TRANSCODE

A SYSTEM OF CODING FOR THE FERRANTI MARK I COMPUTER

BY

J. N. P. HUME

B. H. WORSLEY

COMPUTATION CENTRE
UNIVERSITY OF TORONTO
CANADA

FERUT LIBRARY ROUTINE

Class Composite Date 1 Oct. 1954 Name TRANSCODE

Purpose TRANSCODE is a comprehensive system of routines which enables a Ferranti Mark I Computer to utilize a three-address, mnemonic code of instructions operating on floating-decimal numbers. Tape controls are provided for facilitating organization of the programme and input of both data and instructions.

TRANSCODE is intended to provide a system of coding which is simple to learn and to apply, and which requires only a cursory knowledge of the specifications of the computer. This system will first be described, assuming such a background.

An appendix is also included for the benefit of programmers familiar with the Ferranti machine code and the Toronto system of Input and Organization. However, a study of this appendix is not necessary for learning to operate in the TRANSCODE manner.

TRANSCODE is expected to be particularly useful in solving single-shot problems, and in doing exploratory work in production problems for which scaling presents unusual difficulties. It is believed that the main loss of speed in working with TRANSCODE rather than in the machine code is only that inherent in the use of a floating-point system of notation.

Source J.N.P.H. and B.H.W. (Toronto)

Code

The calculation must be broken down into a logical sequence of operations in terms of the code of available Operational Instructions. Thus a "programme" is written in the form

Instruction number	Instruction	Three-Address Operand		
001	LOOP	012.0	000.6	000.0
002	MULT	X12.6	Y12.6	Z12.6
⋮	⋮	⋮	⋮	⋮
		and generally		
00n	ABCD	$\alpha . B_\alpha$	$\beta . B_\beta$	$\gamma . B_\gamma$
⋮	⋮	⋮	⋮	⋮

where ABCD is a 4-letter word indicating what the instruction does,

α, β, γ are 3-character entities indicating either an address or a decimal integer,

B_α, B_β, B_γ are single decimal digits.

The code of operational instructions is given on pages 5 and 6.

Numbers

TRANSCODE deals with numbers in floating decimal form. The general form is

$$\pm a.bb \ldots b \times 10^{\pm n}$$

where a = 1,2,3 -- 9; b = 0,1,2--9 and n = 0,1,2, -- .
All numbers entering into a calculation should be divided into three classes as follows:

1. Data, which must be grouped into sets called pages. Two types of pages, X and Y, are available simultaneously in the electronic store, although many such pages may be stored initially on the drum. The contents of each X or Y page may also be transferred to or from the drum at any stage of the calculation.

2. Constants, only one such set per programme, which are input initially and separately from the data, and are distributed automatically to be available as required by the programme.

3. Intermediate Values, which may temporarily be placed on a Z page in the electronic store. The Z page can thus be thought of as temporary working space, and may not be transferred to and from the drum.

Hence all numbers can be called upon by the operational instructions according to their addresses, α, β or γ where these are of the form:

X01	Y01	C01	Z01
X02	Y02	C02	Z02
⋮	⋮	⋮	⋮
			Z13
X21	Y21	C21	

X,Y and Z addresses can be B-modified in the manner described on page 7. They can be filled as the result of obeying an operational instruction. C addresses cannot be B-modified, and may not be filled as the result of obeying an operational instruction.

Input and Organization

Any given calculation proceeds by the execution of its programme. However, it is necessary to arrange for the input and organization of the programme, data, and associated constants. In TRANSCODE, this is greatly facilitated by the provision of a set of Tape Controls which operate from the input tape, and which should now be referred to. See page 4.

All pages of data are read into the electronic store by means of NUMB, and must immediately be stored on the magnetic drum by means of the DRUM control. X and Y pages of data are indistinguishable during input. The necessary set of constants is read in by means of CNST, and automatically stored.

