PART 1: INTERCODE SPECIFICATION

CONTENTS

1. INTRODUCTION
2. ATNM OF INTERCODE
3. GENERAL FORM OF AN INTERCODE PROGRAM:
4. INTERCODE INSTRUCTIONS
   4.1 General
   4.2 Composition
   4.3 Main Parts of an Instruction
   4.4 Continuation lines
   4.5 Continuation Sheets
5. THE ACTION CODE
   Group 0 - Arithmetic on Variables
   Group 1 - Arithmetic on Literals
   Group 2 - Floating Point Arithmetic
   Group 3 - Pattern Manipulation
   Group 4 - File Control
   Group 5 - Re-arrangements
   Group 6 - Endix Facilities
   Group 7 - Sequence Changes
   Group 8 - Subroutines and Switches
   Group 9 - Modification Register Controls
   Groups 10 to 13 - Computer Code Actions
   Group 14 - Double Length Arithmetic
   Group 15 - Master Program Controls
6. TABLES
   6.1 Purpose
   6.2 Table Entry Sheets - General
   6.3 Heading Block Entries
   6.4 Table Entries - Details of Columns
   6.5 Use of Tables for more than one Action
   6.6 Note on Edit Tables

7. CONSTANTS
   7.1 General
   7.2 Heading Block Entries
   7.3 Details of Method of Entering Constants

8. PROGRAMME SHEETS
   8.1 General
   8.2 The Programme Heading Parameters Sheet
   8.3 The Section Description Sheet
   8.4 The Programme End Sheet
   8.5 Sheet Numbers
   8.6 Start and End of Reel Sheets

9. ANNOTATION

10. CHAPTERS
    10.1 General
    10.2 Extra Chapters
    10.3 Chapter Lengths

11. MODIFICATION OF INSTRUCTIONS
    11.1 Modification Registers
    11.2 Modification Types
    11.3 Modification Times
    11.4 Example
    11.5 Notes on Negative Numbers in Modification Registers

Amendment No. 21
January 1964
12. FILE CONTROL
   12.1 General
   12.2 Sentinel Blocks
   12.3 Input and Output Notes
   12.4 Re-alignment
   12.5 Non-Standard Card Output
   12.6 Non-Standard Card Input
   12.7 Non-Standard Paper Tape Input
   12.8 Non-Standard Paper Tape Output
   12.9 Dual Input (Paper Tape/Cards)
   12.10 Dual Printer

13. BLOCK LAYOUTS
   13.1 Magnetic Tape
   13.2 Alpha/Numeric Punched Card Input
   13.3 Alpha/Numeric Punched Card Output
   13.4 Paper Tape Input
   13.5 Paper Tape Output
   13.6 High Speed Printer

14. TRANSIT AREAS
   14.1 Definition
   14.2 Reference to Input and Output Sections
   14.3 Overwriting and Corruption
   14.4 Total Area Required by Annexes
   14.5 Sets of Transit Areas
   14.6 Example of Store Space Required by Transit Areas

15. PROGRAMME AMENDMENTS
   15.1 Introduction
   15.2 Form of Amendments
   15.3 Amendment of Programme Heading Information
   15.4 Amendments of Instructions, Table Entries and Constants
   15.5 Amendments to the Section Description
16. TRIAL FACILITIES
   16.1 Programme Trials System
   16.2 Trial Data
   16.3 Post-Nortem Points
   16.4 Use of the Translator to Amend Trial Data or Post-Nortem Points

17. THE OPERATIONAL PROGRAMME
   17.1 Programme Files
   17.2 Allocation
   17.3 Programme Loading Sequence
   17.4 Parameters

Appendices
A. TRANSLATIONS OF INTERCODE ACTIONS
B. INTERCODE ACTION TIMES
C. SPECIMEN PROGRAMME SHEETS (PAPER TAPE)
D. SPECIMEN PROGRAMME SHEETS (PUNCHED CARDS)
1. **Introduction**

Intercode is a new interim code, more convenient and powerful than existing computer code but not requiring the complex compiling process of a complete machine code. Programmes written in Intercode require translating into computer code before they can be run; the programme which performs this task is known as the Intercode Translator.

This specification describes the Intercode programming system, including the various parts of an Intercode programme, the actions available, programming techniques, translation into computer code, and programme amendments. It assumes knowledge of the "hardware" of the LEO III computer and the form in which information can be held, which are not peculiar to Intercode.
2. **AIMS OF INTERCODE**

The aims of Intercode are:

i. To reduce coding labour and hence coding errors; in particular, to ensure that inter-division addressing and transmit area addressing are dealt with automatically.

ii. To ensure that programmes are presented to the Master Routine in the standard form which it requires. (The Master Routine is a programme, held permanently in the store during normal running, which controls the operation of all programmes including loading, file control, communications with human operator and unloading).

iii. To simplify the amendment of programmes.

iv. To allow the automatic production and maintenance of a fully annotated printed version of the programme.
GENERAL FORM OF AN INTERCODE PROGRAMME

A programme in Intercode as presented to the translator consists of the following:

Programme loading Parameters sheet to identify the programme and give other general information (see section 6.2.)

Section description sheet to specify sections of the store required by the programme for data, results, etc. (see section 6.3.)

Procedures to specify the operations the computer must perform to carry out the job (see section 4 for form of instructions and section 5 for action code).

Tables to specify the format of data and results (see section 6).

Constants needed by the programme (see section 7).
(The procedures, tables and constants need not be in this order)

Programme end sheet (see section 6.4.)

Start and End of Real sheets (see section 8.6.)

The programme in this form is translated (see section 16) and an operational computer code programme (see section 17) produced. The programme may later be amended (see section 15).
Section 4

4. INTERCODE INSTRUCTIONS

4.1 General

The Intercode programme consists of a set of procedures, one procedure representing a logical step in the execution of the job. Each procedure is normally sufficiently short to fit on one coding sheet (see specimens, Appendices C and D) though continuation sheets may be used if required (see section 4.5). The procedures are numbered in the range 100 to 998 and will be translated in the order of submission which need not be numerical order. Each instruction within a procedure is labelled with a serial number so that any instruction may be referred to by quoting the procedure number followed by the instruction number. On continuation sheets serial numbers should be arranged to follow in sequence through the section, but no serial number may exceed 97.

4.2 Composition

An Intercode instruction is written in six main parts which are:

- Action
- Reference
- Item
- Discriminant
- Modifier
- Literal

These are written by the programmer:

\[ A \quad R \quad I \quad D \quad M \quad L \]

In addition the instruction has a serial number within the procedure, pre-printed on the coding sheet, and may have annotation (see section 9).

4.3 The significance of the six main parts is as follows:

4.3.1 Action

Intercode actions are divided into 16 groups with up to 10 variants in each group. The action number is written as the group number followed by the variant number, e.g. Group 9 variant 3 is simply written 93. Some actions have continuation lines; on these the letter C is entered in the action column.

