APPENDIX I

EMIDEC 6-BIT INTERNAL ALPHANUMERIC CODE

CHARACTER	CODE	CHARACTER	CODE
0	0 0 0 0 0 0	No Character	1 0 0 0 0 0
1	0 0 0 0 0 1	½	1 0 0 0 0 1
2	0 0 0 0 1 0	- (Minus)	1 0 0 0 1 0
3	0 0 0 0 1 1	Space	1 0 0 0 1 1
4	0 0 0 1 0 0	Buffer Control	1 0 0 1 0 0
5	0 0 0 1 0 1	Spare	1 0 0 1 0 1
6	0 0 0 1 1 0	A	1 0 0 1 1 0
7	0 0 0 1 1 1	B	1 0 0 1 1 1
8	0 0 1 0 0 0	C	1 0 1 0 0 0
9	0 0 1 0 0 1	D	1 0 1 0 0 1
10	0 0 1 0 1 0	E	1 0 1 0 1 0
11	0 0 1 0 1 1	F	1 0 1 0 1 1
12	0 0 1 1 0 0	G	1 0 1 1 0 0
%	0 0 1 1 0 1	H	1 0 1 1 0 1
£	0 0 1 1 1 0	I	1 0 1 1 1 0
&	0 0 1 1 1 1	J	1 0 1 1 1 1
*	0 1 0 0 0 0	K	1 1 0 0 0 0
/	0 1 0 0 0 1	L	1 1 0 0 0 1
.(stop)	0 1 0 0 1 0	M	1 1 0 0 1 0
¼	0 1 0 0 1 1	N	1 1 0 0 1 1
⅝	0 1 0 1 0 0	O	1 1 0 1 0 0
'	0 1 0 1 0 1	P	1 1 0 1 0 1
(	0 1 0 1 1 0	Q	1 1 0 1 1 0
)	0 1 0 1 1 1	R	1 1 0 1 1 1
+	0 1 1 0 0 0	S	1 1 1 0 0 0
x	0 1 1 0 0 1	T	1 1 1 0 0 1
1/3	0 1 1 0 1 0	U	1 1 1 0 1 0
⅞	0 1 1 0 1 1	V	1 1 1 0 1 1
∞	0 1 1 1 0 0	W	1 1 1 1 0 0
Figures	0 1 1 1 0 1	X	1 1 1 1 0 1
Letters	0 1 1 1 1 0	Y	1 1 1 1 1 0
CR/LF	0 1 1 1 1 1	Z	1 1 1 1 1 1

N.B. ∞ is a block marker for guillotine control.

APPENDIX II - BRIDGE INSTRUCTION CODE

	c address (B28-35)		b address (B18-27)	a address (B8-17)	Function (B3-7)	Mod. (B0-2)	Approx. Time µs	See Section No.
Halt Conditional Halt			0 1	Condition	0		270 390	3.3
Register to Register transfer			Destination	Source	1		120	3.3
Shift left	No. of shifts		"	"	2		160 + 10n	3.8
Shift right	"		"	"	3		"	3.8
Double-length shift left	"		"	"	4		200 + 10n	3.8
Double-length shift right	"		"	"	5		"	
Collate *			Source	"	6		140	3.8
Add			Source and Destination	"	7		"	3.3
Subtract			"	"	8		170	3.3
Multiply **			Multiplier	Multiplicand	9		1260	3.9
Divide **			Quotient	Divisor	10		1440	3.9
Test zero			Next Instruction	Source register	11		140- 190	3.10
Test non-zero			"	"	12		140- 190	3.10
Test positive			"	"	13		150	3.10
Test negative			"	"	14		140	3.10
Transfer from drum	No. of blocks		First block No.	First destination register	15		5.15 -51.5 msec.	4.2
Transfer to drum	"		"	First Source register	16		4.9 -51.05 msec.	4.2
Peripheral unit to register block transfer	Unit No.		Unit Control (where needed)	First destination register	17		1190	7.1.5.
Register to peripheral unit block transfer	"		"	First Source register	18		1130	7.1.6.
Input decimal	"	No. of chars.	"	Destination	19		270- 1680	7.1.5.
Input sterling	"	"	"	"	20		"	7.1.5.
Input alphanumeric	"	"	"	"	21		340	7.1.5.
Output decimal	"	"	"	Source	22		2990	7.1.6.
Output sterling	"	"	"	"	23		3320	7.1.6.
Output alphanumeric	"	"	"	"	24		290	7.1.6.
Count test ***			Next instruction	Subtractor source	25		195- 230	3.10
Double-length add			Source and destination	Source	26		299	3.3
Double-length subtract			"	"	27		310	3.3
Output sterling with spaces	Unit No.	No. of chars.	Unit control (where needed)	"	28		3720	3.11
Set Link and Jump ****			Next instruction	Number	29		110	3.11
Negative Sum	No. of Registers		Destination	First Source register	30		160- 1300	3.11

