weekly)</i>				BLOCK No. <i>10</i>
		PROGRAMMER:-				SHEET No. <i>1/4</i>
C	I	D	F	A	R	NARRATIVE
1		B				
2	0	9	10	0	-	Ensure that indicator 10 is unset.
			60	1	18	Test if Wk. 1 des. (0 or 1 in Register B)
	1	4	02	3	B	If Week 1 case jump to word 3
			60	0	18	If not Wk. 1 case basic Week No. into Reg. B
2	8	10	0	-	If not Week 1 case set indicator 10	
		4	00	3	B	Jump to word 3
3	3		62	2	18	Add number of Holiday Weeks to given 'Tax Week'
			42	0	1	Store on 0/1
	4		21	0	-	Set Decimal Point Register = 0
			22	7	-	Set Sterling Register = 7
	5		79	3	18	Week No. x Week 1 Free Pay = Tax free pay to date
			42	1	1	Store on 1/1
4	6		37	4	18	Gross brought forward into Register B
			72	5	18	Add Gross this week to give Gross c/f
	7		42	0	20	Store on 0/20 for output
		4	10	9	B	If not Week 1 case jump to word 9
	8		37	5	18	If Week 1 case replace Gross c/f by Gross Wage
4		00	9	B	Jump to word 9	
5	9		73	1	1	Subtract Free Pay to Date to give Net Taxable Income (N.T.)
		4	02	11	B	If N.T. > 0 jump to word 11
	10		57	12	-	If N.T. ≤ 0 zeroize
		4	00	38	B	If N.T. < 0 jump to word 38 skipping tax calculations
11		42	2	1	Store N.T. (unrounded) on 2/1	
		35	2	13	Mask to extract unit shillings and pence	
6	12		73	0	13	Subtract 5/- from unit shillings and pence
			37	2	1	Net taxable into Register B
	13		57	5	-	Eliminate unit shillings and pence
			54	5	-	
	14	4	03	15	B	If unit shillings and pence < 5/- jump to word 15
			72	0	13	If unit shillings and pence ≥ 5/- add 5/- to complete 5/- round down

Figure 46: MAIN PROGRAM

I.C.T COMPUTERS

1300 SERIES PROGRAM SHEET		JOB <i>P.A.Y.E (Weekly)</i>				BLOCK No. <i>10</i>
		PROGRAMMER:-				SHEET No <i>2/4</i>
C	I	D	F	A	R	NARRATIVE
7	15		42	2	1	<i>N.T. transferred back to 2/1</i>
			37	0	1	<i>'Tax Week No.' into Register B</i>
	16		54	7	-	<i>Shift to 100's and 10's £s position</i>
			73	2	1	<i>Subtract N.T. from 10 x Week Number</i>
	17		37	2	1	<i>N.T. into Register B</i>
		4	02	19	B	<i>If N.T. &lt; 10 x Week No. jump to word 19</i>
8	18		57	5	-	} <i>If N.T. ≥ 10 x Week No. eliminate unit shillings and pence to complete 10/- round down</i>
			54	5	-	
	19		42	2	1	<i>N.T. back to 2/1</i>
			37	0	1	<i>'Tax Week No.' into Register B</i>
	20		69	1	13	<i>Week No. x 16 on £s position in Reg. B</i>
			73	2	1	<i>Subtract N.T. from 16 x Week No.</i>
9	21		37	2	1	<i>N.T. into Register B</i>
		4	02	23	B	<i>If N.T. &lt; 16 x Week No. jump to word 23</i>
	22		57	6	-	} <i>If N.T. ≥ 16 x Week No. eliminate all shillings to complete £1 round down</i>
			54	6	-	
	23		42	2	1	<i>N.T. (round down) to 2/1</i>
			37	5	13	<i>.06806 (1/9 rate decimal) into Reg. B</i>
10	24		21	5	-	<i>Set Decimal Point Register = 5</i>
			79	2	1	<i>Whole of N.T. x 1/9 rate decimal = Tax @ 1/9 per £</i>
	25		42	3	1	<i>Transfer Tax @ 1/9 rate to 3/1</i>
			37	0	1	<i>'Tax Week No.' into Register B</i>
	26		79	3	13	<i>Week No. x Week 1 Factor = current week factor (£60 range)</i>
			75	2	1	<i>Subtract £60 factor from N.T.</i>
11	27	4	03	38	B	<i>If N.T. &lt; £60 factor jump to word 38</i>
			37	6	13	<i>.09722 (2/6 rate dec.) into Register B</i>
	28		21	5		<i>Set Decimal Point Register = 5</i>
			79	2	1	<i>Remainder of N.T. x 2/6 rate dec. = Tax at further 2/6 per £</i>
	29		74	3	1	<i>Add on to Tax in 3/1</i>
			37	0	1	<i>'Tax Week No.' into Register B</i>

Figure 46: continued



I.C.T COMPUTERS

1300 SERIES PROGRAM SHEET		JOB <i>P.A.Y.E. (weekly)</i>				BLOCK No. <i>10</i>
		PROGRAMMER:-				SHEET No <i>3/4</i>
C	I	D	F	A	R	NARRATIVE
12	30		79 75	4 2	13 1	<i>Week No. x Week 1 Factor = current week factor (£150 range) Subtract £150-factor from remainder of N.T. If remainder of N.T. &lt; £150 factor, jump to word 38</i>
	31	4	03 37	38 7	B 13	<i>.07778 (2/- rate dec.) into Register B</i>
	32		21 79	5 2	- 1	<i>Set Decimal Point Register = 5 Remainder of N.T. x 2/- rate dec. = Tax at further 2/- per £.</i>
13	33		74 37	3 0	1 1	<i>Add to Tax in 3/1 Tax Week No. into Register B</i>
	34		79 75	4 2	13 1	<i>Week No. x Week 1 Factor = current week factor (second £150 range) Subtract second £150 factor from remainder of N.T.</i>
	35	4	03 37	38 8	B 13	<i>If remainder of N.T. &lt; second £150 factor jump to word 38 .05833 (1/6 rate decimal) into Register B</i>
14	36		21 79	5 2	- 1	<i>Set Decimal Point Register = 5 Remainder of N.T. x 1/6 rate dec. = Tax at further 1/6 per £</i>
	37	4	74 00	3 38	1 B	<i>Add on to Tax in 3/1</i>
	38	4	37 10	3 40	1 B	<i>Tax c/f (Week 1 case - Tax this week) into Register B If not Week 1 case jump to word 40</i>
15	39	4	72 00	6 40	18 B	<i>If Week 1 case Tax b/f added to Tax this week gives Tax carried forward Jump to word 40</i>
	40		42 73	1 6	20 18	<i>Transfer Tax carried forward to 1/20 for output Subtract Tax brought forward to give Tax this week (plus or minus)</i>
	41	4	40 03	2 43	20 B	<i>Ensure that 2/20 is zero If tax negative (refund) jump to word 43</i>
16	42	4	42 00	2 44	20 B	<i>If tax positive transfer to 2/20 If tax positive, jump to word 44</i>
	43		75 76	2 2	20 20	<i>If tax negative subtract from 2/20 to turn true If tax negative add refund des. into 2/20</i>
	44		42 37	4 5	1 18	<i>Transfer Tax (±) to 4/1 Gross this week into Register B.</i>

Figure 46: continued



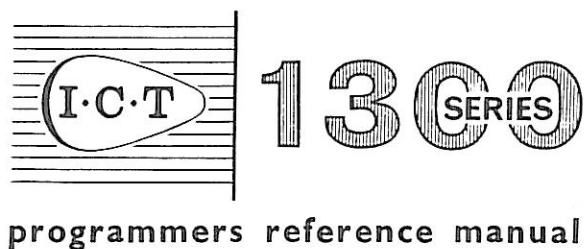












programmers reference manual

## SOFTWARE FACILITIES

### Contents

	Page
5.1 THE I.C.T. SUBROUTINE LIBRARY ... ..	1
5.1.1 Index of Library Routines ... ..	1
5.1.2 Subroutines ... ..	2
5.1.3 I.C.T. 1300-series Library Subroutines ... ..	2
Specification Sheets ... ..	3
Flowcharts and Program Sheets ... ..	3
Entry to and Exit from Main Program ... ..	3
Storage and Relativizers ... ..	4
Indicators ... ..	4
5.2 AUTOCODES ... ..	4
5.2.1 Rapidwrite ... ..	6
5.2.2 TAS ... ..	6
5.2.3 MPL ... ..	7
5.2.4 MAC ... ..	8
5.3 P.P.F. PROGRAMS AND INPUT/OUTPUT ROUTINES ...	8
5.3.1 P.P.F. Programs ... ..	8
5.3.2 Input/Output Routines ... ..	9
5.4 MAGNETIC TAPE CONTROL AND UTILITY ROUTINES ...	10
5.4.1 Tape Layout ... ..	10
Tape Labels ... ..	10
Block Layout ... ..	10
Record Layout ... ..	11



Contents continued	Page
5.4.2 Tape Housekeeping Routines ... ..	11
Job Set-up ... ..	12
Write Program to Tape ... ..	13
Magnetic-tape Sorting ... ..	13
Merging Using Four Tape Decks ... ..	13
Merging Using Three Tape Decks ... ..	13
Sorting Using Three Tape Decks and the Magnetic Drum ... ..	14
5.5 GENERATOR ROUTINES ... ..	14
5.6 DIAGNOSTIC ROUTINES ... ..	15
5.6.1 Validity Check ... ..	15
5.6.2 Proving Storage ... ..	15
5.6.3 Memory Dump ... ..	15
5.6.4 Trace Routine ... ..	16
5.6.5 Program Updating Routine . ... ..	18
5.7 UTILITY ROUTINES ... ..	18
5.7.1 Division Routines ... ..	19
5.7.2 Parity Error Routines ... ..	19
5.7.3 Zero Suppression Routines ... ..	19
5.7.4 Punching Program into Fast-read Cards ... ..	20
5.7.5 Sorting Routines ... ..	20
Merging Sort ... ..	20
Extraction Sort ... ..	22
Exchanging Sort ... ..	22
5.8 COMMERCIAL ROUTINES ... ..	22
5.8.1 P.A.Y.E. ... ..	22
5.8.2 Graduated Pension Contributions ... ..	23
5.8.3 Sterling and Decimal Conversions ... ..	23
5.9 MATHEMATICAL AND STATISTICAL ROUTINES ... ..	23
5.9.1 Floating Point Arithmetic ... ..	23
5.9.2 Matrix Arithmetic ... ..	24
5.9.3 Double-length Arithmetic ... ..	24
5.9.4 Special Mathematical Routines ... ..	25
5.9.5 Manchester AutoCode ... ..	25

## Illustration

Figure 51 Sorting Using Three Tape Decks and the Magnetic Drum ... ..	14
--------------------------------------------------------------------------	----

### THE I.C.T. SUBROUTINE LIBRARY

5.1

There is a library of general purpose programs written for the 1300-series computers which are available from I.C.T.

The majority of these routines are subroutines which can be incorporated into the user's own program. There are also programs which are complete in themselves and for which the user need only supply data. Autocodes, which provide facilities for programs to be written in a simpler form than machine-coded instructions, are available and are discussed in more detail later. This part of the manual does not cover every routine in the library but is intended as a guide to the types of routines available.

#### Index of Library Routines

5.1.1

An index to the routines available is supplied to 1300-series users, together with Specification Sheets that contain sufficient information about each routine to enable the user to select an appropriate routine for his purpose. In some instances a broad class of routines will be the subject of a general paper explaining the best use of each individual routine within that class.

Each individual routine is classified and numbered. The class letter gives the functional division. This is followed by a two-digit number allocated to further identify the function of the routine. The final two-digit number differentiates between routines that have similar functions or achieve the same result in different ways.

For example, a group of routines for printing control is classified as follows:

A/--/--	Indicates an Input/Output routine.
A/02/--	Indicates a print routine.
A/02/06	Prints on one bank only and spaces n lines.
A/02/07	Prints on two banks and spaces n lines.
A/02/08	Prints on three banks and spaces n lines.

## Subroutines

## 5.1.2

A subroutine is a self-contained section of program which may be used more than once in a program, or has a general application and can form part of a number of different programs. In scientific work, for example, the trigonometrical functions sine and cosine, and square roots are required in many routines.

Once programmed as subroutines, any program can include them. In commercial data processing, there is not quite so much scope for generalized subroutines. The P.A.Y.E. calculation is one example of a standard routine that can be incorporated into a program.

On the 1300-series Computers, division has to be programmed and is written as a subroutine. Input and output (P.P.F. programs) are also programmed in subroutine form.

The object, therefore, of a subroutine can be either:-

- (a) To save storage space by writing a section of program once only and jumping to that section whenever it is required in the main program, or,
- (b) To make programming quicker and easier by incorporating pre-programmed subroutines when available.

Subroutines are useful inasmuch as they can:

- (a) Save storage space
- (b) Save programming time
- (c) Save testing time, as a library subroutine is well planned and fully tested,
- (d) Be used time and again with no further effort.

## I.C.T. 1300-series Library Subroutines

## 5.1.3

The I.C.T. 1300-series Subroutines comprise a collection of well tested routines, which cover:-

Input/Output  
Magnetic Tape  
Commercial (P.A.Y.E. etc.) Routines  
Arithmetical (Division etc.) Routines  
Mathematical Routines  
Utility Routines  
Diagnostic Routines.

Library routines should be used whenever possible for they are proved routines planned to economize in the use of storage space and computer time. By linking them to a main program, the user will reduce considerably the amount of programming and testing time required to achieve successful running of the program.

A Library subroutine complies with a standard format so that a complete subroutine comprises:

Specification Sheet      Flowchart  
Program Sheets          Program Card Pack.

### Specification Sheets

The following information is shown as fully as possible on all subroutine specification sheets:-

(a) Entry Points

The entry point or points are stated. If more than one, the distinction between them is stated.

(b) Entry Conditions

Locations of variables and parameters assumed by the subroutine on entry are given; also details of the required indicators, and their state.

(c) Results

Results produced by the subroutine together with locations occupied are given.

(d) Storage

The number of words required for the storage of the program and constants is given. Under the heading of temporary storage is shown the number of words used as working space and whose original content is immaterial.

(e) Time

If possible, either exact timings are given or else formulae from which processing time can be calculated.

(f) Limitations

Any limitations in the size of factors and accuracy of results are given.

(g) Error Conditions

A brief account of tests made by the subroutine for any error conditions, together with the action taken if such errors are detected, is given.

(h) Notes

Any other necessary information about the subroutines, not given in other sections, is included here.

### Flowcharts and Program Sheets

The specification sheet should normally contain sufficient information for the user to be able to incorporate the subroutine into his program. If, however, further knowledge of the subroutine is required then flowcharts and program sheets can usually be obtained. Some programming points relevant to subroutines are considered below.

### Entry to and Exit from Main Program

The instruction to jump from the main program to the first instruction of a subroutine is a normal jump instruction.

For example, either

C	I	D	F	A	R
	10	4	00	0000	17

or

C	I	D	F	A	R
	21	4	00	0000	17

would effect a jump from the main program to the first instruction of a subroutine in I.A.S. location 0 of block 17.

The subroutine is so written that, when it has been completed, it automatically restores control to the instruction in the main program immediately following the jump to the subroutine.

### **Storage and Relativizers**

Storage addresses written relative to the first location of a subroutine will use block relativizer B. Temporary storage will use relativizer 1. If other relativizers are needed by the subroutine (e.g. for a block of data), relativizers 3 to 9 will be used.

### **Indicators**

Unless specified as an entry condition, library subroutines make no assumptions about the state of indicators on entry to the subroutines, and users of subroutines should make no assumptions about the states of indicators used by subroutines when re-entering the main program.

Programmed indicators employed by subroutines will be used in the reverse order 19, 18, 17....

Library subroutines will not test the I.A.S. parity indicator unless this is stated on the specification sheet.

It is important to note that Library subroutines can be assumed to be resetting unless a definite statement to the contrary appears on the specification sheet.