4.3.2 Reference

Instructions need to refer to the following types of information:

(i) Procedures, constants sections or tables, numbered in the range 100 to 998.

(ii) Input/Output and working sections, numbered in the range 1 to 99.

Amendment No. 24
May 1964
(iii) File identifiers (for file control actions), consisting of a letter followed by a number.

The reference column may also contain:

(iv) 'P', indicating an address relative to the start of the current procedure.

(v) '999', indicating an address relative to the start of the current chapter (see section 10).

(vi) Zero, indicating an absolute address; used either for fixed addresses in division 0 (e.g. floating point accumulator), or in conjunction with 15-bit modification (see section 11).

(vii) A number in the range 1001 to 1020. This is used within subroutines to address subroutine parameters, i.e. the C-lines of the 'Enter SR' action (see action 90 definition). Reference (1000+x) corresponds to the xth C-line, and causes the address therein to be added to the item/discriminant/modifier/literal of the current action. The use of this facility is restricted to actions addressing a real store compartment and excludes:

- Computer code actions (100 to 131)
- Sequence change actions (70 to 79, 85, 94, 97, 150)
- Switch actions (83 to 85)
- Table lookup (action 96)
- C-lines of action 80
- File control actions (40 to 49)

4.3.3 Item

This shows the number of the item referred to within its section or procedure. Negative numbers are permitted when meaningful in modified instructions (see section 4.3.5).

The item may not be greater than 99 when addressing a procedure/table/constant section, and ±8191 when addressing a working section. Reference over the end (or start) of a section will address the appropriate part of the next (or previous) section.

Exceptionally, a modified reference to a constants section may have the item in the range ±99.

Amendment No. 24
May 1964
Section 4.3 (Cont'd)

When referring by any action to a section holding condensed data (see section 5, actions 51 and 53), it may be convenient to give the address of the required location not relative to the start of the section, but relative to the next compartment to which data is to be condensed or from which data is to be unpacked. In this case the number in the item column, k say, must be followed by a plus sign (preceding the minus sign if any), and refers to the kth compartment on from this next compartment. If k is zero the entry should be simply a plus sign.

It is not possible to use the same section for both condensing and unpacking if the 'i' facility is being used.

4.3.4 Discriminant

The discriminant may have the value 0, 1 or 2. 0 and 1 imply reference to a short or a long word respectively. Discriminant 2 also refers to a long word but to the numeric part of this word, i.e. the odd numbered quartets. Thus it is possible to make alpha to numeric and numeric to alpha transfers between the store and the arithmetic unit, the numeric item being one in the arithmetic unit. Discriminant 2 may be used with the following actions only:

00 to 09, 16, 17, 18, 34, 36, 37, 46 to 49 (continuation lines),
54 (continuation line), 55 (third line), 56, 59 (continuation line),
63, 64, 65, 84, 92, 95.

4.3.5 Modifier

Any reference to a modification register or to a modified address will have the entry in this column. Up to 20 modification registers are available; for the optimum use of these and other details of the system see section II.

4.3.6 Literal

The literal specifies a numeric constant to be worked on directly by the action. The maximum allowable size is 15 digits (plus sign). Literals are not modifiable.

4.4 Continuation lines

Some actions have or may have one or more continuation lines. Continuation lines must have a C in the action column and they contain further details for the action which has been specified in the first line. No sequence change may ever be made to a continuation line.

4.5 Continuation Sheets

If a procedure is too long to go on one coding sheet, continuation sheets may be used. For specimen see Appendices C and D. The serial number of the instructions must be inserted on the sheets and arranged so that they are in sequence right through the procedure.

Amendment No. 24
May 1964
5. **THE ACTION CODE**

The Intercode actions are divided into groups. Each action is described below under the following headings:

(a) **Definition**
(b) **Notes**
(c) **Modification Type**

**Notation**

A = Upper Accumulator
B = Lower Accumulator
A' = Floating Point Arithmetic Accumulator
N = The address of the instruction (defined by R,I,D,N)
R = Entry in reference column (see section 4.3.2)
I = Entry in item column (see section 4.3.3)
D = Entry in discriminant column (see section 4.3.4)
M = Entry in modifier column (see section 4.3.5)
L = Entry in literal column (see section 4.3.6)
C_m = Continuation line m
NC_m = Address on continuation line m
RC_m = Reference on continuation line m
LO_m = Literal on continuation line m

The use of brackets denotes the contents of the location or register specified.

**Notes**

Radix

Unless otherwise stated the radix set in register C is used.

Modification

Modification types and associated lines are defined in section 11.

Translation

The machine code translation of each action is given in Appendix A.

Timing

The time taken by each Intercode action is given in Appendix B.
Representation of Numbers

Except for floating point data, and certain special actions, numbers are held in the store in sign and modulus form, and in the arithmetic unit in sign and complement form. The normal operations of arithmetic, e.g. select, copy, add, subtract, multiply, divide, will complement negative numbers in store - arithmetic unit exchanges.

Representation of -0

-0 is normally represented in the store and registers as +0. -0 can appear in the store. It is represented by a word clear except for the sign bit. The most common occurrence of -0 in the store is at the most significant end of a long word holding a negative number. For instance, if \((74') = -3\) \((74) = +3\) and \((75) = -0\). Certain forms of input, in particular non-standard card input, can cause -0 to be read into the store.

If -0 is selected into register A it will appear as +0, and if +0 or -0 is copied or transferred from register A to the store, it will always appear in the store as +0. Therefore, if a compartment is suspected of containing -0 the following procedure should be adopted for testing it:

(i) Copy the suspected compartment numeric \(\rightarrow\) alpha (Action 55).

The destination will contain the alpha representation of -0, i.e. \(Q^{10} = 1, Q^9 \rightarrow Q^1 = 0\).

(ii) Select and test the alpha representation by normal methods.

-0 can be placed in a store compartment by the reverse process, i.e. by copying the alpha representation alpha \(\rightarrow\) numeric (Action 55).

-0 can be generated in registers A and B by starting with any negative number in the register, and shifting all the digits out of the most significant end. This will leave A or AB with the sign bit set and all quartets at zero. If tested -0 will be found to be non-zero and negative. Therefore, to establish whether the accumulator contains -0 or a negative number, the contents of the register should be transferred to a store location and selected into the accumulator in the normal way. If the original number in the register was -0 it will have been complemented on the transfer and thus will appear as +0 when selected into the register again. This can be tested in the normal way (Action 70).

Certain forms of data handling actions, e.g. Edit (Action 52) and Condense (Action 55) do not interpret -0 correctly in a source compartment (see Action Descriptions).

The Collate action (35) cannot be used to test the store for -0 and the replaced selected bits action (37) cannot be used to set the sign bit of a zero word.