* The result of a collate operation appears in register 10.

** The product of a multiplication appears in register 8 and 9 the dividend is placed in these registers for division.

*** The count register is register 7.

**** The number is set in the B address position of register 7.

APPENDIX III- PAPER TAPE

5-hole INPUT-OUTPUT CODE

FIGURES CASE	LETTERS CASE	TAPE 5 4 3 . 2 1	DECIMAL VALUE
Figures	Shift	0 0 0 . 0 0	0
%	P	0 0 0 . 0 1	1
£	Q	0 0 0 . 1 0	2
- (Minus)	R	0 0 0 . 1 1	3
*	S	0 0 1 . 0 0	4
/	T	0 0 1 . 0 1	5
. (Stop)	V	0 0 1 . 1 0	6
¼	W	0 0 1 . 1 1	7
½	X	0 1 0 . 0 0	8
'	Y	0 1 0 . 0 1	9
(	Z	0 1 0 . 1 0	10
)	D	0 1 0 . 1 1	11
+	M	0 1 1 . 0 0	12
x	U	0 1 1 . 0 1	13
¾	&	0 1 1 . 1 0	14
Space	Space	0 1 1 . 1 1	15
0	A	1 0 0 . 0 0	16
1	B	1 0 0 . 0 1	17
2	C	1 0 0 . 1 0	18
3	E	1 0 0 . 1 1	19
4	F	1 0 1 . 0 0	20
5	G	1 0 1 . 0 1	21
6	H	1 0 1 . 1 0	22
7	I	1 0 1 . 1 1	23
8	J	1 1 0 . 0 0	24
9	K	1 1 0 . 0 1	25
10	L	1 1 0 . 1 0	26
11	N	1 1 0 . 1 1	27
12	O	1 1 1 . 0 0	28
CR/LF/NE	CR/LF/NE	1 1 1 . 0 1	29
LETTERS	SHIFT	1 1 1 . 1 0	30
ERASE	ERASE	1 1 1 . 1 1	31

APPENDIX IV - PUNCHED CARD INPUT CODE

Numbering the hole positions of the card from top to bottom

A B C 1 2 3 4 5 6 7 8 9

A to I	Position	A and 1 to A and 9
J to R	"	B and 1 to B and 9
S to Z	"	C and 1 to C and 8
No Character	"	C and 9
1 to 9	"	1 to 9
10	"	A
11	"	B
0	"	C
½		3 and 1
/		3 and 2
¢		3 and 4
1/3		3 and 5
X		3 and 6
&		3 and 7
.		3 and 8
-		3 and 9

By special arrangement further combinations may be made available for the following characters.

*	£
¼	#
¾	[illegible]
'	12
(	
)	
+	

APPENDIX V - SAMASTRONIC PRINTER CHARACTERS

The range of characters available on the Samastronic printer is:-

0 to 11
A to Z
½ ⅓ ⅔ + -(minus) * (asterisk)

%
£
. (Stop)
/
$
& (Ampersand)
' (Apostrophe)
[redacted] (Guillotine Control)
1/3
x
(
)
12

All 57 symbols are available

The printer leaves a blank where it is sent a signal for either 'space' or 'no character'.

Characters sent to the printer buffer which are not included in the character range chosen for the printer will also have the same effect as 'space' and 'no character'.