The set of operational instructions constituting the programme is input by means of INST. Only one such set is permissible, and must be punched without interruption. Spaces may precede any of the Tape Controls, and a STOP (I//G) inserted whenever it is desired to stop the tape from being read-in.

For lengthy programmes it may be convenient to break the set of instructions down into sub-sections. Each of these may be written separately in TRANSCODE, and prepared for further use as described in the Section under "Write-Tape Procedure". Library tapes are also provided for the more commonly occurring processes such as evaluating the simple functions, integrating differential equations, etc. A list of all functions to be used in a programme should be made out, and these numbered consecutively beginning at .001. Tapes for these may then be inserted, in any sequence, directly into the complete problem tape, provided each is preceded by $\left\{\begin{array}{l}\text{Spaces}\\ \text{FNTN 00n}\\ \text{Spaces}\end{array}\right\}$ and read in

before ENTR. A stop (FF/G) is provided at the end of each FNTN tape so that it is not necessary to punch a STOP. These functions are called into a calculation by means of FNTN used as an operational instruction.

TAPE CONTROLS

INST oop	To read in the programme. INST 00p should be followed directly by p operational instructions, punched without instruction number or decimal point, and terminated by QUIT. p = 001, 002, 003 ---.
CNST abb---b±n± abb---b±n± ⋮ abb---b±n±	To read in the set of constants. Assimilation by the programme is automatic.
NUMB abb---b±n± ⋮ abb---b±n±	To read in a page of at most 21 numbers to the electronic store. Should be followed by a DRUM order.
DRUM 00m	To copy a set of numbers in the electronic store to the magnetic drum position m = 001, 002 ---. Drum positions are divided into two sets. See p.13, A.8.
FNTN 00α	To copy a library function tape, or the write-taped instructions for a sub-section of the programme, into function location α in the magnetic store. α = 001, 002 ---.
STOP	To stop the tape from being read in (I//G).
ENTR	To initiate translation of the instructions, read in by means of the above tape controls, into the form required by the machine, and to start the actual calculation. A stop (ØØ/L) separates the translation from the calculation.
REEN	To re-start the operation manually just after the ØØ/L stop of ENTR. This assumes that the entire process up to this point has once taken place successfully, and that no one else has used the computer in the meantime.
Blank tape	No effect. Can precede any tape control.
KOPY	Initiates the output of all translated instructions into a form suitable for re-input. See pages 13 and appendix. Read in FNTN 000 tape to KOPY.

OPERATIONAL INSTRUCTIONS (1) TRANSCODE

General Form $\boxed{\text{ABCD}}\ \boxed{\alpha.B_\alpha}\ \boxed{\beta.B_\beta}\ \boxed{\gamma.B_\gamma}$

ADDN $[\alpha.B_\alpha] + [\beta.B_\beta] \rightarrow \gamma.B_\gamma$	α,β = X, Y, Z or C addresses. γ = X, Y or Z address. B_α, B_β, B_γ are B-line mods. (C address can't be B-mod) ★ $\alpha.B_\alpha$ and $\beta.B_\beta$ remain unaltered.
SUBT $[\alpha.B_\alpha] - [\beta.B_\beta] \rightarrow \gamma.B_\gamma$	Addresses as for ADDN. 1/2 QRT assume $[\alpha.B_\alpha] \geqslant$ zero, $N\frac{1}{2}$/G stop if $[\alpha.B_\alpha]$ negative.
MULT $[\alpha.B_\alpha] \times [\beta.B_\beta] \rightarrow \gamma.B_\gamma$	
DIVD $[\alpha.B_\alpha] \div [\beta.B_\beta] \rightarrow \gamma.B_\gamma$	
1/2 QRT $\sqrt{[\alpha.B_\alpha]} \longrightarrow \gamma.B_\gamma$	
KOMP $\lVert[\alpha.B_\alpha]\rVert - [\beta.B_\beta] \rightarrow \gamma.B_\gamma$	
ZERO $\alpha.B_\alpha$	Place zero in storage location α of the XY or Z type, B-mod.
OVER $[\alpha.B_\alpha] \rightarrow \gamma.B_\gamma$	α = X, Y, Z or C address. γ = X, Y or Z address. B_α, B_γ are B - line mods. (C address can't be B-mod) $[\alpha.B_\alpha]$ remains unaltered.
LOOP α.0 0.B 0.0	Prepare to cycle through the subsequent set of instructions α times, using B-line B to alter any X, Y or Z addresses modified by this B line. The set of instructions should be terminated by a B-conditional TRNS order involving the same B-line. See p. 7.
TRNS α.0 0.0 0.0	Jump (i.e. transfer control) unconditionally to instruction number α. (α = 001, 002, ---)
TRNS α.0 0.B 0.0	To be used in connection with the LOOP instruction.Subt's. 3 and tests. See page 7 and Appendix.
TRNS α.0 0.0 γ.0	Jump to instruction number α if $[\gamma] \geqslant 0$, otherwise go on to next instruction. γ = X, Y or Z address. $[\gamma]$ remains unchanged.