Amendment No. 29
December 1964
GROUP Q = ARITHMETIC ON VARIABLES

Note: The first zero in each action number need not necessarily be present.

Action 00: SELECT
Definition: Select (N) into A
Note: Overflow will occur if the contents of A plus the contents of C exceed $2^{50}-1$ at the start of the action; otherwise the radix set in C is immaterial for positive numbers.

Modification type: 1

Action 01: ADD
Definition: Add (N) to (A)

Modification type: 1

Action 02: SUBTRACT
Definition: Subtract (N) from (A)

Modification type: 1

Action 03: COPY
Definition: Copy (A) to N
Notes: (i) (A) is unchanged
(ii) The radix set in C is immaterial for positive numbers

Modification type: 1

Amendment No. 10
March, 1963
Action 04 : AUGMENT

Definition: Augment (N) by (A)

Notes: (i) (A) is unchanged
      (ii) Overflow will occur if the initial contents of N and A are zero and minus zero respectively.

Modification type: 1

Action 05 : TRANSFER

Definition: Transfer (A) to N

Notes: (i) A is cleared
      (ii) The radix set in C is immaterial for positive numbers

Modification type: 1

Action 06 : MULTIPLY

Definition: Multiply (A) by (N) and place in AB

Notes: (i) (A) and (N) must be in a common uniform radix
      (ii) (A) may be up to 9 digits initially.

Modification type: 1

Action 07 : MULTIPLY and ADD

Definition: Multiply (N) by (NC,) and add to (A)

Notes: (i) (N) must be in decimal. If any quartet of the number exceeds 9, the action will treat that quartet as zero in forming the product. (NC,) and the result are in the radix specified by C.
      (ii) Register B is left clear.
      (iii) Overflow will occur if:
              (a) $10^k \times (NC,) > 2^{AD-1}$, where k is the number of significant digits of (N);
              (b) the contents of N and NC, are zero and minus zero respectively.

Modification type: First line 2
Continuation line 1

Amendment No. 27
August 1964
Section 5 (Cont'd)

**Action OA : MULTIPLY and SUBTRACT**

**Definition:** Multiply (N) by (NC₁) and subtract from (A).

**Notes:**
1. (N) must be in decimal. If any quartet of the number exceeds 9, the action will treat that quartet as zero in forming the product (NC₁) and the result are in the radix specified by C.
2. Register B is left clear.
3. Overflow will occur if \( 10^k \times (NC₁) > 2^{10} - 1 \), where \( k \) is the number of significant digits of (N).

**Modification type:** First line 2
Continuation line 1

**Action OB : DIVIDE**

**Definition:** Divide (AB) by (N) leaving quotient in A and remainder in B.

**Notes:**
1. (AB) and (N) must be in a common uniform radix specified by C.
2. Q<sup>10</sup> of the modulus of (A) must be zero initially.
3. The remainder is always positive and is aligned to Q<sup>2</sup> of B.
4. (N) must be \( \leq \frac{R^{10}}{L^{1-1}} \) where R is the radix.
5. Q<sup>1</sup> of B is left clear.
6. Overflow will occur if:
   a. the divisor is zero or minus zero;
   b. (AB) is minus zero initially;
   c. the quotient is >10 digits.

**Modification type:** 1
**Group 11: ARITHMETIC ON LITERALS**

**Note:** Each digit of the literal represents one quartet. Arithmetic is performed according to the current setting of Register C.

**Action 10: SELECT LITERAL**

**Definition:** Select L into A

(i) L may be positive or negative

(ii) Overflow will occur if the contents of A plus the contents of C exceed $2^{40} - 1$ at the start of the action; otherwise the radix set in C is immaterial for positive numbers.

**Modification Type:** Non-modifiable

**Action 11: ADD LITERAL**

**Definition:** Add L to (A)

**Note:** L must be positive

**Modification Type:** Non-modifiable

**Action 12: SUBTRACT LITERAL**

**Definition:** Subtract L from (A)

**Note:** L must be positive

**Modification Type:** Non-modifiable

Amendment No. 10
March, 1963
Action 16: MULTIPLY LITERAL

Definition: If N is non-zero, multiply (N) by L and place in A.
If N is zero, multiply (A) by L.

Notes:
(i) L must be in decimal. If any quartet of the number exceeds 9, the action will treat that quartet as zero in forming the product. (N) or (A) and the result are in the radix specified by C.
(ii) L may be positive or negative.
(iii) Register B is left clear.
(iv) Overflow will occur if:
(a) L exceeds 9 digits;
(b) \(10^k \times L > 2^{40} - 1\), where \(k\) is the number of significant digits of (N);
(c) (N) is zero and L is minus zero.

Modification type: 1

Action 17: MULTIPLY LITERAL AND ADD

Definition: Multiply (N) by L and add to (A).

Notes:
(i) L must be in decimal. If any quartet of the number exceeds 9 the action will treat that quartet as zero in forming the product. (N) and the result are in the radix specified by C.
(ii) L may be positive or negative.
(iii) Register B is left clear.
(iv) Overflow will occur if:
(a) L exceeds 9 digits;
(b) \(10^k \times L > 2^{40} - 1\), where \(k\) is the number of significant digits of (N);
(c) (N) is zero and L is minus zero.

Modification type: 1
Action 19: DIVIDE LITERAL

Definition: If \( N \) is non-zero, divide \((N)\) by \( L \), leaving quotient in \( A \) and remainder in \( B \). If \( N \) is zero, divide \((A)\) by \( L \), leaving quotient in \( A \) and remainder in \( B \).

Notes:

(i) \( N \) or \( A \) and \( L \) must be in a common uniform radix specified by \( C \).
(ii) \( L \) may be positive or negative.
(iii) The remainder is always positive and is aligned to \( Q2 \) of \( B \).
(iv) \( Q1 \) of \( B \) is not left clear, unless it was clear at the start of the action.
(v) Overflow will occur if:

(a) the divisor is zero, minus zero or exceeds 10 digits;
(b) \((AB)\) is minus zero initially.

Modification type: 1
Group 2: **FLOATING POINT ARITHMETIC** (only available on computers with relevant computer code facilities)

**Note:** Floating point arithmetic is performed on numbers consisting of a half-word exponent and a full-word numerical part stored consecutively. The numerical part of mantissa \( m \) is a binary number of 36 bits and sign in sign and complement form, in the range \(-1 < m < 1\), the binary point being assumed to be between bits 36 and 37. The exponent \( e \) is a binary number of 20 bits and sign in sign and modulus form, and is artificially restricted to lie in the range \(-2^{19} \leq e \leq 2^{19}\).

The exponent precedes or follows the numerical part depending on whether the address is odd or even. The address entered is that of the first half-word.

A floating point 'accumulator' denoted by \( A^* \) is used. This holds a full-word mantissa as above and a full-word exponent in the range \(-2^{40} \leq e < 2^{40}\). Mantissa and exponent are in sign and complement form.