APPENDIX VI

THE USE OF SIMULATORS

1. Inhibiting the printer

If the speed of a job is governed by printing, and if it is desired to run the job quickly without producing printed output (in testing, for example, or to reconstitute a magnetic tape), this can be simply achieved by use of a plugboard at the bottom of the computer rack. The board has sockets corresponding to the sixteen computer input/output channels, and plugs corresponding to the buffers of peripheral units, normally plugged into their appropriate channel number; there are also five plugs which simulate buffers ready to transfer. All that is required to inhibit the printing process is to remove the printer buffer plug from its socket (normally No.9) and insert instead one of the 'dummy buffer' plugs. Then, whenever the program contains a print instruction, the information will be immediately output and lost, as the dummy buffers appear to the computer to be always ready for information. No delay is caused to the program by the printer, and no amendments are required.

2. Other Peripheral Units

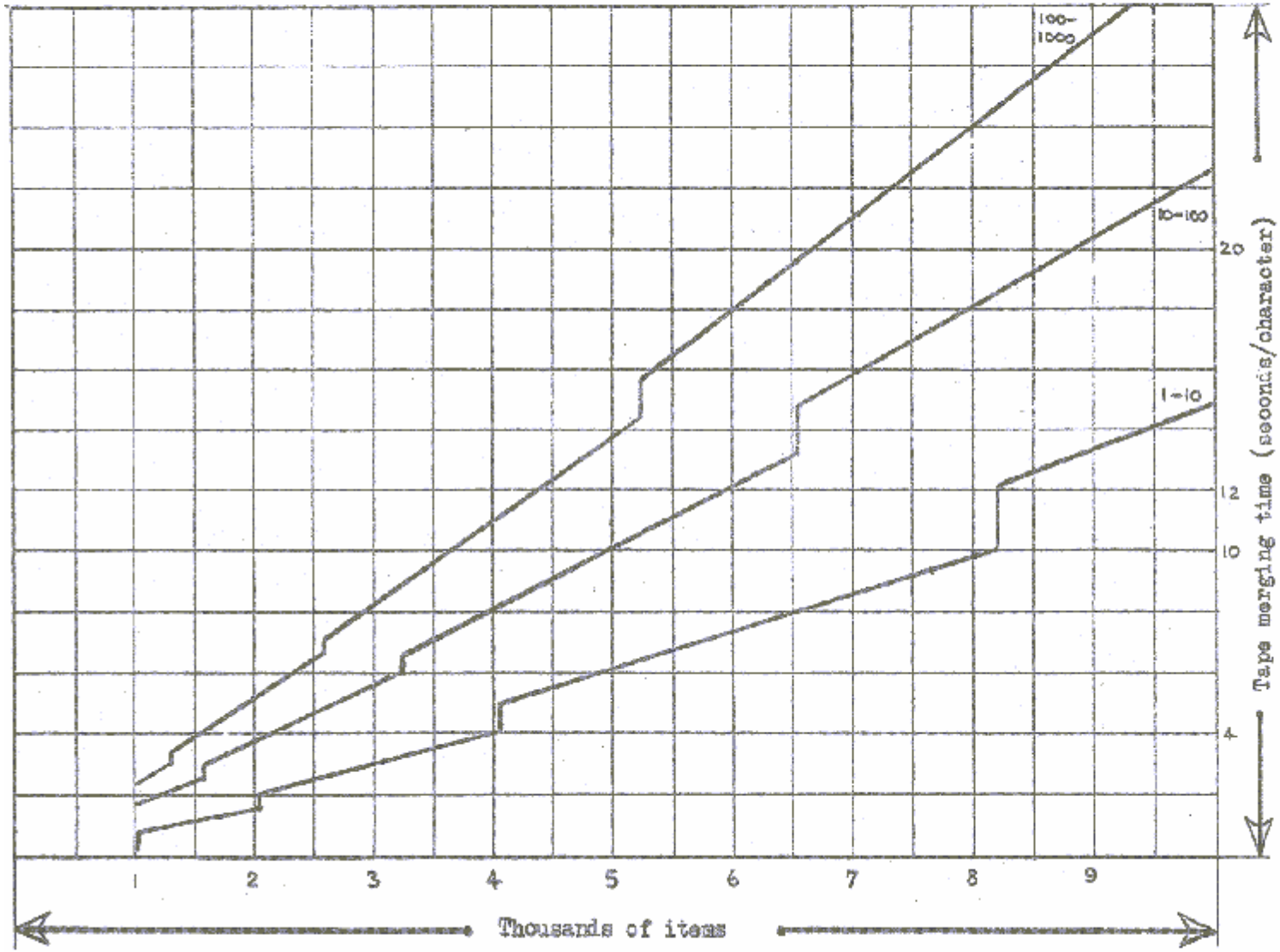
The same process can be applied to any input/output unit, but it is anticipated that the inhibiting of printing is likely to be the most valuable.

