803 AUTOCODE SPECIFICATION

MARK III

ELLIOTT

COMPUTING DIVISION

(803 A 3 & 103)

A SPECIFICATION OF

THE MARK 3 AUTOCODE FOR

THE 803

ELECTRONIC DIGITAL COMPUTER

July 1962

Copyright Reserved

Elliott Computing Division,
Elstree Way, Borehamwood, Hertfordshire, England

NCR Electronics, National Cash Register Co. Ltd.,
206-216, Marylebone Road, London, N.W.1.

803 LIBRARY PROGRAMMES A 3 & A 103)

803 AUTOCODE MARK 3

CONTENTS LIST

THE 803 AUTOCODE MARK 3

1. INTRODUCTION

1.1 General

The 803 Autocode Mark 3 is an automatic coding system intended to be used as an aid in preparing programmes for the National-Elliott 803 Computer. It has been designed primarily for the scientific user, but is also of use to commercial users.

1.2 Method

The calculation to be carried out is first stated as a sequence of instructions written in a mnemonic code which closely resembles ordinary mathematical notation. This *mnemonic programme* is then transcribed to punched tape.

The *Autocode Translator* (Tape 1) and the *Functions and Ancillaries* (Tape 2) are then placed in the 803, and the computer is used to translate the mnemonic programme and run it, that is, to carry out that particular calculation. This method of translating a programme and running it immediately is termed *Load and Go*. While the Autocode Translator and the Functions and Ancillaries are in the 803 it is permissible to translate and run as many programmes as is required, one after another.

Alternatively, it is possible to carry out the *translating* and *running* processes separately, as was necessary with earlier Autocodes. In this case the Autocode Translator outputs the translated programme on punched tape, and this translated programme together with the Functions and Ancillaries is placed in the computer whenever it is desired to run the programme. If this method is being used it is possible to specify a limited part of the computer store in which the programme is to be run (Translating onto tape for a limited part of the store).

1.3 Comparison of 803 A 3 & 803 A 103

803 A 3 is the 803 library programme of the 803 Autocode Mark 3, and is intended for use on fixed-point National-Elliott 803 machines. The 803 library programme 803 A 103 is a version of the 803 Autocode Mark 3 specially prepared to make use of the automatic floating-point unit fitted on many National-Elliott 803 computing systems, *and it can only be used on these computers*.

In regard to mnemonic instructions, 803 A 103 is entirely compatible with 803 A 3. Mnemonic tapes of programmes written for use

with the former can, in general, be both translated *and* run with the Translator and Functions and Ancillaries of the latter, and vice-versa.

However, if any machine code block in a programme written for use with 803 A 3 employs one or more of the floating-point subroutines of that autocode, such a programme cannot be used with 803 A 103. Similarly, if in a machine code block in any programme written for use with 803 A 103 there occur any automatic floating-point instructions or direct entries to the input and print subroutines of 803 A 103, that programme cannot be used under 803 A 3.

Programmers who intend that their programmes should be universally usable should avoid producing programmes with these features.

Except where otherwise indicated, this specification applies to both 803 A 3 and 803 A 103.

1.4 Compatibility with Earlier Autocodes

All programmes written for use with 803 A 2 or 803 A 102 can be translated and run with 803 A 3 or 803 A 103 respectively, if the translating and running processes are being carried out separately. When using the Load and Go method, however, there is a slight restriction on the length of programmes which can be accommodated. These programmes must be both translated and run with 803 A 3 or 803 A 103, no matter what method is being used. In other words, programmes translated by Tape 1 of earlier Autocodes (including Floating-point versions) may not be run with Tape 2 (Functions and Ancillaries tape) of 803 A 3 or 803 A 103.

Mnemonic programmes written for use with earlier Autocodes or their floating-point versions (if any) can be translated correctly by Tape 1 of 803 A 3 or its floating-point version, 803 A 103, respectively.

1.5 Programme Sheets

The programme sheets for 803 A 3 and 803 A 103 are not published in the library.

2. OPERANDS

2.1 General

An operand is either a *variable*, or a *constant*, and it may be a *floating-point quantity* or an *integer*. There are therefore four classes of operands:

(i) Floating-point variables

(ii) Floating-point constants

(iii) Integer variables

(iv) Integer constants.

There is no need for the person writing programmes in Autocode to know exactly what the terms floating-point quantity and integer imply, provided he remembers that:

(a) On fixed-point machines (i.e. when 803 A 3 is being used), floating-point quantities are positive or negative numbers which are held to an accuracy approximately corresponding to 9 significant decimal digits, and can range in magnitude from 1.73×10^{-77} to 5.8×10^{76} approximately, or be zero. Any quantity which, as a result of some arithmetic step, would have less magnitude than 1.73×10^{-77} appears as zero. Section 25.5 (a) describes the effect obtained if an attempt is made to produce a result having greater magnitude than 5.8×10^{76}.

The method of representation is described in Appendix 3.

(b) On automatic floating-point machines (i.e. when 803 A 103 is being used), floating-point quantities may range in magnitude from 4.3×10^{-78} to 5.8×10^{76} approximately, or be zero. Any quantity which, as a result of some arithmetic step, would have less magnitude than 4.3×10^{-78} appears as zero. If any attempt to produce a result having greater magnitude that 5.8×10^{76} is made, the computer will stop (see Section 25.6 (a)).

The method of representation is described in the 803 Programming Guide (Appendix 5 of the 5th edition), and elsewhere.

(c) Integers are numbers which may take on any integral value between ± 274 877 906 943.

(d) Arithmetical operations involving integers take less time to perform that those involving floating-point quantities.

(e) It is *not* permissible for floating-point quantities to enter into arithmetical operations involving integers, and vice versa.

2.2 Variables

Any letter of the alphabet can be used to represent any variable. The programmer must decide which letters shall represent floating-point variables and which shall represent integer variables throughout any one programme. It is possible to set lists of variables to their initial values (see Section 11).

Capital letters should always be used, as the teleprinter equipment used to transcribe the programme to punched tape has no lower case.

2.3 Constants

The constants which occur in the calculation should be written in the programme in normal decimal notation and will automatically be treated as floating-point constants or as integer constants according to the type of variable with which they are associated.

In instructions involving one or more integer variables, a decimal point may not appear. Constants of the form 5 and -720 may appear, 5 must be written as such, *the forms* 5. *and* 5.0 *are not permitted.*

In instructions involving one or more floating-point variables, constants of the form 5, 5.0, -720, -.0003, and 241.57 may be used. Normal notation must be adhered to : 1.25×10^{-6} should be written .00000125 (The form 1.25/-6 must *not* appear on a programme sheet).

There must not be more than 12 digits altogether. If there are 12 digits, their "integer magnitude" must be less than 274 877 906 944. (By "integer magnitude" is meant the magnitude of the number which would be seen if the decimal point were deleted).

2.4 Conventional Allocation of Letters in this Text

To reduce the amount of explanation needed, the following conventions are adopted throughout the remainder of this next.

A, B, C and D	represent	Floating-point variables
I, J, K and L	represent	Integer variables
l, m and n	represent	*Positive* integer constants
p, q and r	represent	Any integer constants
x, y and z	represent	Floating-point constants

2.5 Suffices

Positive and zero integers may be used as suffices to any type of variable. A suffix may be written in any of several forms; these involve integer constants and/or variables. To take account of the fact that teleprinter equipment has no multi-level printing facility, it is customary to write suffices on the same level as the letters which they qualify, that is to write A4, not A_4.

The table below shows the forms permitted. On the first four lines it will be noted that two or three different forms are grouped together; *these are alternative ways of specifying the same quantity.*

Floating-point Variables	Integer Variables
A or A0 or A(0)	K or K0 or K(0)
An or A(n)	Kn or K(n)
AI or A(I)	KI or K(I)
A(I+n) or A(n+I)	K(I+n) or K(n+I)
A(I-n)	K(I-n)
A(n-I)	K(n-I)
A(I±J)	K(I±J)
A(I±nJ)	K(I±nJ)
A(mI±J)	K(mI±J)
A(mI)	K(mI)
A(mI±n)	K(mI±n)
A(m±nI)	K(m±nI)

In the examples below, whenever A, B, C, D, I, J, K *or* L *appear as variables, any of the above forms may be substituted unless otherwise stated.* In this connection, variables of the forms shown on the first two lines of the above table are regarded as "having numerical suffices", those on the ~~next~~ first five lines as "having simple suffices", the remainder as "having complex suffices".

Note carefully that *suffices should never be negative.* Errors will occur if negative suffices are used.

2.6 Brackets

The combination of (and) is used to surround suffices, and for no other purpose. They are obligatory where the suffix contains more than one component. Thus A(I+4) corresponds to the mathematical notation a_{i+4}; there is no parallel in the notation of this autocode to the mathematical form (a+b)(c+d).

3. STYLE OF WRITING

Instructions are written in columns Spaces are left after words and quasi-words, but not elsewhere. These points are dealt with more fully in section 20, but to give a preliminary idea of what a simple

programme in autocode looks like, the instructions needed for a complete programme to read in values of A and B, form and print $\sqrt{A^2+B^2}$ are:

```
SETV ABC
SETF SQRT
SETR 1
1) READ A
READ B
A=A*A
B=B*B
C=A+B
C=SQRT C
LINE
PRINT C
STOP
START 1
```

4. ARITHMETIC INSTRUCTIONS

Arithmetic instructions are of the form

A= an expression involving floating-point quantities

or

I= an expression involving integers

e.g.

```
A=B+C
I=J-2
```

and mean "evaluate the expression on the right and then set the variable on the left to this value".

The range provided is

(a) Setting to a value:	A=B	I=J
(b) Addition:	A=B+C	I=J+K
(c) Subtraction:	A=B-C	I=J-K
(d) Multiplication:	A=B*C	I=J*K
(e) Division:	A=B/C	-

In which

(a) B and C may be replaced by constants, or by A, or be identical.

For example: A=2*A and A=A+A are alternative methods of doubling A.

(b) J and K may be replaced by integer constants, or by I, or be identical.

For example: I=I+1 increases the value of I by 1

(c) A - sign may follow immediately after the = sign.

For example: A=-A negates A

(d) The symbol * is used for multiplication to avoid confusion with letter X.

(e) Division is indicated by /

(f) Note that *division of integers is not permitted.*

Examples of permitted instructions:

J=3

I=-4*J

C(I+2J)=-AI-B(2J)

A(I+6)=-B(I+6)/4.275

5. FUNCTION INSTRUCTIONS

Most function instructions are of the form:

A=FUNCTION B or A=FUNCTION x

and again mean "evaluate the expression on the right and then set the variable on the left to this value". A - sign may follow the = sign. B may be identical with A, but must *not* be preceded by a - sign.

The following functions are used as above, that is, with a floating-point variable on the left and a floating-point quantity as argument:

SIN	sine	in which the unit of angle is π radians, and the angle must have less magnitude than 268 435 456 π.
COS	cosine	
TAN	tangent	
ARCTAN	arctangent	in which the angle is given in units of π radians, and lies between $\pm \pi/2$ exclusive

LOG	natural logarithm, that is, to base e	
EXP	exponential	
SQRT	square-root	(SQRT 0=0)
INT	integral part	
FRAC	fractional part	
MOD	modulus	

The integral part of a negative number is minus the integral part of its modulus, and its fractional part is such that: integral part + fractional part = whole.

For example:

INT -3.4 is -3

FRAC -3.4 is -.4

Certain function instructions are provided which involve integers. These are of the form:

I=INT A Take integral part of A, convert it to integer form, and set I to this value.

A=STAND I Take I, convert it to floating-point form, and set A to this value. (The technical term for this process is *standardisation.)*

I=MOD J Make I equal to modulus of J.

Additional functions may be added to the above range: see Appendix 4.

6. REFERENCE NUMBERS

In general, the computer performs the steps in a calculation in the sequence in which the instructions are written down, and for this reason it is not necessary to number all the instructions in a programme. However, occasions arise in which it is desirable that the computer should break sequence and "jump" to some other part of the programme. Also, it must be possible to specify which instruction is to be obeyed first, and to programme the computer to stop when it has finished its calculations.

The instructions provided to achieve the above are detailed in sections 7,8,9,10 and 12. Their form is such that it must be possible to *make reference* to one instruction in another. All instructions to which reference is so made must be given different *reference numbers.* In writing

the programme, reference numbers appear on the left, each followed by a) thus:

```
4)A=B+C
12)I=INT B
```

As a rule, if there are n reference numbers in a programme, they will be the integers 1, 2, ... n, but they need not occur in that sequence. If, as a result of some afterthought, it becomes necessary to delete one or two reference numbers it is *not* essential to close the gaps.

7. SETTING INSTRUCTIONS AND START

7.1 General

To enable the Autocode Translator to construct the translated programme correctly, it must receive certain general information before it starts translating the mnemonic instructions. Up to four *setting* instructions, which appear at the beginning of a programme, are used to do this.

In addition, a START instruction appears at the end. This has two roles. Firstly, it specifies which of the instructions of the programme is to be obeyed first. (Afterthoughts are often placed at the end, and it is quite common for the instruction which is first to be obeyed to be nearly the last one written and to have a high reference number.) Secondly, it is written after all other mnemonic instructions, to indicate to the translator that there are no more instructions to be translated.