★ $[\alpha]$ is used to designate the number stored in address α.

OPERATIONAL INSTRUCTIONS (2)

READ	$\alpha.0$	0.B	$\gamma.0$	Copy the contents of DRUM position α, modified by B-line B, to electronic page position γ. α = 001, 002, . . . 00n. γ = X00 or Y00. See Page 7.
WRTE	$\alpha.0$	0.B	$\gamma.0$	Copy the contents of electronic page position γ to DRUM position α. α, β, and γ as for READ. See page 7.
HALT	0.0	0.0	0.0	Insert a stop (///G) into the programme.
VOID	0.0	0.0	0.0	Insert a dummy instruction into the programme.
COKE	0.0	0.0	0.0	Refresh material permanently required in the electronic store. Must be inserted after every 5 or 10 minutes of calculation.
QUIT	0.0	0.0	0.0	Make final preparations for obeying the programme. Must be the last instruction in any set, although not last on the tape.
PRNT	$\alpha.B_\alpha$	$\beta.0$	$\gamma.0$	Print and/or punch floating decimal numbers. α numbers, selected consecutively. B_α numbers per paper line. β digits per amplitude. ($\beta \leqslant 10$, and not rounded-off). γ-address of first number (X,Y or Z). Two digits per exponent. One space after amplitude, sign preceding. Two spaces after exponent, sign preceding. Extra line-feed at end. Assumes $-75 <$ exponent $\leqslant 99$, otherwise incorrect number output or closed loop occurs. At most 64 digits, signs, etc. in a paper line. Alters B2 and B3.
BSET	0.B	$\beta.0$	0.0	Put integer β in B-line B.
	0.B	0.0	$\gamma.0$	Put $[\gamma]$ in B-line B. γ = X,Y,Z or C address. See page 16.
JOTB	0.B	0.0	$\gamma.0$	Plant [B-line B] in γ. γ = X, Y or Z address, see page 16.
INCB	0.B	$\beta.0$	0.0	Add the integer β to [B-line B].
NEGB	0.B	$\beta.0$	0.0	Subtract integer β from [B-line B].
	0.B	0.0	$\gamma.0$	Subtract $[\gamma]$ from B-line B. γ = X,Y,Z or C address. See page 16.
FNTN	$\alpha.0$	$\beta.B_\beta$	$\gamma.B_\gamma$	Place the function numbered α, of $[\beta]$, in γ. α = 001, 002, 003, . . as assigned for the programme. β=X,Y,Z or C address; γ=X,Y or Z address. β,γ(B mod). Can be used for functions with more than one argument or answer. Then set β = 000, γ = Z14, and consult separate description.

B- Modification

In order to cut down on the total number of instructions required to carry out a repetitive process, the B-modification facility should be used.