## **AUTOCODES**

### **5.2**

An autocode enables a programmer to write a program in a symbolic language which uses alphabetic names and characters instead of numeric references. In some autocodes this use of alphabetic names enables the program to be written in comprehensible English. A special program is then employed to translate or interpret the autocode language program into machine coding before it is obeyed by the computer. Thus the autocode method of programming is designed to:

- (a) enable programs to be written in less time,
- (b) reduce the amount of writing (and similar clerical work), and therefore the chance of error with a consequent reduction in testing time,
- (c) simplify program maintenance,
- (d) make computer procedure comprehensible to people not trained in the machine code and to enable these people to prepare programs with the minimum amount of training,
- (e) standardize coding tactics and therefore permit the exchange of ideas and programs among computer users.

When writing a program in autocode, the programmer does not have the same control over the instructions or positions of storage as when writing a machine-coded program. Consequently the programmer cannot adapt the program to suit his own requirements ideally. Thus a program written in autocode is not usually quite as efficient in time or storage as a program written in machine code. This disadvantage of autocode program efficiency is usually more than offset by the advantages listed above.

Initially, a source program is written in the autocode, which is a restricted form of language conforming to certain defined rules. The source program must then be transformed into a machine-coded program: this machine-coded program is termed the object program. The transformation is done by the computer under the direction of a processor program.

The processor program may be either an interpreter or a translator. A translator transforms the entire source program into the machine-language object program; the resultant object program is then stored in its entirety and is either output in the form of punched cards or obeyed immediately.

If the object program is obeyed immediately the complete translation of the source program is achieved, then the system is known as a load-and-go system. The term load-and-go is derived from the fact that the complete process of translation and execution of the translated program is achieved in a single run on the computer.

An interpreter transforms the source program into machine-coded instructions and each machine-coded instruction is obeyed immediately it has been created. Thus no complete object program is produced and is therefore not available as output.

The programming language may be a machine-oriented language or a procedure-oriented language.

A processor for a machine-oriented language is termed an assembly system or assembler. A machine-oriented language is devised for a particular type of computer and consists of a code which is easier to learn than machine code. Programs can also be written more quickly using an assembly language than using machine code. There is, however, a direct connection between the two, and one source program instruction often results in one machine-coded instruction in the object program. An exception to this arises for input and output which is concisely achieved using an assembly system.

A processor for a procedure-oriented language is termed a compiler. A procedure-oriented language is not necessarily devised for a particular computer. The same language may be used for several different types of computer, each computer having its own compiler for converting the source program into the appropriate machine-coded program. The source program for a compiler resembles a restricted form of language which conforms to the rules laid down for the autocode. A compiler is more powerful than an assembler and one statement in a procedure-oriented language usually results in several machine-coded instructions in the object program.

Further, an assembly system or compiler will be oriented towards the application to which the source programs refer. For example, Rapidwrite is a commercially-oriented language and MAC (Manchester AutoCode) is a scientifically and mathematically-oriented language.

There are at present four autocodes recommended for use on I.C.T. 1300-series Computers. They are

<i>TAS</i> ( <u>T</u> hirteen-hundred <u>A</u> ssembly <u>S</u> ystem)	<i>Rapidwrite</i>
<i>MPL</i> ( <u>M</u> nemonic <u>P</u> rogramming <u>L</u> anguage)	<i>MAC</i> ( <u>M</u> anchester <u>A</u> uto <u>C</u> ode)

Brief introductory details of these four autocodes are given in the following paragraphs and a reference can be made to the appropriate manuals, which give full details.

The use of these autocodes is dependent upon certain minimum machine requirements (such as the size of I.A.S. and magnetic drums etc.). Thus several variations of each processor may exist. The specification sheets for the I.C.T. Subroutine Library Routines give full details of machine requirements and storage.

## **Rapidwrite**

### **5.2.1**

Rapidwrite is a commercially-oriented autocode. The source program is converted to machine code by a translator/compiler program.

Initially, a source program is prepared using specially designed pre-printed forms and dual-purpose cards. The latter enable the program to be built up on the lines of a flowchart with moveable blocks. The fixed format of the dual-purpose cards and the forms reduces the amount of writing and punching and, in consequence, errors resulting from these operations are also reduced. The source program is read into the computer and a preliminary routine produces a full descriptive print-out of the program in English. When the program has been corrected by using this routine, it is again read into the computer and the object program is compiled. The object program will be both punched and printed. The print-out of the object program will contain a reference to the print-out obtained from the preliminary routine. The object program will be ready for testing after insertion of any required subroutines and the addition of a standard program.

A Rapidwrite source program is written in three clearly defined divisions: Environment, Procedure and Data divisions.

The Environment division, which is filled in on a pre-printed form, deals with the specification of the computer and includes such data as the size of I.A.S. and drum storage and console indicators which are to be used.

The Procedure division deals with instructions handling the processing of the problem and is written on special dual-purpose cards and one card is allocated for each instruction such as READ, WRITE, COMPUTE, IF or GO.

The Data division which deals with the organization of data is written on special forms. Data names not exceeding five characters in length are used and if required these can be translated into full data names, up to 30 characters in length, by supplying the processor with a list showing the full names against their abbreviations. The data form when complete will contain all the relevant information on the input and output files required by the processor for the storage of data, the kind of information held in a field and the allocation of working storage.

## **TAS**

### **5.2.2**

TAS (Thirteen-hundred Assembly System) is a commercially-oriented autocode and is provided as an intermediate language which lies between machine language and a full autocode language such

as Rapidwrite. Thus, a programmer writing in TAS will require a greater knowledge of the machine code than one using Rapidwrite. The source program is converted to machine code by a translator/assembler program. Each TAS instruction generates an *average* of four machine code instructions and thus writing time is considerably reduced. TAS is ideal in cases where programs are needed urgently or for small programs e.g. those requiring less than two weeks machine coding.

The source program is written in TAS language on special stationery from which one card is hand punched for each line of entry. These cards are then fed to the computer and the object program is produced on fast-read program cards in one run. A print-out of the source program and other print-outs to aid the programmer can also be produced.

To write a TAS program, it is necessary to:

- (a) Specify tables for storage on the drum.
- (b) Describe each Input card.
- (c) Describe each Output line.
- (d) Describe each Output card.
- (e) Write the procedure program from a selection of functions with the facility of adding subroutines and machine-coded instructions.

Data read from a card or printed on a line are referenced in the program as a field name, the name being introduced when the programmer describes each card or line of print. Data of a bulky or lengthy nature are best handled in the form of tables which are stored on the drum.

There is an assembler available which allows the programmer to use magnetic-tape storage and special instructions are included in the autocode for controlling the reading and writing of tape. Facilities are provided for the programmer to specify his own tape records, each record being described as a series of field names.

## MPL

## 5.2.3

MPL (Mnemonic Programming Language) is a commercially-oriented language specially designed for users with a small machine configuration which does not permit the use of TAS or Rapidwrite etc. MPL may be used on a machine with the basic configuration of 400 words I.A.S. and a 3,000 word drum. The MPL processor may be used as a load-and-go assembler or may punch out the object program, the appropriate mode of operation being selected by means of a switch.

The source program is written in the MPL language which has a one-character function code and five-character operand. Most MPL instructions result in a corresponding machine-code instruction being generated i.e. on a one-for-one basis. The main exceptions to this are the powerful input/output and division macro-instructions.

When the object program has been created, it is stored automatically by the assembler. When the object program has been stored, the assembly program is overwritten by a standard control program pack which contains an input/output package. This package may incorporate either a P.P.F. control routine or a routine for serial batch processing.



When the standard control program pack has been read in, it is possible to run the program, for it is not necessary to punch out the object program before testing takes place. When the object program has been satisfactorily tested, a proved object program may be punched on cards.

## MAC

### 5.2.4

MAC (Manchester AutoCode) is a mathematically and scientifically-oriented autocode and has been designed for programs which comprise the solution of mathematical equations or formulae.

The MAC compiler operates on the load-and-go principle only. Thus the object program is compiled each time a job is run. This is not however significant since the time taken to compile the object program is small. Further, since the problems for which MAC is used are of a mathematical nature, the object program is not normally run at regular intervals as is a commercial program.

A MAC source program comprises autocode instructions and directives. Usually a MAC instruction will result in several machine-coded instructions being produced: these instructions forming the object program.

Directives transmit information to the translator for allocation of storage and do not form part of the object program.

The translator converts MAC instructions in the source program to machine-coded instructions which will produce for example, the procedure to solve an equation. A trigonometrical or logarithmic function may be initiated in MAC using a single instruction.

## P.P.F. PROGRAMS AND INPUT/OUTPUT ROUTINES

### 5.3

### P.P.F. Programs

#### 5.3.1

It is usual for the computer to process jobs that involve the use of the card reader and punch and the printer. In general, one card being read will not result in one card being punched or one line being printed. Assume, for a practical example, that the duty cycle consists of five cards read, one card punched and six lines printed. This task could be tackled serially, as follows:-

Program to read five cards  
Program to process data  
Program to punch one card  
Program to print six lines.

On a machine with a 600 cards a minute reader, 100 cards a minute punch and 600 lines a minute printer the time taken to read five cards is 500 milliseconds; the time taken to punch one card is 600 milliseconds and the time taken to print six lines is 600 milliseconds. Consequently, the input/output share of the duty cycle takes 1.7 seconds. If the three input/output programs were integrated into one program, so that the time taken was the time of the longest operation, i.e. 600 milliseconds, far greater efficiency would result and the task would be tackled in parallel as follows:-

Program to read five cards punch one card and print six lines

Program to process data.

The punch and the printer are, of course, operating on output data from the previous processing.

The peripheral units are cyclic in operation in that, having produced or accepted one unit of information, they will not produce or require another unit of the information until a set time interval has elapsed. The program dealing with this unit of information may not take as long as this interval and there may be spare time that can be used by the other units.

Therefore, integration is possible without taking any additional time over that taken for the longest single operation.

Print, Punch and Feed (P.P.F.) programs are written as three separate programs - a card reader program, a card punch program and a line print program linked together by means of a control routine which ensures that the priority of the different programs is in accordance with the requirements of the input/output units. With a P.P.F. in operation there is no further spare time in which to process data, so that duty cycle is:-

#### P.P.F. - Process.

The main program sets certain keys before entering a P.P.F. These keys instruct the P.P.F. as to how many cards are to be read and punched and how many lines are to be printed. These keys are not necessarily constant and may vary throughout a program to permit, for example, the number of lines of print produced to be different each time the P.P.F. is entered. Several comprehensive P.P.F. programs are available from the I.C.T. Subroutine Library.

As card readers and line printers with differing speeds are available, the most efficient duty cycle and the timings will be different for different machines. The example given above applies to a machine with a 600 cards a minute reader and 600 lines a minute printer. For a 300 cards a minute reader and 300 lines a minute printer, the variables could be as follows:-

Duty cycle:-	3 cards read	Time to read 3 cards is 600 ms
	1 card punched	Time to punch 1 card is 600 ms
	3 lines printed	Time to print 3 lines is 600 ms.

Thus the input/output share of the duty cycles is 1.8 seconds but using P.P.F. the time taken will be 600 milliseconds.

### Input/Output Routines

### 5.3.2

Besides the P.P.F. routines, there are also Library routines available for the individual control of each peripheral unit. Some of these provide an exit from the subroutine to enable the programmer to time-share some of his main program with the peripheral unit program.

General routines are also available which distribute data into the format required by the output routines, or alternatively distribute information read by an input routine into a form suitable for processing. The positions of the data fields for processing are communicated to the distribution routine by keys which are set by the main programmer.

## **MAGNETIC-TAPE CONTROL AND UTILITY ROUTINES**

5.4

### **Tape Layout**

5.4.1

The layout and organization of data on magnetic tape should conform to an I.C.T. standard format. Essentially, data are recorded on tape in blocks and the length of a block is defined by an end of block marker; see 3.8.1. Data within blocks can be grouped into fixed- or variable-length records. It is the programmer's responsibility to arrange data into conveniently sized records and blocks.

There are three main considerations in tape layout:

- (a) Tape Labels
- (b) Block Layout
- (c) Record Layout.

#### ***Tape Labels***

Two identification labels must be written to tape; these are the beginning of tape label and the end of tape label.

The beginning of tape label precedes the first data block. This label contains information by which the reel can be identified, together with a date at which the tape can be overwritten and a count of the number of times the tape has been written. The standard format for the beginning of tape label is described in the Tape Housekeeping Manual.

The use of beginning of tape labels ensures that the correct tapes have been fitted and prevents master information being overwritten if a wrong tape is loaded.

The end of tape label must appear at the end of every tape reel. For a single reel file the end of tape label is an end of file label. However, if a file comprises several reels, all but the final reel must bear an end of reel label. The end of tape label contains a count of the number of blocks in the reel. The standard layouts for the end of reel and end of file labels are described in the Tape Housekeeping Manual.

Other than the beginning of tape label block, all label blocks are short blocks, i.e., blocks of exactly four words, including the end of block marker.

#### ***Block Layout***

A block will consist of a number of sorted or unsorted records. Each record is identified by a key. The first word (word 0) of a block consisting of sorted records must contain the most

-significant word of the key of the last record in the block (i.e. the highest key). If a block consists of unsorted records, the contents of the first word are immaterial.

Word 0 is followed by the first record in the block.

The last word of the block is the end of block marker i.e. a word of  $\overline{15}$ s.

*Example*

INTER-BLOCK GAP	
Word 0	1st word of key of record number 3
Word 1	007 XXX XXX XXX
.	}
Word 3	
Word 4	2nd word of key
.	}
Word 8	
.	}
Word 10	
Word 11	2nd word of key
.	}
Word 33	
.	}
Word 35	
Word 36	2nd word of key
.	}
Word 49	

INTER-BLOCK GAP

### Record Layout

The three most-significant digits of the first word of each record must contain the number of words in that record.

The key, identifying the record, may consist of either alphabetic or numeric data. If an alphanumeric key is required to be used during sorting, then the data must conform to the ZNZNZN ----ZN format (where Z = Zone and N = Numeric), i.e. the code components for each character must be adjacent. The key may consist of any number of consecutive words (with the exception that some routines, sorting for example, limit the number of words in a key) and the first word of the key may be located anywhere in the record. The position of the first word of the key within the records of one file must be constant. For example in the illustration above, two words precede the first word of the key in each record. The number of words preceding the first word of the key will be a parameter that has to be given to Tape Sorting and other routines. Keywords must have values in the range of 000 000 000 001 to 499 999 999 999 inclusive.

### Tape Housekeeping Routines

5.4.2

I.C.T. provide programs to cover all aspects of reading data from, and writing data to, magnetic tape. This series of programs is known collectively as Tape Housekeeping Routines. A manual is available that gives full details of these routines.

Housekeeping routines are provided to simplify the using of magnetic tape by assuming control of all reading, writing and associated contingencies. These contingencies include correction of errors, treatment of end of reel, end of file and beginning of reel conditions.

The routines are usually in the form of program packages, that is, a number of interdependent programs controlling all the various aspects of reading or writing tape.

For example a Tape Write Package is a package consisting of programs to facilitate:

Tape Preparation

Write Label Check

Tape Write

Write Exceptions.

Briefly, these programs cover such contingencies as:

- (a) checking the initial setting-up of the appropriate deck by testing for mechanical readiness,
- (b) testing for presence or absence of a writing ring,
- (c) the checking and identification of labels,
- (d) the writing of tape,
- (e) error detection and correction, end of tape condition and short blocks etc.

Packages are available for writing and reading tape and also for controlling both reading and writing.

A full list of Tape Housekeeping routines available will be found in the Subroutine Library Index of Specification Sheets.

It should be noted that certain conventions are observed in the writing and subsequent using of the Housekeeping routines. These conventions are described fully in the Tape Housekeeping Routines Manual.