Two modes of arithmetic are available:

- **Significant Digit Arithmetic** where the number of significant bits in the result is as many as is justified by the significant bits in the data.

- **Normalized Arithmetic** where all the available bit positions are used by restricting the range of the mantissa to \( \frac{1}{2} \leq m < 1 \) or \( -1 < m < -\frac{1}{2} \), except for zero, which is represented as mantissa 0, exponent \(-2^{19}\).

All results are rounded off automatically in both systems.

The mode of arithmetic must be set by action 66 or 67 before any floating-point actions are used.

The original contents of \( A \) and \( B \) are not preserved, and after all floating point actions \( (A) = (A^*) \) mantissa.

Amendment No. 23
April 1964

All issues
Action 20 : SELECT (N) F/Point
Definition: Select (N) into A*
Modification type: 1

Action 21 : ADD (N) F/Point
Definition: Add (N) to (A*)
Note: If the sum is zero, (A*) are set equal to floating point zero if normalised mode is set.
Modification type: 1

Action 22 : SUBTRACT (N) F/Point
Definition: Subtract (N) from (A*)
Note: If the difference is zero, (A*) are set equal to floating point zero if normalised mode is set.
Modification type: 1

Action 23 : COPY F/Point
Definition: Copy (A*) to W
Notes: (i) If the exponent of (A*) is less than \(-2^{19}\), floating point zero is copied to store.
(ii) If the exponent of (A*) is greater than \(2^{19}\), the overflow condition is set and no copy is made.
Modification type: 1

Action 24 : AUGMENT F/Point
Definition: Augment (N) by (A*)
Modification type: 1

Amendment No. 23
April 1964
Action 25 : TRANSFER F/Point

**Definition:** Transfer (A*) to N

**Notes:**
(i) If the exponent of (A*) is less than \(-2^{19}\), floating point zero is transferred to store.
(ii) If the exponent of (A*) is greater than \(2^{19}\), the overflow condition is set and no transfer is made.

**Modification type:** 1

Action 36 : MULTIPLY F/Point

**Definition:** Multiply (A*) by (N) and place in A*

**Note:** If the product is zero (A*) are set equal to floating point zero if normalized mode is set.

**Modification type:** 1

Action 29 : DIVIDE F/Point

**Definition:** Divide (A*) by (N) and place in A*

**Note:** If (N) are zero, (A*) remain unchanged and overflow condition is set.

**Modification type:** 1
Group 3: Pattern Manipulation

Action 30: Shift (A) by Quarters

Definition: Shift (A) by L quarters; if L is positive shift left, if L is negative shift right.

Notes:

1) (B) and the sign bit of A are not affected.

2) The shift is arithmetic; i.e., when negative numbers are shifted right the quarters vacated are filled with "reflections" of the top digit of register C.

3) L must not exceed 31

Modification type: Non-modifiable

Action 31: Shift (A) by Bits

Definition: Shift (A) by L bits; if L is positive shift left, if L is negative shift right.

Notes:

1) (B) and the sign bit of A are not affected.

2) Right shift consists of a left shift of \( \left\lceil \frac{L}{4} \right\rceil \) bits followed by a right shift of \( \left\lfloor \frac{L}{4} \right\rfloor \) quarters.

3) Up to 3 bits may thus be lost from the more significant end of (A).

iii. The quartet part of the right shift is arithmetic as in note (ii) to action 30.

Modification type: Non-modifiable
**Action 32: SHIFT (AB) BY QUARTETS**

**Definition:** Shift (AB) by L quartets; if L is positive shift left, if L is negative shift right.

**Notes:**
1. The sign bits of A and B are not affected.
2. The shift is arithmetic; i.e., when negative numbers are shifted right the quartets vacated are filled with "reflections" of the top digit of register G.
3. L must not exceed 31.

**Modification type:** Non-modifiable

---

**Action 33: SHIFT (AB) BY BITS**

**Definition:** Shift (AB) by L bits; if L is positive shift left, if L is negative shift right.

**Notes:**
1. The sign bits of A and B are not affected.
2. Right shift consists of a left shift of \( \left( \frac{L}{4} \right) \) x 4 + L bits followed by a right shift of \( \left( \frac{L}{4} \right) \) quartets.
3. The quartet part of the right shift is arithmetic as in note (ii) to action 32.

**Modification type:** Non-modifiable

---

**Action 34: SET (B) IN B**

**Definition:** Select (B) into B in sign and modulus form

**Notes:** The radix set in C is immaterial

**Modification type:** 2
**Action 35: PREPARE FOR DIGIT COLLATION**

**Definition:**
- For each of the least significant ten digits of L which is 1, set the corresponding quartet in B as "1111",
- and for each which is 0, set the corresponding quartet in B as "0000".
- Copy the sign of L into the sign of bit of B.

**Notes:**
1. L must consist only of a pattern of 1's and 0's, and may be followed by a negative sign.
2. The radix set in C is immaterial

**Modification Type:** Non-modifiable.

---

**Action 36: COLLATE AND ADD**

**Definition:**
Collate (A) with (B), change the result from sign and modulus to sign and complement form, and add it to (A).

**Notes:**
1. The "result" consists of those bits present in both (B) and (A).
2. If the result of the collation is minus zero it is treated as zero.
3. Overflow will occur if (A) is minus zero initially and the result of the collation is negative.

**Modification Type:** Non-modifiable.
Action 77: REPLACE SELECTED BITS

Definition: Replace those bits of (N) which correspond to bits present in (A) by the corresponding bits of the sign and modulus form of (A)

Note:
(i) Negative numbers are held in A in sign and complement form.
(ii) If (A) is minus zero it is treated as zero.

Modification type: 1

Action 58: ROUND OFF

Definition: Round off (A) by adding 1 to (A) if the most significant quartet of B is > 5.

Note:
(i) Register B is left clear.
(ii) Overflow will occur if the contents of A plus the contents of B exceed 2^60 - 1 at the start of the action.

Modification type: Non-modifiable.

Action 59: ROUND UP

Definition: Round up (A) by adding 1 to (A) if (B) is non-zero.

Note:
Register B is not cleared

Modification type: Non-modifiable.

Amendment No. 10
March, 1963
Group A : FILE CONTROL

Notes:  
(i) All the actions in this group are carried out under Master Programme control.
(ii) Further rules governing the use of the actions are given in section 12.
(iii) The radix set in C is immaterial for all these actions.
(iv) None of these actions is modifiable.