7.2 Details of the setting and START instructions

(a) SETS

The SETS instruction specifies the letters to be used as integer variables. It may be omitted if no integer variables occur in the programme. The sequence in which the letters are written is immaterial.

After each letter one must insert, in brackets, a number indicating the highest value of any suffix attached to that letter in the programme. If, for example, JI occurs in the programme, one must take into account the highest value which I possesses at any time at which an instruction referring to JI is obeyed. If no suffices or only suffix 0 are used, no number is needed.

Thus:

```
SETS IJ(4)K
```

indicates that I or I0, J or J0, J1, J2, J3, J4, and K or K0 are or may be used in the programme.

(b) SETV

SETV specifies the letters used to represent floating-point variables in exactly the same way as SETS specifies letters used for integer variables. It may be omitted if no floating-point variables occur in the programme.

(c) SETF

SETF specifies the functions which are required in the calculation, and certain types of peripheral equipment. The abbreviation TRIG *must* be used to cover any combination of SIN, COS and TAN. The two functions MOD and STAND are always available, and need not be mentioned. The sequence in which the functions are written is immaterial. SETF may be omitted if no functions (other than STAND and MOD) are required, and if there are no instructions referring to Film Handlers or Double Paper Tape Station (see section 13 or 17 respectively.)

Thus:

SETF TRIG SQRT LOG EXP

is suitable for a calculation requiring some or all of:

standardised forms, moduli, sines, cosines, tangents, square roots, logarithms and exponentials.

(d) SETR n

This instruction must always be used: n specifies the highest reference number used in the programme.

(e) START m

This instruction must always be used: m specifies the reference number of the first instruction to be obeyed.

7.3 Positioning of Setting and START Instructions

The setting instructions are written in a group at the beginning of the mnemonic programme, with SETR last. The mnemonic instructions should follow immediately below the SETR instructions, the one on the line below the SETR necessarily having a reference number.

The START instruction is written at the end of the mnemonic programme, that is after the last instruction to be translated, which will generally be JUMP, EXIT or STOP.

8. JUMP INSTRUCTIONS

8.1 Unconditional Jumps

JUMP @n (or JUMP @I)

means "break sequence and obey next the instruction whose reference number is n (or ... is equal to the current value of I), and then proceed sequentially from there".

In the form JUMP @I, the I may ~~not~~ have an integer ~~form of~~ suffix.

8.2 Conditional Jumps

JUMP IF A=B@n (or .....@I)

JUMP UNLESS A=B@n (or .....@I)

The first of these means "If A=B, act as for JUMP @n (or I), but if A≠B, do nothing". The second has the reverse effect, that is "If A=B, do nothing, but if A≠B, act as for JUMP @n (or I)".

In the form JUMP IF (or UNLESS) A=B@I, the I may not have any form of suffix.

The condition A=B in the above examples may be replaced by any permitted arithmetic instruction form as given in section 4, any permitted function instruction form as given in section 5, or certain film instructions as given in section 13. Furthermore, the = sign may be replaced by > or < .

The following are a few examples of the vast number of different conditions that can be stipulated in JUMP IF and JUMP UNLESS instructions:

A=B	A<0	A>4.35
A=B/2.5	A<B+C	A>4*B
A=MOD B	A<SIN B	A>MOD B
I=J	I<0	I>7
I=J*3	I<MOD J	A>STAND I

A(I+2J)<-BI-C(2J)

FILM(I) SEARCHING

9. SUBROUTINE ENTRY AND EXIT

SUBR n

means "jump to the subroutine which starts with the instruction having reference number n".

SUBR I or I n n = integer

means "jump to the subroutine which starts with the instruction having reference number equal to present value of I".

EXIT

means "return to the main programme or superior subroutine at the instruction immediately after the SUBR which last caused entry to this subroutine".

Where the entry to a subroutine is by SUBR instruction, the exit must always be by EXIT, not by a jump or conditional jump.

There may be subroutines within subroutines up to six in depth. i.e. at no point may more than 6 SUBR instructions have been obeyed without a corresponding EXIT instruction.

10. STOP AND WAIT

10.1 STOP

Causes the computer to stop calculating.

There may be more than one STOP instruction in a programme which has several branches, corresponding to the end of calculation or the reaching of dead-end positions.

10.2 WAIT

This instruction is written in a programme at any point at which it is desired that the calculation should be halted until the operator has had a chance to take some special action or other. (It has the effect of stopping the programme until the setting on the N2 buttons of the keyboard has been changed from odd to even or vice versa).

11. INITIAL VALUES

SET A(n:m) = x, y, z, ... a list of (m-n+1) floating-point constants
SET I(n:m) = p, q, r, ... a list of (m-n+1) fixed-point constants

By means of these mnemonic instructions it is possible to set lists of variables to their initial values.

The number of constants in the list is only limited by the setting instructions SETV A(i) or SETS I(i), where i ⩾ m. There is no check on this limit in the translator.

Mnemonic instructions which set variables to their initial values must not be confused with Setting Instructions (see Section 7). They may occur at any point in a programme *after* the SETR instruction, but when the programme is translated they come into effect *immediately after* the Setting Instructions, i.e. before the actual calculation commences. For this reason they must not be immediately preceded by a reference number.

Example

```
SETS I(6)
SETR 1
1) .
   .
   .
SET I(1:4) = 5,6,7,8,
   .
   .
```

The effect of this instruction is to set the initial values 5, 6, 7 and 8 in I(1), I(2), I(3) and I(4) respectively before the first instruction (in this case that with reference number 1) is obeyed.

It must be understood that a programme including such orders can only be re-run correctly, without re-input, if the initial values remain unaltered.

The effect of writing a reference number before a SET instruction is that on translation the reference number will appear before the next instruction which is *not* itself a SET instruction.

Example

The following pairs of Instructions give identical translations:

```
3)SET I(0:2)=0,1,2,          SET I(0:2)=0,1,2,
  A=6                        3)A=6
```

12. VARY AND CYCLE INSTRUCTIONS

The VARY and CYCLE facilities are provided in order that the computer may be caused to carry out a set or *loop* of instructions several times. A loop is commenced with one of six different types of instruction involving the word VARY or the word CYCLE: this specifies that the loop shall be repeated several times with a particular variable, the *counting variable*, at different values. A REPEAT instruction ends the loop. *To each* VARY *or* CYCLE *instruction there must be one, and only one,* REPEAT instruction.

No reference numbers have to be given to VARY, CYCLE or REPEAT instructions unless desired. However, greater advantage can be taken of the trace facility described in section 18.2 if each REPEAT instruction is numbered.

Throughout sections 12.1 to 12.4 below, the word (LOOP) represents "any set of instructions which does not alter any of the variables mentioned in the VARY or CYCLE instruction". It is also assumed that the variables to the right of the = sign in the VARY or CYCLE instruction are not identical with that on the left. More complex cases, to which these conditions do not apply, are discussed in sections 12.6 and 12.7.

12.1 VARY Instructions

(a) VARY I=J:K:L

(LOOP)

REPEAT I

The loop is performed with I = J, J+K, J+2K, ..., J+(L-1)K, that is L times in all.

I may be of any permitted integer variable form.

J and K may be preceded by - signs, or may be replaced by integer constants. L must be positive; it may be replaced by a positive integer constant. J, K and L may not have complex suffices.

(b) VARY A=B:C:L

(LOOP)

REPEAT A

The loop is performed with A = B, B+C, B+2C, ..., B+(L-1)C, that is L times in all.

A may be of any permitted floating-point variable form.

B and C may be preceded by - signs, or may be replaced by constants.

L must be positive; it may be replaced by a positive integer constant.

B, C and L may not have complex suffices.

12.2 CYCLE Instructions, Type 1

(a) CYCLE I=J:K:L

(LOOP)

REPEAT I

The loop is performed with I = J, J+K, J+2K, ..., L that is $(\frac{L-J}{K} + 1)$ times in all. It is essential that $\frac{L-J}{K}$ is integral, and $\geqslant 0$.

I may be of any permitted integer variable form.

J, K and L may be preceded by - signs, or may be replaced by integer constants. They must not have complex suffices.

(b) CYCLE A=B:C:D

(LOOP)

REPEAT A

The loop is performed with A = B, B+C, B+2C, ..., D. $(\frac{D-B}{C}$ must be $\geqslant 0$, but provided it is $> \frac{1}{2}$ it need not be exactly integral. The value of A in the final performance is adjusted to be exactly equal to D, and the number of times the loop is performed is the nearest integer to $(\frac{D-B}{C} + 1)$. Thus:

(i) For CYCLE A=1:1:3.1
the loop is performed with A = 1, 2, 3.1

(ii) For CYCLE A=1:1:3.9
the loop is performed with A = 1, 2, 3, 3.9

(iii) For CYCLE A=1:1:3.5
the loop is performed with A = 1, 2, 3.5

A may be of any permitted floating-point variable form. B, C and D may be preceded by - signs, or may be replaced by constants. They must not have complex suffices.

12.3 Comparison of the VARY and CYCLE Type 1 Facilities

(a) With an integer counting variable

```
CYCLE I=J:K:L
     and
VARY I=J:K:n
```

are identical if (n-1)K = L-J, so the choice is a matter of convenience.

(b) With a floating-point counting variable

```
CYCLE A=B:C:D
     and
VARY A=B:C:n
```

where (n-1)C = D-B, are theoretically identical. However the practical points arising out of the use of floating-point arithmetic influence the choice. These are:

(i) The VARY version is much faster, but, unless the values of B, C and D are all exactly expressible in the binary notation used in the computer, the final value of A, which is formed by successive addition of C to B, may not be identically equal with the desired final value D.

(ii) The CYCLE version guarantees that the final value of A is identical with the desired final value D, but is performed more slowly.

12.4 CYCLE Instructions, Type 2

(a)
```
CYCLE I=p,q,r, ....
    (LOOP)
REPEAT I
```

The loop is performed with I = p, q, r, ...

There are no restrictions on the form of I.

p, q, r etc. are integer constants, and there may be any number of them. There must be no comma after the last number in the sequence.

(b)
```
CYCLE A=x,y,z, ....
    (LOOP)
REPEAT A
```

The loop is performed with A = x, y, z, ...

There are no restrictions on the form of A.

There must be no comma after the last number in the sequence.

x, y, z etc. are constants, and there may be any number of them.

12.5 Loops within Loops

CYCLE Type 1, CYCLE Type 2 and VARY loops may not occur inside each other *in any combination* to more than five in depth.

If the Load and Go facility is *not* being used, however, the above restrictions are relaxed to a certain extent, as follows:

(i) CYCLE Type 1 loops may occur inside each other to any depth.

(ii) CYCLE Type 2 and VARY loops may not occur inside each other in any combination to more than five in depth.

(iii) Provided that the restriction given under (ii) above is obeyed, CYCLE Type 1, CYCLE Type 2 and VARY loops may occur inside each other in any combination and to any depth.

12.6 Altering the Counting Variable within the Loop

If one or more instructions to alter the counting variable are included in the loop, the effects stated in 12.1, 12.2 and 12.4 will be modified as below.

(a) With VARY.

The loop is performed L times, the value of the counting variable being increased by the increment (K or C) between the end of each performance and the beginning of the next, but not after the last.

(b) With CYCLE Type 1.

(i) With an integer counting variable.
If the value of the counting variable when the REPEAT instruction is reached is exactly equal to the final value, L, cycling will cease.

Otherwise, the counting variable will be increased by the increment and the loop repeated. See note.

(ii) With a floating-point counting variable.

The conditions under which the value of a floating-point counting variable can, with safety, be altered within a CYCLE Type 1 loop, are so restrictive that it is more convenient to arrange the programme in such a way that the need is avoided than to comply with them.

See note.

Note If, due to the intervening instructions, the value of the counting variable when the REPEAT instruction is reached is never equal to the final value, cycling will continue indefinitely, unless some other provision, such as the incorporation of a JUMP IF instruction in the loop, is made.

(c) With CYCLE Type 2.

The loop is performed as many times as there are values in the CYCLE instruction, *starting* each performance with the counting variable at each of the values in turn.

12.7 Other Special Effects

By writing VARY and CYCLE Type 1 instructions and loops in which the variables to the right of the = sign are either (i) identical with the counting variable or (ii) altered within the loop, one may obtain various special effects. A few examples are given here:

(a)
```
VARY I=I:I:n
    (LOOP)
REPEAT I
```

In the first performance of the loop, I retains whatever value it possesses when the VARY instruction is reached. In subsequent performances it is increased by itself. We may therefore say, writing I_0 for the initial value of I, that the loop is performed with I = I_0, $2I_0$, $4I_0$, ... $2^{n-1}I_0$.

(b)
```
B=0
CYCLE A=1:B:16
    (LOOP)
B=B+1
REPEAT A
```

The loop is performed with A = 1, 2, 4, 7, 11, 16.

(c)
```
CYCLE I=J:K:L
    (LOOP containing an instruction altering L)
REPEAT I
```
Effect as in 12.6(b) (i)

(d) VARY A=B:C:L

(LOOP containing an instruction altering L)

REPEAT A

Cycling continues until the number of times the loop has been performed is exactly equal to the value of L at the time the REPEAT instruction is reached.

