KDP 10

Programming Manual

ENGLISH ELECTRIC-LEO-MARCONI COMPUTERS LTD
KIDSGROVE STOKE-ON-TRENT STAFFORDSHIRE Telephone: Kidsgrove 2141

Publication 1022 100165 price: three guineas

CONTENTS

Section Page

Section Page

GENERAL DESCRIPTION

The KDP 10 Computer and associated peripheral equipment together form a medium sized general purpose Electronic Data Processing System specially designed to handle commercial data. The use of printed circuits and transistors in the logic elements and magnetic cores in the computer storage system ensures the high standard of reliability essential in this type of machine.

The following features of the system make it especially suitable for commercial applications:

1. **COMPLETELY VARIABLE DATA ORGANISATION.** The system is capable of accepting and processing alpha-numeric data originating on punched paper tape, punched cards and magnetic tape. Information from each type of input medium is stored in the internal memory in character form and every character is addressable. Special instructions in the order code permit the processing of variable length items and operands and it is not necessary to reserve space for maximum capacity items and messages. This ability to dispense with redundant padding characters, necessary in fixed word length machines, saves space on tape and in the high speed memory and decreases processing time.

2. **ADDRESSABLE REGISTERS.** The registers used by the programme control unit in the execution of orders are accessible to the programmer by the use of special instructions in the order code. The final contents of these registers helps the programmer to control variable length data operations.

3. **DECIMAL ARITHMETIC.** In addition to the normal binary arithmetic operations there are provided the four usual arithmetic operations which handle decimal information. Two variable length operands may be added, subtracted, multiplied or used in division operations.

4. **ACCURACY CHECKING.** Built in checking ensures the correct transfer of information between peripheral equipment and computer and within the computer itself. Arithmetic operations are checked by repeat operations using the complements of the operands.

5. **SIMULTANEOUS OPERATION.** Time sharing of the control organisation by the input-output equipment and the processing circuits within the computer permit simultaneous operation. At the same time the system may be performing any different two of the following operations:

   (a) Reading information from tape
   (b) Writing information on tape
   (c) Computing
   (d) Paper advancing on the High Speed Printer

6. **EXPANDABILITY.** The system is flexible and readily capable of expansion in terms of number of tape units, size of internal memory and type of peripheral equipment.

The Computer is a digital machine in which information is stored in binary coded characters. Each character, whether on tape or within the computer, uses six information digits and a parity digit. Sequential control governs the order in which instructions are obeyed and normally one instruction must be complete before the next instruction starts. This, of course, does not apply to the simultaneous mode of operation where certain selected operations may proceed together. The use of a core store provides random access to internally stored information. A KDP 10 Computer is made up from a selection of on-line units chosen from the following:

   (a) High Speed Memory
   (b) Computer Programme Control
   (c) Tape Selecting and Buffer Unit
   (d) Control Console
   (e) Monitor Printer
   (f) Paper Tape Reader
   (g) Paper Tape Punch

The HIGH-SPEED MEMORY is a random access, magnetic core device which provides storage and work area for programmes and data. The memory is available in increments of 16,384 character locations and may be expanded to a maximum of 262,144 locations. Each location is individually addressable and can store any one of the sixty-four characters. These characters (KDP 10 Code) include all the letters of the alphabet, the ten decimal digits, control symbols and special marks. One character or four characters in parallel can be addressed, brought into the Memory Register and regenerated in their original locations in one 15-microsecond cycle.

The PROGRAMME CONTROL is the arithmetic and logical control element of the Computer. It interprets and executes the instructions of the programme stored in the High-Speed Memory and performs the automatic accuracy checks. The Computer and the on-line peripheral devices operate in accordance with a stored programme of two-address instructions. The instructions that can be executed by the Programme Control include all the categories necessary for processing of data: Input-Output, Data Handling, Arithmetic, and Decision and Control. Each instruction is made up of eight characters and consists of four parts: (1) an operation code (read, multiply, transfer, etc.), (2) an A address (usually the High-Speed Memory address of an operand or the left boundary of an operand), (3) a B address (usually the High-Speed Memory address of an operand or right boundary), and an N code. The N code permits automatic modification of address A and/or B through the use of any of the seven (four static and three dynamic) Address Modifiers.

The TAPE SELECTING AND BUFFER UNIT-A permits connection of one to eight Tape Stations to the Computer, and controls reading from and writing to magnetic tape on these stations. Programme instructions designate the appropriate Tape Station for input or output. The number of Tape Stations directly controlled by the Computer may be increased to as many as sixty-two by connection of a TAPE SELECTING UNIT-B to each of the Unit-A trunk lines. Write-out from the Computer to magnetic tape is at the rate of 16,667 or 33,333 characters per second. Read-in from magnetic tape is at the rate of up to 33,333 characters per second.

The CONSOLE provides for complete monitoring of operation of the Computer and the on-line devices with panel display and control of register and status level action. Automatic and manual operation, maintenance, programme insertion and programme testing can be accomplished from the Console.

Console facilities include :

1. Manual start and stop at a given instruction or at a given address.
2. Control and display of registers and counters.
3. Accuracy-checking indicators.
4. Indicators for currently performed instruction.
5. Indicators for last or currently selected Tape Station.
6. Control and display of Character Recognition flip-flops.
7. Breakpoint switches.
8. Alarm indicators.

The MONITOR PRINTER is an on-line device, similar to an electric type-writer, that prints on paper stock from information received directly from the Computer's memory. It operates at the rate of ten characters per second and is used primarily for programme operational control, programme testing, and exceptional types of output. The Paper Tape Punch associated with this device can produce seven-hole punched tape simultaneously with the Monitor Printer's output of hard copy.

The PAPER TAPE READER accepts seven-hole punched paper tape and operates at the rate of 1000 characters per second. It is used largely for initial programme insertion, programme testing, and insertion of periodically changing constants.

The PAPER TAPE PUNCH may be one of two models operating at either 60 characters per second or 300 characters per second and producing punched seven hole paper tape.

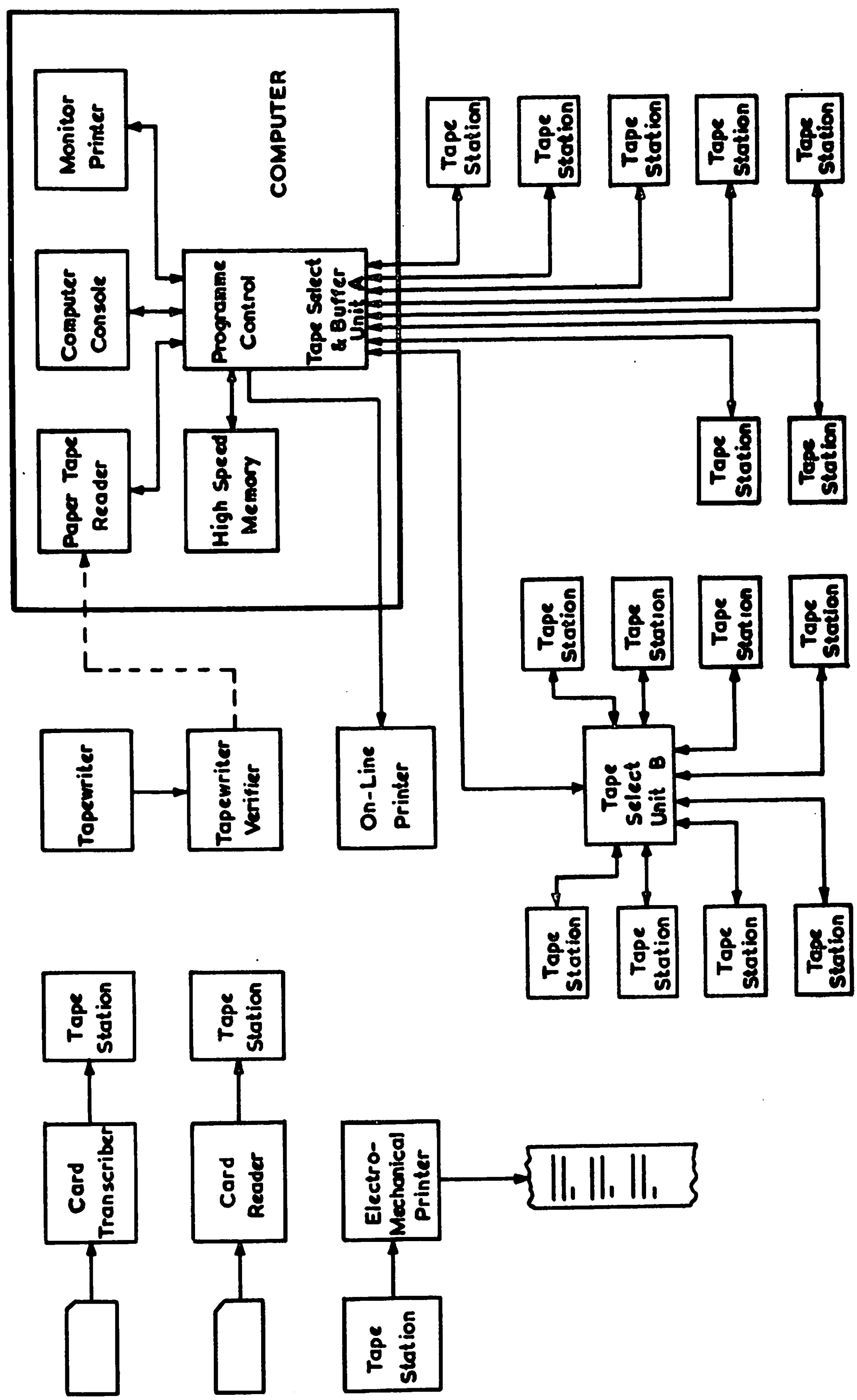


FIG. 1.1 DIAGRAM OF EQUIPMENT INTERCONNECTIONS

High-Speed printing can be accomplished on-line (ON-LINE PRINTER) or off-line (ELECTRO-MECHANICAL PRINTER) in the KDP 10 System. The print line capacity is 120 characters and the print rate is 600 lines per minute. The On-Line Printer accepts data directly from the Computer memory and operates under the direction of the stored programme. The Printer is converted to off-line by the addition of a DATA EDITOR, operating then under the direction of a plugboard programme and a paper tape loop, with information received directly from a Magnetic Tape Station.

In addition to the on-line equipment described above the KDP 10 System includes a choice of peripheral off-line equipment comprising Card Transcriber, Card Reader, Card Punch, Tapewriter and Tapewriter Verifier. These are described in detail in the KDP 10 Functional Specifications and only brief descriptions are included here.

The CARD TRANSCRIBER comprises two units, a CARD READER and a CARD EDITOR. The Card Reader may be used without the Editor, in which case editing is reserved for the Computer. This device converts characters on eighty-column punched cards to coded characters on magnetic tape, at the rate of up to 400 cards per minute. The Card Reader includes a control panel, an automatic card-handling mechanism and two card-reading stations. Each card is read at both stations, and the readings are compared as an accuracy check. The Card Editor permits rearrangement and selective transcription of card data and insertion of additional characters. The Card Transcriber employs transistor circuitry and accuracy checking, including parity, comparison and multi-punch checks.

The TAPEWRITER and TAPEWRITER-VERIFIER are used for original preparation and verification of coded, seven-hole punched paper tape for subsequent input to the Computer via the Paper Tape Reader. These devices are keyboard operated and simultaneously print on paper stock the same information that is being punched on tape. The Tapewriter-Verifier automatically checks the accuracy of its output by comparison with a previously prepared (Tapewriter) punched paper tape. Whenever a character being punched on the Tapewriter-Verifier is not in agreement with the related character on the original tape, both the keyboard and the punch lock. Both devices will function at typing speeds up to 10 characters per second, and both include parity checking.

SUMMARY OF EQUIPMENT PERFORMANCE

ON-LINE EQUIPMENT

MAGNETIC TAPE	-	33,333 characters per second maximum.
PAPER TAPE	-	Input - 1000 characters per second. Output - 100 or 300 characters per second.
MONITOR PRINTER	-	10 characters per second with punched paper tape at this rate also.
HIGH SPEED PRINTER	-	600 lines per minute. 120 characters per line.

OFF-LINE EQUIPMENT

CARD TRANSCRIBER) CARD READER)	-	400 cards per minute.
TAPEWRITER and TAPEWRITER VERIFIER	-	10 characters per second.
ELECTRO MECHANICAL PRINTER	-	600 lines per minute. 120 characters per line.
CARD PUNCH	-	150 cards per minute.

ACCURACY CHECKING FEATURES

The KDP 10 design incorporates several accuracy checking features which aim to prevent incorrect information from entering or leaving the System. The provision of extra equipment for checking must be selective or there is a danger of overloading the System with so much extra equipment that further check circuits are required to check the original checking equipment. The following description indicates the techniques used and the area of application.

PARITY CHECKING

Each character on magnetic tape and in the computer memory carries an extra 'bit', or binary digit, to make the total number of '1' bits an odd number. On paper tape the number of '1' bits is an even number. Correct parity is ascertained on read-in, during computer operations and on write-out.

During read operations the characters are checked in the Tape Station before they are stored in the HSM. On output operations the Tape Stations are arranged to provide echo returns from the tape recording heads and so indicate whether a correct character has been written. Parity checking is equally important in the off-line equipment and is extensively used in card-to-tape and tape-to-card conversions and in the off-line printer.

DUAL RECORDING ON MAGNETIC TAPE

The bits of each character together with a timing pulse are recorded in duplicate on sixteen channels across the width of the tape. All dually recorded characters are read or written simultaneously and good characters from either track are accepted. By this means either one of the two recorded spots for a single bit may be missing and the character still be read successfully. Failure to read a particular bit may occur through flaws in the tape or through deterioration of the tape after many passes through the tape unit. Dual recording, besides improving the accuracy of recording also lengthens tape life.

REPEATED OPERATIONS

If a parity error occurs on tape during a read operation the tape is automatically returned to the beginning of the block or message and the read instruction repeated. If, on the second run the same, or another parity error is detected the Computer stops and an alarm light indicates the reason for the stoppage. This feature is known as ROLLBACK and further details are given later in this manual. Magnetic tape writing operations are protected by a system of FLAW MARKING and FLAW DETECTION. Known bad areas of tape may be identified and when these are encountered during a write operation the Computer stops, returns to the beginning of the block, erases the incomplete message or block and re-writes the whole block or message beyond the flaw.

ARITHMETIC CHECKS

All additions and subtractions are checked by repeated operations using the complements of the operands, character by character. As decimal multiplication and division are performed by repeated additions and subtractions these operations are also checked. If agreement is not obtained the Computer stops and indicates the cause of the stoppage. This checking does not increase the arithmetic operation times.

APPLICATION OF CHECKING TECHNIQUES

PROGRAMME CONTROL. The following conditions occurring within the Computer will cause the machine to stop:

1. Incorrect parity in MEMORY ADDRESS REGISTER.
2. Incorrect parity in MEMORY REGISTER.
3. ARITHMETIC UNIT failure.
4. Incorrect parity in BUS ADDER.
5. Invalid operand in a DECIMAL OPERATION.
6. Incorrect parity in NORMAL OPERATION REGISTER.
7. Incorrect transfer of an operation from NORMAL to SIMULTANEOUS MODE.
8. More than one PREVIOUS RESULT INDICATION.
9. Failure of TIME PULSES.

INPUT OUTPUT EQUIPMENT. The following input output conditions cause the Computer and input or output unit to stop:

1. Tape Station detects signals in an INTERMESSAGE GAP.
2. MISSING CLOCK PULSE.
3. Tape Station control signals incorrectly obeyed.
4. Odd number of characters in a paper-tape block read.
5. Second parity error after ROLLBACK.
6. Incorrect Tape Selection.
7. Incorrect data format, e.g., incorrect Start Message - End Message sequence.
8. Incorrect parity return from recording-head during a Tape WRITE operation.
9. Incorrect paper tape parity.
10. ON-LINE PRINTER not operable.
11. ON-LINE PRINTER paper supply low.

INFORMATION REPRESENTATION

2

GENERAL INTRODUCTION

All Electronic Computers, irrespective of their application to scientific problems or commercial data processing, require information as basic raw material. Briefly, they need to be given something on which to work and must be supplied with instructions which specify what to do with it.

The information supplied is thus of two kinds, (a) problem input-data and (b) a programme of instructions. Both types of information are supplied in the form of binary-coded characters which, on magnetic tape, paper tape or within the machine, not only look alike but are alike. There is, however, no real danger of confusion in practice. The machine is informed, at the start of a run, that the information in specified places within the Computer represents instructions and these it interprets in accordance with a set of rules known as the Order-Code.

Although all the information within the Computer is of the same form, the programmer prepares the information in different forms. Problem data is visualised within the machine as it appears elsewhere. A name and address, an alphabetic description or a decimal number are visualised as they actually appear. Instructions, however, are prepared in the form of coded numbers having three main parts. One part of the number, the Operation Code, specifies the operation. Two other parts, the addresses, specify the exact position in the machine at which will be found the two pieces of information which are the subject of the operation.

All decimal and alphabetic information is stored in the form of binary-coded characters, instructions are prepared on coding sheets using the octal system of notation and the machine can perform arithmetic operations in either the decimal or binary systems. The programmer requires to know and become familiar with a machine character code, and to have some experience in handling binary and octal numbers. These are the subject of the present chapter.

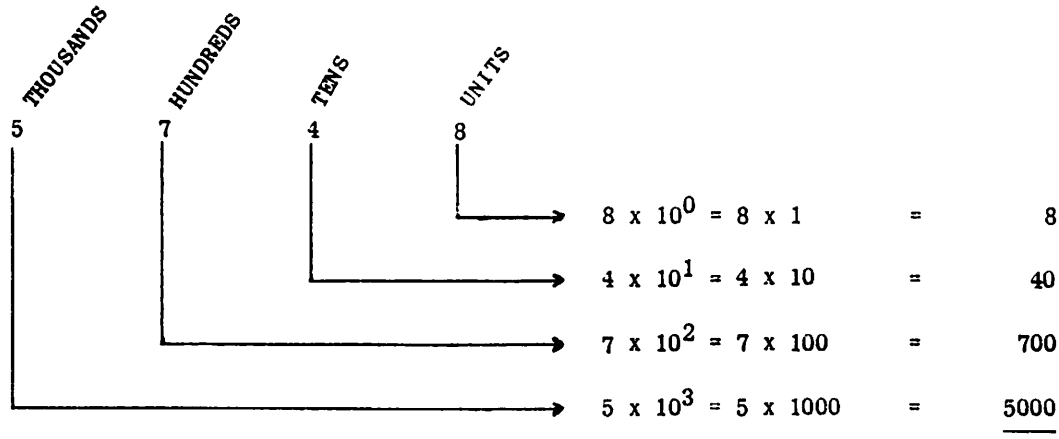
BINARY ARITHMETIC

In the decimal system a number, five thousand seven hundred and forty-eight, is written.

5748

This is a **positional** number system using a base or radix of ten. It requires ten different cyphers or symbols, 0, 1, 2, ... 8, 9 to indicate the ten ideas of nothing, one thing, two things and so on up to nine things. There is no single cypher which represents ten things. To do this it is necessary to write the two cyphers, 10 indicating that the number has one thing of a different value (namely tens) and no units.

5748 is an abbreviation for: $5 \times 10^3 + 7 \times 10^2 + 4 \times 10^1 + 8 \times 10^0$



In the BINARY SYSTEM the base or radix is two, leading to a positional system in which the columns of a number represent powers of two.

The positional weights (powers) in a binary number are shown below:-

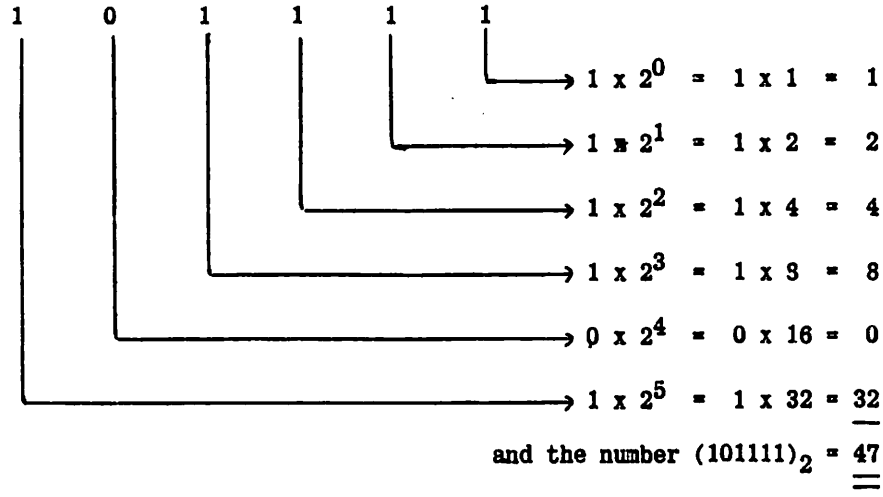
THIRTY-TWOS	SIXTEENS	EIGHTS	FOURS	TWOS	ONES
X	X	X	X	X	X
32	16	8	4	2	1
2^5	2^4	2^3	2^2	2^1	2^0

The great advantage of this system for automatic computing is the fact that only two cyphers are now required, 0 and 1 representing nothing or one thing. To represent a quantity two, it is necessary to write two cyphers, a one in the two's column and zero in the units column.

Example:

47 decimal = 101111 binary

or $(47)_{10} = (101111)_2$



A six column binary number, usually referred to as a six digit number, contains sufficient positions to represent all numbers from $(000000)_2$, the smallest, to $(111111)_2 = (63)_{10}$, the largest, a total of sixty-four numbers in all.

Some of the sixty-four different patterns obtainable with six binary digits are shown below. The reader should complete the table for himself and should find no difficulty if he notices that the units column alternates 0, 1, 0, 1; the next column alternates in pairs 00, 11, 00, 11; the next column in fours 0000, 1111 and so on, throughout all columns. This is a characteristic feature of a binary table of the natural numbers in consecutive sequence.

BINARY							DECIMAL
0	0	0	0	0	0	=	0
0	0	0	0	0	1	=	1
0	0	0	0	1	0	=	2
0	0	0	0	1	1	=	3
0	0	0	1	0	0	=	4
0	0	0	1	0	1	=	5
0	0	0	1	1	0	=	6
0	0	0	1	1	1	=	7
0	0	1	0	0	0	=	8
		.					
		.					
		.					
		.					
1	1	1	0	1	1	=	59
1	1	1	1	0	0	=	60
1	1	1	1	0	1	=	61
1	1	1	1	1	0	=	62
1	1	1	1	1	1	=	63

One disadvantage from a programmer's point of view is the length of the numbers required. In the table above six binary digits are needed to represent the quantity 63 which requires only two decimal digits. A ten digit decimal number requires about thirty binary digits or a ratio of about 3 to 1.

To overcome the tediousness of writing large numbers as long strings of ones and noughts the binary number is partitioned in groups of three and represented by a number composed of the decimal equivalent of each group, in the order in which they appear from end to end.

Example:

The binary number 110111101011010 is written:

110	111	101	011	010
6	7	5	3	2

The partitioning into groups of three must start at the right-hand end so that any incomplete group appears at the left-hand end.

Example:

	1	101	011	001
=	001	101	011	001
=	1	5	3	1

In the examples provided the binary numbers may be represented as 67532 and 1531 respectively. These, however, are not the decimal equivalents of the binary numbers and some means must be provided to distinguish between 67532 representing the decimal representation of a binary pattern and 67532 meaning sixty-seven thousand five hundred and thirty-two.

This distinction may be made by writing the abbreviation in brackets thus $(\quad)_8$, i.e.,

$$(67532)_8 = (110\ \ 111\ \ 101\ \ 011\ \ 010)_2$$

$$(1531)_8 = (001\ \ 101\ \ 011\ \ 001)_2$$

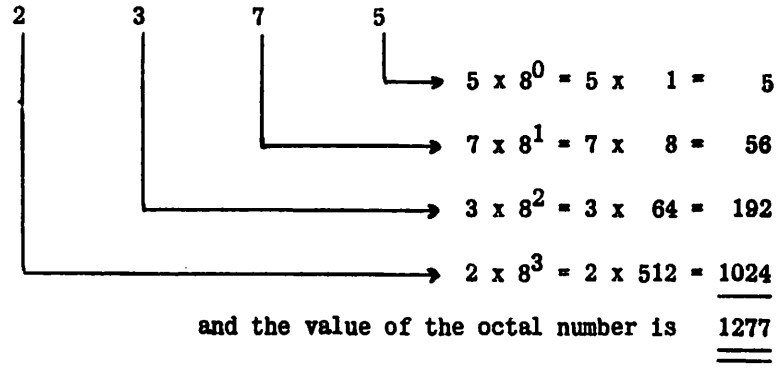
The notation $(\quad)_8$ is used because the abbreviation is really a number in the scale of eight or octal system, which will now be described.

OCTAL ARITHMETIC

In this system the base or radix is eight and the eight basic cyphers are 0, 1, 2, 3, 4, 5, 6, 7.

The octal number $(2375)_8$ represents, in value, the same quantity as the decimal number $(1277)_{10}$ as shown in the following example:

$$(2375)_8 = 2 \times 8^3 + 3 \times 8^2 + 7 \times 8^1 + 5 \times 8^0$$



The binary equivalent of $(1277)_{10}$ is $(010\ \ 011\ \ 111\ \ 101)_2$. Grouping this in threes and writing the abbreviation as indicated in the previous discussion gives:

010	011	111	101
2	3	7	5

The octal notation and the group of three abbreviations of a binary number are the same thing. Taking the earlier examples:

$$(110\ \ 111\ \ 101\ \ 011\ \ 010)_2 = (67532)_8 = (28506)_{10}$$

$$(001\ \ 101\ \ 011\ \ 001)_2 = (1531)_8 = (857)_{10}$$

The octal system is so much more convenient than the binary system as a notation for writing binary numbers that it is adopted in many computing machines, including the KDP 10. Whenever reference is made to a **binary** number or character stored within the KDP 10, the number is specified using octal notation. All KDP 10 instructions are written in the octal notation and all the indicator lights on the computer console are arranged in groups of three to facilitate reference to the numbers displayed in the octal system. The octal system is used only when necessary or convenient. Alpha-numeric data within the computer is referred to in the usual way and it is quite unnecessary to refer to it in either binary or octal though the programmer must know the octal equivalent of each character represented in the KDP 10 Code.

NUMBER SYSTEM CONVERSION

DECIMAL TO BINARY - INTEGERS

To convert whole numbers from decimal to binary proceed as follows:

1. Divide the decimal number by two; the remainder will be either one or zero.
2. If the remainder is one the least significant binary digit is one, if it is zero the least significant binary digit is zero.
3. Divide the quotient by two: a remainder of one indicates that the next significant digit is a one, otherwise it is zero.
4. Repeat the above steps until a quotient of zero is obtained.

Example:

Convert $(15)_{10}$ to binary. It is useful to work upwards as follows:

	0		
2	1	 1	4th remainder
2	3	 1	3rd remainder
2	7	 1	2nd remainder
2	15	 1	1st remainder

$$\therefore \quad (15)_{10} = (1111)_2$$

Example:

Convert $(43)_{10}$ to binary

	0		
2	1	 1	6th remainder
2	2	 0	5th remainder
2	5	 1	4th remainder
2	10	 0	3rd remainder
2	21	 1	2nd remainder
2	43	 1	1st remainder

$$\therefore (43)_{10} = (101011)_2$$

The method is suitable for small numbers, and as very little conversion is needed, this not very sophisticated method is sufficient.

BINARY TO DECIMAL CONVERSION

To convert a binary number to decimal proceed as follows:

1. Take the top (most significant) binary digit and multiply by two.
2. Add the next binary digit to the result obtained in 1 and multiply by two again.
3. Repeat the operations until the last binary digit has been added.
4. Note the sequence of operations:-

   Multiply, add, multiply, add and so on. The last operation is an addition.

Example: Convert $(1111)_2$ to decimal:

1 x 2 = 2	Multiply
2 + 1 = 3	Add
3 x 2 = 6	Multiply
6 + 1 = 7	Add
7 x 2 = 14	Multiply
14 + 1 = 15	Add

thus $(1111)_2 = (15)_{10}$

Example: Convert $(101011)_2$ to decimal:

1 x 2 = 2	Multiply
2 + 0 = 2	Add
2 x 2 = 4	Multiply
4 + 1 = 5	Add

5	x 2	=	10	Multiply
10	+ 0	=	10	Add
10	x 2	=	20	Multiply
20	+ 1	=	21	Add
21	x 2	=	42	Multiply
42	+ 1	=	43	Add

thus $(101011)_2 = (43)_{10}$

DECIMAL TO OCTAL CONVERSION

To convert a number from decimal to octal proceed as follows:

1. Divide the number by eight.
2. The first **remainder** (less than eight) is the least significant octal digit.
3. Divide the quotient by eight.
4. The next **remainder** is the next least significant octal digit.
5. Repeat these operations until a quotient of zero is reached.

Example: Convert $(129)_{10}$ to octal.

	0		
8	2	 2	3rd remainder
8	16	 0	2nd remainder
8	129	 1	1st remainder

Thus $(129)_{10} = (201)_8$

Example: Convert $(4273)_{10}$ to octal.

	0		
8	1	 1	5th remainder
8	8	 0	4th remainder
8	66	 2	3rd remainder
8	534	 6	2nd remainder
8	4273	 1	1st remainder

Thus $(4273)_{10} = (10261)_8$

OCTAL TO DECIMAL CONVERSION

To convert an octal number to decimal proceed as follows:

1. Multiply the most significant octal digit by eight.
2. Add the next octal digit to the result.
3. Multiply the result so far obtained by eight.
4. Add the next octal digit to the result.
5. Repeat the operations until the last octal digit has been added to the result.

Example: Convert $(201)_8$ to decimal.

$$\begin{aligned} 2 \times 8 &= 16 \\ 16 + 0 &= 16 \\ 16 \times 8 &= 128 \\ 128 + 1 &= 129 \end{aligned}$$

$$\text{Thus } (201)_8 = (129)_{10}$$

Example: Convert $(10261)_8$ to decimal.

$$\begin{aligned} 1 \times 8 &= 8 \\ 8 + 0 &= 8 \\ 8 \times 8 &= 64 \\ 64 + 2 &= 66 \\ 66 \times 8 &= 528 \\ 528 + 6 &= 534 \\ 534 \times 8 &= 4272 \\ 4272 + 1 &= 4273 \end{aligned}$$

$$\text{Thus } (10261)_8 = (4273)_{10}$$

RULES OF BINARY ARITHMETIC

The binary addition rules are:

$$\begin{aligned} 0 + 0 &= 0 \\ 1 + 0 &= 1 \\ 1 + 1 &= 0 \text{ and carry one} \end{aligned}$$

Examples:

1011 +		1101 +		10111 +	
1010		1000		11010	
10101	SUM	10101	SUM	110001	SUM
1 1	CARRIES	1	CARRIES	1111	CARRIES

For Binary Subtraction the rules are:

$1 - 0 = 1$
$1 - 1 = 0$
$0 - 1 = 1$ and borrow one

Examples:

1101 -		1001 -		10101 -	
0100		110		1010	
1001	Difference	0011	Difference	01011	Difference
	Borrow	11	Borrow	1 1	Borrow

Octal addition and subtraction is rather confusing at first because the range of digits is so very nearly the customary decimal range and the habits of years are hard to break.

For instance, $2 + 3 = 5$ in either octal or decimal, but $3 + 7 = (10)_{10}$ whereas $3 + 7 = (12)_8$.

The rules are:

1. Add two octal digits as though they were decimal.
2. If the result is less than eight accept it.
3. If the result is more than eight add two more.

Examples:

(a) $3 + 4 = 7$; less than 8 $\therefore$ $(7)_8$.
(b) $3 + 7 = 10$; more than 8 $\therefore$ $(12)_8$.
(c) $(5762)_8 + (2453)_8 = (10435)_8$.

When a carry is generated by the addition of the digits in any column, a one is added in the next column to the left.

$$\begin{array}{r} (2743651)_8 \\ (0177567)_8 \\ \hline (3143440)_8 \end{array} +$$

Subtraction is just the reverse. Provided a small digit is subtracted from a larger one there is no difficulty. If, however, a large digit is subtracted from a smaller one it is necessary to borrow one from the next higher position. Remember, however, that borrowing one from the next higher position means that eight is borrowed in the column under consideration.

Examples:

(a)
$$\begin{array}{r} (75641)_8 \\ (02321)_8 \\ \hline (73320)_8 \end{array} -$$

No borrowing is necessary.

(b)
$$\begin{array}{r} (23647)_8 \\ (17673)_8 \\ \hline (03754)_8 \end{array}$$

In this example, in the second column from the right 7 must be subtracted from 4. Borrowing one from the next column gives:

$$(8 + 4) - 7 = 12 - 7 = 5$$

The borrow is repaid in the usual way by adding one to the subtrahend in the next higher position.

One other point is worth emphasising in connection with octal numbers. All octal numbers ending in 7 are followed in the natural number sequence by a number ending in zero.

e.g. $(27)_8 + 1 = (30)_8$

$(77)_8 + 1 = (100)_8$

It will take some time before those newly introduced to octal cease to follow 47 by 48.

KDP 10 CHARACTER CODE

Most readers will have heard of the Morse Code or the teleprinter code in which numerals and letters of the alphabet are represented by dots and dashes or holes and absence of holes in paper tape. The KDP 10 code is very similar. The digits of the code are binary digits which may be ones or zeros in manuscript, holes or the absence of holes in paper tape, magnetic marks or the absence of such on magnetic tape or magnetic flux in one of two directions in magnetic cores.

In most situations about fifty-four characters are considered sufficient for present day communication. These are the ten decimal numerals, twenty-six letters of the alphabet and a suitable selection of punctuation marks and abbreviations. The number of patterns obtained from six binary digits is 64, which adequately covers the fifty-four communication characters and leaves some over for a special purpose which will be discussed later.

The KDP 10 Code showing the allocation of binary numbers to characters is given on page 2.11. This allocation is to some extent arbitrary, but certain rules must be observed. All the decimal digits must be in consecutive order and have binary values less than the letters of the alphabet. This makes sorting problems easier.

It will be seen from the code that the following patterns represent a few familiar characters (omitting the parity digit).

A	=	$(100\ 000)_2$	= $(40)_8$
G	=	$(100\ 110)_2$	= $(46)_8$
M	=	$(101\ 100)_2$	= $(54)_8$
X	=	$(110\ 111)_2$	= $(67)_8$
Zero	=	$(010\ 011)_2$	= $(23)_8$
One	=	$(010\ 100)_2$	= $(24)_8$
Two	=	$(010\ 101)_2$	= $(25)_8$

The reader should become familiar with the octal equivalent of each character from which the binary equivalent may be obtained quite simply.

KDP 10 CHARACTER CODE

OCTAL EQUIV.	CHARACTER DESCRIPTION	SYMBOL	7	6	5	4	3	2	1
				Channel Number					
			p	BIT VALUE 2^5	2^4	2^3	2^2	2^1	2^0
00	Blank		1	0	0	0	0	0	0
01	Space		0	0	0	0	0	0	1
02	Cross	‡	0	0	0	0	0	1	0
03	Open Parenthesis	(	1	0	0	0	0	1	1
04	Close Parenthesis	)	0	0	0	0	1	0	0
05	Quotes	"	1	0	0	0	1	0	1
06	Colon	:	1	0	0	0	1	1	0
07	Pound sterling	£	0	0	0	0	1	1	1
10	Per cent	%	0	0	0	1	0	0	0
11	Semicolon	;	1	0	0	1	0	0	1
12	Ampersand	&	1	0	0	1	0	1	0
13	Apostrophe	'	0	0	0	1	0	1	1
14	Minus	@	1	0	0	1	1	0	0
15	Asterisk	*	0	0	0	1	1	0	1
16	Full Stop	.	0	0	0	1	1	1	0
17	Carriage Shift (CS)		1	0	0	1	1	1	1
20	Page Change (PC)		0	0	1	0	0	0	0
21	Line Shift (LS		1	0	1	0	0	0	1
22	Stroke	/	1	0	1	0	0	1	0
23	Zero (Numeric)	0	0	0	1	0	0	1	1
24	One	1	1	0	1	0	1	0	0
25	Two	2	0	0	1	0	1	0	1
26	Three	3	0	0	1	0	1	1	0
27	Four	4	1	0	1	0	1	1	1
30	Five	5	1	0	1	1	0	0	0
31	Six	6	0	0	1	1	0	0	1
32	Seven	7	0	0	1	1	0	1	0
33	Eight	8	1	0	1	1	0	1	1
34	Nine	9	0	0	1	1	1	0	0
35	Comma	,	1	0	1	1	1	0	1
36	Number	#	1	0	1	1	1	1	0
37	Carriage Normal		0	0	1	1	1	1	1
40	A	A	0	1	0	0	0	0	0
41	B	B	1	1	0	0	0	0	1
42	C	C	1	1	0	0	0	1	0
43	D	D	0	1	0	0	0	1	1
44	E	E	1	1	0	0	1	0	0
45	F	F	0	1	0	0	1	0	1
46	G	G	0	1	0	0	1	1	0
47	H	H	1	1	0	0	1	1	1
50	I	I	1	1	0	1	0	0	0
51	J	J	0	1	0	1	0	0	1
52	K	K	0	1	0	1	0	1	0
53	L	L	1	1	0	1	0	1	1
54	M	M	0	1	0	1	1	0	0
55	N	N	1	1	0	1	1	0	1
56	O	O	1	1	0	1	1	1	0
57	P	P	0	1	0	1	1	1	1
60	Q	Q	1	1	1	0	0	0	0
61	R	R	0	1	1	0	0	0	1
62	S	S	0	1	1	0	0	1	0
63	T	T	1	1	1	0	0	1	1
64	U	U	0	1	1	0	1	0	0
65	V	V	1	1	1	0	1	0	1
66	W	W	1	1	1	0	1	1	0
67	X	X	0	1	1	0	1	1	1
70	Y	Y	0	1	1	1	0	0	0
71	Z	Z	1	1	1	1	0	0	1
72	End File (EF)		1	1	1	1	0	1	0
73	End Data (ED)		0	1	1	1	0	1	1
74	Item Separator (ISS)	•	1	1	1	1	1	0	0
75	End Message (EM)	>	0	1	1	1	1	0	1
76	Start Message (SM)	<	0	1	1	1	1	1	0
77			1	1	1	1	1	1	1

There are very good reasons for choosing the values $(23)_8$ $(24)_8$ etc., for the decimal digits. Some explanation of this choice is given in the discussion of the excess-three binary code on page 2.13.

The extra digit on the left of each group is a PARITY BIT. It may take on the value one or zero depending on the number of ones in the six information bits.

If the total number of ones in the information bits is EVEN, the PARITY BIT is ONE otherwise it is zero. The total number of ones in a complete character, including the parity bit, is, therefore, odd and the machine inspects characters at various stages of computing processes to check that a valid parity condition exists. The computer will stop if a parity error is detected.

On paper tape an EVEN parity system is used. The total number of ones in a character is an even number. This method is adopted to permit blank tape to qualify as an acceptable character.

Following the discussion of binary numbers and the KDP 10 Code it is now possible to indicate the different methods of visualising information in the KDP 10. The High Speed Memory of the system consists of character storage locations. Each location is individually addressable and has a unique number associated with it known as its address. Storage locations are rather like small pigeon-holes in a large array of shelves, each one capable of holding a letter of the alphabet, a numeric digit, a punctuation mark or some other special symbol.

A five digit decimal number may be stored in five consecutive locations and the name JOHN-SMITH requires ten storage positions.

Two examples are now given illustrating the different ways of visualising information in KDP 10.

Example: The decimal number 7513 is stored in the H.S.M. as follows:

BINARY

P	$2^5 2^4 2^3 2^2 2^1 2^0$	P	$2^5 2^4 2^3 2^2 2^1 2^0$	P	$2^5 2^4 2^3 2^2 2^1 2^0$	P	$2^5 2^4 2^3 2^2 2^1 2^0$
0	0 1 1 0 1 0	1	0 1 1 0 0 0	1	0 1 0 1 0 0	0	0 1 0 1 1 0

The programmer thinks of the information in the normal way as:

DECIMAL

7	5	1	3

There are occasions, however, when it is necessary to consider the KDP 10 representation and when this occurs the programmer should use the OCTAL EQUIVALENT values of each character, i.e.

OCTAL

32	30	24	26

Example: The letters PAYE are contained in four consecutive storage locations in:

BINARY	P	$2^5 2^4 2^3 2^2 2^1 2^0$	P	$2^5 2^4 2^3 2^2 2^1 2^0$	P	$2^5 2^4 2^3 2^2 2^1 2^0$	P	$2^5 2^4 2^3 2^2 2^1 2^0$
	0	1 0 1 1 1 1	0	1 0 0 0 0 0	0	1 1 1 0 0 0	1	1 0 0 1 0 0

The same information is thought of quite normally as:

ALPHA	P	A	Y	E

or, when necessary as:

OCTAL	57	40	70	44

KDP 10 instructions are composed of OCTAL numbers. They are visualised in the machine exactly as they are written on coding sheets.

Example: The instruction 51 072746 00 024360 is stored in eight character locations and should be thought of as shown below:

51	07	27	46	00	02	43	60

It should never be thought of as:

J	£	4	G	Blank	‡	D	Q

EXCESS-3 BINARY CODE

The following table lists the binary equivalents, excess-3 binary equivalents and the KDP 10 character representation of the natural numbers.

DECIMAL	BINARY	EXCESS-3	KDP 10 CODE	OCTAL
0	0000	0011	01 0011	$(23)_8$
1	0001	0100	01 0100	$(24)_8$
2	0010	0101	01 0101	$(25)_8$
3	0011	0110	01 0110	$(26)_8$
4	0100	0111	01 0111	$(27)_8$
5	0101	1000	01 1000	$(30)_8$
6	0110	1001	01 1001	$(31)_8$
7	0111	1010	01 1010	$(32)_8$
8	1000	1011	01 1011	$(33)_8$
9	1001	1100	01 1100	$(34)_8$

The numbers in the Excess-3 column are simply the normal binary equivalents with 3 added in each case. The KDP 10 Codes for the decimal numbers are shown with the top two bits separated from the bottom four bits to emphasise that the bottom four bits are the Excess-3 coded representation of the decimal numbers in each case. When two numbers are added, the computer considers only the four least significant bits of each number. The result of the operation is reconverted to Excess-3 code and the standard top two bits (01) are tacked on to produce the correct six-bit character representation of the decimal sum.

The Excess-3 code is used because it has two advantages in decimal arithmetic using binary coded characters:

1. The nines complement of a number can be obtained easily by changing ones to zeros and vice versa.

2. A carry is propagated with two Excess-3 operands whenever a carry would be propagated with the same decimal operands.

This note on Excess-3 binary coding has been included in part explanation of the choice of KDP 10 codes for the natural numbers. Knowledge of the Excess-3 code is not necessary in KDP 10 programming and it will not be mentioned again.

KDP 10 CONTROL CHARACTERS

Five code combinations have been allocated to special characters which are unique to the KDP 10 SYSTEM. These are:

$(72)_8$	=	EF,	END FILE
$(73)_8$	=	ED,	END DATA
$(74)_8$	=	ISS •,	ITEM SEPARATOR SYMBOL
$(75)_8$	=	EM >,	END MESSAGE
$(76)_8$	=	SM <,	START MESSAGE

The above symbols are KDP 10 Control characters which are used to mark the start and finish of magnetic tape files, tape reels, messages and items. The exact use of these characters is made clear in the formal definitions of the different data groups in the next section.

One character which deserves special mention is (01) - space. Space symbols serve as terminal characters in certain operations and are treated like item separator symbols. They are also used as positive signs in decimal arithmetic and they also serve as 'erase' symbols. If any part of the HSM is cleared by the use of the instructions provided, space symbols are placed in each character location of the area nominated.

ORGANISATION OF DATA ON TAPE

GENERAL. All information enters the KDP 10 from tape, either magnetic tape or paper tape. It is, therefore, necessary to describe the form in which this information is maintained and to give some idea of how the programmer visualizes it.

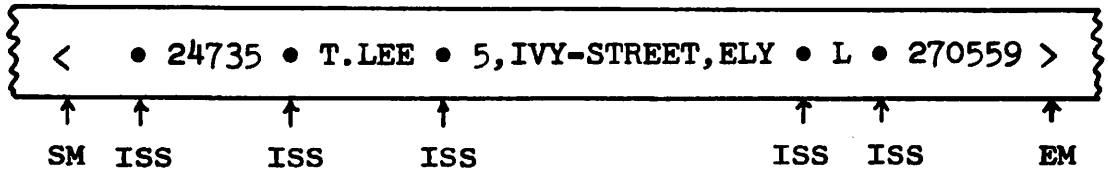
As an example, suppose the following related information refers to an insurance policy:

POLICY NUMBER	24735
NAME	T. LEE
ADDRESS	5, IVY STREET, ELY
TYPE OF POLICY	LIFE (L)
DATE	27TH MAY 1959

Using the code letter (L) for a life policy and abbreviating the date to 270559 this might appear on tape as:-

24735T.LEE5IVYSTREETELYL270559

in which each of the characters in the group is recorded as the appropriate KDP 10 character across the tape. As it stands the information is not very useful. The complete group represents a message containing five items of information with no indication of where one item finishes and the next starts. This is where KDP 10 Control characters prove useful. The same information is given below and includes KDP 10 Control characters and space(-) and punctuation marks:-



DEFINITIONS

BIT. A bit is a single binary digit, having a value of either 0 or 1.

CHARACTER. A KDP 10 character consists of six information bits and one parity bit combined to represent a decimal digit, a letter of the alphabet, a punctuation or other special mark, or a control symbol (see The KDP 10 Code).

ITEM. An item consists of such characters as are necessary to specify a particular unit of information (a numerical quantity, an alphabetic name, a street address, a stock number, etc.). An item is preceded by an Item Separator symbol (ISS).

MESSAGE. A message consists of a Start Message symbol (SM); one or more related items, each preceded by an Item Separator symbol; and an End Message symbol (EM), in that order.

BLOCK. On magnetic tape, a block consists of eight or more characters, preceded and followed by an Inter-message Gap. Intra-block blanks must constitute less than 75 microseconds of tape time.

On paper tape, a block consists of an even number of characters equal to or greater than sixteen, without intra-block blanks; it is preceded and followed by an Intermessage Gap. (Corrective overpunching, to delete a character, is ignored in this character count). The characters must

represent only the decimal digits 0 through 7 (octal 23 through 32). (See discussion of decoding circuitry, Paper Tape Reader, page 4.2).

Blocks on magnetic or paper tape are read and written without regard to message structure rules - they are delineated by Gaps, rather than by control symbols, and need not contain any control symbols.

LINE. A line is composed of the characters from a single message which may be read from magnetic tape into the Electro-Mechanical Printer during a single read cycle. Each such line is terminated with an End Message (EM) or Line Shift (LS) symbol. The LS symbol appears only as the last character of a line. When a Page Change (PC) symbol or an Item Separator symbol is used to provide an additional paper control symbol, it must always be followed by LS or EM and then the Intermessage Gap. Also, either EM or LS preceded and followed by an Intermessage Gap is a line and must be treated as such on tape. (In the Electro-Mechanical Printer, the maximum number of characters that can actually be printed is 120 per line. Each character space to appear in the printed line decreases the 120 maximum by one. On tape, a line may include non-print characters and control symbols in addition to the 120).

FILE. A file consists of any number of related information units, in message or block format; it may consist of several tapes (reels) or any part of one tape. A file is terminated by an End File (EF) symbol, preceded and followed by an Intermessage Gap.

In a multi-tape file, all but the last tape are terminated by an End Date (ED) symbol alone, preceded and followed by an Intermessage Gap. The end of file on the last tape is indicated by an EF, preceded and followed by an Intermessage Gap. If this is a full tape, or a partially filled tape with no other valid information following the file data, an ED follows the EF, separated from it and followed by an Intermessage Gap.

If more than one file appears on a tape, each file except the last is terminated by an EF only. The last complete file on a tape is terminated by an EF followed by an ED.

ED and EF are each preceded and followed by an Intermessage Gap. They are never accompanied by Start and End Message symbols, and they may never appear within a message.

If a file is read or written in block format, the ED and the EF when standing alone are treated as separate blocks. If a file is read or written in message format, the ED and the EF are treated as separate messages.

INTERMESSAGE GAP. An Intermessage Gap is a length of unrecorded tape sufficient to allow for Gap detection and stopping and starting of the tape. This is a minimum of 0.34" on magnetic tape (for block or message format). On paper tape, it is a minimum of seven character positions for message format and a minimum of eight character positions for block format.

ARRANGEMENT OF DATA ON TAPE

ARRANGEMENT OF BITS TO FORM CHARACTERS. Figure 3.1 illustrates the arrangement of information on paper tape. There are seven data hole positions per row. Each row represents a character. The presence of a punched hole represents a 'one' bit, and the absence of a hole represents a 'zero' bit. The positions of the bits are numbered 2^0 through 2^6, corresponding to the information and parity bit positions of a KDP 10 binary coded character. A row with all seven channels (plus an extra, eighth, channel) punched indicates that the character in this row has been deleted; such punching does not represent a KDP 10 character (see The KDP 10 Code).