### ***Job Set-up***

The Job Set-up routine is employed at the commencement of any job using magnetic tapes and ensures that the correct tapes have been fitted for the job and that the tape decks are mechanically ready. The routine will create labels on write tapes, check labels on read tapes, and set up areas of information on the drum and in I.A.S. without which other Tape Housekeeping routines will not function correctly. Thus, this routine will ensure that the tape decks are correctly loaded and ready for a job to be run. On exit from this routine, the tapes are positioned for writing or reading after the beginning of tape label. If the user program is held on tape this routine will load that program on the drum.

### ***Write Program to Tape***

A program stored on the drum may be written to tape by using two Subroutine Library routines: 'Write Program to Tape' and a selected write to tape program. The former routine prepares a program held in absolute form on the drum for writing to tape and the selected write routine enables this program to be written to tape.

The Write Program to Tape routine performs no tape operations but prepares the program to be written to tape from the drum in the following manner:

- (a) The spurious end of block markers are removed i.e. any constants consisting of words of  $\overline{15}$ s which are not intended as end of block markers.
- (b) The program is arranged in contiguous locations.
- (c) Standard program labels are manufactured.
- (d) The program is arranged in blocks of a specified size.

An exit is made to the user's write routine for the writing of each block.

### ***Magnetic-Tape Sorting***

Several routines which facilitate the sorting of data on magnetic tape are available from the Subroutine Library. These routines usually require the use of four or three tape decks or three tape decks and a magnetic drum.

#### ***Merging Using Four Tape Decks***

The most fundamental method of sorting is the merge. In this routine records are written from two input tapes, formed into pairs in sequence and read to two output tapes. This sequence of operations is known as a pass. On the next pass, the tapes reverse their rôles, that is, the tapes that were used previously for output are now used for input and vice versa. The pairs of records are merged to form groups of four and on the next run groups of eight and so forth, the length of the group being doubled on each pass. A group of records in sequence of this nature is known as a string. The merging is continued until all the records form one string. It will require  $\log_2 N$  passes, where N is the number of records to be sorted, the logarithm being taken as the next highest integer.

In order to save computer time, before entering the merge routine, the records are normally partially ordered (using a special pre-stringing routine) into strings of ordered records. These strings are each an integral number of blocks. The pre-stringing routine arranges the blocks from one tape onto two tapes ready for input to the merge routine. The merging sort then requires  $\log_2 S$  passes, where S is the number of strings.

#### ***Merging Using Three Tape Decks***

A Tape Record Merge using three tape decks operates in basically the same manner as that required for the four tape deck version. Merging takes place between two input tape decks, the resultant strings being written on a third tape deck. The next pass reads the strings from the third tape deck so that the strings are written alternately to the two tape decks initially employed

for input. Thus two tapes are prepared for a subsequent merge pass. Merging using three tape decks doubles the number of passes required to produce a completely sorted tape by using four tape decks.

This routine should be used in preference to the Three Tape and Drum Sort described below when large volumes of data are involved.

**Sorting Using Three Tape Decks and the Magnetic Drum**

If part of the magnetic drum is available for use in the sort, a routine is available so that sorting may be achieved by the following method.

Initially a batch of records are read from deck address 1 to the magnetic drum; see Figure 51.

These records are then sorted on the drum and written to tape on deck address 2. On the next pass a second batch of records are read from deck address 1 to the magnetic drum, sorted internally and then merged with the records on deck address 2 to deck address 3.

Subsequent merging is from deck address 3 to deck address 2, then back again until the whole of the input file has been sorted.

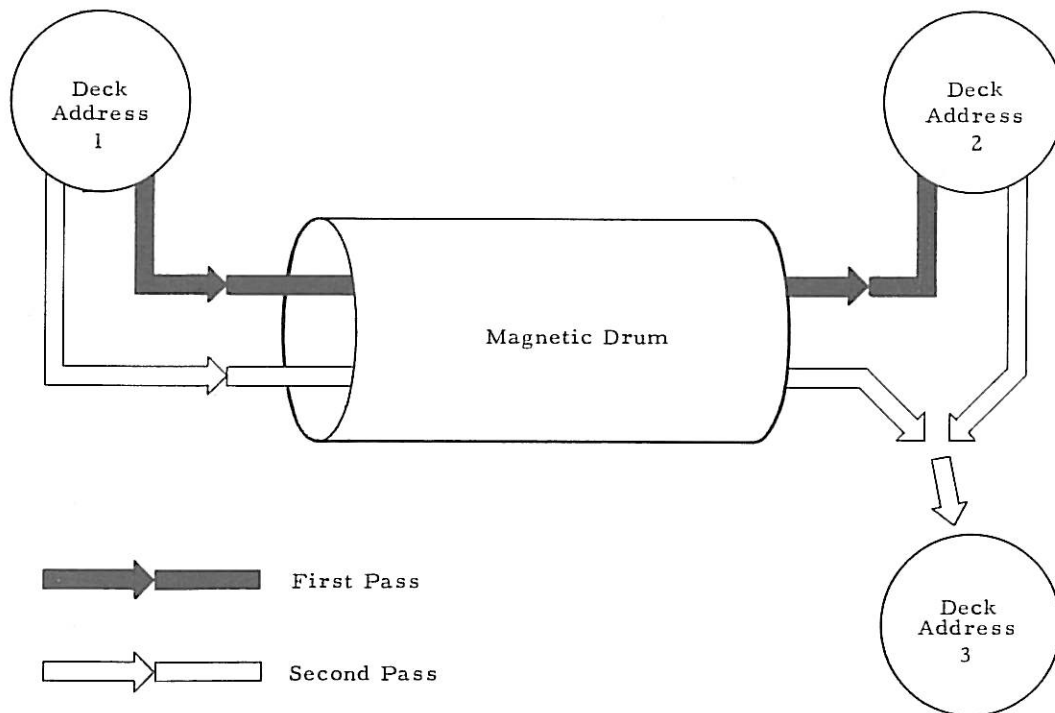


Figure 51: SORTING USING THREE TAPE DECKS AND THE MAGNETIC DRUM

**GENERATOR ROUTINES**

5.5

A generator is a routine that, when provided with parameters, creates instructions to form a routine which the programmer requires. Generators are available for creating input and output distribution routines and drum sorting routines.

A generator is provided with keys similar to those provided for a general program. For example, for an input distribution program, the keys specify the position into which the data fields are to be distributed. The generator differs from the general routine in that having extracted the required information from the keys, it does not obey the routine but punches it on cards. The punched routine can then be included in the main program as a subroutine.

It is likely that, in a given job, the keys for a general routine will be the same every time it is used. In this case, it is preferable to use a generator and include the generated program in the main program rather than the general routine. This will save the computer time used by a general routine to extract the same information from keys every time the program is run.

## **DIAGNOSTIC ROUTINES**

**5.6**

A diagnostic routine is a routine which has been designed to locate an error in programming; i.e. to determine where in a faulty program a program error has occurred.

The testing of a program by using diagnostic routines is accomplished in what may be considered as three basic steps:

- (a) Checking the validity of data punched in the program cards.
- (b) Checking storage of the program.
- (c) Testing the program itself, in sections.

I.C.T. diagnostic routines are available from the Subroutine Library to facilitate the testing of programs as outlined above.

### **Validity Check**

**5.6.1**

With the aid of the I.C.T. Validity Check routine, a validity check may be made on the program pack to ensure that all the instructions in a program conform to the correct layout required for the machine specification; any instructions which are in error (i.e. they do not conform with the correct format) are printed out. Enough information is printed out to enable such errors to be located and corrected. This routine ensures that if errors arise in a program, it is not due to incorrect program instructions (instructions not in the correct format) being read in.

### **Proving Storage**

**5.6.2**

A diagnostic routine, called the I.A.S. and Drum Print-out routine is available. This routine can be used for obtaining a complete print-out of the program as it is stored on the relevant area of the drum. This diagnostic routine will enable any incorrectly set relativizers to be detected.

### **Memory Dump**

**5.6.3**

The examination of a program during a test run may be achieved by using the I.C.T. Memory Dump routine. This diagnostic routine enables the contents of I.A.S. and certain drum channels to be automatically printed out at specified points in the program.



The routine is read into the machine with the program under test together with certain specifications. These specifications will state the following:-

- (a) The exact location of points in the program under test at which certain areas of storage are to be printed out.
- (b) Which groups of drum channels are to be printed out. One group of drum channels to be printed out is specified for each break-in point in the program under test.

Essentially, the Memory Dump routine may be considered as being in two sections - a preliminary routine and a print-out routine. Control is initially transferred to the preliminary routine (by arrangement of card packs) so that this routine, by using the specifications in (a) above, can substitute into the program under test, instructions to facilitate the entry into the print-out at the required points. The program under test is then run normally on test data as if the Memory Dump routine were not present. At the required points in the program under test, the substituted instructions cause entry to the print-out routine and all the contents of I.A.S. are printed out together with the specified drum channels.

The program under test then resumes control and continues to run in the normal manner.

The print-out will appear in blocks of two hundred words these being fifty lines of four words to a line, e.g.

0010	370012	420038	0060	450110	030027	0110
0011	640097	570011	0061	360170	560006	0111
0012	450001	030010	0062	420409	380021	0112

The above is an example of a segment of such a print-out.

It should be noted that the original state of the program under test is restored when the print-out routine is entered, thus the instructions referring to the Memory Dump routine are not printed and no space need be left in I.A.S. for the Memory Dump when writing the program under test.

The Memory Dump routine enables the programmer to determine which sections of program contain errors. To find the *exact* point in a program which is in error a Trace routine must be employed.

### Trace Routine

### 5.6.4

To pin-point an error in a section of a program it is of course possible to step through the section of program manually on the computer observing the contents of registers etc., but this involves the cost of much machine time. The same results may be obtained much more efficiently with the I.C.T. Trace routine. This diagnostic routine is a great aid to the programmer in testing programs. It produces:

- (a) A print-out of the instructions obeyed by the program under test in the order in which they were obeyed.

(b) A print-out of other relevant information about the effects produced by these instructions.

The Trace routine is used in much the same way as the Memory Dump routine, the main differences being in the specifications given with the Trace routine and the information printed out.

The specifications give the start and end points of the sections of the program under test which it is desired should be traced. When the program under test reaches the substituted instructions to enter the trace (substituted by a preliminary routine as in the Memory Dump) and the trace print-out is called in, the trace obeys the program under test one instruction at a time. After each instruction has been executed the following information is printed out:

- (a) The I.A.S. location of the instruction just obeyed.
- (b) The instruction just obeyed.
- (c) The contents of Register B.
- (d) The contents of the I.A.S. location specified in the instruction.
- (e) The time which the program would have taken to reach this point.

An example of a segment of a trace print-out shown below.

I.A.S. Location of Instruction obeyed	Designation	Function	Address	Contents of Register B after obeying current instruction	Contents of I.A.S. referred to after obeying current instruction	Time Accumulated so far (Microseconds)
0001	0	67	0019	00000000101	00000000001	375
	4	02	0001	00000000101		
0001	0	67	0019	00000000101	00000000000	412
	4	02	0001	00000000101		
0002	0	57	0011	00000000000	00000000000	453
	0	42	0001	00000000000	00000000000	474
0003	0	37	0002	570011420001	570011420001	507
	0	36	0003	770013760003	770013760003	528
0004	0	62	0003	540027520006	770013760003	561
	0	40	0004	540027520006	00000000000	582
0005	4	00	0004	540027520006		
0004	0	00	0000	540027520006	00000000000	606
	0	00	0000	540027520006	00000000000	618
0005	4	00	0004	540027520006		
0004	0	00	0000	540027520006	00000000000	642
	0	00	0000	540027520006	00000000000	654

The information displayed in the print-out should allow program errors to be easily located.

Other trace routines with special characteristics are available, which may be used in preference to the full I.C.T. Trace routine.

If full details provided by the I.C.T. Trace routine are not required, the Indicator Trace routine might be used. This routine operates under the control of a manual indicator and prints either as the full trace or else suppresses printing on all except jump instructions (i.e. successful indicator tests).

The I.C.T. Trace routine slows down the rate at which instructions in the program under test are executed. Thus the I.C.T. Trace routine cannot be run on sections of a program which contain input, output or magnetic-tape operations. If the program uses standard proven library subroutines, there is no need to waste time tracing them.

The Trace routine can therefore be obtained in such a form that subroutines which are not required to be traced can be automatically avoided.

### **Program Updating Routine**

**5.6.5**

Having traced an error in the testing of a program, it then becomes necessary to rectify the programming error.

The amendment of the program by means of the insertion or deletion of words, besides creating the problem of re-allocation of storage, causes other difficulties. Jumping over an unwanted portion of program is wasteful of storage and untidy. Jumping out to previously unused storage locations in order to effect insertions can cause great confusion as the number and complexity of amendments increases. The actual deletion of a section of program or insertion of a group of words, besides making a rewriting of the particular block of program necessary, invalidates addresses elsewhere in the program. These addresses could be located and converted but the labour involved could be immense.

The I.C.T. Program Updating routine is designed to reduce this labour. If amendments have been made by inserting or deleting words throughout the program card pack and resetting the relativizer settings to the re-allocated storage locations, then the program pack can be fed as data cards to the Program Updating routine.

The Program Updating routine pack, together with specifications indicating the size and location of deletions and insertions which have already been made to the program under test is read into the machine. The program under test is then placed in the card read hopper and is read and processed by the Program Updating routine. Depending on the setting of manual indicators, the routine produces a printed and/or punched version of the program under test in which all addresses made invalid by the insertions or deletions have been corrected. The printed output is in the format of a normal program giving the absolute-address form of the program alongside the relative-address form.

### **UTILITY ROUTINES**

**5.7**

Routines are available that do not belong to any particular category but have such general applications that they may be included in almost any type of program. These Utility routines cover such programs as division routines, sorting routines and routines for detecting parity errors and taking appropriate action.

## Division Routines

5.7.1

Division is not built into the hardware of the computer but is accomplished in subroutines by repeated subtraction. Routines are available which cover both decimal and sterling division for positive and negative numbers.

## Parity Error Routines

5.7.2

Various routines are available for testing the drum and I.A.S. parity indicators after a drum transfer has been obeyed. The programming implications of parity errors are described in 4.8. It should be noted that:

- (a) An I.A.S. parity check is made when data are transferred from I.A.S. Hence an I.A.S. parity error can only be detected when the transfer is from I.A.S. to the drum. If the I.A.S. parity indicator is found to be set then the computer is brought to a stop (11 1006 displayed in CR3). If this stop is encountered then a return must always be made to the previous restart point.
- (b) A drum parity check is made when data are transferred from the drum. Hence a drum parity error can only be detected when the transfer is from the drum to I.A.S. If the drum parity indicator is found to be set then the computer is brought to a stop with 11 1007 displayed in CR3. When the Start button is pressed a further attempt is made to obey the drum transfer. If there is a persistent failure, then a return must be made to the previous restart point.
- (c) A parity error routine (or a programmed test of indicator 07) should be used for *every* transfer from the drum. If a parity failure is detected on a transfer from the drum which is not followed by a test of indicator 07 then the program will detect this error the next time indicator 07 is tested. Indicator 07 will be unset by testing and a repeat of the drum transfer preceding the indicator test (which is not the transfer in which the parity check failed) will apparently correct the error.

For some of the library routines the drum transfer is given as a parameter and is obeyed in the subroutine. It is recommended that these should be used since they make it possible for a drum transfer instruction to overwrite its own storage location in I.A.S. If a drum parity error is detected then, as the drum transfer is retained in the subroutine, another attempt can be made to effect the transfer.

## Zero Suppression Routines

5.7.3

During output distribution, particularly prior to printing, it is very often necessary to distinguish between significant and non-significant digits in a data field. For example, if a quantity is allocated six positions on the printed sheet, and a particular value is only a four-digit number, then it is likely that it will be required to suppress the two non-significant zeros (leaving spaces) and print only the four-digit number. This process is known as zero suppression. It consists of

giving a zone component of zero to non-significant zeros and a zone component of one to significant digits. Standard zero suppression routines are available for both decimal and sterling fields.