13. FILM INSTRUCTIONS

13.1 General

Many National-Elliott 803 Computing systems are fitted with a Magnetic Film Backing Store. There may be up to 4 *film handlers* associated with such a system, each handler holding one spool of film, and whenever a film is referred to in an instruction, the *handler number* of that film must be specified.

Each film consists of 4096 *blocks* of 64 *words* each, the blocks being numbered from 0 to 4095. Mnemonic Film Instructions deal solely with complete blocks of 64 words, and by means of these instructions it is possible to write the contents of 64 consecutive computer store locations onto a block of film, or to read the contents of a block of film into 64 consecutive store locations.

13.2 Method

There is a *programme block number count* for each film handler in use, this count specifying the number of the next film block which will be read from or written into when such a mnemonic instruction is given. Each Read or Write instruction increases this count by one, but neither can set it. To do this, a Search instruction must be given.

There must be a Search instruction before the first Read or Write instruction on any film.

13.3 Details

(a) FILM(I) SEARCH J(K) or FILM(I) SEARCH n

Search for block J(K) or n on Film (I), and store the new block number in the programme block count.

(b) FILM(I) TO J(K) or FILM(I) TO A(K)

Read next block on Film (I) into locations J(K) to J(K+63), or A(K) to A(K+63) in the computer store, and increase the programme block count by 1.

(c) FILM(I) FROM J(K) or FILM(I) FROM A(K)

Write the contents of computer store locations J(K) to J(K+63) or A(K) to A(K+63) onto the next block on Film(I), and increase the programme block count by 1.

(d) JUMP IF FILM(I) SEARCHING @L, or JUMP IF FILM(I) SEARCHING @n

(e) JUMP UNLESS FILM(I) SEARCHING @L,
or JUMP UNLESS FILM(I) SEARCHING @n

(f) FILM(I) BLOCK NUMBER TO J(K)

Store the programme block number count of the next block on Film (I) to be read from or written onto in location J(K). (For the information of those familiar with machine code, the block number count obtained here is set by instruction (a) and then kept up-to-date. It is *not* the result of a 75 1027 order).

NOTES

(i) In all the above FILM(I) may be replaced by FILM(n) where $1 \leqslant n \leqslant 4$ (I) cannot be suffixed. It is itself a suffix

(ii) K may be any form of suffix

(iii) L cannot be suffixed.

(iv) In the instruction FILM(I) BLOCK NUMBER TO J(K), J(K) may not be replaced by a constant.

13.4 Setting Instructions

If there are to be any mnemonic instructions referring to film in a programme, then FILM must be set as a function in the setting instruction SETF (See Section 7.2 (c)).

13.5 Write Suppression

A. Writing on film is assumed to be not required unless specifically demanded. The mnemonic instruction

FILM(I) ALLOW WRITE

must be given for each handler that it is required to write on, and this instruction sets a programme marker indicating that writing is permitted on that handler.

From then on every time a Film Write instruction is encountered a check is made that writing has been allowed for this handler.

(i) If so the state of the manual Write Permit switch on the handler is tested:

(a) If the manual Write Permit switch is on, the programme continues normally.

(b) If the manual Write Permit switch is off, the programme will wait until it is put on and then continue normally, *unless* the F2=04 button on the keyboard is kept depressed during the running of the programme.

In this case the programme will not wait, but will continue even though not actually writing on film (envisaged for testing purposes).

(ii) If writing has not been allowed the programme will stop.

B. A check will be made that the Write Permit switch is off before obeying any Read or Search instruction on a film for which writing has not been allowed. If the switch is not off, the programme will wait until it is put off before continuing. If writing has been allowed, there is no such check.

C. If, having written on a film, it is required in a later part of a programme to protect it with write suppression, the instruction

FILM(I) PREVENT WRITE

should be given. The programme will then wait until the manual Write Permit switch is off.

D. For details of checks and output pertaining to film write suppression during run-time, see Section 25.7(d).

14. INPUT INSTRUCTIONS

14.1 READ A and READ I

both mean "read the *next* number on the input tape, and set the specified variable to the value read".

In addition to its main purpose, that of reading numbers into the computer, the READ function also provides facilities by means of which the course of the programme can be controlled by *triggers* and *stops* punched on the data tape. Additionally, if it is desired that any explanatory notes should appear on the output, these can be punched on the data tape in the form of *labels*, and will be copied on to the output tape. For details, see section 21.

14.2 INPUT I

means "read the next character on the input tape, and set the specified integer variable to its numerical value".

The INPUT function is useful for data tape control of the programme, being primarily intended for reading control characters. The numerical values of characters in the Elliott Telecode are:

Character		Value	Character		Value
Letter Shift	Figure Shift		Letter Shift	Figure Shift	
blank		0	P	0	16
A	1	1	Q	(	17
B	2	2	R	)	18
C	*	3	S	3	19
D	4	4	T	?	20
E	≠	5	U	5	21
F	=	6	V	6	22
G	7	7	W	/	23
H	8	8	X	@	24
I	'	9	Y	9	25
J	,	10	Z	£	26
K	+	11	figure shift		27
L	:	12	space		28
M	-	13	carriage return		29
N	.	14	line feed		30
O	%	15	letter shift		31

15. PRINT INSTRUCTIONS

15.1 General

A PRINT instruction has the meaning "punch on the output tape the characters corresponding to the value of the specified variable in the specified mode". However, it is more convenient to speak in terms of the characters which will be printed when the output tape is interpreted: this method of description will therefore be used.

When a PRINT instruction is obeyed, a negative number is preceded by - , a positive number by sp. Non-significant leading O's are replaced by sp's. All numbers are followed by sp sp . The characters automatically appear on figure shift unless 1s has been punched under OUTPUT 31 : see section 16.5.

No page layout characters are punched except when the variable to be printed is too great for the mode of printing specified: the details of this are given overleaf.

Although a floating-point quantity may be printed to more than 9 places, only the first 9 significant decimal digits are accurate. If the form in (c) below is used, there can be no purpose in making m > 9.

If the programme is being run under 803 A 3, then any attempt to print the maximum possible positive number, approximately 1.16×10^{77} (see Appendix 5), will result in ∅ being output instead.

15.2 Detail

(a) PRINT A,m:n or PRINT A,I:J

A is printed with m or I digits before and n or J digits after the point.

There are (m + n + 4) or (I + J + 4) characters altogether.

If either (i) $|A| \geqslant 10^m$ or $|A| \geqslant 10^{11}$ or both

or (ii) $|A| \geqslant 10^I$ or $|A| \geqslant 10^{11}$ or both,

then, in either case, the character ? followed by the output corresponding to PRINT A,9/ (see below) is printed, a total of 18 characters, on a new line.

(b) PRINT A,m or PRINT A,I

A is printed with m or I digits altogether with the decimal point in the appropriate position. There are (m + 4) or (I + 4) characters altogether.

If $|A| < .5 \times 10^{-m}$ or if $|A| < .5 \times 10^{-I}$

the result appears as zero.

If either (i) $|A| \geqslant 10^m$ or $|A| \geqslant 10^{11}$ or both,

or (ii) $|A| \geqslant 10^I$ or $|A| \geqslant 10^{11}$ or both,

then, in either case, the character ? followed by the output corresponding to PRINT A,9/ (see below) is printed, a total of 18 characters, on a new line.

(c) PRINT A,m/ or PRINT A,I/

A is printed as a decimal fraction of m or I digits, followed by a (d) and a decimal exponent expressed as sp or - and a two-digit number. Altogether (m+8) or (I+8) characters are printed.

(d) PRINT A

A is printed in the same mode as the last floating-point variable printed. If no floating-point variables have yet been printed, the mode used corresponds to PRINT A,9/.

(e) PRINT I,m or PRINT I,J

I is printed as an m- or J- digit integer. A total of (m+3) or (J+3) characters is produced. m or J must not be greater than 12.

If $|I| \geqslant 10^m$ or $|I| \geqslant 10^J$ the character ? followed by the output corresponding to PRINT I,12 is printed, a total of 16 characters, on a new line.

(f) PRINT I

I is printed in the same mode as the last integer printed. If no integers have yet been printed, the instruction is obeyed as for PRINT I,4.

16. MISCELLANEOUS OUTPUT INSTRUCTIONS

16.1 TITLE

This instruction is used to cause a heading or other explanatory note to be printed.

All characters which appear on the mnemonic programme tape between the sp which follows the word TITLE and the next bl are punched on the output tape, together with the necessary initial shift and a final fs , each time the TITLE instruction is obeyed. Care should be taken, in writing a TITLE instruction, to ensure that all necessary page layout characters and the final bl are specifically indicated. See Appendix 6.

16.2 LINE

Is used when it is desired to print the next number on a new line. It causes cr lf cr to be punched.

16.3 LINES n or LINES I

Similar to LINE, but n or I lf's are punched between the two cr's.

16.4 SPACES n or SPACES I

Cause n or I sp's to be punched.

16.5 OUTPUT n or OUTPUT I

Cause the character with numerical value corresponding to (the five least significant binary digits of) the integer n or I to be punched. See table in Section 14.2. I may only have a numerical suffix.

When printed, the characters punched under OUTPUT instructions appear on figure shift, unless 1s has been punched under OUTPUT 31. If this is done, OUTPUT 27 must be incorporated before the next PRINT instruction, unless a TITLE instruction intervenes.

17. USE OF DOUBLE PAPER TAPE STATION

17.1 General

In sections 14 to 16 inclusive on Input and Output Facilities it is assumed that the computing system being used is equipped with one tape reader and one punch.

Many 803 systems, however, are equipped with a Double Paper Tape Station, that is, two tape readers and/or two punches, and in some cases with a directly connected teleprinter. The instructions given below make it possible for a programme written in Mark 3 Autocode to read in data from either or both tape readers if fitted, and to output results by means of either or both punches, or by a directly connected teleprinter, if fitted.

17.2 Setting Instructions

If it is intended to make use of double paper tape station facilities in a programme, it is necessary to set READER and/or PUNCH as a function in the setting instruction SETF (see section 7.2 (c)).

For more than one reader set READER

For more than one punch, and/or for directly connected teleprinter, set PUNCH

For more than one reader *and* for more than one punch and/or directly connected teleprinter, set either READER PUNCH or PUNCH READER

17.3 Mnemonic Instructions

(a) READER n or READER I

(where n or I has value 1 or 2)

Reader n or Reader I is set as the currently available input device.

All subsequently executed input instructions (see section 14) will be taken to refer to this device

(b) PUNCH n or PUNCH I

(where n or I has value 1, 2, 3)

Punch n or Punch I is set as the currently available output device.

All subsequently executed output instructions (see sections 15 and 16) will be taken to refer to this device.

If n or I has value 3, the teleprinter is set as the currently available output device.

(c) TELEPRINTER

The directly connected teleprinter is set as the currently available output device. All subsequently executed output instructions (see sections 15 and 16) will be taken to refer to this device.

The instructions TELEPRINTER and PUNCH 3 have an identical effect.

17.4 Notes

(a) The Translator sets Punch 1 and Reader 1 as the currently available mechanisms at the beginning of any programme which has PUNCH and/or READER set in the SETF instruction.

(b) The mnemonic programme tape(s) will always be read in from Reader 1.

(c) The translated programme will always be output on Punch 1.

(d) All error indications occurring during the running of a programme will be output on Punch 1.

18. CHECK INSTRUCTIONS AND TRACE FACILITY

18.1 CHECK A or CHECK I

These instructions provide optional print-out of intermediate results. It is advisable to include them at suitable points in all long programmes, to assist in detecting programming errors.

If the B-digit button on the keyboard is kept depressed during the running of a programme, they cause A or I to be printed *on a new line*, preceded by * and in the mode corresponding to PRINT A,9/ or PRINT I,12. If the B-digit button is not kept depressed, they are ignored.

Note, however that CHECK instructions are only included in the translated programme *if the B-digit button is kept depressed during translation*. Thus, when a programme has been tested and found correct, a new translated tape without CHECK'S can be prepared from the original mnemonic tape.

18.2 Trace facility

If a programme is not running its proper course, the trace facility can be used to determine at what point the error arises.

If 40 is set up and retained on the F2 buttons of the keyboard during the running of a programme, the reference number of each numbered instruction obeyed is printed on a new line, followed by) .

Note, however, that the trace facility is only included in the translated programme *if 40 is set up and retained on the F2 buttons during translation.*

If a habit is made of numbering every REPEAT instruction, the trace facility can be used to determine how many times each CYCLE or VARY loop is obeyed.

19. BLOCKS OF MACHINE CODE PROGRAMME

19.1 General

For the convenience of programmers who are familiar with the 803 machine code, it is possible to include blocks written in a style similar to the TI code amongst the mnemonic instructions of a programme written in autocode.

The first word of such a block must be preceded by @ which itself may be preceded by a reference number and) if desired. The last word of the block must be followed by) on a new line on its own. It is not possible to give a reference number to a word in the middle of a block.

Absolute locations 12, 13 and 14 may be used as working space.

Entry to a machine code block may be made either by a mnemonic JUMP type instruction to the reference number of its first word, or in normal sequence whereby the machine code instruction immediately following @ is obeyed after the mnemonic instruction preceding it. Similarly exit from the machine code block may be made by using function 40, 41, 42 or 43 to any instruction with a reference number, or in normal sequence from the last instruction before) to the first mnemonic instruction after it.