Action 40 : OPEN FILE

Definition: Open files E, RC, EC, etc. The type of each file is defined by L, LC, LC, LC, etc. as follows:
  0 = Output file
  1 = Input file

Notes:  
(i) Input and output files are opened by this action.
(ii) Standard alignment procedures are carried out, so that for an input file the first 'Input' instruction will make the first data block available to the programme.
(iii) A file may not be used until it has been opened, and may not be used after a 'rewind' instruction until it has been re-opened.
(iv) The C-lines of a 40 actions can have two meanings; where the C-line discriminant is zero, this indicates that a further file RC is to be opened according to LC. Where the discriminant is non-zero, the C-line reference and item specify an address referring to the previous line of the 40 action, such that a file is opened for input or output on the indicated route, but (NC) specify the expected or required file identity and programme identity on that route. The even half of (NC) gives the programme identity, and the odd half the file identity right-aliigned. The report 'D' will be made against C-lines with non-zero discriminants, to call attention to possible accidental use of the facility; the report can be suppressed by the usual method.
(v) The current run number (from the programme control card or paper tape, see section 19) is left in register A in decimal; B is clear when the programme is running under Issues 1 to 3 Master, and for MT files only, contains the appliance number when the programme is running under Issue 4 or later Master.

Amendment No. 28
November 1964
Section 5 (Cont'd)

(vi) For magnetic tape input files only, additional information is left in register A; the layout is:
   Q1 - Q4 Current run number
   Q5 Blank
   Q6 - Q9 Number of run which produced this file
   Q10 File Identity Number of file when output.

(vii) The following checks are made when opening magnetic tape files and on each subsequent reel of these files:

Output: Each reel must be on the 'released tapes' list, or must have been written as the same reel of the same file of the same programme in the same run. See Volume IV, Part 3, section 1.3.1 (ii) for full details.

Input: Each reel must have been written as part of a file by a programme in this suite. See Volume IV, Part 3, section 1.3.2 (ii) for full details.

(viii) On opening output magnetic tape files, control data for the external tape control system is produced.

(ix) If the previous file control order for this route was 'close reel', then the 40 action will be interpreted as 'open reel' rather than 'open file'. An output reel will have its reel number stepped on by one; an input reel will have its reel number tested, and an alarm will be raised if the check fails. In this case the operator can accept the magnetic tape, and the programme must check the first block of the new reel. (see note (ii) to the 43 action.)
Section 5 (Cont'd)

**Action 41 : CLOSE FILE**

**Definition:** Close files R, RC, RC', etc. The type of each file is defined by L, L0, L0', etc. as follows:

- 0 = Output file
- 1 = Input file

**Notes:**

(i) All files opened by a programme must be closed using this action before the 'unload' instruction (action 151) is reached.

(ii) There may be any number of continuation lines, or none at all.

(iii) For magnetic tape files only, L may be 2 or 3 instead of 0 or 1. In this case the action will be interpreted as 'close reel', rather than 'close file', as follows:

- L = 2: close output reel
- L = 3: close input reel

**Action 42 : INPUT**

**Definition:** Input block from file R to section R0.

**Notes:**

(i) Standard block checks are performed.

(ii) An input instruction may only be given on a file which is currently being used for input.

(iii) The action can have an optional second C-line, where NC2 gives a procedure start address: this is used in conjunction with PT, card or MT input to detect a doubtful block and deal with it by program. When the Master Program detects doubtful block and the second C-line is not present, the normal routines are obeyed; if the C-line is present, then:

(a) MT: the block is re-read up to 6 times, returning to NC2 with binary radix and registers A,B indeterminate, the input section containing the corrupt block (unless one of the 6 attempts has read the block successfully). Corrupt MT characters have bit 8 = 1, and the BE word should be replaced before output.

Amendment No. 28
November 1964
(b) PT: returns to programme with the section containing the corrupt block, (A, B, C) as above. Note that DB usually occurs because of a parity error, and only complete 5-character words are transferred between the assembler and the store; hence the corrupt character will not be present in the section, and nor will the BE character be present (unless left in the sections from a previous read order). The next read order will begin at the character following the one out of parity.

(c) Card: returns to programme with the section containing the corrupt card, but no indication as to which character is wrong, (A, B, C) as in (e). The next read order will read the next card.

This facility is intended mainly for PT or card input, to reduce operator intervention, and so as to read past a corrupt RT block (e.g., to patch a rerun of part of the tape into the original).

In case of DB processed this way the Master Programme will make a log comment of the form: 'DB (channel, route) (programme identity)'
Action 43 : OUTPUT

Definition: Output block to file R from section EC.

Note: (i) An output instruction may only be given on a file which is currently being used for output.

(ii) For magnetic tape files the action may have a second C-line, where NC2 = a procedure start. This is used to detect the 'advance warning of end' condition, to give the programmer some control on the division of an output file into reels which may be processed individually in the next programme of the suite. For an example, suppose that every 43 action on a certain file has NC2 given; then blocks 1 to X will be written normally, and following the writing of block X the 'advance warning of end' condition is detected by the Master Programme. When the 43 action is obeyed for the (X+1)th block, this will be written on the current output tape and control will be returned to the programme at NC2 with (A), (B) indeterminate and binary radix in register C. In procedure NC2, an end of reel block of the programmer's choosing is written by a normal 43 action, which will be block (X+2) on the output tape; then, using another normal 43 action, a start of reel block (X+3) will be written at the head of a new output tape, where the Master Programme has called automatically for the new tape when it was entered to write block (X+2). The 43 action with NC2 is then used for all further output on the second reel, until the 'advance warning of end' condition is again round.

Alternatively in procedure NC2, a 'close reel' order may be given (see action 41) to terminate the reel without a special block, or after the writing of block (X+4) the reel may be closed with a 41 action.

If no second C-line is used, the change of output reel will be automatic with no extra blocks introduced.
(iii) For non-standard PT output (route type 16), the action must have a second C-line, where NC2 = a procedure start. This is a combined warning of end and doubtful block link, such that if during the output of a block (which will be one character at a time) a character is punched doubtfully, control is returned after punching that character to the first instruction of procedure NC2 with (A), (B) indeterminate and (C) binary radix. The programmer will then output his own doubtful block mark with another 43 action, and repeat the output of the block containing the doubtful character with a third 43 action.

If warning of end occurs after having punched the current block, control is returned to the second instruction of procedure NC2, so that the programmer can punch his own end of real sentinel. A fuller description of route type 16 is given in section 12.5.
Action 44: SET/SELECT REEL COUNTER

Definition:
(a) \( L = 2 \) Set the least two significant quartets of \( A \) as the reel counter for file \( R \).
(b) \( L = 3 \) Select the current (2 digit) reel number of file \( R \) into \( A \).

Notes:
(i) The radix in \( C \) is immaterial, and is preserved during the action.
(ii) The file \( R \) may be WP for both actions, or PT for 'select counter' only.
(iii) Register \( B \) is zero after both actions, and register \( A \) is indeterminate after 'set reel counter'.
(iv) The reel counter is in decimal, and does not exceed 99.
(v) The reel counter is zero before a file is opened and zero after a file is closed. After a reel is closed, the counter is set at the reel number + 1. When a file is opened the Master Programme expects the reel given by the counter unless this is zero, when reel 1 is expected.