### **Punching Program into Fast-read Cards**

**5.7.4**

Fast-read program cards (preceded by a control word of designation F) enable five 12-column words per program card to be read in under Initial Orders. Each word is punched exactly as it appears within the computer i.e. all addresses are absolute, negative numbers are in complementary form etc. Thus, by using fast-read cards, Initial Orders merely read and store the data read in the form in which they are punched. A full description of Initial Orders and fast-read cards will be found in the Initial Orders Manual.

Fast-read cards may be punched in the correct format from data held on the magnetic drum, i.e. a proved or compiled program may be punched into fast-read cards from the drum for later use as input.

Several subroutines are available from the Subroutine Library which enable fast-read cards to be punched in the required format from the absolute information as it is held on the drum.

### **Sorting Routines**

**5.7.5**

The arrangement of data records into ascending numeric sequence, is an essential part of computer usage. The sequence is arranged according to a key held in a given position within each record. Sorting methods are available which enable source items to be examined in their initial sequence and thence to be rearranged in a required sorted sequence. Many subroutines are available from the Subroutine Library which cater for most aspects of sorting of fixed-length or variable-length records. Data held on the drum or in I.A.S. may be sorted and several sorting methods are available. The present subroutines available provide three basic methods of sorting data in I.A.S. and on the drum; these methods are:

Merging Sort	(drum or I.A.S.)
Extraction Sort	(I.A.S.)
Exchanging Sort	(I.A.S.)

Subroutines are also available which facilitate the insertion of a record into its appropriate place within a string of variable-length or fixed-length records.

#### *Merging Sort*

Sorting by merging consists of taking two or more ordered groups of data (called strings) and collating these into one ordered group or string. When data are presented in random order, straight merging may be employed. That is, the initial strings are considered as being composed of one item only. The routines available for I.A.S. sorting are all able to employ the straight merge because of the fast internal speed. The method is as follows: The first two keys are compared and their corresponding items placed in their correct order in one receiving area. The

next two items are merged in a similar manner and the result placed in a second receiving area. Each new string subsequently created from a pair of original items is then placed alternately into one of the two receiving areas, after the strings already contained in the area. When all the data have been examined in this way the first pass is complete and two streams of data now exist, each consisting of ordered strings of pairs of the original items. The two receiving areas are now used as source areas and the process repeated using two other receiving areas. This time the pairs are merged into groups of four. The data are thus passed back and forth, the areas alternating as receiving and source areas until one completely ordered string emerges. The sort is then complete. This method of sorting by merging using two input areas and two output areas is sometimes called a two-on-two sort.

EXAMPLE OF A 2 ON 2 STRAIGHT MERGE

Initial Order	1st Pass		2nd Pass	
	Area 1	Area 2	Area 3	Area 4
8	4	1	1	2
4	<u>8</u>	<u>7</u>	4	3
7	<u>2</u>	<u>3</u>	7	5
1	<u>9</u>	<u>5</u>	<u>8</u>	<u>9</u>
9	<u>6</u>		<u>6</u>	
2				
3				
5				
6				
	3rd Pass		4th Pass	
	Area 1	Area 2	Area 3	Area 4
	1	<u>6</u>	1	
	2		2	
	3		3	
	4		4	
	5		5	
	7		6	
	8		7	
	<u>9</u>		8	
			<u>9</u>	

In practice, areas 1 and 2 are combined into a single area the length of the original data area, and areas 3 and 4 are combined to overwrite the original data area.

For drum sorting, records are stored in blocks. These blocks are of a given decade length, usually 20 decades to make channel transfers possible although 10 decades are often used on 400 I.A.S. machines.

The first part of a drum sort entails merging the records within each block using an I.A.S. merging sort and then storing these blocks on the drum.

A drum merging sort is then used to sort these blocks into their own area on the drum in a sequence defined either by an index in I.A.S. or by a word in each block containing the address of the next block. To do this pairs of blocks are brought into I.A.S., merged together and returned to the drum, giving strings of pairs of ordered blocks. These pairs of blocks are then merged to give strings of four blocks in order. This process is repeated until there is only one string. A Drum Reshuffle routine uses the index to put the blocks thus ordered into a completely sorted file on the drum.

### **Extraction Sort**

An extraction sort entails the sorting of records according to a keyword which is the first word of a record. Routines are available for an extraction sort on both variable-length and fixed-length records.

Basically, an extraction sort involves the extraction of the keywords and their addresses from the original data for an index which is in effect a set of two word records. These key records are sorted by a merging sort, and the addresses are then used to transfer the original records from their data areas to the output area in ascending keyword order. One advantage of using this method is that the I.C.T. subroutines allow the output area to be used as a working area to the index merging sort so that no extra storage area is required other than the original data and output area storage.

### **Exchanging Sort**

An exchanging sort entails the sorting of records (according to a keyword which is the first word of a record) within the original data storage area. The method given below is used in the standard I.C.T routine.

The first key is compared with one half-way through the list of data. An exchange takes place if necessary. The second two keys of each half are compared and so on until all the data has been covered. A second pass over the data is made, the distance between the two keys being compared now being half what it was before. Passes continue, the distance between keys being halved at each successive pass, until finally each key is compared with the next. The sort is then complete. If at any time during a pass an exchange takes place, the lesser item is further compared with earlier items and shifted to its correct position. If the calculation of the distance between keys to be compared does not produce an integer, the next lowest integer is taken as the value.

## **COMMERCIAL ROUTINES**

**5.8**

The I.C.T. Subroutine Library contains a number of subroutines which cover some of the essential commercial programs that are commonly required, e.g. P.A.Y.E., sterling and decimal conversions, Graduated Pensions etc.

### **P.A.Y.E.**

**5.8.1**

Routines are available for both monthly and weekly P.A.Y.E. calculations and these routines are constantly amended to conform to Budget changes.

Computation of weekly or monthly P.A.Y.E. contributions is made on data usually required in P.A.Y.E. accountancy, i.e.

Gross Pay brought forward  
Gross Pay for period  
Tax brought forward  
P.A.Y.E. Code Number  
Week (Month) Number  
Tax Basis Code  
Fixed Amount of Tax.

Given the data above, it is possible for the subroutines to compute:

Gross Pay carried forward  
Tax carried forward  
Tax for period (positive or negative, negative being an indication of a refund of tax).

### **Graduated Pension Contributions**

**5.8.2**

I.C.T. subroutines are available which compute contributions for the Graduated Pension Scheme for a pay period of one week, multiples of one week or a calendar month. Once the gross pay for the pay period has been determined (i.e. gross pay for one month or N weeks etc.) the Graduated Pension Contributions subroutine is entered to calculate the appropriate contributions.

### **Sterling and Decimal Conversions**

**5.8.3**

A number of subroutines are available which enable such conversions as:

Sterling to £ and decimals of a £  
Sterling to pence  
Pence to sterling  
Sterling amounts (in numeric form) to English (in alphabetic form).

## **MATHEMATICAL AND STATISTICAL ROUTINES**

**5.9**

### **Floating-point Arithmetic**

**5.9.1**

In arithmetic calculation it is often convenient to represent numbers by two factors, a decimal quantity and some power of a chosen radix. For example, in denary arithmetic the radix is 10. The number 83256817000000 can be represented by  $0.83256817 \times 10^{14}$ . Similarly for very small numbers:  $0.0000000078325763 = 0.78325763 \times 10^{-8}$ . This is known as floating-point representation and can be very useful in overcoming the limiting size of registers. Packages are available for performing the normal arithmetic operations of addition, subtraction, multiplication and division with numbers which range in magnitude from  $1 \times 10^{-51}$  to  $1 \times 10^{+49}$  retaining 9 significant digits throughout the range.



**Matrix Arithmetic**

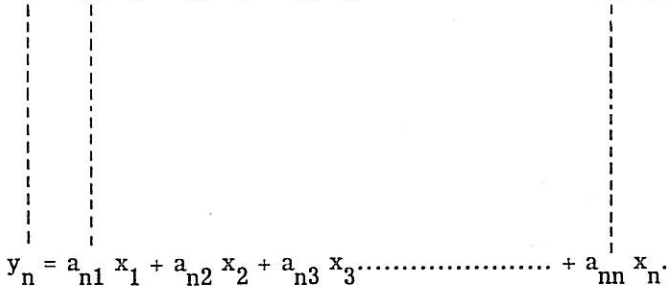
**5.9.2**

Many calculations, especially in engineering, are concerned with linear relationships of variables (e.g. Hooke's Law), and problems often arise in the form of several linear equations relating one set of variables  $y_1, y_2, y_3, \dots, y_n$  to another set  $x_1, x_2, x_3, \dots, x_n$ .

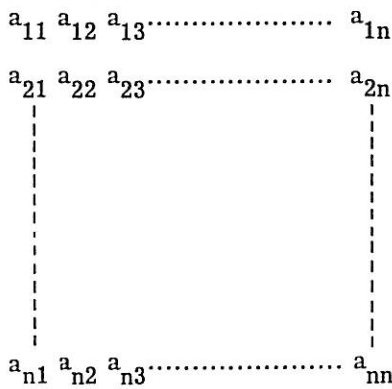
This gives n simultaneous equations which written in full are:

$$y_1 = a_{11} x_1 + a_{12} x_2 + a_{13} x_3 \dots \dots \dots + a_{1n} x_n.$$

$$y_2 = a_{21} x_1 + a_{22} x_2 + a_{23} x_3 \dots \dots \dots + a_{2n} x_n.$$



where  $a_{n1}$  is the coefficient of  $x_1$  in the expression for  $y_n$ . The convenient abbreviated form of this set of coefficients is called a matrix and is written as follows:



The equations are denoted by the single expression:

$$y = Ax$$

where A is the matrix above of n rows and n columns. A set of subroutines is available for carrying out the common operations of linear algebra, i.e. addition, subtraction, multiplication, transposition, inversion, calculation of eigen roots and eigen vectors, and the solution of simultaneous equations. Facilities are also provided for reading in and printing out a matrix.

**Double - length Arithmetic**

**5.9.3**

For use in those cases where the numbers used can vary moderately in size, and in which a high degree of precision is required, a double-length arithmetic package is available which gives facilities for carrying out the normal arithmetic operations with numbers which are 22 digits long instead of the usual 11 digits.

### Special Mathematical Routines

5.9.4

Linear Programming is used to optimize a linear function of a set of variables subjected to linear constraints, by means of the Simplex Method. Facilities are provided for taking into account the imposition of upper and lower bounds on the variables.

Statistical Programs are available for the analysis of sets of experimental observations. These have the ability to draw inferences regarding the representation which most closely fits the observations.

Engineering Programs have been developed to evaluate the stresses and deflections induced in common structural forms by the loads imposed upon them.

A set of subroutines are available for the evaluation of commonly occurring Mathematical and Trigonometrical Functions such as exponential, sine, tangent, etc.

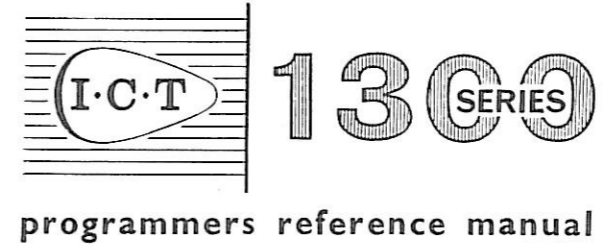
### Manchester AutoCode

5.9.5

The Manchester AutoCode provides facilities for incorporating into a program MAC routines which have been written independently of that program. Certain general purpose MAC routines are available from the I.C.T. Subroutine Library. The mathematical and statistical library routines therefore fall into three groups:

- (a) Complete routines for which the user need only supply data. These may be written either in MAC or in machine code.
- (b) Subroutines written in machine code which may be incorporated into machine-coded programs.
- (c) General purpose routines written in MAC which may be incorporated into MAC programs.





REFERENCE TABLES  
AND GLOSSARY

Contents

Table No.	Subject	Page
1	SUMMARY OF CHARACTERISTICS OF 1300-SERIES COMPUTER SYSTEM ... ..	1
2	SUMMARY OF FUNCTIONS AND FUNCTION TIMINGS ... ..	3
2A	Summary of Functions and Timings ... ..	3
2B	Drum Transfer Times ... ..	5
2C	Multiplication Times ... ..	5
3	1300-SERIES COMPUTER FUNCTION CHART ...	7
4	PERIPHERAL AND MAGNETIC DRUM INSTRUCTIONS ... ..	9
4A	Magnetic Drum and Block Transfer Instructions..	9
4B	Peripheral Equipment Instructions ... ..	10
	Card Reader	
	Line Printer	
	Card Punch	
	Paper-tape Reader	
	Interrogating Typewriter	
	Paper-tape Punch	
4C	Magnetic-tape Instructions ... ..	11

Table No.	Subject	Page
5	SUMMARY OF INDICATORS ... ..	13
5A	Central Processor Indicators ... ..	13
5B	Peripheral Equipment Indicators ... ..	14
5C	One-inch (90 kc/s) and Half-inch (22½ kc/s) Magnetic-tape Indicators ... ..	15
5D	Quarter-inch (16 kc/s) Magnetic-tape Indicators	16
6	STANDARD CODE FOR CARD READER AND CARD PUNCH ... ..	17
7	CODE FOR LINE PRINTER ... ..	19
8	RECOMMENDED PAPER-TAPE CODES ... ..	21
8A	Recommended Paper-tape 6-Track Code ... ..	21
8B	Recommended Paper-tape 7-Track Code ... ..	22
8C	Recommended Paper-tape 8-Track Code ... ..	23
9	CODE FOR INTERROGATING TYPEWRITER ...	25
10	PERIPHERAL EQUIPMENT TIMINGS .. ..	27
10A	Card Reader Timings ... ..	27
10B	Card Punch Timings ... ..	28
10C	Line Printer Timings ... ..	29
10D	Paper-tape Reader Timings ... ..	29
10E	Paper-tape Punch Timings ... ..	29
10F	Interrogating Typewriter Timings ... ..	29
11	TIMINGS AND STATISTICS FOR ONE-INCH (90 kc/s) AND HALF-INCH (22½ kc/s) MAGNETIC-TAPE SYSTEMS ... ..	31
12	TIMINGS AND STATISTICS FOR QUARTER-INCH (16 kc/s) MAGNETIC-TAPE SYSTEMS ... ..	33
13	PROGRAM FLOWCHART SYMBOLS ... ..	35
14	SYSTEMS FLOWCHART SYMBOLS ... ..	37
15	OPERATORS' AND PROGRAMMERS' DISPLAY AND SWITCH PANELS ... ..	39
15A	Computer Console ... ..	39
15B	Peripheral Equipment ... ..	40
	(i) Card Reader	
	(ii) Card Punch	
	(iii) Line Printer	
15C	Magnetic-tape Units ... ..	41
	(i) Quarter-inch (16 kc/s) Magnetic-tape Unit	
	(ii) Half-inch (22½ kc/s) Magnetic-tape Unit	
	(iii) One-inch (90 kc/s) Magnetic-tape Unit	
	GLOSSARY OF TERMS USED IN THE 1300-SERIES PROGRAMMERS REFERENCE MANUAL ... ..	43

**Table I**  
**Summary of Characteristics of the I300-series Computer System**

<b>INPUT/OUTPUT</b>					
<b>CARD READER</b>	<b>CARD PUNCH</b>	<b>LINE PRINTER</b>	<b>PAPER-TAPE READER</b>	<b>PAPER-TAPE PUNCH</b>	<b>INTERROGATING TYPEWRITER</b>
80 Column Cards 600 or 300 cards a minute	80 Column Cards 100 cards a minute	Line of 80 or 120 Characters 600 or 300 Lines a minute	5-, 6-, 7- or 8-Track Tape 1,000 Characters a second	5-, 6-, 7- or 8-Track Tape 300 Characters a second	Line of up to 120 Characters 10 Characters a second
<b>MAGNETIC TAPE</b>					
Up to 8 magnetic-tape units can be used. Three alternative systems are available:-					
i) 1" tape operating at a rate of 90,000 decimal digits a second. Maximum Reel length 3,600 feet.					
ii) 1/2" tape operating at a rate of 22,500 decimal digits a second. Maximum Reel length 3,600 feet.					
iii) 1/4" tape operating at an average rate of 16,500 decimal digits a second. Maximum Reel length 1,800 feet.					
<b>COMPUTER</b>					
<b>STORAGE</b>					
<b>IMMEDIATE ACCESS STORE (I.A.S.)</b>			<b>MAGNETIC DRUM STORE</b>		
Magnetic Core Storage ranging from 400 to 2,000 word capacity is available in units of 400 word capacity. Alternatively may have 4,000 word capacity.			1 to 8 drums are available. Capacity of each drum is 12,000 words held on 60 channels, each channel having a capacity of 200 words. Alternatively, a machine may be fitted with a single drum of 3,000 or 6,000 words. Each drum has an additional 2 channels of reserved storage. Speed 5,240 revolutions a minute. Average access time less than 6 ms. Channel transfer to or from I.A.S. within 12 ms including access.		
<b>ARITHMETIC UNIT</b>					
Comprises Mill and three Registers. Functions available include: add, subtract and multiply in decimal or sterling with variable f. s. d. positions; transfer between registers and to and from storage; logical operations; shifts; etc.					
<b>INDICATORS</b>					
Provided to record the result of past events, manual switch setting and states of input/output units. A jump instruction, causing a jump out of sequence in the order of instruction obeyed, is provided by a test of an indicator that is set.					
<b>GENERAL</b>					
Pulse Rate of 1 Mc/s (Megacycle). Binary Coded Representation of decimal or sterling quantities. Digits transferred in series, the binary-codings of that digit being recorded on 4 lines. Word length - 12 digits including sign. Core store has 2 parity bits with each word. Drum store has one check digit of 4 bits with each word. Instructions affecting arithmetic unit and I.A.S. only, except multiplication, take between 17 and 34 μs to be obeyed. Instruction length, normally 6 digits. Transfer of instruction pair to control register takes 12 μs. Multiplication takes approximately 175 μs per multiplier digit on average.					