A machine code block may be entered by SUBR instruction by giving its first word a reference number, and ensuring that its exit leads to an EXIT instruction.

Automatic floating-point functions may, of course, only be used in machine code blocks included in programmes which are to be run under 803 A 103.

19.2 Instructions

Instructions are written in a fashion similar to that used in the TI code and may contain addresses of four types:

(a) Absolute

```
30    13    :    55    5
```

(b) Relative to the first word of the block:

```
26    13    :    30    14,
```

(c) Referring to variables having numerical suffices, suffix zero being omitted if desired:

```
24    I     :    30    A4
```

(d) Reference numbers in the mnemonic part of the programme, or at the start of other machine code blocks. These may only be used with functions 40 to 43:

```
43    3)    :    40    5)
```

Reference to variables having suffices which are not numerical may be made by using B-lined instructions. Provided such suffices are not complex, this is not at all involved. For example, to bring A(I+2) in to the accumulator, write

```
00    I     /    30    A2
```

19.3 Constants

Constants may be fixed-point integers, unscaled fixed-point fractions of magnitude less than 1, or floating-point numbers. The usual capacity restrictions apply.

Fixed-point integers must consist of + or - followed by one or more decimal digits, without a decimal point.

Fixed-point fractions must consist of +. or -. followed by one or more decimal digits.

Floating-point numbers must consist of + or - followed by one or more decimal digits and a decimal point. Further digits may follow the point if needed. The form +.125/12 is *not* permitted.

Examples:

Fixed-point	-123	+.345
Floating-point	-123.	+0.345

19.4 Floating-point Arithmetic

The functions specified in machine code blocks are obeyed directly, not interpreted. Thus 30 B : 04 A would be carried out in fixed-point fashion so that if A and B are non-zero floating-point quantities, the result would be nonsense.

The floating-point subroutines employed by the translated programme may be entered from machine code blocks. Their specifications are given in Appendix 5.

19.5 Miscellaneous Points

Zero instructions *must* be punched as 000 and zero addresses as 0. The zero word should be punched as +0 cr lf or 000 : 000 cr lf.

? cancels all characters since the last lf or @ . @ and any reference number which precedes it can only be cancelled by over-punching.

spec ⍰ = what change sign

20. PREPARING THE MNEMONIC PROGRAMME

20.1 The Written Form-Main Features

The mnemonic programme is written as a column or a number of columns of instructions, with the setting instructions at the top of the first and the START instruction at the bottom of the last.

An example of the type of programme sheet used at Elliott Brothers (London) Ltd., and a full description of its use, are given in Appendix 6. The following is an abridged description.

The REF. column is used for reference numbers. Instructions and remarks are written in the appropriate columns. The use of the FLOW column is at the discretion of the programmer. The END column is only used in special cases. Titles are written out in full detail and encircled.

In writing the instructions, a space should be left between each word or quasi-word and the letter, figure or word which follows it, i.e. after SIN, COS, TAN, ARCTAN, LOG, EXP, SQRT, INT, FRAC, MOD, STAND, TRIG, the group which specified any additional FUNCTION incorporated,

READER, PUNCH, TELEPRINTER, FILM(*I*), FILM(*n*), JUMP, IF, UNLESS, TO, FROM, SEARCH, SEARCHING, BLOCK, NUMBER, ALLOW, PREVENT, WRITE, SET, SUBR, VARY, CYCLE, REPEAT, READ, INPUT, PRINT, TITLE, LINES, SPACES, OUTPUT, CHECK, SETS, SETV, SETF, SETR, START, and *nowhere else, with the exception of instructions setting* Initial Values (see Section 11).

20.2 Punching the Mnemonic Tape-Main Features

The mnemonic programme tape should commence with a lead-in of blanks, ls (obligatory), cr lf (optional) and the instructions of the programme, each terminated by cr lf. Shifts should be punched wherever required. The translator ignores bl , cr and superfluous shifts. lf is ignored if no instruction has been read, so additional lf's may be punched if desired to spread out the print-up. Errors should be erased by overpunching the wrong characters with ls's, the last of these being followed by fs if a figure-shift character follows.

One sp must be punched after each of the words and quasi-words listed in 20.1 above, except that sp is optional after the last function specified in the SETF instruction. sp is also optional after STOP, WAIT, LINE, EXIT and TELEPRINTER. *No other sp's may occur, with the following exception:*

The constants on the right hand side of an instruction setting variables to their Initial Values (see Section 11) may be separated by spsp lf or ,

Thus the number of sp's in	Is
An arithmetic instruction	0
WAIT, STOP, LINE, EXIT or TELEPRINTER	0 or 1
JUMP IF A=B@n	2
JUMP UNLESS A=B@n	2
SETF TRIG LOG SQRT	3 or 4
Any other type of instruction, with the exception of Initial Values instructions (see above).	1

20.3 Writing and Punching-Special Points

(a) > and <

Neither > nor < exists in the Elliott telecode.

When punching, for < use $ (value 5: this is & on some teleprinters), for > use % (value 15).

(b) **Programme Name**

The name of the programme may be punched on the mnemonic tape *before* the setting instructions, if desired, so that it will appear at the top of the print-up. It should be preceded by cr lf fs : : and terminated by cr lf. The translator will ignore it: it will *not* be copied on to the translated programme tape.

If the name extends to more than one line : : must punched at the beginning of each line.

(c) **Explanatory Notes**

Explanatory notes may be punched on the mnemonic programme tape if desired, so that they will appear on the print up. It is possible to punch one short note *after* each instruction and symbol in the programme *except* SETS, SETV, SETF, SETR, the @ which precedes a machine block if it is on a line by itself, the) which terminates a machine code block, the bl which terminates a TITLE instruction, READER I (or n) and PUNCH I (or n).

To include a note on the tape, punch : : and the note itself between the last character of the instruction proper and the cr lf which usually follows it. The translator ignores all characters between : : and cr lf. There is no provision for explanatory notes which extend to more than one line.

These :'s can only be erased by overpunching.

: : may not appear *before* a mnemonic instruction, with the exception of the first setting instruction (see (b) above).

(d) **Machine code blocks**

These should be punched in the normal TI manner, but note that only the conventions mentioned in section 19 may be used, that zero instructions and addresses must be punched as 000 and 0 respectively, and that the zero word should be 000 : 000 cr lf or +0 cr lf.

(e) **Mnemonic Programmes Punched in Parts**

If it is desired to punch a mnemonic programme on more than one length of tape, the end of each length except the last should be indicated by sp cr lf immediately after the cr lf which follows a mnemonic instruction, and there should be a shift at the beginning of each new length, unless it is certain that no change of shift is required.

21. PREPARATION OF DATA TAPES

21.1 All Numbers

(a) Negative numbers should be preceded by a - sign.

(b) A + sign before positive numbers is optional.

(c) The end of a number is signified by sp sp or by cr lf.

Single spaces are ignored, so the digits of long numbers may be grouped if desired. No account is taken of sp sp or cr lf unless a decimal digit or point has been read, so additional page layout characters may be punched to provide a reasonable print format.

21.2 Floating-point Numbers

(a) May be punched in normal decimal form, with or without a decimal point, thus:

12.345 sp sp 142 cr lf
372.0 cr lf

Any number of digits up to 12 may be punched. If 12 digits are punched, their "integer magnitude" must be less than 274 877 906 944.

(b) The *argument*, that is the form in (a) above, may be followed by / or e and a decimal *exponent* p, say, implying multiplication by 10^p. Thus 12.345 can be represented by: 12.345 cr lf or 12345/-3 cr lf or .12345/2 cr lf etc.

(c) To represent the maximum possible positive number, punch $ cr lf. (It is *not* permitted to punch - $ cr lf.)

(i) On fixed-point machines (i.e. when 803 A 3 is being used) the maximum possible is approximately 1.16×10^{77}.

The distinction between the maximum permitted magnitude 5.8×10^{76} and the maximum possible magnitude 1.16×10^{77} is dealt with in Section 25.5, and more fully in Appendix 3.

(ii) On automatic floating-point machines (i.e. when 803 A 103 is being used), the maximum possible and maximum permitted values coincide (at 5.8×10^{76}).

21.3 Integers

Integers must be punched straightforwardly, without a decimal point, and must be of less magnitude than 274 877 906 944. The

character * may be used as a termination symbol instead of sp sp or cr lf.

21.4 Labels

Tapes may be labelled, that is to say they may have identifying information punched on them which does not constitute part of the data to be dealt with by the computer.

If while a READ instruction is being obeyed, an = sign is found, all following characters up to the next bl will be copied on to the output tape, but otherwise ignored. After the bl has been copied, fs will be punched and the computer will recommence the READ action; any character read before the label was reached having been erased by the = .

The label facility may be used to cause explanatory notes to be printed on the output, but this method is inferior to the use of TITLE described in section 16.1 unless the notes vary from one batch of data to another.

21.5 Triggers and Stops

The data tape may be empowered to control the programme by punching triggers and stops on it.

If the data tape is so arranged that a trigger, that is, a group of characters of the form n(where n is a positive integer, is encountered while a READ A or READ I instruction is being obeyed, the computer will jump to the instruction with reference number n. The READ A or READ I instruction will not actually be obeyed: that is, the value of A or I will remain unchanged.

If the data tape is so arranged that the character) is encountered while a READ A instruction is being obeyed, the computer will ~~stop~~. wait

21.6 Error Erasure

The character ? may be used to erase incorrectly punched data. It has the effect of cancelling all decimal digits of the number being punched, together with any minus sign, decimal point, or stroke. It does *not* cancel a wrong character (see 21.8), nor * $ = ( ) double sp or cr lf.

21.7 Wrong Characters and Overflow

The effects of wrong characters and of overflow while data is being read are given in sections 25.5 (d), 25.6 (a) and 25.7 (b).

21.8 Summary of Effects of Characters

The following table summarises the effects of all characters, other than decimal digits, encountered while a READ instruction is being obeyed. Where the effect differs according to the type of number being read, the two effects are given thus:

A: *effect while reading floating-point quantity*

I: *effect while reading integer*

Character	Effect
bl	Ignored except where it ends a label
*	A: wrong I: as for lf
$	A: maximum I: wrong
=	Erase previous characters and commence label
'	Wrong
,	Wrong
+	Ignored
:	Wrong
-	Negates a number
.	A: decimal point I: wrong
%	Wrong
(	Trigger to specified instruction
)	Stop
?	Erase previous characters
/	A: exponent follows I: wrong
@	Wrong
£	Wrong
fs	Noted
sp	Single sp: ignored Double sp: as for lf
cr	Ignored
lf	End of number if a decimal digit has been read, otherwise ignored.
ls	Noted
A to Z	Wrong

22. STORAGE LIMITATIONS

22.1 General

When a programme is being translated and run the Autocode Translator and the Functions and Ancillaries are always in the store, along with that programme. There must also be room for each of the variables and constants which are required for the calculation. The capacity of the store of the computer is limited, and this imposes limitations by which the lengths of programmes and/or amounts of data which can be handled at one time are restricted.

If a programme is *not* being translated and run by the Load and Go method, however, these limitations are not so severe, for the following reasons. When such a programme is being translated the Autocode Translator only and certain data relating to the programme must be held in the computer's store. During the running stage, the translated programme must be held in the store along with the necessary Functions and Ancillaries, and there must also be room for each of the variables and constants which are required.

22.2 Applicability of 803A 3 and 803 A 103 with Reference to Store Size

The capacity of the computer's store is measured in *words*. Each word occupies a *location*, the locations in the store being numbered from 0 upwards. Different 803's have stores of different sizes: the standard size has 4096 locations numbered 0 to 4095. The other sizes have 1024 and 8192 locations respectively.

Both 803 A 3 and 803 A 103 are intended for use in machines fitted with 4096-word or 8192-word stores, and they automatically adjust themselves to make use of the storage available. *Neither 803 A 3 nor 803 A 103 can be used in 803's with 1024-word stores.*

When the Load and Go method is not being used it is possible to run a translated programme on a machine with a different size of store from that in which the programme was translated. In other words, translated tapes produced on machines with 4096-word stores can be run on machines with 8192-word stores, and programmes requiring storage capacity *of up to and including 4096 words* can be translated on machines with 8192-word stores and then run on machines with 4096-word stores.

22.3 Translating for a Limited Part of the Store

By means of this facility it is possible to specify a limited part of the store in which the programme is to be run.

Control Tape

If it is intended to make use of this facility a *control tape* must be prepared. This tape specifies the upper limit of the computer store in which the programme is to be run, and the desired lower limit.

This control tape should be punched thus:

Either at the leading end of the mnemonic programme tape (or the first length of the mnemonic programme tape) and separated from the programme by some blanks,

```
+X
+Y
bls
SETV
 .
 .
 .
```

Or on a separate tape with a closed bracket immediately after the last line-feed,

```
+X
+Y
 )
```

where, in both cases,

(i) X is the required lower limit, $X \geqslant 17$

(ii) Y = 4096 or 8192

Both X and Y must always be set

22.4 Capacity Calculation

To ascertain whether a long programme or one which uses a lot of data at one time can be run, one may

either (a) Endeavour to translate it and observe the effect achieved. See Sections 24, 25.1 (c) and 25.2 (c).

or (b) Add together the number of locations which the programme's instructions, constants, variables, functions, etc., *will* occupy in the store (see (i) next page and compare this sum with the number of locations *allocated* to the programme in the computer store. (see (ii) overleaf.