Bits are recorded on magnetic tape as magnetic spots in rows across the tape (Figure 3.2). Each bit is written in two locations as an accuracy control measure, giving 16 tracks across the

width of the tape (including the timing bit). Each of the two locations is capable of producing a standard signal. Thus, either one of the two recorded spots for a single bit of information may be missing, and the bit can still be read.

ARRANGEMENT OF CHARACTERS TO FORM ITEMS. All characters are recorded on tape serially so that the characters making up an item follow one another in sequence from most to least significant. Each item is preceded by an Item Separator symbol.

Each item of a message may have variable length. Inclusion of the Item Separator symbols allows the use of variable length items and the omission of items, without changing the positional significance of any item in the message.

ARRANGEMENT OF ITEMS TO FORM MESSAGES. The items of a message follow one another in sequence, each being preceded by an Item Separator symbol. In every message of a given type, the nth item always has a given connotation. Therefore, a count of the Item Separator symbols, starting from the first or from a programme-oriented point in a message, permits location and identification of any item.

If a particular item is omitted, the Item Separator symbol can be recorded on tape in its proper sequence when it is necessary to preserve the positional significance of the items that follow. However, the Item Separator symbol for an omitted item may also be omitted if there is no valid information following it in the message. In this case, the End Message symbol follows immediately after the last item present. This avoids writing an unnecessary number of consecutive Item Separator symbols at the end of a message.

All messages consist of a Start Message symbol followed by an Item Separator symbol and the characters of the first item, the succeeding items (each preceded by an Item Separator symbol), and an End Message symbol, in that order. The SM and EM symbols appear only in the positions defined herein.

MISCELLANEOUS

Equipments that record data on magnetic tape provide for erasing a minimum length of 1.35" at the beginning of each tape before recording. The equipment recognises the beginning of the tape by a Beginning of Tape Level which is generated by a permanent indicator (Beginning of Tape Control BTC) in a fixed position on the magnetic tape (not a KDP 10 character).

An end of tape warning (ETW) device is provided to indicate that approximately 5 feet of usable magnetic tape are available; this permits recording of an entire message and an End File and/or End Data symbol after the warning is received. This warning is not a KDP 10 character, but is a signal generated by a permanent indicator in a fixed position on the tape. Beyond the ETW marker there exists a device for marking the Physical End of Tape (PET). This is a permanent indicator in a fixed position on tape.

VARIABLE ITEM AND MESSAGE LENGTH

Data storage in the KDP 10 System incorporates TRUE VARIABLE item length. This concept may be more fully appreciated if prefaced with a discussion of FIXED and FIXED VARIABLE word and block length.

'Word' is generally defined as a fixed number of consecutive characters or character locations, and 'block' as a fixed number of consecutive words, in primary or secondary storage. These terms, word and block, are more aptly used with respect to FIXED and FIXED VARIABLE systems, but are not particularly applicable to a TRUE VARIABLE system.

In a computer system employing fixed word length, the number of characters per word and the number of words per block are characteristic of the system, incorporated in the circuitry. A computer with a fixed word length of 12 characters dictates the use of some multiple (not fraction) of 12 for each and every item (employee number, name, pay rate, etc.), in a message and a fixed number of words for each message in a file. If the employee number is made up of five digits, the word in which this item was stored would be filled out with redundant zeros or spaces (e.g., +64398000000). This would be true for each employee number in the file. The alternative of utilising these zero-filled positions by packing more than one item into a word entails additional programme instructions, with consequent increased processing time, and is possible only to a limited degree. Even with packing of words, the fixed block size may entail zero-filling of one or more entire words at the end of each block.

In a fixed variable (non-standard maximum item length) system, the number of character positions for each item is assigned in accordance with the anticipated maximum for that item. In such a system, then, item (data field) layout is analogous to that used for punched cards. These lengths may be individually predetermined for each file, but remain constant for each message in the file. For example, in an inventory file, cost per unit might vary from one penny for one stock number to some five-digit figure for another. This maximum would dictate that five character positions be used even where there is only one significant digit; one penny might then be written as 00001. Stock numbers in this file, however, could be assigned a different number of character positions, stock description still a different number, and so on. Thus, fixed variable word length provides greater flexibility than does fixed word length.

Data storage in a TRUE VARIABLE item length system does not have the limitations imposed by fixed or fixed variable systems. In the KDP 10, the use of control symbols and the ability to address each character location individually permits the length of any item in any message to be in strict accordance with that item's actual character count. This allows for total variability of item and message length, but does not preclude the use of fixed or fixed variable lengths when the programmer finds this expedient.

In each of these categories - fixed, fixed variable, and true variable - the method of internal storage is extended to external storage; when redundant zeros or space characters are required to fill out a word in the computer memory, they are also carried on tape. With a given tape density (number of characters packed per inch) and a given tape speed (number of inches per second), a characteristic business file would require less tape footage and could be read and written out in less time when true variable item length is utilised.

DESCRIPTION OF ON-LINE EQUIPMENT

PAPER TAPE READER

The Paper Tape Reader is used to transcribe incoming data to magnetic tape via the Computer. It is used for 'first-time' input of all forms of data, other than that transcribed from punched cards direct on to magnetic tape, for the insertion of control information such as parameters and dates and, most important of all, programme coding information.

The last requirement, programme insertion, is the reason for a special facility provided on the Paper Tape Reader known as Block Read Decoding. A description of the process is given later in the section and the reason for the decoding process will now be explained.

If, for example, today's date, is required in the Computer it will be read in from paper-tape in Message Format. July 12th 1960 would be punched on tape and appear as:

< JULY-12-1960 >

being a total of fourteen characters. When the information is in the HSM it will appear in the same form with one character in memory for one character on paper tape.

Programme instructions in the KDP 10 present a slight problem. As will be shown in the next section one instruction occupies eight character positions but the instructions are written in OCTAL notation and require sixteen figures in all.

As an example, the instruction:

22 013047 40 243562

occupies eight successive character positions and might be written as:

(22) (01) (30) (47) (40) (24) (35) (62)

or, using the KDP 10 code to transpose into single characters,

/ - 5 H A 1 , S.

If these characters are punched on paper tape they would appear in the HSM as the instruction shown above. The effort required to do this would be quite considerable and this is where Block Read Decoding is used.

The instruction is punched as sixteen decimal characters and each pair of decimal characters are brought together automatically to form the appropriate octal pair.

The Paper Tape Reader is photoelectric and operates at the rate of 1000 characters per second. It uses one-inch wide, seven-hole punched tape; the seven channels across the width of the tape correspond to the seven bit positions (six information bits and one parity bit) of a KDP 10 character. Characters on punched paper tape have even parity. (See Figure 3.1). When a character position on paper tape contains a punch in all seven channels, plus an eighth punch, this is interpreted as 'delete character'; it is not a KDP 10 character and no attempt is made to read it into the Computer. Information on paper tape may be in either message or block format.

Since the tape does not stop immediately after the last character is read in, but may glide the equivalent of three character positions, a Gap (unpunched tape) of seven character positions is left between successive messages, and eight character positions between successive blocks.

As explained above, the insertion of programmes calls for a special method when block reading from paper tape. Programmes are written, on the code sheet, in octal notation. A KDP 10 tapewriter is used to create a punched paper tape from the code sheet, so that the programme can be read into the Computer. One decimal key is depressed for every octal digit on the code sheet, thereby producing on the tape one character (two octal digits) for every octal digit on the code sheet.

In a block read from paper tape (unless the Block Read by-pass button on the Console is depressed), characters automatically enter decoding circuitry before they are transferred into the HSM unless the Block Read by-pass button on the console is depressed. Decoding may be explained as follows: $(23)_8$ is subtracted (binary subtraction) from the first character on tape, and the rightmost three bits of the difference are stored as the left-most three bits of the decoded character; $(23)_8$ is subtracted from the second character on tape, and the rightmost three bits of the difference are stored as the rightmost three bits of that decoded character. The combined result, with a parity bit generated, is then stored in the High-Speed Memory in the same fashion as are characters read from magnetic tape. The process is then repeated for each character on the paper tape until a Gap is recognised. On a block read from paper tape, then, the leftmost three bits (left octal digit) of each character stored in the HSM will be derived from the first, third, fifth, seventh, etc., character read in from the tape; and the rightmost three bits (right octal digit) will be derived from the second, fourth, sixth, eighth, etc., character read in from the tape. When the last character on tape has entered the decoding circuitry, an alarm occurs if the count is less than sixteen, (CIG alarm) or an odd number (OPF alarm).

This process may be represented as follows:

Instruction characters as they appear on code sheet:

			0			A			N			B				
			05			014103			00			600000				
Octal digits punched on paper tape	23	30	23	24	27	24	23	26	23	23	31	23	23	23	23	23
Automatically subtracted in decoding circuitry	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23
	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Difference	00	05	00	01	04	01	00	03	00	00	06	00	00	00	00	00
Decoded characters stored in HSM		05		01		41		03		00		60		00		00

This decoding occurs only when paper tape is read in block format permitting the digits 0 through 7 (octal 23 through 32) to be properly decoded. Block format on paper tape is largely restricted to programmes for initial insertion into the Computer. Data (as against programme instructions) on paper tape are punched and read in message format, in standard KDP 10 Code, and are not subjected to code conversion.

Data in Block Format may be read-in from Paper Tape without code conversion if a Block Read by-pass button on the P.T.R. is depressed.

The following formula is used for paper tape read time:

Total No. of char. ÷ 1000 = total time in seconds

Start time is negligible for paper tape and is not considered a time factor.

MONITOR PRINTER

The Monitor Printer associated with the Computer Console prints out information from the High-Speed Memory of the Computer under the direction of the stored programme at the rate of ten characters per second. Two modes of operation are possible:

1. IN COMPUTER OR PROGRAMME TESTING, a character will be printed for every character that might be stored in the designated area of the High-Speed Memory. Carriage return and a line shift automatically occur after print-out of an End Message symbol or when the right-hand paper margin is reached.

The table below shows the character that will be printed for each octal number in the High-Speed Memory e.g., a space symbol, $(01)_8$, in the HSM will be printed out as an underline, $(23)_8$ will be printed out as a decimal zero, $(24)_8$ as a decimal one, etc.

1st digit of octal no. \ 2nd digit of octal no.	0	1	2	3	4	5	6	7
0	a	Under-line	‡	(	)	"	:	£
1	%	;	&	'	minus sign	*	.	h
2	q	r	/	0	1	2	3	4
3	5	6	7	8	9	,	#	x
4	A	B	C	D	E	F	G	H
5	I	J	K	L	M	N	O	P
6	Q	R	S	T	U	V	W	X
7	Y	Z	s	t	•	>	<	v

2. FOR EDITED OUTPUT, printing of space symbols, Item Separator symbols and End Message symbols can be suppressed. The End Message symbol, though suppressed, will still effect a carriage return and line shift. A suppressed space symbol will cause a horizontal shift of one character position. The Monitor Printer can thus be used for output of documents (summary totals, cost distributions, etc.), utilising either pre-printed or blank paper stock. Associated with the Monitor Printer is a Paper Tape Punch (see next page). Monitor Printer output time (with or without paper tape output) may be computed on the basis of the number of characters involved (total no. of char. ÷ 10 = total time in seconds). Start time is negligible and is not considered a time factor.

The PAPER TAPE PUNCH associated with the Monitor Printer is used for manual preparation of short paper tapes and for output of data from the Computer. Punching is activated by flipping the Punch Switch on the Monitor Printer to the ON position, in which case both punched tape and printed copy on the Monitor Printer are obtained. This mode of operation is subject to the restriction that printing of a lower-case letter (see Monitor Printer chart above) will cause the octal value of the corresponding upper-case letter to be punched in the tape. For example, if the character $(00)_8$ is read out of the HSM, the lower-case letter 'a' will be printed (Monitor Printer) and $(40)_8$, the octal value of an upper-case 'A', will be punched on tape. Of course, if $(40)_8$ is read out of the HSM, an upper-case 'A' will be printed and $(40)_8$ will be punched on tape.

For punching at higher speeds, either of two additional paper tape punches may be electrically connected to Computer trunk $(76)_8$. The COMPUTER PAPER TAPE PUNCH punches tape at the rate of 60 characters per second. The COMPUTER Hi-SPEED PAPER TAPE PUNCH punches tape at the rate of 300 characters per second. Each is available with the option of five-level or seven-level tape.

ON-LINE PRINTER

The On-Line Printer in the KDP 10 System is an all-transistor device which prepares output documents, printing data directly from the High-Speed Memory of the Computer. Data editing is accomplished in the Computer, under the direction of the stored programme. Line skipping is controlled by the Computer programme, either directly or through a Tape Loop on the Printer Unit.

Two Computer instructions are directly associated with the On-Line Printer - one initiates printing and the other positions the paper for the next line of printing. The latter instruction may specify the number of lines the paper is to be advanced, or it may refer to one of two information channels in the Tape Loop on the Printer Unit. One channel (Vertical Tabulation) is referenced in order to advance the paper to specific lines within the confines of a page, and the other channel (Page Change) is referenced to move the paper to the start of a new page.

Maximum print capacities are 120 characters per line, 10 characters per horizontal inch, and 6 lines per vertical inch. One line is printed in 66.7 milliseconds. For single-line paper advance, the paper motion time is 30 milliseconds. Printing and paper motion time, then, total less than 100 milliseconds, permitting single-spaced printing at the rate of at least ten lines per second (600 lines per minute). When more than three lines are skipped at one time, however, the paper advance rate is at least 70 lines per second.

Paper stock may be single or multiple sheet fanfold, from 3 to 22 inches in width and up to 17 inches in sheet length. One original plus up to three carbons (11-pound paper and 7½-pound carbon) may be used. Multilith master stock may also be used.

Fifty-one characters can be printed by the On-Line Printer. These include the 10 decimal digits, the 26 letters of the alphabet, and the following punctuation marks and symbols:

,	comma	*	asterisk
;	semicolon	&	ampersand
:	colon	/	stroke
.	full stop	%	per cent
'	apostrophe	£	pound sterling
(	open parenthesis	#	number sign
)	close parenthesis	-	minus sign
"	ditto or quotes		

The occurrence in an HSM print-out sector of any character other than one of the fifty-one listed above will leave a blank in the related position in the printed line, with the exception that $(00)_8$ will result in an overprinting of the three symbols =, + and •. (These symbols, though present on the print wheel, normally have no KDP 10 equivalents for printing purposes).

For accuracy control, a Printer Unit Inoperable alarm, which stops both the Printer and the Computer, is incorporated. Also, the printer, by means of a micro-switch, can sense a 'low paper' condition and send a warning signal to the Computer. When the Computer programme calls for Tape Loop activation of a page change and a 'low paper' signal is present, both the Computer and the Printer stop, after accomplishment of the page change, to permit replenishment of the paper supply. Thus, printing on a page is completed before the operation is stopped.

MAGNETIC TAPE STATION

The Magnetic Tape Station is a fully automatic device with transistor circuitry, which performs reading, writing and erasing operations on ½-inch wide Mylar base magnetic tape, under control of the user equipment. On each Tape Station, two 10½-inch reels are mounted - a full reel and a take-up reel. The capacity of a reel is 2400 feet of tape, providing a minimum of 2275 feet of usable tape. Tape Station design facilitates manual interchange of reels, which can be accomplished in less than one minute.

The Tape Station can be instructed to move the tape in a forward or reverse direction. It can be directed to move the tape with or without writing. It can read the tape with or without transferring characters into the High-Speed Memory. It can be instructed to read all of the data serially or search for specified symbols. The Tape Station can, in response to one instruction, unwind the tape to the end, or rewind it to the beginning. Writing on magnetic tape is in the form of significant configurations of magnetic spots (see Organisation of Data on Tape and Figure 3.2).

The Tape Station writes at the rate of up to 33,333 characters per second (33.3 KC). It accomplishes this by writing 333.3 characters to the inch while moving the tape at a speed of 100 inches per second. Information on magnetic tape is read at this same rate. It takes only 30 microseconds to read a character on tape into the High-Speed Memory, or to write a character from the memory on to magnetic tape.

In reading information from magnetic tapes, in order to ensure that the tape is started and stopped with the read-write head positioned within a gap, a minimum gap of 0.34 inch must appear between messages or blocks.

In the execution of each write instruction, a 3.5 millisecond delay is introduced between the start of tape movement and the writing of the first character. A 1 millisecond delay is introduced between the writing of the last character of the block or message and the issuance of the tape stop command. Therefore, when one write instruction follows another write instruction to the same tape station, a 4.5 millisecond lapse during which tape is moving occurs between writing the last character of a block or message and writing the first character of the next block or message. This 4.5 millisecond lapse is the minimum time necessary to create the proper gap size.

In the execution of a read instruction no delay is introduced between reading the last character and the issuance of the tape stop command. Therefore, if a write instruction is directed to a tape station which previously executed a read instruction, only the 3.5 millisecond delay introduced at the commencement of the write instruction effectively generates a gap, and this gap would be less than the minimum allowable gap size. To provide the programmer with the ability to request a write after read combination, all write instructions may be written with a special indication that the write after read condition exists. This indication will cause a 4.5 millisecond delay to occur between tape start and write-out of the first character, thereby generating the required gap.

Up to 62 Tape Stations can be directly addressed by the Computer. Eight Tape Stations can be connected to Tape Switching and Buffer Unit-A. As additional Tape Stations are required in an installation, a Tape Switching Unit-B can be substituted for each of the original eight Tape Stations. The addition of one such unit will, therefore, permit as many as 15 Tape Stations to be connected (seven original and eight added) to the Computer. One A Unit and eight B Units are required to connect sixty-two Tape Stations to the Computer. Each Tape Station has a unique (octal) address, $(00)_8$ - $(75)_8$. The 64th address, $(77)_8$, is reserved for the Monitor Printer and the Paper Tape Punch and for the Paper Tape Reader, and the 63rd address $(76)_8$ is reserved for the on-line high speed Paper Tape Punch.

The first eight Tape Stations (attached to Unit-A) are Computer identified by the left digit of the tape selection number (00, 10, 20, 30 ... 70). With only these eight Tape Stations in a system, then, any number from 10 to 17 will select Tape Station 10; 30 to 37 will select Tape Station 30, etc. Each Tape Station connected to a B-type unit must be addressed by its individual Tape Station number. For instance, if Tape Stations are connected to 30, 31, 34, 35 and 37 of the 30-37 octet, and 32, 33 or 36 is addressed, a 'non-operable' alarm will occur and the Computer will stop.

Tape Station accuracy controls include manual lockout and 'write interlock'. Manual lockout is provided on each Tape Station to ensure safe manual operation procedures. When the Lockout Switch of a Tape Station is in the ON position nothing can be written on the tape at that Station. A 'write interlock' feature safeguards reference tapes from human error by preventing writing (and erasing).

THE COMPUTER

HIGH SPEED MEMORY

The High Speed Memory (HSM) of the KDP 10 System is constructed in modules. Each module is made up of twenty-eight plane matrices of magnetic cores, with 64 cores in each side of the plane. Each matrix, therefore, contains 64 x 64 = 4096 cores and a complete module contains 4096 x 28 = 114,688 cores. Each core in the array is capable of storing one bit (binary digit) from which it may be seen that one module can store $\frac{114,688}{7}$ = 16,384 characters in as many individually addressable locations.

The HSM is expandable from one to sixteen modules (four banks, each with a capacity of 65,536 characters), or up to a maximum of 262,144 character locations. Each location in memory has a unique address, consisting of three characters (= three octal numbers = six octal digits), so that each location, or the contents thereof, is individually addressable. Though somewhat over-simplified, the HSM may be pictured as a rectangular array of locations, with the smallest address in the upper left-hand corner and the largest address in the lower right-hand corner. The lowest address in the HSM is always $(000000)_8$. Since the addressing scheme in the KDP 10 employs the octal number system, the highest address in a one-module memory is $(037777)_8$ and the highest address in a sixteen-module memory is $(777777)_8$.

The lowest address in the HSM is at the MOST SIGNIFICANT end of the memory and the highest address is at the LEAST SIGNIFICANT end of the memory. These terms come about through consideration of the order in which decimal digits are stored in the machine with the most significant digit at the left-hand (lowest address) end.

To decrease processing time, the HSM is constructed to provide access to four characters (twenty-eight bits) occupying four consecutive character locations in a single memory cycle. The Computer memory cycle is 15 micro-seconds; this means that characters may be addressed, brought into the Memory Register, and regenerated in their original locations every 15 microseconds.

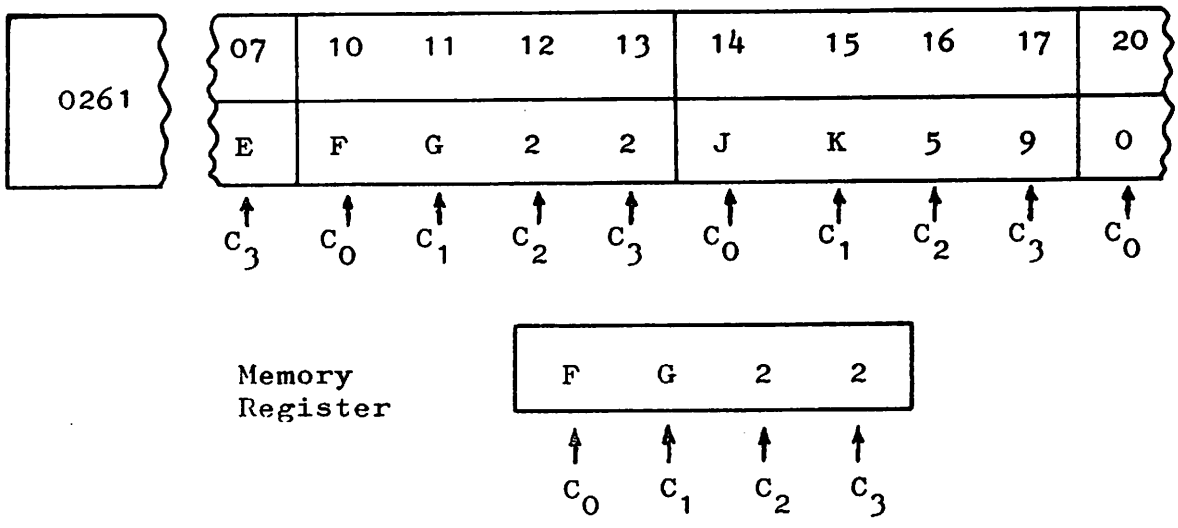
Each of these groups of four locations, or the contents thereof, is called a tetrad. A tetrad begins in a location addressed $((-----0)_8$ or $(-----4)_8$ and ends in a location addressed $(-----3)_8$ or $(-----7)_8$, respectively. In diagrammatic representations of portions of the HSM throughout this manual, tetrads are delineated by heavy vertical lines, as shown below.

	06	07	10	11	12	13	14	15	16	17	20	21	22
0261	D	E	F	G	2	2	J	K	5	9	0	3	X

The HSM address of each location shown in the above diagram consists of two octal digits in the upper portion preceded by the four digits at the left (026106, 026107, etc.). The characters stored in these locations are shown in the lower portion of the diagram.

As stated previously, each character location has a unique address. A tetrad address, however, may be the address of any one of the four locations comprising that tetrad. No matter which one of the four is addressed, the contents of the entire tetrad will be brought into the Memory Register. Depending on the instruction being executed, since some KDP 10 instructions deal with characters singly, and some by tetrads - all four characters in the tetrad, the rightmost three, or only the character from the specified location will be processed.

For convenience in referring to the individual locations or characters within a tetrad, when the specific address is not pertinent, the symbols C_0, C_1, C_2 and C_3 are used. These symbols apply to any tetrad in the HSM and to characters in the Memory Register.



The primary purpose of the HSM is the storage of programmes and data. These may be stored in any area of the memory, except that the first 164 locations (000000 - 000243) are reserved as Standard HSM Locations. This reserved area stores data used for address modification, special accuracy routines, and data for which special counters would otherwise be required.

The table provided gives some idea of the size of memory in terms of the maximum octal address used and the instruction storage capacity. Included in the Table for comparison with other computers which define storage capacity in 'words', is a list showing the equivalent 28-bit word store for each module.

No. of Modules	No. of Characters	No. of Instructions	No. of 28-bit words	Highest Octal Address
1	16,384	2,048	4,096	037777
2	32,768	4,096	8,192	077777
3	49,152	6,144	12,288	137777
4	65,536	8,192	16,384	177777
5	81,920	10,240	20,480	237777
6	98,304	12,288	24,576	277777
7	114,688	14,336	28,672	337777
8	131,072	16,384	32,768	377777
9	147,456	18,432	36,864	437777
10	163,840	20,480	40,960	477777
11	180,224	22,528	45,056	537777
12	196,608	24,576	49,152	577777
13	212,992	26,624	53,248	637777
14	229,376	28,672	57,344	677777
15	245,760	30,720	61,440	737777
16	262,144	32,768	65,536	777777

THE BASIC INSTRUCTION

INSTRUCTION FORMAT

Each of the forty-nine KDP 10 instructions consists of four parts, in the following order:-

1. Operation Code (a one-character code for add, subtract, item transfer, etc.)
2. A Address (three-character HSM address of the augend, minuend, original location of an item, left boundary of a sector, etc.)
3. N Character (one-character code, that can call for automatic modification of the A and/or the B address; the N character is explained in greater detail under AUTOMATIC ADDRESS MODIFICATION)
4. B Address (three-character HSM address of the addend, subtrahend, destination location for an item, right boundary of a sector, etc.)

An instruction, then, is made up of eight characters with the format OAAANBBB.

O	AAA	N	BBB
Operation Code	A address	N character	B address

In most of the instructions, the entire A address refers to a High-Speed Memory location (or tetrad) and the entire B address refers to another High-Speed Memory location (or tetrad). In these cases, the components of the A address (AAA) or the B address (BBB) need not be differentiated. In some instructions, however, only a portion of the A address or of the B address is used, or one component may designate one value and the other component another value; for example, one part of the A address may be used to specify a symbol and the other part to designate a count, or one part of the B address denotes the Tape Station and the other part is ignored. In these instructions, the components of the A address are referred to as A_1, A_2 and A_3, and the components of the B address as B_1, B_2 and B_3. This is illustrated below under CODED INSTRUCTION.

Since each location in the High-Speed Memory is individually addressable, whenever instructions do not utilize the entire capacity of the A or B address, it is usually feasible for the programmer to employ the unused (ignored) portion of the address for the storage of constants.

CODED INSTRUCTION

Instructions are coded in octal notation. Since the octal equivalent of a character consists of two octal digits, a coded instruction will contain sixteen octal digits.

Example of an instruction as coded by the programmer: 16 720003 10 400000

16	720003	10	400000
	$A_1A_2A_3$		$B_1B_2B_3$
Operation Code	A address	N character	B address

STORAGE OF INSTRUCTIONS

Instructions are stored sequentially in the High-Speed Memory. Each instruction is stored in two successive HSM tetrads (eight locations) so that the Operation Code falls in the leftmost (C_0) location of the first tetrad, with the A address in the remaining three locations (C_1, C_2 and C_3). The N character and the B address are in the same relative positions in the second tetrad - N in C_0 and B in C_1, C_2 and C_3.

STATICIZING

An Instruction can be interpreted and executed by Programme Control only after it is brought out of the High-Speed Memory locations in which it has been stored, and its components placed in the proper registers. This process is called **staticizing** and is accomplished in two status levels.

A **status level** lasts 15 microseconds and is the term applied to a series of pulses which open certain paths over which information can travel. A status level that opens paths leading to or from the High-Speed Memory is called a **memory cycle.** (**Status level** and **memory cycle** are not synonymous, since not all status levels are concerned with opening the paths leading to or from the High-Speed Memory). Each status level has a specific function. In staticizing each instruction, the first status level brings the tetrad OAAA into the Memory Register, from which O is sent to the Normal Operation (NO) Register and AAA to the A Register. The second status level brings the tetrad NBBB into the Memory Register, from which N is sent to the N Register and BBB to the B register.

Thus, staticizing time of 30 microseconds is constant for every instruction. The number of status levels involved, and their sequence, for execution (after staticizing) of a given instruction depends upon what must be accomplished by that instruction.

AUTOMATIC ADDRESS MODIFICATION

The first status level following staticizing checks the two octal digits comprising the N character in the N Register, if these digits are OO, the instruction will be executed as it was stored in the High-Speed Memory. If the first, or left-hand, octal digit is other than O, the quantity stored in the location indexed by that digit will be added to the contents of the A Register (which has received the A address of the instruction) before the instruction is executed. If the second, or right-hand, octal digit is other than O, the quantity stored in the location indexed by that digit will be added to the contents of the B Register (which has received the B address of the instruction) before the instruction is executed.

The addition is octal. In effect, subtraction may be performed by storing the eight's complement of the subtrahend (either the unmodified address or the contents of the indexed location). The N character can thus effect a decrease or increase of the contents of either the A or the B Register, or of both.

Six status levels are used to modify each address. Automatic address modification time, therefore, is 90 microseconds if one address is modified, and 180 microseconds if both addresses are modified.

The locations, and their contents, accessed by the digits of the N character are called **Address Modifiers.** They permit modification not only of addresses, but data also.

The locations associated with each octal digit that could appear in the N character are as follows:

Octal Digit	Location of Modifier
1	HSM locations 000111 - 000113
2	HSM locations 000221 - 000223*
3	HSM locations 000131 - 000133
4	P Register ⁄
5	HSM locations 000151 - 000153
6	T Register
7	HSM locations 000171 - 000173

* See STA, page 5.11

⁄ Note that the P Register always contains the address of the next instruction in sequence (not the address of the instruction currently being executed).

Four of these Address Modifiers (1, 3, 5 and 7) are static; that is, the contents remain as originally stored unless an instruction specifies that they be altered.

Three of the Address Modifiers (2, 4 and 6) are dynamic. The contents of the P Register will change with each instruction performed; the contents of the T Register will change with each instruction that utilizes this register; the contents of standard HSM locations 000221 - 000223 will change with each instruction in which STA is performed.

Example of Automatic Address Modification:

Assume the following instruction is stored in HSM locations 000460 - 000467 (000460 serves as the address of this instruction).

0	A	N	B
21	003100	03	000612

Assume, also, that a part of the memory looks like this:

Address	000131	000132	000133
Contents	04	20	13

The octal digit 3 in the N character of the instruction will access HSM locations 000131 - 000133 and cause the contents 042013 to be added (octally) to the contents of the B Register (000612). Since the left-hand octal digit of the N character is 0, the contents of the A Register (003100) will not be changed. Consequently, the instruction actually executed will be

21	003100	00	042625

The instruction remains in HSM locations 000460 - 000467 as it was originally written and stored:

21	003100	03	000612

PROGRAMME CONTROL

Programme Control is the arithmetic and control unit of the Computer. It interprets and executes the instructions stored in the High-Speed Memory, directs the sequence of operations in the Computer, controls operation of the on-line input-output devices, and performs the automatic accuracy checks. It includes the circuitry for electronic control, switching and buffering of up to eight input-output trunk lines.

Programme Control includes a number of specialized (by function) devices. Those which are of interest to the programmer are shown in Figure 5.1 and are briefly described below, along with certain automatic Programme Control functions.

REGISTERS

The **MEMORY ADDRESSING REGISTER** stores the HSM address of the tetrad to be processed. The capacity of this register is three characters (six octal digits).

The MEMORY REGISTER has a capacity of four characters. It receives the tetrad contents that emerge from or are to be placed in the High-Speed Memory. A series of MEMORY OUTPUT GATES permit or inhibit entrance into the Memory Register of any or all of the four characters that emerge from the HSM.

The P REGISTER holds the HSM address of the next instruction in sequence.

The A REGISTER has a capacity of three characters. It receives the A address of an instruction and, when necessary, holds the address of each character (or tetrad) being processed in the Normal Mode.

The B REGISTER has a capacity of three characters. It receives the B address of an instruction and, when necessary, holds the address of each character or tetrad being processed in the Normal Mode. When an instruction shifts from the Normal to the Simultaneous Mode, the B Register no longer holds the B address of the shifted instruction, but is utilized by the instruction which then occupies the Normal Mode.

The B address is not sent to the B Register in three of the forty-nine instructions: Transfer Control (71), Set Register (72) and Store Register (73).

The S REGISTER has a capacity of three characters. It assumes the function of the A Register for an operation when it shifts into the Simultaneous Mode.

The T REGISTER has a capacity of three characters. It holds the third address when required by an instruction (e.g., HSM address for the quotient in a Decimal Divide Instruction). In some instructions it is used as an internal counter.

The NO (NORMAL OPERATION) REGISTER has a capacity of one character. It holds the Operation Code of the instruction currently being executed in the Normal Mode.

The SO (SIMULTANEOUS OPERATION) REGISTER has a capacity of one character. It holds the Operation Code of the instruction currently being executed in the Simultaneous Mode.

The N REGISTER has a capacity of one character. It holds the N character of the currently processed instruction.

The SR (SELECT READ) REGISTER has a capacity of one character. It holds the address of the input device used in a read operation.

The SW (SELECT WRITE) REGISTER has a capacity of one character. It holds the address of the output device used in a write operation. The SW Register is not used in the Print (On-Line Printer) instruction.

ADDRESSABLE REGISTERS. A, B, P, S and T are all addressable registers. Their contents may be conveniently set and/or stored for subsequent programme reference.

BUSES

The ADDRESS BUS is a three-character pathway connecting the Memory Addressing Register with the A, B, P, S and T registers. Its terminal points are the Memory Addressing Register and the AB-DB Separator (see page 5.8, under Register Gates).

The DATA BUS is a four-character pathway between the Memory Register and the Interchange (see page 5.8, under Register Gates), with branches to allow the C_1, C_2 and C_3 characters to enter the AB-DB Separator and thus pass into the Address Bus. The C_0 character is drawn from this bus for the NO or N Register.

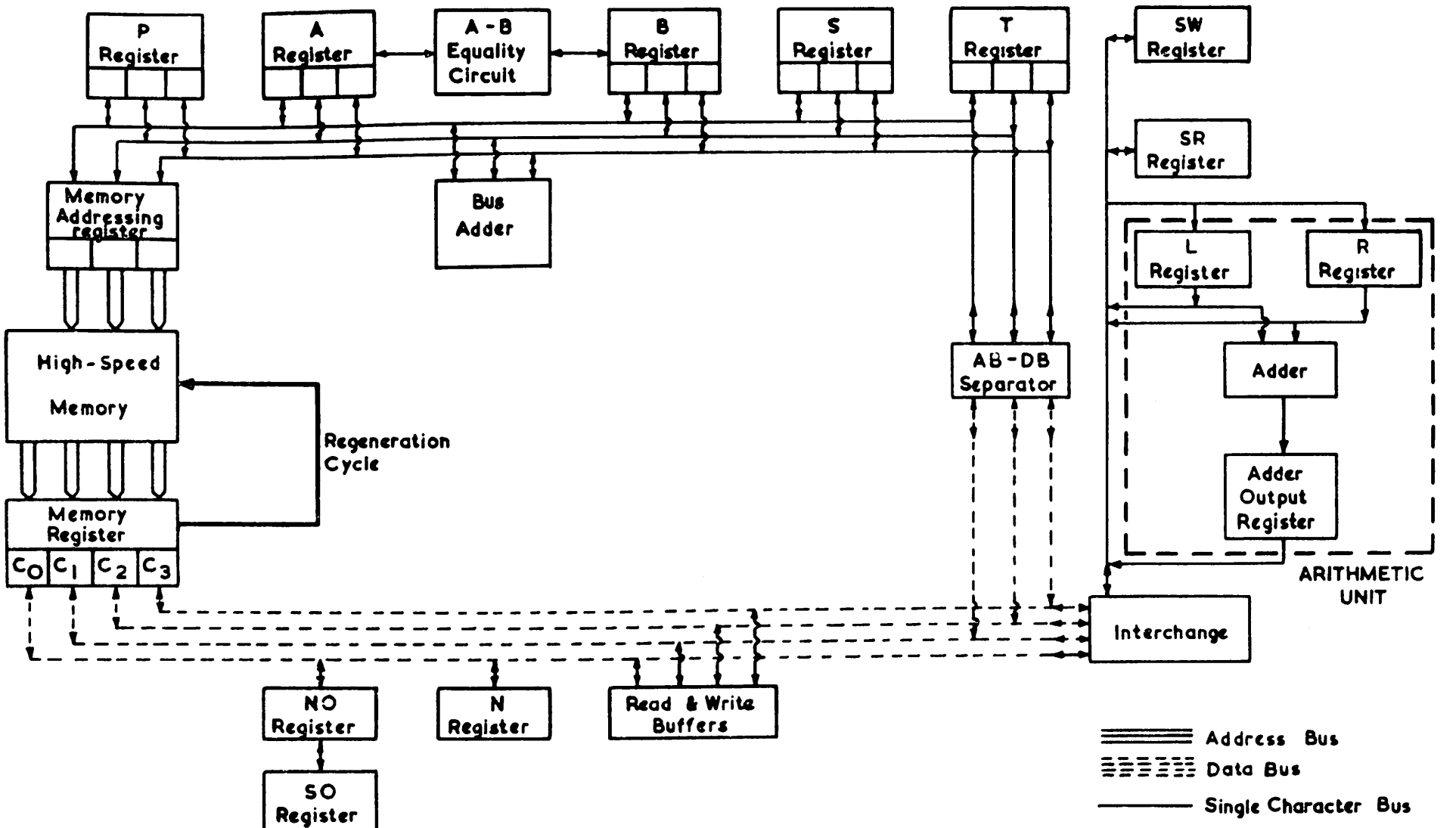


Fig. 5.1 PROGRAMME CONTROL

The SINGLE-CHARACTER BUS forms the connection between the Interchange and the Arithmetic Unit.

REGISTER GATES

Basically, the Register Gates control the entrance into, and the exit from, the various registers and buses. In addition to the two sets at each register (one set for entrance and the other for exit of information), there are two major groups of gates controlling information flow between the various buses:

The ADDRESS BUS - DATA BUS (AB-DB) SEPARATOR serves as the connection between the Address Bus and the Data Bus.

The INTERCHANGE links the Data Bus with the Single-Character Bus. It selects the character to be passed from the four-character Data Bus on to the Single-Character Bus. It also selects the Data Bus line that is to transport the one-character output of the Arithmetic Unit.

BUS ADDER

The Bus Adder is located along the Address Bus, separated from it by a set of register gates. The function of the Bus Adder is to modify the contents of the various three-character address registers (A, B, P, S and T). The Bus Adder can increase the contents of these registers by $(01)_8$ or $(04)_8$ or can decrease them by $(01)_8$, $(04)_8$ or $(10)_8$, thus permitting register contents, for example, to be directly related to the currently processed characters or tetrads.

A-B EQUALITY CIRCUIT

The A-B Equality Circuit is located between the A and B registers and is used to compare their contents. When they are equal, the A Counter-B Counter Equality Flip-Flop is set, indicating to the Computer that the instruction-defined sector has been processed and the instruction can be terminated. In instructions that specify the left and right boundaries of the HSM sector to be processed, (1) the contents of the A Register are increased, toward A-B equality, when the direction of operation is from left to right and (2) the contents of the B Register are decreased, toward A-B equality, when the direction of operation is from right to left.

ARITHMETIC UNIT

The Arithmetic Unit includes three registers, each of one-character capacity, and an Adder Circuit. In an arithmetic operation, the L (left) Register holds one character of one operand and the R (right) Register holds one character of the other operand. These characters are added and the sum is placed in the one-character Adder Output Register, with a flip-flop indicating whether a carry was generated. The sum is accuracy-checked in the Adder Output Register, then transferred on to the One-Character Bus to be returned, via the Memory Register, to the High Speed Memory.

PREVIOUS RESULT INDICATORS (PRI's)

The Previous Result Indicators, a set of three flip-flops, preserve the signs of the result - or the zero result - of an arithmetic operation for automatic reference by a subsequent decision instruction. If the result is positive, the PREVIOUS RESULT POSITIVE (PRP) flip-flop is set; if negative, the PREVIOUS RESULT NEGATIVE (PRN) flip-flop is set; if zero, the PREVIOUS RESULT ZERO (PRZ) flip-flop is set.

The PRI's are set in all decimal arithmetic operations, in most of the binary arithmetic operations, in compare and certain search and transfer operations.

In addition to being automatically referenced by a decision instruction, the PRI's are directly addressable by programme. Current PRI settings can be stored in instruction-designated HSM locations and any one of the PRI's can be set in accordance with programme needs.

AUTOMATIC STORAGE OF FINAL CONTENTS OF THE A REGISTER (STA)

STA is an automatic operation which occurs at the conclusion of forty of the forty-nine instructions. In STA, the final contents of the A Register are automatically stored in the standard High-Speed Memory locations 000221-000223. This permits the indirect use of the A Register as a dynamic Address Modifier. It is also a highly convenient programming device which can be utilized to eliminate memory-searching time for subsequent processing. If STA were not automatically performed, the final A Register contents for one instruction would be destroyed as soon as the next instruction was staticized.

STA takes 15 microseconds and is performed in every instruction except those in the following list:

Transfer Control (71)
Conditional Transfer of Control (61)
Tally (66)
Set Register (72) except Set A Register
Store Register (73)
Tape Sense (63)
Sense Simultaneous Gate (65)
Control Simultaneous Gate (75)
Sense Simultaneous Mode (62)

AUTOMATIC STORAGE OF CONTENTS OF P REGISTER (STP)

STP is an automatic operation which occurs whenever control is to be transferred; that is, whenever the next instruction to be performed is not the one stored immediately following the current instruction. STP automatically stores the contents of the P Register in standard High-Speed Memory locations 000241 - 000243. Since the P Register always holds the address of the next instruction in sequence, the standard HSM locations may be accessed to construct, elsewhere in the HSM, a record of the instructions to which the programme would have proceeded if transfer had not occurred.

STP always occurs in the following two instructions:

Transfer Control (71)
Sense Simultaneous Gate (65)

In the following four instructions, STP is performed **only when control is actually transferred**:

Conditional Transfer of Control (61)
Tally (66)
Tape Sense (63)
Sense Simultaneous Mode (62)

Unlike STA, the STP function is not a time factor; it does not add to basic instruction processing time.

SIMULTANEITY (TIME-SHARING OPERATIONS)

Simultaneity in the Computer is defined as coincident execution of two instructions, both or one of which is an input-output instruction.

All instructions are staticized in the Normal Mode. Some instructions must be totally executed in the Normal Mode. All but three of the input-output instructions can be completed in either mode and are termed **potentially simultaneous (PS)** instructions.

A potentially simultaneous instruction automatically shifts (any time after it is staticized) into, and is then completed in, Simultaneous if that mode is unoccupied by a previous instruction and if the Simultaneous Gate is open. This permits initiation (in the freed Normal Mode) of the next instruction in sequence. The two instructions are then executed with total or partial coincidence in time (time-shared). The exceptions are that two 'read' instructions or two 'write' instructions cannot be executed simultaneously. If, for instance, a 'read' instruction is in process in the Simultaneous Mode and another 'read' instruction is staticized in the Normal Mode, the latter instruction is not executed until the instruction in the Simultaneous Mode has been completed.

Reading and writing may be accomplished simultaneously as long as the two instructions do not involve the same Tape Station. If, for example, a 'read' from Tape Station 20 is staticized while writing is in process at that Tape Station, the 'read' will not be executed until writing has been completed.

Trunk $(77)_8$, however, can be time-shared by a paper tape 'read' and a 'write-out' to the Monitor Printer.

Time-shared electronics in the Computer permits the following simultaneous operations:

compute and magnetic tape write
compute and paper tape punch
compute and monitor print

paper advance* and compute
paper advance* and magnetic tape write
paper advance* and paper tape punch
paper advance* and monitor print

paper tape read and compute
paper tape read and magnetic tape write
paper tape read and paper advance*
paper tape read and paper tape punch
paper tape read and monitor print

magnetic tape read and compute
magnetic tape read and magnetic tape write
magnetic tape read and paper advance*
magnetic tape read and paper tape punch
magnetic tape read and monitor print

Any one of the above-listed combinations is possible while at the same time, any number of tapes are rewinding.

Simultaneous operation within the Computer is made possible by the low-duty cycle of the High-Speed Memory during execution of most of the input-output instructions. The majority of the time required for the execution of a tape instruction, for example, is used up in tape movement; The High-Speed Memory is involved only a very small fraction of that time. If properly controlled, therefore, the memory is available for other functions while it is waiting for the tape to be advanced. In order to accomplish this with a minimum of buffering and additional hardware, an interruption technique is employed. That is, the sequence of instructions being executed simultaneously with the tape function is automatically interrupted when the memory must receive or transmit information in connection with the tape operation.

* With an On-Line Printer.

The KDP 10 includes two Read buffers and two Write buffers; each has a capacity of four characters. One character from tape is clocked into the first Read Buffer in 30 microseconds, and the buffer is filled in 120 microseconds. The entire contents shift into the second Read Buffer and then are transferred in parallel, in one status level (15 microseconds) into an HSM tetrad. The 30-microsecond clocking of characters from tape to buffer continues uninterrupted until the read instruction has been completed.

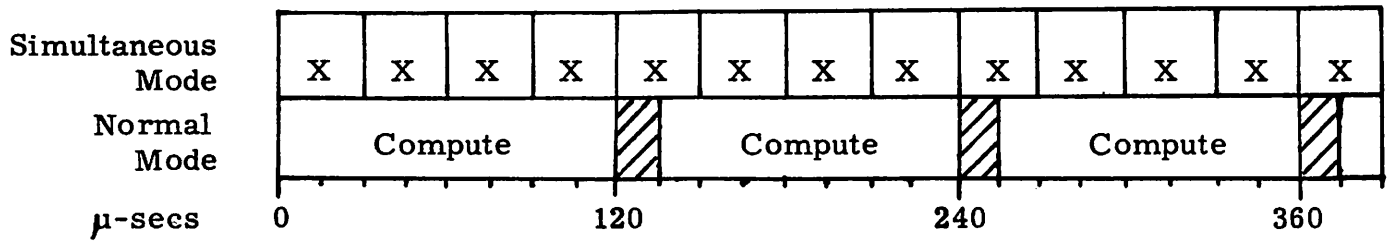
During 105 μS of the 120 μS buffer-filling time, the Computer is free to execute another instruction or instructions. The number of instructions that can be executed simultaneously with any one read instruction depends upon the number of characters to be read in. Execution of these instructions is interrupted only for the 15-microsecond, transfer time.

These same factors apply to 'write' operations, except that the 15 microsecond interruption occurs with tetrad transfer from the HSM to the Write Buffer and the Computer is free during seven-eighths (105 μS) of the 120 microseconds required to write out the four characters.

Assuming a 'read' instruction being executed in the Simultaneous Mode and 'compute' functions (add, locate, compare, etc.) in the Normal Mode, simultaneity may be illustrated by the diagram below. (Each x represents a character read in from tape; the shaded areas represent interruption of 'compute' during buffer-to-HSM transfer).

The programmer can control the use of simultaneity by employing instructions which sense the state of and control the Simultaneous Gate. When open, this gate permits transition of potentially simultaneous instructions from the Normal to the Simultaneous Mode. When closed, the gate prevents such transition so that all instructions are performed in the Normal Mode. It is thus possible to run programmes entirely in the Normal Mode or to bracket off certain parts of programmes in which transition to the Simultaneous Mode is not desirable.

In addition, use of the instruction 'Sense Simultaneous Mode' provides the information that the Simultaneous Mode is engaged in a 'read', a 'write' or a paper advance, or that it is totally unengaged.



THE KDP 10 ORDER CODE

INTRODUCTION

The Computer operates under the direction of forty-nine basic, wired-in, two-address instructions. For descriptive purposes, these instructions may be classified into four general categories: (1) Input-Output, (2) Data-Handling, (3) Arithmetic and (4) Decision and Control.

INPUT-OUTPUT INSTRUCTIONS

These instructions enable the Computer to communicate with the on-line peripheral devices (Magnetic Tape Stations, Paper Tape Reader, Monitor Printer and Paper Tape Punch, On-Line Printer). They perform the functions of positioning or searching tapes, bringing data from an input medium into the Computer, or sending data from the Computer to an output medium.

Most of the instructions in this group are potentially simultaneous (PS); i.e., they can be executed in the Simultaneous Mode, so that operational time for these instructions can overlap that of other instructions. (See Simultaneity, page 5.9). Two instructions, Single Sector Write (11) and Multiple Sector Write (13), are not PS instructions. One instruction, Print (02), occupies both modes and cannot be executed simultaneously with any other Computer-controlled operation.

One input-output instruction, Rewind to BTC (17), is initiated by the Computer but, once underway, operates completely independent of the Computer. Any number of tapes may be rewinding while two instructions (unrelated to the rewinding tapes) are being simultaneously executed.