3. Register Clearance

A particular use of simulators is found when it is desired to clear 16 consecutive registers in the Core Store. By using a Block Input instruction [function 17] in connection with a simulated peripheral unit (normally unit No 15), the 16 register starting at Register 'a' may be cleared in 1.2 milliseconds.

APPENDIX VII

CALCULATION OF SORTING TIMES FOR STANDARD EMIDEC PROGRAM



Instructions:

1. Calculate time for reading in data.
2. Allow 1 min. per 1000 items for internal sorting.
3. Using appropriate scale and line on graph, read off tape merging time per Character.
4. Multiply this by the number of characters in an item, and convert to minutes.
5. Add 1, 2 and 4.

Notes:

1. Times given assume 4 magnetic tape units; if only 3 units add 50% to tape merging time.
2. No operating allowance is included.
3. These times may not prove completely accurate for very large or very small records.

For 10,000 - 100,000 or 100,000 - 1,000,000 items, multiply both scales by a factor of 10 or 100 and use the appropriate line.

APPENDIX VIII

RESTARTING A JOB AFTER DISCOVERY OF A FAULT

Short Jobs

Possibly the simplest and cheapest way of dealing with a major breakdown on a short job (of say one hour or less) would be to clear the computer and re-run the job completely after the fault has been located and corrected.

Long Jobs - Restart Groups

In a longer run of say 4 hours, to avoid the loss of time of a complete re-run, the fault is isolated after examining programmed reconciliations printed periodically during the run. Thus, the earlier discovery of errors would save a complete re-run and so avoid time lost. If, at reconciliation points, totals are accumulated and recorded on an output medium, they can be re-fed and any re-run started from that point. The division of a large job into such restart groups reduces time lost due to breakdowns to the time taken between reconciliation points, plus an allowance for re-feeding the totals and aligning the input.

Refeeding totals

Paper tape or punched card output may be satisfactorily used for recording restart totals if they are available, but as magnetic tape is the most common form of recording medium, it will be considered here. The restart totals, including an identifiable word and numbered sequentially, are recorded as they occur on one of the tape units, used for output. The normal output is also recorded on this unit. On discovery of an error the unit is rewound and set to read by means of program instructions or manually. With each print out of reconciliation results the restart group number has also been printed, and the number of the preceding correct restart group is fed to the computer by means of the Switches and using register 16. The machine is restarted and the magnetic tape is searched until the correct restart group totals are found and stored. The tape unit is then manually set to write, by depressing the button situated in the unit cabinet, its buffer is cleared by means of the appropriate switch in the buffer control cabinet and the computer is restarted.

Aligning Input

The alignment of the input data must now be considered and a fairly simple method is to record with the restart totals the last item of input data dealt with in that group and also to print it with the reconciliation

figures. In many cases an identifying number will suffice and this is used to align the current inputs for restarting purposes. Broadly speaking, a restarting routine will arrange for input tape units to be rewound, in some cases replaced by the previous reel if the size of the restart group warrants it, and then to re-read the input, searching for the last item dealt with in the correct group.

It will be necessary when recording restart totals on magnetic tape to record 4 sixteen word blocks of zeros following the totals. This will ensure that, irrespective of where in the magnetic tape buffer the last restart totals occur, the remainder of the buffer, if any, will be filled with zeros and, therefore, what has so far been read from the magnetic tape will not include any suspected incorrect data. Clearing the buffer at this stage ensures that the restart totals are not recorded twice. It follows from this that, whenever a tape containing restart totals is read, there will be a variable number (3 or 4) of zero blocks following the totals.