The LOOP instruction is intended to be used in conjunction with the B-dependent, TRNS instruction to cycle through an intermediate set of instructions a given number of times. Any X, Y or Z addresses occurring in the intermediate instructions are then modified progressively. The same B line should be used consistently, and the X, Y or Z addresses of the last set of numbers to be operated on should be written into the programme.

For example: Place zero in each of the locations X01, X02 --- X21, beginning with X01.

```
     001 | LOOP  021.0   000.3   000.0 |
   ┌→002 | ZERO  X21.3   000.0   000.0 |
   └─003 | TRNS  002.0   000.3   000.0 |
```

These 3 instructions can therefore be used in place of the following 21:

```
         | ZERO  X01.0   000.0   000.0 |
         | ZERO  X02.0   000.0   000.0 |
         |         |       |           |
         |         |       |           |
         | ZERO  X21.0   000.0   000.0 |
```

Instructions referring to drum storage locations can also be modified progressively, but this must be done by direct setting and modification of some B-line.

For example: Write the X page on drum position 001 + [B4], where B4 is to contain n.

```
         | BSET  000.4   00n.0   000.0 |
         | WRTE  001.0   000.4   X00.0 |
```

If n is to take on the values 0, 1, 2 -- 15 in succession, these instructions should be imbedded into a loop as follows:

```
     001 | BSET  000.4   000.0   000.0 |
     002 | LOOP  016.0   000.3   000.0 |
   ┌→003 | WRTE  001.0   000.4   X00.0 |
   │ 004 | INCB  000.4   001.0   000.0 |
   └─005 | TRNS  003.0   000.3   000.0 |
```

Note In terms of the actual machine, when an instruction is B-modified, the contents of the B-line mentioned are added to the instruction just before it is obeyed, although the instruction itself remains unaltered in the electronic store.

Print-Checking

In the development stages of testing a programme on the computer, it may be found useful to print out intermediate values occurring in the calculation which would not be desired in the production runs. This can readily be achieved by inserting extra PRNT instructions in the programme, later altering these to VOID instructions by hand-punching and re-perforating the test programme tapes.

Speed

Reading-in of INST'ns., 5 or 6 instructions per Sec.

* Reading-in of NUMB or CNST, 27 msec.✓✓ per decimal digit.
Translation into real code, 4 or 5 instructions per Sec.

	QUIT	1 to 2 Sec.
**	ADDN	73 to 89 msec.
	SUBT	73 to 89 msec.
	MULT	63 msec.
	DIVN	(126 + 15n) msec., n = 1,2,...6.
	1/2 QRT	(155 + 26n) msec., n = 1,2,...6.
	READ	75 msec.
	WRTE	108 msec.
	TRNS	50 msec. (4 msec. if back to same track)
	All B-line instructions	.96 m Sec.
	ZERO	5 m Sec.
	COKE	100 m Sec.
*	Printing	$6\frac{2}{3}$ characters /Sec.
*	Punching	24 " "

* There is a slight delay in the conversion of numbers with decimal exponents larger than about 10, this being proportional to the magnitude of the exponent. For exponents of order 1000, the delay is about 2 Sec., for 2000 it is about 5 Sec.

** The results of all numerical operations are automatically normalized and tested for being within range. If the result is too large, a dynamic stop occurs (DS/O); if too small, it is replaced by zero. If all significant figures are lost, a stop occurs (SF/G).

✓✓ 1 msec. = 1/1000 sec.

$+\frac{1}{4} \leqslant \left|\text{normalized number's amplitude}\right| < +\frac{1}{2}$.

Production All production output should be punched, then printed out separately to save time on the computer.

Tape Preparation

1. The generalized floating decimal number is punched as abb...b+n+.
   If n = $\bar{0}$, it is sufficient to punch
   abb...b++
   Whilst "a" must be ≠ 0 and therefore punched, only the significant digits, b, need be punched. Any number of b-digits may be punched, but only the first 10 will be meaningful.