After staticizing and address modification, a 'read' instruction is automatically stored in standard High-Speed Memory locations 000020 - 000027. The instruction in the standard locations is then available for a Rollback (rerun) routine if an error is detected by the checking circuitry (see description of Rollback, Appendix I) or for other programme needs.

Four 'write' instructions - Linear Write (12), TCW (10), Single Sector Write (11) and Multiple Sector Write (13) - are automatically stored (after staticizing and address modification) in standard High-Speed Memory locations 000030 - 000037.

DATA-HANDLING INSTRUCTIONS

These are non-arithmetic instructions for manipulation of data stored in the High-Speed Memory. The instructions included in this group permit operation control by symbol or by address, and transfer of data with or without editing.

With the exception of Locate nth Symbol in Sector (31), Zero Suppress (32 and Random Distribute (27), all data-handling sector and item instructions operate from right to left, i.e., processing begins with the larger specified HSM address and progresses to the smaller or to the Item Separator symbol.

ARITHMETIC INSTRUCTIONS

Four of the instructions in this group are decimal arithmetic instructions, five are binary, and two are used to alter the bit configuration of an operand.

The decimal instructions operate in accordance with arithmetic rules and are designed to handle operands of unequal and practically unlimited length. They employ the Computer's ability to recognize control symbols, so that the arithmetic process, in effect, is performed with alignment of the least significant digits and is terminated when the Item Separator symbol (ISS) or a space character to the left of the longer operand is encountered. Thus, the need for the programmer to pre-position operands, by shifting, is largely eliminated, since proper alignment will be achieved even if one (or both) of the operands as addressed contains a series of spaces to the right of the sign.

The binary instructions handle operands of equal but unlimited length. Here length is not defined by the presence of a control symbol, but by address specification. Alignment of the operands must be programmed, since it is not automatically performed as in the decimal instructions. All the characters in the operands are treated as numerics, and the bits in each bit position are added together in accordance with the stated binary rules.

Two instructions, Logical 'and' (47) and Logical 'or' (46), constitute what may be considered a separate arithmetic category. They are used to alter the bit configuration of an operand, by the employment of a second operand to 'mask out' or to insert '1' bits.

The Previous Result Indicators (PRI's) preserve the sign of the result of an arithmetic instruction for reference by a subsequent decision instruction. (See page 5.8). Binary Add and Three-character Add do not set the PRI's.

DECISION AND CONTROL INSTRUCTIONS

Seven of the decision and control instructions influence the sequence of operation. Four of these are conditional; that is, they choose a path according to PRI settings, the kind of instruction currently in the Simultaneous Mode, the state of the Simultaneous Gates, or the status of a designated Tape Station. Two of the seven are unconditional commands, and one enables the Computer to execute the same subroutine any designated number of times.

Two instructions enable the programmer to address registers directly, one instruction controls the Simultaneous Gate; and two instructions are available for stopping Computer Operation, one with and one without error indication.

INSTRUCTION REFERENCE SECTIONS

The formal definitions of the operation of each instruction are to be found in Sections 10 to 17 of this manual. Instructions are presented in these sections in ascending order of operation code from 01 (PES) to 77 (RAI). In an attempt to assist the reader the sections are so arranged that all instructions with 0 as the first octal digit of the operation code are presented in Section 10, those with 1 as the first octal digit in Section 11 and so on. The reference information for each instruction includes details of functional logic, timing and post-operational register settings. Some notes on these are given below together with two lists of KDP 10 instructions.

The first list, which is in ascending order of operation code, provides a brief description of each instruction and indicates the place in the reference section at which a full description of the instruction may be found. The second list, in alphabetic order of instruction serves as an alternative index to the reference sections of the manual.

Logic

A logic sub-section supplements the general description of each instruction. The details of the logic section are included for three reasons: (1) to help the programmer towards a better understanding of Computer operation, (2) to permit him to adapt the instructions to unusual circumstances, and (3) to enable him to develop advanced programming techniques.

Timing

Owing to the variable item length concept in the KDP 10 System, instruction times can be expressed only as a function of the number of significant characters involved in a given operation. For example, the time required to write out to tape depends on the number of characters to be written, and the time required to add two numbers together depends on the number of digits in the operands.

The timing formulas for read and write instructions are applicable to magnetic tape since this is the most frequently used input-output medium. For other types of input or output, timing is governed largely by the input or output rate of the device used (see page 1.3 for input-output performance, Monitor Printer, Paper Tape Punch, Paper Tape Reader).

The time, or timing formula, listed for each instruction does not include staticising, automatic address modification or STA time. As stated previously:

Staticising time is 30 microseconds and is constant for each instruction.

Automatic address modification occupies 90 microseconds if the A or the B address is modified,; 180 microseconds if both A and B are modified.

STA occupies 15 microseconds. It is automatically performed in every instruction except in those in which it is specifically stated otherwise.

Examples

Wherever possible, the examples that accompany the instructions include a representation of the affected portion or portions of the High-Speed Memory. The contents of the HSM locations are shown as octal values in most of the binary instructions, and as decimal digits, alphabetic characters, symbols, etc., in other instructions.

In each example, the subhead 'HSM before Instruction is executed' is to be interpreted as before execution but after staticising. The contents of the memory locations are not affected by staticising. The initial register settings $(A)_i$, $(B)_i$, $(T)_i$, however, reflect register contents after staticising.

LIST OF KDP 10 INSTRUCTIONS

01: PROGRAMME ERROR STOP (PES) (Page 10.1)

This instruction inhibits the Staticising of any further instructions, halting the Computer after completion of any instruction in the Simultaneous Mode and illuminating an alarm light on the Computer Console.

02: PRINT (PR) (Page 10.2)

This instruction transfers the characters stored in 120 consecutive HSM locations to the Line Printer, causing one line to be printed. The operation is initiated only when the Simultaneous Mode is free, since it uses both the Normal and Simultaneous Modes.

03: PAPER ADVANCE (PA) (Page 10.4)

This instruction positions the paper in the Line Printer for the next line of printing. It can advance the paper a specified number of lines, designated either by A_2A_3 or by the punches in a tape loop on the Printer. This is a potentially simultaneous instruction. While in the Simultaneous Mode, it does not restrict the use of any other instruction in the Normal Mode except Print or another Paper Advance.

04: LINEAR READ REVERSE (LRR) (Page 10.6)

This instruction transfers one message from magnetic or paper tape to the HSM. It is a potentially simultaneous instruction. Though the tape is read in reverse, the characters will be placed in the HSM in their proper relative positions. LRR is most useful in sort routines since it saves rewind time.

05: BLOCK READ REVERSE (BRR) (Page 10.8)

This instruction transfers a block of characters from magnetic or punched paper tape into the HSM. Transfer begins with the first character following a gap and ends when the next gap is sensed. Though the tape moves in reverse, the characters will be placed in the HSM in their proper relative positions. The instruction is potentially simultaneous.

06: UNWIND n SYMBOLS (UNS) (Page 10.10)

This instruction causes a selected magnetic tape to be moved forward until a specified number of a designated symbol have been counted. The HSM is not altered. The instruction is potentially simultaneous, but no 'read' instruction can be executed simultaneously with it.

10: TRANSCRIBING CARD PUNCH WRITE (TCW) (Page 11.1)

This instruction is the same as LINEAR WRITE (12) except that characters are written to tape at half speed.

11: SINGLE SECTOR WRITE (SSW) (Page 11.2)

This instruction writes on to magnetic tape (or the Monitor Printer), in block format, the contents of a sector of any size, located in any area of the HSM. This instruction can be executed only in the Normal Mode. However, a prior 'read' may, at the same time, be in the Simultaneous Mode.

12: LINEAR WRITE (LW) (Page 11.4)

This instruction transfers one message from the HSM to magnetic tape, the Monitor Printer or to paper tape via the Monitor Printer. It is a potentially simultaneous instruction.

13: MULTIPLE SECTOR WRITE (MSW) (Page 11.6)

This instruction writes on to magnetic tape (or the Monitor Printer), a single block comprising the contents of any number of sectors taken from various parts of the HSM under the direction of a stored list of addresses. This instruction is executed only in the Normal Mode.

14: LINEAR READ FORWARD (LRF) (Page 11.10)

This instruction brings one full message from magnetic or punched paper tape into the HSM. It is a potentially simultaneous instruction.

15: BLOCK READ FORWARD (BRF) (Page 11.12)

This instruction brings a block of characters from magnetic or punched paper tape into the HSM. Transfer from tape begins with the first character following a gap, and ends when the next gap is sensed. This instruction is potentially simultaneous.

16: REWIND n SYMBOLS (RNS) (Page 11.14)

This instruction causes a selected magnetic tape to be moved backward through a specified number of symbols. This instruction is potentially simultaneous, but no 'read' instruction can be executed simultaneously with it. The HSM is not altered.

17: REWIND TO BTC (RWD) (Page 11.16)

This instruction causes a designated magnetic tape to be completely rewound. Once the operation has been initiated, the rewind proceeds totally independent of the Computer, occupying neither the Normal nor the Simultaneous Mode. The Computer, after initiating the rewind, is free to execute other instructions.

21: ITEM TRANSFER (IT) (Page 12.1)

This instruction transfers an item from one group of successive HSM locations to another.

22: ONE-CHARACTER TRANSFER (OCT) (Page 12.3)

This instruction transfers the contents of one HSM location into another HSM location. It may be used to modify portions of instructions, transfer one-character constants, etc.

24: SECTOR TRANSFER BY CHARACTER (STC) (Page 12.4)

This instruction transfers a series of characters from an area (sector) between, and including, two designated HSM locations to another HSM area (sector).

25: THREE-CHARACTER TRANSFER (TCT) (Page 12.6)

This instruction transfers, in parallel, the contents of the right-most three locations of one tetrad to the rightmost three locations of another tetrad in the HSM. It is useful for placing addresses into stored instructions, setting Address Modifiers, etc.

26: SECTOR TRANSFER BY TETRAD (STT) (Page 12.8)

This instruction transfers the contents of one tetrad or any number of consecutive HSM tetrads between, and including, two specified tetrad addresses, into another specified tetrad or consecutive series of tetrads. This instruction differs from Sector Transfer by Character (24) in that it transfers four characters at a time, and is, therefore, four times faster.

27: RANDOM DISTRIBUTE (RD) (Page 12.10)

This instruction distributes successive items in the HSM, to locations designated by a stored list of addresses.

31: LOCATE n^{th} SYMBOL IN SECTOR (LNS) (Page 13.1)

This instruction searches through the contents of successive HSM locations between, and including, two given addresses, counting the occurrences of a designated symbol. The operation ceases when (1) the specified count is reached or (2) the rightmost location in the sector has been searched.

32: ZERO SUPPRESS (ZS) (Page 13.4)

This instruction is used to delete the non-significant zeros to the left of the MSC of the result of a decimal arithmetic operation.

33: JUSTIFY RIGHT (JR) (Page 13.7)

This instruction, used to effect right columnar alignment, (1) adjusts and transfers an item from one series of successive HSM locations to another, or (2) adjusts the item, leaving it in the same group of HSM locations. All the space symbols which were originally located to the right of the sign position, are placed between the ISS and the MSD in the destination area. (The sign position is the HSM location immediately to the right of the LSD).

34: SECTOR CLEAR BY CHARACTER (SCC) (Page 13.11)

This instruction places space characters in all the locations between, and including, two HSM addresses.

35: SECTOR COMPRESS - RETAIN REDUNDANT ISS's (SCR) (Page 13.14)

This instruction transfers a sector of characters from one part of the HSM to another, removing, in the process, all spaces located to the right of the rightmost non-space character within each item in the sector.
Note: A space in the sign position (i.e., positive sign) is also deleted.

36: SECTOR CLEAR BY TETRAD (SCT) (Page 13.16)

This instruction inserts spaces $(01)_8$ in the HSM locations between, and including, two given tetrad addresses. It differs from the Sector Clear by Character Instruction (34) in that it clears four characters at a time, and is, therefore, four times faster.

37: SECTOR COMPRESS - DELETE REDUNDANT ISS's (SCD) (Page 13.18)

This instruction transfers a sector of data from one part of the HSM to another, deleting, in the process, (1) all ISS's originally located to the right of the rightmost non-ISS, non-space character in the sector, and (2) all spaces located to the right of the rightmost non-space character within each item in the sector. Note: A space in the sign position (i.e., positive sign) is also deleted.

41: BINARY ADD (BA) (Page 14.2)

This instruction performs binary addition of two equal length operands and places the sum in the HSM locations originally occupied by the augend. The operands may be of any length. Each character (including control symbols) is treated as if it were numeric.

42: BINARY SUBTRACT (BS) (Page 14.4)

This instruction performs binary subtraction of one operand from another operand of equal length, placing the difference in the HSM locations originally occupied by the minuend. The operands may be of any length. Each character (including control symbols) is treated as if it were numeric.

43: SECTOR COMPARE (SC) (Page 14.8)

This instruction is used to determine the relative magnitude of two operands of equal length. A single operand may consist of one character or any number of alpha-numeric characters and/or symbols. Binary subtraction is performed, but the difference is not stored in the HSM. However, the resultant PRI settings permit alternative sequences of instructions.

44: THREE-CHARACTER ADD (TCA) (Page 14.10)

TCA is mainly designed to modify addresses of instructions and to keep octal counters. As such, it performs binary addition of an augend stored in the rightmost three locations of a tetrad and a three-character addend. The result is automatically stored in the locations previously occupied by the augend.

45: THREE-CHARACTER SUBTRACT (TCS) (Page 14.12)

Like the Three-Character Add, TCS is mainly designed for address modification and octal counters. It performs binary subtraction of two operands stored in the rightmost three locations of their respective tetrads. The difference is automatically stored in the locations previously occupied by the minuend.

46: LOGICAL 'OR' (LO) (Page 14.14)

This instruction performs a function similar to the 'or' in machine logic, inserting '1' bits from a specified modifier into a specified operand of equal length.

47: LOGICAL 'AND' (LA) (Page 14.16)

This instruction performs a function similar to the 'and' in machine logic. It may be used to extract '1' bits from an operand according to a second operand ('mask') of equal length.

51: DECIMAL ADD (DA) (Page 15.1)

This instruction performs decimal addition, in accordance with algebraic rules, producing a non-zero-suppressed sum, which is stored in the HSM locations originally occupied by the augend. The operands may be of any length and, also, of unequal lengths.

52: DECIMAL SUBTRACT (DS) (Page 15.6)

This instruction performs decimal subtraction on variable length operands. The subtraction is algebraic. Specifications as to operands, storage of the difference and end conditions are exactly the same as in Decimal Add.

53: DECIMAL MULTIPLY (DM) (Page 15.10)

This instruction performs decimal multiplication in accordance with algebraic rules, producing a non-zero-suppressed product. If a quantity is pre-stored in the product area, the product is added (absolute addition) to it, permitting round-off by any number and multiply-accumulate. However, the sign of the product will be assigned to the accumulated (absolute) result.

54: DECIMAL DIVIDE (DD) (Page 15.14)

This instruction performs decimal division in accordance with algebraic rules and produces a non-zero-suppressed quotient. The non-zero-suppressed remainder is stored in the HSM locations originally occupied by the dividend. Each operand must carry a sign. The operands may be of any length. The length of each operand is defined by the first space to the left of a non-space, non-minus character, or by an ISS. If the divisor contains more digits than the dividend, the quotient will be a zero. Unlike Decimal Multiply, the quotient is not added to a prestored quantity.

61: CONDITIONAL TRANSFER OF CONTROL (CTC) (Page 16.1)

This instruction chooses one of three sequences of instructions, in accordance with the setting of the PRI's.

62: SENSE SIMULTANEOUS MODE (SSM) (Page 16.2)

This instruction selects one of four sequences of instructions, depending upon whether the Simultaneous Mode is (1) unoccupied, (2) occupied by a 'read' instruction, (3) occupied by a 'write' instruction, or (4) occupied by a Paper Advance.

63: TAPE SENSE (TS) (Page 16.3)

This instruction tests the status of a given Tape Station, permitting programme direction to one of two sequences of instructions.

65: SENSE SIMULTANEOUS GATE (SSG) (Page 16.5)

This instruction chooses one of two sequences of instructions, depending upon whether or not the Simultaneous Gate is open.

66: TALLY (TA) (Page 16.6)

This instruction permits looping through a sequence of operations by automatically reducing a prestored quantity each time control is transferred to the beginning of the sequence. When the quantity has been exhausted, the Tally ends and the instruction following it is performed.

71: TRANSFER CONTROL (TC) (Page 17.1)

This instruction either causes an unconditional break in the sequence of instructions or takes action according to the settings of the Breakpoint Switches on the Computer Console.

72: SET REGISTER (SET). (Page 17.2)

This instruction replaces the contents of a specified register with the A address of the instruction or sets a PRI.

73: STORE REGISTER (STR) (Page 17.3)

This instruction places the contents of a selected register (or PRI setting) into the rightmost three locations of a designated tetrad.

75: CONTROL SIMULTANEOUS GATE (CSG) (Page 17.5)

This instruction is used to open or close the gate which controls entrance into the Simultaneous Mode, making it possible to either prevent or permit simultaneous operations.

76: STOP (ST) (Page 17.6)

This instruction inhibits the staticising of any further instructions, halting the Computer after completion of any instruction in the Simultaneous Mode.

77: RETURN AFTER INTERRUPT (RAI) (Page 17.7)

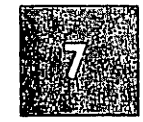
This instruction is used to re-enter a programme after an unscheduled interruption, such as for Rollback or for a higher priority programme.

The RAI was designed not only to transfer control, but to permit both the A and B Registers to be properly set in the process. Thus, when the main programme is re-entered, all of the pertinent conditions prevailing at the time of interruption can be re-established.

ALPHABETIC INDEX OF KDP 10 INSTRUCTIONS

MNEMONIC	INSTRUCTION NAME	OP CODE	PAGE
BA	Binary Add	41	14.2
BS	Binary Subtract	42	14.4
BRF	Block Read Forward	15	11.12
BRR	Block Read Reverse	05	10.8
CSG	Control Simultaneous Gate	75	17.5
CTC	Conditional Transfer of Control	61	16.1
DA	Decimal Add	51	15.1
DS	Decimal Subtract	52	15.6
DM	Decimal Multiply	53	15.10
DD	Decimal Divide	54	15.14
IT	Item Transfer	21	12.1
JR	Justify Right	33	13.7
LA	Logical AND	47	14.16
LNS	Locate n^{th} Symbol	31	13.1
LO	Logical OR	46	14.14
LRF	Linear Read Forward	14	11.10
LRR	Linear Read Reverse	04	10.6
LW	Linear Write	12	11.4
MSW	Multiple Sector Write	13	11.6
OCT	One Character Transfer	22	12.3
PA	Paper Advance	03	10.4
PES	Programme Error Stop	01	10.1
PR	Print	02	10.2
RAI	Return after Interrupt	77	17.7
RD	Random Distribute	27	12.10
RNS	Rewind n Symbols	16	11.14
RWD	Rewind to BTC	17	11.16
SC	Sector Compare	43	14.8
SCC	Sector Clear by Character	34	13.11
SCD	Sector Compress Delete Redundant ISS	37	13.18
SCR	Sector Compress Retain Redundant ISS	35	13.14
SCT	Sector Clear by Tetrad	36	13.16
SET	Set Register	72	17.2
SSG	Sense Simultaneous Gate	65	16.5
SSM	Sense Simultaneous Mode	62	16.2
SSW	Single Sector Write	11	11.2
ST	Stop	76	17.6
STR	Store Register	73	17.3
STC	Sector Transfer by Character	24	12.4
STT	Sector Transfer by Tetrad	26	12.8

MNEMONIC	INSTRUCTION NAME	OP CODE	PAGE
TA	Tally	66	16.6
TC	Transfer Control	71	17.1
TCA	Three Character Add	44	14.10
TCW	Transcribing Card Punch Write	10	11.1
TCS	Three Character Subtract	45	14.12
TCT	Three Character Transfer	25	12.6
TS	Tape Sense	63	16.3
UNS	Unwind n Symbols	06	10.10
ZS	Zero Suppress	32	13.4



FIXED FIELD PROGRAMMING

INTRODUCTION

All computers can perform a limited number of basic operations. They are able to transfer information from input equipment to the computer, manipulate data within the computer and transfer the results to output equipment. To do these various tasks computers are provided with the ability to interpret coded instructions. Every instruction performs a well defined operation and the programmer must understand and learn the scope of each instruction in the order code. To write a successful programme the programmer must know:

a) What needs to be done

b) How to do it

and c) How to cause the computer to do it.

It is assumed that conditions (a) and (b) are satisfied and the present discussion is concerned only with condition (c).

A list of KDP 10 instructions will be found on Page 6.4. Each instruction is fully described, with examples, in the reference section of this manual starting on page 10.1. Some idea must be given of how to synthesise groups of instructions to perform specified programming tasks. To do this it will be assumed, in the first instance, that the KDP 10 is a very elementary machine with rather limited capabilities. The order code will be restricted to six instructions and examples of the use of these will be given. As the examples become more ambitious it will be apparent that the limitations of the original six instructions must be removed by the inclusion of other instructions.

This general pattern will be followed and the reader should soon be in a position to follow the steps of large programmes.

Instructions are referred to either by name, by a mnemonic abbreviation or by an octal number.

MNEMONIC CODE	NAME	OPERATION CODE
LRF	LINEAR READ FORWARD	14
OCT	ONE CHARACTER TRANSFER	22

Programmers soon acquire familiarity with all three designations and it is well worth while to devote serious effort to memorising the mnemonic and operation codes of each instruction. After all, no one has yet learned to spell without first memorising the alphabet. This subject is one in which memory assists understanding and the sooner a programmer starts to speak the language of the KDP 10 the sooner will he appreciate its capabilities.

SIX BASIC KDP 10 INSTRUCTIONS

The instructions chosen for illustrating simple examples of programming are:-

(A)	LRF	LINEAR READ FORWARD	Page 11.10
(B)	LW	LINEAR WRITE	Page 11.4
(C)	DA	DECIMAL ADD	Page 15.1
(D)	DS	DECIMAL SUBTRACT	Page 15.6
(E)	IT	ITEM TRANSFER	Page 12.1
(F)	TC	TRANSFER CONTROL	Page 17.1

(A) LINEAR READ FORWARD, reads messages from paper tape or magnetic tape. The instruction must specify:

a) which input unit is required

b) where the first character of the message shall be placed in the HSM

(B) LINEAR WRITE, writes messages from a designated area of the HSM to tape on a specified output unit.

(C) DECIMAL ADD, adds two decimal numbers situated anywhere in the HSM, placing the result over (or to the right of) one of the original operands. The operands can be of any length, not necessarily equal.

(D) DECIMAL SUBTRACT, subtracts one variable length operand from another, storing the result over, or to the right of, the minuend.

(E) ITEM TRANSFER, transfers an item from one storage location to another (leaving a copy of the item in the original locations).

(F) TRANSFER CONTROL, causes the next instruction to be taken under the direction of the TC instruction. Normally the next instruction is the one immediately following the previous instruction.

PROGRAMMING EXAMPLE No. 1

Tape Unit 10 is loaded with a reference file. The file is made up from **fixed field** messages and each contains a Part Number, Stock Balance, Total Quantity Issued and Total Quantity Received. On Tape Unit 20 is mounted a transaction file composed of **fixed field** messages defining **issues** from stock. There is one, and only one, transaction message for every reference message. The message layout for the two files is given below. It is required to update the **Stock Balance** and **Issues Total** and prepare an updated reference file on Tape Unit 30.

REFERENCE FILE

● PART NO.	● STOCK BALANCE	● TOTAL ISSUES	● TOTAL RECEIPTS
5 char.	8 char.	6 char.	6 char.
	+ sign	+ sign	+ sign

TRANSACTION FILE

● PART NO.	● QUANTITY ISSUED
5 char.	5 char.

A fixed field message is one in which all the component items of the message **always** contain the same number of characters. In the example the Stock Balance always has 8 characters. It is further assumed that the non-significant characters are decimal zeros, so that a Stock Balance of, for example, 3576 is contained in the message as 00003576. Similar remarks apply to Total Issues and Total Receipts.

The action required is fairly simple. The quantity issued as given in the transaction message must be subtracted from the Stock Balance and added to the Total Issue of the corresponding reference information, and it will be assumed further that both files are in the same order of message sequence by Part Number.

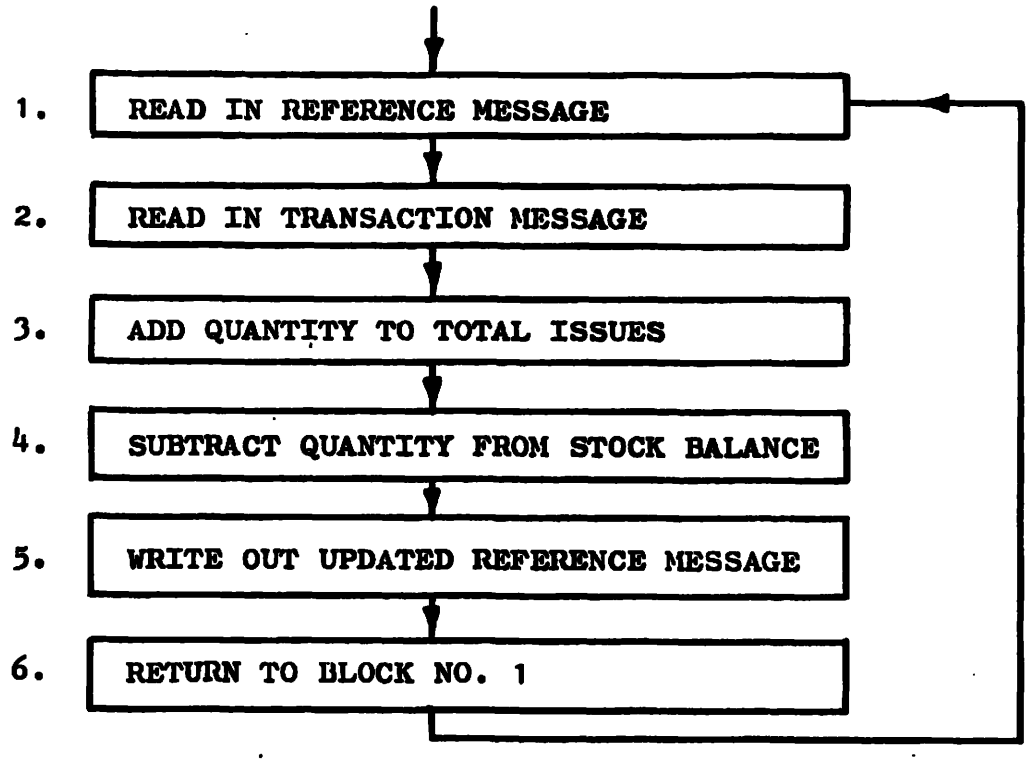
Additions and subtractions can only occur within the computer so the first task is to read messages from both files into the HSM. Where these messages are placed is a decision for the programmer. Let it be decided that the Read-in areas for the two messages are:

a) REFERENCE MESSAGE 010000

b) TRANSACTION MESSAGE 010100

When the messages are read into the HSM they will appear as shown on the COMPUTER HSM RECORD on Page 7.6. Note that all quantities (except Part No.) in the reference message have sign locations at the right-hand end of the items. These sign locations are essential if additions are to be performed in the Read-in area of the Reference Message. The Decimal Add instruction insists that a sign location (or space to the right of the sign) must be nominated in the A address of the instruction, otherwise a failure occurs.

The next task is to prepare a logical flow diagram of the operations as follows:-



The instructions appear on a CODING SHEET or PROGRAMME RECORD as shown below, the coding starts at HSM location 010200.

HSM	OP	A	N	B	Remarks
010200	14	010000	00	10 00 00	LRF
010210	14	010100	00	20 00 00	LRF
010220	51	010030	00	01 01 14	DA
010230	52	010020	00	01 01 14	DS
010240	12	010000	00	30 00 00	LW
010250	71	010200	00	00 00 00	TC
010260					

As it appears above the coding sheet is not, in itself, very informative. It should be interpreted in conjunction with the high speed memory layout and the following notes indicate points of interest.

a) Six instructions complete with example. These instructions must be stored somewhere in the HSM. They can not be stored at 010000 or at 010100 or they would be overwritten by the incoming messages. Storage location 010200 is chosen as the first instruction position. Instructions have eight characters and 010200 is actually the storage location of the OPERATION CODE character of the first instruction. The next available space for the following instruction is at 010210 and it will be noticed that instruction addresses increase by $(10)_8$ for successive instructions.

b) The first instruction is LRF. It is required to read-in from Tape Unit 10 so $(10)_8$ is placed in the B_1 position. The SM of the message must fall in 010000, so this location is nominated in the A address. (010001, 010002 or 010003 would also have the same effect - why?).

c) The next instruction reads from Tape Unit 20. The operation code is again 14 for LRF and this time the B_1 character is $(20)_8$ and the A address is 010100. At this point two messages are in the HSM at the positions specified.

d) The next step is to add the QUANTITY from the Transaction Message to the TOTAL ISSUES and place the result over the original TOTAL ISSUES. The sign position of TOTAL ISSUES is at the extreme right of the TOTAL ISSUES ITEM with address 010030. This figure is placed in the A address of the DA instruction. The LSD of the QUANTITY is at the extreme right of the appropriate item in the Transaction Message with address 010114. This figure is placed in the B address. The sign of the result is placed in 010030.

e) Decimal Subtract is very similar to DECIMAL ADD. The sign position of the STOCK BALANCE is 010020 and this figure goes in the A address. The B address is the LSD of the Quantity Issued, i.e. 010114.

f) The reference message is now updated and may be written out using a LINEAR WRITE instruction. The octal character $(30)_8$ in B_1 specifies TAPE UNIT 30 and the output message starts in 010000 so this figure is placed in the A address.

g) The instructions have been obeyed one after the other starting at 010200 in the order:

010200
010210
010220
010230
010240

At this point it is required to direct the computer to start again at 010200 and repeat all the operations for the next messages from both files. Instruction 010250 is a TRANSFER CONTROL. The operation code is 71 and the A address specifies the address of the next instruction.

The area occupied by these instructions is indicated on the HSM Record. It would be quite impossible to use the HSM Record for instructions and another type of form is used. This is shown on Page 7.7 and the entries are completed by the addition of suitable remarks on each line and by the use of cross references. The Chart Reference column links the coding to the logical flow diagram and the column headed 'FROM INST. LOC.' provides cross references for transfers of control.

PROGRAMMING EXAMPLE No. 2

Programming Example No. 1 may be extended to introduce a slightly greater degree of realism into the problem. For this purpose it will be assumed that transactions of both types (issue or receipt) may occur in the transaction file but only one transaction may occur for one reference message. It will further be assumed that transactions do not occur for all references in the file, which implies that transaction messages and reference messages must be compared and action taken only where Part Number agreement is found.

SPECIFICATION OF PROGRAMME

A reference file is mounted on Tape Unit 10 composed of fixed field messages with the following format:-

PART NO.	STOCK BALANCE	TOTAL ISSUES	TOTAL RECEIPTS	DATE
5	8	6	6	6

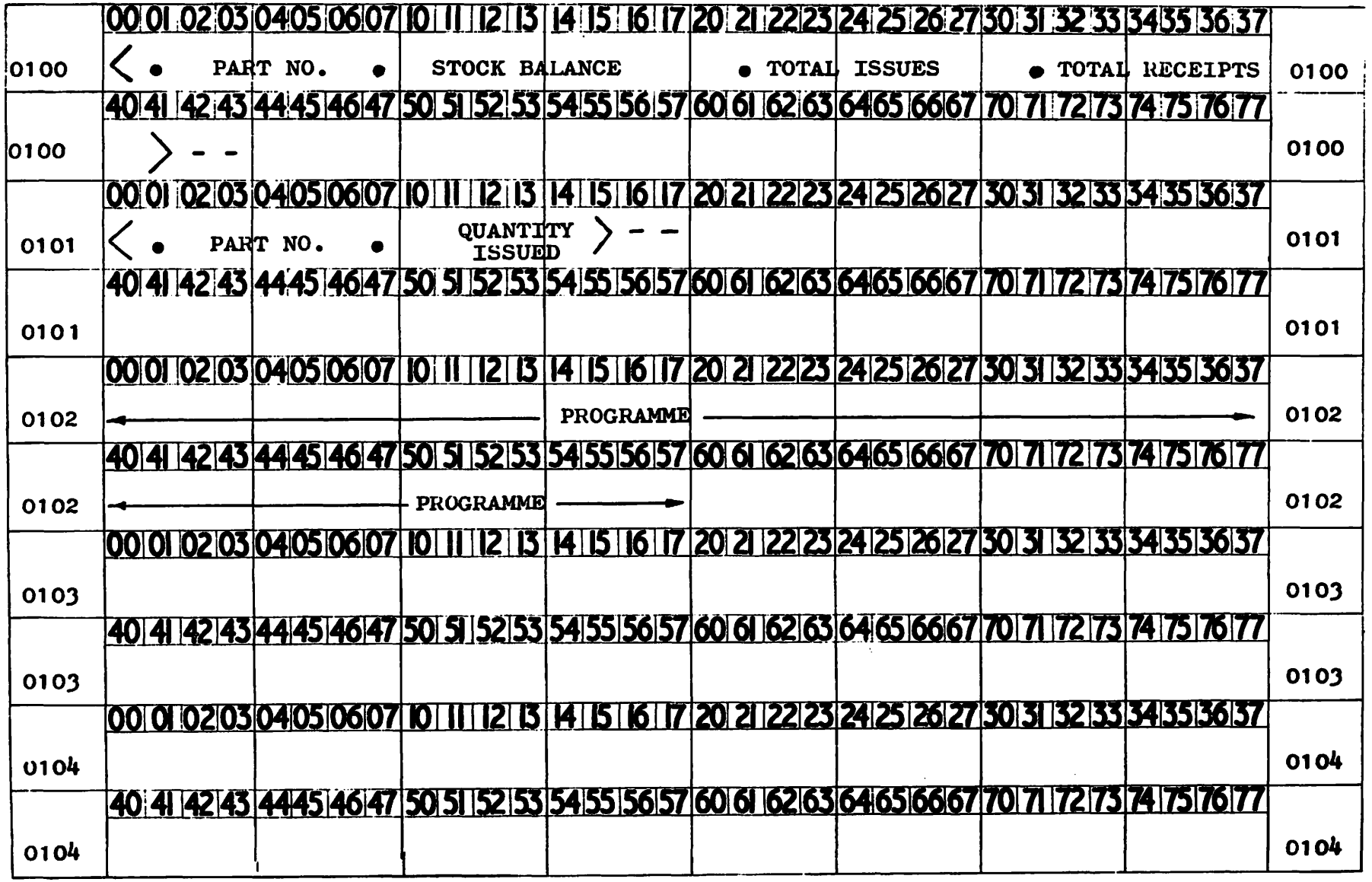
N.B. Sign positions are omitted from the STOCK BALANCE, TOTAL RECEIPTS and TOTAL ISSUES and all these items are extended to maximum length by the use of SPACE (01) symbols, i.e., a STOCK BALANCE of 3576 is held as • ---- 3576.

A transaction file is mounted on Tape Unit 20 composed of fixed field messages as follows:-

I / R	PART NO.	QTY. ISSUED OR RECEIVED
1	5	5

The transaction code is a single character I = ISSUE or R = Receipt.

KDP10 COMPUTER HSM RECORD



TITLE **PROGRAMMING EXAMPLE NO. 1**

Title PROGRAMMING EXAMPLE NO. 1

'ENGLISH ELECTRIC' KDP 10
COMPUTER PROGRAMME RECORD

Date 10th August 1961

Index No.

Block No.

From Instr. Loc.	H.S.M. Loc.	OO OP 0	A 1	A 2	A 3	OO N 4	B 5	B 6	B 7	Chart Ref.	Remarks	Constants
010250	0102 0	14	01	00	00	00	10	00	00	1	LRF. REF. MSG. R.I. Area	
	1	14	01	01	00	00	20	00	00	2	LRF. TRANS. MSG. R.I. Area	
	2	51	01	00	30	00	01	01	14	3	DA. TOT. ISSUES + ISSUES → TOT. ISSUES	
	3	52	01	00	20	00	01	01	14	4	DS. STK. - ISSUES → STK.	
	4	12	01	00	00	00	30	00	00	5	LW. REF. MSG.	
	5	71	01	02	00	00	00	00	00	6	TC. TO CHT. REF. 1	
	6											
	7											
	0103 0											
	1											
	2											
	3											
	4											
	5											
	6											
	7											
	0104 0											
	1											
	2											
	3											
	4											
	5											
	6											
	7											

EEDPCS 91 H.S.M. Loc. Coded by Sheet 1 of 1

The reference file must be updated and written out to a new tape mounted on Tape Unit 30. All updated messages are to have the date changed to today's date which must be read in from paper tape with the format:

<YRMODA>

PROGRAMME PLAN

The original six instructions are not sufficient and further instructions are now introduced as follows:

(A)	OCT	ONE CHARACTER TRANSFER	22	(Page 12.3)
(B)	RWD	REWIND TO B.T.C.	17	(Page 11.16)
(C)	SCC	SECTOR CLEAR BY CHARACTER	34	(Page 13.11)
(D)	SCT	SECTOR CLEAR BY TETRAD	36	(Page 13.16)
(E)	SET	SET REGISTER	72	(Page 17.2)
(F)	STR	STORE REGISTER	73	(Page 17.3)
(G)	SC	SECTOR COMPARE	43	(Page 14.8)
(H)	CTC	CONDITIONAL TRANSFER OF CONTROL	61	(Page 16.1)
(J)	STC	SECTOR TRANSFER BY CHARACTER	24	(Page 12.4)
(K)	STT	SECTOR TRANSFER BY TETRAD	26	(Page 12.8)

Each of the above instructions should be studied before attempting to follow the coding for this example.

The broad plan for the programme will be to read in a message from each file. If the Part Numbers agree the transaction code is inspected, the appropriate action taken and the updated reference written out to Tape Unit 30. If the Transaction Part Number is **greater** than the Reference Part Number the reference message is written out unaltered and a new Reference Message is read in. If the Transaction Part Number is **less** than the Reference Part Number an error has occurred, since both files should be in ascending order of part number, requiring a run through both files in the forward direction only.

This overall plan is now filled out with detailed steps and a logical block diagram (Page 7.11) produced to indicate what instructions are required. Decisions will require to be taken about Read-in Areas, Work Areas and so on, and a H.S.M. Record is drawn up as shown on Page 7.11. Finally, the programme is coded in conjunction with the H.S.M. Record resulting in the Programme Record given on Pages 7.13 & 7.14. The Remarks section of the Programme Record includes references to the following explanatory notes:-

a) Tapes are rewound to BTC (BEGINNING OF TAPE CONTROL) to ensure that the first messages actually read are the first messages on the tapes.

b) Sector Compare requires two instructions, SET T and SC. The address of the rightmost location of the Transaction Part Number is placed in the T register. The A and B addresses of the Sector Compare instruction are the left and right-hand ends of the Reference Part Number. After the instruction the Previous Result Indicators are set.

c) The CTC instruction inspects the PRI settings left from the previous Sector Compare instruction. If PRP is set the programme transfers to an unspecified error stop (PES), if PRN is set the programme transfers to 010520 where a write-out instruction is located. If PRZ is set the programme takes the next instruction in sequence.

d) The Balance in the Reference read-in area has no sign position. It is, therefore, not possible to add to or subtract from it in its original location. For processing it is transferred to a work-area (WA1) where a sign location has been created in 010063 by the SCT instruction which cleared the work areas. Notice that the Balance is transferred **before** the test for I or R, since in either case action must be taken on the Stock Balance.

e) Set T and Sector Compare to check the transaction code for a letter I. In the KDP 10 Code an 'I' is $(50)_8$. This constant is stored in an otherwise unused character position in the coding, and SET T addresses this storage location at 010277. The A and B addresses of the SC instruction are the same and nominate the address of the transaction code.

f) Creates a sign location in WA2 for ISSUES.

g) Note that the A address of the DA instruction nominates the sign location in WA2.

h) The DA instruction places an ISS at the completion of the addition. All quantities are fixed field and this extra ISS should be deleted. The T register addresses the ISS on completion of Decimal Add and may be called upon as an address modifier in the OCT instruction immediately following the addition. The KDP 10 Code for a space is $(01)_8$ and this constant is stored in 010337, an otherwise unused location in the coding. The second OCT instruction replaces an ISS in the correct position.

j) The updated ISSUES are transferred to the read-in area from WA2 leaving the sign behind.

k) Note the sign location nominated by the A address.

l) The unwanted ISS could have been removed, as described in (h), immediately following the DS instruction. To illustrate technique it is removed after transfer to the read-in area. Following the IT instruction which transfers WA1 to the Balance position of the read-in area, the B register addresses a location one to the left of the unwanted ISS. The contents of B are stored in Address Modifier No. 1 (AM1) which is used by the OCT instruction to effect the transfer of the space constant. Notice that the B address of the OCT instruction is 000001, to compensate for the fact that the address in AM1 is one character position to the left of the ISS location. The second OCT instruction places an ISS in the correct position.

m) The form of today's date, YRMODA, does not contain an ISS. To transfer this to the DATA position of the read-in area it is necessary to use a Sector Transfer by Character. The T register is set to the rightmost location of the destination required while the A and B addresses of the STC instruction are the left and right ends of YRMODA respectively.

n) The updated reference message is written out to Tape Unit 30. The next operation requires new messages from both the transaction and reference files so control is transferred to the appropriate instruction at 010250, i.e., to Chart Reference 2.

p) The next instruction space on the coding sheet is available for one of the side paths, i.e., for sections of the programme which are not in the direct line down the centre of the logical diagram. These instructions write out the reference message and transfer control to read-in a new reference message as required by Chart References 12 and 13.

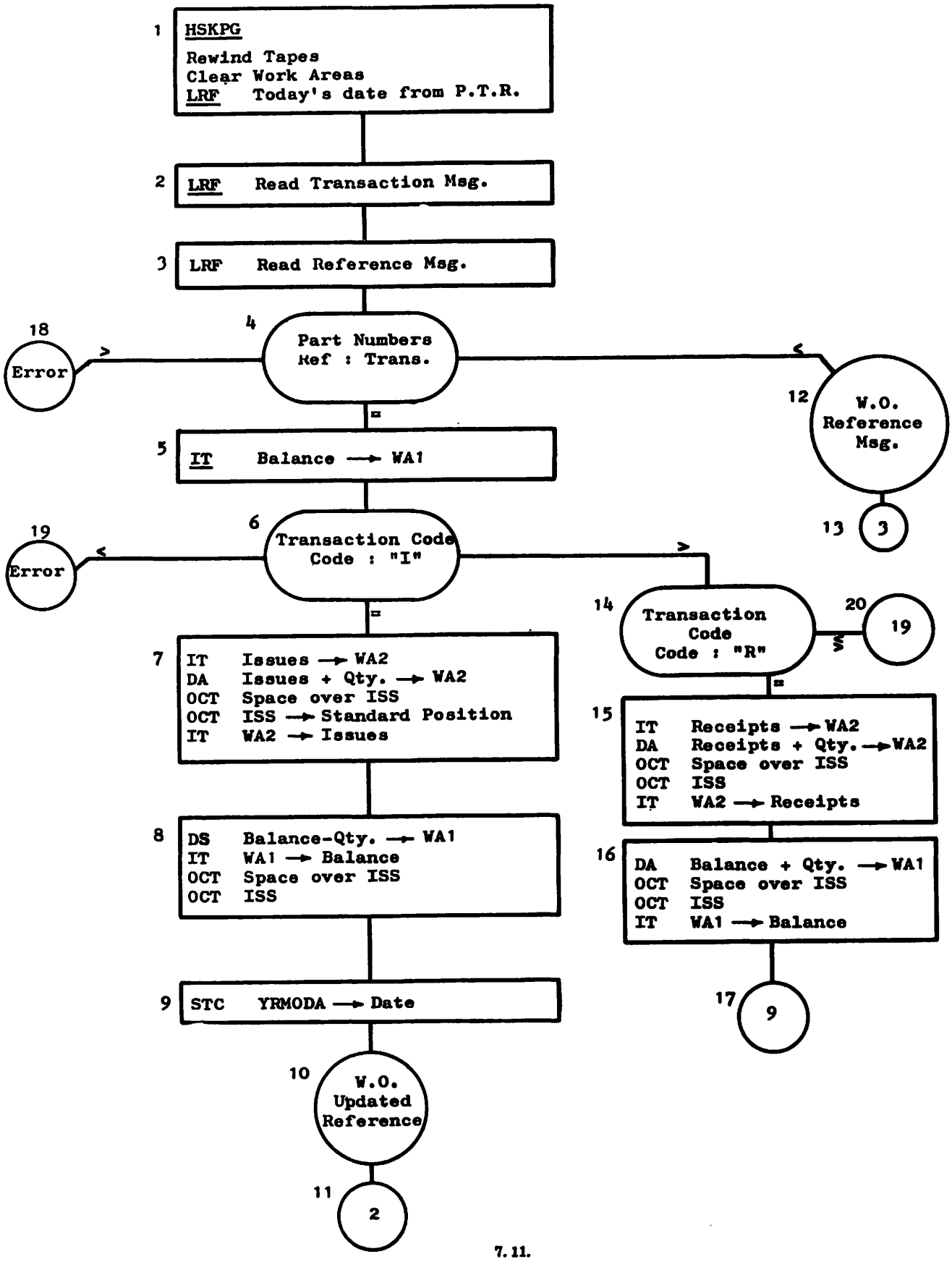
q) Another side-path section, that starting at Chart Reference 14. The first two instructions SET T and SC check the transaction code for a letter 'R'. To illustrate a useful dodge 'R', which is code $(61)_8$ is not stored deliberately, as was the case with 'I'. Instead a $(61)_8$ is 'borrowed' from the Operation Code location of a CTC instruction already in the programme.

r) All remaining instructions have been covered, so far as points of detail are concerned, by previous notes about similar instructions.

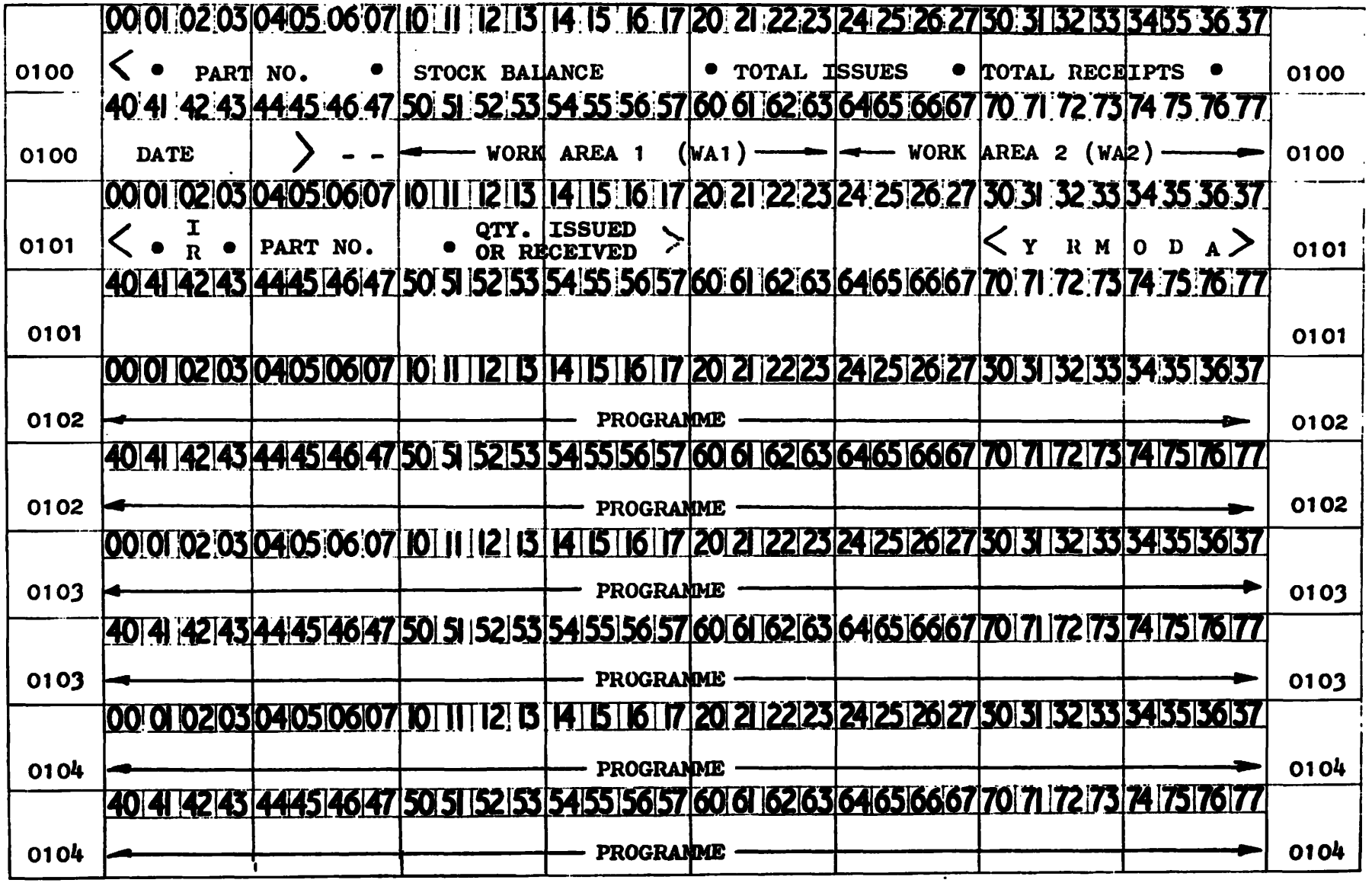
It is emphasised that the example described is a training example of an artificial kind. The KDP 10 can handle variable length messages and items whereas both examples have been restricted to fixed field data. This is done quite deliberately to allow the reader to appreciate the fundamental aspects of the subject before introducing him to the wide scope offered by the KDP 10 order code in relation to variable length working. It may be imagined that problems could arise if the reference messages were allowed

to vary in length from, say, 38 characters, as in Programming Example No. 2, to a maximum of 64 characters. WA1 and WA2 would need to be elsewhere or the longer messages would overlap the work areas. As for the problem of knowing where the right or left-hand ends of items are, that is quite a different matter. The solution is given in the next section which describes variable length programming.