**Table 2**  
**Summary of Functions and Function Timings**

Func.	Description of Function	Address	Inds. affected	Register A	Time micro-seconds
00	Do nothing				12
11	Stop			Original	-
21	Set Decimal Point Register	No. Dec.			12
22	Set Sterling Position Register	10/- pos.			12
30	Set Row Binary Register from I.A.S.				
31	Create Row Binary 1 (1 Stream) in Register B				
32	Create Row Binary 2 (2 Stream) in Register B				
33	Create Row Binary 3 (4 Stream) in Register B				
34	Create Row Binary 4 (8 Stream) in Register B				
		I.A.S.		I.A.S. Word	21
35	Logical AND I.A.S. to Register B		Mill		
36	Logical OR I.A.S. to Register B and I.A.S.			Result	
37	Transfer from I.A.S. to Register B			I.A.S. Word	
38	Input/Output Control				
39	Magnetic-tape Control	See Table 4		Original	12, or 24 if double-length instruction
40	Write Zero to I.A.S.			0	
41	Transfer Register A to I.A.S.			Original	
42	Transfer Register B to I.A.S.				
43	Transfer Register C to I.A.S.			Result	
		I.A.S.			21
44	Transfer Register C to Register B			Original	17
45	Block Transfer I.A.S. to I.A.S.	I.A.S.		X X X	See Table 4
54	Circulate Left in Register B			Original	17
55	Left Shift Register B, entering zeros		No.	0	34
			of		
56	Right Shift Register B, propagating sign		posns.	Original	17
57	Right Shift Register B, entering zeros				

Table 2A: SUMMARY OF FUNCTIONS AND TIMINGS

Table 2A: continued

Func.	Description of Function	Address	Inds. affected	Register A	Time micro-seconds
60	Clear Add I.A.S. to Register B	Decimal Arithmetic I.A.S.	Mill and Over-flow	I.A.S. Word	21
61	Clear Subtract I.A.S. from Register B				
62	Add I.A.S. to Register B				
63	Subtract I.A.S. from Register B				
64	Add Register B to I.A.S.			Result	25
65	Subtract Register B from I.A.S.				
66	Add 1 to I.A.S.				
67	Subtract 1 from I.A.S.			26	
68	Compare I.A.S. with Register B			X X X	See Table 2C
69	Multiply I.A.S. (decimal) by Register B (decimal) into Register B and Register C				
70	As functions 60-68, with Arithmetic in Sterling	I.A.S.	Mill and Over-flow	I.A.S. Word	21
71					
72					
73					
74				Result	25
75					
76					
77	26				
78					
79	Multiply I.A.S. (Sterling) by Register B (decimal) into Register B and Register C	X X X	See Table 2C		
80	Decade transfer to drum	See Table 4		X X X	See Table 2B
81	Decade transfer from drum				
82	Channel transfer to drum				
83	Channel transfer from drum				
84	Decade transfer to reserved store				
85	Decade transfer from reserved store				
86	Channel transfer to reserved store				
87	Channel transfer from reserved store				



Decade

Average :  $5.7 + 0.57n$  ms

Maximum :  $11.4 + 0.57n$  ms

Where  $n$  = number of decades

If change of drum or drum section:

Average :  $11.7 + 0.57n$  ms

Maximum :  $23.4 + 0.57n$  ms

Channel

Average : 11.7 ms

Maximum : 12.0 ms

If different drum of drum section from last transfer:

Average : 17.7 ms

Maximum : 24 ms

Table 2B: DRUM TRANSFER TIMES

Minimum:  $44(n + 1 + m)$   $\mu$ s

Maximum:  $44(6n + 2 + m)$   $\mu$ s

Average:  $22(7n + 3 + 2m)$   $\mu$ s

Where  $n$  = number of digits in multiplier (excluding non-significant zeros) and  $m = P - n$  if positive, and 0 otherwise; where  $P$  is the number entered in the Decimal Point Register.

Table 2C: MULTIPLICATION TIMES

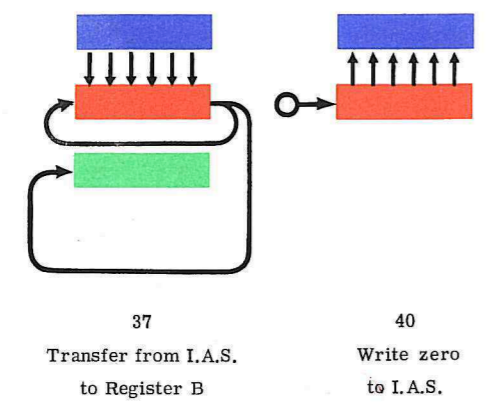
### 1300-SERIES COMPUTER FUNCTION CHART

3200/37 (2.64)

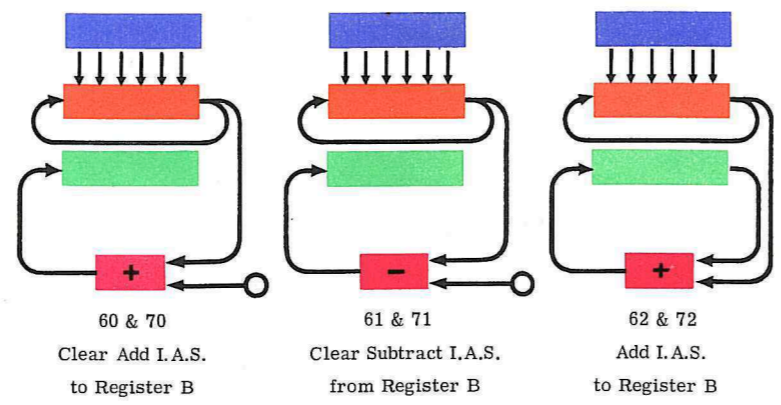
**COLOUR CODE**

- I.A.S.
- REG. A
- REG. B
- REG. C
- MILL

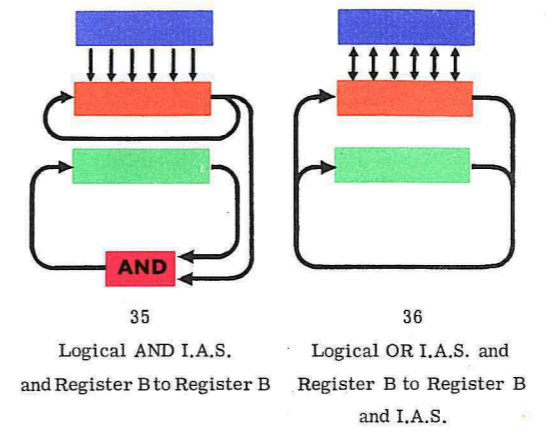
**TRANSFER FUNCTIONS**



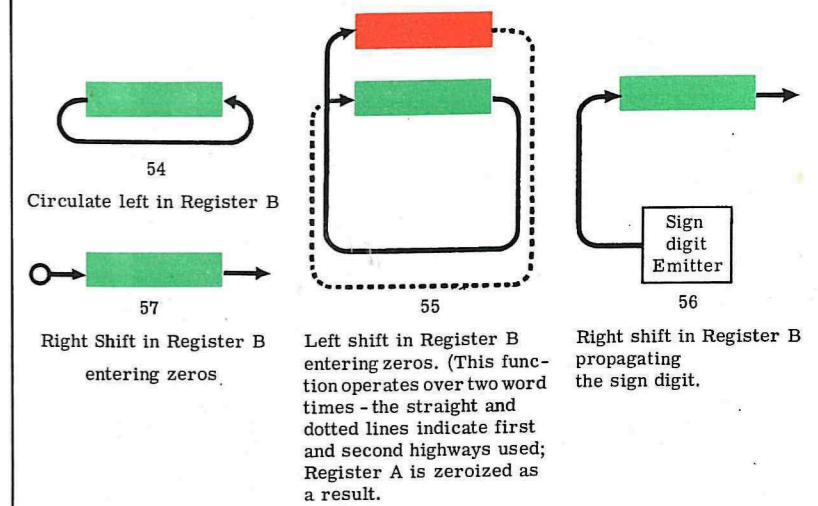
**ARITHMETIC FUNCTIONS**



**LOGICAL FUNCTIONS**



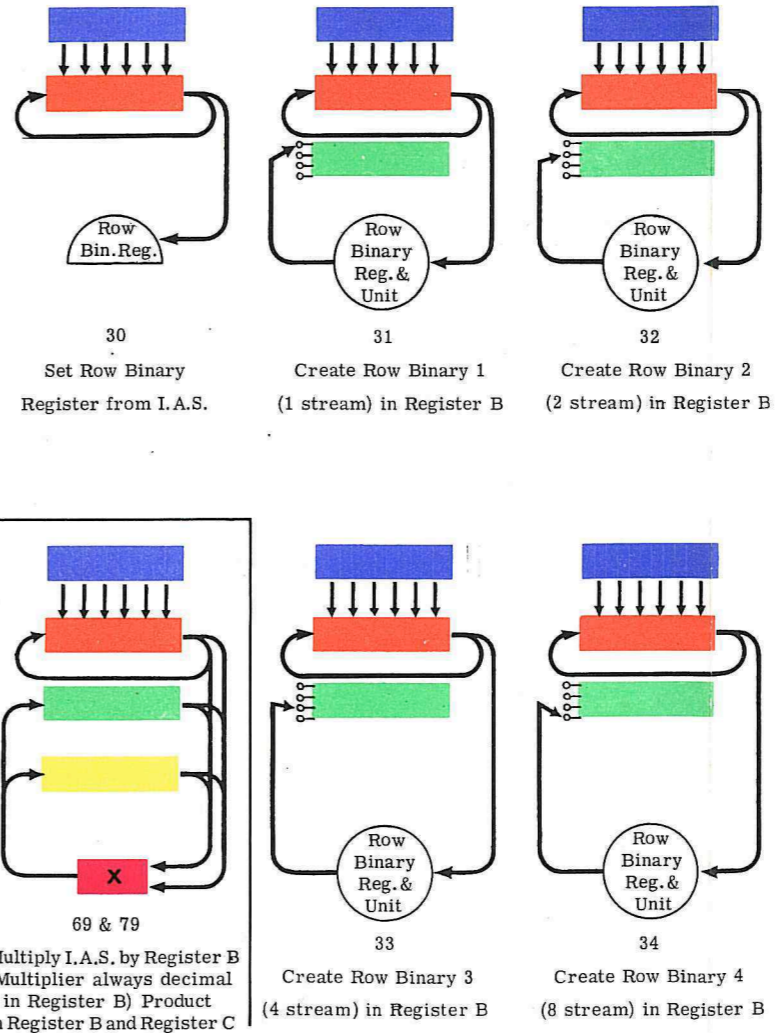
**SHIFT FUNCTIONS**



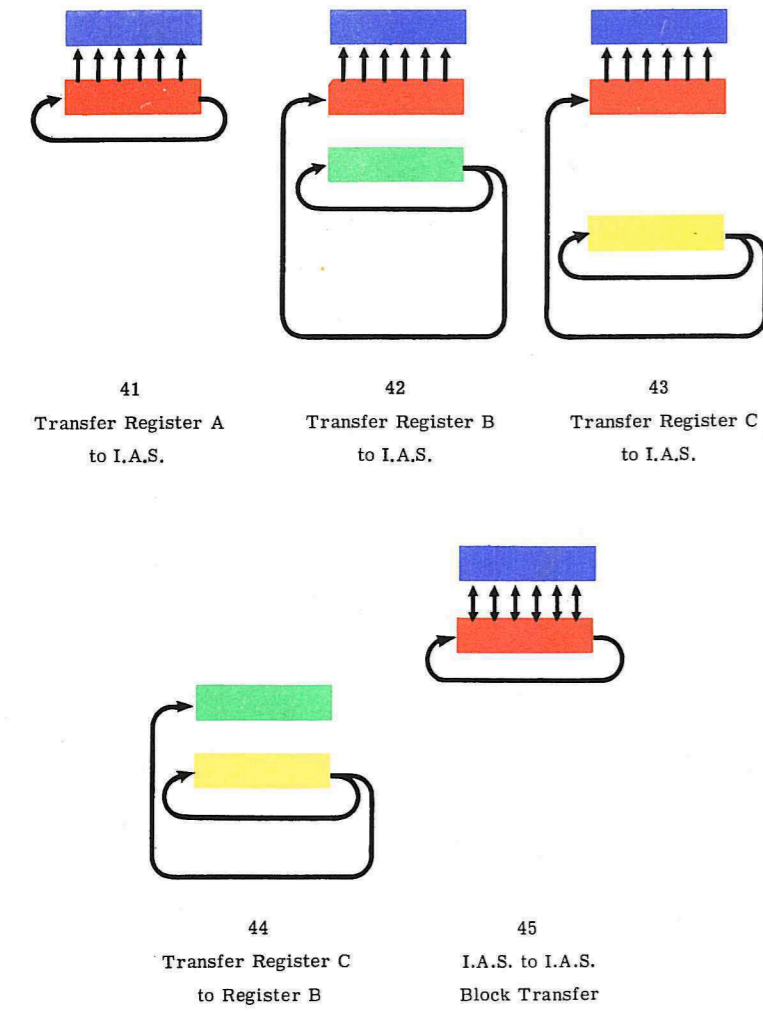
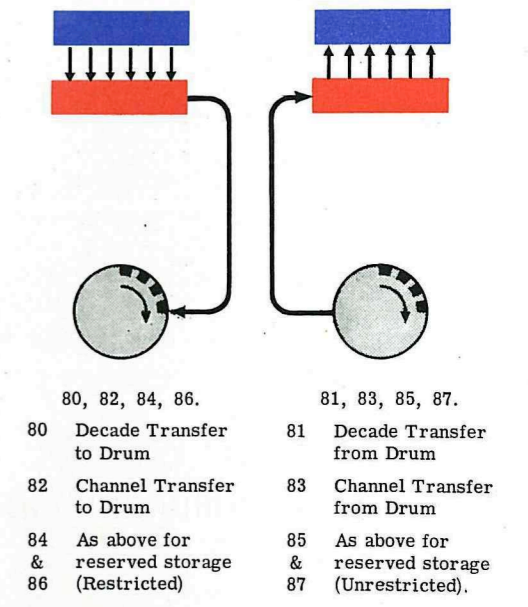
Left shifting is achieved in the 1300-series computers by right shifting

The sixty range of functions is for decimal operations and the seventy range is for sterling operations in conjunction with the setting of the Sterling Position Register.