Note: This action is intended for use with Issue 4 or later Master Programmes after being translated by Issue 5 or later Translator.

Issues 1, 2, 3 Master interpret action 44 as 'open working file' whether translated by Issue 5 Translator or earlier issues.
Rewind file R. L defines the use of the
file (before rewind) as follows:-
0 = Output file
1 = Input file

Notes:
1. On output files, a standard 'File End'
   procedure is carried out.
2. After rewinding a file must be
   re-opened before use.
3. The action can be used for magnetic
   tape files only.
**Action 46 : RUN BACK**

**Definition:**
Run to specified block containing alignment mark on file R. I indicates which word in the block contains the identifier (4, 6, 8, etc). \( L_1 \), or \( (N_1) \) if \( L_1 \) = zero, specifies the identifier, as 5 numeric characters. I defines the current use of the file as follows:

- 0 = Output
- 1 = Input

**Notes:**

(i) When running back, the first block to be scanned is block \( x \), where \( x \) is the last block written or made available to the program.

(ii) Before running back on an output file, a FINIS block and end of file and an END block are written.

(iii) After the action, the tape is aligned such that an output instruction will overwrite the block following the specified block or an input instruction will make available the specified block.

(iv) The alignment mark consists of at least four characters with control quartet 6 and basic quartet 15, occupying consecutive character positions within any word in the block.

(v) The identifier must be numeric, positive and non-zero.

---

Amendment No. 7
February, 1963
(vi) The identifier may itself contain the alignment mark, or it may be in a separate word.

(vii) A series of identifiers must be written in ascending order, though not necessarily consecutively.

(viii) Blocks containing identifier zero are ignored.

(ix) It is possible to arrange for several series of identifiers throughout a file, different series occupying different positions in the block as specified by I. If no member of a set of identifiers is required in a block containing an alignment mark, the associated word of the block must be zero.

(x) Where convenient, identifiers should have unique numeric values within their specified words for all blocks in the file.

(xi) For use of magnetic tape files only.

---

Amendment No. 7
February, 1963
Run to specified block containing alignment mark on file B. I indicates which word in the block contains the identifier (4, 6, 8 etc). LC₁ or (NC₁) if LC₁ = zero, specifies the identifier, as 5 numeric characters.

(i) This action may only be used when the current use of the file is input.

(ii) When running on the first block to be scanned is block x+2 where x is the last block made available to the programme or written. Block x+1 will only be scanned after a change of direction has taken place.

(iii) After the action the tape is aligned such that an output instruction will overwrite the block following the specified block or an input instruction will make available the specified block.

(iv) The alignment mark consists of at least four characters with control quartet 6 and basic quartet 15, occupying consecutive character positions within any word in the block.

(v) The identifier must be numeric, positive and non-zero.

Amendment No. 7
February, 1963
(vi) The identifier may itself contain the alignment mark, or it may be in a separate word.

(vii) A series of identifiers must be written in ascending order, though not necessarily consecutively.

(viii) Blocks containing identifier zero are ignored.

(ix) It is possible to arrange for several series of identifiers throughout a file, different series occupying different positions in the block as specified by I. If no member of a set of identifiers is required in a block containing an alignment mark, the associated word of the block must be zero.

(x) Where convenient, identifiers should have unique numeric values within their specified words for all blocks in the file.

(xi) For use of magnetic tape files only.

Amendment No.7
February, 1963
**Action 48 : STEP BACK**

**Definition:**
Step back $LC$, blocks on file $R$; if $LC = 0$, step the number of blocks indicated by $(NC)$.
$LC$, or $(NC)$, must be in decimal. $L$ defines the
current and future use of the file as follows:

- $0 = $Output file to be used for output;
- $1 = $Input file to be used for input;
- $2 = $Output file to be used for input;
- $3 = $Input file to be used for output.

**Notes:**

(i) Before stepping back on an output file, a
FINISH block, an end of file sentinel and
an (END) block are written.

(ii) After the action the tape is aligned so that
an output instruction will overwrite, or an
input instruction will make available, block
$x+n$, where $x =$ the block last written or
made available, and $n =$ the number of blocks
to be stepped.

(iii) When $L = 3$, $LC$, or $(NC)$ may be zero. This
will be interpreted as a change of function
of the file. In no other circumstances may
$LC$, or $(NC)$, be zero. Under no circumstances
at all may they be negative.
Action 43: STEP FORWARD

Definition: Step forward LC blocks on file A; if LC = zero, step the number of blocks specified by (NC), LC, or (NO), must be in decimal. L defines the current use of the file as follows:

0 = Input file to be used for output;
1 = Input file to be used for input.

Notes:
(i) This action may only be used when the current use of the file is input.
(ii) After the action the tape is aligned so that an output instruction will overwrite, or an input instruction will make available block x+n+1, where x = the last block made available, and n = the number of blocks to be stepped.
(iii) Under no circumstances at all may LC, or (NC), be negative or zero.

Amendment No. 18
September 1963
Group 5: REARRANGEMENTS

Notes: Actions 52 (edit) and 53 (condense) represent all characters in a 6-bit code; actions 50 and 51 (unpack) assume that they are in an 8-bit code. An input assembler extends the 6-bit code to an 8-bit code. Special care must be taken to ensure that the data is in the correct form for comparison. In particular it is necessary to bear in mind that unpack variable expects an 8-bit code for line end and number end.

Action 50: UNPACK FIXED FIELD DATA

When data is read from cards or other fixed field media, the characters are stored in sets of five in successive long words in the transit area. This action unpacks the various items (which are of specified length) and stores them in specified compartments, thereby making them available for normal use.

Definition: Unpack data held in alpha form in a fixed field format in long compartments starting at N, as specified by the table starting at NC, until the last table entry is reached.

Notes:
1. Tables are described in section 6.
2. All unpacked items are aligned to the least significant end of their destination compartment.
3. If any item is too long for its destination compartment the more significant characters are lost.
4. Registers A and B are left clear.
5. The radix set in C is immaterial.

Modification types:
First line: 2
Continuation line: 2
**Action 51: UNPACK VARIABLE FIELD DATA**

When data is read from paper tape or variable field magnetic tape, the characters are stored in sets of five in successive long words in the transit area. The items are variable in length and are separated by 'number end' characters. This action unpacks items held in this form and stores them in specified locations, thereby making them available for normal use.

**Definition:**
Unpack data held in alpha form in a variable field format in long locations starting at N, as specified by the table starting at N01, until either the last table entry is reached or a line end or block end character is reached.