PROGRAMMING EXAMPLE NO. 2. - LOGICAL FLOW DIAGRAM



KDP10 COMPUTER HSM RECORD



TITLE PROGRAMMING EXAMPLE NO. 2

Title PROGRAMMING EXAMPLE NO.2

'ENGLISH ELECTRIC' KDP 10
COMPUTER PROGRAMME RECORD

Date 10th August, 1961.
Index No.
Block No.

From Instr. Loc.	H.S.M. Loc.	OO				OO				Chart Ref.	Remarks			Constants
		OP	A			N	B							
		0	1	2	3	4	5	6	7					
	0102 0	17	00	00	00	00	10	00	00	1	RWD.	TK 10 REF. INPUT		
	1	17	00	00	00	00	20	00	00	1	RWD.	TK 20 TRANS. INPUT	(a)	
	2	17	00	00	00	00	30	00	00	1	RWD.	TK 30 REF. OUTPUT		
	3	36	01	00	50	00	01	00	77	1	SCT.	CLEAR WORK AREAS		
	4	14	01	01	30	00	77	00	00	1	LRF.	TODAY'S DATE FROM P.T.R.		
010530	5	14	01	01	00	00	20	00	00	2	LRF.	TRANSACTION MSG.		
010550	6	14	01	00	00	00	10	00	00	3	LRF.	REFERENCE MSG.		
	7	72	01	01	10	00	60	00	50	4	SET T.	TRANS. PART NO.	(b)	I
	0103 0	43	01	00	02	00	01	00	06	4	SC.	PART NOS. REF: TRANS.		
	1	61	P	E	S	00	01	05	40	4	CTC.	+ PES - WO REF. MSG.	(c)	
	2	21	01	00	17	00	01	00	62	5	IT.	BALANCE → WA1	(d)	
	3	72	01	02	77	00	60	00	01	6	SET T.	"I"	(e)	SP
	4	43	01	01	02	00	01	01	02	6	SC.	CODE: "I"		
	5	61	01	05	60	00	P	E	S	6	CTC.	+ TEST FOR "R", - PES		
	6	21	01	00	26	00	01	00	76	7	IT.	ISSUES → WA2	(f)	
	7	51	01	00	77	00	01	01	16	7	DA.	WA2 + QTY. → WA2	(g)	
	0104 0	22	01	03	37	06	00	00	00	7	OCT.	SPACE OVER ISS		
	1	22	01	04	57	00	01	00	70	7	OCT.	ISS TO STANDARD POSITION	(h)	
	2	21	01	00	76	00	01	00	26	7	IT.	WA2 → ISSUES	(j)	
	3	52	01	00	63	00	01	01	16	8	DS.	WA1 - QTY. → WA1	(k)	
	4	21	01	00	62	00	01	00	17	8	IT.	WA1 → BALANCE		
	5	73	00	01	13	00	30	00	74	8	STR B.	LOCATION OF ISS - $(01)_8$		ISS
	6	22	01	03	37	01	00	00	01	8	OCT.	SPACE OVER ISS	(e)	
	7	22	01	04	57	00	01	00	07	8	OCT.	ISS TO STANDARD POSITION		

EEDPCS 3 H.S.M. Loc. Coded by Sheet 1 of 2

Title PROGRAMMING EXAMPLE NO.2

'ENGLISH ELECTRIC' KDP 10
COMPUTER PROGRAMME RECORD

Date 10th August, 1961.
Index No.
Block No.

From Instr. Loc.	H.S.M. Loc.	OO				OO				Chart Ref.	Remarks		Constants
		OP	A			N	B						
		0	1	2	3	4	5	6	7				
010720	0105 0	72	01	00	44	00	60	00	00	9	SET T	RH LOCATION OF DATE (m)	
	1	24	01	01	31	00	01	01	36	9	STC	YRMODA → DATE	
	2	12	01	00	00	00	30	00	00	10	LW	UPDATED REFERENCE (n)	
	3	71	01	02	50	00	00	00	00	11	TC	TO READ TRANS.	
010310-	4	12	01	00	00	00	30	00	00	12	LW	REFERENCE (p)	
	5	71	01	02	60	00	00	00	00	13	TC	TO READ REFERENCE	
010350+	6	72	01	03	50	00	60	00	00	14	SET T	"R"	
	7	43	01	01	02	00	01	01	02	14	SC	CODE : "R" (q)	
	0106 0	61	P	E	S	00	P	E	S	14	CTC	+ PES	
	1	21	01	00	35	00	01	00	76	15	IT	RECEIPTS → WA2	
	2	51	01	00	77	00	01	01	16	15	DA	WA2 + QTY. → WA2	
	3	22	01	03	37	06	00	00	00	15	OCT	SPACE OVER ISS	
	4	22	01	04	57	00	01	00	70	15	OCT	ISS TO STANDARD POSITION	
	5	21	01	00	76	00	01	00	35	15	IT	WA2 → RECEIPTS	
	6	51	01	00	63	00	01	01	16	16	DA	WA1 + QTY. → WA1 (r)	
	7	22	01	03	37	06	00	00	00	16	OCT	SPACE OVER ISS	
	0107 0	22	01	04	57	00	01	00	52	16	OCT	ISS TO STANDARD POSITION	
	1	21	01	00	62	00	01	00	17	16	IT	WA1 → BALANCE	
	2	71	01	05	00	00	00	00	00	17	TC	TO TRANSFER YRMODA	
	3												
	4												
	5												
	6												
	7												

EEDPCS 31 H.S.M. Loc. Coded by Sheet 2 of 2

8

PROGRAMMING VARIABLE LENGTH DATA

GENERAL INTRODUCTION

One of the important features of the KDP 10 is its ability to handle variable length data. The fixed-word length concept of many other computers is replaced by the more appropriate idea of variable length items, messages, sectors or blocks. This is made possible by having every character in the HSM addressable and by the use of special control characters (ISS, SM, EM, etc.) to define the position of items within messages and to define the bounds of messages themselves.

Variable length data raises two main problems (1) the number of characters in a message or item is unknown, which chiefly affects timing calculations and (2) the exact address of a particular unit of information is not fixed. It is this last aspect of variable length data which causes difficulty to programmers accustomed to fixed-word length computers. The difficulty arises from the belief that it is only possible to write a programme if the addresses of all relevant data are known to the programmer. In a variable length data machine the address of a particular item may vary for every message processed. How then is it possible to specify an operation such as addition or transfer for an item whose position is variable? It all depends on what is meant by position. The item position within a message must remain fixed but the address of the item may vary depending on the number of characters which precede it in earlier items of the meesage. Provided this requirement is satisfied the KDP 10 can co-operate with the programmer and provide the exact address of any particular item.

The logical design of the KDP 10 System includes two other features which make variable length programming relatively simple. These are the provision of addressable control registers and the provision of special instructions in the order-code.

In fixed-field programming the programmer keeps a record of all data locations. In variable length programming the programmer keeps such record of data as is possible and the KDP 10 does the rest. After every instruction the KDP 10 leaves evidence in its control registers about the data length of items or sectors referred to by the instruction, and this information is both useful and accessible to the programmer.

DATA DESCRIPTIONS

Before writing a programme for KDP 10 certain necessary preliminary operations must be carried out. File descriptions must be provided for all files involved in a programme and these must include statistics about the number of entries in the file, date of compilation and so on, and must include details of all messages in the file. An example of a file description is given on Page 8.3. It will be noted that the body of the form lists all the items contained in each message of the file and includes figures for the average number and maximum number of characters in each message.

ITEM NUMBER. Each item in a message is given a number for listing convenience. The order of items in all messages of the file must be the order shown on the data description form.

SUB ITEM. Parts of items are designated as sub-items. These are not items by the KDP 10 definition since they do not carry Item Separator Symbols. They must, however, be what is known as Fixed-and-Always-Appearing (FAA).

FAA ITEMS AND SUB ITEMS. A fixed-and-always-appearing item is one which always has a constant number of characters (i.e. fixed in length, always occupies the same item (or sub-item) position (i.e. fixed in position) and which always appears in all messages.

DESCRIPTION. Suitable names are provided for each item and sub-item in the message.

NUMBER OF CHARACTERS. The average number of characters determines the overall processing time. The maximum number of characters determines the storage space which must be provided in the HSM to accommodate the longest message which is likely to occur. Note that control characters are not included in the character count but space for these must be provided in the HSM storage area.

% OCCURRENCE. This column lists the frequency of occurrence of each item as a percentage. Items with an occurrence less than 100% must appear as the last items of a message.

WTD. AVERAGE. This column lists the weighted average of the number of characters in each item. The figure is obtained from:

$$\text{WTD. AVERAGE} = \frac{\text{AVG. NO. OF CHARACTERS} \times \text{\% OCCURRENCE}}{100}$$

TOTAL CHARACTERS. The grand total of characters in a message is the sum of the total information characters and the KDP 10 control characters. These will consist usually of one ISS for every item together with a Start Message (SM) and End Message Character. The total character count determines the HSM storage locations which must be reserved for the message read-in area.

The following observations are made as a guide to programmers:-

1. ALL FAA ITEMS such as PART NUMBER, POLICY NUMBER, TRANSACTION CODE etc., should appear as the **first items of a message.**

2. Items should be arranged in descending order of % occurrence, FAA items first, then 100% items and then non 100% items.

3. It is not necessary to retain the ISS symbols for non 100% items beyond the last items for which information exists.

VARIABLE LENGTH TECHNIQUES

FINAL REGISTER SETTINGS. Throughout the execution of an instruction the contents of the A, B or T registers change as the instruction proceeds. The final contents of these registers may be used by the programmer in subsequent instructions, through the use of instruction modification. For example, after a LINEAR READ FORWARD instruction the final A register setting contains the address of the End Message. The address of the right hand end of the last item in the message (irrespective of length) is then given by:

$$(A)_f \quad -(01)_8$$

where $(A)_f$ is the final content of the A register after LRF. If $(A)_f$ is stored immediately following the LRF instruction (by the next instruction) it may be used subsequently in the programme to address the last item of the message.

Another example occurs after DECIMAL SUBTRACT when ZERO SUPPRESS is required. After the subtraction of two operands the difference may be of unknown length and may contain non-significant zeros. The ZERO SUPPRESS instruction will operate if it addresses zeros and it is essential to cause the ZERO SUPPRESS instruction to address the character immediately to the right of the ISS of the difference.

Following DECIMAL SUBTRACT the T register addresses the ISS in the difference and the address of the first character required by ZERO SUPPRESS is $(T)_f + (01)_8$.

Knowledge of the final contents of registers is a necessary step towards the development of advanced programming techniques. Some useful Programming devices which use final register contents are the following:-

Example 1.

The sign location of a minuend is 010273, and of the subtrahend 010446. It is required to ZERO SUPPRESS the difference following a DECIMAL SUBTRACT instruction.

'ENGLISH ELECTRIC' KDP 10

E.D.P. DATA SHEET

☐ INPUT ☐ REFERENCE	☐ OUTPUT ☐ INTERMEDIATE	IDENTIFICATION STOCK INVENTORY			DATE 10.8.61
Source		USE			SCHEDULE
SEQUENCE ASCENDING PART NUMBER		DISPOSITION		NO. OF MESSAGES PEAK:	/NORMAL: 20,000
DOCUMENT DESCRIPTION TYPE:		/SIZE:	WIDE X	LONG/NO. OF COPIES	
Remarks					

Item No.	Sub. No.	DESCRIPTION	No. of Chars. MAX	No. of Chars. AVG.	% occurrence	Wtd. Avg.
1		PART NUMBER	7	7	100	7
2		REORDER CODE	1	1	100	1
3		SUPPLIER - NAME	20	15	100	15
4		SUPPLIER - ADDRESS STREET	30	18	100	18
5		SUPPLIER - ADDRESS TOWN, COUNTY	25	16	100	16
6		BALANCE ON HAND	10	8	100	8
7		BALANCE ON ORDER	6	4	100	4
8		LEAD TIME	9	7	100	7
9		TOTAL ISSUED TO DATE	10	7	100	7
10		PRICE	6	4	100	4
11		PURCHASE TAX CODE	2	2	50	1
12		OWN GOODS CODE	1	1	40	0.4
		TOTAL CHARACTERS OF INFORMATION	127			88.4
		ADDED CONTROL CHARACTERS	14			13.3
		TOTAL CHARACTERS	141			101.7

The instructions are:

DS	51	010273	00	010446
ZS	32	000001	60	010273

The T register addresses the ISS after the DS instruction is complete and this is used as the A Address Modifier in the following ZS instruction. The constant 000001 in the A address ensures that the left-hand end of the sector suppressed is one character position to the right of the ISS of the difference.

Example 2.

To place an ISS immediately to the left of the most significant figure of a number which has been zero-suppressed.

The instructions are:

DS	52	010273	00	010446
ZS	32	000001	60	010273
OCT	22	Address of ISS	02	000000

Following a zero-suppress operation the A register addresses the last zero suppressed. This is where the ISS should be placed and the A register is available as an Address Modifier in an OCT instruction immediately following the ZS instruction. The ISS is stored as the constant $(74)_8$ at some otherwise unused location in the coding.

Example 3.

To store the ISS at the most significant end of a product formed by the DECIMAL MULTIPLY instruction. The sign of the product is in 012340, the multiplicand is at 010222 and the multiplier at 011377.

The instructions are:

SET	72	012340	00	600000
DM	53	010222	00	011377
OCT	22	Address of ISS	06	000000

The programmer does not know the exact length of the product, which in any case varies from one multiplication to the next. The T register always knows the address one to the left of the MSD of the product and the programmer is thus able to call up the T register in the OCT instruction which places the ISS.

Example 4.

To store the contents of STA after a LINEAR READ FORWARD instruction and use the address in a subsequent instruction to item transfer the last item of the Message.

LRF	14	010000	00	100000
TCT	25	000223	00	000113
	.	.	.	.
	.	.	.	.
	.	.	.	.
	.	.	.	.
	.	.	.	.
IT	21	777777	10	012043

The LRF reads a message from Tape Unit 10 to the HSM placing the SM in 010000. The TCT (THREE CHARACTER TRANSFER) instruction transfers the contents of STA (000223) to Address Modifier 1 (000113). The IT transfer instruction nominates AM1, which contains the address of the EM of the message, and subtracts 1 from the address (by adding 777777) before using it to transfer the final item of the message to 012043.

Example 5.

To locate the address of an SM after SECTOR COMPRESS and use this as the first character in a LINEAR WRITE instruction.

SET	72	011157	00	600000
SCR	35	011000	00	011157
LW	12	000001	60	300000

After SCR (or SCD) the T register addresses the character one position to the left of the SM. The LW instruction nominates the T register as Address Modifier 6 adding 000001 to address the SM.

VARIABLE LENGTH INSTRUCTIONS

Four specially designed instructions in the order-code for use with variable length data are:

LNS	LOCATE n^{th} SYMBOL IN SECTOR	31	Page 13.1
RD	RANDOM DISTRIBUTE	27	Page 12.10
SCR	SECTOR COMPRESS RETAIN REDUNDANT ISS	35	Page 13.14
SCD	SECTOR COMPRESS DELETE REDUNDANT ISS	37	Page 13.18

Some description of the techniques applicable to each of these will be given.

LNS, LOCATE n^{th} SYMBOL IN SECTOR, 31

Suppose it is required to find the 5th item of a message. The right-hand end of the item is one character position to the left of the 6th ISS counting from the SM of the message. The instruction LNS is designed for just such an occasion. The final content of the A register (in STA) addresses the character one to the left of the n^{th} ISS.

LNS should be used intelligently. If eight items are to be located it would be a waste of time to use LNS eight times. Other techniques exist through the use of RANDOM DISTRIBUTE and in general LNS should be used to locate isolated items on occasions when RANDOM DISTRIBUTE is not essential.

RD. RANDOM DISTRIBUTE. 27

It is frequently necessary to process several items in a variable length message. When this requirement exists,the message, occupying less than maximum storage space, is expanded and transferred to part of the HSM designated as a Random Distribute Area, in such a way that all the items are spaced out in fixed positions, usually the positions which they would occupy if each item contained the maximum number of characters. From this point the message may be treated as a fixed-field message in which the address of all items is determined.

Random Distribute is, however, very much more powerful and useful. The items of a message may be re-arranged in the RD area and any items needed for processing are distributed to work-areas. Sufficient space may be reserved in the RD area for items which are taken out to work areas so that these items may be transferred back into the message after processing is complete.

Several programming rules must be observed in connection with Random Distribute. These are:-

1. ALWAYS CLEAR RD AND WORK AREAS BEFORE DISTRIBUTING A MESSAGE. This ensures that all character locations of the RD and work areas not occupied by distributed characters contain space symbols.

2. ALWAYS LEAVE EXTRA CHARACTERS FOR SIGNS IN THE RD AREA. This applies to items distributed to work areas and which are later returned to the RD area by transfer instructions.

3. ALWAYS LEAVE EXTRA CHARACTERS FOR SIGNS IN WORK AREAS.

4. NEVER PROCESS ITEMS IN RD AREAS. There is a danger of creating extra ISS symbols and spurious items unless this rule is observed.

5. LEAVE THE TRANSFER OF WORK AREA ITEMS TO THE RD AREA AS LATE AS POSSIBLE. This is another rule to avoid the creation of spurious items and it will be found preferable to return the items immediately before the point in the programme at which the message is compressed and written out.

Two important points about RANDOM DISTRIBUTE are the ability to throw away items in the original area, by the use of throw-away addresses, 777777, and the special treatment of the control symbol EM. It will be recalled that some messages may contain all items whereas in other messages some or all in the less than 100% occurrence category may be missing. Sufficient distribution addresses must be provided for messages containing all items. When an EM is detected and further distribution addresses are still unused, the RD instruction rightly assumes these belong to items of less than 100% occurrence and it distributes ISS symbols in the remaining address positions.

THE SECTOR COMPRESS INSTRUCTIONS

After a message has been distributed to an RD area and all items returned from work areas to the RD area, the area contains many unwanted space characters. These are removed by the use of SECTOR COMPRESS RETAIN if redundant ISS symbols are to be retained, or by SECTOR COMPRESS DELETE if the ISS symbols of items not containing information are to be deleted. SCR and SCD are complementary to Random Distribute. The latter expands messages to maximum length in order to fix the addresses of items at the expense of increased message length, while SCR and SCD return the expanded message to a length determined by the information characters only.

The problem of knowing the address of the SM in the compressed message has been mentioned in Example 5 in the discussion of the usefulness of final register contents.

SAMPLE INVENTORY PROBLEM

I. STATEMENT:

This problem is to update an inventory stock record master file with transactions of two types. There are Receipt Transactions (R) and Issue Transactions (I).

II. ASSUMPTION:

A. There are 10,000 Reference messages; 7500 Issue transactions; and 1500 Receipt transactions.

B. The Reference File, message format on tape, has been completely verified and contains no error. It is sorted in ascending order, by stock number sequence.

C. Transactions are all on one tape and sorted in ascending order, by stock number and identification code.

D. A transaction message refers to no more than one reference message and there can be only one type of transaction against a reference message.

III. MESSAGES:

A. Reference Message:

(1)	(2)	(3)	(4)	(5)
Max. 1	Max. 3	Max. 9	Max. 8	Max. 30
ID. Code	PROC. Code	STOCK No.	BAL. ON HAND	MFG. NAME
Avg. 1	Avg. 3	Avg. 9	Avg. 6	Avg. 14
100%	100%	100%	100%	100%

(6)	(7)	(8)
Max. 16	Max. 36	Max. 10
ADDRESS (ST)	ADDRESS (City & County)	Total Issued (TITD)
Avg. 12	Avg. 12	Avg. 7
100%	100%	100%

(9)	(10)
Max. 7	Max. 12
REORDER LEVEL	LEAD TIME
Avg. 3	Avg. 2
100%	50%

B. Receipt Transactions:

(1)	(2)	(3)	(4)
Max. 1	Max. 3	Max. 9	Max. 7
ID Code	PROC. Code	STOCK No.	QUANTITY
Avg. 1	Avg. 3	Avg. 9	Avg. 4

C. Issue Transactions:

Max. 1	Max. 3	Max. 9	Max. 7
ID Code	PROC. Code	STOCK No.	QUANTITY
Avg. 1	Avg. 3	Avg. 9	Avg. 4

IV. PROBLEM:

A. Read in Reference Message from TK 10, starting at 010000.

B. Read in Transaction Message from TK 20, starting at 010300.

C. Compare Reference Message Stock Number to Transaction Message Stock number.

1. If Transaction Less than Reference:
   This indicates an error path; place an 'A' in ID code, write out Transaction Message to TK 30 and read in next transaction.
2. If Transaction Greater than Reference:
   This would indicate a write-out of the Reference Message to TK 40. Read in next reference.
3. If Transaction Equal to Reference:
   This indicates the main path to be followed.

D. Determine type of transaction, sense for I (issue).

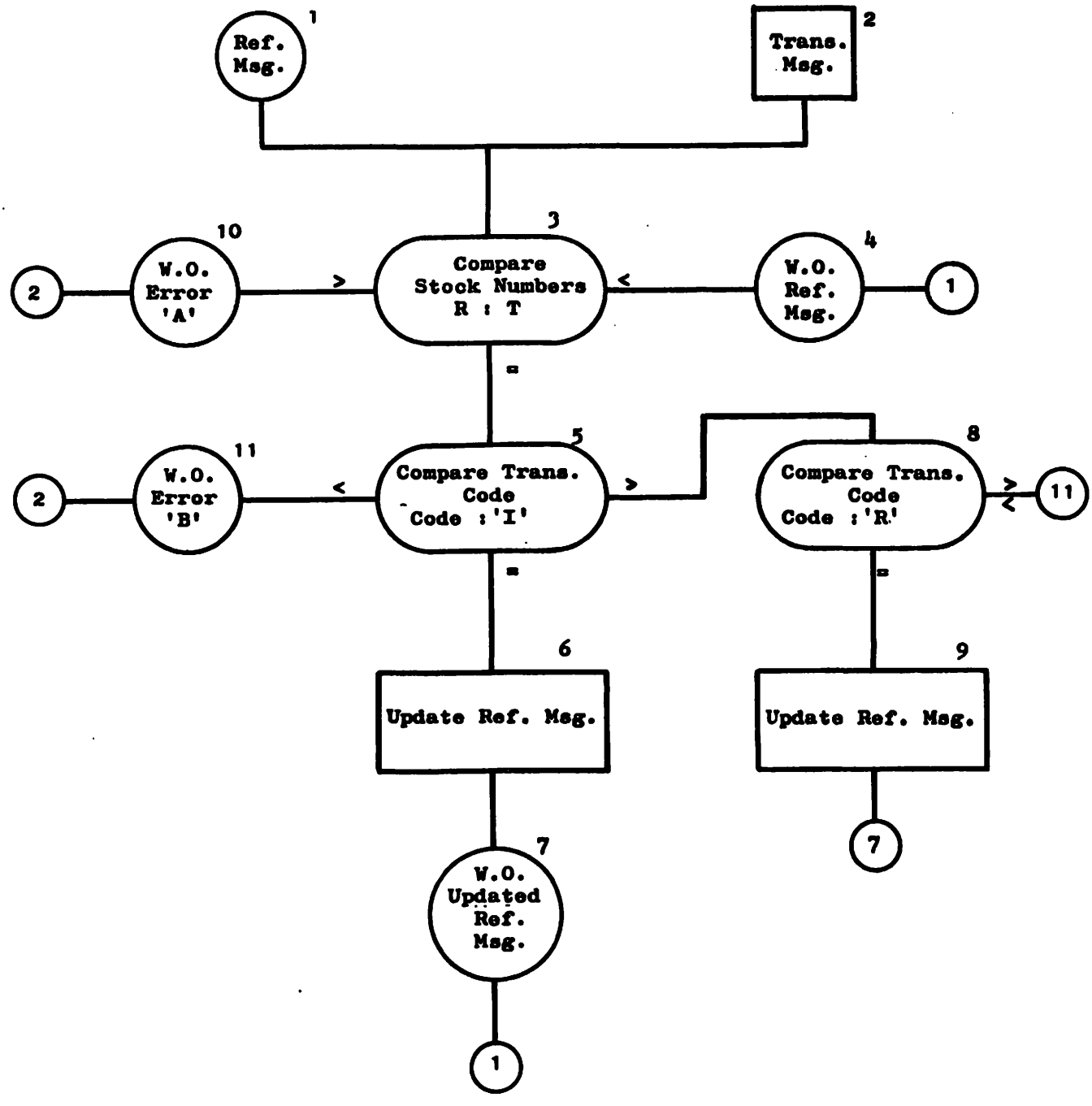
1. ID Code Less than Constant I:
   This indicates an error path; place a 'B' in ID Code, write out Transaction Message to TK 30 and read in next Transaction.
2. ID Code Greater than Constant I:
   This indicates the need to sense for R.
   (a) If not an 'R', this indicates an error path; place a 'B' in ID code, write out Transaction Message to TK 30 and read in next transaction.
   (b) If an 'R', update BOH, write out updated reference message to TK 40 and read in next reference.
3. ID Code Equal to Constant I:
   This indicates the main path to be followed; update BOH and TITD.
   Write out updated reference message to TK 40 and read in next reference.

E. Start coding at 011500 with housekeeping.

V. ASSIGNMENT:

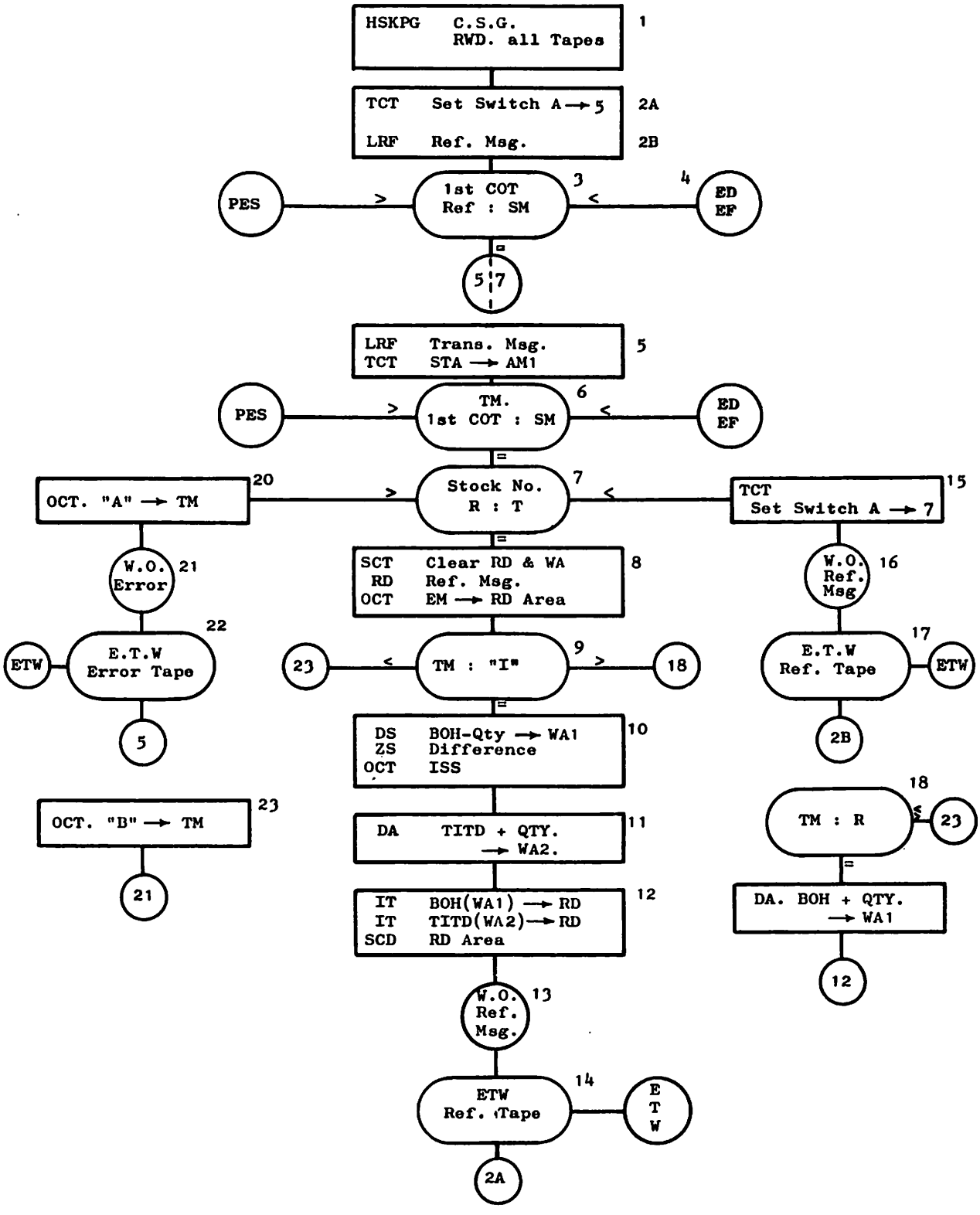
1. Data Sheets.
2. Functional Chart.
3. Detail Process Chart.
4. Code Main Paths.
5. High-Speed Memory Layout.

FUNCTIONAL FLOW CHART

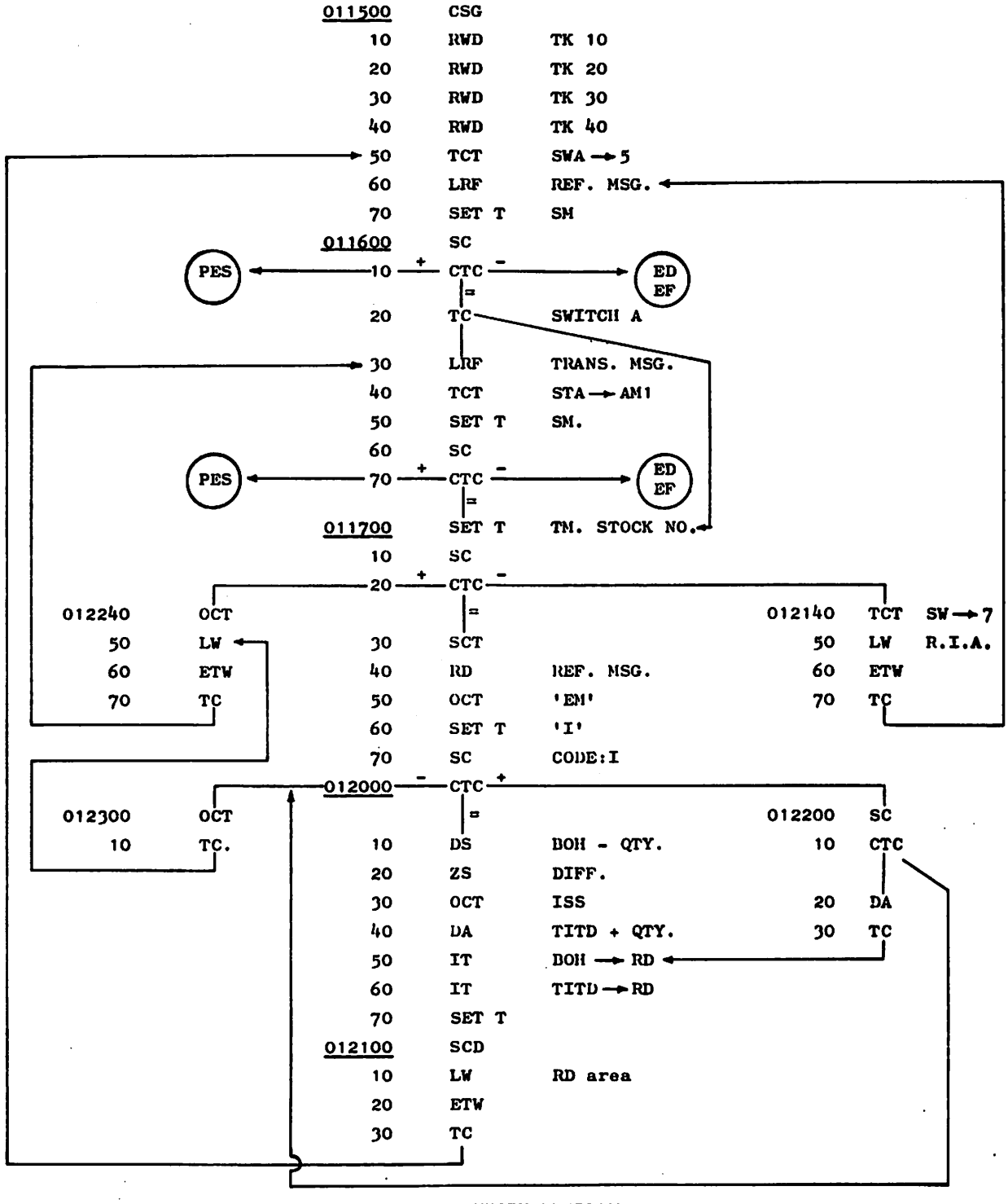


SAMPLE INVENTORY PROBLEM

DETAIL FLOW CHART



SAMPLE INVENTORY PROBLEM



SAMPLE INVENTORY PROBLEM

'ENGLISH ELECTRIC' KDP 10

E.D.P. DATA SHEET

	INPUT		OUTPUT	IDENTIFICATION	DATE
	REFERENCE		INTERMEDIATE	SAMPLE INVENTORY PROBLEM	10.8.61
Source		USE			SCHEDULE
		MASTER REFERENCE FILE			
SEQUENCE		DISPOSITION		NO. OF MESSAGES	
				PEAK:	/NORMAL:
DOCUMENT DESCRIPTION					
TYPE:		/SIZE:	WIDE X	LONG/NO. OF COPIES	

Remarks

MESSAGE FORMAT.

Item No.	Sub. No.	DESCRIPTION	No. of Chars. MAX	No. of Chars. AVG.	% occurrance	Wtd. Avg.
1		IDENTIFICATION CODE	1	1	100	1
2		PROCUREMENT CODE	3	3	100	3
3		STOCK NUMBER	9	9	100	9
4		BALANCE ON HAND	8	6	100	6
5		MANUFACTURER'S NAME	30	14	100	14
6		ADDRESS, STREET	16	12	100	12
7		ADDRESS, TOWN, COUNTY	36	12	100	12
8		TOTAL ISSUED TO DATE	10	7	100	7
9		REORDER LEVEL	7	3	100	3
10		LEAD TIME	12	2	50	1
		TOTAL CHARACTERS OF INFORMATION	132			68
		ADDED CONTROL CHARACTERS	12			11.5
		TOTAL CHARACTERS	144			79.5

'ENGLISH ELECTRIC' KDP 10

E.D.P. DATA SHEET

INPUT	OUTPUT	IDENTIFICATION	DATE
REFERENCE	INTERMEDIATE	SAMPLE PROBLEM	10.8.61

Source	USE	SCHEDULE
	TRANSACTION FILE	

SEQUENCE	DISPOSITION	NO. OF MESSAGES
		PEAK: /NORMAL:

DOCUMENT DESCRIPTION
TYPE: /SIZE: WIDE X LONG/NO. OF COPIES

Remarks

Item No.	Sub. No.	DESCRIPTION	No. of Chars. MAX	AVG.	% occurrance	Wtd. Avg.
1		IDENTIFICATION CODE	1	1	100	1
2		PROCUREMENT CODE	3	3	100	3
3		STOCK NUMBER	9	9	100	9
4		QUANTITY	7	4	100	4

	MAX	Wtd. Avg.
TOTAL CHARACTERS OF INFORMATION	20	17
ADDED CONTROL CHARACTERS	6	6
TOTAL CHARACTERS	26	23

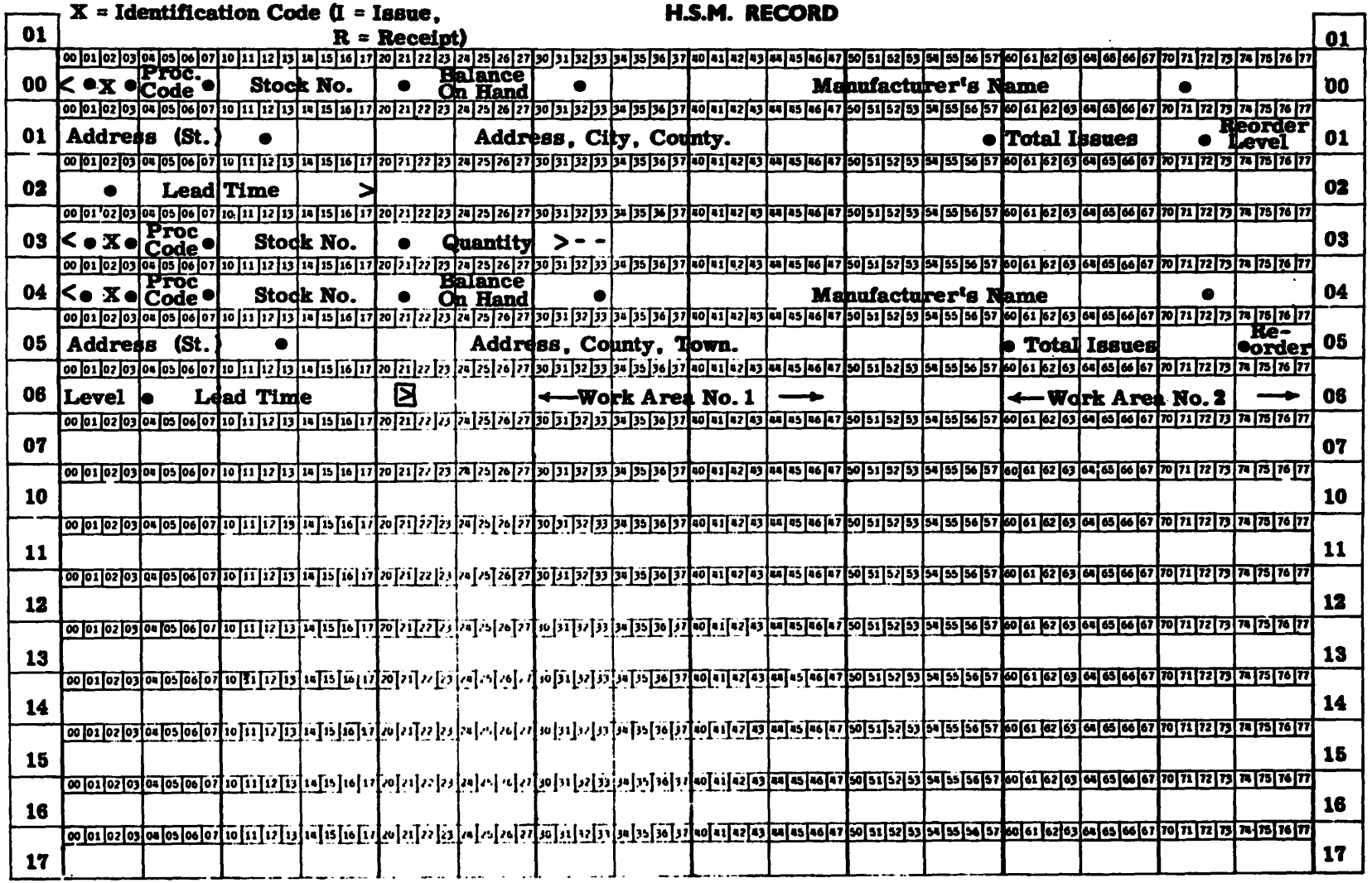
EEDPCS 23

Page of

'ENGLISH ELECTRIC' KDP 10

X = Identification Code (I = Issue, R = Receipt)

H.S.M. RECORD



TITLE: Sample Problem (Inventory) BLOCK NO.: INDEX NO: PROGRAMMER: DATE: PAGE OF

Title SAMPLE INVENTORY PROBLEM A

'ENGLISH ELECTRIC' KDP 10
COMPUTER PROGRAMME RECORD

Date 10.8.61.
Index No.
Block No.

From Instr. Loc.	H.S.M. Loc.	OP	A			N	B			Chart Ref.	Remarks	Constants
		OO				OO						
		0	1	2	3	4	5	6	7			
	0115 0	75	00	00	00	00	01	00	00	1	CSG.	
	1	17	00	00	00	00	10	00	00	1	RWD. TK10	
	2	17	00	00	00	00	20	00	74	1	RWD. TK20 ISS	
	3	17	01	17	00	00	30	00	00	1	RWD. TK30 Switch A path 7	
	4	17	01	16	30	00	40	00	00	1	RWD. TK40 Switch A path 5	
012130	5	25	01	15	43	00	01	16	23	2	TCT. Set switch A to path 5	
	6	14	01	00	00	00	10	00	00	2	LRF. Ref. Msg.	
	7	72	01	15	77	00	60	00	76	3	SET T. SM	
	0116 0	43	01	00	00	00	01	00	00	3	SC. 1st COT RM:SM	
	1	61	P	E	S	00	ED	/EF		3	CTC. + PES - ED/EF	
	2	71	00	00	00	00	00	00	00	3	TC. Switch A	
012270	3	14	01	03	00	00	20	00	00	5	LRF. Trans. Msg.	
	4	25	00	02	23	00	00	01	13	5	TCT. STA → AM1. Loc^{n}. Sign/LSD QTY.	
	5	72	01	15	77	00	60	00	00	6	SET T. Loc. SM	
	6	43	01	03	00	00	01	03	00	6	SC. 1st COT:SM	
	7	61	P	E	S	00	ED/	EF		6	CTC. + PES - ED/EF	
	0117 0	72	01	03	20	00	60	00	75	7	SET T. Trans. Msg. Stock No.	
	1	43	01	00	10	00	01	00	20	7	SC. RM:TM Stock Nos.	
	2	61	01	22	40	00	01	21	40	7	CTC. + Error A - WO Ref. Msg.	
	3	36	01	04	00	00	01	06	77	8	SCT. RD & WA1 & WA2	
	4	27	01	00	00	00	01	23	20	8	RD. Ref. Msg.	
	5	22	01	17	07	00	01	06	21	8	OCT. EM → RD area	
	6	72	01	17	67	00	60	61	50	9	SET T. "I"	
	7	43	01	03	02	00	01	03	02	9	SC. TM Code: I	

8.15.

FEDPCS 31 H.S.M. Loc. Coded by Sheet 1 of 3

Title SAMPLE INVENTORY PROBLEM A.

'ENGLISH ELECTRIC' KDP 10
COMPUTER PROGRAMME RECORD

Date 10.8.61.
Index No.
Block No.

From Instr. Loc.	H.S.M. Loc.		OP 0	A 1	A 2	A 3	N 4	B 5	B 6	B 7	Chart Ref.	Remarks		Constants
			OO				OO							
	0120	0	61	01	22	00	00	01	23	00	9	CTC.	+ Code > I → Sense R - Code < I Error	
		1	52	01	06	47	01	77	77	77	10	DS.	BOH - QTY → WA1	
		2	32	00	00	01	60	01	06	47	10	ZS.	Difference	
		3	22	01	15	27	02	00	00	00	10	OCT.	ISS	
		4	51	01	06	77	01	77	77	77	11	DA.	TITD + QTY. → WA2	
012230		5	21	01	06	47	00	01	04	32	12	IT.	BOH → RD area	
		6	21	01	06	77	00	01	05	73	12	IT.	TITD → RD area	
		7	72	01	06	20	00	60	41	40	12	SET T.	RH RD area	
	0121	0	37	01	04	00	00	01	06	20	12	SCD.	RD area	
		1	12	00	00	01	60	40	00	00	13	LW	Updated Ref. Msg.	
		2	63	E	T	W	00	40	02	00	14	TS	ETW TK 40	
		3	71	01	15	50	00	00	00	00	14	TC	Set Switch Path	
011720-		4	25	01	15	33	00	01	16	23	15	TCT	Set Switch Path 7	
		5	12	01	00	00	00	40	00	00	16	LW	RI area	
		6	63	E	T	W	00	40	02	00	17	TS	ETW TK 40	
		7	71	01	15	60	00	00	00	00	17	TC	LRP Ref. Msg.	
012000+	0122	0	43	01	03	02	00	01	03	02	18	SC	Code:R (T) = 011766 "R"	
		1	61	01	23	00	00	01	23	00	18	CTC	+ Error B	
		2	51	01	06	47	01	77	77	77	19	DA	BOH + QTY. → WA1	
		3	71	01	20	50	00	00	00	00	19	TC		
011720+		4	22	01	20	77	00	01	03	02	20	OCT	"A" → Trans. Msg.	
012310		5	12	01	03	00	00	30	00	00	21	LW	Error Msg. A	
		6	63	E	T	W	00	30	02	00	22	TS	ETW TK 30	
		7	71	01	16	30	00	00	00	00	22	TC		

8.16.

Issue 21 H.S.M. Loc. Coded by Sheet 2 of 3

Title

'ENGLISH ELECTRIC' KDP 10
COMPUTER PROGRAMME RECORD

Date

Index No.

Block No.

From Instr. Loc.	H.S.M. Loc.		OO				OO				Chart Ref.	Remarks	Constants
			OP	A			N	B					
			0	1	2	3	4	5	6	7			
012000 012210	0123	0	22	01	20	76	00	01	03	02	23	OCT 'B' Code → TM	
		1	71	01	22	50	00	00	00	00	23	TC IW Error Msg.	
		2	01	01	04	00	00	01	04	01			
		3	01	01	04	03	00	01	04	07			
		4	01	01	06	30	00	01	04	33		RD Address List	
		5	01	01	04	72	00	01	05	13			
		6	01	01	06	60	00	01	05	74			
		7	01	01	06	04	00	00	00	00			
		0											
		1											
		2											
		3											
		4											
		5											
		6											
		7											
		0											
		1											
		2											
		3											
		4											
		5											
		6											
		7											

8.17

EEDPCS 31 H.S.M. Loc. Coded by Sheet 3 of 3

'ENGLISH ELECTRIC' KDP 10 COMPUTER
TIME ACCUMULATION SHEET

DETAIL PROCESS CHART

Computer Operation No._____ Page 1 of 2

Programme: Sample Problem 'A' Timed by: _____

Box Inst.	STAT	STA	Inst. Time	Time (ms)	Volume	Total Time (ms)
CSG.	.03	-	.015	.045	1	.045
RWD	4 x .03	4 x .015	4 x .3	1.38	1	1.38
TCT	.03	.015	.03	.075	9,000	675
LRF	.03	.015	3.575 + .03 x 80	6.02	10,000	60,200
SET	.03	-	.015	.045	10,000	450
SC	.03	.015	.045	.09	10,000	900
CTC	.03	-	-	.03	10,000	300
TC	.03	-	.015	.045	10,000	450
LRF	.03	.015	3.575 + .03 x 23	4.31	9,000	38,790
TCT	.03	.015	.03	.075	9,000	675
SET	.03	-	.015	.045	9,000	405
SC	.03	.015	.045	.09	9,000	810
CTC	.03	-	-	.03	9,000	270
SET	.03	-	.015	.045	10,000	450
SC	.03	.015	.405	.45	10,000	4,500
CTC	.03	-	-	.03	9,000	315
	.03		.015	.045	1,000	
TCT	.03	.015	.03	.075	1,000	75
LW	.03	.015	3.575 + .03 x 80	6.02	1,000	6,020
ETW	.03	-	.03	.06	1,000	60
TC	.03	-	.015	.045	1,000	45
SCT	.03	.015	.015 x 48	.765	9,000	6,885
RD	.03	.015	2.664	2.709	4,500	24,624
			2.718	2.763	4,500	
OCT	.03	.015	.03	.075	9,000	675
SET	.03	-	.015	.045	9,000	405
						147,980.425

'ENGLISH ELECTRIC' KDP 10 COMPUTER
TIME ACCUMULATION SHEET

Computer Operation No. _____ Page 2 of 2

Programme: Sample Problem 'A' Timed by: _____

Box Inst.	STAT	STA	Inst. Time	Time (ms)	Volume	Total Time (ms)
					From page 1	147,980,425
SC	.03	.015	.045	.09	9,000	810
CTC	.03	-	-	.03	7,500	292.5
			.015	.045	1,500	
SC	.03	.015	.045	.09	1,500	135
CTC	.03	-	-	.03	1,500	45
DA	.03	.015	.555	.6	1,500	900
TC	.03	-	.015	.045	1,500	67.5
DS	.03	.015	.555	.6	7,500	4,500
ZS	.03	.015	.12	.165	7,500	1,237.5
OCT	.03	.015	.12	.165	7,500	1,237.5
DA	.03	.015	.57	.615	7,500	4,612.5
IT	.03	.015	.24	.285	9,000	2,565
IT	.03	.015	.27	.315	9,000	2,835
SET	.03	-	.015	.045	9,000	405
SCD	.03	.015	3.375	3.42	9,000	30,780
LW	.03	.015	3.575 + .09 + .03 x 80	6.11	9,000	54,990
ETW	.03	-	.03	.06	9,000	540
TC	.03	-	.015	.045	9,000	405
						254,337.925

Total Processing Time = $4\frac{1}{4}$ Minutes.