2. To aid in the preparation of numerical tapes, two facilities are available.

   (i) Any number of non-significant b digits (Say b = 0's) may be punched after the significant ones to permit splicing of tapes in this area.
   (ii) If an error is detected before the final sign has been punched for any one number, a ∅ may be punched and the number repunched correctly. The ∅ causes the part of the number currently being read-in to be ignored.

3. The number zero must be arbitrarily defined in floating form. It should be punched as 1 + 100 -.

4. All the tape control words and all the operational instruction words consist of 4 letters. However, only the first of these letters is interpreted, so that spelling of the word is immaterial. It may therefore be found convenient to punch SUBT, for example, as SSSS, or VOID as VVVV and so on. Both punching and tape-recognition are then simplified.

5. Characters in the addresses of the instructions which are prescribed as the decimal digit "zero" should be punched as "zero" or "stroke" on the keyboard. Care should be taken not to use the key for "capital letter O". There are times when these zeros need not necessarily be zeros, but this should never be assumed except in connection with the instructions VOID, HALT, QUIT and COKE. Hence it is easy to alter PRNT to VOID without hand-punching the entire tape.

6. Tapes should be punched once on each keyboard and compared for accuracy.

7. The tape controls ENTR and REEN are generally punched at the end of any complete TRANSCODE tape. If these are punched as EEEE //// RRRR, an easily recognizable pattern results which helps one to guard against placing the wrong end of the tape in the FERUT reader.

8. The symbols ∅ and ≹ can be used interchangeably. They each represent five holes across on the tape.

Examples

1. Form the scalar product of the Vectors $a_1 \times 10^{-b1}$, $a_2 \times 10^{-b2}$, ---, $a_{20} \times 10^{-b20}$ and $-c_1 \times 10^{d_1}$, $-c_2 \times 10^{d_2}$, --, $-c_{20} \times 10^{d_{20}}$ and add it to the constant $e\text{x}10^{f}$.
A tape to solve this problem should be punched as follows:

```
Spaces
CNST    e + f + "
Spaces
NUMB    a1 + b2 -

        a2 + b2 -


        a20 + b20 - "

Spaces
DRUM    001
Spaces
NUMB    c1 - d1 +

        c2 - d2 +


        c20 - d20 + "

Spaces
DRUM    002
Spaces
INST 010
READ     0020     0000     Y000
READ     0010     0000     X000
LOOP     0200     0006     0000
OVER     C010     0000     Z010
MULT     X206     Y206     Z020
ADDN     Z020     Z010     Z010
TRNS     0050     0006     0000
PRNT     0011     0100     Z010
HALT     0000     0000     0000
QUIT     0000     0000     0000
Spaces
ENTR
Spaces
REEN
Spaces
```

Spaces should not be punched except where expressly indicated, or between the a_i and the + sign or the c_i and the - sign.

2. Tabulate $\frac{1}{n!}$ for n = 1, 2, 3 --- to five significant decimal places, 4 entries to a line, and in blocks of 4 lines.

A programme to do this could be written as follows:

```
     INST   011
001  OVER   C01.0    -       Z01.0
002  OVER   Z01.0    -       Z02.0
003  OVER   Z01.0    -       Z03.0
004  LOOP   016.0   000.5     -
005  DIVD   Z02.0   Z03.0    Z02.0
006  OVER   Z02.0    -       X16.5
007  ADDN   Z03.0   Z01.0    Z03.0
008  TRNS   006.0   000.5     -
009  PRNT   016.4   005.0    X01.0
010  TRNS   005.0    -        -
011  QUIT    -       -        -
     ------------------------------
     CNST  1 + + "
     STOP
     ENTR
```

The method used above is quite direct. Answers are stored in sets of 16 on page X, and printed out when each set is complete. Z01 is used to retain +1. At any instant, Z02 contains $\frac{1}{n!}$ and Z03 contains (n + 1).