N.B. This method is only approximate.

(i) The following are the numbers to be added together:-

For		Locations (803 A 3)	Locations (803 A 103)
Each mnemonic instruction, an average of		2½	2
Each different variable specified in SETS AND SETV		1	1
Each different floating-point constant and each different integer constant		1	1
If the largest reference number is n		$\frac{n}{2}$	$\frac{n}{2}$
(But if trace facility is required)		$\frac{7n}{2}$	$\frac{7n}{2}$
Functions specified in SETF (if Load and Go method is *not* being used):-			
	TRIG	67	66
	ARCTAN	59	39
	EXP	50	53
	LOG	34	36
	SQRT	18	25
	INT	39	39
	FRAC	26	22
	READER/PUNCH/ TELEPRINTER	56	53
	FILM	80	80
Each word in machine code block		1	1
Every 6 non-shift characters specified in a TITLE instruction		1	1

Note

It must again be emphasised that when the Load and Go method is being used the numbers given against SETF functions above *should be ignored.*

(ii) The number of locations which a programme may occupy in the computer store is determined by three factors:-

A. Whether or not the programme is being run using Load and Go.

B. Which version of the Autocode (803 A 3 or 803 A 103) is being used.

C. The size of the store.

The following table lists the appropriate numbers to be compared with the sum calculated from (b) (i). The numbers in italics are for 8192-word store machines.

Method of Use	803 A 3	803 A 103
Load and Go	800 *4896*	940 *5068*
Translate and Run Separately	3396 *7492*	3546 *7642*

Notes (1) The numbers given against Translate and Run Separately will of course be altered if it is intended to Translate onto Tape for a Limited Part of the Store.

(2) If using Load and Go and *not* using Film, a modification is available, self-triggering and in checked binary at the end of Tape 1, which permits the translated programme to overwrite a section of the Translator. This makes approximately an extra 210 locations available for programme, variables, etc. If this facility is used, the appropriate Tape 1 will have to be re-read into the store before input of any subsequent programme using Film.

23. OPERATING INSTRUCTIONS

It is assumed in this and subsequent sections that the reader knows how to operate the computer. Except where otherwise indicated, every reference from here on to reader or punch is a reference to Reader 1 and Punch 1.

23.1 Load and Go

(a) Place the appropriate Tape 1, the Autocode Translator, in the tape reader.

(b) Set up 40 0 on the F1 N1 buttons of the keyboard and cause this to be read in.

The computer will now read in Tape 1. Input is sum-checked, and the continuous output of sp's indicates sum-check failure.

(c) If it is intended to make use of the extra 210 locations available when not using Film (see 22.4 (b) Note 2) proceed thus:

(i) When the computer has read in Tape 1, the trailing end of the tape will not have gone through the reader. This is the modification which must be read.

(ii) To do this, set up 40 0 on the F1 N1 buttons of the keyboard and cause this to be read in.

The computer will now read in the modifications.

(d) Place the appropriate Tape 2, the Functions and Ancillaries Tape, in the tape reader.

(e) Set up 40 0 on the F1 N1 buttons and cause this to be read in.

The computer will now read in Tape 2.

Input is sum-checked, and the continuous output of sp's indicates sum-check failure.

(f) Place the mnemonic tape in the tape reader.

(g) Set up 40 7 on the F1 N1 buttons

Set up 40 on the F2 buttons if Trace facility is required.

Depress the B-digit button if CHECK's are to be translated.

Cause these keyboard settings to be read in.
The computer will now read in the mnemonic tape and translate it into the store.

(h) If the menemonic programme has been punched in parts, as described in section 20.3 (e), the Computer will stop when the sp is reached.

Place the next length of mnemonic tape in the reader.
Either (i) set up 40 17 on the F1 N1 buttons and cause this to be read in

or (ii) change the sign of the keyboard word, that is, change the setting on the F1 buttons from 40 (negative) to 00 (positive), or vice-versa.

The computer will now read in the length of tape.
The above sequence should be followed whenever the Computer reaches the end of a length of tape, with the exception of the last length.

N.B. The above mentioned re-entry point 40 17 cannot be used after the START instruction has been translated.

(i) Translating is complete when the limits of the programme are output on tape.

(j) Place the data tape, if any, in the appropriate tape reader.

(k) Either (i) To cause the calculation to start at the instruction specified in the START instruction, set up 40 16 on the F1 N1 buttons, ensuring that the N2 buttons are clear

Or (ii) To cause the calculation to start at the instruction with reference number n, set up 40 16 on the F1 N1 buttons, and n on the N2 buttons.

(l) Set up 40 on the F2 buttons if Trace facility is required. Depress the B-digit button if CHECK'S are to be translated.

(m) Cause the keyboard setting(s) specified under (k) (and (l)) to be read in.

(n) If the programme is being run under Tape 2 of 803 A 3 and it is thought that floating-point overflow may occur, ensure that the setting 40 is retained on the F1 buttons. See section 25.5 (a).

(o) Each time the programme reaches any WAIT instruction it contains, it will stop until the setting on the N2 buttons is changed from odd to even or vice versa.

(p) If it is desired to translate and run another programme by the Load and Go method, proceed as from (f).

23.2 Translation Stage Only

The following paragraphs describe the operating procedure necessary at the translation stage if the Load and Go facility is not being used:

(a) If it is not intended to translate onto tape for a limited part of the store (see Section 22.3), proceed thus:

(i) Place the appropriate Tape 1, the Autocode Translator, in the tape reader.

(ii) Set up 40 0 on the F1 N1 buttons of the keyboard and cause this to be read in. The computer will now read in Tape 1. Input is sum-checked, and the continuous output of sp's indicates sum-check failure.

(iii) Place the mnemonic tape in the tape reader and ensure that there is plenty of tape in the output punch.

(iv) Set up 40 5 on the F1 N1 buttons of the keyboard. Set up 40 on the F2 buttons if Trace facility is required.
Depress the B-digit button if CHECK'S are to be translated.
Cause the keyboard setting(s) to be read in.
The computer will now read in the mnemonic tape and begin to output the translated programme. This translated programme tape is described in Section 24.

(v) If the mnemonic programme has been punched in parts, as described in section 20.3(e), the Computer will stop when the sp is reached.

Place the next length of mnemonic tape in the reader, set up 40 17 on the F1 N1 buttons (or change the sign of the F1 buttons) and cause this to be read in.

The Computer will now read in the length of tape.

The above sequence should be followed whenever the Computer reaches the end of a length of tape, with the exception of the last length.

N.B. The above mentioned re-entry point 40 17 cannot be used after the START instruction has been translated.

(b) If translating onto Tape for a Limited Part of the Store, and if the control tape is punched on the beginning of the mnemonic programme tape, proceed as detailed under (a), except that the setting on the keyboard F1 N1 buttons as specified under (iv) should be 40 6 *not* 40 5.

(c) If Translating onto Tape for a Limited Part of the Store, and if the control tape is separate from the mnemonic tape, proceed thus:-

(i) Place the appropriate Tape 1, the Autocode Translator, in the tape reader.

(ii) Set up 40 0 on the F1 N1 buttons of the keyboard and cause this to be read in.

The computer will now read in Tape 1. Input is sum-checked and the continuous output of sp's indicates sum-check failure.

(iii) Place the control tape in the tape reader and ensure that there is plenty of tape in the output punch.

(iv) Set up 40 6 on the F1 N1 buttons.

Set up 40 on the F2 buttons if Trace facility is required.

Depress the B-digit button if CHECK'S are to be translated.

Cause the keyboard setting(s) to be read in.

The computer will now read in the control tape, output a closed bracket and stop. (If a closed bracket is not output then it has come to an error stop and the control tape is incorrect).

(v) Place the mnemonic programme tape in the tape reader.

(vi) Set up 44 6 on the F1 N1 buttons and cause this to be read in.

The computer will now read in the mnemonic tape and begin to output the translated programme. This translated programme tape is described in Section 24.

(vii) Now proceed sequentially from (a) (v)

(d) It should be noted that it is not necessary to continually re-read the appropriate Tape 1 into the computer if it is desired to translate one mnemonic programme after another.

On the other hand, if it is desired to change the entry point for the mnemonic programme from 40 5 (see 23.2(a)) or 40 6 (see (see 23.2,(b) and (c)) to 40 7 (see 23.1,(g)), or vice versa, the Translator must be re-read into the computer. If this is not done the computer will come to a dynamic stop.

23.3 Running Stage Only

The following paragraphs describe the operating procedure necessary at the *running* stage if the Load and Go facility is *not* being used.

(a) Place the translated programme tape in the tape reader.

(b) Set up 40 0 on the F1 N1 buttons of the keyboard and cause this to be read in. The computer will now read in the programme tape. Continuous output of sp's at any point indicates sum-check failure or a control character incorrectly punched on the tape.

(c) Place the appropriate Tape 2, the Functions and Ancillaries tape, in the tape reader.

(d) Set up 40 0 on the F1 N1 buttons and cause this to be read in.

The computer will now read in Tape 2. Input is sum-checked, and the continuous output of sp's indicates sum-check failure, in which case start again at (a).

(e) Now proceed sequentially from 23.1,(j).

24. THE TRANSLATED PROGRAMME

The translated programme tape is punched in a special binary code.

Two extra groups of characters are punched at the end of the tape, separated by several blanks from the) which marks the end of the programme. These are of the form 16 - P and Q - R and indicate (provided the programme is to be run on a computer with the same size of store as that in which it has been translated) that locations 16 to P will be used for the main programme, locations Q to R for constants, variables, function subroutines and the reference number decode block, and locations R+1 to the end of the store for the ancillary subroutines of the system.

If an overlong programme is translated to its end, as suggested in section 25.2,(c), P will not be less than Q. The excess is, in fact, (P - Q + 1) words.

A programme which has been translated in an 8192-word machine will work in a 4096-word machine if and only if Q - P > 4096.

25. ERROR INDICATIONS AND ACTION TO BE TAKEN

25.1 Error Stops during Translation

The error stops described below are those which may occur while the operations specified under 23.1 (a) to (i), and 23.2 are being carried out.

The Translator will stop

(a) If there is an obvious error in the setting instructions.

(b) If there is an obvious error in any other part of the mnemonic programme.

(c) If the translated programme is becoming so long that there will not be room for it and its data in the computer's store at the running stage. In this case, the translator punches) before it stops.

Examples of (a)

(i) Word or quasi-word spelt wrongly.

(ii) Unrecognised word used.

(iii) Same letter set twice.

(iv) Omitting SETR.

Examples of (b)

(i) Using a letter or FUNCTION which has not been set (N.B. Use of A8 where only A(7) has been set, and use of AI when I < 0 will *not* be detected by the Translator, and errors will arise).

(ii) Using a constant with excessive "integer magnitude".

(iii) Using integers and floating-point quantities in wrong combinations.

(iv) Using an integer variable on either side of a FUNCTION instruction not applicable to integers.

(v) Using an integer variable in an instruction to divide.

(vi) Using a constant on the left-hand side of any instruction.

(vii) Word or quasi-word spelt wrongly, or unrecognised word used.

(viii) Giving an instruction a reference number higher than the maximum specified in the SETR instruction, or giving the same reference number twice.

(ix) Using VARY, CYCLE Type 1 and CYCLE Type 2 loops to excessive depths.

25.2 Action to be Taken on an Error Stop During Translation

(a) If the translator stops because of an error in the setting instructions the tape should be removed, a corrected tape punched, and translation recommenced *from the beginning*. Use may be made of the method described in 20.3 (e) and 23.1 (h), but the whole of the setting instructions must be repunched.

(b) If the translator stops because of an obvious error in the body of the programme, the following courses are open:

(i) Take the tape away and correct it. This is not very profitable, as there may be more errors on it.

or (ii) Mark the tape at the point at which the error was found.

Set up 04 on the F2 buttons of the keyboard and change the sign of the keyboard word, that is, change the setting on the F1 buttons from 40 (negative) to 00 (positive), or vice-versa. The translator will 'read over' the rest of the mnemonic tape and check it for obvious errors but no further output will take place. It will stop each time it finds an error, whereupon the tape should be marked, and checking recommenced by again changing the sign of the keyboard word. Then take the tape away and correct all errors.

or (iii) Mark the tape at the point at which the error occurred and remove it from the tape reader. Punch a correction tape consisting of:

blanks
a shift
the correct version of the instruction
cr lf
a shift, if necessary, to ensure that the first character of the next instruction on the main tape will be read correctly.
sp cr lf
blanks

Place this in the tape reader, and cause it to be read by changing the sign of the keyboard word (see (ii) above). Then put the main mnemonic tape back in the tape reader, with the first character of the next instruction in the reading position, and again change the sign of the keyboard word.

(c) If the translated programme is becoming so long that there will not be room for it and its data in the computer's store, the translator punches) and stops. If it is desired to continue translation, in order to determine the exact amount by which store capacity is exceeded, change the sign of the keyboard word. See section 24.

25.3 Insertion of Additional Instructions

Additional instructions may be inserted or substituted for existing instructions on the mnemonic tape, during the translation stage, as follows:

(a) Punch the insertion tape:

blanks

a shift

the instructions to be inserted, each followed by cr lf

a shift, if necessary, to ensure that the next character on the main tape to be read after the insertion will be read correctly

sp cr lf

blanks

(b) Mark the mnemonic tape at the points at which it is to be interrupted and restarted: these must coincide with the ends of instructions.