**Notes:**

i. Tables are described in section 6.

ii. An item unpacked to alpha form is aligned to the most significant end of its destination compartment. An item unpacked to numeric form is aligned to the less significant end of its destination compartment.

iii. If any item is too long for its destination compartment the following rules apply
   a. **Alpha items** The first five characters are unpacked to the specified compartment, the next five to the next long compartment and so on.
   b. **Long numeric items** The first ten characters are unpacked to the specified compartment, the next end to the next long compartment, and so on.
   c. **Short numeric items** The least significant five characters of the first set of ten are unpacked to the specified compartment, the least significant five of the next set of ten to the next short
compartment, and so on. The more significant
five characters of each set of ten are
ignored. If an item does not consist of a
multiple of ten characters the last 'set' will
consist of less than ten characters. The
least significant five characters of this set
are unpacked and the remaining (more sig-
nificant) characters are ignored.

If items which 'overflow' in this way are unpacked
using one table entry for each item (see section
6.4.4) no change is made to the destination
addresses of later items. If items which 'over-
flow' are being unpacked 'to Block End' into
consecutive compartments, using a common table
entry, destinations of later items are adjusted
to avoid overwriting the 'overflow'.

(iv) If a line end character has terminated the
unpacking process, one (or more) further
unpack instructions will be required to deal
with the rest of the block. These further
instructions are exactly the same as the
initial instruction, except that a plus sign
is written in the item column in place of
the item of the source address (i.e. in the
item column of the first line). If the
unpacking process is to begin again, not
immediately but from k long words further
on in the source area, the entry in the item
column should be k followed by a plus sign.
The remaining characters in the word contain-
ing the line end will be ignored, and unpacking
will restart with the most significant character
of the (k+1)th word following that containing
the line end.

(v) At the end of the action Register A contains
the location after the last location from
which data was unpacked, in 15-bit binary
form.
Action 51 (Cont'd) (vi) Register B is left clear.

(vii) The radix set in C is immaterial.

(viii) When unpacking alpha to numeric, a question mark (5/13) will be recognised as a number end.

(ix) When unpacking alpha to numeric, a control quartet of 5 in the 10th character position of an item (excluding signs) will be treated as a negative sign.

(x) When unpacking alpha to numeric, a line end character will not be recognised if it follows ten other characters (excluding signs).

(xi) This action expects information to be in full alpha character form. It will not recognise block end and number end characters in the six-bit form produced by Action 53 (condense) unless such information has since been input from an external medium.

(xii) Whenever a character with control quartet 1, 2, 5, 9, 10, 13, 14 is discovered on unpacking numeric items, A is set negative and the basic quartet ignored.

Modification types:

First line: 2
Continuation line: 2
Action 27: EDIT

Blocks of results to be output on fixed field media (e.g. cards) are held as sets of five characters in successive long words in the transit area, the various items being of specified length. This action 'packs up' items from specified compartments into this form.

Definition: Edit data into a fixed field format in long compartments starting at % as specified by the table starting at NC, until the last table entry is reached.

Notes:
(i) Zeros, spaces, and sign characters are inserted as specified by the table (see section 6).
(ii) Registers A and B are left clear.
(iii) The radix set in C is immaterial.

Modification types:
First line: 2
Continuation line: 2
Action 21: CONDENSE

Blocks of results to be output on paper tape or variable field magnetic tape are held as sets of five characters in successive long words in the transit area. The items are variable in length and are separated by 'blank end' characters. This action 'picks up' items from specified locations into this form.

Definitions

Condense data into a variable field format in long locations starting at N, as specified by the table starting at NO, until the last table entry is reached. Non-significant zeros are deleted and blank ends are inserted after each item.

Notes:

(i) Tables are described in section 6.

(ii) A block will in general be produced by several CONDENSE instructions. The second of these and all subsequent ones must have the entry "a" in the item column of the first line (see note (iv)).

(iii) Block and character or words are not inserted by the action automatically; when required they must be added using the "a" facility (see section 4.3.1).

(iv) The first character produced by a 'CONDENSE' action is placed at the beginning of the next free word in the transit area, not in the next free character position. If the CONDENSE process is to begin again not in the next free word but k short words further on, the entry in the item

Amendment No.10

Section 5 (Cont'd)

Action 53 (Cont'd) column should be followed by a plus sign.

(v) At the end of the action Register A contains the compartment after the last compartment into which data was condensed in 15-bit binary form.

(vi) Register B is left clear.

(vii) The radix set in C is immaterial.

(viii) This action is only available on computers with relevant computer code facilities.

(ix) This action must be followed by an output instruction. The information cannot be unpacked as the condense action packs the information as six-bit characters and therefore Number End and Block End characters will not be recognized by the unpack action. Subsequent input will restore all characters to their correct octet form for unpacking.

(x) If -0 is condensed overflow occurs.

Modification type: First line 2
Continuation line 2

Action 54 : BULK CLEAR

Definition: Clear compartments from N onwards. The number of compartments to be cleared is specified by L in decimal if L ≠ 0, or by (NC) in binary if L = 0.

Notes: (i) If D = 0, short words are cleared; if D ≠ 0, words are cleared long, but L or (NC) still specify the quantity in short words (cf. action 55 Bulk Copy). A maximum of 4095 words can be cleared long.

(ii) The radix set in C is immaterial.

(iii) Registers A, B are cleared by the action.

Modification type: First line 2
Continuation line 1
Action 55: BULK COPY

Definition:
Copy the contents of locations M onwards to locations NO\(_1\) onwards. If L is non-zero it indicates, in decimal, the number of items to be copied in short words relative to the source. If L is zero, (NO\(_2\)), in binary gives the number of items to be copied in short words relative to the source.
A copy from alpha form is indicated by a non-zero discriminant on the first line.
A copy from numeric form is indicated by a zero discriminant on the first line.
A copy to alpha form is indicated by a non-zero discriminant on the second line. A copy to numeric form is indicated by a zero discriminant on the second line.

(i) The radix set in C is immaterial.
(ii) Registers A and B are left clear
(iii) The maximum number of items which can be copied is 4095. (If these are in alpha form they will be specified as 8190 short words)
(iv) (NO\(_2\)) must not be zero or negative.

Modification type:
First and second line 2
Third line 1

Amendment No. 10
March, 1963
Section 5 (Cont'd)

**Action 56 : TABLE LOOK-UP**

**Definition:** Search locations from \( N \) (modified) onwards long or short as indicated by the discriminant, until location \( P \) is found such that \( (P_xA) \) then place \( P - N \) (modified), i.e. (number of locations examined) \(-1\), in \( A \) in binary. \( A \) will be zero if the first location examined satisfies the condition.

**Notes:**

(i) The radix set in \( C \) is immaterial.

(ii) Register \( B \) is left clear.

(iii) Register \( A \) must not be negative at the start of the action, and overflow may occur if the locations examined in the table are negative.

(iv) The reference must not be a transit area.