In some instances, it will be possible to read an output tape to a specified restart group, re-recording this tape on to a second tape at the same time. This makes it unnecessary to change a deck manually from read to write, and ensures that the correct point on the second tape is reached for further recording without the need to consider the state of the buffers.

The method is a little unwieldy, however, from an operator's point of view, particularly if there is more than one output tape concerned.

From the foregoing it will be appreciated that restart groups and totals will involve some considerable programming and operating, and it should be stressed that there should be a strong case for it before this procedure is used. In many cases the timing of a job is not significant and in other, longer jobs, the time loss can be reduced considerably by re-running the job with suppressed printed output, a simple manual adjustment.

APPENDIX IX

PROGRAMMING NOTES FOR THE EMIDEC 1100 AND 1101

WITH THE ENLARGED CORE STORE

1. Introduction

A new version of the Emidec 1100 machine with an immediate access store of 4096 words instead of 1024 is now available to customers.

This appendix explains how the enlargement of the core store has made additional facilities available in the use of transfer functions 15 and 16.

2. Block Location

The original immediate access store remains as the working store of 1024 words (or 256 blocks). The additional core storage of 3072 words (768 blocks) is a backing facility and cannot be used as a working store. The core store as such now has a capacity of 4096 words (1024 blocks) which are all directly accessible.

A block is defined as four consecutively selectable words which occur at 1024 fixed points throughout the core store and throughout the drum store.

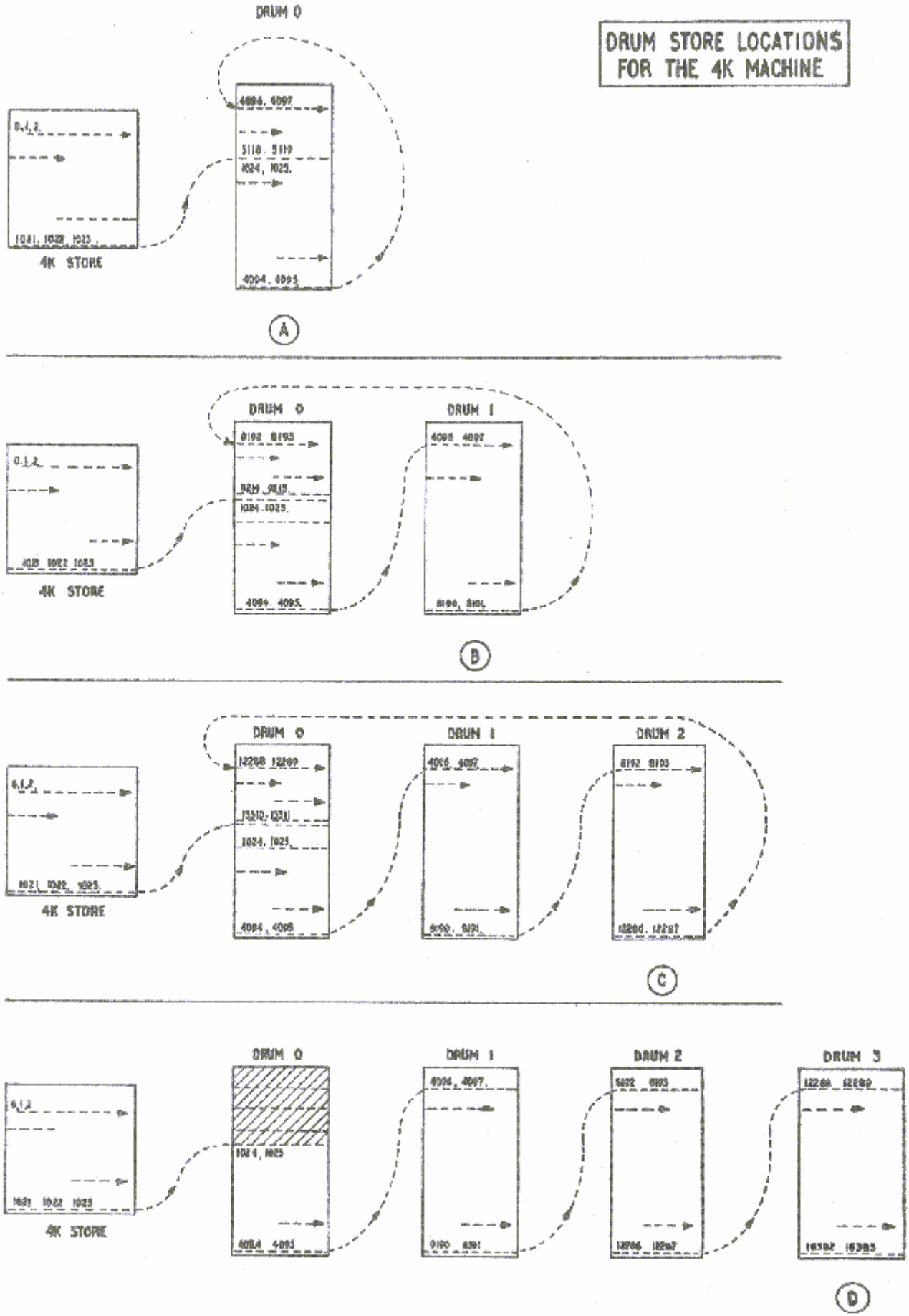
Diagrams A, B, C and D show how subsequent blocks are arranged on the drums of machines which have 1, 2, 3, and 4 drums. Although the physical layout of these blocks does not affect programs, it is necessary knowledge for monitoring drum store data.