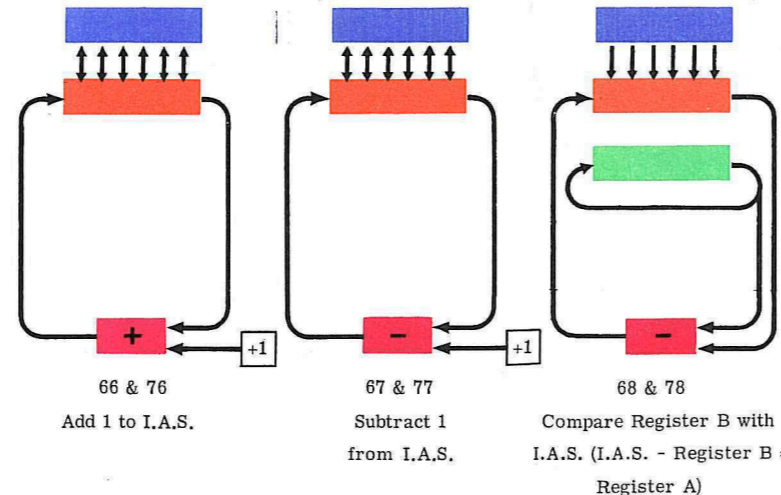
**ROW BINARY FUNCTIONS**



**DRUM FUNCTIONS**



The appropriate sign indicator is set on all functions where figures pass through the Mill. Similarly the overflow indicator may be set by the same functions with the exception of the 35 Logical AND instruction.



69 & 79 Multiply I.A.S. by Register B (Multiplier always decimal - in Register B) Product in Register B and Register C

**OTHER FUNCTIONS**

Function No.	Description
00	Do Nothing
11	Stop
21	Set Decimal Point Register
22	Set Sterling Position Register
38	Input Output Control
39	Magnetic Tape Control

**Table 4**  
**Peripheral and Magnetic Drum Instructions**

F	A	F	A	DESCRIPTIONS	
45	First I.A.S. Address Source	N	First I.A.S. Address Destination	Transfer N words from source to destination Timing = 26N us	BLOCK TRANSFERS
80	First I.A.S. Address	N	First Drum Decade Address	Transfer N decades from I.A.S. to drum	MAGNETIC DRUM N = 1 to 20
81				Transfer N decades from drum to I.A.S.	
82		20		Transfer channel from I.A.S. to drum	
83				Transfer channel from drum to I.A.S.	
84	First I.A.S. Address	N	First Drum Decade Address	Transfer N decades from I.A.S. to reserved store *	MAGNETIC DRUM RESERVED STORAGE N = 1 to 20
85				Transfer N decades from reserved store to I.A.S.	
86		20		Transfer channel from I.A.S. to reserved store *	
87				Transfer channel from reserved store to I.A.S.	

*Available only under engineers' control

**Table 4A: MAGNETIC DRUM AND BLOCK TRANSFER INSTRUCTIONS**

F	A	DESCRIPTIONS	
38	0002	Call Card Feed	CARD READER
38	0007	Reject Card	
38	0013	Set Row Binary Register from Print Counter	LINE PRINTER (PRINT)
38	0014	Print Left-hand Bank	
38	0015	Print Centre Bank	
38	0016	Print Right-hand Bank	
38	0020	Lift all Sprags	LINE PRINTER (SPACE)
38	0021	Drop Sprag 1, lift all others	
38	0022	Drop Sprag 2, lift all others	
38	0023	Drop Sprag 3, lift all others	
38	0024	Drop Sprag 4, lift all others	
38	0025	Drop Sprag 5, lift all others	
38	0026	Drop Sprag 6, lift all others	
38	0042	Call Punch Feed	CARD PUNCH
38	0043	Punch Left-hand Bank	
38	0044	Punch Right-hand Bank	
38	0045	Check Read Left-hand Bank	
38	0046	Check Read Right-hand Bank	
38	0047	Off-set Card	
38	0050	Read One Code	PAPER-TAPE READER
38	0051	Select Reader No. 1	
38	0052	Select Reader No. 2	
38	0070	Select TYPE IN Mode	INTERROGATING TYPEWRITER
38	0071	Select TYPE OUT Mode	
38	0072	Type one character to or from Register B according to mode	
38	0076	Punch one code, check one code	PAPER-TAPE PUNCH

Table 4B: PERIPHERAL EQUIPMENT INSTRUCTIONS

F	A	F	A	DESCRIPTION	
39	001X	00	First I.A.S. Address	Write one Block	X = deck address 1 to 8
39	002X			Read one block (read on Track A for $\frac{1}{4}$ " tape)	
39	003X	} Single -length instructions	Backspace one block		
39	004X		Cancel one block (one section on $\frac{1}{4}$ " tape)		
39	005X		Rewind to start of tape		
39	006X		Unload - tape completely rewound on spool		
39	007X	00	First I.A.S. Address	Read one block on track B ( $\frac{1}{4}$ " tape only)	

Table 4C: MAGNETIC-TAPE INSTRUCTIONS

**Table 5**  
**Summary of Indicators**

TYPE	IND No.	SET BY	UNSET BY
	00	PERMANENTLY SET	
MILL	01	= 0	≠ 0
	02	Last Number > 0 - i.e. 4 in through Mill sign digit	Last Number ≠ 0 through Mill
	03	< 0 - i.e. 5 to 15 in sign digit	≠ 0
OVERFLOW	04	Sign digit ≠ 0 or 9, or last number through mill = 900000000000	Program when tested
ERROR	06	I.A.S. Parity Error	Program when tested
	07	Drum Parity Error	
PROGRAM	10 to 19	Instruction (Designation 8)	Instruction (Designation 9)
MANUAL	20 to 29	MANUAL CONTROL ON CONSOLE	

**Table 5A: CENTRAL PROCESSOR INDICATORS**

TYPE	IND No.	SET BY	UNSET BY
CARD READER	35	Card Reader Ready	Instruction 380002 and Card Reader Interlock
	36	6 Columns Read	Program when tested or automatically if not tested
	37	6 Columns Missed	Program when tested or between card cycles
	38	Mischeck	
LINE PRINTER	42	Printer Ready	Printer Interlock
	43	Print Index Point Time	Program when tested or automatically if not tested
	44	Print Character Time	Program when tested
	45	Line Space Time	Program when tested or automatically if not tested
	47	Paper Trolley Empty	New supply of paper inserted
INTERR-OGATING TYPE-WRITER	49	Print Counter Error	Program when tested
	50	Paper Supply Low	New supply of paper inserted
	51	Typewriter Ready	Instruction to Type In or Out given
	52	Request Type-in	Program test
CARD PUNCH	53	Carriage at End	Carriage leaving left-hand stop
	59	Typewriter Mechanical Failure	Program test
	54	Punch Ready	Punch Interlock & Instruction 380042
	55	Punch Index Point Time	Program when tested or automatically if not tested
	56	Check Index Point Time	Program when tested or automatically if not tested
PAPER -TAPE READER	57	Punch Index Point Time missed	Program when tested
	58	Check Index Point Time missed	
PAPER -TAPE READER	60	Tape Reader Ready	Automatically if Reader not ready
	61	Parity Error	Program test
PAPER -TAPE PUNCH	65	Tape Supply Low	More than 20 feet of tape on spool
	66	Tape Punch Ready	automatically if Punch not ready
	67	Tape Punch Error	Program test

Table 5B: PERIPHERAL EQUIPMENT INDICATORS

INDICATOR No.	TITLE	SET BY	UNSET BY
70	Write Unit Ready	Write Unit not busy	Write or Cancel Instruction
71	Write Master	W74, W76, W77 becoming set	Unsetting of all three Indicators
72	Read Unit Ready	Read unit not busy	Read or Backspace Instruction
73	Read Master	R75, R76, R77 becoming set	Unsetting of all three Indicators
W74 74 R74	Write Any Errors Read Any Errors	Any Bit Errors Written Any Bit Errors Read	Write or Cancel Instruction Read Instruction
W75 75 R75	Write Multiple Errors Read Multiple Errors	Multiple Bit Errors Written Multiple Bit Errors Read	Write or Cancel Instruction Read Instruction
W76 76 R76	Write Final End of Tape Read Final End of Tape	Final End of Tape Marker Final End of Tape Marker	Program Test Program Test
W77 77 R77	Write Early End of Tape Read Short Block	Early End of Tape Marker Short Block Read	Program Test Program Test
79	Writing Ring Present	Writing Ring present on spool, Tape Unit mechanically ready and Address seized, for the last Deck tested	Writing ring not present on spool or Tape Unit not mechanically ready or address not seized, for last deck tested
80	Tape Order Error	Unacceptable Instruction	Program Test
81 to 88	Deck Address (1 to 8)	Address seized and Tape Unit mechanically ready and not busy	Address not seized or tape unit busy or not mechanically ready
89	Transport mechanically ready and address seized	Tape unit mechanically ready and address seized	Tape unit not mechanically ready or address not seized.

NOTE: R stands for Read, W for Write

Table 5C: ONE-INCH (90kc/s) AND HALF-INCH (22½kc/s) MAGNETIC-TAPE INDICATORS



INDICATOR No.	TITLE	SET BY	UNSET BY
70	Write Unit Ready	Write Unit not busy	Write or Cancel Instruction
71	Write Master	Indicator 74 or 76 becoming set	Unsetting of both Indicators 74 and 76
72	Read Unit Ready	Read Unit not busy	Read or Back Space Instruction
73	Read Master	Indicators 75 or 77 becoming set	Unsetting of both Indicators 75 and 77
74	Write Errors	Any Errors during Writing	Write or Cancel Instruction
75	Read Errors	Any Errors during Reading	Read Instruction
76	End of Tape	End of Tape Marker During Writing	Program Test
77	Short Block	Short Block Read	Program Test
79	Writing Ring Present	Writing Ring present on spool, Tape Unit mechanically ready and Address seized, for the last Deck tested	Writing ring not present on spool or Tape Unit not mechanically ready or address not seized, for last deck tested
80	Tape Order Error	Unacceptable Instruction	Program Test
81 to 88	Deck Address (1 to 8)	Address seized and Tape Unit mechanically ready and not busy	Address not seized or tape unit busy or not mechanically ready
89	Transport mechanically ready and address seized	Tape unit mechanically ready and address seized	Tape unit not mechanically ready or address not seized.

Table 5D: QUARTER-INCH (16kc/s) MAGNETIC-TAPE INDICATORS

Table 6  
Standard Code for Card Reader and Card Punch

Card Punching	Numeric No Overpunch	Numeric + 10 Overpunch	Numeric + 11 Overpunch	Numeric + 0 Overpunch	Numeric + 1 Overpunch
10	10				
11	11				
0	0				
1	1	A	J	&	
2	2	B	K	S	%
3	3	C	L	T	$\frac{1}{4}$
4	4	D	M	U	-
5	5	E	N	V	/
6	6	F	O	W	$\frac{1}{2}$
7	7	G	P	X	.
8	8	H	Q	Y	@
9	9	I	R	Z	$\frac{3}{4}$
Computer Coded Zone Component	1	2	3	4	5

**Table 7**  
**Code for Line Printer**

Numeric Component	Zone Component 1	Zone Component 2	Zone Component 3	Zone Component 4	Zone Component 5
0	0	11	10	*	£
1	1	A	J	&	\$
2	2	B	K	S	%
3	3	C	L	T	$\frac{1}{4}$
4	4	D	M	U	-
5	5	E	N	V	/
6	6	F	O	W	$\frac{1}{2}$
7	7	G	P	X	.
8	8	H	Q	Y	@
9	9	I	R	Z	$\frac{3}{4}$

Recommended Paper-tape Codes

This is the basic code having six data bits with no parity bit.  
The main use of this code is as the foundation on which the other codes are based.

Numeric Component	Zone			
	0	1	2	3
0	Space	0		P
1		1	A	Q
2	New Line	2	B	R
3	Paper Throw	3	C	S
4	Tabulate	4	D	T
5	Backspace	5	E	U
6	Shift Out	6	F	V
7	Shift in and run out	7	G	W
8	(	8	H	X
9	)	9	I	Y
10		10	J	Z
11	£	11	K	
12			L	
13	&	+	M	
14	*	-	N	Escape ◇
15	/	.	O	Erase

Numeric Component	Track No.			
	4	3	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

REPRESENTATION WITHIN THE COMPUTER

Zone	Track No.	
	6	5
0	0	0
1	0	1
2	1	0
3	1	1

The Zone is registered in tracks 6 and 5 thus:

REPRESENTATION ON PAPER TAPE

Table 8A: RECOMMENDED PAPER-TAPE 6-TRACK CODE

The only difference from the 6-track Code is the provision for an ODD parity bit in track 5 and the positioning of the zone punching in tracks 6 and 7.

Numeric Component	Zone			
	0	1	2	3
0	Space	0		P
1		1	A	Q
2	New Line	2	B	R
3	Paper Throw	3	C	S
4	Tabulate	4	D	T
5	Backspace	5	E	U
6	Shift Out	6	F	V
7	Shift in & Run out	7	G	W
8	(	8	H	X
9	)	9	I	Y
10		10	J	Z
11	£	11	K	
12			L	
13	&	+	M	
14	*	-	N	Escape ◊
15	/	.	O	Erase

REPRESENTATION WITHIN THE COMPUTER

Numeric Component	Track No.			
	4	3	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

The Zone is registered in tracks 6 and 7 thus:

Zone	Track No.	
	7	6
0	0	0
1	0	1
2	1	0
3	1	1

REPRESENTATION ON PAPER TAPE

Table 8B: RECOMMENDED PAPER-TAPE 7-TRACK CODE

This code is an extended version of the 6-track code. It has the same six data bits with the addition of even parity and an eighth track, used in this specification to obtain a shift into lower case.