9

FILE MAINTENANCE ROUTINES

INTRODUCTION

A File is a collection of groups or items of information arranged and classified for convenient reference. In data processing systems files are usually kept on one or more magnetic tapes and this section describes some of the standard procedures applicable to magnetic-tape file maintenance on KDP 10.

The order or sequence of the items of information on a file is determined by some chosen criterion known as a KEY. In inventory files this might be the part number of stock items and in insurance applications it might be a policy number.

Transactions may occur for every reference entry but it is more usual to find that transactions only occur for a proportion of the total entries on the reference file. There is, however, no reason why there should not be several transactions for a given reference message. In considering the logical structure of file maintenance programmes it is advantageous for training purposes to make a distinction between Single-Transaction and Multi-Transaction types, although the former is a particular case of the latter.

Single Transaction problems, in which not more than one transaction may occur for any one reference entry, form a suitable introduction to the more general case.

SINGLE TRANSACTION CASE

Two files, a Reference file and a Transaction file, are sorted in ascending order by the same key. Not more than one transaction occurs for any one reference and, it is required to construct a functional flow diagram illustrating file maintenance procedure. The first step is to consider the necessary course of action.

Messages from both files are read into the Computer. The keys are compared and one of three events may occur. If the transaction key is greater than the reference key the reference message is written out to an output tape, a new reference is read in and the keys are compared again. If the transaction key is less than the reference key an error has occurred in arranging the sequence of amendments on the transaction file. The transaction is written out to an error-tape, a new transaction is read in and the keys are again compared. If the keys agree the information in the transaction message is used to change the reference message which is then written out on the updated reference tape and new messages are brought in from both tapes. Logical diagrams illustrating this procedure are shown in Figs. 9.1. and 9.2. It will be noted that the order in which the files are read affects the structure of the diagram.

MULTI-TRANSACTION CASE

In the general case, where more than one transaction may occur for a given reference message, affairs are rather different. Once only, at the start of the programme, are messages brought in from both files. Thereafter new messages are brought in from one or other of the reference or transaction files but never both. If the transaction key is greater than the reference key new messages are brought in from the reference file until a 'match' is found (i.e. until the keys agree). Updating action follows and a new message is brought in from the transaction file only. The keys are again compared and, since the next transaction may also refer to the current reference, agreement may again occur leading to further amendments to the existing reference message.

When all the transaction messages for a given reference have been dealt with, the next transaction key will compare high against the reference key. The updated reference is then written out, a new reference is read in and the process continues. The logical flow diagram for this type of problem is shown in Fig. 9.3 and it will be noticed that three switches are used to control the flow of information. Switch A_1 arranges that messages are read from both files at the start of the programme to get the process going. Once this has occurred Switch A changes over to A_2 ensuring that

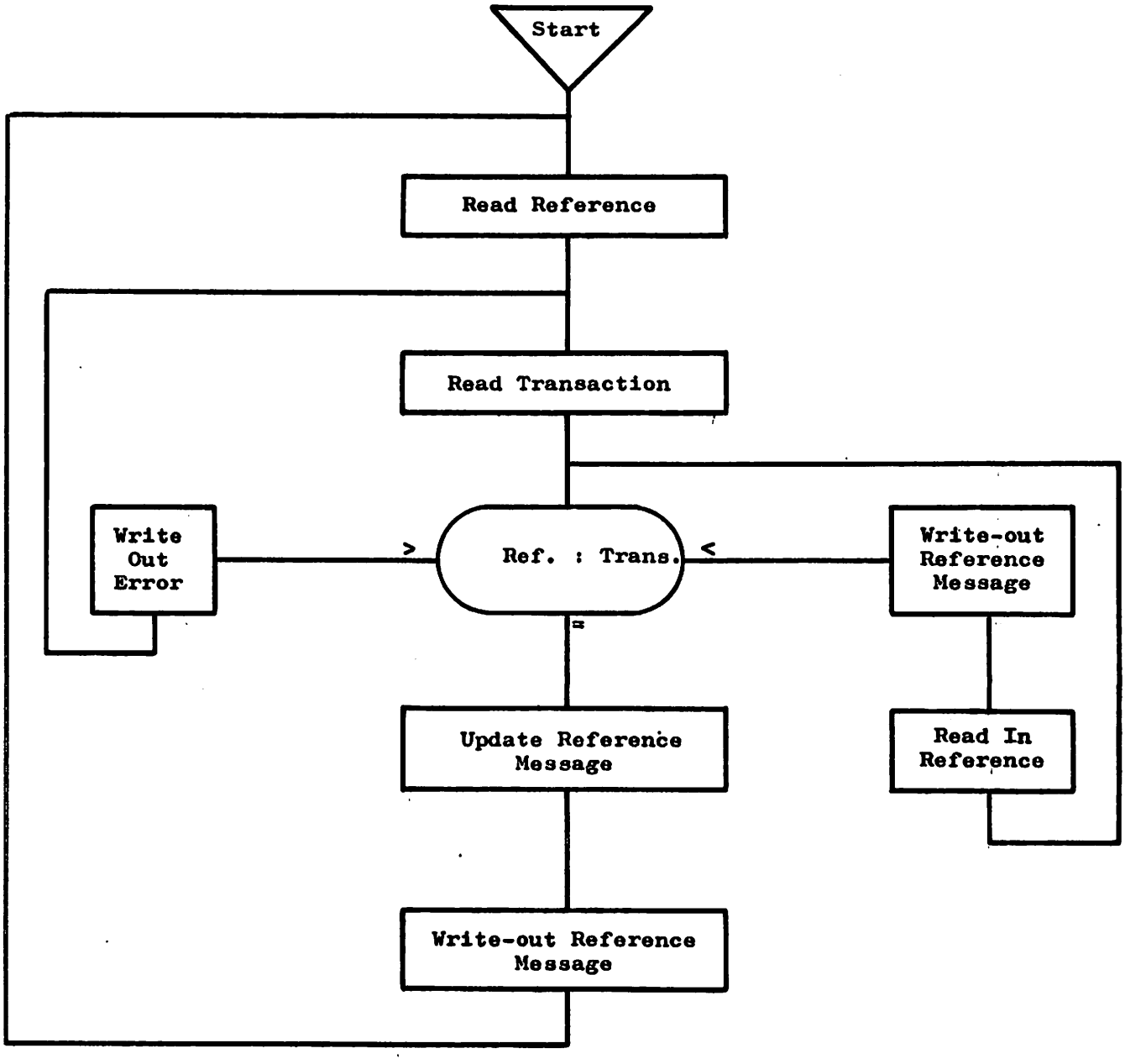


Fig. 9.1. Single Transaction Case. Reference Before Transaction.

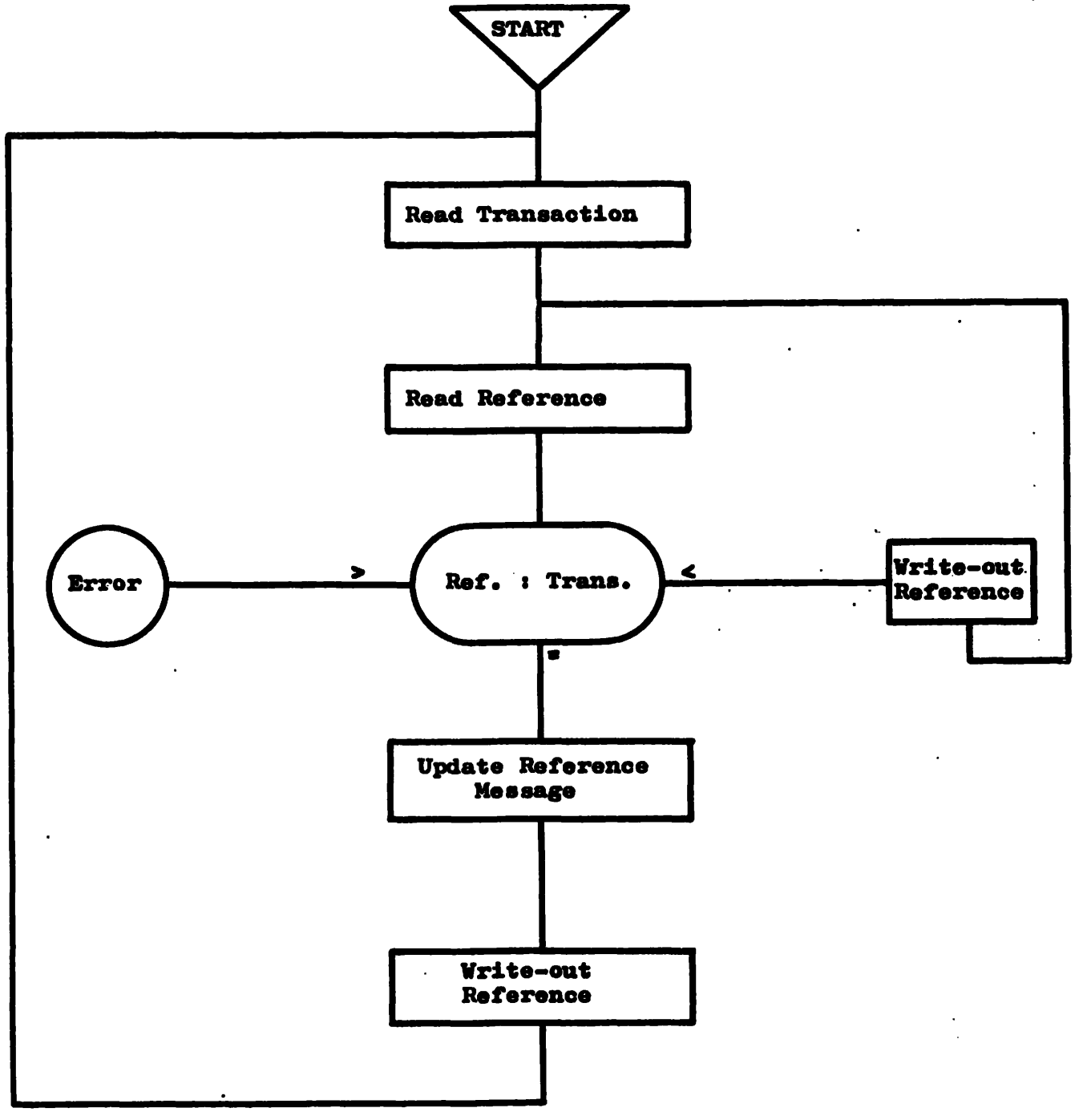


Fig. 9.2. Single Transaction Case. Transaction Before Reference.

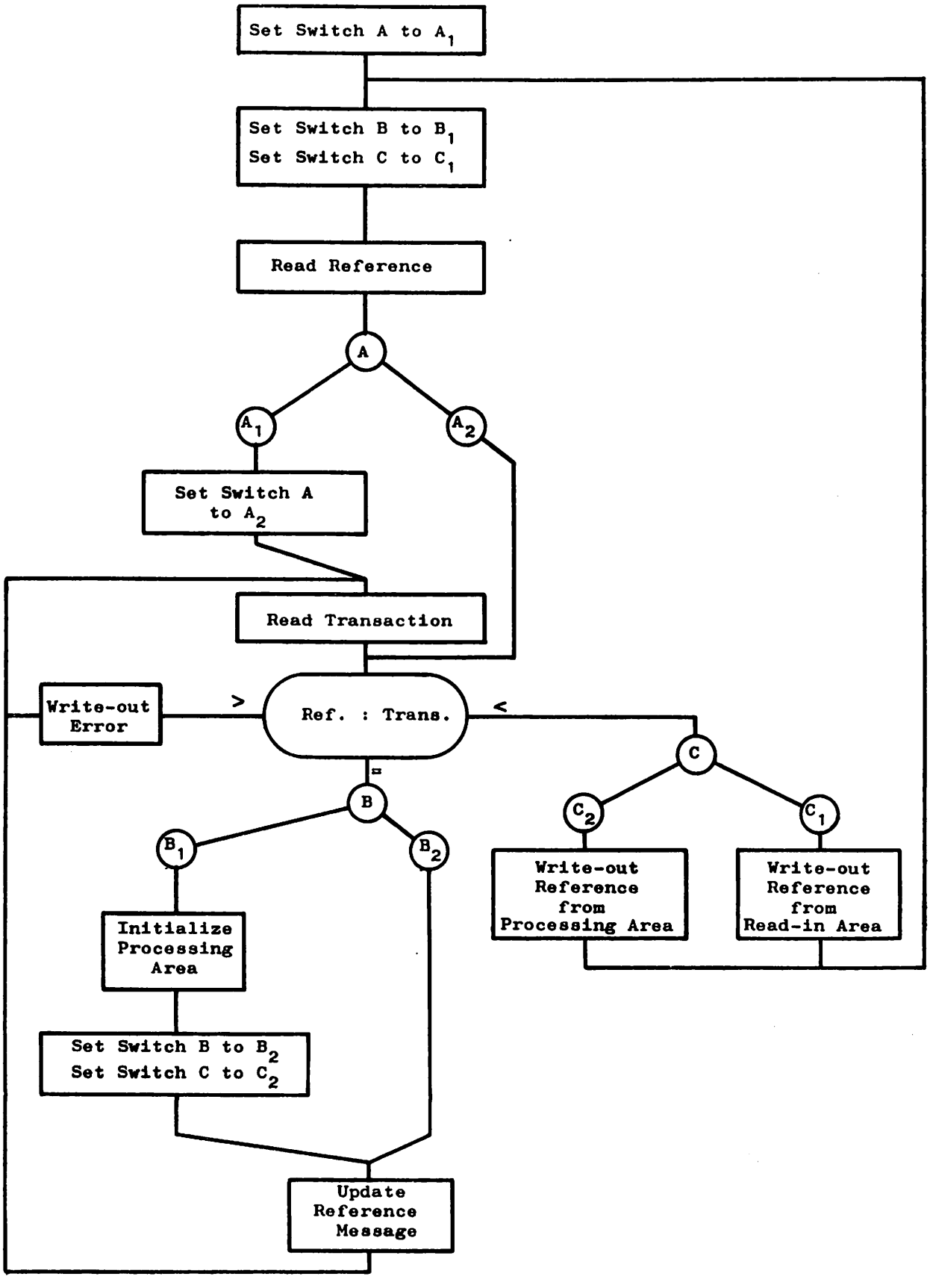


FIG. 9.3. MULTI-TRANSACTION CASE.

'Read Reference' is never again followed by 'Read Transaction'. Switch B, in position B_1, takes care of any action, (e.g. Random Distributing the reference message and clearing work areas) which is special to the first transaction relating to a given reference. Such action need occur once only for any batch of transactions with the same key. Switch C ensures that output messages are written out from the appropriate areas. If no updating has occurred the reference message is written out from the read-in area (C_1) but if processing has taken place it is written out from the processing area (C_2).

ED AND EF DETECTION

The messages on an input tape may include the special cases, ED, indicating the end of a reel, or EF, indicating the end of a file. Every message read in from tape must be checked to see whether the 1st character of the message is SM, ED or EF. If it is SM processing may continue, if it is ED the programme should enter an end-of-reel routine and if it is EF the programme should take action appropriate to an end-of-file condition.

The programme steps necessary for checking the first character on tape are shown in Fig. 9.4. An SM $(76)_8$ is stored as a constant and the first character of the read-in area is compared with the SM using a Sector Compare and Conditional Transfer of Control. If the result is positive an error has occurred and the programme should lead to a PES routine. If the result is negative further tests are made to determine whether the character is ED, EF or neither. In the last event the programme should again lead to a PES routine.

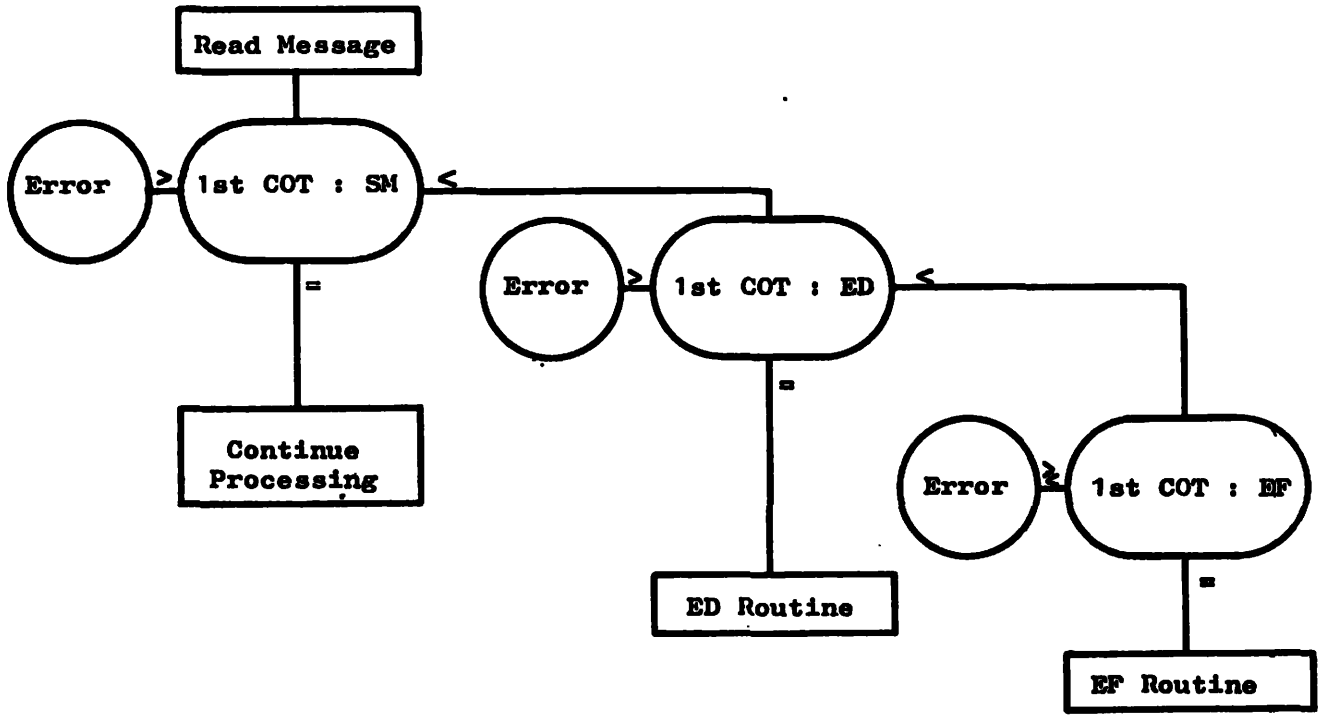


FIG. 9.4. CHECKING 1ST CHARACTER ON TAPE MESSAGE.

END OF OUTPUT REEL DETECTION

A distinction must be made between tapes from which information is being read (input tapes) and those on which information is being written (output tapes). The input tapes carry ED and EF symbols which may be used to detect an end of reel condition as explained above. There is no guarantee that output tapes carry such symbols and in any case it would be impractical to read the output tape in advance of writing merely to detect the end of a reel. The tape units of KDP 10 send out a warning signal known as ETW (End of Tape Warning) when 5 feet of usable tape remain on a reel. ETW may be detected by a Tape Sense instruction and the procedure is indicated as shown in Fig. 9.5.

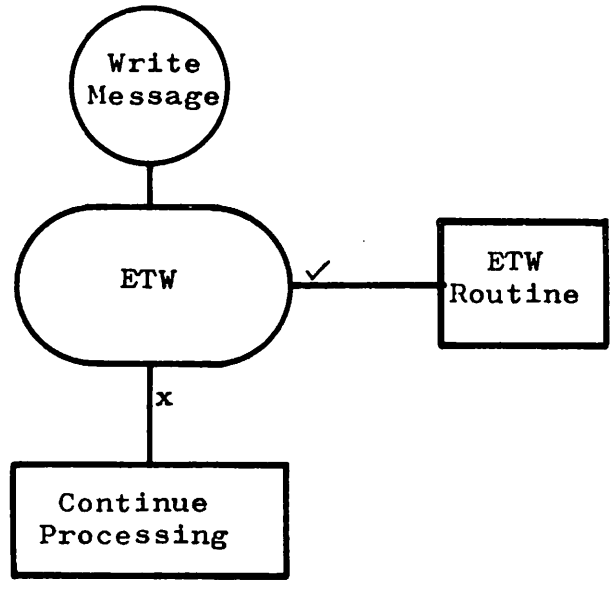
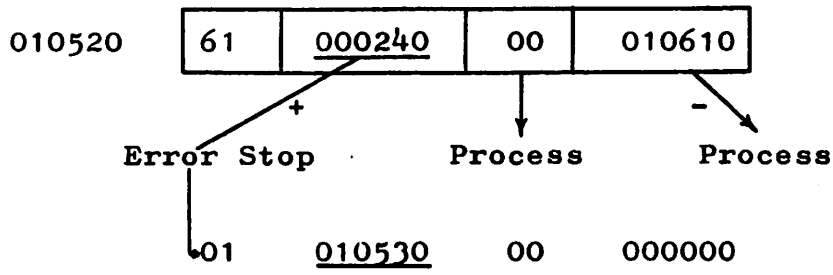


FIG. 9.5. DETECTION OF END-OF-OUTPUT REEL CONDITION

PES ROUTINE

When a programmer is charting out the logic of a programme he should make sure that a path exists for every conceivable situation. There are many instances when the only reason for following some of the paths is a machine failure and it is essential to provide a clear indication of the point at which the failure occurs, whenever this is possible. One method of unique identification is to use separate PES instructions for every failure path, providing each PES instruction with a different A address for indicative purposes. This will permit convenient failure location but may become rather expensive in instructions. Two programming techniques may be used which have the same effect and use less instructions. Both methods employ the STP facility of KDP 10. When an instruction transfers control, the contents of the P register are stored in STP at 000241 - 000243. When the machine stops the contents of STP indicate the address of the next instruction following the one that causes the failure. If an $(01)_8$ constant (the PES OPERATION CODE) is pre-stored at 000240, all PES failures may transfer control to this address. Every error stop will then have a different number in the A address, namely the address of the instruction following the failure exit.

As an example, suppose the failure occurs in a CTC instruction at 010520, as overleaf, assuming that $(01)_8$ is stored in 000240



This instruction is displayed on the console and the A address indicates where the error occurred.

If the programmer feels that it would be more convenient to cause the A address of the STP error stop to indicate the exact failure address and not $(10)_8$ in advance of it, he may do so in one of two ways:-

(a) By transferring STP to the A address of a PES instruction and subtracting 000010 from it before leading to it.

(b) By pre-storing 01 at 000240, subtracting 000010 from STP and transferring control to STP by a SET P instruction to avoid disturbing STP again.

These methods are shown below:-

(a)
```
TCT    STP → A address of PES
TCS    B address of PES from A address
PES
```

The constant 000010 is conveniently stored in the otherwise unused B address of the PES instruction.

(b)
```
TCS    000010 from STP
SET    P to 000240
PES
```

The constant 000010 could be conveniently stored at 000243 - 000247 along with the pre-stored $(01)_8$ at 000240.

ETW ROUTINES

When an ETW condition exists an output tape needs attention. An ED should be written on the tape, which should then be rewound to BTC. A message should be issued calling the operator's attention to whichever tape unit is affected and the computer should stop to permit the old reel to be demounted and a new reel mounted in its place. After the operator restarts the computer the programme should rewind the new reel to BTC and transfer control back to the instruction immediately following the Tape Sense instruction which detected ETW. In this way processing continues at the point where it was interrupted by the ETW condition. These operations are essential in all ETW routines.

As it is likely that most programmes will use more than one output tape it is natural to enquire whether separate ETW routines are necessary for every output tape. This is not so; a common ETW subroutine may be used provided certain precautions are taken. Separate Tape Sense instructions must be used with every write instruction, each having the appropriate Tape Unit address in the B_1 character position. The address of the affected tape unit is, therefore, available from whichever Tape Sense instruction detects ETW. The re-entry point is available from the contents of STP and these may be transferred as the subroutine link address. One routine, among many which may be used, is shown in Fig. 9.6 to which the following explanatory notes refer.

1. Each Tape Sense Instruction should have the following addresses:-

A address	-	Entry to ETW Subroutine
B_1 address	-	Octal Tape Unit Address
B_2 address	-	$(02)_8$ to sense ETW
B_3 address	-	Octal equivalent of the first octal digit of the Tape Unit Address.

Example: Tape Unit 10

63	Subroutine Address	00	10	02	24

Example: Tape Unit 50

63	Subroutine Address	00	50	02	30

By this means the B address of each Tape Sense instruction identifies not only the Tape Unit, in B_1, but also provides the character X which must be printed in the message <REPLENISH TAPE XO>. The B register is stored in the Linear Write instruction of Box 3 in the subroutine.

2. Each Tape Sense instruction leads to the ETW subroutine from a different part of the programme. STP contains the address of the instruction to which the subroutine should return. Box 2 transfers the address from STP to the Transfer Control Link of Box 12.

3. Writes an ED on the exhausted tape.

4. Transfers the tape unit address to the Rewind instruction in Box 7.

5. Does the same for Box 11.

6. Transfers the first character X of the Tape Unit Address to the operator message. It should be noted that the characters XO which appear on the Monitor Printer must be stored in the operator message area as (24) (23) if X = 1, or as (30 (23) if X = 5.

7. Rewinds the old tape.

8. Writes out to the Monitor Printer a message indicating which tape unit is rewinding.

9. Stops the machine to permit tape changing.

10. The operator re-starts the Computer.

11. The new tape is rewound in case the reload procedure has resulted in an excessive amount of tape on the take-up reel.

12. Transfers control back to the main programme.

This routine is adequate for any series of tape units in which the second octal digit of the tape address is constant. In the example above the series is (00) (10) (20) etc. If the second digit of the octal tape address varies for different output units a slightly different approach is required. This is explained in the section describing programming techniques.

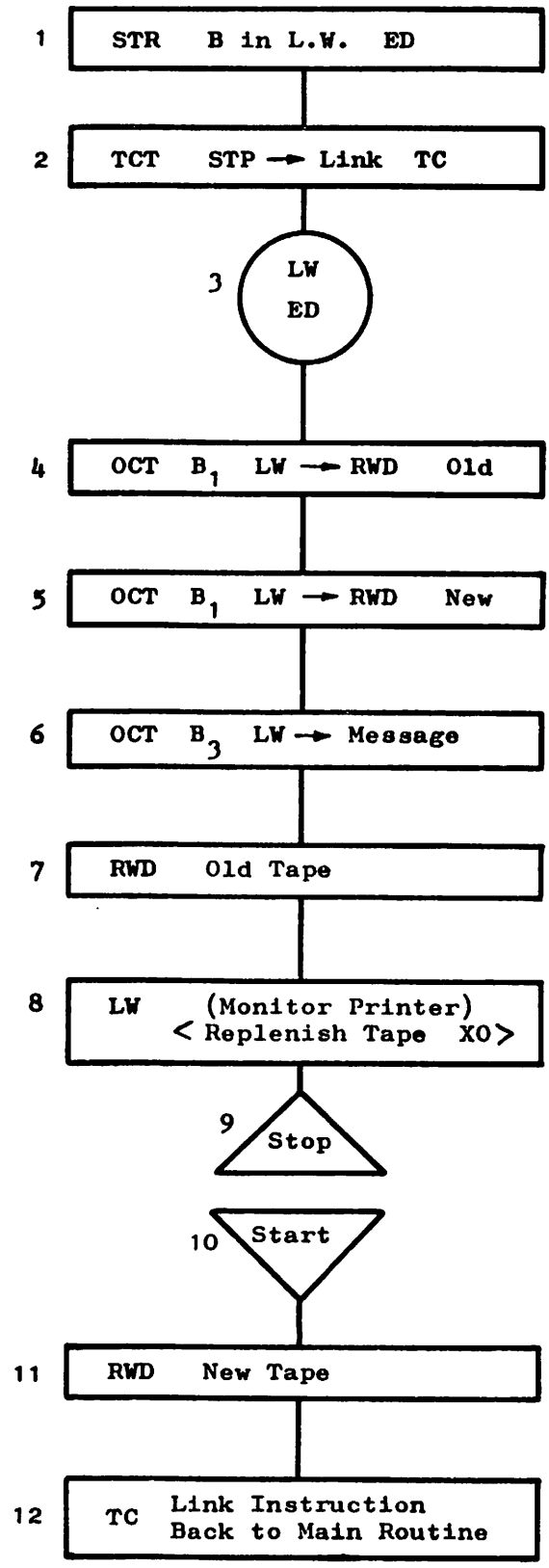


FIG. 9.6. ETW SUBROUTINE.

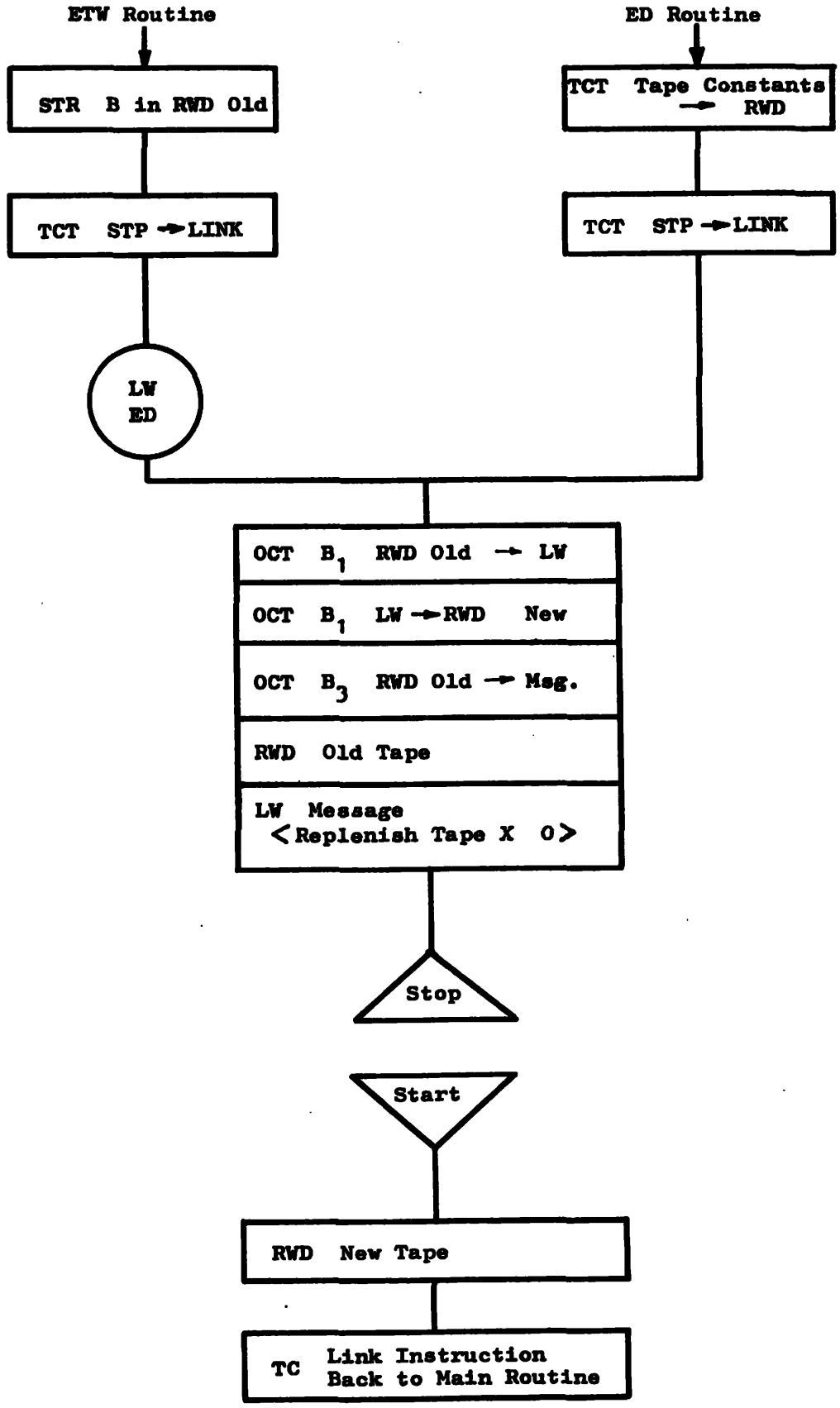


FIG. 9.7. A METHOD OF COMBINING ED AND ETW ROUTINES.

ED ROUTINES

An end of reel condition on an input tape calls for very similar action to that for an exhausted output tape. There is no requirement to write an ED, otherwise the instructions are the same. It may even be possible to combine the ETW and ED routines as shown in Fig. 9.7.

EF ROUTINES

The action taken when EF is sensed will depend very much on the requirements of a particular application and also on which file, Reference or Transaction, finishes first. If the Transaction file finishes before the Reference file all remaining reference messages must be written out. If the Reference file finishes before the Transaction file new entries must be generated on the Reference file. When the last EF has been sensed it only remains to tidy up and put everything carefully away. An EF (and possibly an ED) should be recorded on output tapes and all tapes should be rewound to BTC. The Computer should then stop.

KDP 10 INSTRUCTIONS 01 - 06

01 PROGRAMME ERROR STOP PES

GENERAL DESCRIPTION

This instruction inhibits the staticising of any further instructions, halting the Computer after completion of any instruction in the Simultaneous Mode and lighting a P.E.S. lamp on the Computer console.

FORMAT

A address - ignored. B address - ignored.

FINAL REGISTER CONTENTS

The A register store contains the A address of the instruction modified by N_A. The B register contains the B address of the instruction modified by N_B. All other registers and register stores are unaltered.

TIMING

Staticising and STA time only.

OUTLINE OF LOGIC

If the Simultaneous Mode is unoccupied when the stop instruction is staticised, the Computer stops immediately. If it is occupied, the Simultaneous instruction is completed before the Computer stops. However, if a 'read' parity error is detected in the Simultaneous Mode after a P.E.S. instruction is staticised, the Computer will stop without attempting to perform Rollback.

The P.E.S. lamp on the Computer console lights up.

The instruction goes through STA.

02 PRINT PR

GENERAL DESCRIPTION

This instruction transfers the characters stored in 120 consecutive HSM locations to the Line Printer, causing one line to be printed. The operation is initiated only when the Simultaneous Mode is free, since it uses both the Normal and Simultaneous Modes.

FORMAT

A Address

Address of the leftmost character of the sector to be printed.
A_2 must be an even number.
A_3 must be $(00)_8$.

B address

Ignored.

DIRECTION OF OPERATION

Left to right.

TERMINAL CONDITION

The operation terminates when a complete line of 120 characters has been transferred to the Line Printer, and the print drum has made one complete revolution during which the line has been printed.

FINAL REGISTER CONTENTS

The A register store contains the address of the leftmost character in the sector to be printed. The B register contains the address of the first character to the right of the sector to be printed. The T register contains $(003124)_8$ which is the product of the number of tetrads in the sector times the number of character lines around the circumference of the print drum.

TIMING

One line is printed in 67 milliseconds.

EXAMPLE (PR)

02 023200 00 000000

HSM before and after the instruction is executed:

0232	00	01	02	03	04	05	06	07	10	11	12	13	14	15	16	17	20	21	22	23	24	25	26	27	30	31	32	33	34	35	36	37	0232
	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	

0232	40	41	42	43	44	45	46	47	50	51	52	53	54	55	56	57	60	61	62	63	64	65	66	67	70	71	72	73	74	75	76	77	0232
	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	

0233	00	01	02	03	04	05	06	07	10	11	12	13	14	15	16	17	20	21	22	23	24	25	26	27	30	31	32	33	34	35	36	37	0233
	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	

0233	40	41	42	43	44	45	46	47	50	51	52	53	54	55	56	57	60	61	62	63	64	65	66	67	70	71	72	73	74	75	76	77	0233
	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P									

This instruction will effect printing, on the Line Printer, of the contents B HSM locations $(023200)_8$ - $(023367)_8$ inclusive, i.e. the area filled with P's.

Final register contents:

STA = 023200

B = 023370

T = 003124

Time:

67 ms.

OUTLINE OF LOGIC

The contents of the A Register are initially duplicated in the B Register, which is increased by $(04)_8$ after each tetrad processed.

As the drum revolves, the Line Printer sends signals to the Computer, indicating which row will next be in the print position. The sector of 120 characters is read out of the HSM by tetrads and scanned for the character that will next be in print position. Each time the full sector has been scanned, the Line Printer's Shift Register contains 120 '1' and '0' bits, the '1' bits corresponding to the locations in which the character is to be printed. All occurrences of the character are printed at one and the same time. The process is repeated for each of the 54 rows of characters (including the three normally nonprintable characters =, +, and @, so that printing of the 120-character line is completed when the print drum has made one full revolution.

One character is completely printed when the contents of the B Register have been increased by $(170)_8$. [Note: $(170)_8 = (120)_{10}$ = number of characters in the sector.] The operation is terminated when the T Register has been increased to $(003124)_8$. Initially, the T Register is automatically cleared to zeros; $(01)_8$ is added each time a tetrad is scanned [$(120/4)\ (54) = (1620)_{10} = (3124)_8$].

03 PAPER ADVANCE PA

GENERAL DESCRIPTION

This instruction positions the paper in the Line Printer for the next line of printing. It can advance the paper a number of lines, specified either by A_2A_3 or by the punches in a tape loop on the Printer. This is a potentially simultaneous instruction. While in the Simultaneous Mode, it does not restrict the use of any other instruction in the Normal Mode except Print or another Paper Advance.

FORMAT

Loop controlled Advance:

A address: B address - ignored

A_1 - denotes the type of advance.

An odd number (octal) = Vertical Tabulation (VT).

An even number (octal) = Page Change (PC)

[If $(00)_8$ is used, no paper advance occurs]

A_2A_3 - $(0000)_8$.

Instruction Controlled Advance:

A address: B address - ignored

A_1 - ignored.

A_2A_3 - the number (octal count) of lines the paper is to be advanced.

FINAL REGISTER CONTENTS

If the operation is concluded in the Normal Mode, the A register store contains the A_1 part of the instruction in the C_1 position and $(0000)_8$ or $(0000)_8$ minus the number of lines shifted according to whether the advance was instruction controlled or loop controlled, in the C_2C_3 position. The B register is unaltered.

If the operation is concluded in the Simultaneous Mode, the S register settings are analogous to the A register store settings above.

The T register and P register store are unaltered.

TIMING

Total time (paper motion time) in milliseconds:

30 for single line shifting

30 + (n - 1) 20 for multiline shifting

where n is the number of lines advanced.

EXAMPLES (PA)

Loop Controlled:

Instruction:

0	A	N	B
03	010000	00	000000

In this case the paper on the Line Printer will be advanced in accordance with the punches in the VT column of the tape loop on the Printer

Instruction:

0	A	N	B
03	020000	00	000000

In this case the paper will be advanced in accordance with the punches in the PC column of the tape loop on the Printer.

Computer Controlled:

Instruction:

0	A	N	B
03	000011	00	000000

In this case the paper will be advanced nine lines.

OUTLINE OF LOGIC

The A_2A_3 characters are tested. If they are not $(0000)_8$, a signal is sent to the Printer to advance the paper. When one line has been shifted, the Printer signals the Computer and the A_2A_3 characters (in the A or S Register) are decreased by $(0001)_8$. This process continues until $(0000)_8$ emerges from the Bus Adder; the Computer then signals the Printer to stop the Paper Advance. If the A_2A_3 characters are initially $(0000)_8$, the A_1 character is examined, and the function (VT or PC) specified is performed in accordance with the punches in the tape loop on the Printer. [$(0001)_8$ is subtracted from the A_2A_3 characters (in the A or S Register) with each line shifted.]

This instruction does not use the SW or SR Register, nor is it stored in standard HSM locations.

04 LINEAR READ REVERSE LRR

GENERAL DESCRIPTION

This instruction transfers one message from magnetic or paper tape to the HSM. It is a potentially simultaneous instruction. Though the tape is read in reverse, the characters will be placed in the HSM in their proper relative positions. LRR is most useful in sort routines since it saves rewind time.

FORMAT

A address

Address of the tetrad in which the EM ED or EF is to be stored. Any one of the four locations may be specified; the EM, however, will always be placed in C_3.

B address

B_1 - Address of the input unit. [$(77)_8$ addresses the Paper Tape Reader.]

B_2B_3 - ignored.

DIRECTION OF OPERATION

Right to left.

DIRECTION OF TAPE MOVEMENT

Magnetic tape moves in reverse. Paper tape moves in the forward direction. In LRR, however, characters on paper tape must come in as if the tape were moving in reverse, i.e., EM first and SM last. This may be accomplished by manually reversing and inverting a paper tape which was prepared in normal fashion. (LRR from paper tape is used in programme testing).

TERMINAL CONDITION

The operation terminates when an SM, ED or EF has been read. If this character does not fall into a C_0 position, spaces are generated to fill out the last tetrad.

FINAL REGISTER CONTENTS

Provided the operation is concluded in the normal mode, the A register store contains the address of the SM, ED or EF and the B register contains the B address of the instruction modified by N_B. If the operation is terminated in the Simultaneous Mode, the S register contains the address of the SM, ED or EF. The T register and P register stores are unaltered.

TIMING

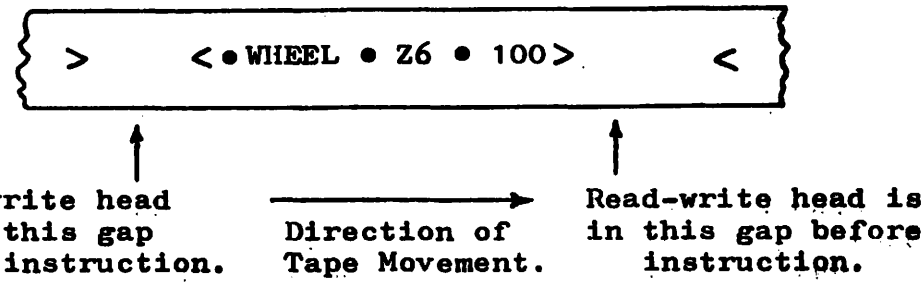
Total time in milliseconds = 3.575 + .03n

where n is the total number of characters transferred.

EXAMPLE (LRR)

Instruction: 04 120771 00 130000

Tape (on Tape Station 13):



HSM before Instruction is executed:

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
1207			K K	K K K K	K K K K	K K K K	K K K K	K K	1207

HSM after Instruction is executed:

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
1207			K K	- < • W	H E E L	• Z 6 •	1 0 0 >	K K	1207

FINAL REGISTER CONTENTS:

STA or S = 120755

B = 130000 (unless the instruction is concluded in the Simultaneous Mode)

TIME

3.575 + (.03 x 15) = 4.025 ms.

OUTLINE OF LOGIC

A start signal is sent to the specified input unit. Coincident with each pulse (or sprocket hole) in the timing track, a character is brought into a recognition circuit. A check is performed to verify that the first character is an EM, ED or EF. (If the first character is not one of these three, a CIG alarm stop occurs). The EM and (unless an SM intervenes) the three characters following it are serially admitted into the upper half of the Read Buffer. These characters are then shifted (four-character parallel transfer) into the lower half of the Read Buffer and then into the tetrad specified by the contents of the A Register, which is then decreased by $(000004)_8$. Tape-to-Buffer transfer continues to overlap Buffer-to-HSM transfer until an SM is sensed (in the recognition circuit), at which point a stop signal is sent to the input device so that no further reading occurs. The instruction is then terminated with the HSM address of the SM in the A (or S) Register.

If the SM is not placed in C_0 of a tetrad, then one, two or three spaces are generated to fill out the tetrad.

Gaps intervening on tape between the EM and the SM are ignored by this instruction.

The control symbols ED and EF, when standing alone (with a Gap on either side), are treated as complete messages in themselves.

05 BLOCK READ REVERSE BRR

GENERAL DESCRIPTION

This instruction transfers a block of characters from magnetic or punched paper tape into the HSM. Though the tape moves in reverse, the characters will be placed in the HSM in their proper relative positions. The instruction is potentially simultaneous. (See discussion of automatic decoding of characters on block read from paper tape, page 4.2).

FORMAT

A address:

Address of the tetrad which is to receive the first character (following a gap) read from the tape. Any one of the four locations in the tetrad may be specified; the first character, however, will always be placed in C_3.

B address:

B_1 - address of the input unit. [$(77)_8$ addresses the Paper Tape Reader.]

B_2B_3 - ignored.

DIRECTION OF OPERATION

Right to left.

DIRECTION OF TAPE MOVEMENT

Magnetic tape moves in reverse. Paper tape moves in the forward direction. In BRR, however, characters on paper tape must come in as if the tape were moving in reverse. This may be accomplished by manually reversing and inverting a paper tape which was prepared in normal fashion. (BRR from paper tape is used in programme testing.

TERMINAL CONDITION

The operation terminates when a gap on tape is sensed. If the last character does not fall in a C_0 position, spaces are generated to fill out the last tetrad.

FINAL REGISTER CONTENTS

Provided the operation is terminated in the normal mode, the A register store contains the address of the last character read from tape (i.e. the first character in the block) and the B register contains the B address of the instruction modified by N_B. If the operation is terminated in the Simultaneous Mode, the S register contains the address of the last character read from tape. The T register and P register store are unaltered.

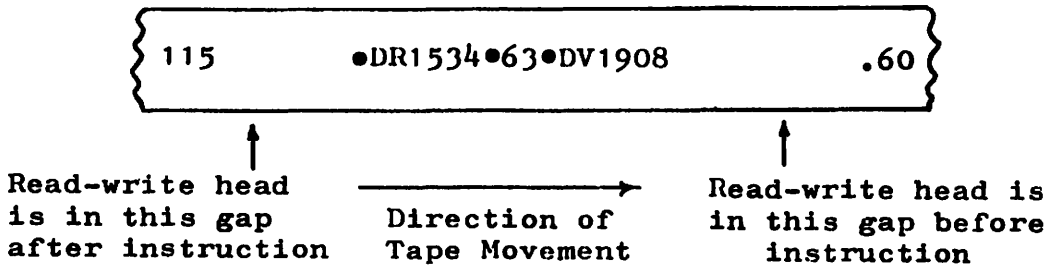
TIMING

Total time in milliseconds = 3.575 + .03 (n + 6) where n is the total number of characters transferred.

EXAMPLE (BRR)

Instruction: 05 076025 00 200000

Tape (on Tape Station 20):



HSM before Instruction is executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0760	K K	K K K K	K K K K	K K K K	K K K K	K K K K	K K		0760

HSM after Instruction is executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0760	K K	- - - •	D R 1 5	3 4 • 6	3 • D V	1 9 0 8	K K		0760

FINAL REGISTER CONTENTS:

STA or S = 076007

B = 200000 (unless the instruction is concluded in the Simultaneous Mode).

TIME:

3.575 + .03 (17 + 6) = 4.265 ms.

OUTLINE OF LOGIC

A start signal is sent to the specified input unit. The first four characters following the gap are placed in the Read Buffer and then (four-character parallel transfer) into the HSM tetrad specified by the A Register. The first character read from tape is placed in C_3. The operation continues until a gap is detected. If the last character of the block is placed in C_1, C_2 or C_3 of a tetrad, one, two or three spaces, respectively, will be generated to fill out the tetrad. The A (or S) Register is reset so that it holds the HSM address of the last character read.

An ED or EF standing alone (with a gap on either side) is treated as a complete block.

06 UNWIND n SYMBOLS UNS

GENERAL DESCRIPTION

This instruction causes a selected magnetic tape to be moved forward until a specified number of a designated symbol have been counted. The HSM is not altered. The instruction is potentially simultaneous, but no 'read' instruction can be executed simultaneously with it.

FORMAT

A address:

A_1 - the symbol; this may be EM, ED, EF or Gap. Gap is designated by $(00)_8$.

A_2A_3 - (in octal count) the number of symbols to be counted; this may be $(0000)_8$ to $(7777)_8$.