3. Functions 15 & 16

3.1.1 General

The most important difference between the capabilities of the transfer functions 15 and 16 on the enlarged and original core stores is that, whereas on the original machines transfers can only be effected between drums and core store, on the enlarged ones inter-core store transfers can be made as well. On machines with the large core store, functions 15 and 16 are known as block transfer and not as drum transfer functions. Programers should think in terms of block transfers, irrespective of the location of the source and destination registers.

DRUM STORE LOCATIONS FOR THE 4K MACHINE



3.1.2 The B and C_1 Addresses as a Compound Location

The greatest quantity which can be expressed in the B address, which normally comprises 10 bits, is 2^{10} or 1024, but since this is the number of blocks contained in the core store, reference could not be made to a drum block by employing the B address by itself. In order to be able to refer to drum blocks, the C_1 address with its capacity of 4 bits, combines with the B address upon a 15 or 16 directive to form a compound location of 14 bits, wherein can be specified a maximum quantity of 2^{14} or 16384, which is the number of blocks available on a 4 drum machine. It follows that when the C_1 address is non-zero a drum transfer is called for, and that when C_1 is empty the operation demanded is an inter-core store transfer.

3.1.3 Addressing

For a 15 or 16 function, the A address always refers to a word number of the working store and the $(B + C_1)$ address to a block number, irrespective of whether the block specified in the $(B + C_1)$ address exists in the core or the drum store, and irrespective of whether the transfer demanded is an inter-core or core to drum operation.

3.2.1 Transfer Instructions

Function 15 demands that the number of blocks specified in the C_2 address (maximum is 16 blocks when C_2 is empty) shall be transferred from the locations whose first block number is given in the $(B + C_1)$ address to the working store, starting at the register specified in the A address.

Example 1. Transfer or read 3 blocks from (drum) block No. 1026 into registers 32-35 of the working store.

F	A	B	C_1	C_2	M
15	32	2	1	3	0

Example 2. Transfer or read 2 blocks from (drum) block No. 10 relative to start of second drum and place them in registers 100-107.

F	A	B	C_1	C_2	M
15	100	10	4	2	0

Example 3. Transfer or read 16 blocks from block 260 of the backing store and place them in the working store starting at register 60.

F	A	B	C_1	C_2	M
15	60	260	0	0	0

Example 4. Read the 16 registers numbered 128-143 inclusive to locations 256-271

	F	A	B	C_1	C_2	M
	15	256	128/4	0	16/4	0
i.e.	15	256	32	0	4	0

Function 16 demands that the number of blocks specified in the C_2 address (maximum 16 blocks when C_2 is empty) shall be transferred from the locations whose first word number is given in the A address to a group of consecutive blocks whose first block number is given in the $(B + C_1)$ address.