(c) Commence translation in the normal way, but with the N2 buttons on the keyboard set at 0.17.

(d) As the marked part of the mnemonic tape approaches the tape reader, depress the selected stop button, and repeatedly depress the operate bar until the marked point is reached.

(e) Remove the main tape from the tape reader, and replace it with the insertion tape. Clear the selected stop button and depress the operate bar.

(f) When the insertion tape has been translated, remove it, put the main tape back in the tape reader and then change the sign of the keyboard word to cause translation to recommence.

25.4 Fast Method Detecting Errors on the Mnemonic Tape

If the translation process is being carried out separately, quite a lot of time can be wasted in attempting to translate mnemonic tapes in which there are errors. Some users avoid this possibility by always making the translator "read over" a mnemonic tape before actually attempting to translate it. This is done by operating as in 23.2 but with the F2 buttons set at 04 . No output takes place, but the translator rapidly "reads over" the mnemonic tape, stopping if any obvious error is found. The actions taken on a stop are similar to those stated in 25.2,(a) and 25.2 (b)(ii).

25.5 Error Indications During Running, for 803 A 3 only

The error indications described in this and subsequent chapters of this section are those which may occur while the operations specified under 23.1,(j) to (o), and 23.3 are being carried out.

(a) If the correct result of any operation on a floating-point variable has magnitude greater than the *permitted* maximum, approximately 5.8×10^{76}, the answer is set to the *maximum possible positive value*, $+1.16 \times 10^{77}$, irrespective of the sign of the correct result. The character @ is printed, and, if and only if the keyboard word is negative, the programme will stop.

(b) If the argument of either of the functions EXP or TAN is such that the function overflows, the character @ is printed, and, if and only if the keyboard word is negative, the programme will stop.

(c) If the argument of an I=INT A instruction is too large, the character 4 is output continuously.

(d) While a READ instruction is being obeyed:

(i) If the "integer magnitude" of any number read exceeds 274 877 906 943, the programme stops and outputs lf's continuously.

(ii) If a floating-point number in excess of capacity is read, to which (i) does not apply, action is as in (a) above.

25.6 Error Indications During Running, for 803 A 103 Only

(a) If the correct result of any operation on a floating-point variable has magnitude greater than 5.8×10^{76}, the computer will stop, with the 'floating-point overflow' lamp on the keyboard lit. The actual result generated by the computer will be meaningless. The computer can be restarted by depressing the operate bar.

It should be observed that the division of (floating-point) zero by zero will induce floating-point overflow. (This does not occur when 803 A 3 is being used).

(b) If, while a READ instruction is being obeyed, a number whose "integer magnitude" is greater than 274 877 906 944 is read, then the programme stops and outputs lf's continuously.

(c) If the argument of the function EXP is greater than 254 $\log_e 2$ the character 3 is output continuously.

(d) If the argument of an I=INT A instruction is too large for fixed-point representation, the computer outputs D and stops.

25.7 Error Indications During Running, For both 803 A 3 and 803 A 103

(a) If a FUNCTION type instruction has an impossible or useless argument, the indication is as given below.

Function	Argument	Character Printed	Further Effect
SIN COS TAN	Greater than 2^{28}, so that no fractional part exists	1	Continuous output
LOG	Negative or Zero	2	Continuous output
SQRT	Negative	5	Continuous output
READER I/n	I/n > 2	?	Continuous output
PUNCH I/n	I/n > 3	?	Continuous output

(b) If, while a READ instruction is being obeyed, a "wrong" character is found on the data tape, the programme stops and outputs sp's continuously.

(c) If subroutines within subroutines using SUBR and EXIT exist to a greater depth than 6 in the mnemonic programme, the translator is not able to detect the excess, and will therefore produce a corrupt translated programme. However, when the corrupt point is reached, the translated programme will output $ continuously.

(d) **Details of checks and output pertaining to film write suppression during run-time**

(i) If H *I* 6666..... is output, where I is an integer, and the machine has reached a dynamic stop, then a film write instruction is being attempted on a handler for which no FILM(I) ALLOW WRITE instruction has been obeyed.

This is an error in programming.

(ii) If H *I* 7777..... is output, where I is an integer, then write suppression is required on film handler I. i.e. either a FILM(I) PREVENT WRITE instruction has been given or a read/search instruction, on a handler for which there has been no FILM(I) ALLOW WRITE instruction given, is being attempted.

Until further action is taken the machine will cycle in a loop which is testing the actual state of the write permit switch on film handler I. To continue the running of the programme the write permit switch on film handler I must be switched *off*.

(iii) If H *I* 8888..... is output, where I is an integer, then write permit is required on film handler I. i.e. either a FILM(I) ALLOW WRITE instruction or a write instruction is being attempted.

Until further action is taken the machine will cycle in a loop which is testing the actual state of the write permit switch on film handler I.

To continue the running of the programme the write permit switch on film handler I must be switched *on*.

(iv) If H *I* 9999..... is output, where I is an integer, then film handler I is not available for use by the computer. i.e. handler switched off, or handler on manual.

Until further action is taken the machine will cycle in a loop which is testing the actual state of the film handler I. To continue the running of the programme the film handler must be made available to the computer by manual intervention.

(v) After a read/write instruction the film control word is checked for parity errors.

If there is a parity error a £ sign is output and the last block read/written is re-read/re-written. The programme will not continue until the block has been correctly read/written. There is no check on block address parity after a Search instruction.

(vi) If H I : : : : .... is output, where I is an integer, and the machine has reached a dynamic stop, then there is a wrong block number. (wrong in so far as it is not what the programme expected). The block number of the last block read/written is checked against the block number held in the block number count. The machine has reached a dynamic stop.

F.M. Mitchell and S.E. Gilbert *July, 1962.*

APPENDIX 1 SUMMARY OF INSTRUCTIONS

Note. In the examples below,

A, B, C and D	represent	Floating-point variables
I, J, K and L	represent	Integer variables
l, m and n	represent	Positive integer constants
p, q and r	represent	Any integer constants
x, y and z	represent	Floating-point constants

Any variable except the one before the = sign and that used in certain Film instructions may be replaced by a constant.

ARITHMETIC	A=B	A=-B	I=J	I=-J
(Section 4)	A=B+C	A=-B+C	I=J+K	I=-J+K
	A=B-C	A=-B-C	I=J-K	I=-J-K
	A=B*C	A=-B*C	I=J*K	I=-J*K
	A=B/C	A=-B/C	-	-

FUNCTION	A=SIN B	
(Section 5)	A=COS B	
	A=TAN B	
	A=ARCTAN B	
	A=LOG B	
	A=EXP B	
	A=SQRT B	
	A=INT B	I=INT A
	A=FRAC B	I=MOD J
	A=MOD B	
	A=STAND I	

APPENDIX 1 - CONTINUED

SETTING AND START

(Section 7)

SETS (Integer variables)
SETV (Floating-point variables)
SETF (Functions and peripheral equipment)
SETR n (Maximum reference number)
START n
In SETF (i) TRIG covers SIN, COS and TAN.
(ii) MOD and STAND need not be mentioned.

JUMP

(Section 8)

JUMP @K
JUMP IF A=B@K JUMP IF I=J@K
JUMP UNLESS A=B@K JUMP UNLESS I=J@K
(K may not have any form of suffix)
Any permitted arithmetical instruction or function instruction and certain Film instructions may be substituted for A=B or I=J, and may have > (%) or < (¢) in place of =

OTHER CONTROLS

(Sections 9 and 10)

SUBR n
EXIT
STOP
WAIT

INITIAL VALUES

(Section 11)

SET A(n:m) = x,y,z, ...
SET I(n:m) = p,q,r, ...

APPENDIX 1 - CONTINUED

VARY AND CYCLE

(Section 12)

VARY A=B:C:L
CYCLE A=B:C:D
CYCLE A=x,y,z,...
REPEAT A

VARY I=J:K:L
CYCLE I=J:K:L
CYCLE I=p,q,r,...
REPEAT I

(B, C, D, J, K, and L may have simple suffices only)

FILM

(Section 13)

FILM(I) TO J(K)
FILM(I) FROM J(K)
FILM(I) BLOCK NUMBER TO J(K)
(In the above five examples J and A may not be replaced by a constant).
FILM(I) SEARCH J(K)
JUMP IF FILM(I) SEARCHING @L
JUMP UNLESS FILM(I) SEARCHING @L
FILM(I) ALLOW WRITE
FILM(I) PREVENT WRITE
(In all the above examples, (I) and L may not have any form of suffix. (I) is itself a suffix to FILM, and may only be replaced by a bracketed constant, (n).

FILM(I) TO A(K)
FILM(I) FROM A(K)

INPUT

(Section 14)

READ A
READ I
INPUT I

APPENDIX 1 - CONTINUED

OUTPUT

(Sections 15 and 16)

PRINT A,n:m
PRINT A,n
PRINT A,n/
PRINT A
PRINT I,n
PRINT I
OUTPUT I
(In OUTPUT I, I may have a numerical suffix only)
LINE
LINES I
SPACES I
TITLE

PRINT A,I:J
PRINT A,I
PRINT A,I/
PRINT I,J

DOUBLE PAPER TAPE STATION

(Section 17)

READER I

PUNCH I
TELEPRINTER

CHECK

(Section 18)

CHECK A

CHECK I

APPENDIX 2 SUMMARY OF OPERATING INSTRUCTIONS AND ERROR INDICATIONS

A. LOAD AND GO

1. Operation

(a) To read in appropriate Tape 1, enter **40 0**
Input is sum-checked.

(b) To read in modification when not using Film, enter **40 0**

(c) To read in appropriate Tape 2, enter **40 0**
Input is sum-checked.

(d) To read in mnemonic tape, enter **40 7** with :

B = 1 to translate CHECK'S

F2 = 40 to Incorporate TRACE facility

(e) If the mnemonic programme is punched in parts, to recommence translation enter **40 17** or change the sign of the keyboard word.

(f) To enter the programme at the point specified in the START instructions, enter **40 16 with keyboard N2 = 0.** To enter at the instruction with reference number n, enter **40 16 with keyboard N2 = n.** In either case keep the keyboard set with :

B = 1 for CHECK'S to be obeyed.

F2 = 40 for TRACE facility to be used.

F1 = 40 if floating-point overflow is likely (this applies to 803 A 3 only).

(g) If there are any WAIT instructions in the programme, change N2 from odd to even, or vice versa, when ready for calculation to continue.

2. Error Stops During Translation and Action to be taken

If the translator stops without completing the translation, including the output of the details of storage space required by the translated programme, then :

(a) If the only instructions read so far are setting instructions, remove the tape and correct it.

(b) If (a) does not apply, and the last character punched on the translated tape is *not*) then:

APPENDIX 2 - CONTINUED

EITHER (i) Mark the tape

(ii) Set F2 = 04 on the keyboard

(iii) Change the sign of the keyboard word, that is, change F1 from 40 to 00 or vice versa.

(iv) Repeat (i) and (iii) each time the translator stops until the end of the tape is reached. Then remove the tape, correct it at all marked points, and recommence.

OR (i) Mark the tape and remove it.

(ii) Punch a correction tape comprising :

blanks shift correct instruction cr lf shift sp cr lf blanks

(iii) Change the sign of the keyboard word to read correction tape.

(iv) Change the sign of the keyboard word to continue reading main tape at beginning of next instruction.

(c) If the last character punched is) the programme is overlong. See section 25.2(c) for recommended action, and refer to sections 22.4 and 24.

3. Error Indications During Running

Character(s)	Other Action	Error Indicated
@	Stop if keyboard F1 = 40	Floating point overflow has occurred in arithmetic operation, READ, TAN or EXP *(803 A 3 only).*
Computer stops with 'floating-point overflow' lamp lit.		Floating-point overflow *(803 A 103 only).*
1	Continuous output	SIN, COS or TAN with argument too great.
2	Continuous output	LOG with negative or zero argument.
3	Continuous output	EXP with argument greater than 254 $\log_e 2$ *(803 A 103 only)*

APPENDIX 2 - CONTINUED

Character(s)	Other Action	Error Indicated
4	Continuous output	I = INT A with argument too great *(803 A 3 only)*
5	Continuous output	SQRT with negative argument
D	Computer stops	I = INT A with argument too large for fixed-point representation *(803 A 103 only)*
ß	Continuous output	SUBR's to excessive depth.
sp	Continuous output	Wrong character on data tape
lf	Continuous output	Number read has excessive "integer magnitude"
?	Continuous output	READER I/n where I/n > 2, or PUNCH I/n where I/n > 3
cr lf cr ?	Programme continues	Number too great for mode specified in PRINT.
H I 66 ...	Continuous output of 6	No FILM(I) ALLOW WRITE before film write instruction on handler (I) i.e. error in programming.
H I 77 ...	Continuous output of 7	FILM(I) PREVENT WRITE, or *no* FILM(I) ALLOW WRITE before a read or search instruction. Switch off write permit switch on handler(I).
H I 88 ...	Continuous output of 8	Write permit required on handler(I). Rectify this.
H I 99 ...	Continuous output of 9	Film handler(I) not available. Rectify this.
£	Last film block read/written is re-read/written.	Parity error on film.
H I : : ..	Continuous output of :	Wrong block number on handler (I).