Numeric Component	Zone							
	0	1	2	3	4	5	6	7
0	Blank Tape	0		P	Space	0		p
1		1	A	Q		1	a	q
2	New Line	2	B	R		2	b	r
3	Paper Throw	3	C	S		3	c	s
4	Tabulate	4	D	T		4	d	t
5	Backspace	5	E	U		5	e	u
6	Shift Out	6	F	V	Shift Out	6	f	v
7	Shift In and Run Out	7	G	W	Shift In and Run Out	7	g	w
8	(	8	H	X		8	h	x
9	)	9	I	Y		9	i	y
10		10	J	Z			j	z
11	£	11	K				k	
12			L				l	
13	&	+	M				m	
14	*	-	N	Escape ◊			n	
15	/	.	O				o	Erase

Numeric Component	Track No.			
	4	3	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

REPRESENTATION WITHIN THE COMPUTER

The Zone is registered in tracks 6, 7 & 8 thus:

Zone	Track No.		
	8	7	6
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

REPRESENTATION ON PAPER TAPE

Table 8C: RECOMMENDED PAPER-TAPE 8-TRACK CODE

Table 9  
Code for Interrogating Typewriter

NUMERIC COMPONENT	ZONE							
	0	1	2	3	4	5	6	7
0	Space	0			*	£	,	¢
1	Do nothing	1	A	J	&	\$	=	×
2	Tabulate	2	B	K	S	%	<	÷
3	Set Tab	3	C	L	T	$\frac{1}{4}$	>	?
4	Clear Tab	4	D	M	U	-	+	#
5	Carriage Return/ Line Feed	5	E	N	V	/	↑	□
6		6	F	O	W	$\frac{1}{2}$	(	:
7		7	G	P	X	.	)	;
8		8	H	Q	Y	@	"	!
9		9	I	R	Z	$\frac{3}{4}$	↓	◇
10		10						
11		11						
12		12						
13		13						
14		14						
15		15						

**Table 10**  
**Peripheral Equipment Timings**

OPERATION	TIME FOR OPERATION	
	600 cards a minute Reader (Minimum times)	300 cards a minute Reader (Minimum times)
Card Reading Rate		
Complete Card Cycle	100 ms	200 ms
Time after 14th 6 Columns Read in which a Call Card instruction may be given to maintain continuous running of the reader	2.44 ms	7.3 ms
Time between calling a card and the first 6 Columns Read if the card reader has been unlatched. Register C comes into use for card reading before the first 6 Columns Read. Therefore if multiplication is taking place this should be reduced to:	36.1 ms  32.4 ms	70.4 ms  60.75 ms
Time between 14th 6 Columns Read and 1st 6 Columns Read of next card during continuous running. If multiplication is taking place this should be reduced to:	40 ms  35.4 ms	80 ms  70.8 ms
Interval between successive 6 Columns Read. Multiplication cannot be carried out during this time.	3.18 ms	6.36 ms
Time after 14th 6 Columns Read during which a Reject Card instruction can be given.	25.4 ms	50.8 ms
Time after which 6 Columns Read indicator (36) is automatically unset if not previously unset by program.	538 $\mu$ s	1076 $\mu$ s

**Table 10A: CARD READER TIMINGS**



OPERATION	MINIMUM TIME OF OPERATION
Card Punching rate	100 cards a minute
Card Cycle	600 ms
Calling Punch from rest, with motor running, to 1st Punch Index Point Time on.	35.83 ms
Index Point Interval	40.7 ms
Punch Index Point Time indicator (55) duration	6.52 ms
Check Index Point Time indicator (56) duration	14.66 ms
Interval between last setting of indicator 56 to last time to give 380042 instruction for continuous running.	58.22 ms
Interval between last setting of indicator 56 and 1st setting of indicator 55 when feeding continuously	102.2 ms
Time between last setting of indicator 56 and last time to give 380047 instruction	41.94 ms

Table 10B: CARD PUNCH TIMINGS

OPERATION	TIME OF OPERATION	
	600 l.p.m. PRINTER (Minimum times)	300 l.p.m. PRINTER (Minimum times)
Character time interval	1.2 ms	2.6 ms
Line space indication interval at speed	6.9 ms	6.9 ms
Duration of Line Space Time indicator (45) being set if not tested	3.1 ms	3.1 ms
Duration of Print Index Point Time indicator (43) being set if not tested	550 to 800 $\mu$ s	1150 to 1700 $\mu$ s
Time to space first line	32 ms	32 ms
Time to space subsequent lines	7.56 ms	7.56 ms

**Table 10C: LINE PRINTER TIMINGS**

OPERATION	TIME OF OPERATION
Reading Speed	1,000 characters a second
Character Read Interval	1 ms
Time to transfer character to buffer after sensing leading edge of hole	22 $\mu$ s
Time in which buffer must be unloaded to maintain continuous running	978 $\pm$ 50 $\mu$ s

**Table 10D: PAPER-TAPE READER TIMINGS**

OPERATION	TIME OF OPERATION
Punching Speed	300 characters a second
Character Time Interval	3.33 ms

**Table 10E: PAPER-TAPE PUNCH TIMINGS**

OPERATION	TIME OF OPERATION
Printing Speed	10 characters a second

**Table 10F: INTERROGATING TYPEWRITER TIMINGS**

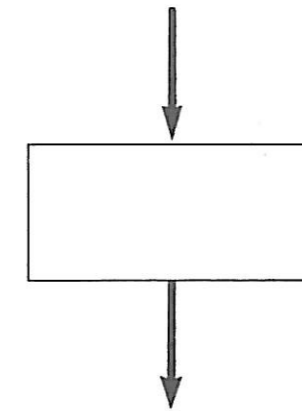
Timings and Statistics for One-inch (90kc/s) and Half-inch ( $22\frac{1}{2}$ kc/s) Magnetic-tape Systems

OPERATION	ONE-INCH	HALF-INCH
Time for a queued tape unit to come under computer control.	800 milliseconds after the unload instruction has been accepted	800 milliseconds after the unload instruction has been accepted
Fixed Delay Period on reading or writing	0.48 ms	2.04 ms
Delay before writing first block	2 seconds	2 seconds
Rewinding or unloading speed	Two speeds available, choice depending on engineers adjustment. (a) 150 inches a second. (b) 300 inches a second, subject to last 600 feet (approx.) of tape being rewound at 150 inches a second; previous 2,400 feet (approx.) or remainder of tape if less than 3,000 feet, is rewound at the full speed of 300 inches a second; remainder up to 600 feet, is rewound at 150 inches a second.	The rewinding speed is not constant but increases as the length of tape on the loading spool decreases. The time to rewind half a reel is therefore more than half the complete rewind time.
Time to rewind a 3,600 foot reel.	At 150 inches a second, 4 minutes 48 seconds At 300 inches a second, 3 minutes 12 seconds	Under 4 minutes
Length of long gap.	1.35 inches	1.24 inches
Length of short gap.	1.12 inches	1.0 inches
Time to create long gap on writing	11.2 ms	18.8 ms
Time to create short gap on writing	7.5 ms	13.4 ms
Time to traverse long gap on reading-with stop/start	11.2 ms	18.8 ms
Time to traverse long gap on reading-without stop/start	9.0 ms	16.6 ms
Time to traverse short gap on reading-with stop/start	9.7 ms	15.6 ms
Time to traverse short gap on reading-without stop/start	7.5 ms	13.4 ms
Time for a word to be transferred from Register G to I.A.S. during reading	15 $\mu$ s	15 $\mu$ s
Time for a word to be transferred from I.A.S. to Register F during writing	15 $\mu$ s	15 $\mu$ s
Minimum distance between final end of tape marker and actual end of tape	15 inches	15 inches
Minimum distance between early end of tape marker and final end of tape marker	15 feet	15 feet

**Table 12**  
**Timings and Statistics for Quarter - inch (16kc/s) Magnetic - tape Systems**

Maximum length of tape on spool	1,800 feet
Minimum distance between end of tape marker and actual end of tape	25 feet
Distance between beginning of tape and beginning of tape marker	7 to 10 feet
Tape Speed	37½ inches a second
Delay before writing first block	5 seconds
Rewind time for a complete reel	under 3 minutes
Packing density - assuming random distribution of digits	440 digits an inch
Packing density - assuming 25% of digits are zero - otherwise random distribution	480 digits an inch
Digit rate - assuming random distribution	16,500 digits a second
Digit rate - assuming 25% of digits zero otherwise random distribution	18,000 digits a second
Digit rate for all zeros	32,000 digits a second
Digit rate for all fifteens	8,000 digits a second
Average time for one word to pass the read/write heads - assuming random distribution	727 μs
Length of inter-block gap	approximately 0.8 inches
Time to traverse inter-block gap on writing, and write block start marker - excluding time when tape is awaiting an instruction	20 to 25 ms
Time to traverse inter-block gap and block start marker on reading - excluding time when tape is awaiting an instruction	20 to 25 ms
Break-in time when transferring a word to or from I.A.S.	16 μs

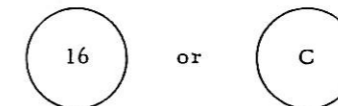
**Table 13**  
**Program Flowchart Symbols**



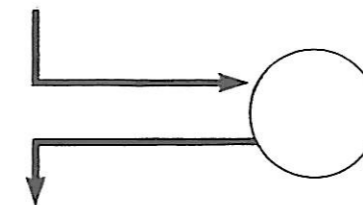
Normal Arithmetic or Transfer Process



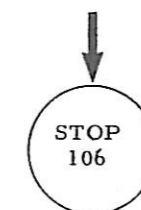
Decision or Test



Fixed or Variable Connector (Switch)

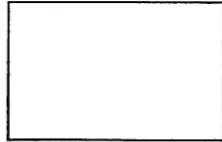
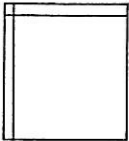

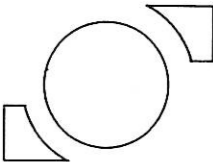

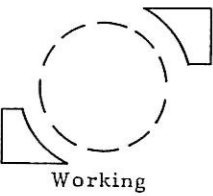
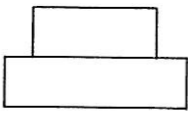
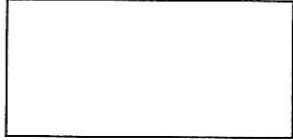
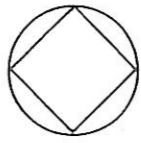
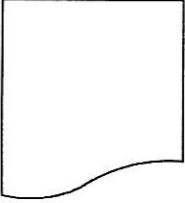
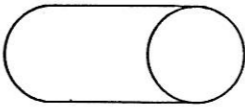
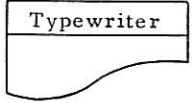


Exit to and return from Subroutine



Stop (or Start) point

**Table 14**  
**Systems Flowchart Symbols**

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

NON-COMPUTER EQUIPMENT


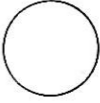
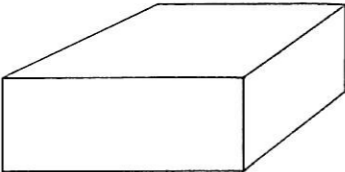


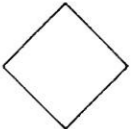
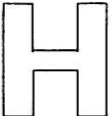
Description	I.C.T. Symbol	Description	I.C.T. Symbol
Punch and Verify		Sort	
File of Punched Cards		Tabulator	
Interpret, Match, Collate, Interpolate or Reproduce		Calculator	
		Manual Operations	

Table 15

Operator's and Programmer's Display and Switch Panels

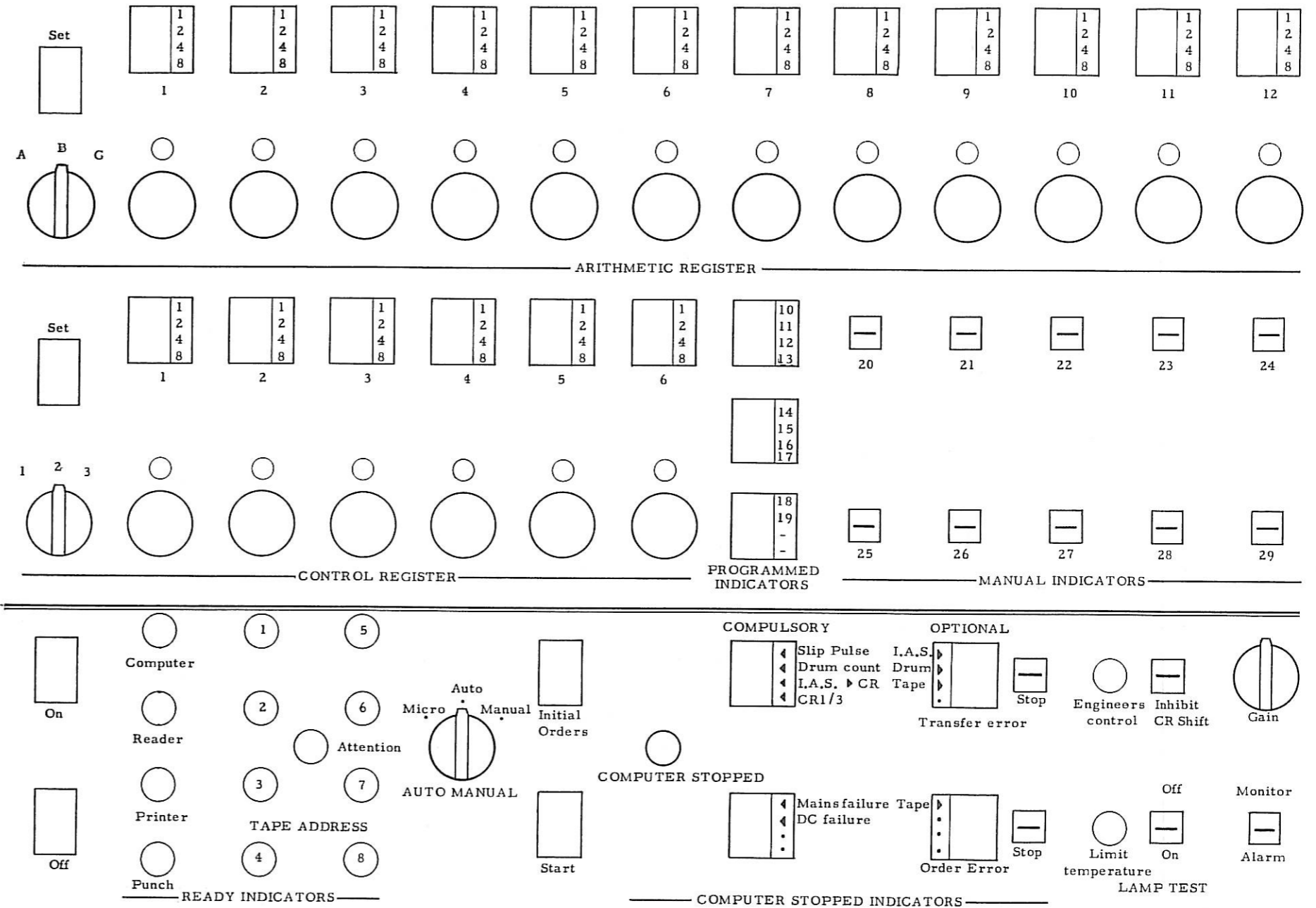
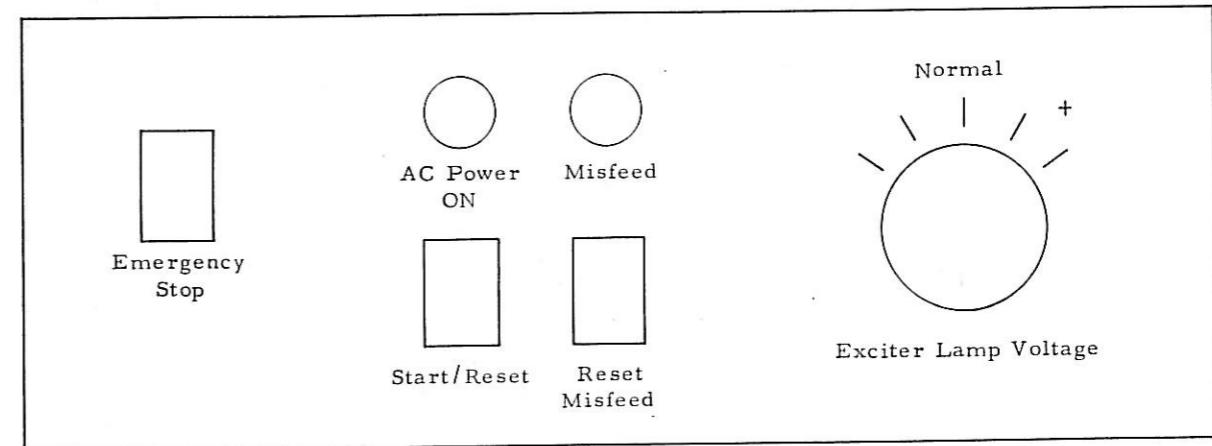
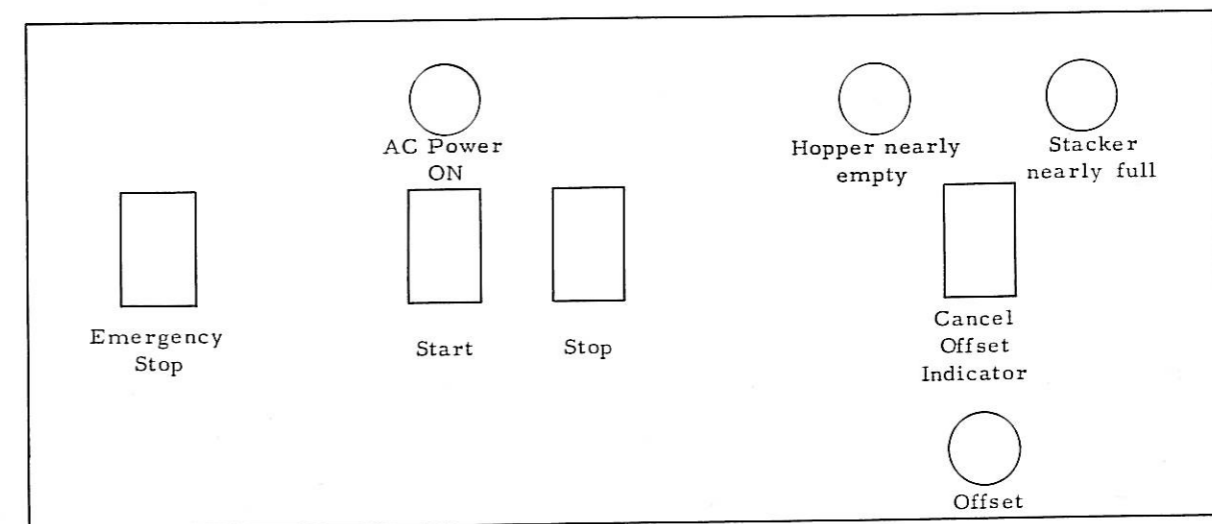