Example 1. Write 4 blocks from registers 30-45 to (drum) blocks 1024-1027

F	A	B	C_1	C_2	M
16	30	0	1	4	0

Example 2. Write 10 blocks on (core) blocks 300-309 from registers 61-100

F	A	B	C_1	C_2	M
16	61	300	0	10	0

Example 3. Write 16 words on registers 256-271 from registers 128-143

	F	A	B	C_1	C_2	M
	16	128	256/4	0	16/4	0
i.e.	16	128	64	0	4	0

3.3.1 Overlapping

In an inter core-store transfer, words are transferred one at a time, and each word is written in its destination before the start of the transfer of the next word. Thus, in the case of an overlap

of the source and destination locations, some registers will be overwritten before they are used as sources. This produces a repetitive pattern along the destination locations and would also lose some source data as a result of overlapping.

Example. The following overlap instruction requires 4 words to be transferred from registers 198-201 to registers 200-203.

F	A	B	C_1	C_2	M
16	198	50	0	1	0

The effect of this instruction is indicated in table A.

Table A.

Register Number	Contents Before Transfer	Contents After Transfer
197	A	A
198	B	B
199	C	C
200	D	B
201	E	C
202	F	B
203	G	C
204	H	H

3.3.2 Clearing Registers

Here is a typical application of the principle indicated in the previous example. The following two instructions would empty registers 399-419.

F	A	B	C_1	C_2	M
1	0	399	0	0	0
16	399	100	0	5	0

3.3.3 "Shifting Down"

It is evident that when overlapping destination registers are lower down the store than the source registers, overwriting will not occur. The effect will just be an overall displacement.

Example. See table B for the result of:

F	A	B	C_1	C_2	M
16	50	12	0	2	0

Table B.

Register Number	Contents Before Transfer	Contents After Transfer
47	A	A
48	B	D
49	C	E
50	D	F
51	E	G
52	F	H
53	G	J
54	H	K
55	J	L
56	K	K
57	L	L
58	M	M

3.4 Sequence of Transfer

In the drum backing store blocks are transferred sequentially within one track of a drum. For example, the sequence of transfer of 4 blocks starting at block 14 of track 0 of drum 0 would be

14. 15. 0. 1.

In the core backing store, words are transferred sequentially with block 0 following the last block 1023; thus word 0 follows word 4095.

To illustrate this point, consider the instruction

F	A	B	C_1	C_2	M
16	1018	255	0	2	0

and refer to table C.

Table C.

Register Number	Contents Before Transfer	Contents After Transfer
1017	A	A
1018	B	B
1019	C	C
1020	D	B
1021	E	C
1022	F	B
* 1023	G	C
1024	H	B
1025	J	C
1026	K	Contents of Register 0
1027	L	Contents of Register 1
1028	M	M

* = End of working store.

3.5 1K Programs on 4K Machines

It must be remembered that the lower block numbers on a 4K machine specify the lower register numbers on a 1K store, so that programs written for 1K machines may easily prove abortive if run on a 4 K machine. On a 4K machine, attempts to write into blocks 0 to 15 (i.e. Track 0, Drum 0 on a 1K machine) will lose some data by writing into registers 0-63. Data will be lost on the special number registers, and program in registers 32 onwards would be overwritten.

This difficulty can be overcome without rewriting programs however. A means is provided whereby a program intended for a 1K machine can be proved on a 4K machine and which enables the use of common test programs. An engineering facility exists whereby a 4K machine can quickly be turned into a 1K system. When certain plug connections on the control matrix plugboard are changed, blocks 0 to 1023 will select tracks 0 to 63 of drum 0 and the core storage will no longer be accessible as backing store.

4. Operation Times

Function 15. From Core Backing Store

T = 240 + 180n Microseconds for n blocks.

From Drum Backing Store

T = same as 1K machine less 20 Microseconds.

Function 16. To Core Backing Store

T = 240 + 180n Microseconds for n blocks.

To Drum Backing Store

T = same as 1K machine less 20 Microseconds.