APPENDIX 2 - CONTINUED

B. TRANSLATION STAGE ONLY

1. Operation

(a) To read in appropriate Tape 1, enter **40 0**.

Input is sum-checked.

Do NOT read in the modification at the end of the tape.

(b) (i) If it is *not* intended to translate onto tape for a limited part of the store, to translate mnemonic tape, enter **40 5**.

(ii) If it *is* intended to translate for a limited part of the store, and if control tape is punched on the beginning of mnemonic tape, then to translate mnemonic tape enter **40 6**.

(iii) If translating for a limited part of the store and control tape is separate from mnemonic tape, to read in control tape enter **40 6**. To read in mnemonic tape enter **44 6**.

In all cases enter with :

B = 1 to translate CHECKS

F2 = 40 to incorporate TRACE facility.

(c) If the mnemonic programme is punched in parts, to recommence translation enter **40 17**, or change sign of the keyboard word.

2. For a summary of Error Stops during Translation and Action to be taken, refer to **A.2** above.

C. RUNNING STAGE ONLY

1. Operation

(a) To read in translated tape, enter **40 0**

Input is sum-checked.

(b) To read in appropriate Tape 2, enter **40 0**

If sum-check failure occurs, start again at (a).

(c) Now proceed sequentially from **A.1** (f).

2. For a summary of Error-Indications during Running, refer to **A.3** above.

APPENDIX 3 FLOATING-POINT REPRESENTATION

(a) General

Any number A can be represented in many ways by a pair of numbers (a, b) which satisfies the equality

$$A = a \times 2^b \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \qquad (1)$$

In which a is called the mantissa or argument and b the (binary) exponent.

For example, 6 can be represented by (6, 0) or by (.75,3) or by (12,-1).

(b) Standardised Floating-Point Representation within 803's as used in 803 A 3.

In most programmes by means of which floating-point arithmetic is carried out in 803's not fitted with automatic floating-point units, the form used is such that b is a positive or negative integer and

$$\left.\begin{array}{l} \text{if } A > 0 \text{ , } \tfrac{1}{2} \leqslant a < 1 \\ \text{if } A = 0 \text{ , } \quad a = 0 \text{ , } \quad b = -255 \\ \text{if } A < 0 \text{ , } -1 \leqslant a < -\tfrac{1}{2} \end{array}\right\} \qquad (2)$$

The mantissa and exponent are both packed into one word of 39 binary digits. The 30 most significant digits hold the mantissa, using the same convention by which a 39-bit fixed-point fraction is normally held in a whole word. The 9 least significant digits represent the integer (b+255): this must be positive and cannot exceed 511 so the *possible* range of values of b is from -255 to +256.

Unpacked Standardised and Unstandardised Forms

During the actual calculation of results which are to be expressed in standardised floating-point form, it is necessary to deal with the mantissa and exponent separately, so that they must for the time being be held as two separate words.

In unpacked standardised representation, the two quantities a and b are held in separate locations, but they still obey both conditions (1) and (2) above. In unstandardised form they obey only condition (1).

Generally speaking, the calculation of a standardised floating-point result proceeds in three stages.

Firstly, an unstandardised result is produced, by a process which forms any fraction a and any integer b satisfying $a.2^b = A =$ desired result.

APPENDIX 3 - CONTINUED

This is then standardised, that is to say, a and b are adjusted until they obey conditions (2). Finally they are rounded and packed so that the result is represented in one word. The rounding and packing process consists of : rounding off a to 30 digits, adding 255 to b , and then placing (255 + b) and a in the same word.

Underflow and Overflow: Range of Representation

If any number generated during calculation has an exponent ≤ -255, it is considered to have underflowed and is set to zero, which is represented by thirty-nine 0's.

If any number generated during calculation has an exponent $\geq +256$, it is considered to have overflowed: it is set to

01111111111111111111111111111111111111111

which represents approximately 1.16×10^{77}, and the appropriate overflow indication is given.

Thus the exact range of *permitted* positive numbers is from

$\frac{1}{2} \times 2^{-254}$ to $(1 - 2^{-29}) \times 2^{255}$

and that of *permitted* negative numbers is from

$(-\frac{1}{2} - 2^{-29}) \times 2^{-254}$ to -1×2^{255}

This may be abbreviated by saying that the approximate range is

$\pm 2^{-255}$ to $\pm 2^{255}$, and zero.

or about

$\pm 1.73 \times 10^{-77}$ to $\pm 5.8 \times 10^{76}$, and zero.

(c) Representation in 803's Fitted with Automatic Floating-Point Units as used in 803 A 103.

The representation used in 803's with Automatic Floating-Point Units is described in the Guide to Programming the National-Elliott 803 Electronic Digital Computer (Appendix 5 of the 5th Edition) and elsewhere.

The most important difference from the representation described above is that the nine least significant digits of the word represent (b+256), not (b+255).

APPENDIX 4 ADDITION OF FURTHER FLOATING-POINT FUNCTIONS

Note: *All examples in this appendix assume that it is desired to add three new functions, and that these will be specified by mnemonic instructions of the forms:*

A=LIM B

A=MIN B

A=NORM B

where LIM, MIN and NORM are the names of the functions and A and B are floating-point variables.

It is not possible to use additional floating-point functions in a programme which is being translated and run by the Load and Go method.

SUMMARY

In order to be able to use floating-point functions which are not already available in 803 A 3 or 803 A 103 the following actions are needed:

1. Allocate a name to each such function.
2. Write machine code subroutines by which the functions are formed.
3. *EITHER* (a) Produce a tape of these subroutines, punched in the conventions of the autocode system *(Additional Functions Tape)*, to be used in conjunction with tape 2 of 803 A 3 or 803 A 103.

   OR (b) Produce a modified version of the appropriate tape 2, incorporating these additional subroutines *(803 A 3 or 803 A 103 with Additional Functions, Tape 2)*.
4. *EITHER* (a) Produce a tape by means of which the Autocode Translator can be adjusted each time it is required to translate a mnemonic programme using these additional functions *(Additional Names and Lengths Tape)*,

   OR (b) Produce a modified version of the appropriate tape 1 which will translate mnemonic programmes using the additional functions *(803 A 3 or 803 A 103 with Additional Functions, Tape 1)*.

APPENDIX 4 - CONTINUED

5. Specify any additional functions used in a mnemonic programme in the normal way, that is, by including their names in the SETF instruction, and use the additional or modified tapes correctly.

6. Ensure that modified versions of Tapes 1 and 2 of 803 A 3 or 803 A 103 do not get mixed up or used in conjunction with the standard versions, or differently modified versions.

1. ALLOCATING NAMES TO FUNCTIONS

Each name may be any combination of at least 3 of the letters of the alphabet, provided that the last 7 of them are different from the last 7 letters of any other function name in use.

for example: LIM MIN and NORM

2. WRITING THE REQUISITE SUBROUTINES

Each subroutine must be written as one block of machine-code programme, relatively addressed, and must comply with the following specification:

(i) The entry point is the first instruction in 0,.

(ii) Entry is made with the variable B in standardised floating-point form, in the accumulator.

(iii) If 803 A 3 is being used exit is by one of two methods.

EITHER (a) Exit is made by the instructions

00 15 / 40 1

with A, the required function of B, in the accumulator in standardised floating-point form.

OR (b) Exit is made by the instructions

73 7659 : 40 7714

40 7738 :

with the fraction a in 7648 and the integer b in 7649 where a and b have any values within machine capacity which satisfy

$a.2^b = A =$ required function of B

APPENDIX 4 - CONTINUED

Exit method (b) above used the standardising subroutine and the rounding-and-packing subroutine described in Appendix 5, and thereby achieves the same effect as exit method (a).

(iv) If 803 A 103 is being used, exit *must* be made by the instructions

00 15 / 40 1

with A, the required function of B, in the accumulator in standard automatic floating-point form.

3. (a) PRODUCING AN ADDITIONAL FUNCTIONS TAPE

(i) Punch T.I. Code tapes of the subroutines, complying with the methods of punching used in preparing tapes for input by 803 T 2, and with the following special provisions:

The first subroutine must be preceded by * and one * must occur between each subroutine and the next.

The last subroutine must be followed by)

If the subroutines are punched on more than one tape, each tape except the last must end with - -

The subroutines may be punched in any sequence, but a note should be kept of the actual sequence used; say (1) LIM, (2) MIN, (3) NORM.

(ii) Place Tape 3 of 803 A 3 or 803 A 103 in the tape reader and enter 40 0. Input is sum-checked; the continuous output of sp's indicates sum check failure.

(iii) Place the T.I. code tape of the additional subroutines in the tape reader and enter 40 512. The coding routines on the appropriate Tape 3 will now translate the subroutines into a form suitable for use in conjunction with the appropriate Tape 3, producing what is known as an Additional Functions Tape. If the subroutines have been punched on more than one tape, then when each tape has been translated, replace it by the hext in turn, and re-enter 40 256 to cause translation to continue.

3 (b) PRODUCING A MODIFIED TAPE 2 OF 803 A 3 OR 803 A 103

(i)-(iii) Proceed exactly as in 3(a) except that the last subroutine must end with 40 8189:(instead of). Then, without tearing off the Additional Functions Tape, continue as below.

APPENDIX 4 - CONTINUED

(iv) Place the leading end of a standard Tape 2 of 803 A 3 or 803 A 103 in the tape reader, and enter 40 544 (0001000100000). This will cause the relevant portions of that Tape 2 to be copied, so that the whole output constitutes "803 A 3 (or 803 A 103) with Additional Functions, Tape 2". Care should be taken to mark it properly, e.g. "803 A 3 (or 803 A 103) with LIM, MIN and NORM, Tape 2".

4 (a) ADJUSTING THE AUTOCODE TRANSLATOR

(i) Prepare an Additional Names and Lengths tape, consisting of

ls NAME sp fs n cr lf

for each function, where n is the number of locations taken up by the subroutine for the function.

The functions must be specified in the *opposite* sequence to that in which they were punched for coding, so in our example we would have, say,

ls NORM sp fs 42 cr lf

ls MIN sp fs 20 cr lf

ls LIM sp fs 33 cr lf

(ii) Read the appropriate Tape 1 into the computer in the normal way.

(iii) Place the leading end of the Additional Names and Lengths tape in the tape reader, and enter 40 1734 (0011011000110) if 803 A 3 is being used, or 40 1745 (0011011001111) if 803 A 103 is being used.

The Autocode Translator will read the first name and length on the tape, adjust itself accordingly, and stop. Re-enter 40 1734 (or 40 1745 if 803 A 103 is in use) repeatedly until all the names and lengths have been read and dealt with.

(iv) The Autocode Translator is now adjusted to translate mnemonic programmes using the additional functions, and can be used in the normal way.

4 (b) PREPARING A MODIFIED AUTOCODE TRANSLATOR

(i)-(iii) Proceed exactly as in 4 (a) (i) - (iii), and then continue as below.

(iv) Make sure that there is plenty of tape in the punch, and enter 40 1818 (0011100011010) if 803 A 3 is being used, or 40 1829 (0011100100101) if 803 A 103 is being used.

APPENDIX 4 - CONTINUED

(v) The Autocode Translator will now punch out a suitably modified version of itself, and care should be taken to mark it properly, e.g. "803 A 3 (or 803 A 103) with LIM, MIN and NORM Tape 1".

5. USING 803 A 3 OR 803 A 103 WITH ADDITIONAL FUNCTIONS

The following notes amplify the provisions of the basic specification of 803 A 3 & 103.

(a) Writing the Mnemonic Programme

The names of any additional functions used in a mnemonic programme must be specified in the SETF instruction in the normal manner and the way in which each additional function instruction is written must conform with the usual conventions.

(b) Translating the Mnemonic Programme

Either use a modified form of the appropriate Tape 1, produced as described in 3 (b) above, *or* use the standard form, modifying the translator after input by the method described in 3(a).

(c) Running the Translated Programme

If a modified form of Tape 2 has been punched, as described in 4(b) above, this should be used instead of the standard form, operating in the usual way.

If only an Additional Functions Tape has been produced, this should be read in (entering 40 0) after the translated programme but before the standard Tape 2 of 803 A 3 or 803 A 103. Otherwise, operate in the usual way.

6. INCOMPATIBILITY OF STANDARD AND MODIFIED VERSIONS OF THE TAPES

A mnemonic programme which does not in fact use any additional functions may, if desired, be translated by a modified version of the translator, but the translated programme tape so produced can only be run under a correspondingly modified version of Tape 2.

Care should be taken to ensure that each modified tape is kept with its pair, and does not get mixed up with standard tapes or with tapes modified for other sets of additional functions.

APPENDIX 5 DETAILS OF FLOATING-POINT SUBROUTINES

SYNOPSIS

This appendix gives sufficient details of certain subroutines on Tape 2 of 803 A 3 and 803 A 103 to enable users to employ them in machine code blocks (see section 19) and additional function subroutines (see Appendix 4).

The first section gives details of floating-point subroutines in 803 A 3 which perform the functions of addition, subtraction, multiplication, division, negation, standardisation, rounding-and-packing, reading and printing. Details of integer reading and printing subroutines are also included.