B address:

B_1 - address of the Tape Station.

B_2B_3 - ignored

DIRECTION OF TAPE MOVEMENT

Forward

TERMINAL CONDITION

The operation terminates when the tape has been unwound the specified number of symbols, and the tape stops with the Read-Write head positioned in the gap (or the gap following the EM, ED or EF). If the PET is sensed prior to this condition being satisfied, the operation terminates and the computer stops (alarm).

FINAL REGISTER CONTENTS

If the operation terminates in the Normal Mode, the A register store contains the specified symbol in C_1 position, and $(0000)_8$ in the C_2C_3 positions unless the PET is reached, in which case the C_2C_3 positions contain an octal count of the number of symbols remaining to be found. The B register contains the B address of the instruction modified by N_B. If the operation terminates in the Simultaneous Mode, the S register contains the specified symbol in the S_1 position and $(0000)_8$ in the S_2S_3 positions unless the PET is reached, in which case the S_2S_3 positions contain an octal count of the number of symbols remaining to be found.

TIMING

Total time in milliseconds = 3.575 + .03n + 4m

where n is the total number of characters read, including symbols counted and m is the number of gaps encountered.

EXAMPLE (UNS)

Instruction: 06 750015 00 200000

This instruction will move the tape (on Tape Station 20) forward through 13 End Message symbols, unless the PET is reached before the full count. With the count completed, the tape stops with the Read-Write head in the gap following the thirteenth EM, positioned, for example, for a Linear Read Forward of the subsequent message, or a Linear Read Reverse of the message associated with the thirteenth EM counted.

OUTLINE OF LOGIC

A start signal is sent to the specified Tape Station. The characters on the tape pass into the recognition circuit, but do not enter the Read Buffer. When an instance of the specified symbol is found, the Computer is signaled and $(0001)_8$ is subtracted from A_2A_3 by the Bus Adder. When the output of the Bus Adder is $(0000)_8$, a stop signal is sent to the input device and the operation is concluded.

If the PET is sensed, the Tape Station and the Computer will stop.

If n is initially $(0000)_8$, the tape will not move. The instruction, however, will be stored in the standard HSM locations in which a read instruction is stored.

When unwinding EM's an alarm stop occurs if the data are not in message format.

When unwinding gaps, a CIG alarm stop occurs if there are fewer than eight characters in a message or block. Exception: ED and EF stand alone (they may never appear within a message).

An alarm stop occurs if a nonpermissible symbol is specified.

KDP 10 INSTRUCTIONS 10-17

10 TRANSCRIBING CARD WRITE TCW

GENERAL DESCRIPTION

This instruction is the same as 12 (LW) in every respect except that characters are written to tape at half speed. Tapes written in this way are used as input for the Transcribing Card Punch.

11 SINGLE SECTOR WRITE SSW

GENERAL DESCRIPTION

This instruction writes on to magnetic tape (or the Monitor Printer), in block format, the contents of a sector of any size, located in any area of the HSM. This instruction can be executed only in the Normal Mode.

FORMAT

Preset T Register

T_1 - address of output unit [$(77)_8$ addresses MONITOR PRINTER, $(76)_8$ addresses the on-line PAPER TAPE PUNCH].

T_2 - 2^0 bit = 0, = 3.5 ms write delay.
2^0 bit = 1, = 4.5 ms write delay.

T_3 - ignored.

A address:

Address of leftmost character of sector.

B address:

Address of rightmost character of sector.

DIRECTION OF OPERATION

Left to right.

DIRECTION OF TAPE MOVEMENT

Forward.

TERMINAL CONDITION

The operation terminates when the rightmost character of the sector has been written out.

FINAL REGISTER CONTENTS

The A register store contains the address of the rightmost character of the sector. The B register also contains this address. The T register remains as it was preset. The P register store is unaltered.

TIMING

Total time in milliseconds (T_2 2^0 = 0) = 3.575 + .03n where n is the total number of characters transferred. When T_2 2^0 = 1 change 3.575 to 4.575.

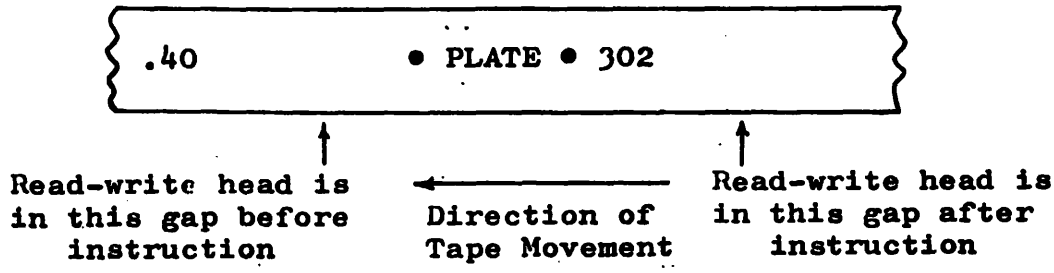
EXAMPLE (SSW)

Instructions:	72	060000	00	600000
	11	024551	00	024562

HSM before and after instructions:

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0245			4 • P L	A T E •	3 0 2 •				0245

Tape (on Tape Station 06) after execution:



FINAL REGISTER CONTENTS:

STA = 024562
B = 024562
T = 060000

TIME:

3.575 + (.03 x 10) = 3.875 ms.

OUTLINE OF LOGIC

A start signal is sent to the designated output unit. The tetrad containing the character whose address is specified by the A Register is brought into the Memory Register, and the A Register is increased by $(000004)_8$. The characters in the Memory Register are transferred, in parallel, into the lower half of the Write Buffer, then shifted (four-character parallel transfer) into the upper half of the Buffer.

If the output unit is other than a tape station, serial write-out of characters from the upper half of the Buffer, beginning with the character whose address was initially specified by the A Register, begins without delay. Unless the B Register specifies termination within this same tetrad, the next tetrad, addressed by the A Register, is transferred to the lower half of the Buffer, via the Memory Register, during serial write-out of characters from the upper half of the Buffer. Tetrad transfer to the upper half of the Buffer continues to overlap write-out from the lower half until the character whose address is specified by the B Register is written out.

If the designated output unit is a tape station, write-out of characters from Buffer to tape is delayed according to the 2^0 bit of the T_2 character. Unless the A and B Registers initially address the same tetrad, the entire Write Buffer is filled (eight characters), with the A Register increased by $(000010)_8$, before serial write-out from the Buffer begins.

At the conclusion of SSW, the A Register holds the HSM address of the last character written out.

12 LINEAR WRITE LW

GENERAL DESCRIPTION

This instruction transfers one message from the HSM to magnetic tape, the Monitor Printer or to paper tape via the Monitor Printer. It is a potentially simultaneous instruction.

FORMAT

A address:

Address of the leftmost character in the message to be written.

B address: [To address Relocation.

B_1 - address of output unit; $[(77)_8$ addresses the Monitor Printer, $(76)_8$ addresses the On-line Paper Tape Punch].

B_2 - If $2^0 = 0$, 3.5 ms write delay occurs.
If $2^0 = 1$, 4.5 ms write delay occurs.

B_3 - ignored.

DIRECTION OF OPERATION

Left to right.

DIRECTION OF TAPE MOVEMENT

Forward.

TERMINAL CONDITION

The operation terminates when an EM, ED or EF has been written out.

FINAL REGISTER CONTENTS

Provided the operation is terminated in the Normal Mode, the A register store contains the address of the EM, EF or ED and the B register contains the B address of the instruction modified by N_B. If the operation is terminated in the Simultaneous Mode, the S register contains the address of the EM, EF or ED. The T register and P register store are unaltered.

TIMING

Total time in milliseconds (B_2 $2^0 = 0$) = 3.575 + .03n where n is the total number of characters transferred. If B_2 $2^0 = 1$ change 3.575 to 4.575.

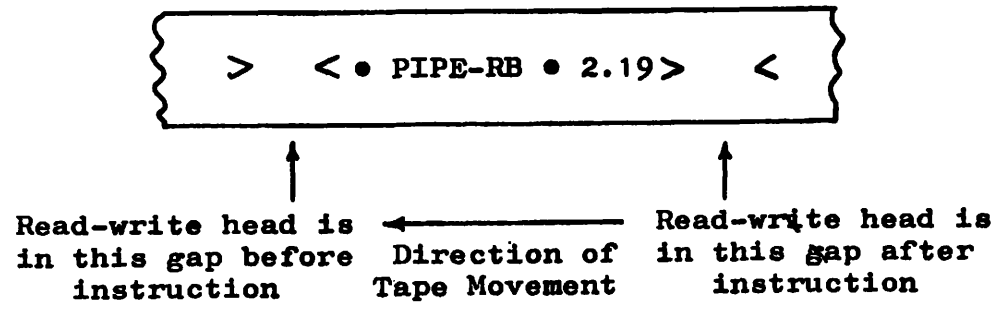
EXAMPLE (LW)

Instruction: 12 072144 00 020000

HSM before and after Instruction is executed:

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0721		< • P I	P E - R	B • 2 .	1 9 > -				0721

Tape (on Tape Station 02) contents after execution:



FINAL REGISTER CONTENTS

STA or S = 072162
B = 020000 (unless the instruction is concluded in the Simultaneous Mode)

TIME:

3.575 + (.03 x 15) = 4.025 ms.

OUTLINE OF LOGIC

A start signal is sent to the designated output unit. The tetrad containing the character whose address is specified by the A register is brought into the Memory Register, and the A Register is increased by $(000004)_8$. The contents of the Memory Register are transferred in parallel, into the lower half of the Write Buffer, then shifted into the upper half of the Buffer. If the character from the location initially specified by the A Register is an ED or an EF, it will be the only character written out. If an EM is found in any location in the tetrad, it will be the last character written out. In either case, the A Register is reset so that it finally holds the address of the terminating control symbol.

If none of these control symbols is sensed in the first tetrad as described, and the output unit is not a tape station, serial write-out of characters from the upper half of the Buffer, starting with the character from the HSM location originally addressed by the A Register, begins without delay. Tetrad transfer of characters from the HSM to the lower half of the Buffer, via the memory Register, overlaps write-out from the upper half until an EM is written, at which point the instruction is concluded, with the HSM address of the EM in the A Register.

If the designated output unit is a tape station, writing is delayed according to the 2^0 bit of the B_2 character of the instruction. Unless an EM, ED or EF occurs in the first tetrad as described above, the entire Write Buffer is filled (eight characters), and the A register is increased by $000010)_8$, before the first character is written from Buffer to tape.

13 MULTIPLE SECTOR WRITE MSW

GENERAL DESCRIPTION

This instruction writes on to magnetic tape (or the Monitor Printer), a single block comprising the contents of any number of sectors taken from various parts of the HSM under the direction of a stored list of addresses. This instruction is executed only in the Normal Mode. However, a prior 'read' may, at the same time, be in the Simultaneous Mode.

FORMAT

A address:

Address of the leftmost tetrad of the list of addresses. The contents of each tetrad in the list must be in the form KXXX, where XXX is the HSM address of the leftmost character of the sector to be written and K is the number (octal count) of character locations in the sector. When K is initially $(00)_8$, sixty-four characters are written out.

In the last address in the list, XXX must be $(000000)_8$ and K is ignored.

B address:

B1 - address of output unit; [$(77)_8$ addresses the Monitor Printer, $(76)_8$ addresses the on-line Paper Tape Punch].

B_2 - If 2^0 bit = 0, 3.5 ms write delay occurs. If 2^0 bit = 1, 4.5 ms write delay occurs.

B_3 - ignored.

DIRECTION OF OPERATION

Left to right.

DIRECTION OF TAPE MOVEMENT

Forward.

TERMINAL CONDITION

The operation terminates when the address $(000000)_8$ is encountered in the list of addresses.

FINAL REGISTER CONTENTS

The A register store contains the tetrad address of the terminal address in the list. The B register contains the B address of the instruction modified by N_B. The T register contains $(000000)_8$. The P register store is unaltered.

TIMING

Total time in milliseconds (B_2 2^0 = 0) = 3.575 + .03n + .03 (m - 1)

where n is the total number of characters transferred, m is the total number of addresses in the list. If B_2 2^0 = 1 change 3.575 to 4.575.

EXAMPLE (MSW)

Instruction: 13 067000 00 060000

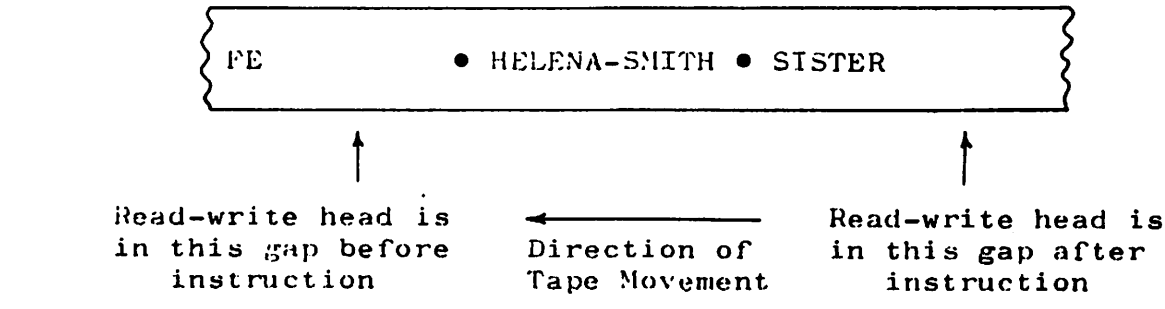
HSM before and after Instruction is executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0670	15 04 17 07	07 04 17 42	00 00 00 00						0670

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0417		•	H E L E	N A - S	M I T H				0417

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0417	• S	I S T E	R • 1 6						0417

Tape (on Tape Station 06) after execution:



FINAL REGISTER CONTENTS:

STA = 067010
B = 060000
T = 000000

TIME:

3.575 + (.03 x 20) + (.03 x 2) = 4.235 ms.

OUTLINE OF LOGIC

A start signal is sent to the designated output unit. The contents of the tetrad addressed by the A Register are brought into the Memory Register, and the A Register is increased by $(000004)_8$. The XXX of the tetrad in the Memory Register is sent to the T Register and to the Bus Adder, and K is placed in the L Register. XXX is examined first; if it is $(000000)_8$ (tested in the Bus Adder), the operation ends with the A Register reset so that it holds the address of the list-terminating tetrad. If XXX is not $(000000)_8$, the contents of the tetrad addressed by the T Register are brought into the Memory Register and the T Register is increased by $(000004)_8$, to address the next tetrad in the sector. The quantity in the L Register is decreased by $(04)_8$; the result is automatically adjusted if XXX does not refer to C_0 of a tetrad or if the quantity in the L Register before decrease was less than $(04)_8$. (Final K, then (in the L Register), will always be $(00)_8$). When the quantity in the L Register after decrease (and adjustment when necessary) is $(00)_8$, the next address in the list is placed in the T Register (and the Bus Adder).

The logic with respect to Buffer filling and delay write-out of the very first character (to magnetic tape) or non-delay (other output units) is the same for Multiple Sector Write as for Single Sector Write.

In MSW, blanks generated on magnetic tape between one KXXX and the next will constitute less than 75 microseconds of tape time.

14 LINEAR READ FORWARD LRF

GENERAL DESCRIPTION

This instruction brings one full message from magnetic or punched paper tape into the HSM. It is a potentially simultaneous instruction.

FORMAT

A address:

Address of the tetrad which is to receive the SM, ED or EF. The SM (ED or EF) will be placed in C_0 of the tetrad no matter which of the four locations is specified by the A address.

B address:

B_1 - address of the input unit; [$(77)_8$ addresses the Paper Tape Reader].

B_2B_3 - ignored.

DIRECTION OF OPERATION

Left to right.

DIRECTION OF TAPE MOVEMENT

Forward.

TERMINAL CONDITION

The operation terminates when an EM, ED or EF has been read. If this character does not fall in a C_3 position, spaces are generated to fill out the tetrad.

FINAL REGISTER CONTENTS

Provided the operation is terminated in the Normal Mode, the A register store contains the address of the EM, ED or EF and the B register contains the B address of the instruction modified by N_B. If the operation is terminated in the simultaneous mode, the S register contains the address of the EM, ED or EF. The T register and P register store are unaltered.

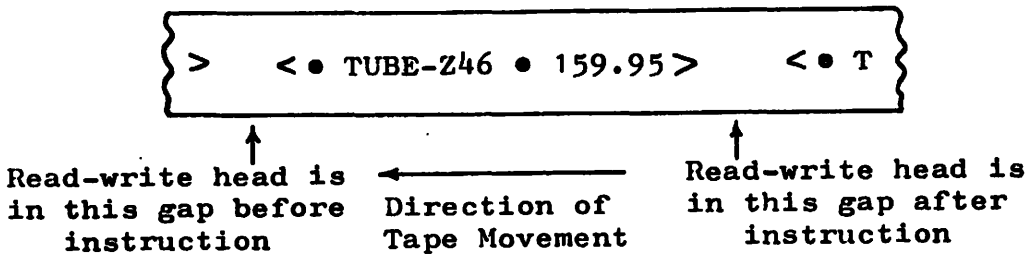
TIMING

Total time in milliseconds = 3.575 + .03n where n is the total number of characters transferred.

EXAMPLE (LRF)

Instruction: 14 116706 00 060000

Tape (on Tape Station 06):



HSM before Instruction is executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
1167	K K	K K K K	K K K K	K K K K	K K K K	K K K K	K K		1167

HSM after Instruction is executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
1167	K K	< • T U	B E - Z	4 6 • 1	5 9 . 9	5 > - -	K K		1167

FINAL REGISTER CONTENTS:

STA or S = 116725

B = 060000 (unless the instruction is concluded in the Simultaneous Mode).

TIME

3.575 + (.03 x 18) = 4.115 ms.

OUTLINE OF LOGIC

A start signal is sent to the designated input device. Coincident with each pulse (or sprocket hole) in the timing track, a character is brought into a recognition circuit. A check is performed to verify that the first character is an SM, ED or EF; if it is not one of these, a CIG alarm stop occurs.

The SM and (unless an EM intervenes) the three characters following it are serially admitted into the upper half of the Read Buffer. These characters are then shifted (four-character parallel transfer) into the lower half of the Read Buffer and then into the tetrad specified by the contents of the A Register, which is then increased by $(000004)_8$. Tape-to-Buffer transfer continues to overlap Buffer-to- HSM transfer until an EM is sensed (in the recognition circuit, i.e., before entering the Buffer), at which point a stop signal is sent to the input device so that no further reading occurs. The instruction is then terminated with the HSM address of the EM in the A (or S) Register.

If the EM is not placed in C_3 of a tetrad, then one, two or three spaces are generated to fill out the tetrad.

Gaps on tape intervening between the SM and the EM are ignored by this instruction. The message is read as if it were free of gaps.

The control symbols ED and EF, when standing alone (with a Gap on either side), are treated as complete messages in themselves.

15 BLOCK READ FORWARD BRF

GENERAL DESCRIPTION

This instruction brings a block of characters from magnetic or punched tape into the HSM. This instruction is potentially simultaneous.

(See discussion of automatic decoding of characters on block read from paper tape, page 4.2).

FORMAT

A address:

Address of the tetrad which will receive the first character in the block; this character will always be placed in C_0 of the tetrad no matter which of its four locations is specified.

B address:

B_1 - address of the input unit; [$(77)_8$ addresses the Paper Tape Reader];

B_2B_3 - ignored.

DIRECTION OF OPERATION

Left to right.

DIRECTION OF TAPE MOVEMENT

Forward.

TERMINAL CONDITION

The operation terminates when a gap on tape is sensed. If the last character does not fall into a C_3 position, spaces are generated to fill out the last tetrad.

FINAL REGISTER CONTENTS

Provided the operation is terminated in the normal mode, the A register store contains the address of the last character in the block and the B register contains the B address of the instruction modified by N_B. If the operation is terminated in the Simultaneous Mode, the S register contains the address of the last character in the block. The T register and P register store are unaltered.

TIMING

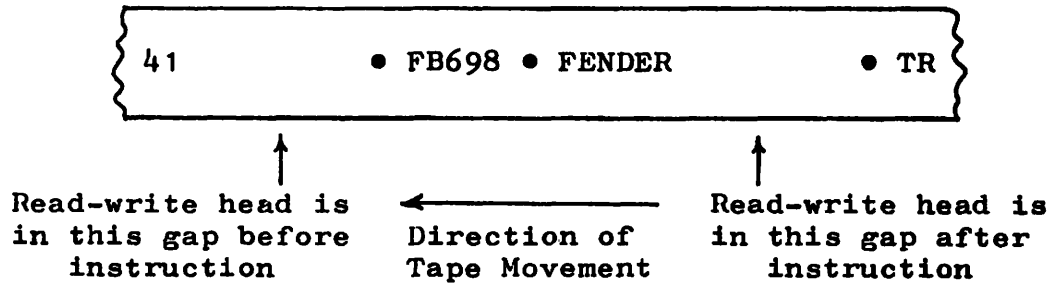
Total time in milliseconds = $3.575 + .03\ (n + 6)$

where n is the total number of characters transferred.

EXAMPLE (BRF)

Instruction: 15 123045 00 010000

Tape (on Tape Station 01):



HSM before Instruction is executed:

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
1230	K K	K K K K	K K K K	K K K K	K K K K	K K			1230

HSM after Instruction is executed:

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
1230	K K	• F B 6	9 8 • F	E N D E	R - - -	K K			1230

FINAL REGISTER CONTENTS:

STA or S = 123060

B = 010000 (unless the operation is concluded in the Simultaneous Mode).

TIME:

3.575 + .03 (13 + 6) = 4.145 ms.

OUTLINE OF LOGIC

A start signal is sent to the designated input unit. When the first four characters following the gap have been placed in the upper half of the Read Buffer, they are shifted (four-character parallel transfer) into the lower half of the Read Buffer and then into the tetrad specified by the A Register, which is then increased by $(000004)_8$. Tape-to-Buffer transfer continues to overlap Buffer-to-HSM transfer until a gap is detected by the input device. If the last character in the block is not placed in C_3 of a tetrad, one, two or three spaces are generated to fill out the tetrad. When the instruction is completed, the A (or S) Register holds the HSM address of the last character read.

An ED or EF standing alone (with a Gap on either side) is treated as a complete block.

16 REWIND n SYMBOLS RNS

GENERAL DESCRIPTION

This instruction causes a selected magnetic tape to be moved backward through a specified number of symbols. This instruction is potentially simultaneous, but no 'read' instruction can be executed simultaneously with it. The HSM is not altered.

FORMAT

A address:

A_1 - the symbol, which may be SM, EF, ED or Gap. Gap is $(00)_8$

A_2A_3 - (in octal count) the number of symbols to be counted.

B address:

B_1 - the address of the Tape Station.

B_2B_3 - ignored.

DIRECTION OF TAPE MOVEMENT

Reverse.

TERMINAL CONDITION

The operation terminates when the tape has been rewound the specified number of symbols, and the tape stops with the Read Write head positioned in the gap (or the gap preceding the SM, ED or EF). If BTC is sensed prior to this condition being satisfied, the operation terminates ~~and the computer stops (alarm).~~ continues to the next instruction.

FINAL REGISTER CONTENTS

If the operation terminates in the normal mode, the A register store contains the specified symbol in the C_1 position, and $(0000)_8$ in the C_2C_3 positions unless the BTC is reached in which case the C_2C_3 positions contain an octal count of the number of symbols remaining to be found. The B register contains the B address of the instruction modified by N_B. If the operation terminates in the Simultaneous Mode, the S register contains the specified symbol in the S_1 position and $(0000)_8$ in the S_2S_3 positions unless BTC is reached, in which case the S_2S_3 positions contain an octal count of the number of symbols remaining to be found.

TIMING

Total time in milliseconds = 3.575 + .03n + 4m

where n is the total number of characters read, including symbols being counted, and m is the number of gaps encountered.

OUTLINE OF LOGIC

A start signal is sent to the specified Tape Station. The characters on the tape pass into the recognition circuit, but do not enter the Read Buffer. When an instance of the specified symbol is found, the Computer is signaled and $(0001)_8$ is subtracted from A_2A_3 by the Bus Adder. When the output of the Bus Adder is $(0000)_8$, or BTC is sensed, a stop signal is sent to the input device and the operation is concluded.

If n = $(0000)_8$ initially, the tape does not move. The instruction, however, is stored in the standard HSM locations for a read instruction.

When rewinding SM's, an alarm stop occurs if the data are not in message format.

When rewinding gaps, if there are fewer than eight characters in a message or block, a CIG alarm stop will occur. Exception: ED and EF stand alone (they may never appear within a message).

If a non-permissible symbol is specified, an alarm stop will occur.

17 REWIND TO BTC RWD

GENERAL DESCRIPTION

This instruction causes a specified magnetic tape to be completely rewound. Once the operation has been initiated, the rewind proceeds totally independent of the Computer, occupying neither the Normal nor the Simultaneous Mode. The Computer, after initiating the rewind, is free to execute other instructions.

FORMAT

A address - ignored.

B address:

B_1 - address of the Tape Station.

B_2B_3 - ignored.

DIRECTION OF TAPE MOVEMENT

Reverse.

FINAL REGISTER CONTENTS

The A register store contains the A address of the instruction modified by N_A. The B register contains the B address of the instruction modified by N_B. All other registers and register stores are unaltered.

TIMING

Total Computer time:

300 μS if the BTC is not positioned at the read-write head when the RWD instruction is given.

105 μS if the BTC is already positioned at the read-write head when the RWD instruction is given.

OUTLINE OF LOGIC

A start signal is sent from the Computer to the specified Tape Station, whereupon the Computer is completely divorced from the operation; rewinding proceeds independently. If, while a tape is rewinding it is selected by an instruction entailing forward movement, the instruction will not be executed until the current rewind operation has been completed (i.e., until the BTC has been reached). If an instruction involving backward movement of a tape is given while it is rewinding, or follows a completed RWD without an intervening operation involving forward movement of that tape, the instruction will merely go through STA (after the BTC is reached) and the Computer will proceed to the next instruction. The RWD instruction is not stored in standard HSM locations.

KDP 10 INSTRUCTIONS 21 - 27

21 ITEM TRANSFER IT

GENERAL DESCRIPTION

This instruction transfers an item from one group of successive HSM locations to another.

FORMAT

A address:

Address of the rightmost character of the item to be transferred.

B address:

Rightmost destination address.

DIRECTION OF OPERATION

Right to left.

TERMINAL CONDITION

The operation terminates when the ISS has been transferred.

FINAL REGISTER CONTENTS

The A register store addresses the first character to the left of the ISS in the original area. The B register addresses the first character to the left of the ISS in the destination area. The T register and P register stores are unaffected.

TIMING

Total time in microseconds = 30n
where n is the number of characters transferred.

EXAMPLE (IT)

Instruction: 21 001011 00 001146

HSM before Instruction is executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0010		K • - 1	2 - K K						0010

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0011	X X X X	X X X X	X X						0011

HSM after Instruction has been executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0010		K • - 1	2 - K K						0010

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0011	X X • -	1 2 - X	X X						0011

FINAL REGISTER CONTENTS:

STA = 001004 PRP is set

B = 001141

TIME:

30 x 5 = 150 μS.

OUTLINE OF LOGIC

The character stored in the location specified by the A register is read out of the HSM. It is then placed in the HSM location specified by the B register. The contents of the A and B registers are each decreased by $(01)_8$, so that they now address the HSM locations immediately to the left of the processed locations. The operation ends when an ISS has been transferred.

If the item to be transferred contains only spaces (to the right of the ISS), PRN is set. If it contains one or more non-space characters, PRP is set.

If the designated destination area overlaps the location of the ISS in the original area, all the contents of the HSM to the left of the location where overlap began will be destroyed.

22 ONE-CHARACTER TRANSFER OCT

GENERAL DESCRIPTION

This instruction transfers the contents of one HSM character position into another HSM character position.

FORMAT

A address:

Address of character to be transferred.

B address:

HSM destination address.

FINAL REGISTER CONTENTS

The A register store addresses the character to be transferred. The B register addresses the transferred character. The T register and P register stores are unaffected.

TIMING

30 μS.

EXAMPLE (OCT)

Instruction: 22 014344 00 014363

HSM before Instruction is executed:

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0143	K K	• K K K			X X X -	1 4 4 -			0143

HSM after Instruction has been executed:

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0143	K K	• K K K			X X X •	1 4 4 -			0143

FINAL REGISTER CONTENTS:

STA = 014344

B = 014363

TIME

30 μS.

OUTLINE OF LOGIC

The character stored in the location specified by the A register is transferred to the location specified by the B register, which terminates the operation.

24 SECTOR TRANSFER BY CHARACTER STC

GENERAL DESCRIPTION

This instruction transfers characters from one sector to another sector.

FORMAT

T register (Preset):

Rightmost destination address.

A address:

Address of leftmost character of the sector to be transferred.

B address :

Address of rightmost character of the sector to be transferred.

DIRECTION OF OPERATION.

Right to left.

TERMINAL CONDITION

The operation terminates when the leftmost character of the sector has been transferred.

FINAL REGISTER CONTENTS

The A register store contains the address of the leftmost character of the sector to be transferred. The B register contains the address of the first character to the left of the sector to be transferred. The T register contains the address of the first character to the left of the transferred sector. The P register store is unaltered.

TIMING

Total time for the operation in microseconds = 30n

where n is the number of characters transferred.

EXAMPLE (STC)

Instructions: 72 003461 00 600000

24 141027 00 141034

HSM before Instructions have been executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
1410					K	K < • 6	1 4 3 2	1 • K K	.1410

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0034			X X X X	X X X X	X X X X				0034

HSM after Instructions have been executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
1410					K	K < • 6	1 4 3 2	1 • K K	1410

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0034			X X X X	6 1 4 3	2 1 X X				0034

FINAL REGISTER CONTENTS:

STA = 141027

B = 141026

T = 003453

TIME:

30 x 6 = 180 μS.

OUTLINE OF LOGIC

The character stored in the location specified by the B register is placed in the location specified by the T register. $(01)_8$ is subtracted from the B register and from the T register. These steps are repeated until the character in the location specified by the A register has been processed

25 THREE-CHARACTER TRANSFER TCT

GENERAL DESCRIPTION

This instruction transfers the contents of the rightmost three character positions of one tetrad to the rightmost three character positions of another tetrad in the HSM. It is useful for placing addresses into stored instructions, setting Address Modifiers, etc.

FORMAT

A address:

Address of the tetrad which contains, in C_1, C_2 and C_3, the characters to be transferred.

B address:

Address of the destination tetrad.

DIRECTION OF OPERATION

Parallel transfer of all three characters.

FINAL REGISTER CONTENTS

The A register store addresses the tetrad from which transfer takes place. The B register addresses the destination tetrad. The T register and P register stores are unaltered.

TIMING

30 μS.

EXAMPLE (TCT)

Instruction: 25 010406 00 000111

HSM before Instruction is executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0104	K	K 07 32 00	K K K K						0104

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0001		X X	X X X X	X X					0001

HSM after Instruction has been executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0104	K	K 07 32 00	K K K K						0104

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0001		X X	X 07 32 00	X X					0001

FINAL REGISTER CONTENTS:

STA = 010406

B = 000111

TIME:

30 μS.

OUTLINE OF LOGIC

The contents of the tetrad addressed by the A register are transferred to the Memory Register, from which the rightmost three characters (C_1, C_2 and C_3) are admitted to, and retained the Bus Adder. The tetrad addressed by the B register is then read out of the HSM, but only the leftmost character (C_0) enters the Memory Register. The contents of the Bus adder are returned to the Memory Register, the entire contents of which are transferred to the tetrad addressed by the B register, completing the operation. The destination tetrad now contains the C_1, C_2 and C_3 characters of the original tetrad; its C_0 character remains unaltered.

26 SECTOR TRANSFER BY TETRAD STT

GENERAL DESCRIPTION

This instruction transfers tetrads of characters from one sector of tetrads to another sector of tetrads. This instruction differs from Sector Transfer by Character (24) in that it transfers four characters at a time, and is, therefore, four times faster.

FORMAT

T Register (Preset):

Address of rightmost destination tetrad.

A address:

Address of the leftmost tetrad of the sector to be transferred.

B address:

Address of the rightmost tetrad of the sector to be transferred.

DIRECTION OF OPERATION

Right to left

TERMINAL CONDITION

The operation terminates when the leftmost tetrad has been transferred.

FINAL REGISTER CONTENTS

The A register store addresses the leftmost tetrad of the sector to be transferred. The B register addresses the tetrad to the left of the sector to be transferred. The T register addresses the tetrad to the left of the transferred sector.

TIMING

Total time in microseconds = 30n
where n is the number of tetrads transferred.

EXAMPLE (STT)

Instructions:	72	160167	00	600000
	26	051721	00	051733

HSM before Instruction is executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0517				K K	R E L Q	A D - T	A P E -	K K	0517

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
1601			X X	X X X X	X X X X	X X X X	X X		1601

HSM after Instruction has been executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0517				K K	R E L O	A D - T	A P E -	K K	0517

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
1601			X X	R E L O	A D - T	A P E -	X X		1601

FINAL REGISTER CONTENTS:

STA = 051721

B = 051717

T = 160153

TIME:

30 x 3 = 90 μS.

OUTLINE OF LOGIC

The rightmost tetrad to be transferred, addressed by the B Register, is read out of the HSM and into the Memory Register and is retained there. The tetrad specified by the T Register is then read out of the HSM but is inhibited from reaching the Memory Register. The contents of the Memory Register are placed in the tetrad designated by the T Register, thus transferring the rightmost tetrad of the sector into its destination location. The contents of the B and T Registers are each decreased by $(04)_8$ and the process is repeated until the tetrad addressed by the A Register (leftmost tetrad in the sector) has been processed.

27 RANDOM DISTRIBUTE RD

GENERAL DESCRIPTION

This instruction redistributes successive items in the HSM, to locations specified by a stored list of addresses.

FORMAT

A address:

Address of leftmost character of leftmost item to be dispersed. (The first character to cause distribution is an ISS, SM or EM. The first character dispersed is an ISS or SM).

B address:

Refers to the stored list of addresses; it specifies the address of the tetrad which contains the destination address for the SM or the ISS of the leftmost item to be dispersed.

The list comprises successive tetrads which contain the destination addresses in their rightmost three locations (C_1, C_2 and C_3) the C_0 character in each of these tetrads is ignored. When the destination address is $(777777)_8$, the associated item is not transferred. The destination address $(000000)_8$ terminates the list.

DIRECTION OF OPERATION

Left to right.

TERMINAL CONDITION

The operation terminates when the address $(000000)_8$ is encountered in the list of addresses.

FINAL REGISTER CONTENTS

The A register store addresses the last control symbol (ISS, SM or EM) found in the original area. The B register addresses the tetrad to the right of the terminal address in the list. The T register contains $(000000)_8$.

TIMING

Total time in microseconds = $33n_1 + 18n_2 + 45n_3$

where: n_1 is the total number of characters transferred; n_2 is the total number of characters whose distribution address is $(777777)_8$; n_3 is the number of distribution addresses left when an EM is found. (The address of the EM must be included in n_3).

EXAMPLE (RD)

Instruction: 27 002604 00 140140

HSM before and after Instruction is executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0026	K K K K	< • R •	C O D E	• S T O	C K # •	Q U A N	T I T Y	>	0026

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
1401	01 00 10 00	00 77 77 77	01 00 10 04	00 00 10 12	00 00 10 44	00 00 10 32	00 00 10 37	00 00 00 00	1401

HSM before the Instruction is executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0010	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	0010

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0010	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	0010

HSM after Instruction has been executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0010	< - - -	• C O D	E - • S	T O C K	# - - -	- - - -	- - • -	- - - •	0010

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0010	- - - -	• Q U A	N T I T	Y - - -	- - - -	- - - -	- - - -	- - - -	0010

FINAL REGISTER CONTENTS

STA = 002634

B = 140200

T = 000000

PRN is set.

TIME:

(33 x 24) + (18 x 2) + (45 x 2) = 918 μS.

OUTLINE OF LOGIC

The leftmost character of the leftmost item addressed by the A register, is read out of the HSM and into the R register. Unless this character is an ISS, SM, or EM, it will not be further processed. [The contents of the A register are increased by $(01)_8$, whether or not the character is transferred.] Instead, the next character to the right is tested. If this is an ISS or SM, the tetrad designated by the B register is read (from the list) into the Memory Register. The rightmost three characters of this tetrad are placed in the T register. If these characters are not $(777777)_8$ or $(000000)_8$, successive transfer of characters begins, starting with the ISS or SM, and continues until another ISS or SM is found. The contents of the B register are increased by $(04)_8$ and the address contained in the next tetrad (to the right) in the list is then placed in the T register, and the process is repeated.

When an EM is found, an ISS (rather than EM) is placed in the location designated by the T register. If, when the EM is found, the list has not been exhausted, ISS's will also be placed in the remaining destination locations designated in the list.

When the destination address contained in the list is $(777777)_8$ the associated item is not transferred; instead, characters are read out successively and the contents of the A register are increased by $(01)_8$ for each character until another ISS or SM is found.

The operation is completed when the terminal address, $(000000)_8$, is found in the list.

If the terminal address $(000000)_8$ is sensed prior to an EM, PRZ is set. If both the terminal address and an EM have been sensed, PRP is set. If an EM has been sensed prior to the terminal address, PRN is set.

KDP 10 INSTRUCTIONS 31 - 37

31 LOCATE n^{th} SYMBOL IN SECTOR L N S

GENERAL DESCRIPTION

This instruction searches through a sector counting the occurrences of a designated symbol.

FORMAT

T Register (Preset):

T_1 - symbol. Symbol cannot be $(00)_8$.

T_2T_3 - count (octal).

A address:

Address of leftmost character in sector to be searched.

B address:

Address of rightmost character in sector to be searched.

DIRECTION OF OPERATION

Left to right.

TERMINAL CONDITION

The operation terminates when (1) the specified count is reached or (2) the rightmost location in the sector has been searched.

FINAL REGISTER CONTENTS

1) If the specified count is reached, the A register store addresses the character to the left of the last symbol counted, the B register addresses the rightmost character in the sector and the T register has the symbol in T_1 and $(0000)_8$ in T_2T_3.

2) If the whole sector has been searched, the A register store addresses the character to the left of the last character in the sector, the B register addresses the rightmost character in the sector and the T register has the symbol in T_1 and an octal count of symbols remaining to be found in T_2T_3.

TIMING

If the specified count is greater than zero and if this sector has more than one character in it:

$$15m + 30n + 45\ \mu S$$

where m = total number of locations searched, and n = total number of occurrences counted.

If the specified count is zero, time is 15 μS.

If the sector is one character in length, time is 60 μS.

EXAMPLE (LNS)

Instructions:	72	740003	00	600000
	31	012506	00	012524

HSM before and after execution of Instruction:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0125		< • R •	B O H •	C O D E	• Q T Y	>			0125

FINAL REGISTER CONTENTS:

STA = 012517

B = 012524

T = 740000

PRZ is set.

TIME:

(15 μS x 11) + (30 μS x 3) + 45 μS = 300 μS.

OUTLINE OF LOGIC

The T_1 character is sent to the L register. The character in the HSM location addressed by the A register is sent to the R register and the contents of the A register are increased by $(01)_8$. Each time the L and R registers compare equal, T_2T_3 is decreased by $(01)_8$. The operation ends (1) when the end of the sector is reached (i.e., when the character in the rightmost location has been examined, and the contents of the A register are equal to those of the B register) or (2) when T_2T_3 is decreased to $(0000)_8$. The A register is then decreased by $(01)_8$, so that it finally contains the address of the HSM location immediately to the left of (1) the rightmost location in the sector searched or (2) the n^{th} symbol.

The PRI's are set as follows:

PRZ if the n^{th} symbol is found before the sector is completely examined.

PRP if the n^{th} symbol is found in the last character position in the sector.

PRN if the sector is completely examined and the n^{th} symbol is not found.

If the octal count in T_2T_3 is initially zero, PRZ is set.

32 ZERO SUPPRESS ZS

GENERAL DESCRIPTION

This instruction deletes non-significant zeros to the left of the most significant character of a decimal number stored in a specified sector.

FORMAT

A address:

Address of leftmost character in sector in which the zeros are to be suppressed.

B address:

Address of rightmost character in sector in which the zeros are to be suppressed.

DIRECTION OF OPERATION

Left to right

TERMINAL CONDITION

The operation terminates when the first non-zero, non-space character is found or a space is found after at least one zero has been suppressed, or when the complete sector has been examined.

FINAL REGISTER CONTENTS

The A register store addresses the character to the left of the terminal character (i.e., first non-zero, non-space character or first space after a zero) unless no terminal character is found in which case it addresses the rightmost character in the sector. The B register addresses the rightmost character in the sector. The T register addresses the terminal character unless there is no terminal character, in which case it addresses the first character to the right of the sector if some zero suppression has taken place or the rightmost character of the sector if only spaces have been found.

TIMING

Normally:

$$15m + 30n + 15\ \mu S$$

If there is no terminal character:

$$15m + 30n$$

where m is the number of spaces preceding the first non-space character and n is the number of zeros suppressed.

If no zeros or spaces were found:

$$30\ \mu S$$

EXAMPLES (ZS)

1. Instruction: 32 024312 00 024317

HSM before Instruction is executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0243			• 0 0	0 5 0 3					0243

HSM after Instruction has been executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0243			• - -	- 5 0 3					0243

FINAL REGISTER CONTENTS:

STA = 024314

B = 024317

T = 024315

TIME:

(30 μS x 3) + 15 μS = 105 μS.

2. Instruction: 32 005103 00 005111

HSM before Instruction is executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0051	-	- 0 0 -	0 1						0051

HSM after Instruction has been executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0051	-	- - - -	0 1						0051

FINAL REGISTER CONTENTS

STA = 005106

B = 005111

T = 005107

TIME:

(15 μS x 2) + (30 μS x 2) + 15 = 105 μS.

OUTLINE OF LOGIC

The character in the location specified by the A register is read out of the HSM, and the contents of the A register are transferred to the T register. If this character is a space, it remains unchanged; if it is a zero, it is replaced by a space. The contents of the A register are increased by $(01)_8$ with each location searched. The contents of the T register are increased by $(01)_8$ with each location processed. These steps are repeated until (1) a space is found after at least one zero has been suppressed, or (2) a character which is neither a space nor a zero is found, or (3) the contents of the A and B registers are equal. At the conclusion of ZS, the A register is reset as shown under Final Register Contents.

33 JUSTIFY RIGHT JR

GENERAL DESCRIPTION

This instruction, used to effect right columnar alignment, (1) adjusts and transfers an item from one series of successive HSM locations to another, or (2) adjusts the item leaving it in the same group of HSM locations. All the space symbols which were originally located to the right of the sign position, are placed between the ISS and the MSD in the destination area. (The sign position is the HSM location immediately to the right of the LSD).

FORMAT

A address:

Address of rightmost character of the item to be justified.

B address:

Address of rightmost destination character position.

Note:

If the A and B addresses are the same and refer to a non-space, non-minus character, a sign will not be generated. (In this case, no change in the item is effected by the JR instruction).

If the A and B addresses are not the same, and the rightmost character of the item is neither a space nor a minus, a space (which in the sign position is interpreted as a plus sign) will be generated in the destination (B address) location.

If the B address falls between the A address and the original location of the ISS, all of memory to the left of, and including, the B address will be destroyed.

DIRECTION OF OPERATION

Right to left.

TERMINAL CONDITION

The operation terminates when the ISS has been transferred.

FINAL REGISTER CONTENTS

The A register store addresses the character to the left of the ISS of the item to be justified. The B register addresses the character to the left of the ISS of the justified item. The T register contains $(000000)_8$ if the first character examined is a space or minus sign, and $(777777)_8$ if the first character examined is a non-space, non-minus character. The P register store is unaltered.

TIMING

Total time in microseconds = 30(n + m)

where n = number of space and/or minus characters to the right of the first non-minus, non space character encountered.

where m = total number of characters in the item (i.e. including n and ISS).

EXAMPLES (JR)

1. Instruction: 33 001433 00 154172

HSM before Instruction is executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0014					X X X •	1 3 6 9	4 8 - -	X X X X	0014

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
1541					K K K K	K K K K	K K K K	K K K	1541

HSM after Instruction is executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0014					X X X •	1 3 6 9	4 0 - -	X X X X	0014

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
1541					K K • -	- 1 3 6	9 4 0 K	K K K	1541

FINAL REGISTER CONTENTS:

STA = 001422

B = 154161

T = 000000

TIME:

30(3 + 9) = 360 microseconds.

2. Instruction: 33 015157 00 015157

HSM before Instruction is executed:

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0151		•	3 9 6 4	1 - - -					0151

HSM after Instruction has been executed:

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0151		•	- - 3 9	6 4 1 -					0151

FINAL REGISTER CONTENTS:

STA = 015146

B = 015146

T = 000000

TIME:

30(3 + 9) = 360 microseconds.

3. Instruction: 33 022663 00 022663

HSM before and after Instruction is executed:

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0226				• 4 1 2	3 9 6 7				0226

FINAL REGISTER CONTENTS:

STA = 022653

B = 022653

T = 777777

TIME:

30(0 + 8) = 240 microseconds.

OUTLINE OF LOGIC

Beginning with the HSM location specified by the A register, and proceeding to the left, the characters in the item are successively processed. With each space or minus sign found, $(01)_8$ is added to the contents of the T register which was automatically set to $(000000)_8$ during staticising. When the first non-space non-minus character is found, the sign is generated in the location specified by the B register and the contents of the T register are decreased by $(01)_8$. The sign of the item will be minus if a minus sign is sensed in any of the locations between the A address and the rightmost non-minus, non-space character. The remaining characters addressed by the A register are then transferred into the locations specified by the B register. The A and B registers are decreased by $(01)_8$ with each location processed. When the A register addresses a location which contains an ISS, space symbols equal in number to the contents of the T register are generated in the destination locations immediately to the left of the MSD. The ISS is then transferred, completing the operation.

If the item contains only spaces and an ISS or only an ISS, PRN is set. It it contains any non-space character (in addition to the ISS), PRP is set.

34 SECTOR CLEAR BY CHARACTER SCC

GENERAL DESCRIPTION

This instruction places space characters in a specified sector.

FORMAT

A address:

Address of leftmost character in the sector to be cleared.

B address:

Address of rightmost character in the sector to be cleared.

DIRECTION OF OPERATION

Right to left.

TERMINAL CONDITION

The operation terminates when the leftmost character in the sector has been replaced by a space.

FINAL REGISTER CONTENTS

The A register store addresses the leftmost character in the sector. The B register addresses the first character to the left of the sector.

TIMING

Total time in microseconds = 15n, where n is the total number of locations cleared.

EXAMPLE (SCC)

Instruction: 34 036143 00 036152

HSM before Instruction is executed:

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0361	K K K K	K K K K	K K K K						0361

HSM after the Instruction has been executed:

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0361	K K K -	- - - -	- - - K						0361

FINAL REGISTER CONTENTS:

STA = 036143

B = 036142

TIME:

15 x 8 = 120 μS.

OUTLINE OF LOGIC

The tetrad containing the rightmost character to be cleared, specified by the B register, is read out of the HSM. This character, however, is inhibited from reaching the Memory Register. Instead, a space is generated and inserted into the empty location in the Memory Register. When regeneration occurs, the tetrad, with the inserted space, is returned to the HSM. The contents of the B register are decreased by $(01)_8$ with each location cleared. This process is repeated until the leftmost character, specified by the A register, is replaced with a space, so that the entire sector is cleared.

When HSM locations not included in the system are addressed, the Computer will perform as much of the instruction as possible and then continue on to the next instruction in sequence. Sector Clear by Character does not stop the Computer, with parity error notification, when it contains a programming error of this type.

35 SECTOR COMPRESS-RETAIN REDUNDANT ISS's SCR

GENERAL DESCRIPTION

This instruction transfers a sector of characters from one part of the HSM to another, removing, in the process, all spaces located to the right of the rightmost non-space character within each item in the sector.

Note: A space in the sign position (i.e., positive sign) is also deleted.

FORMAT

T register (preset)

Rightmost destination address.

A address:

Address of leftmost character of the sector to be compressed and transferred.

B address:

Rightmost HSM location of the sector to be compressed and transferred.

DIRECTION OF OPERATION

Right to left.

TERMINAL CONDITION

The operation terminates when the leftmost character in the sector has been transferred.

FINAL REGISTER CONTENTS

The A register store addresses the leftmost character of the sector to be compressed. The B register addresses the first character to the left of the sector to be compressed. The T register addresses the first character to the left of the compressed sector. The P register store is unaltered.

TIMING

Total time in microseconds = 15n + 15m

where n is the number of characters actually transferred, and m is the total number of characters in the original sector.