$\frac{1}{(n+1)!}$ is then calculated as $\left(\frac{1}{n!}\right) \div (n + 1)$.

Read-in Procedure:

All tape controls are interpreted on the warning character principle, so that the tape is input by a manual entry in the usual way with H = ZA@/. Set write-current on, print, punch, print-and-punch switch as desired, /G Stop as desired, /L Stop off. All working-track switches should be open.

ENTR should follow the input of all data and operational instructions, and hence all INST, CNST, NUMB, DRUM or FNTN controls on the tape. STOP leads to an I//G stop. The reading-in of each FNTN is terminated by an FF/G Stop. ENTR initiates the translation of the information thus far read in, into the form required by the machine. The translation of the QUIT instruction completes the internal preparation of the material, and this step is terminated by Stop (ØØ/L). A prepulse then initiates the actual calculation, leading automatically to any programmed output. The programmed HALT leads to a ///G Stop.

All magnetic transfers of data are checked, any failure giving rise to a "motor-boat" hoot Stop.

Restart Procedures:

1. Before the programme be suspected at fault, a test or development case should always be run twice, completely, from the beginning, and the two results agree exactly.

2. A production run should always be preceded by a test run for which the answers are known, to ensure that input and translation of the programme has been performed correctly by the machine.

3. Should the input and translation be known to have taken place correctly, re-start of the calculation is possible by manual entry at **REEN**.

4. Should the input be known to have taken place correctly, re-start of the translation and calculation is possible by manual entry at ENTR.

5. Should the reading-in process break down, re-start is possible at any of the tape controls except DRUM. DRUM is excepted only because it assumes data present in the electronic store.

6. Should additional or alternate data be required to be read-in after the calculation has reached a certain stage, this can be achieved by direct input of the new data tape. The calculation may then be re-started by the REEN control, provided the machine has not been used for another problem in the intervening time, and provided the same CNST's pertain.

Write-Tape Procedures

The order KOPY initiates the output punching of all translated instructions in a form suitable for re-input by means of the FNTN and REEN orders. The exact details of this output are given in the appendix. This facility is intended to be used in either of two ways:

1. To compose sub-routines: If a programme contains say more than 30 or 40 operational instructions, it may be found convenient to write portions of it in the form of sub-sections, which would be called in by the main part of the programme as FNTN's, with suitably assigned numbers, n, just as if they were library FNTN's. There are the restrictions that each sub-section prepared in this way should not be more than one FNTN in length, and should terminate with the last instruction to be obeyed before exit. The tape thus output should have the initial FNTN 101*and the final RRRR chopped off, and be incorporated into the main problem tape preceded by FNTN oon. (See appendix page iv.)

2. To prepare final copies of programmed INST's: In the event of programmes to be used repeatedly for production runs over a long period of time, it may be worthwhile to prepare a "write-taped" version of the INST part of the tape for future input. This should be attached, exactly as it comes from the machine, to the end of all sections of the tape containing data and FNTN input sequences, and replacing all INST, CNST and ENTR sequences. It has the advantage of eliminating the translation process and providing input check-sum facilities. It has the disadvantage that the write-tape is, on the average, 3 times as long as the original tape.

---

* Provision is made in the input section of TRANSCODE to store FNTN's numbered 101, 102, 103, ... in the consecutive drum locations normally used to house the translated instructions. The true FNTN's should be numbered 001, 002, ... and are stored on a separate set of drum locations. Also, one of the duties of the QUIT instruction is to arrange for the machine order NS/P to be placed immediately after the last machine order in the programme just translated.

A stop (DS/Q) terminates the entire writetape procedure. Care should be taken that this point is reached to ensure that all the translated instructions have been output.