Table 15A: COMPUTER CONSOLE

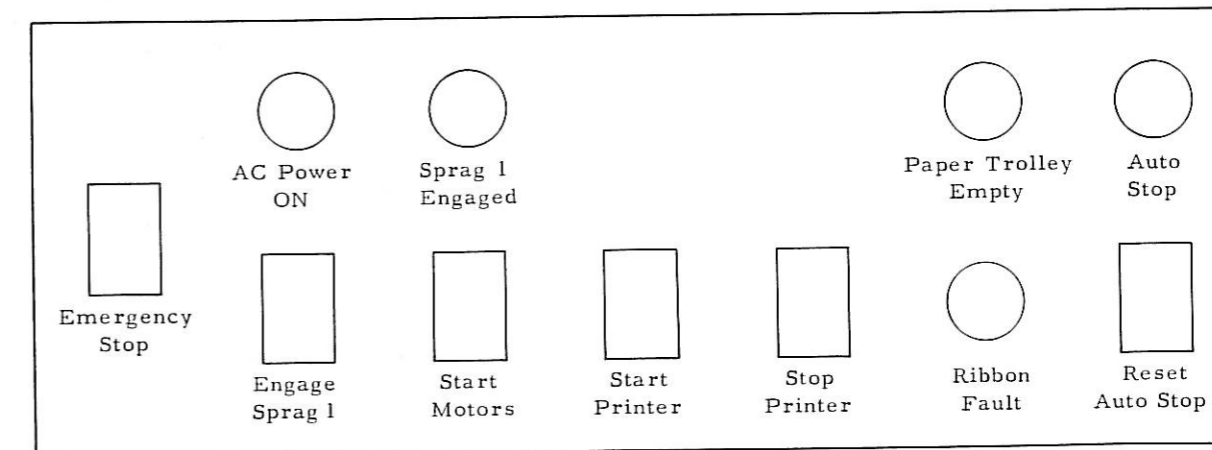




(i) CARD READER



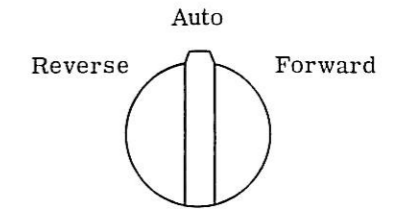
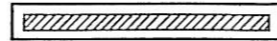
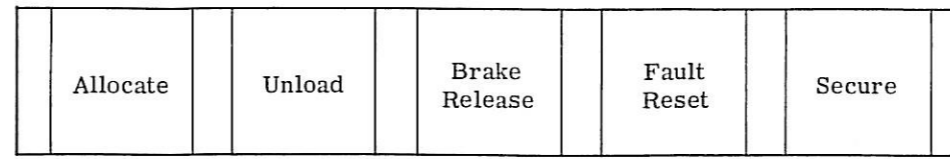
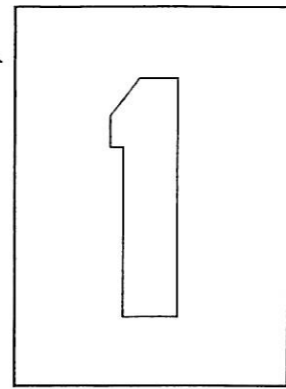
(ii) CARD PUNCH



(iii) LINE PRINTER

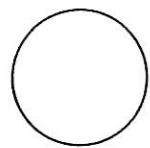
Table 15B: PERIPHERAL EQUIPMENT

Address allocated in this space.

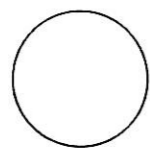


(i) QUARTER-INCH (16kc/s) MAGNETIC-TAPE UNIT

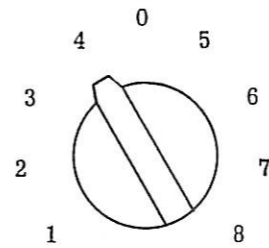
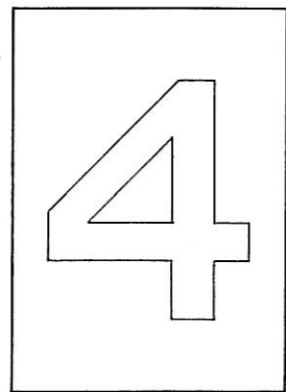
Address allocated in this space.



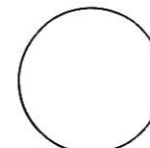
AC Power ON



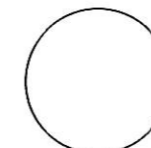
Transport ON



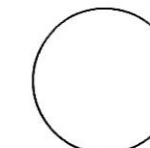
Selector



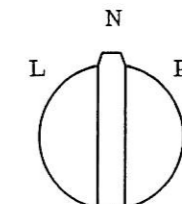
Secure



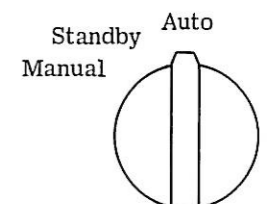
Overload



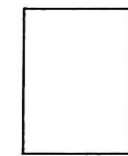
Lockout



Marginal Test



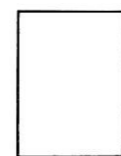
Auto/Manual



Emergency Stop



Start



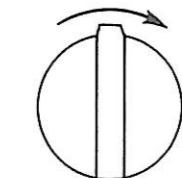
Allocate



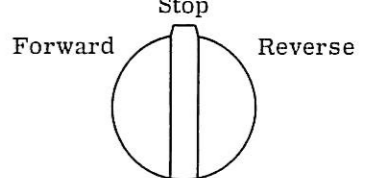
Unload



Reset



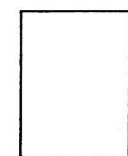
Leader Drive



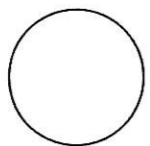
Selector

(ii) HALF-INCH (22½kc/s) MAGNETIC-TAPE UNIT

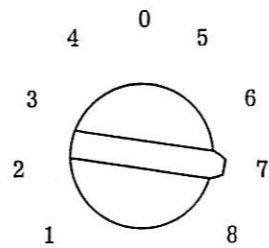
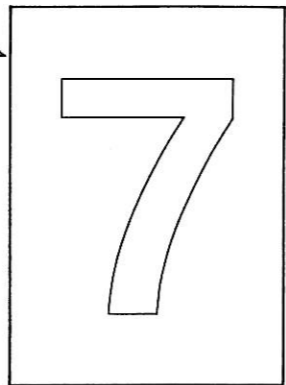
Address allocated in this space.



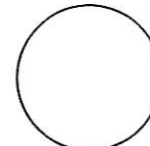
Emergency Stop



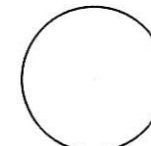
AC Power ON



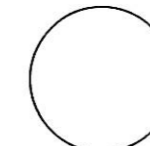
Selector



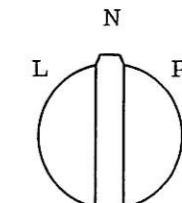
Secure



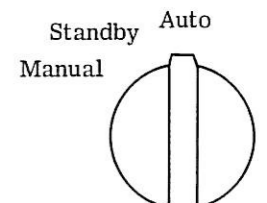
Overload



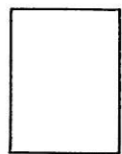
Lockout



Marginal Test



Auto/Manual



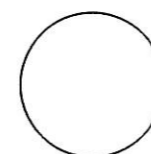
Start



Allocate



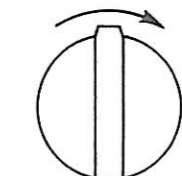
Unload



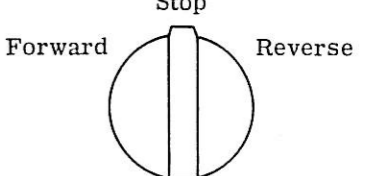
Loop



Reset



Leader Drive



Selector

(iii) ONE-INCH (90kc/s) MAGNETIC-TAPE UNIT

Table 15C: MAGNETIC-TAPE UNITS

### **Access Time**

The time elapsing between giving an instruction which operates on data held in store (e.g. I.A.S. or magnetic drum) and the instant when the data are so positioned that the operation can commence.

### **Address**

The name or number identifying a location either in a store (e.g. I.A.S.) or in some other part of the computing system (e.g. a tape deck).

**Absolute Address** The number identifying a *specific* location in a store.

**Relative Address** A number identifying a location which is relative to another address, and is therefore, not a specific storage location.

### **Assembler (Assembly System)**

A program which converts autocode instructions into a machine-coded program.

### **Autocode**

A programming language intended to simplify programming by the use of macro-instructions (q.v.) written in an elementary programming code.

### **Binary Coded Decimal (B.C.D.)**

A system of binary notation in which the decimal digits 0 to 15 are represented by four bits which have values of 1, 2, 4 and 8 respectively.

### **Binary Notation**

A system of positional notation in which the digits are coefficients of powers of a base of two.

### **Bit (Binary Digit)**

A digit (0 or 1) in binary notation.

### **Block**

A group of consecutive data or program words considered or transferred as a whole.

**Block Relativizer**

The relativizer used for the I.A.S. and drum settings given in the block relativizer control words. As each block is read by Initial Orders the block relativizer setting is used to convert addresses specifying relativizer 'B' to absolute form.

**Buffer**

A storage device (either a special register or an area of I.A.S.) used to compensate for the difference in rates of flow of information or in times of occurrence of events when transferring information from one device to another, as from input unit to I.A.S. or I.A.S. to output unit.

**Channel**

A recording band on the magnetic drum comprising 200 locations.

**Compiler**

A program which has the characteristics of an assembler (i.e. converting autocode instruction to machine-coded instructions) but is more comprehensive. It usually produces several machine-coded instructions from one autocode instruction and assembles a complete program by allocation of storage etc.

**Decade**

The smallest unit of transfer from the magnetic drum; it comprises 10 consecutive storage locations on one channel i.e. there are 20 decades per drum channel.

**E - Card**

A card bearing the designation E which is the last program card in a pack. An 'entry word' punched in the E-card effects an entry to the program under Initial Orders (q.v.).

**E - Word**

The 'entry word' which is punched in a program pack after the last program word and which effects an entry under Initial Orders. It is usual to punch the E-word on a separate card, the E-Card.

**End of Block Marker**

A marker to indicate the end of a data or program block.

*Applied to magnetic tape* A word after the last data word which contains 15 in every digit position.

*Applied to program cards* A punching with a non-zero numeric component in column 17 of the last card of the block.

**Frame**

A cross-section of one-inch or half-inch magnetic tape which consists of one bit position for each tape track. According to the number of tape tracks, a frame may be used to record one or more characters.

**Hardware**

A term for the computer and the mechanical, magnetic, electronic and electrical devices from which it is made. A colloquial term for apparatus as opposed to program.

**Housekeeping**

Programs or instructions which are included for organizational purposes and do not actually perform calculations. The term is particularly applied to magnetic tape organization.

**Initial Orders**

An input routine permanently stored on the reserved channels of the magnetic drum which enables programs to be read into the computer and stored on the drum.

**Key**

- (a) A group of characters usually representing an item which is used to identify a record, or
- (b) A parameter (q.v.).

**Link**

An instruction which returns control from a subroutine to the main program. The link is stored by the subroutine in a word allocated for the purpose.

**Loop**

A group of instructions which may be obeyed more than once. A loop will normally be conditional upon the testing of an indicator. Depending on the state of an indicator (set or unset) a number of instructions will be repeated and the indicator tested again. The loop will continue until the state of the indicator is changed.

**Macro - instruction**

Usually used in autocode programs where one program instruction, a macro-instruction, corresponds to several machine-coded instructions.

**Mask**

A constant used with a Logical AND instruction to extract information from part of a word.

**Microsecond**

One millionth of a second ( $\mu$ s).

**Millisecond**

One thousandth of a second (ms).

**Object Program**

A complete program in machine code produced by the action of a compiler or assembly system upon a source program (q.v.).

**Off - line**

A data processing machine which operates when not directly linked (electronically or mechanically) to the computer is said to be used off-line.

**On - line**

A data processing machine which operates when directly linked to a computer (e.g. for input/output purposes) is said to be used on-line.

**Packing Density**

The number of digits that can be accommodated on a given amount of storage. The term is especially applied to magnetic tape.

**Parameter**

Information or requirements supplied to a subroutine by the main program, the format and address for the information being set down in the subroutine specification.

**Parity Bit**

A bit which is automatically generated and appended to an array of bits (usually representing a character) to make the sum of the number of 1-bits in the array either odd or even, as nominated. The parity bit is included for checking purposes only.

**P.P.F. (Print Punch Feed)**

A utility program allowing input/output units to operate in parallel with maximum time-sharing under a control routine.

**Relativizers**

A number which can be set to give an I.A.S. and drum setting for the appropriate R.R.N. The relativizer settings are used by Initial Orders to convert relative addresses to absolute addresses.

**Queueing**

If, on one-inch (90 kc/s) or half-inch ( $22\frac{1}{2}$  kc/s) magnetic-tape systems, a tape deck is allocated an address and that address has been previously seized by another tape deck, then the second tape deck to be allocated the address is said to be 'queued'. The queued tape deck will seize the address when that address becomes available.

**R.R.N. (Relativizer Reference Number)**

A reference number which is associated with a block of program or data enabling the words in the block to be addressed in relative form.

**Search Code**

A code is held in I.A.S. which is compared with codes which are input from a paper-tape reader or punch, interrogating typewriter etc., a successful comparison indicating that a specific code has been read on input.

**Software**

An antonym of 'Hardware' (q.v.). A colloquial term for programs available (e.g. subroutines and autocode programs) as opposed to facilities provided by the apparatus of the computer.

**Source Program**

A program written in an autocode programming language which will be converted to machine code by a compiler or assembly system (q.v.).

**Subroutine**

A self-contained section of program which can be incorporated into a complete program.

**Time - sharing**

A method of programming used to reduce the running time on the computer. When an instruction has been initiated and the execution of that instruction is time consuming but does not involve Register A or Register B (e.g. an instruction governing a peripheral unit) control can be transferred to another section of program so that the two operations are carried out simultaneously.

**Track**

A longitudinal section of magnetic or paper tape which consists of a series of recording positions. A cross-section of the tape consists of one recording position for each track and may be used to represent one or more characters.

**Writing Ring**

A safety device used on magnetic-tape spools to prevent the overwriting of 'Master' information.