EXAMPLE (SCR)

Instructions: 72 011776 00 600000

35 006200 00 006235

HSM before the instructions are executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0062	< • J O	N E S -	- - • 8	2 4 B -	- • - 1	2 • 0 4	• • • -	- >.	0062

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0117	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	0117

HSM after the instructions have been executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0062	< • J O	N E S -	- - • 6	2 4 B -	- • - 1	2 • 0 4	• • • -	- >	0062

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0117	- - - -	- - - -	< • J O	N E S •	6 2 4 B	• - 1 2	• 0 4 •	• • > -	0117

FINAL REGISTER CONTENTS:

STA = 006200

B = 006177

T = 011747

TIME:

(15 μS x 23) + (15 μS x 30) = 795 μS.

OUTLINE OF LOGIC

The rightmost character of the sector, specified by the contents of the B register, is read out of the HSM and stored in the R register. The contents of the B register are decreased by $(01)_8$. If the character is a space symbol, it is not transferred; instead, the next character to the left is read out and tested. When a non-space character is found, it is transferred to the location specified by the T register. The Computer then continues sequential transfer of all the characters to the left of this location until an ISS or EM is transferred. The contents of the T register are decreased by $(01)_8$ with each transfer. Beginning with the location to the left of the ISS or EM, the Computer again tests for space symbols and does not resume transferring until the next non-space character is found. These steps are repeated until the leftmost character of the sector, specified by the A register, has been processed.

If the sector contains any non-space, non-EM, non-ISS characters, PRP is set. If it contains only space, EM, and/or ISS characters, PRN is set.

36 SECTOR CLEAR BY TETRAD SCT

GENERAL DESCRIPTION

This instruction places space characters in a specified sector comprising an exact number of tetrads. It differs from the Sector Clear by Character Instruction (34) in that it clears four characters at a time, and is, therefore, four times faster.

FORMAT

A address:

Address of leftmost tetrad to be cleared.

B address:

Address of rightmost tetrad to be cleared.

DIRECTION OF OPERATION

Right to left.

TERMINAL CONDITION

The operation terminates when the leftmost tetrad in the sector has been cleared to spaces.

FINAL REGISTER CONTENTS

The A register store addresses the leftmost tetrad in the sector. The B register addresses the first tetrad to the left of the sector. The T register and P register stores are unaffected.

TIMING

Total time in microseconds = 15 n, where n is the number of tetrads cleared.

EXAMPLE (SCT)

Instruction: 36 141360 00 141365

HSM before Instruction is executed:

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
1413				K K K K	K K K K	K K K K	K K K K		1413

HSM after Instruction has been executed:

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
1413				K K K K	- - - -	- - - -	K K K K		1413

FINAL REGISTER CONTENTS:

STA = 141360

B = 141355

TIME:

15 x 2 = 30 μS.

OUTLINE OF LOGIC

The characters in the rightmost tetrad to be cleared, specified by the B register, are read out of the HSM, and four spaces are generated and inserted in their place.

The contents of the B register are decreased by $(04)_8$ and the tetrad immediately to the left of the cleared tetrad is read out of the HSM. The process is repeated for each tetrad until the leftmost tetrad in the sector, specified by the A register, is filled with spaces. The Computer will attempt to carry out the instruction even when HSM locations not included in the system are addressed, performing as much of the instruction as possible before continuing on to the next instruction in sequence. Sector Clear by Tetrad does not stop the Computer, with parity error notification, when it contains a programming error of this type.

37 SECTOR COMPRESS-DELETE REDUNDANT ISS's SCD

GENERAL DESCRIPTION

This instruction transfers a sector from one part of the HSM to another, deleting, in the process, (1) all ISS's originally located to the right of the rightmost non-ISS, non-space character in the sector, and (2) all spaces located to the right of the rightmost non-space character within each item in the sector. Note: A space in the sign position (i.e., positive sign) is also deleted.

FORMAT

T register (preset):

Rightmost destination address.

A address:

Address of leftmost character in sector to be compressed.

B address:

Address of rightmost character in sector to be compressed.

DIRECTION OF OPERATION

Right to left.

TERMINAL CONDITION

The operation terminates when the leftmost character in the sector has been transferred.

FINAL REGISTER CONTENTS

The A register store addresses the leftmost character of the sector to be compressed. The B register addresses the first character to the left of the sector to be compressed. The T register addresses the first character to the left of the compressed sector.

TIMING

Total time in microseconds = 15n + 15m

where n is the number of characters actually transferred and m is the total number of characters in the original sector.

EXAMPLE (SCD)

Instructions: 72 011775 00 600000
37 006200 00 006234

HSM before Instructions are executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0062	< • J O	N E S -	- - • 6	2 4 B -	- • - 1	2 • 0 4	• • • -	- >	0062

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0117	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	0117

HSM after Instructions have been executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0062	< • J O	N E S -	- - • 6	2 4 B -	- • - 1	2 • 0 4	• • • -	- >	0062

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0117	- - - -	- - - -	- - - <	• J O N	E S • 6	2 4 B •	- 1 2 •	0 4 - -	0117

FINAL REGISTER CONTENTS:

STA = 006200

B = 006177

T = 011752

PRP is set.

TIME:

(15 μS x 19) + (15 μS x 29) = 720 μS.

OUTLINE OF LOGIC

The rightmost character of the sector to be compressed is fetched using the address in the B register, and then tested. If it is an ISS or a space symbol, it is not transferred; instead the next character to the left is read out and tested. When a non-space, non-ISS character is found, it is transferred to the location specified by the T register. From this point on, until the leftmost character of the sector, specified by the A register, has been processed, transfer is continuous, excluding groups of successive spaces immediately to the left of ISS's but including all the remaining ISS's in the sector.

Note: If the B address is that of the EM or of a location to the right of the EM, redundant ISS's to the left of the EM will not be deleted.

If the sector contains any non-space, non-ISS characters, PRP is set. If it contains only spaces and/or ISS characters, PRN is set.

KDP 10 INSTRUCTIONS 41 - 47

NOTE: All the Instructions in this group perform Binary Operations and two instructions in the group include the term Binary in the name of the instruction. It should be pointed out that it would be equally true to describe the function of these instructions as Octal Operations. It is not necessary to describe the instructions in terms of binary patterns and all the data provided in the illustrative examples is given in octal notation.

41 BINARY ADD BA

GENERAL DESCRIPTION

This instruction performs binary addition of two equal length operands and places the sum in the HSM locations originally occupied by the augend. The operands may be of any length. Each character (including control symbols) is treated as if it were numeric.

FORMAT

T register (preset):

Address of rightmost character of addend.

A address:

Address of the leftmost character of the augend (and sum)

B address:

Address of the rightmost character of the augend (and sum).

DIRECTION OF OPERATION

Right to left.

TERMINAL CONDITION

The operation terminates when the leftmost characters of the operands have been added and their sum stored.

FINAL REGISTER CONTENTS

The A register store addresses the leftmost character of the sum. The B register addresses the first character to the left of the sum. The T register addresses the first character to the left of the addend. The P register store is unaltered.

TIMING

Total time in microseconds = 45n, where n is the number of characters in the augend.

EXAMPLE (BA)

Instruction: 41 000703 00 000707

T register is preset to 002155

HSM before Instruction is executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0007	X X 40	30 20 10 01	X X						0007

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0021		K K	K 42 01 74	05 05 K K					0021

HSM after Instruction has been executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0007	X X 02	32 14 15 08	X X						0007

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0021		K K	K 42 01 74	05 05 K K					0021

FINAL REGISTER CONTENTS:

STA = 000703
B = 000702
T = 002150

TIME:

45 x 5 = 225 μS.

OUTLINE OF LOGIC

Binary add is executed according to the following rules:

1. There is no search for the signs or least significant characters of operands and there are no special control symbols. Every character enters into the addition and is treated as if it were numeric.

2. The addition considers all six bits of each character in the operands.

3. Any carry from the most significant bit position of the leftmost character of the sum is discarded.

4. The PRI's are not affected by BA.

5. The rules for binary addition may be stated as follows:

	Sum	Carry
0 + 0 =	0	0
0 + 1 =	1	0
1 + 1 =	0	1
1 + 1 + 1 =	1	1

The B register is used to address the augend and the sum. Immediately before a character is picked up from the augend the contends of the A and B registers are matched for equality. The BA operation is concluded after A - B equality is attained.

42 BINARY SUBTRACT B S

GENERAL DESCRIPTION

This instruction performs binary subtraction of one operand from another operand of equal length, placing the difference in the HSM locations originally occupied by the minuend. The operands may be of any length. Each character (including control symbols) is treated as if it were numeric.

FORMAT

T register (preset):

Address of rightmost character of subtrahend.

A address:

Address of the leftmost character of the minuend (and difference).

B address:

Address of the rightmost character of the minuend (and difference).

DIRECTION OF OPERATION

Right to left.

TERMINAL CONDITION

The operation terminates when the leftmost characters of the operands have been differenced.

FINAL REGISTER CONTENTS

The A register store addresses the leftmost character of the difference.
The B register addresses the first character to the left of the difference.
The T register addresses the first character to the left of the subtrahend.
The P register store is unaltered.

TIMING

Total time in microseconds = 45n
where n is the number of characters in the minuend.

EXAMPLE (BS)

Instruction: 42 001105 00 001111
T register is set to 001067.

HSM before Instruction is executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0011	X	X 01 44 54	64 74 X X						0011

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0010					K K K 02	05 04 04 04	K K		0010

HSM after Instruction has been executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0011	X	X 77 37 50	60 70 X X						0011

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0010					K K K 02	05 04 04 04	K K		0010

FINAL REGISTER CONTENTS:

STA = 001105
B = 001104
T = 001062

TIME:

45 x 5 = 225 μS.

OUTLINE OF LOGIC

Binary Subtract works according to the following rules:

1. There is no search for signs or the least significant characters of operands and there are no special control symbols. Every character enters into the subtraction and is treated as if it were numeric.

2. The PRI's are set according to the relative magnitude of the operands. However, the sign of the difference is not stored in the HSM.

   PRI settings:

   PRZ if the difference = $(0)_8$.
   PRP if minuend > subtrahend.
   PRN if minuend < subtrahend.

3. All six bits of each character in the operands are considered in the subtraction.

4. The 'ones' complement of the subtrahend is taken and a binary addition is performed, with a 'one bit' automatically added to the least significant bit position of the difference.

5. Any carry from the most significant bit position of the difference is discarded. If there is a carry, the difference is positive; if no carry, difference is negative.

6. The digits of the minuend are replaced, one by one, with digits of the difference as the operation proceeds.

7. Binary Subtract ends after A - B equality has been attained.

43 SECTOR COMPARE SC

GENERAL DESCRIPTION

This instruction is used to determine the relative magnitude of two operands of equal length. A single operand may consist of one character or any number of alpha-numeric characters and/or symbols. Binary subtraction is performed, but the difference is not stored in the HSM. However, the resultant PRI settings permit alternative sequences of instructions to be executed.

FORMAT

T register (preset)

Address of rightmost character of the subtrahend.

A address:

Address of the leftmost character of the minuend.

B address:

Address of the rightmost character of the minuend.

DIRECTION OF OPERATION

Right to left.

TERMINAL CONDITION

The operation terminates when the leftmost characters in the sectors have been differenced. PRP will be set if the result is positive, PRZ if the result is zero and PRN if the result is negative.

FINAL REGISTER CONTENTS

The A register store addresses the leftmost character of the minuend. The B register addresses the first character to the left of the minuend. The T register addresses the first character to the left of the subtrahend.

TIMING

Total time in microseconds = 45n
where n is the number of characters in the minuend.

EXAMPLES (SC)

1. Instructions: 72 020217 00 600000
43 010104 00 010112

HSM before and after Instruction is executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0101	X X •	A J 8 1	0 4 3 X	X X					0101

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0202		K K	• A F 7	0 0 0 0	K K				0202

FINAL REGISTER CONTENTS:

STA = 010104
B = 010103
T = 020210

PRP is set.

TIME:

45 x 7 = 315 μS.

2. Instructions: 72 021556 00 600000
43 012315 00 012315

HSM before and after Instruction is executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0123				X < • X	X X				0123

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0215				K K < K	K				0215

FINAL REGISTER CONTENTS:

STA = 012315
B = 012314
T = 021555

PRZ is set

TIME:

45 μS.

OUTLINE OF LOGIC

Subtraction is performed exactly as in Binary Subtract (42) except that the difference is not stored in the HSM.

44 THREE CHARACTER ADD TCA

GENERAL DESCRIPTION

TCA is designed to modify addresses of instructions and to keep octal counters. It performs binary addition of an augend stored in the rightmost three locations of a tetrad and a three-character addend. The result is automatically stored in the locations previously occupied by the augend.

FORMAT

A address:

Address of rightmost character of the augend (and sum).

B address:

Address of rightmost character of the addend.

DIRECTION OF OPERATION

Right to left.

TERMINAL CONDITION

The operation terminates when the C_1 character of the augend has been replaced by the sum.

FINAL REGISTER CONTENTS

The A register store addresses the first character to the left of the sum. The B register addresses the first character to the left of the addend. The T register and P register store are unaffected.

TIMING

Time in microseconds = 45n
n = 1, 2, 3 or 4.

This can be a 1, 2, 3 4 Characters add depending on the C position of the Augend

EXAMPLE (TCA)

Instruction: 44 001003 00 001226

HSM before Instruction is executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0010	12 76 25 74	X X X X							0010

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0012					K K	40 07 13 K	K K		0012

HSM after the Instruction has been executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0010	12 36 35 07	X X X X							0010

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0012					K K	40 07 13 K	K K		0012

FINAL REGISTER CONTENTS:

STA = 001000
B = 001223

TIME:

45 x 3 = 135 μS.

OUTLINE OF LOGIC

The TCA instruction operates according to the following rules:

1. There is no search for the least significant characters or signs of the operands. Every character enters into the addition.

2. The PRI's are not affected by the TCA, and a sign is not stored in the sum.

3. The addition considers all six bits of the characters in the operands.

4. Any carry from the most significant bit position of the sum is discarded.

5. All characters are treated as numeric; there are no special control symbols. The addition ends when the two least significant bits in the A register are 01. (Note that, although this termination condition is constant, addition actually starts with the characters in the locations designated by the A and B addresses. TCA can, therefore, also be used as a one or two-character add if the A address refers to C_1 or C_2 of a tetrad, or as a four-character add if the A address refers to C_0 of the tetrad to the right. This is also applicable to the Three-Character Subtract instruction).

6. The sum replaces the augend in the HSM, character for character.

45 THREE CHARACTER SUBTRACT TCS

GENERAL DESCRIPTION

Like Three-Character Add, TCS is designed for address modifications and octal counters. It performs binary subtraction of a minuend stored in the rightmost three locations of a tetrad and a three-character subtrahend. The difference is stored in the locations previously occupied by the minuend.

FORMAT

A address:

Address of rightmost character of the minuend (and difference).

B address:

Address of rightmost character of the subtrahend.

This can also be a 1, 2, 3, 4 character subtract depending on the C portion of the Minuend.

DIRECTION OF OPERATION

Right to left.

TERMINAL CONDITION

The operation terminates when the C_1 character of the minuend has been replaced by the difference.

FINAL REGISTER CONTENTS

The A register store addresses the first character to the left of the difference. The B register addresses the first character to the left of the subtrahend. The T register and P register store are unaffected.

TIMING

Time in microseconds = 45n
n = 1, 2, 3 or 4.

EXAMPLE (TCS)

Instruction: 45 001167 00 001313

HSM before Instruction is executed:

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0011					X	X 04 56 31	X X		0011

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0013		K	K 00 00 10	K K					0013

HSM after Instruction has been executed:

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0011					X	X 04 56 21	X X		0011

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0013		K	K 00 00 10	K K					0013

FINAL REGISTER CONTENTS:

STA = 001164
B = 001310

TIME:

45 x 3 = 135 μS.

OUTLINE OF LOGIC

The TCS instruction operates according to the following rules:

1. Every character enters into the subtraction. There is no search for the signs or least significant characters of operands and there are no special control symbols. The operation ends when the two least significant bits in the A register are 01.

2. A sign is not stored to the right of the difference, although the PRI's are properly set to reflect the relative magnitudes of the operands.

3. The subtraction considers all six bits of characters in the operands.

4. The subtrahend is complemented ('ones' complement) and a binary addition is performed.

5. A '1' (complementary carry) is automatically added into the least significant bit position of the difference. Any carry from the most significant position is discarded. If there was a carry, the difference is positive; if no carry, the difference is negative.

6. The difference replaces the minuend in the HSM.

46 LOGICAL 'OR' LO

GENERAL DESCRIPTION

This instruction inserts '1' bits from a specified modifier into a specified sector of equal length.

FORMAT

T register (preset):

Address of rightmost character of modifier.

A address:

Address of the leftmost character of the sector to be modified.

B address:

Address of the rightmost character of the sector to be modified.

DIRECTION OF OPERATION

Right to left.

TERMINAL CONDITION

The operation terminates when the leftmost character of the sector has been modified.

FINAL REGISTER CONTENTS

The A register store addresses the leftmost character of the modified sector. The B register addresses the first character to the left of the modified sector. The T register addresses the first character to the left of the modifier sector.

TIMING

Total time in microseconds = 45n
where n is the number of characters in the operand to be modified.

EXAMPLE (LO)

Instructions: 72 012451 00 600000
46 015057 00 015060

HSM before Instruction is executed:

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0150				K K 06	03 K K				0150

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0124		X X X	50 43 X X	X					0124

HSM after Instruction has been executed:

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0124		X X X	50 43 X X	X					0124

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0150				K K 56	43 K K				0150

FINAL REGISTER CONTENTS:

STA = 015057
B = 015056
T = 012447

TIME:

45 x 2 = 90 μS.

OUTLINE OF LOGIC

The operation is performed by a series of bit additions on aligned pairs of characters, proceeding from right to left. All six bits of each character enter the addition, but a carry is not propagated from one bit position to the next. There are no special control symbols.

The rules for the addition are as follows:

0 + 0 = 0
1 + 0 = 1
0 + 1 = 1
1 + 1 = 1

For example,

$$\begin{array}{rr} & 011010 \\ + & 101100 \\ \hline & 111110 \end{array}$$

The sum (modified operand) is placed, character by character, in the HSM locations originally occupied by the operand to be modified.

The PRI's are not affected by LO.

47 LOGICAL 'AND' LA

GENERAL DESCRIPTION

This instruction extracts '1' bits from a sector according to a second sector ('mask') of equal length.

FORMAT

T register (preset)

Address of rightmost character of the mask.

A address:

Address of the leftmost character of the sector to be masked.

B address:

Address of the rightmost character of the sector to be masked.

DIRECTION OF OPERATION

Right to left.

TERMINAL CONDITION

The operation terminates when the leftmost character of the sector has been masked.

FINAL REGISTER CONTENTS

The A register store addresses the leftmost character of the masked sector. The B register addresses the first character to the left of the masked sector. The T register addresses the first character to the left of the mask sector.

TIMING

Total time in microseconds = 45n
where n is the number of characters in the operand to be modified.

EXAMPLE (LA)

Instructions:

72	014311	00	600000
47	015057	00	015060

HSM before Instruction is executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0143		X X	07 07 X X	X					0143

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0150				K K 56	43 K K				0150

HSM after Instruction has been executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0143		X X	07 07 X X	X					0143

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0150				K K 06	03 K K				0150

FINAL REGISTER CONTENTS:

STA = 015057
B = 015056
T = 014307

TIME:

45 x 2 = 90 μS.

OUTLINE OF LOGIC

The operation is performed by a series of bit multiplications on aligned pairs of characters, proceeding from right to left. All six bits of each character enter the multiplication, with no carry propagated from one bit position to the next. There are no special control symbols.

The rules for the multiplication are as follows:

$$0 \times 0 = 0$$
$$0 \times 1 = 0$$
$$1 \times 0 = 0$$
$$1 \times 1 = 1$$

For example,

$$\begin{array}{rl} & 011010 \\ \times & 101100 \\ \hline & 001000 \end{array}$$

Characters in the operand to be modified are replaced, one by one, by characters of the product.

The PRI's are not affected by this instruction.

KDP 10 INSTRUCTIONS 51 - 54

51 DECIMAL ADD DA

GENERAL DESCRIPTION

This instruction performs decimal addition in accordance with algebraic rules, producing a variable length non-zero-suppressed sum, which is stored in the HSM locations originally occupied by the augend. The operands may be of any length and, also, of unequal lengths.

FORMAT

A address:

Address of the rightmost character (ISS, sign or space to the right of the sign) of the augend (and sign of the sum).

B address:

Address of the rightmost character (ISS, LSD, sign or space to the right of the sign) of the addend.

DIRECTION OF OPERATION

Right to left.

TERMINAL CONDITION

The operation terminates when the ISS to the left of the most significant digit of the sum has been stored. The length of each operand is defined by the first space to the left of a non-space, non-minus character or by an ISS.

FINAL REGISTER CONTENTS

The A register store addresses the first character to the left of the ISS or space that terminated the augend. The B register addresses the first character to the left of the ISS or space that terminates the addend. The T register addresses the ISS of the sum. The P register store is unaltered.

TIMING

Time in microseconds = $15n_1 + 45n_2 + 30n_3 + 90$

n_1 is the total number of space and/or minus characters found to the right of both operands;

n_2 is the number of digits in the shorter operand;

n_3 is the difference in number of digits of the operands.

For negative sum, add 30 (n + 1) + 15 where n = number of digits in the sum, (if this results from unlike signs in original operands).

EXAMPLES (DA)

1. Instruction: 51 001016 00 000524

HSM before Instruction is executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0010			X X X •	1 2 - X					0010

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0005				K	K • 9 9	- K K K			0005

HSM after Instruction has been executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0010			X X • 1	1 1 - X					0010

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0005				K	K • 9 9	- K K K			0005

FINAL REGISTER CONTENTS:

STA = 001012
B = 000520
T = 001012

PRP is set.

TIME:

(15 x 2) + (45 x 2) + 90 = 210 μS.

2. Instruction: 51 002023 00 001545

HSM before Instruction is executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0020				X X -	- - - •	X X			0020

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0015	K K - 4	9 9 K K							0015

HSM after Instruction has been executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0020				X X -	• 4 9 9	X X			0020

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0015	K K - 4	9 9 K K							0015

FINAL REGISTER CONTENTS:

STA = 002022
B = 001541
T = 002020
PRN is set

TIME:

(15 x 1) + (30 x 2) + 90 + 30 (2 + 1) + 15 μS. = 270 μS

3. Instruction: 51 003031 00 012560

HSM before Instruction is executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0030				X X • 1	2 6 - -	- - - -	- - X X		0030

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0125			K	K K • 5	9 K K K				0125

HSM after Instruction has been executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0030				X X • 1	2 6 - -	- • 1 8	5 - X X		0030

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0125			K	K K • 5	9 K K K				0125

FINAL REGISTER CONTENTS:

STA = 003015
B = 012555
T = 003025

PRP is set

TIME:

(15 x 8) + (45 x 2) + (30 x 1) + 90 μS. = 330 μS.

OUTLINE OF LOGIC

The initial contents of the A register (HSM address of the rightmost character of the augend) are transferred to the T register, which is used thereafter to place the sum in the HSM. Next, a search is made in each operand for the rightmost non-space, non-minus character. The contents of the A register are decreased by $(01)_8$ with each location searched in the augend; the contents of the B register are decreased by $(01)_8$ with each location searched in the addend.

The search is concluded for each operand when the character addressed is neither space nor minus.

The sign of the sum is then placed in the HSM location specified by the contents of the T register: The contents of the T register are then decreased by $(01)_8$. If the signs are unlike, the sum is initially assumed to be positive and is changed later, if necessary. The searching process, with decrease of A and B register contents, permits right column alignment of the operands so that the first addition is performed on the least significant digits.

After a sign has been stored, addition takes place as follows:

1. If the signs of the operands are unlike, the digits of the negative operand are complemented (tens complement) before addition takes place.

2. The tetrad containing the LSD of the augend is read out of the HSM and into the Memory Register, and the contents of the A register are decreased by $(01)_8$. If this character is not a control symbol, it is transferred to, and retained in, the L register in the Arithmetic Unit.

3. The tetrad containing the LSD of the addend is read out of the HSM and into the Memory Register, and the contents of the B register are decreased by $(01)_8$. If this character is not a control symbol, it is transferred to the R register in the Arithmetic Unit.

4. The contents of the L register are added to those of the R register, and the LSD of the sum is placed in the HSM location specified by the T register. The contents of the T register are then decreased by $(01)_8$. The carry, if any, is retained in the Arithmetic Unit.

The process described above is repeated until the end of either operand is reached. An operand is considered ended when an ISS is sensed or when a space to the left of the MSD is sensed.

If the operands were of unequal length, the carry (if any) is added to the next digit(s) of the longer operand. When there is no carry, a zero is added to each of the remaining digits of the longer operand to complete the addition.

If the signs of the operands were alike, or if they were unlike and the sum is positive, the operation is concluded with an ISS placed in the HSM location immediately to the left of the MSD of the sum. However, if the signs of the operands are unlike and the sum is negative, the sum is re-complemented and the initially assumed positive sign is changed to negative. An ISS is placed to the left of the sum as in the above cases.

If the A register initially addresses an ISS and the B register initially addresses the LSD or the sign of the addend, the addend will be transferred into the augend (sum) locations, with the sign stored in the location that originally held the ISS of the augend. In this case, if the LSD of the addend is initially addressed, the sign will be assumed to be plus. If both the A and B registers initially address an ISS, a space (plus sign) will be stored in the location originally occupied by the ISS of the augend; followed by an ISS immediately to the left.

If both operands contain only space and/or minus characters to the right of the ISS, the sign will be placed in the location initially addressed by the A (and T) register, and an ISS in the location immediately to the left of the sign.

Non-significant zeros are not suppressed by this instruction. When the result of an addition is zero, zeros will appear in the sum locations, and the sign will be positive. The PRI's are set after the instruction according to the sign of the sum, except that a zero result will set PRZ.

Note: The first character addressed in the augend must be a space, minus or ISS. If it is any other character, an alarm stop occurs.

52 DECIMAL SUBTRACT DS

GENERAL DESCRIPTION

This instruction performs decimal subtraction in accordance with algebraic rules, producing a variable length, non zero-suppressed difference, which is stored in the HSM locations originally occupied by the minuend. The operands may be of any length and also of unequal lengths.

FORMAT

A address:

HSM location of the rightmost character (sign or space to the right of sign) of the minuend (and sign of the difference).

B address:

HSM location of the rightmost character (LSD, or sign, or space to the right of the sign) of the subtrahend.

DIRECTION OF OPERATION

Right to left.

TERMINAL CONDITION

The operation terminates when the ISS to the left of the most significant digit of the difference has been stored. The length of each operand is defined by the first space to the left of a non-space, non-minus character or by an ISS.

FINAL REGISTER CONTENTS

The A register store addresses the first character to the left of the ISS or space that terminated the minuend. The B register addresses the first character to the left of the ISS or space that terminated the subtrahend. The T register addresses the ISS of the difference. The P register store is unaltered.

TIMING

Time in microseconds = $15n_1 + 45n_2 + 30n_3 + 90$

where n_1 = total number of spaces and/or minuses to the right of both operands.

n_2 = number of digits in shorter operand.

n_3 = difference in number of digits of the operands.

For negative results, add $30(n+1) + 15$ if this results from like sign in the original operands where n = number of digits in the difference.

EXAMPLES (DS)

1. Instruction: 52 010031 00 005755

HSM before Instruction is executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0100					X	X • 1 2	1 - X X	X	0100

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0057		K	K K • 0	5 - K K	K				0057

HSM after Instruction has been executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0100					X	X • 1 1	6 - X X	X	0100

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0057		K	K K • 0	5 - K K	K				0057

FINAL REGISTER CONTENTS:

STA = 010024
B = 005751
T = 010025
PRP is set

TIME:

(15 x 2) + (45 x 2) + (30 x 1) + 90 μS. = 240 μS.

2. Instruction: 52 012047 00 006761

HSM before Instruction is executed:

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0120	X • 1 2	- - - -	X X X						0120

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0067			K	K K K -	1 2 K K	K			0067

HSM after Instruction has been executed:

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0120	X • 1 2	• 0 0 -	X X X						0120

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0067			K	K K K -	1 2 K K	K			0067

FINAL REGISTER CONTENTS:

STA = 012040
B = 006756
T = 012044
PRZ is set

TIME:

(15 x 4) + (45 x 2) + 90 μS. = 240 μS.

3. Instruction: 52 014051 00 007763

HSM before Instruction is executed:

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0140	K	K K K -	8 0 K K	K					0140

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0077				X X	X • 9 -	X X			0077

HSM after Instruction has been executed:

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0140	K	K K • 1	7 9 K K	K					0140

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0077				X X	X • 9 -	X X			0077

FINAL REGISTER CONTENTS:

STA = 014046
B = 007760
T = 014046

PRN is set.

TIME:

(15 x 2) + (45 x 1) + 90 μS. = 165 μS.

OUTLINE OF LOGIC

The operands are searched for the least significant non-space, non-minus character.

The sign of the subtrahend is reversed.

If the original signs of the operands are unlike, the sign of the minuend is the sign of the difference. If the signs are alike, the sign is assumed positive and changed later if necessary.

From this point Decimal Subtract is exactly the same as Decimal Add.

NOTE: The first character addressed by the A Register must be a space symbol, minus sign or ISS. If it is any other character, an alarm stop occurs.

53 DECIMAL MULTIPLY DM

GENERAL DESCRIPTION

This instruction performs decimal multiplication in accordance with algebraic rules, producing a variable-length, non-zero-suppressed product. If a quantity is pre-stored in the product area, the product is added (absolute addition) to it, permitting round-off by any number and multiply-accumulate. However, the sign of the product will be assigned to the accumulated (absolute) result.

FORMAT

T register (preset):

Address of character that is to receive the sign of the product. (The product may not be stored in the HSM locations of the operands).

A address:

Address of the rightmost character (ISS, LSD, sign, or space to the right of the sign) of the multiplicand.

B address:

Address of the rightmost character (ISS, LSD, sign, or space to the right of the sign) of the multiplier.

DIRECTION OF OPERATION

Right to left.

TERMINAL CONDITION

The operation terminates when the most significant digit of the product has been stored. Neither space nor ISS is stored to the left of this digit automatically, and if one or other is required it must be placed by another instruction. The length of each operand is defined by the first space to the left of a non-space, non-minus character or by an ISS.

FINAL REGISTER CONTENTS

The A register store addresses the first character to the left of the ISS or space that terminates the multiplicand. The B register addresses the first character to the left of the ISS or space that terminates the multiplier. The T register addresses the first character to the left of the most significant digit of the product, i.e., the T register contains the address for placing the terminal character (space or ISS) of the product.

TIMING

Total time in milliseconds:

a. If $n_1 > 0$ and $n_2 > 0$

$$.015\ [10 + (12n_1 + 32)n_2] + .015n_3.$$

b. If $n_1 = 0$ and $n_2 > 0$

$$.015\ (n_2 + n_3 + 3)$$

c. If $n_2 = 0$ and $n_1 > 0$

$$.015\ (n_1 + n_3 + 3)$$

d. If $n_1 = 0$ and $n_2 = 0$ (an ISS alone or all spaces and an ISS)

$$.015\ (n_3 + 3)$$

where n_1 = number of digits in multiplicand

n_2 = number of digits in multiplier

n_3 = total number of spaces (including sign) and/or minuses to right of the LSD's of the operands.

EXAMPLES (DM)

1. Instructions: 72 001563 00 600000
   53 001307 00 002311

HSM before and after instruction is executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0013	X X	X • 6 -	X						0013

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0023		K K K -	3 - K K	K					0023

HSM before Instruction is executed:

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0015		E	E E E E	- - - -	- - - -	E E E E			0015

HSM after Instruction has been executed:

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0015		E	E E E E	- - - -	- 1 8 -	E E E E			0015

FINAL REGISTER CONTENTS:

STA = 001304
B = 002306
T = 001560

PRP is set

TIME:

.015 [10 + (12 x 1 + 32) 1] + (.015 x 2) mS. = 0.840 mS.

2. Instructions: 72 002423 00 600000
53 001513 00 002547

HSM before and after Instruction is executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0015	K K	• 1 4 0	- - - -	K K K					0015

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0025	X X	X • 0 4	X X X						0025

HSM before Instruction is executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0024				E E E	- - 5 -	E E E E			0024

HSM after Instruction has been executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0024				E E E	- 6 1 0	E E E E			0024

FINAL REGISTER CONTENTS:

STA = 001503
B = 002544
T = 002420

PRN is set

TIME:

.015 [10 + (12 x 2 + 32) 2] + (.015 x 5) mS. = 1.905 mS

OUTLINE OF LOGIC

The operands are searched for their LSD's, the contents of the A and B registers being respectively decreased by $(01)_8$ with each location searched. A sign is then placed in the product location and the contents of the T register are decreased by $(01)_8$. The sign of the product will be plus if the operand signs are alike, and minus if the signs are unlike. If an operand is not accompanied with a sign, it is assumed to be plus.

Multiplication is performed by a series of successive additions and shifts. The multiplicand is added to the contents of the product locations a number of times which is equal to the values of the digits of the multiplier.

The only characters not treated as numeric are minus sign, space symbol and ISS.

If the multiplier is all zeros, (and there is no pre-stored quantity in the product area), the product will be exactly like the multiplier (e.g., 634762 x 0000 = 0000).

If the multiplicand is all zeros, the number of zeros in the product will be equal to the number in the multiplicand plus the number of digits in the multiplier minus one, provided that the multiplier is zero suppressed.

If in either or both operands, an ISS is sensed before any other non-space, non-minus character, a plus sign is placed in the sign location of the product area, PRZ is set, and the operation ends.

If round-off or accumulate is desired, the prestored quantity must be properly positioned in the product area, since there is no search of the product before addition begins. If round-off or accumulate is not desired, the product area must be cleared to spaces.

54 DECIMAL DIVIDE D D

GENERAL DESCRIPTION

This instruction performs decimal division in accordance with algebraic rules and produces a a non-zero-suppressed, non-right-justified quotient. The non-zero-suppressed remainder is stored in the HSM locations originally occupied by the dividend. Each operand must carry a sign. The operands may be of any length. If the divisor contains more digits then the dividend, the quotient will be a zero. Unlike Decimal Multiply, the quotient is not added to a prestored quantity.

FORMAT

T register (preset)

Address for MSD of quotient.

A address:

Address of the sign (or space to the right of the sign) of the dividend (and remainder).

B address:

Address of the sign (or space to the right of the sign) of the divisor.

DIRECTION OF OPERATION

Shifting and storing of quotient digits is performed from left to right. Individual subtractions are performed from right to left, as in Decimal Subtract (52).

TERMINAL CONDITION

The operation terminates when the least significant digit of the quotient has been stored. The length of each operand is defined by the first space to the left of a non-space, non-minus character or by any ISS.

FINAL REGISTER CONTENTS

The A register store addresses the first character to the left of the space or ISS that terminates the remainder. The B register addresses the first character to the left of the space or ISS that terminates the divisor. The T register addresses the sign of the quotient. The P register store is unaltered.

TIMING

Total time, in milliseconds:

If $n_1 > n_2$

$$.015\ [26n_1 - 7n_2 + 15n_2\ (n_1 - n_2) + 41] + .015n_3$$

If $n_1 < n_2$

$$.015\ (3n_1 + n_2 + 12) + .015n_3$$

If $n_1 = 0$ (i.e., the dividend is missing) $.015\ (n_2 + 7) + .015n_3$

where n_1 = number of digits in the dividend

n_2 = number of digits in the divisor

n_3 = total number of spaces (including sign) and/or minuses to the right of the LSD's of the operands.

EXAMPLES (DD)

1. Instructions: 72 013011 00 600000
   54 012011 00 012060

HSM before Instruction is executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0120	X X	● 2 0 0	5 - X X	X					0120

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0120				K K - 3	- K K				0120

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0130		E E	● E E E	E E E E					0130

HSM after Instruction has been executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0120	X X	• 0 0 0	1 - X X	X					0120

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0120				K K - 3	- K K				0120

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0130		E E	• 0 6 6	8 - E E					0130

FINAL REGISTER CONTENTS:

STA = 012003
B = 012055
T = 013015
PRP is set

TIME:

.015 [26 x 4 - 7 x 1 + 15 (4 - 1) + 41] + .015 x 2 mS = 2.775 mS.

2. Instructions: 72 000445 00 600000
54 001611 00 002052

HSM before Instruction is executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0016	X X •	4 0 1 5	5 0 X X	X					0016

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0020		K K K -	3 9 - K	K					0020

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0004	E E	- - - -	- - - -	E E E					0004

HSM after Instructions have been executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0016	X X •	0 0 0 2	4 0 X X	X					0016

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0020		K K K -	3 9 - K	K					0020

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
0004	E E	- 1 0 2	9 0 - -	E E E					0004

FINAL REGISTER CONTENTS:

STA = 001602
B = 002046
T = 000451

PRN is set

TIME:

.015 [26 x 5 - 7 x 2 + 30 (5 - 2) + 41] + .015 x 2 mS = 3.735 mS.

3. Instructions: 72 027001 00 600000
54 113667 00 024315

HSM before Instructions are executed:

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
1136				X X	- 4 8 9	- - - -	X X X		1136

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0243		K K	- 1 2 2	5 - K K	K				0243

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0270	- E E E	E E E E	E E E						0270

HSM after Instructions have been executed:

	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
1136				X X	- 4 8 9	- - - -	X X X		1136

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0243		K K	- 1 2 2	5 - K K	K				0243

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0270	- 0 - E	E E E E	E E E						0270

FINAL REGISTER CONTENTS:

STA = 113657
B = 024307
T = 027002

PRZ is set

TIME:

.015 [3 x 3 + 4 + 12] + .015 x 5 mS. = .450 mS.

OUTLINE OF LOGIC

In general, division is performed by successively subtracting the divisor from the dividend with the MSD's of the operands aligned. Quotient digits are accumulated according to the number of valid subtractions performed. (An ISS or space is not automatically stored to the left of the MSD of the quotient.) When a negative remainder is sensed (invalid subtraction), the quotient digit is not increased. The subtractions then begin anew with the divisor shifted to align with the next dividend digit to the right, and a new quotient digit is accumulated.

Initially, a search is made of both operands, during which the following actions are taken:

1. A positive sign is stored in the quotient area if the signs of the operands are alike; a negative sign is stored if they are unlike. Each operand must carry a sign.

2. The divisor is validity checked; an alarm stop occurs if the divisor (a) is missing (i.e., contains only an ISS or spaces and an ISS) or (b) is not zero suppressed.

3. If the divisor contains more digits than the dividend or if the dividend is missing, PRZ is set and a zero with plus sign (space) is placed in the quotient.

Next, the process of subtraction and shifting takes place, with a non-zero-suppressed remainder replacing the dividend after each completed subtraction.

The operation ends when the subtraction process is completed, with the LSD's of the operands aligned.

After the operation the following conditions exist:

1. The non-zero-suppressed remainder occupies all of the HSM locations originally occupied by the dividend. A zero remainder will have a plus sign.

2. The PRI's are set according to the sign of the result. If the divisor is greater in magnitude than the dividend, a single zero and a plus sign will appear in the quotient and PRZ is set.

The number of quotient digits = (number of dividend digits minus number of divisor digits) + 1, provided this is greater than 0, otherwise = 1.

NOTE: Precision of the quotient may be increased by placing significant zeros in the dividend before the DD instruction is executed.

KDP 10 INSTRUCTIONS 61 - 66

61 CONDITIONAL TRANSFER OF CONTROL CTC

GENERAL DESCRIPTION

This instruction chooses one of three next instructions, according to the setting of the PRI's.

FORMAT

A address:

Address of the next instruction to be executed if PRP is set.

B address:

Address of the next instruction to be executed if PRN is set.

NOTE: If PRZ is set, the instruction in the location immediately following the CTC will be executed.

FINAL REGISTER CONTENTS

The A register store is unaltered. The B register addresses the next instruction to be executed if PRN was set. The P register addresses the next instruction to be executed, i.e., it contains the A address of the instruction modified by N_A if PRP was set, the B address of the instruction modified by N_B if PRN was set, and the address of the instruction in the location immediately following the CTC if PRZ was set. If PRP or PRN was set, the P register store addresses the instruction in the location immediately following the CTC, but if PRZ was set the P register store is unaltered. The T register is unaltered.

TIMING

If PRZ was set, only staticising time is required.

If PRP or PRN was set, the operation takes 15 μS in addition to staticising time.

OUTLINE OF LOGIC

The PRI's are examined. If PRZ is set, the next instruction in sequence is staticised. If PRZ is not set, the contents of the P register (which holds the address of the next instruction in sequence) are placed in standard HSM locations. If PRP is set, the contents of the A register are transferred to the P register. If PRN is set, the contents of the B register are transferred to the P register.

NOTE: STA is not performed by this instruction.

62 SENSE SIMULTANEOUS MODE SSM

GENERAL DESCRIPTION

This instruction selects one of four next instructions, depending upon whether the Simultaneous Mode is (1) unoccupied, (2) occupied by a 'read' instruction, (3) occupied by a 'write' instruction, or (4) occupied by a Paper Advance.

FORMAT

A address:

Address of the next instruction to be executed if a 'read' is in the Simultaneous Mode.

B address:

Address of the next instruction to be executed if a 'write' is in the Simultaneous Mode.

NOTE: If the Simultaneous Mode is unoccupied, the instruction in the location immediately following the SSM will be executed. If a Paper Advance is in the Simultaneous Mode, the next instruction to be obeyed is taken from (000200), it being the programmer's responsibility to previously place an instruction here.

FINAL REGISTER CONTENTS

The A register store is unaltered. The B register addresses the next instruction to be executed if a 'write' is in the Simultaneous Mode. The P register addresses the next instruction to be executed according to the rules under 'Format'. The P register store addresses the instruction in the location immediately following the SSM unless the Simultaneous Mode is unoccupied in which case the P register store is unaltered. The T register is unaltered.

TIMING

15 μS.

OUTLINE OF LOGIC

A test is made of the Simultaneous Mode indicator. If it is not set, the SSM ends and the Computer continues to the next instruction in sequence. If it is set, the contents of the P register are stored in standard HSM locations and the 'read', 'write' and 'Paper Advance' indicators are tested, in accordance with which the address of the next instruction to be executed is placed in the P register.

NOTE: STA is not performed in this instruction.

63 TAPE SENSE TS

GENERAL DESCRIPTION

This instruction tests the status of a given Tape Station, permitting programme direction to one of two sequences of instructions.

FORMAT

A address:

Address of the next Instruction to be executed if the condition or one of the conditions being tested is present.

B address:

B_1 - Tape Station number.

B_2 - the tests to be performed (111111) in B_2 will perform all six tests; (000001) will perform only the first test.

'1' Bit in	Tests	
2^0	Is the tape positioned on BTC?	01
2^1	Has ETW been sensed?	02
2^2	Is the tape now stationary or moving forward?	04
2^3	Is the tape now moving in a reverse direction?	10
2^4	Is the tape now in motion?	20
2^5	Is the Tape Station non-operable?	40

B_3 -Not used

If the B_1 character is $(77)_8$, the Monitor Printer will be tested if the SR register is occupied and the Paper Tape Reader will be tested if SR is unoccupied. In either case, the only valid test would be with respect to operability ('1' bit in 2^5).

See KDF 8 Manual for On Line Printer Tests.

FINAL REGISTER CONTENTS

The A register store is unaffected. The B register contains the B address of the instruction modified by N_B. If the condition or one of the conditions being tested is present, the P register contains the A address of the instruction modified by N_A and the P register store addresses the instruction in the location immediately following the TS. If the condition or conditions are not present, the P register addresses the instruction in the location immediately following the TS and the P register store is unaltered. The T register is unaltered.

TIMING

30 μS if no transfer.

45 μS if a transfer is executed.

OUTLINE OF LOGIC

The Tape Station number (B_1) is placed in either the SW or the SR register (whichever is unoccupied). The tests called for by the '1' bits in B_2 are performed. If any one of the conditions tested is present, the contents of the P register are transferred to standard HSM locations and the contents of the A register are then transferred to the P register, effecting transfer of control to the instruction specified by the A register. If a transfer is not called for, the next instruction in sequence will be executed. The TS instruction does not go through STA.

65 SENSE SIMULTANEOUS GATE SSG

GENERAL DESCRIPTION

This instruction chooses one of two sequences of instructions, depending upon whether or not the Simultaneous Gate is open.

FORMAT

A address:

Address of the next instruction to be executed if the Simultaneous Gate is open.

B address:

Address of the next instruction to be executed if the Simultaneous Gate is closed.

FINAL REGISTER CONTENTS

The A register store is unaltered. The B register addresses the next instruction to be executed if the Simultaneous Gate is closed. The P register contains the A address of the instruction modified by N_A if the Simultaneous Gate is open or the B address of the instruction modified by N_B if the Simultaneous Gate is closed. The P register store addresses the instruction in the location immediately following the SSG. The T register is unaltered.

TIMING

15 μS.

OUTLINE OF LOGIC

The contents of the P register are stored in standard HSM locations. The gate controlling entrance to the Simultaneous Mode is examined. If it is closed, the contents of the B register are transferred to the P register. If the gate is open, the contents of the A register are transferred to the P register.

NOTE: STA is not performed by this instruction.

66 TALLY TA

GENERAL DESCRIPTION

This instruction examines an octal counter and either, if it is not zero, reduces it by one and transfers control to an instruction at a specified location, or, if it is zero, continues to the next instruction in sequence. It can be used to loop through a sequence of instructions a number of times given by the octal counter.

FORMAT

A address:

Address of the next instruction to be performed if the counter being tested has not been exhausted.

B address:

Address of the tetrad containing, in its rightmost three locations, the counter to be tested.

FINAL REGISTER CONTENTS

The A register store is unaltered. The B register addresses the tetrad containing the counter. The T register contains the octal counter being tested minus one (if the quantity tested is $(000000)_8$ the T register will contain $(777777)_8$). If the counter has not been exhausted, the P register contains the A address of the instruction modified by N_A and the P register store addresses the instruction in the location immediately following the TA. If the counter has been exhausted, the P register addresses the instruction in the location immediately following the TA and the P register store is unaltered. (~~The T register is unaltered.~~ See see above)

TIMING

30 μS if the quantity tested is $(000000)_8$.
60 μS if the quantity is not $(000000)_8$.

EXAMPLE

At 014240, the instruction

66 015000 00 014167

At 014165-7, the counter

$(000010)_8$

At 015000, a sequence of instructions

— ——— — ———
— ——— — ———
— ——— — ———
— ——— — ———
— ——— — ———
71 014240 00 000000

Then if the instruction at 014240 is executed, the sequence of instruction at 015000 will be executed $(10)_8$, i.e., 8 times before the instruction in 014250 is executed.

OUTLINE OF LOGIC

The rightmost three characters of the tetrad specified by the contents of the B register are read out of the HSM and into the T register. Then $(000001)_8$ is subtracted from this quantity, and the result is tested. If the result is $(777777)_8$, the Tally ends and the next instruction in sequence is executed. If the result is not $(777777)_8$, it is placed in the HSM locations of the original quantity. The contents of the P register are then stored in standard HSM locations, and the contents of the A register are transferred to the P register. STA is not performed by this instruction.

KDP 10 INSTRUCTIONS 71 - 77

71 TRANSFER CONTROL TC

GENERAL DESCRIPTION

This instruction either causes an unconditional break in the sequence of instructions or takes action according to the settings of the Breakpoint Switches on the Computer Console.

FORMAT

A address:

Address of the next instruction to be executed (unless nullified by Breakpoint bits and switch settings).

B address:

B_1 - Breakpoint bits.

B_2B_3 - ignored.

FINAL REGISTER CONTENTS

All addressable registers and register stores are untouched except the P register store, which, unless there is a breakpoint match, contains the address of the instruction stored immediately after the Transfer Control instruction, and the P register itself which contains the A address of the instruction modified by N_A.

TIMING

15 μS.

OUTLINE OF LOGIC

The six bits of B_1 (2^0, 2^1, 2^2, 2^3, 2^4, 2^5) are matched against the 6 two-position Breakpoint Switches numbered 0, 1, 2, 3, 4, 5, on the Console. If any one of the control bits in B_1 is a '1', and its corresponding Breakpoint Switch is in the 'ignore' position, the transfer will not take place and the programme will progress to the next instruction in sequence. If the transfer is performed, the contents of the P register are placed in standard HSM locations and the contents of the A register are placed in the P register. In either case, the B address is not placed in the B register, the Breakpoint bits being examined in either the SW or the SR register. However, address modification, if called for, will be performed on the contents of the B register (left by a previous instruction).

NOTE: STA is not performed by this Instruction.

72 SET REGISTER SET

GENERAL DESCRIPTION

This instruction places the A_1, A_2 and A_3 characters of the instruction into a specified register or register store or sets the PRI's.