In the second section details of the READ and PRINT subroutines in 803 A 103 are given.

The last section lists the approximate operating times for the floating-point arithmetic and function instructions of 803 A 3, and the floating-point function instructions of 803 A 103.

Throughout this appendix, I and J are integers held to scale $\times 2^{-38}$.

1. DETAILS OF FLOATING-POINT SUBROUTINES IN 803 A 3

Throughout this section, all quantities except I and J are floating-point numbers in (packed) standardised form unless otherwise stated. Where floating-point quantities are held in unpacked form in locations 7648 and 7649, the mantissa (argument) is always in location 7648 and the exponent in location 7649.

ADD, SUBTRACT, MULTIPLY, DIVIDE

(a) Single step, or first step of a sequence of steps

To form a result C , which will be

both (i) in the accumulator

and (ii) in locations 7648 and 7649 in unpacked standardised form,

place A in location 7648 and B in the accumulator and enter as follows:

APPENDIX 5 - CONTINUED

For			Enter		
C=A+B	73	15	:	40	7793
C=A-B	73	15	:	40	7797
C=A*B	73	15	:	40	7801
C=A/B	73	15	:	40	7805
C=-A-B	73	15	:	40	7809
C=-A+B	73	15	:	40	7813
C=-A*B	73	15	:	40	7817
C=-A/B	73	15	:	40	7821

(b) Second and subsequent steps of a sequence of steps

To form a result C' , which will be

both (i) in the acumulator

and (ii) in locations 7648 and 7649 in unpacked standardised form,

leave C , the result of a previous step, in unpacked standardised form in locations 7648 and 7649, place D in the accumulator and enter as follows:

For			Enter		
C' =C+D	73	15	:	40	7794
C' =C-D	73	15	:	40	7798
C' =C*D	73	15	:	40	7802
C' =C/D	73	15	:	40	7806
C' =-C-D	73	15	:	40	7810
C' =-C+D	73	15	:	40	7814
C' =-C*D	73	15	:	40	7818
C' =-C/D	73	15	:	40	7822

NEGATE

To form -A , which will be

both (i) in the accumulator

and (ii) in locations 7648 and 7649 in unpacked standardised form,

place A in the accumulator and enter

73	15	:	40	7790

APPENDIX 5 - CONTINUED

UNPACK

No specific unpacking subroutine is available. Unpacking may be achieved by adding to zero and using the unpacked result in 7648 and 7649 if desired.

STANDARDISE

To standardise the unpacked unstandardised number in locations 7648 and 7649 enter

73 7659 : 40 7714

the result is placed in unpacked standardised form in the same locations.

ROUND AND PACK

To round and pack the unpacked standardised number in locations 7648 and 7649 enter

73 15 : 40 7738

the result is placed in the accumulator.

ERROR INDICATIONS

If floating-point overflow occurs during any of the above operations, the usual error indication is given.

See appendix 3 for definition of floating-point overflow, and Section 25.5 (a) for details of the error indication.

READ

Floating-point numbers

To read a floating-point number from the input tape, and place it both in the accumulator and in unpacked standardised form in locations 7648 and 7649, enter

73 15 : 40 7939

Integers

To read an integer from the input tape and place it in the accumulator, enter

73 15 : 40 7940

For details see Sections 14 and 21.

APPENDIX 5 - CONTINUED

PRINT

For Format Corresponding to	Set Parameter	In Location	Place A or I in the Accumulator and Enter
PRINT A,m:n PRINT A,I:J	00 m : 00 m+n 00 I : 00 I+J	7497	73 7659 : 40 7878
PRINT A,m PRINT A,I	+m +I	7497	73 7659 : 40 7880
PRINT A,m/ PRINT A,I/	+m +I	7497	73 7659 : 40 7882
PRINT A	-	-	73 7659 : 40 7887
PRINT I,m PRINT I,J	+m +J	7888	73 7659 : 40 7828
PRINT I	-	-	73 7659 : 40 7828

For details see Section 15.

2. DETAILS OF READ AND PRINT SUBROUTINES IN 803 A 103

Throughout this section, all quantities except I and J are standard floating-point numbers unless otherwise stated.

READ

To read a floating-point number from the input tape, and place it in the accumulator, enter

73 15 : 40 7958

To read an integer from the input tape and place it in the accumulator, enter

73 15 : 40 7959

For details, see Sections 14 and 21.

APPENDIX 5 - CONTINUED

PRINT

For Format Corresponding to	Set Parameter	In Location	Place A or I in the Accumulator and Enter
PRINT A,m:n PRINT A,I:J	00 m : 00 m+n 00 I : 00 I+J	7671	73 7801 : 40 7908
PRINT A,m PRINT A,I	+m +I	7671	73 7801 : 40 7910
PRINT A,m/ PRINT A,I/	+m +I	7671	73 7801 : 40 7912
PRINT A	-	-	73 7801 : 40 7917
PRINT I,m PRINT I,J	+m +J	7918	73 7801 : 40 7858
PRINT I	-	-	73 7801 : 40 7858

For details, see Section 15.

3. APPROXIMATE OPERATING TIMES OF FLOATING-POINT SUBROUTINES IN 803 A 3 AND 803 A 103

APPROXIMATE OPERATING TIMES OF FLOATING-POINT SUBROUTINES IN 803 A 3

FUNCTION OR OPERATION	TIMES ON 803 A's	TIMES ON 803 B's
	(Milliseconds)	(Milliseconds)
SIN	400	200
COS	400	200
TAN	500	250
ARCTAN	600	300
LOG	325	160
EXP	550	275
SQRT	170	85
STAND	55	36
INT	20	15
FRAC	30	20
MOD	5	3
ADD	70	40
SUBTRACT	70	40
MULTIPLY	70	40
DIVIDE	70	40

APPENDIX 5 - CONTINUED

APPROXIMATE OPERATING TIMES OF FLOATING-POINT SUBROUTINES IN 803 A 103

FUNCTION OR OPERATION	TIME ON 803 B's (milliseconds)
SIN	170
COS	170
TAN	180
ARCTAN	120
LOG	130
EXP	90
SQRT	50
INT	15
FRAC	10
MOD	3

Addition, subtraction, multiplication, division and the function STAND are performed by means of the appropriate automatic floating-point machine functions.

APPENDIX 6

EXAMPLE OF USE OF ELLIOTT AUTOCODE PROGRAMME SHEET

The accompanying sheet is an example of the method of writing programmes in autocode employed at Elliott Brothers (London) Ltd.

WRITING ON THE FORM

In general, the REF. column is used for reference numbers, while instructions and remarks are written in their appropriate columns. The use of the FLOW columns is at the discretion of the programmer: one possible method of indicating jumps etc. is shown.

Working through the example, the following special cases will be observed:-

(a) REF. column used for :: before programme name, and the programme name is written across the page. (20.3(b)).

(b) An additional cr lf must be directly specified: the punch operator is instructed to ignore lines left completely blank. (20.2).

(c) Where any instruction is too long to be written on one line, a large x appears in the END column, and instruction is continued on the next line.

(d) TITLE instructions are written out in full. (16.1).

(e) :: is written in the END column where a remark is to be punched. (20.3 c). (If a remark is not to be punched, the END column is left blank).

(f) A space should *never* be left between FILM and the bracketed variable (I) which follows it. For the purpose of inserting spaces, FILM(I) should be regarded as one 'word' i.e. the space comes after the (I). (This is also true when FILM is followed by a bracketed constant).

PUNCHING FROM THE FORM

General

(a) Lead in with blanks and fs or ls followed by cr lf

(b) Punch whatever is written in the REF. and INSTRUCTION columns on one line, then punch cr lf and go on to the next line, and so on.

APPENDIX 6 - CONTINUED

(c) Finish off with cr lf and run out on blanks.

(d) You may put fs or ls in wherever needed: it does not matter if you put in too many.

(e) Punch one sp after the 'words'
ALLOW, ARCTAN, BLOCK, CHECK, COS, CYCLE, EXP, FILM, FILM(*I*), FILM(*n*), FRAC, FROM IF, INPUT, INT, JUMP, LINES, LOG, MOD, NUMBER, OUTPUT, PREVENT, PRINT, PUNCH, READ, READER, REPEAT, SEARCH, SEARCHING, SET, SETF, SETR, SETS, SETV, SIN SPACES, SQRT, STAND, START, SUBR, TAN, TELEPRINTER, TITLE, TO, TRIG, UNLESS, VARY, WRITE and *nowhere else*.

Exceptional Cases

(f) Where :: appears in the REF. column, punch :: and then copy everything else on that line, with spaces between words, and end with cr lf

(g) Where :: appears in the END column, punch :: and then copy the REMARKS column, with spaces between words, before punching cr lf

(h) Where there is no :: in the END column, do not copy the remarks.

(i) Where there is X in the END column, or at the end of the line, go on to the next line *without* punching cr lf

(j) Take no notice of the FLOW column entries.

(k) If you see cr lf written on a line, just punch cr lf once and go on to the next line.

(l) After TITLE punch sp followed by all the characters in the circle followed by cr lf

(m) An entry like sp^6 means six sp's

803 AUTOCODE PROGRAMME

APPENDIX 6 TO 803 A3 AND 103 (Continued)

FLOW	REF.	INSTRUCTIONS	END	FLOW	REMARKS
	::	EXAMPLE OF USE OF AUTOCODE PROGRAMME SHEET			
		cr lf			
		SETS I(64)J(64)K(64)L(5)M(10)	X		
		NP(2)			
		SETV A(10)BC			
		SETF READER PUNCH FILM			
	17)	TELEPRINTER			
		TITLE cr lf sp 6 SPECIFY sp CHARACTERS sp REQUIRED sp INCLUDING sp FINAL sp BLANK cr lf cr sp 6 THEN sp ENCIRCLE sp WHOLE sp TITLE sp AS sp SHOWN bl			
		SET L(0:5) = 0, 1, 2, 3, 4, 5			
		READ N	::		INPUT BLOCK NUMBER
		P=N+12			
		SUBR 18	::		GENERATE J(K)
		CYCLE K=N:4:P			
		CYCLE I=1:1:4			
		FILM(I) ALLOW WRITE			
		FILM(I) SEARCH K			
		FILM(I) FROM J(K)	::	18	WRITE J(K) ON FILM
		JUMP @21			
	7)	REPEAT I			
	8)	REPEAT K			
		CYCLE K=N:4:P			
		CYCLE I=1:1:4			
		FILM(I) SEARCH K		21	
		FILM(I) TO J(K)	::		READ J(K) OFF FILM
		CYCLE M=K:1:64			
		L=M*M			
		L=L+M			
		PUNCH 1			
		@30L:501			
		20L:000			
		)			

FROM SHEET 5 (17)

UNIVERSITY OF HULL

COMPUTATION LABORATORY

Elliott Autocode

New versions of the Autocode compiler tapes have been received. They are labelled:

Autocode Tape 1 (issue 4) F.C.B.
Autocode Tape 2 (issue 3)
Autocode Tape 3 (issue 3)
Autocode Load-and-Go with Verify (Issue 4) F.C.B.

All these tapes have 'legible' headings which must not be read by the computer. They will come into use immediately.

All programs already written in Autocode will be correctly translated by the new compiler tapes. The only exception is where a program has already been translated to paper tape using the old version of Autocode Tape 1 (namely, issue 3). In this case issue 2 of Autocode Tape 2 must be used with the translated tape. This version of tape 2 will be kept in the bottom drawer of the tape cabinet in the computer room until June 30th 1965. By this date anybody who has a tape-1-translated program should have translated it again, using the new version of tape 1, or made an F.C.B. program tape - preferably both !

The main changes incorporated in the new Autocode tapes are as follows:

Additions

(1)) on a data tape means WAIT, not STOP;
i.e. if the wait button is changed, the program will now continue reading;

(2) The exponent of a floating-point number on a data tape may be preceded by / or @; e.g.

-1·2463/-1 or -1·2463@-1

which both mean -0·12463. (This enables autocode programs to read Algol data tapes, so long as the latter contain numbers only and do not use single spaces as the separators between numbers).

(3) Floating-point numbers will be punched with @ instead of / before the exponent; e.g. ·12345678@+00. (The autocode instruction is still PRINT A,n/)

(4) CHECK and the error printing of floating-point numbers will be in the style PRINT A,8/. (Previously 9 digits were printed, but the computer accuracy is only $8\frac{1}{2}$ significant digits. PRINT A,9/ is still allowed).

(5) JUMP @In and SUBR In (where n is an integer only) are allowed. Previously no suffix was allowed.

(6) TITLE instructions will be obeyed more quickly.

(7) Additional functions which have integer or constant parameters can be added.

(8) The space available for program and data is now 5068 locations on Load-and-Go (7588 on Translate-and-Run as before).

Errors in previous version

(9) In the READ routine, the two spaces after a number had to be consecutive to terminate the number. This has been modified to accept ignorable characters (fs,ls,cr,bl) between the spaces.

(10) PRINT n, where n is a constant, and A = COS BKS, are now recognised as errors.

(11) All error indications are output on punch 1 (irrespective of any PUNCH instructions).

(12) sp or lf in a machine-code block is now recognised and causes a wait. (Continue by changing the sign, never by entering 40 17 in this case).

D.G. Burnett-Hall.