Restrictions on a TRANSCODE Programme

A. Imposed by the Coding:

1. At most 21 CNST's.
2. At most 21 NUMB's stored in each DRUM location, and therefore to each X or Y page.
3. At most 13 Z-type working locations.
4. Only B-lines 2,3,4,5, and 6 available. If no B-line modification is required, B0 is automatically applied, and must therefore be clear. Note that PRNT alters the contents of B2 and B3.
5. At most 64 forward transfers of control.

6. Numbers input should lie between 10^{-1000} and 10^{+1000}. For numbers outside this range, read-in is slow or impossible.

   Numbers generated in the course of a calculation must lie between 2^{-216} and 2^{+216}, which is equal approximately to $10^{\pm 10,000}$. If a calculated number should be too small, it is automatically replaced by "zero"; if too large, a stop occurs (DS/0).

   Numbers output must lie between 10^{-75} and 10^{+100}. For numbers outside this range, either an incorrect answer will be output or a closed loop will occur.

7. The first instruction obeyed must have the instruction number 001. However, this can always be an unconditional TRNS.

8. Available DRUM locations are divided equally into two sets, called the lower half and the upper half of the range. Any set of DRUM locations selected by B-modification techniques must lie entirely within one of these two sets.

9. At most 100 FNTN locations per programme, with addresses 001, 002, 003, ... 100.

*B. Imposed by the Normal Track Assignments:

1. At most 256 INST's. For more than 100 INST's, however, a check should be made to see that the translated instructions do not over-run the 16 tracks assigned. This can be done by watching S3 during translation. See Appendix, p. iv.

2. At most 15 FNTN's, with addresses 001, 002, ... 015.

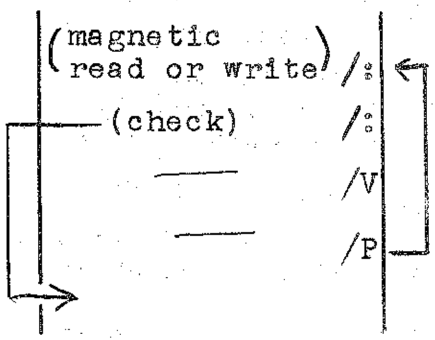
3. At most 32 DRUM locations, allowing 672 NUMB. These are divided into two sets in accordance with restriction A-8 above. The lower half set is numbered 001, 002, ... 016, and the upper half 017, 018, ... 032.

*B-type restrictions can be relieved by altering the INDICES. (Please consult librarian).

Summary of Stops

TRANSCODE

Gontrol line contents						
.11..	.....	.....	.1.11	- I//G	- STOP punched on tape.	
1.11.	1.11.	.....	.1.11	- FF/G	- at end of reading in a FNTN.	
1..1.	1.1..	.....	111.1	- DS/Q	- If in N/, a neg. non-warning character - If in FA, a pos. non-warning character	on input tape
11111	11111	.....	.1.11	- $$/L	- at end of translation after ENTR.	
..11.	...1.	.....	.1.11	- $\overset{1}{\text{N2}}$/G	- if argument for $\overset{1}{\text{2}}$QRT negative.	
.....	.....	.....	.1.11	- ///G	- HALT in executing a programme.	
1.1..	1.11.	.....	.1.11	- SF/G	- if normalizing factor ≥ 2^{39}.	
1..1.	1.1..	.....	...11	- DS/O	- if a calculated number > $2^{2^{16}}$.	
1..1.	1.1..	.....	111.1	- DS/Q	- end of write-taping via KOPY.	
11.1.	1.1..	.....	.1..1	- JS/L	- in RCS/T	



All transfers of numerical data are checked in this way, leading to a "motor-boat" hoot for a consistent magnetic transfer failure. Transfers of instructions are not checked.

Steady Hoot Stop - Check-sum failure on reading in FNTN.

Closed loop in machine operation - this could be due to the programme, or because a number outside the permitted ranges is being input or output.

Note 1. /L and /G are stops which correspond to switches on the console, and are only operative when the corresponding switch is "on". They can each be surmounted by a prepulse, when the operator is satisfied that the progress of the calculation should proceed.