FORMAT

A address:

The three characters (not the address of the three characters) to be placed in the specified register or register store. If setting the PRI's:

$(000001)_8$ sets PRN

$(000002)_8$ sets PRZ

$(000004)_8$ sets PRP

B address:

B_1 = the register, register store, or PRI's to be set:

B_1 Character	Register Selected
$(10)_8$	PRI's
$(20)_8$	A Register Store
$(40)_8$	P Register
$(60)_8$	T Register

B_2B_3 - ignored.

FINAL REGISTER CONTENTS

The B register and P register store are unaltered. The A register store and T register are unaltered unless the T register or the A register store is to be set, in which case that register or store contains the A_1, A_2 and A_3 characters of the instruction.

TIMING

15 μS, unless the A register store is to be set, in which case time is 30 μS.

OUTLINE OF LOGIC

The B_1 character is examined in the SW or the SR register (the B address is not sent to the B register) and used to select the register whose contents are to be replaced. The contents of the A address are then placed into the appropriate register. If the B_1 character is anything other than one of the values listed above, the instruction will not be executed, but the timing will be the same as if it had been executed. No alarm stop will occur; the programme will continue to the next instruction in sequence. If the A register store is designated to be set, the instruction goes through STA; otherwise, it does not.

73 STORE REGISTER STR

GENERAL DESCRIPTION

This instruction places the contents of a selected register (or PRI setting) into the rightmost three locations of a specified tetrad.

FORMAT

A address:

Address of the tetrad that is to receive, in the $C_1C_2C_3$ positions, the contents of the register specified.

B address:

B_1 - the PRI's or the register whose contents are to be stored.

B_1 Character	Register Selected
$(10)_8$	PRI's [The stored $C_1C_2C_3$ are $(000001)_8$ if PRN set $(000002)_8$ if PRZ set $(000004)_8$ if PRP set]
$(30)_8$	B Register
$(40)_8$	P Register
$(50)_8$	S Register
$(60)_8$	T Register

B_2B_3 - ignored.

FINAL REGISTER CONTENTS

The A register store, B register, T register and P register stores are all unaltered by this instruction.

TIMING

15 μS.

EXAMPLE (STR)

Instruction 73 010320 00 100000

Assumption: PRP set as a result of a prior operation.

HSM before Instruction is executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0103					72 00 00 00	00 10 00 00			0103

HSM after Instruction has been executed:

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	
0103					72 00 00 04	00 10 00 00			0103

OUTLINE OF LOGIC

The B_1 character is examined in the SW or the SR register (the B address is not sent to the B register). The contents of the designated register are then stored in C_1, C_2 and C_3 of the tetrad specified by the A address; the original C_0 remains undisturbed.

If the B_1 character is not one of the values listed above, the instruction will not be executed, but the timing will be the same as if it had been executed. No alarm stop will occur; the programme will continue to the next instruction in sequence. (Note that the A register is not included in the list).

75 CONTROL SIMULTANEOUS GATE CSG

GENERAL DESCRIPTION

This instruction is used to open or close the gate which controls entrance into the Simultaneous Mode, making it possible to either prevent or permit simultaneous operations.

FORMAT

A address - ignored

B address:

B_1 - must be even (2^0 bit = '0') if gate is to be opened, and odd (2^0 bit = '1') if gate is to be closed.

B_2B_3 - ignored.

FINAL REGISTER CONTENTS

All addressable registers and register stores are unaltered except the B register, which contains the B address of the instruction modified by N_B.

TIMING

15μS.

76 STOP ST

GENERAL DESCRIPTION

This instruction inhibits the staticising of any further instructions, halting the Computer after completion of any instruction in the Simultaneous Mode.

FORMAT

A address - ignored. B address - ignored.

FINAL REGISTER CONTENTS

The A register store contains the A address of the instruction modified by N_A. The B register contains the B address of the instruction modified by N_B. The T register and P register stores are unaltered.

TIMING

Staticising and STA time only.

OUTLINE OF LOGIC

If the Simultaneous Mode is unoccupied when the stop instruction is staticised, the Computer stops immediately. If it is occupied, the Simultaneous instruction is completed before the Computer stops. However, if a 'read' parity error is detected in the Simultaneous Mode after a stop instruction is staticised, the Computer will stop, without attempting to perform Rollback.

The stop instruction goes through STA.

77 RETURN AFTER INTERRUPT RAI

GENERAL DESCRIPTION

This instruction is used to re-enter a programme after an unscheduled interruption, such as for Rollback or for a higher priority programme.

The RAI was designed not only to transfer control, but to permit the A and B registers to be properly set in the process. Thus, when the main programme is re-entered, some of the pertinent conditions prevailing at the time of interruption can be re-established.

FORMAT

A address:

The 3-character pattern to appear in the A register store when the programme is re-entered.

B address:

The 3-character pattern to appear in the B register when the programme is re-entered.

FINAL REGISTER CONTENTS

The A register store contains the A address of the instruction modified by N_A. The B register contains the B address of the instruction modified by N_B. The P register contains the 3-character pattern from HSM locations $(000001)_8$ to $(000003)_8$. (This will be the address of the next instruction to be executed). The T register and P register stores are unaltered by this instruction.

TIMING

15 μS.

OUTLINE OF LOGIC

The RAI instruction automatically transfers into the P register the contents of standard HSM locations $(000001)_8$ - $(000003)_8$, effecting a transfer of control. Storing of the desired P register setting into these locations must be accomplished by a previous instruction. Although RAI ignores the A and B addresses, whatever is inserted into these addresses will be transferred into the A and B registers when the instruction is staticised, with address modification if indicated, for use on re-entry into the main programme. This instruction goes through STA.

APPENDICES 1—6

APPENDIX 1: ROLLBACK

Rollback is a prestored subroutine which is automatically entered when a parity error on magnetic tape is detected during Normal Or Simultaneous (1) Linear or Block Read Forward, (2) Linear or Block Read Reverse, or (3) Unwind or Rewind SM's, EM's, or Gaps.

After detection of the parity error, the tape is rewound or unwound one block or message (whichever pertains to the instruction involved). The interrupted instruction. which was automatically stored in standard HSM locations $(000020)_8$ - $(000027)_8$, is then re-executed.

Entrance into the routine may be suppressed by manual setting of a Console switch (Rollback) to the Stop position. The Computer, instead of initiating Rollback, will then stop on detection of a parity error, with the erroneous character in the staticiser and register contents preserved.

If the Rollback switch is set to the ON position the tape will continue to move until the read-write head is at the end of the message or block being read. The erroneous character (in the staticiser) and the register contents are automatically preserved for Rollback.

If another parity or other type of error occurs at any time between detection of the first tape parity error and completion of Rollback, the Computer will stop, with notification of the cause of stoppage.

If, when an error is detected in a magnetic tape read, another instruction is being executed (in either mode), it will be completed before the Computer takes action (Stop or Rollback) with respect to the instruction in error.

APPENDIX 2: SUMMARY OF INSTRUCTIONS

Op. Code	Instruction	A Address	B Address	T Register	STA	STP†	Sets PRI's	P.S.
00								
01	Program Error Stop	Ignored	Ignored	Not Preset	Yes			
02	Print	Leftmost tetrad of sector to be printed; A_3 must be $(00)_8$; A_2 must be even	Ignored	Not preset—used as internal counter	Yes			
03	Paper Advance	A_2A_3 = no. of lines to advance A_1 ignored *Loop-controlled:* A_2A_3 = (0000) A_1: 2^0 bit = 1, Vertical Tab. 2^0 bit = 0, Page Change	Ignored	Not preset	Yes			Yes
04	Linear Read Reverse	Destination tetrad of EM (EM will always be placed in C_3)	B_1 = Tape Station* B_2B_3 ignored	Not preset	Yes			Yes
05	Block Read Reverse	Destination tetrad of 1st char. transferred from tape (1st char. will always be placed in C_3)	B_1 = Tape Station* B_2B_3 ignored	Not preset	Yes			Yes
06	Unwind *n* Symbols	A_1 = symbol A_2A_3 = no. of symbols (octal count)	B_1 = Tape Station B_2B_3 ignored	Not preset	Yes			Yes
07								
10	16.7 KC Linear Write	HSM loc. of leftmost char. to be written out	B_1 = Tape Station* B_2: 2^0 = 0, normal 3.5 ms up to speed delay 2^0 = 1, 4.5 ms up to speed delay B_3 ignored	Not Preset	Yes			Yes
11	Single Sector Write	HSM loc. of leftmost char. to be written out	HSM loc. of rightmost char.	Preset T_1 = Tape Station* T_2: 2^0 = 0, normal 3.5 ms up to speed delay 2^0 = 1, 4.5 ms up to speed delay T_3 ignored	Yes			

* $(^{77})_8$ in B_1 will select the Monitor Printer, or Paper Tape Punch via the Monitor Printer, in write instructions, and the Paper Tape Reader in read instructions.

† The contents of the P Register are automatically stored (in standard HSM locations 000241 – 000243) only when control is actually transferred.

Remarks	Post-Operational Register Settings#		
	A Register	B Register	T Register
Causes PES alarm to light			
Occupies both modes The character $(00)_8$ must not appear in print-out sector	$(A)_f = (A)_i$	$(B)_f = (A)_i + (000170)_8 = (120)_{10}$ = No. of characters	$(T)_f = (003124)_8 = (30)_{10} \times (54)_{10}$ = 54 cyc/tetrad × 30 tetrads
A_1 ignored unless $A_2A_3 \neq (0000)_8$ $A = (000000)_8$, PA ignored	1. Normal Mode $(A_1)_f = (A_1)_i$ $(A_2A_3)_f = (0000)_8$ if $(A_2A_3)_i > (0000)_8$ or $(A_2A_3)_f = (0000)_8$ minus no. of lines shifted if $(A_2A_3)_i = (0000)_8$ 2. Simultaneous Mode $(S)_f$ settings analogous to $(A)_f$ settings above	1. $(B)_f = (B)_i$	
	1. Normal Mode $(A)_f$ = HSM loc. of SM, ED or EF 2. Simultaneous Mode $(S)_f$ = HSM loc. of SM, ED or EF	1. $(B)_f = (B)_i$	
	1. Normal Mode $(A)_f$ = loc. of last char. read in 2. Simultaneous Mode $(S)_f$ = HSM loc. of last char. read in	1. $(B)_f = (B)_i$	
A read instr. cannot be simult. with UNS Possible symbols: EM [$(75)_8$], ED [$(73)_8$], EF [$(72)_8$], Gap [$(00)_8$]	1. Normal Mode $(A_1)_f = (A_1)_i$ $(A_2A_3)_f = (0000)_8$ unless PET is reached; then (A_2A_3) = no. of symbols left to be counted 2. Simultaneous Mode $(S_1)_f = (A_1)_i$ $(S_2S_3)_f = (0000)_8$ unless PET is reached; then S_2S_3 = no. of symbols left to be counted	1. $(B)_f = (B)_i$	
Characters are written out at 16.7 KC	1. Normal Mode $(A)_f$ = HSM loc. of EM, EF or ED 2. Simultaneous Mode $(S)_f$ = HSM loc. of EM, EF or ED	$(B)_f = (B)_i$	
	$(A)_f = (B)_i$	$(B)_f = (B)_i$	$(T)_f = (T)_i$

‡ If the B_1 character is $(77)_8$, the Monitor Printer will be tested if the SR Register is occupied and the Paper Tape Reader will be tested if the SR Register is unoccupied. In either case, the only valid test would be with respect to operability ("1" bit in 2').

Register settings will reflect automatic address modification.

Op. Code	Instruction	A Address	B Address	T Register	STA	STP†	Sets PRI's	P.S.
12	Linear Write	HSM loc. of leftmost char. to be written out	B_1 = Tape Station* B_2: 2^0 = 0, normal 3.5 ms up to speed delay 2^0 = 1, 4.5 ms up to speed delay B_3 ignored	Not preset	Yes			Yes
13	Multiple Sector Write	Leftmost tetrad of stored list of addresses	B_1 = Tape Station* B_2: 2^0 = 0, normal 3.5 ms up to speed delay 2^0 = 1, 4.5 ms up to speed delay B_3 ignored	Not preset—used as internal counter	Yes			
14	Linear Read Forward	Destination tetrad of SM (ED or EF) (SM, ED or EF will be placed in C_0)	B_1 = Tape Station* B_2B_3 ignored	Not preset	Yes			Yes
15	Block Read Forward	Destination tetrad of 1st char. read (this char. will be placed in C_0)	B_1 = Tape Station* B_2B_3 ignored	Not preset	Yes			Yes
16	Rewind *n* Symbols	A_1 = Symbol to be counted A_2A_3 = No. of symbols to be counted	B_1 = Tape Station B_2B_3 ignored	Not preset	Yes			Yes
17	Rewind to BTC	Ignored	B_1 = Tape Station B_2B_3 ignored	Not preset	Yes			Yes
20								
21	Item Transfer	HSM loc. of rightmost char. to be transferred	Destination loc. of rightmost char.	Not preset	Yes		Yes	
22	One-Character Transfer	HSM loc. of char. to be transferred	Destination loc. of char.	Not preset	Yes			
23								
24	Sector Transfer by Character	HSM loc. of leftmost char. of sector to be transferred	HSM loc. of rightmost char. of sector to be transferred	Preset—destination loc. of rightmost char.	Yes			
25	Three-Character Transfer	Address of the tetrad containing, in C_1, C_2 and C_3, the characters to be transferred	Address of the tetrad to receive, in C_1, C_2 and C_3, the three characters.	Not preset	Yes			
26	Sector Transfer by Tetrad	Leftmost tetrad of sector to be transferred	Rightmost tetrad of sector to be transferred	Preset—destination (tetrad address) of rightmost tetrad	Yes			

Remarks	Post-Operational Register Settings:#		
	A Register	B Register	T Register
1st char. is checked for ED or EF (not for SM) LW ends when EM (ED or EF) is written	1. Normal Mode $(A)_f$ = HSM loc. of EM, EF or ED 2. Simultaneous Mode $(S)_f$ = HSM loc. of EM, EF or ED	1. $(B)_f = (B)_i$	
MSW ends when XXX in stored list is $(000000)_s$	$(A)_f$ = Address of last tetrad in stored list of addresses	$(B)_f = (B)_i$	$(T)_f = (000000)_s$
	1. Normal Mode $(A)_f$ = HSM loc. of EM 2. Simultaneous Mode $(S)_f$ = HSM loc. of EM	1. $(B)_f = (B)_i$	
	1. Normal Mode $(A)_f$ = HSM loc. of last char. read in 2. Simultaneous Mode $(S)_f$ = HSM loc. of last char. read in	1. $(B)_f = (B)_i$	
A read instr. cannot be simult. with RNS Possible symbols: SM $[(76)_s]$, ED $[(73)_s]$, EF $[(72)_s]$, Gap $[(00)_s]$	1. Normal Mode $(A_1)_f = (A_1)$ $(A_2A_3)_f = (0000)_s$ unless the BTC was reached; then $(A_2A_3)_f$ = no. of symbols left to count 2. Simultaneous Mode $(S_1)_f = (S_1)_i$ $(S_2S_3)_f = (0000)_s$ unless the BTC was reached; then $(S_2S_3)_f$ = no. of symbols left to count	1. $(B)_f = (B)_i$	
Rewind is completely free of Computer after start	$(A)_f = (A)_i$	$(B)_f = (B)_i$	
IT ends on transfer of ISS If item contains one or more non-space characters (to the right of the ISS), PRP is set; if only space symbols, PRN is set	Original HSM loc. of ISS minus $(01)_s$	Destination loc. of ISS minus $(01)_s$	
	$(A)_f = (A)_i$	$(B)_f = (B)_i$	
	$(A)_f = (A)_i$	$(B)_f = (A)_i - (01)_s$	$(T)_f = (T)_i - n$ n = no. of char. transferred
Characters are transferred in parallel	$(A)_f = (A)_i$	$(B)_f = (B)_i$	
	$(A)_f = (A)_i$	$(B)_f = (B)_i - 4n$ n = no. of tetrads transferred	$(T)_f = (T)_i - 4n$ n = no. of tetrads transferred

Op. Code	Instruction	A Address	B Address	T Register	STA	STP†	Sets PRI's	P.S.
27	Random Distribute	HSM loc. of SM or ISS of leftmost item in sector to be distributed	Address of the tetrad in the stored list which contains, (in C_1, C_2 and C_3), the destination loc. of the SM or the ISS of the leftmost item to be distributed	Not preset—used for internal addressing	Yes		Yes	
30								
31	Locate #th Symbol in Sector	Leftmost HSM loc. of sector to be searched	Rightmost HSM loc. of sector to be searched	Preset T_1 = Symbol to be counted T_2T_3 = No. of symbols to be counted	Yes		Yes	
32	Zero Suppress	Leftmost HSM loc. of sector in which zeros are to be suppressed	Rightmost HSM loc. of sector in which zeros are to be suppressed	Not preset	Yes			
33	Justify Right	Rightmost HSM loc. of item to be justified	Destination loc. of sign or LSC of item	Not preset—used as internal counter	Yes		Yes	
34	Sector Clear by Character	Leftmost HSM loc. of sector to be cleared	Rightmost HSM loc. of sector to be cleared	Not preset	Yes			
35	Sector Compress—Retain Redundant ISS's	Leftmost HSM loc. of sector to be compressed	Rightmost HSM loc. of sector to be compressed	Preset—destination loc. of rightmost retained char.	Yes		Yes	
36	Sector Clear by Tetrad	Address of leftmost tetrad to be cleared	Address of rightmost tetrad to be cleared	Not preset	Yes			
37	Sector Compress—Delete Redundant ISS's	Leftmost HSM loc. of sector to be compressed	Rightmost HSM loc. of sector to be compressed	Preset—destination loc. of rightmost retained char.	Yes		Yes	
40								

Remarks	Post-Operational Register Settings#		
	A Register	B Register	T Register
1. If terminal address in list has been sensed and the EM has not been sensed, PRZ is set 2. If terminal address in list has been sensed and EM has been sensed, PRP is set 3. If terminal address in list has not been sensed and EM has been sensed, PRN is set EM is not distributed; ISS placed in destination loc. instead	HSM loc. of last ISS, SM or first EM sensed in the original area	$(B)_f = (B)_i + 4n$ n = no. of addresses in stored list (including terminal and "throw-away" addresses)	$(T)_f = (000000)_8$
PRZ set if $(T_2T_3)_f = (0000)_8$ and A = B PRP set if $(T_2T_3)_f = (0000)_8$ and A = B PRN set if $(T_2T_3)_f = (0000)_8$ and A = B Instruction not performed, but PRI's set: if $(T_2T_3)_i = (0000)_8$ (PRZ) if $(A)_i = (B)_i$ (PRN)	1. If LNS concluded with $(T_2T_3)_f = (0000)_8$: $(A)_f = (A)_i + n - (02)_8$ n = number of locations searched 2. If LNS concluded with A = B $(A)_f = (B)_i - (01)_8$	$(B)_f = (B)_i$	1. $(T_2T_3)_f = (0000)_8$ 2. $(T_2T_3)_f = (T_2T_3)_i$ minus no. of symbols counted
	1. $(A)_f = (A)_i + n - (02)_8$ n = number of locations searched 2. If ZS terminated by A-B equality $(A)_f = (B)_i$	$(B)_f = (B)_i$	1. $(T)_f = (A)_f + (01)_8$ 2. If ZS terminated by A-B equality (a) Only zeros encountered $(T)_f = (B)_i + (01)_8$ (b) Only spaces encountered $(T)_f = (B)_i$
If a non-space char. is encountered prior to the ISS, PRP is set (minus sign = non-space char.) If only an ISS or spaces and ISS, PRN is set	Original loc. of ISS minus (01).	Destination loc. of ISS minus $(01)_8$	$(000000)_8$ if 1st char. in original loc. is space or minus $(777777)_8$ if 1st char. in original loc. is not space or minus
	$(A)_f = (A)_i$	$(B)_f = (B)_i - n = (A)_i - (01)_8$ n = no. of char. cleared	
If the sector contains any non-space, non-EM, non-ISS char., PRP is set If the sector contains only spaces symbols, EM, and/or ISS, PRN is set	$(A)_f = (A)_i$	$(B)_f = (A)_i - (01)_8$	$(T)_f = (T)_i - n$ n = no. of char. transferred
	$(A)_f = (A)_i$	$(B)_f = (B)_i - 4n$ n = no. of tetrads cleared	
If sector contains any non-space, non-ISS char., PRP is set If sector contains only space symbols and/or ISS, PRN is set Redundant ISS's not deleted if $(B)_i$ = loc. of EM or loc. to right of EM	$(A)_f = (A)_i$	$(B)_f = (A)_i - (01)_8$	$(T)_f = (T)_i - n$ n = no. of char. actually transferred

Op. Code	Instruction	A Address	B Address	T Register	STA	STP†	Sets PRI's	P.S.
41	Binary Add	HSM loc. of leftmost char. of augend (and sum)	HSM loc. of rightmost char. of augend (and sum)	Preset—HSM loc. of rightmost char. of addend	Yes			
42	Binary Subtract	HSM loc. of leftmost char. of minuend (and difference)	HSM loc. of rightmost char. of minuend (and difference)	HSM loc. of rightmost char. of subtrahend	Yes		Yes	
43	Sector Compare	HSM loc. of leftmost char. of minuend	HSM loc. of rightmost char. of minuend	HSM loc. of rightmost char. of subtrahend	Yes		Yes	
44	Three-Character Add	HSM loc. of rightmost char. of augend (and sum)	HSM loc. of rightmost char. of addend	Not preset	Yes			
45	Three-Character Subtract	HSM loc. of rightmost char. of minuend (and difference)	HSM loc. of rightmost char. of subtrahend	Not preset	Yes		Yes	
46	Logical "or"	HSM loc. of leftmost char. of operand to be modified (and result)	HSM loc. of rightmost char. of operand to be modified (and result)	HSM loc. of rightmost char. of modifier	Yes			
47	Logical "and"	HSM loc. of leftmost char. of operand to be modified (and result)	HSM loc. of rightmost char. of operand to be modified (and result)	HSM loc. of rightmost char. of modifier (mask)	Yes			
50								
51	Decimal Add	HSM loc. of rightmost char. of augend (and sum)	HSM loc. of rightmost char. of addend	Not preset—used internally to address the sum	Yes		Yes	
52	Decimal Subtract	HSM loc. of rightmost char. of minuend (and difference)	HSM loc. of rightmost char. of subtrahend	Not preset—used internally to address the difference	Yes		Yes	
53	Decimal Multiply	HSM loc. of rightmost char. of multiplicand	HSM loc. of rightmost char. of multiplier	Preset—destination loc. of sign of product	Yes		Yes	
54	Decimal Divide	HSM loc. of rightmost char. of dividend (and remainder); must be sign or space to right of sign	HSM loc. of rightmost char. of divisor; must be sign or space to right of sign	Preset—HSM loc. of leftmost digit of quotient	Yes		Yes	
55								
56								
57								
60								
61	Conditional Transfer of Control	Address of next instr. if PRP is set	Address of next instr. if PRN is set	Not preset		Yes		
62	Sense Simultaneous Mode	Address of next instr. if a "read" is in Simult. Mode.	Address of next instr. if a "write" is in Simult. Mode			Yes		

Remarks	Post-Operational Register Settings#		
	A Register	B Register	T Register
All characters treated as numeric; no carry from most signif. bit position of sum	MSC of sum	$(B)_f = (A)_f - (01)_s$	MSC of addend minus $(01)_s$
No carry from most signif. bit position of difference; sign of difference not stored in HSM; PRZ set if diff. = octal zero; PRP set if minuend > subtrahend; PRN set if minuend < subtrahend	HSM loc. of MSC of difference	$(B)_f = (A)_f - (01)_s$	HSM loc. of MSC of subtrahend minus $(01)_s$
PRI settings same as for Binary Subtract Difference not stored in HSM	$(A)_f = (A)_i$	$(B)_f = (A)_i - (01)_s$	HSM loc. of MSC of subtrahend minus $(01)_s$
TCA ends on a C_i char. See also Binary Add (Remarks)	HSM loc. of MSC of sum minus $(01)_s$	HSM loc. of MSC of addend minus $(01)_s$	
TCS ends on a C_i char. See also Binary Subtract (Remarks)	HSM loc. of MSC of difference minus $(01)_s$	HSM loc. of MSC of subtrahend	
Control symbols treated as data	$(A)_f = (A)_i$	$(B)_f = (A)_i - (01)_s$	HSM loc. of MSC of modifier minus $(01)_s$
Control symbols treated as data	$(A)_f = (A)_i$	$(B)_f = (A)_i - (01)_s$	HSM loc. of MSC of modifier minus $(01)_s$
A Register must initially address a space, minus or ISS	HSM loc. that held the MSD of augend minus $(02)_s$	HSM loc. of MSD of addend minus $(02)_s$	HSM loc. of ISS of sum
A Register must initially address a space, minus or ISS	HSM loc. that held the MSD of minuend minus $(02)_s$	HSM loc. of MSD of subtrahend minus $(02)_s$	HSM loc. of ISS of difference
PRI settings are related to the product (not the accumulated result)	HSM loc. of MSD of multiplicand minus $(02)_s$	HSM loc. of MSD of multiplier minus $(02)_s$	HSM loc. of MSD of product minus $(01)_s$
No. of quotient digits = (No. of divisor digits) + 1 PRZ set if divisor > dividend PRP set if signs of operands alike PRN set if signs of operands unlike	HSM loc. of MSD of remainder minus $(02)_s$	HSM loc. of MSD of divisor minus $(02)_s$	HSM loc. of sign of quotient
Refers to, but does not set, PRI's; if PRZ set, takes next instr. in sequence, and no STP	$(A)_f = (A)_i$	$(B)_f = (B)_i$	
Control transferred to 000200 if PA in Sim. Mode; if Sim. Mode unoccupied, next instr. in sequence and no STP	$(A)_f = (A)_i$	$(B)_f = (B)_i$	

Op. Code	Instruction	A Address	B Address	T Register	STA	STP†	Sets PRI's	P.S.
63	Tape Sense	Address of next instr. if any of the tested conditions are present	Preset B_1 = Tape Station‡ B_2 = Tests to be performed B_3 ignored	Not preset		Yes		
64								
65	Sense Simultaneous Gate	Address of next instr. if Simult. Gate is open	Address of next instr. if Simult. Gate is closed	Not preset		Yes		
66	Tally	Address of next instr. if Tally quantity is not (000000).	Address of tetrad containing (in C_1, C_2 and C_3) the Tally quantity	Not preset—used as internal counter		Yes		
67								
70								
71	Transfer Control	Address of next instr. to be executed	B_1 = Breakpoint bits B_2B_3 ignored	Not preset		Yes		
72	Set Register	Actual value to be placed in the register specified by B_1	B_1 = Register to be set B_2B_3 ignored	Not preset				
73	Store Register	Address of the tetrad to receive (in C_1, C_2 and C_3) contents of register specified by B_1	B_1 = Register whose contents are to be stored B_2B_3 ignored	Not preset				
74								
75	Control Simultaneous Gate	Ignored	B_1 = Even = Open Gate Odd = Close Gate B_2B_3 ignored	Not preset				
76	Stop	Ignored	Ignored	Not preset	Yes			
77	Return After Interrupt	Actual address to appear in A Reg. when program is re-entered	Actual address to appear in B Reg. when program is re-entered	Not preset	Yes			

Remarks	Post-Operational Register Settings#		
	A Register	B Register	T Register
See instr. write-up for conditions tested by B. bits Next instr. in sequence and no STP if none of the conditions tested is present	$(A)_f = (A)_i$	$(B)_f \simeq (B)_i$	
	$(A)_f = (A)_i$	$(B)_f \simeq (B)_i$	
STP not performed if quantity tested is (000000).	$(A)_f = (A)_i$	$(B)_f \simeq (B)_i$	$(T)_f$ = quantity tested minus $(000001)_8$; if $(T)_i = (000000)_8$, $(T)_f = (777777)_8$
TC alterable by Breakpoint switch settings	$(A)_f = (A)_i$	TC does not use B Register $(B)_f \simeq (B)_i$ = contents left by previous instr.	
Can set A, P, T and PRI's	$(A)_f = (A)_i$	SET does not use B Register $(B)_f \simeq (B)_i$ = contents left by previous instr.	
Can store B, P, S, T and PRI's	$(A)_f = (A)_i$	STR does not use B Register $(B)_f \simeq (B)_i$ = contents left by previous instr.	
	$(A)_f = (A)_i$	$(B)_f \simeq (B)_i$	
	$(A)_f = (A)_i$	$(B)_f \simeq (B)_i$	
	$(A)_f = (A)_i$	$(B)_f \simeq (B)_i$	

APPENDIX 3: INSTRUCTION TIMING

Op. Code	Instruction	Timing in milliseconds	STA	Notes
01	Programme Error Stop	Statioising time only	Yes	-
02	Print	67	Yes	One 120-character line
03	Paper Advance	a. 30 b. 30+(n-1)20	Yes	a. Single line shifting b. Multiline shifting n=number of lines advanced
04	Linear Read Reverse	3.575+.03n	Yes	n=number of characters transferred
05	Block Read Reverse	3.575+.03(n+6)	Yes	n=number of characters transferred
06	Unwind n Symbols	3.575+.03n+4m	Yes	n=total number of characters read, including symbols being counted m=number of gaps encountered
10	Transcribing Card Write	3.575+.03n	Yes	n=number of characters transferred
11	Single Sector Write	3.575+.03n	Yes	n=number of characters transferred
12	Linear Write	3.575+.03n	Yes	n=number of characters transferred
13	Multiple Sector Write	3.575+.03n+.03(m-1)	Yes	n=number of characters transferred m=total no. of addresses in stored list
14	Linear Read Forward	3.575+.03n	Yes	n=number of characters transferred
15	Block Read Forward	3.575+.03(n+6)	Yes	n=number of characters transferred
16	Rewind n Symbols	3.575+.03n+4m	Yes	n=total number of characters read, including symbols being counted m=number of gaps encountered
17	Rewind to BTC	a. .3 b. .105	Yes	a. If the BTC is not positioned at the read-write head when the RWD instr. is given. b. If the BTC is already positioned at the read-write head when the RWD instruction is given.
21	Item Transfer	.03n	Yes	n=number of characters transferred
22	One Character Transfer	.03	Yes	
24	Sector Transfer by Character	.03n	Yes	n=number of characters transferred
25	Three Character Transfer	.03	Yes	
26	Sector Transfer by Tetrad	.03n	Yes	n=number of tetrads transferred
27	Random Distribute	$.033n_1+.018n_2+.045n_3$	Yes	n_1=total number of characters transferred n_2=total number of characters whose distribution address is $(777777)_8$ n_3=number of distribution addresses left when EM is found (the address of the EM must be included in n_3)
31	Locate n^{th} Symbol in Sector	.015m+.03n+.045 .015 if count zero .06 if sector is 1 character in length	Yes	n=number of occurrances counted m=total number of locations searched
32	Zero Suppress	a. .015m+.03n+.015 b. .015m+.03n c. .03	Yes	a. In the usual case b. If ZS terminated by A-B equality c. If no zeros or spaces found m=number of spaces preceding first non-space character n=number of zeros suppressed
33	Justify Right	.03n+.03m	Yes	n=number of space and/or minus characters to the right of the rightmost non-minus non-space character m=number of non-space non-minus characters (including ISS) transferred
34	Sector Clear by Character	.015n	Yes	n=total number of locations cleared
35	Sector Compress-Retain Redundant ISS's	.015n+.015m	Yes	n=total number of characters actually transferred m=total number of characters in original sector
36	Sector Clear by Tetrad	.015n	Yes	n=total number of tetrads cleared
37	Sector Compress-Delete Redundant ISS's	.015n+.015m	Yes	n=total number of characters actually transferred m=total number of characters in original sector

Op. Code	Instruction	Timing in milliseconds	STA	Notes
41	Binary Add	$.045n$	Yes	n=number of characters in augend
42	Binary Subtract	$.045n$	Yes	n=number of characters in minuend
43	Sector Compare	$.045n$	Yes	n=number of characters in minuend
44	Three Character Add	$.045n$	Yes	n=1,2,3 or 4
45	Three Character Subtract	$.045n$	Yes	n=1,2,3 or 4
46	Logical "or"	$.045n$	Yes	n=number of characters in operand to be modified
47	Logical "and"	$.045n$	Yes	n=number of characters in operand to be modified
51	Decimal Add	$.015n_1+.045n_2+.03n_3+.09$	Yes	n_1=total number of space and/or minus characters found to the right of both operands n_2=number of digits in the shorter operand n_3=difference in number of digits of the two operands If result negative and sum is re-complemented, add $.03(n+1)+.015$ where n is the number of digits in the result
52	Decimal Subtract	Same as for Decimal Add	Yes	
53	Decimal Multiply	a. $n_1>0$ and $n_2>0$ $.015[10+(12n_1+32)n_2]+.015n_3$ b. $n_1=0$ and $n_2>0$ $.015(n_2+3)$ c. $n_2=0$ and $n_1>0$ $.015(n_1+n_3+3)$ d. $n_1=n_2=0$ $.015(n_3+3)$	Yes	n_1=number of digits in multiplicand n_2=number of digits in multiplier n_3=total number of spaces(including sign) and/or minuses to right of LSD's of operands
54	Decimal Divide	a. $n_1 \geq n_2$ $.015[26n_1-7n_2+15n_2(n_1-n_2)+41]+.015n_3$ b. $n_1<n_2$ $.015(3n_1+n_2+12)+.015n_3$ c. $n_1=0$ (dividend missing, i.e. an ISS alone or all spaces and an ISS) $.015(n_2+7)+.015n_3$	Yes	n_1= number of digits in dividend n_2= number of digits in divisor n_3= total number of spaces(including sign) and/or minuses to the right of LSD's of operands
61	Conditional Transfer of Control	a. Staticising time only b. .015	No	a. If zero path taken b. If plus or minus path taken
62	Sense Simultaneous Mode	.015	No	
63	Tape Sense	a. .03 b. .045	No	a. If no transfer of control b. If transfer executed
65	Sense Simultaneous Gate	.015	No	
66	Tally	a. .03 b. .06	No	a. If quantity tested is $(000000)_8$ b. If quantity greater than $(000000)_8$
71	Transfer Control	.015	No	
72	Set Register	.015	-	STA only if A register set
73	Store Register	.015	No	
75	Control Simultaneous Gate	.015	No	
76	Stop	Staticising time only	Yes	
77	Return After Interrupt	.015	Yes	

The timing formulae do not include staticising, automatic address modification, or STA time. To obtain the overall timing, the following items should be added where appropriate:-

a. Staticising time of .03 milliseconds (constant for each instruction)
b. Automatic address modification time of .09 milliseconds if either the A or the B address is to be modified; or .18 milliseconds if both addresses are modified.
c. STA time of .015 milliseconds.

The read and write instruction timing is for magnetic tape only. The time for the Monitor Printer is 10 characters/second; for the Paper Tape Reader, 400 characters/second. Start time for Monitor Print and Paper Tape Read is negligible and can be ignored.

The timing formula given for Random Distribute (27) is a weighted average.

APPENDIX 4: STANDARD HSM LOCATIONS

HSM LOCATIONS	USE
000001 - 000003	Return After Interrupt. The RAI instruction effects a transfer of control to the instruction address stored in these locations.
000004 - 000007 000010 - 000013 000014 - 000017	Utilised by arithmetic instructions, for temporary storage of addresses, in lieu of special registers. (The contents of these locations are not useful to the programmer).
000020 - 000027	Storage locations for read (including unwind and rewind) instructions after staticising and address modification.
000030 - 000037	Storage locations for write instructions after staticising and address modification. If a read or write instruction contains an N character other than $(00)_8$, the instruction will be stored with the A and/or B address modified and the N character changed to $(00)_8$.
000040 - 000047	Rollback entrance - Normal. Control transferred to 000040 if the instruction in which the error occurred was in the Normal Mode.
000050 - 000057	Rollback entrance - Simultaneous. Control transferred to 000050 if the instruction in which the error occurred was in the Simultaneous Mode.
000111 - 000113 (A.M. 1) 000131 - 000133 (A.M. 3) 000151 - 000153 (A.M. 5) 000171 - 000173 (A.M. 7)	Static Address Modifiers.
000200	Control transferred to this address if a Paper Advance is sensed (Sense Simultaneous Mode instruction) in the Simultaneous Mode.
000221 - 000223	STA and A.M.2
000241 - 000243	STP

OCTAL DIGIT *	LOCATION OF MODIFIER
0	No Modifier
1	HSM locations 000111 - 000113
2	HSM locations 000221 - 000223 (STA)
3	HSM locations 000131 - 000133
4	P Register
5	HSM locations 000151 - 000153
6	T Register
7	HSM locations 000171 - 000173

* Either digit of the N character of an instruction

STANDARD HIGH SPEED MEMORY LOCATIONS

	00	01	02	03	04	05	06	07	10	11	12	13	14	15	16	17	20	21	22	23	24	25	26	27	30	31	32	33	34	35	36	37	
0000		RAI			ARITHMETIC TEMPORARY STORAGE												READ INSTRUCTIONS								WRITE INSTRUCTIONS								0000
	40	41	42	43	44	45	46	47	50	51	52	53	54	55	56	57	60	61	62	63	64	65	66	67	70	71	72	73	74	75	76	77	
0000	ROLLBACK ENTRY NORMAL MODE								ROLLBACK ENTRY SIMULTANEOUS MODE								FLAW DETECTION NORMAL MODE								FLAW DETECTION SIMULTANEOUS MODE								0000
	00	01	02	03	04	05	06	07	10	11	12	13	14	15	16	17	20	21	22	23	24	25	26	27	30	31	32	33	34	35	36	37	
0001										A.M.1																A.M.3							0001
	40	41	42	43	44	45	46	47	50	51	52	53	54	55	56	57	60	61	62	63	64	65	66	67	70	71	72	73	74	75	76	77	
0001										A.M.5																A.M.7							0001
	00	01	02	03	04	05	06	07	10	11	12	13	14	15	16	17	20	21	22	23	24	25	26	27	30	31	32	33	34	35	36	37	
0002	SHOULD HOLD NEXT INSTR. IF PA IN SIMULT. MODE																	STA A.M.2															0002
	40	41	42	43	44	45	46	47	50	51	52	53	54	55	56	57	60	61	62	63	64	65	66	67	70	71	72	73	74	75	76	77	
0002		STP																															0002
	00	01	02	03	04	05	06	07	10	11	12	13	14	15	16	17	20	21	22	23	24	25	26	27	30	31	32	33	34	35	36	37	
0003																																	0003

APPENDIX 5 : GLOSSARY

Access Time. A time interval which is characteristic of a storage device, and is essentially a measure of the time required to communicate with that device. The time interval between (1) the instant at which information is called for from storage and the instant at which it is delivered or (2) the instant at which information is ready for storage and the instant at which it is stored.

Accuracy. The quality of freedom from mistake or error, that is, of conformity to truth or to a rule. Accuracy is distinguished from precision as in the following example: A six-place table is more precise than a four-place table. However, if there are errors in the six-place table, it may be either more or less accurate than the four-place table.

Address (noun). An expression, usually numerical, which designates a particular location in a storage or memory device or other source or destination of information.

> **Absolute Address** (Actual Address). The specific label assigned by the machine designer to a particular storage location. To code in absolute means to write a sequence of instructions in a computer code.
>
> **Instruction Address** (Line Number, Location). An expression used in coding to denote the address of a stored instruction. NOT a part of the instruction itself.
>
> **Symbolic Address.** A label expressed in a pseudo-code. To code using symbolic addresses implies that the sequence of instructions must be translated into absolute before being executed by a computer. Relative addresses are those symbolic addresses which are translated into absolute by sequencing from some specific 'reference' address.

Address Modifier. See text, page 5.5 and Appendix 4.

Batch. Several groups of items in sequence, each separated by a special symbol and the entire grouping bracketed by SM-EM. (This is contrasted to a single group of related items for a message).

Beginning of Tape Control (BTC). A 'window' placed at the beginning of a magnetic tape, where recording of data is not possible and which can be sensed photoelectrically.

Binary Digit. See Digit.

Binary Representation. See Positional Notation.

Bit. Contraction of Binary Digit.

Block. See page 3.2.

Breakpoint Switch. There are 6 two-positional break-point switches on the Console in the KDP 10 System. When one of these is in the 'ignore' position and the related control bit is present in B_1 of a Transfer Control (71) instruction, the transfer will not take place; instead, the programme will proceed to the next instruction in sequence.

Buffer. A storage device used to compensate for a difference in rate of flow of information or in time of occurrence of events when transmitting information from one device to another, as from an input device to the High-Speed Memory, or from the High-Speed Memory to an output device.

Carry. (1) A condition occurring during addition when the sum of two digits in the same column equals or exceeds the base number. (2) The digit to be added to the next higher column.

Character. One of a set of elementary symbols which may be arranged in ordered aggregates to express information. These symbols may include decimal digits 0 through 9, the letters A through Z, punctuation symbols, typewriter symbols, and any other symbols which a computer may read, store, or write. See page 2.12 for list of KDP 10 characters.

Code (noun). A system of symbols and rules for their use in representing information. A language.

Pulse Code. The binary representation of characters.

Operation Code. The code representing an operation (add, subtract, transfer, etc.) built into the hardware of the computer.

Complement (noun). A quantity which is derived from a given computer quantity by the following rules.

a) Complement on n (as in tens complement). Subtract each digit of the given quantity from n-1, add unity to the least significant digit, and perform all resultant carries.

b) Complement on n-1 (as in nines complement). Subtract each digit of the given quantity from n-1.

Constant. A number is said to be a constant if it has the same value under all conditions. For example, in the formula (area of a circle) $= \pi \times (\text{radius})^2$, π is a constant, equal to 3.14159 ---, which applies to all circles.

Control Symbol. A character used to indicate the beginning or the end of a unit of data (item, message, file, etc.).

Counter. A device (register or storage location) for storing integers, permitting these integers to be increased or decreased by unity or by an arbitrary integer, and capable of being reset to zero or to an arbitrary integer.

Criterion (Key). A group of characters, usually comprising an item, used to identify a message.

Decimal Number System. See Positional Notation.

Digit. One of the n symbols of integral value ranging from 0 to n-1, inclusive, in a scale of numbering of base n.

Binary Digits are 0 and 1.

Octal Digits are 0 through 7.

Decimal Digits are 0 through 9.

Edit. To rearrange information. Editing may involve the deletion of unwanted data, the selection of pertinent data, the insertion of invariant symbols such as page numbers and typewriter characters, and the application of standard processes such as zero suppression.

End Data Symbol. See Organisation of Data on Tape (text).

End File Symbol. See Organisation of Data on Tape (text).

End Message Symbol. See Organisation of Data on Tape (text).

End of Tape Warning (ETW). A warning generated by a metal strip located 15 to 20 feet before the physical end of tape.

File. See Organisation of Data on Tape (text).

Flip-Flop. A device having two stable states and two input terminals (or types of input signals), each of which corresponds to one of the two states. (The two states may be considered as corresponding to 'off' and 'on' or to binary 0 and 1. The circuit remains in either state until it is caused to change to the other state by application of the corresponding signal.

Gap. See Organisation of Data on Tape (text).

High-Speed Memory. Magnetic core storage in the Computer in the KDP 10 System. See also Storage.

High-Speed Memory (HSM) Location. A unit of magnetic core storage (high-speed Memory) which can store (hold, remember) one KDP 10 character (one octal number, two octal digits).

Input. (1) Information transferred into the computer. (2) The device by means of which information is fed into the computer.

Instruction. A set of symbols which directs the computer to take a given action.

Intermessage Gap. See Organisation of Data on Tape (text).

Item. See Organisation of Data on Tape (text).

Item Separator Symbol (ISS). Control symbol designating the beginning of an item.

Jump Table. Record indicating executed transfers out of programme sequence.

Justify. Shift an operand to effect right or left columnar alignment.

Key. See Criterion.

Line. See Organisation of Data on Tape (text).

Location. See High-Speed Memory (HSM) Location; Address; Storage.

Mask. A pattern consisting of 0 and/or 1 bits, used to alter the bit configuration of an operand.

Memory. See Storage.

Message. See Organisation of Data on Tape (text).

Number System. See Positional Notation.

Octal. See Positional Notation.

Octonary. See Positional Notation.

Operand. Any one of the quantities entering into an operation.

Output (noun). (1) Information transferred from the computer to external storage. (2) The device to which the computer delivers information.

Patch (noun). A section of coding inserted into a routine (usually by explicitly transferring control from the routine to the patch and back again) to correct a mistake or alter the routine.

Positional Notation. One of the schemes for representing numbers, characterised by the arrangement in sequence of digits which are to be interpreted as co-efficients of successive powers of an integer called the base of the number system.

In the binary number system the successive digits are interpreted as co-efficients of the successive powers of the base 2, just as in the decimal number system they relate to successive powers of the base 10.

In the ordinary number systems the digits are symbols which stand for zero and for the positive integers smaller than the base.

Names of number systems with base from 2 to 20: binary, ternary, quaternary, quinary, senary, septenary, octonary (also octal), nonary, decimal, unidecimal, duodecimal, terdenary, quaterdenary, quindenary, sexadecimal, (also hexadecimal), septendecimal, octodenary, novendenary, and vicenary.

The sexagenary number system has a base of 60. The commonly used alternative of saying 'base-3', 'base-4', etc., in place of ternary, quaternary, etc., has the advantage of uniformity and clarity.

Random Access. Access to storage under conditions in which the next position from which information is obtained, or to which it is delivered, is in no way dependent on the previous one.

KDP 10 Character. See the KDP 10 Code, Page 2.12.

Rerun. Rollback. See Appendix 1.

Rewind. Move a tape in a backward direction.

Rollback. See Appendix 1.

Sector. An area in the High-Speed Memory whose beginning and ending addresses are designated by the instruction.

Sign Position. The location to the right of the least significant digit of an item.

Standard Memory Locations. Designated locations in the HSM which are used for Address Modifiers, automatic storage of the final contents of the A Register, etc. (See Appendix 4).

Start Message Symbol. See Organisation of Data on Tape (text).

Start Time. Time between the command to start an input-output device and the reading or writing of the first character.

Storage Memory. A device into which units of information can be transferred, which will hold this information, and from which the information can be obtained at a later time.

Internal Primary Storage. Storage facilities forming an integral physical part of a computer.

Location. A storage position in the High-Speed Memory. Each location has a specific address and can hold one KDP 10 character.

Register. A storage device with a specifically assigned function and a given unit capacity. Registers in the Computer in the KDP 10 System are of one, three or four-character capacity.

External (Secondary) Storage. Storage facilities which are not an integral part of the computer proper, but comprise units of the data-processing system (magnetic tape, paper tape, etc.).

Working Storage. A portion of the internal storage reserved for intermediate and partial results during computation.

Symbols, KDP 10 Code. See Page 2.12.

Tetrad. A unit consisting of four consecutive HSM locations or the contents thereof. A tetrad starts in a $(\text{-----}0)_8$ or $(\text{-----}4)_8$ address and ends in $(\text{-----}3)_8$ or $(\text{-----}7)_8$ address, respectively. A tetrad address, however, is any one of its four location addresses.

Unwind. Move a tape in the forward direction.

Variable Item Length. See page 3.5.

Word (in Electronic Computers). An ordered set of characters comprising the normal unit, in which information may be stored, transmitted, or operated upon in a fixed-word or fixed-variable-word computer.

APPENDIX 6: ABREVIATIONS USED IN TEXT

ABE	A Counter and B Counter Equality Flip-Flop
AOR	Adder Output Register
BA	Binary Add (41)
BRF	Block Read Forward (15)
BRR	Block Read Reverse (05)
BS	Binary Subtract (42)
BTC	Beginning of Tape Control
CIG	Character present in the Gap
CSG	Control Simultaneous Gate (75)
CTC	Conditional Transfer of Control (61)
DA	Decimal Add (51)
DD	Decimal Divide (54)
DM	Decimal Multiply (53)
DS	Decimal Subtract (52)
ED	End Data Symbol
EF	End File Symbol
EM	End Message Symbol
EMP	Electro-Mechanical Printer
ETW	End of Tape Warning
f	Used as subscript to denote 'final'
HSM	High Speed Memory
i	Used as subscript to denote 'initial'
ISS	Item Separator Symbol
IT	Item Transfer (21)
JR	Justify Right (33)
KC	Thousand Characters Per Second
L to R	Left to Right
LA	Logical 'and' (47)
LNS	Locate nth Symbol in Sector (31)
LO	Logical 'or' (46)
LRF	Linear Read Forward (14)
LRR	Linear Read Reverse (04)
LS	Line Shift
LSC	Least Significant (or rightmost) Character
LSD	Least Significant Digit
LW	Linear Write (12)
MSC	Most Significant (or leftmost) Character
MSD	Most Significant Digit
MSW	Multiple Sector Write (13)
NO	Normal Operation (Register)
OCT	One-Character Transfer (22)
PA	Paper Advance (03)
PC	Page Change
PET	Physical End of Tape
PR	Print (02)
PRI's	Previous Result Indicators
PRN	Previous Result Negative
PRP	Previous Result Positive
PRZ	Previous Result Zero
PS	Potentially Simultaneous
R to L	Right to Left