Note 2. DS/Q and DS/O are dynamic stops, which cannot be surmounted. They should never occur in a correctly designed programme or correctly prepared tape, except at the end of a KOPY run.

Integers

Floating Decimal Representation

The machine is constructed to operate on numbers in fixed-point, binary form. Hence, in order that numbers punched in floating-point,decimal form may be read in, a conversion must take place. Within the machine, each floating-point, decimal number is, in fact, represented in its equivalent floating-point binary form, one long line of machine storage being used to represent the amplitude, and one short line to represent twice the binary exponent. (The factor of two in the exponent is merely for convenience in coding).

Integers as well as fractions punched in floating-decimal form may be read in by means of the tape controls NUMB or CNST. However, owing to the conversion, they are not represented with absolute accuracy when finally in the machine. Integers can be in error by an amount lying between $10^{-9}$and 10^{-12}. Arithmetical operations involving floating decimal integers and the KOMP instruction may not therefore give the desired results, unless compensation is made for these discrepancies, ie. positive integers should be corrected by adding a small quantity to them, negative integers by subtracting a small quantity from them.

B-Line Representation

Numbers in B-lines are not in floating-point form, but in the fixed form of the computer. Thus integers can be represented in B-lines with absolute accuracy.

Integers are normally intended to be picked up by B-lines via the operand β of the BSET, INCB or NEGB instructions. However, it may be desirable to set a B-line with a variable integer in the manner of a pre-set parameter, rather than have it imbedded in the programme. This can be achieved by planting it in the exponent line of a floating-decimal address (X, Y, Z or C type). In this case, it must be read in as an equivalemt floating-decimal number, using NUMB or CNST. It may then be picked up via the operand γ of the BSET or NEGB instructions. Such a representation is still exact, provided that authorized floating equivalents are used. Some tested equivalents are given on page 17. The following examples should be studied carefully before using these techniques.

Example 1. Pick up the integer 100 in B-line 3, where 100 is preset as a CNST.

```
CNST3+14+"
INST00x
BSET 0003 0000 C010 etc.
```

Example 2. Place in Z01 the sum of the n numbers in X01,X02... X0n, where n may be preset.

```
CNST(floating decimal equivalent of (2n-2))"

 INST 009
 BSET 0006 0000 C010
 BSET 0005 0000 0000
 ZERO Z010 0000 0000
┌→ADDN Z010 X015 Z010
│ NEGB 0005 0030 0000
│ INCB 0006 0010 0000
└─TRNS 0040 0006 0000
 HALT 0000 0000 0000
 QUIT 0000 0000 0000
```

Notes
B6 used as counter.
B5 used to modify addresses.
Floating numbers stored
 in reverse sequence.
B-TRNS instr. subtracts
 3 before testing.

INTEGERS

To input the integer n into the exponent line of a floating number, punch as follows, with NUMB or CNST:

n	Punch a
0	3+1-
2	1+ +
4	2+ +
6	4+ +
8	8+ +
10	1+1+
12	2+1+
14	4+1+
16	7+1+
18	2+2+
20	3+2+
22	6+2+
24	2+3+
26	3+3+
28	5+3+
30	9+3+
32	2+4+
34	4+4+
36	7+4+
38	2+5+
40	3+5+
42	6+5+
44	2+6+
46	3+6+
48	5+6+
50	9+6+
60	3+8+
70	9+9+
80	3+11+
90	9+12+
100	3+14+

Note: Only even integers may be input in this manner. They go into the correct location for pick-up by the BSET γ and NEGB γ instructions.

In general, $a = \frac{1}{4} \text{antilog}_{10} \left[n/(2 \log_2 10) \right]$,

or, approximately, a is any integer satisfying

$$\frac{10^{0.151n}}{4} < \text{integer} < \frac{10^{0.151n}}{2},$$

for n = ± 2, ±4, ±6, ...