**Modification Type:** 2
Action 57: MERGE CONSTANT LENGTH ITEMS

This action deals with blocks of data in the following form:

<table>
<thead>
<tr>
<th>FIRST ITEM of k long words</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S )</td>
</tr>
<tr>
<td>( S + 2k - 2 )</td>
</tr>
<tr>
<td>( S + 2k )</td>
</tr>
<tr>
<td>( S + 4k - 2 )</td>
</tr>
<tr>
<td>( S + 4k )</td>
</tr>
<tr>
<td>( S + 6k - 2 )</td>
</tr>
<tr>
<td>( S + 2(n-1)k - 2 )</td>
</tr>
<tr>
<td>( S + 2nk - 2 )</td>
</tr>
</tbody>
</table>

The block consists of \( n \) consecutive items, each of \( k \) consecutive words, held in long locations \( S \) to \( S + 2nk - 2 \) inclusive.

Such a block is said to be specified by a modification register if \( S \) is held in binary in the modifier (less significant half) and \( S + 2nk \) is held in binary in the more significant half.

**Definition:** Merge the blocks specified by modification registers 1 and 2 into a block as specified by modification register 3. Items (one from each of the two 'input' blocks) are compared by binary subtraction of the first word in each; the item with the smaller word is copied to the 'output' block. The comparison process is then repeated, replacing the copied item by its successor in its 'input' block. If the first word in each item is the same, the second word is examined, and so on. If the two items are found to be completely identical, the one
Definition: from the block governed by modification register 1 is copied to the output block. The action ends when either of the two 'input' blocks is exhausted or the 'output' block is full. The item length as an even number of short words (2k) is specified by L (in decimal), or by (L) (in binary) if L is zero.

Notes:
(i) Modification registers may be set by Action 59.
(ii) At the end of the action register A contains (minus the number of short words remaining unused in the 'output' block) in binary.
(iii) Register B is left clear.
(iv) Modification registers 1 and 2 are left set up to deal with the items remaining in the 'input' blocks, if any.
(v) Modification register 3 is left set up to deal with space remaining in the 'output' block, if any.
(vi) The radix set in C on entry to this action is immaterial, and the original contents are replaced in C at the end of the action.
(vii) If an item length of zero is specified the action enters a micro-loop.
(viii) The action is available only on computers with relevant computer code facilities.

Modification type: Non-modifiable
MERE VARIABLE LENGTH ITEMS

This action deals with blocks of data in the following form:

- Long locations

<table>
<thead>
<tr>
<th>ITEM</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIRST ITEM</td>
<td>of k₁ long words</td>
</tr>
<tr>
<td></td>
<td>First long word contains 2k₁ in binary-alpha form (see below)</td>
</tr>
<tr>
<td>SECOND ITEM</td>
<td>of k₂ long words</td>
</tr>
<tr>
<td></td>
<td>First long word contains 2k₂ in binary-alpha form</td>
</tr>
<tr>
<td>THIRD ITEM</td>
<td>of k₃ long words</td>
</tr>
<tr>
<td></td>
<td>First long word contains 2k₃ in binary-alpha form</td>
</tr>
<tr>
<td>NTH ITEM</td>
<td>of kₙ long words</td>
</tr>
<tr>
<td></td>
<td>First long word contains 2kₙ in binary-alpha form</td>
</tr>
</tbody>
</table>

The block consists of n consecutive items of variable length. The length of each is specified by its first word, in binary-alpha form. (Binary-alpha form is that form which when converted from alpha to numeric gives a binary result; i.e., it has 4's in the even quartets and the number in binary in the odd quartets).

Such a block is said to be specified by a modification register if S is held in binary in the modifier (less significant half) and k₂ is held in binary in the more significant half.
Definition: Merge the blocks specified by modification registers 1 and 2 into a block as specified by modification register 3. Items (one for each of the two 'input' blocks) are compared by binary subtraction of the second word in each; the item with the smaller word is copied to the 'output' block. (Note: The first word in each item contains the item length and is therefore not used in the comparison). The comparison process is then repeated, replacing the copied item by its successor in its 'input' block. If the second word in each item is the same, the third word is examined, and so on. If the two items are identical so far when one (or both) of them is exhausted, the one from the block governed by modification register 1 is copied to the output block. The action ends when either of the two 'input' blocks is exhausted or the 'output' block is full.

Notes:
(i) Modification registers may be set by action 59.
(ii) At the end of the action register A contains (minus the number of short words remaining unused in the 'output' block) in binary.
(iii) Register B is left clear.
(iv) Modification registers 1 and 2 are left set up to deal with the items remaining in the 'input' blocks, if any.
Notes:

(v) Modification register 3 is left set up to deal with the space remaining in the 'output' block, if any.

(vi) The radix set in C on entry to this action is immaterial, and the original contents are replaced in C at the end of the action.

(vii) If an item length of zero is specified the action enters a micro-loop.

(viii) The action is available only on computers with relevant computer code facilities.

Modification type: Non-modifiable

Action 59: SET MODIFICATION REGISTER FOR MERGE

Definition: Set modification register M for merging. N and N+(NC) are set as first and (last+2) long locations of block, so that (NC) (in binary) gives length of block as an even number of short words.

Notes:

(i) This action is normally used to set modification registers before merging using actions 57 or 58.

(ii) The action can be used on all modification registers, although its principal use is for registers 1 to 3.

(iii) Registers A and B remain unchanged.

(iv) The radix set in C is immaterial.

Modification type: Non-modifiable

Amendment No. 29
December 1964
Group 6: RADIUS FACILITIES

Action 60: SET DECIMAL RADIX
Definition: Set C for decimal (i.e., as 6.6.6.6.).
Note: A 60 action will be inserted by the
Translator at the start of a programme
Modification type: Non-modifiable.

Action 61: SET STERLING RADIX
Definition: Set C for sterling (i.e., as 6.6.14.6.4)
Modification type: Non-modifiable.

Action 62: SET EXCESS CONSTANTS
Definition: Set (C) at L.
Note: L (which must not exceed five digits)
consists of the complements with respect
16 of the required radix for five digit
positions; e.g., to set radix for pounds,
shillings, pence and halfpennies L would be
entered as 6 14 6 4 14
Modification type: Non-modifiable.

Action 63: CONVERT DECIMAL TO STERLING
Definition: If N is non-zero, convert (N) from decimal
to sterling and place in A.
If N is zero, convert (A) from decimal to
sterling.
Note: i. Sterling radix is set by the action
ii. Register B is left clear
Action 54: CONVERT STERLING TO DECIMAL

**Definition:**
If \( N \) is non-zero, convert \((N)\) from sterling to decimal and place in A.
If \( N \) is zero, convert \((A)\) from sterling to decimal.

**Notes:**
1. Decimal radix is set by the action.
2. Register B is left clear.

**Modification type:**
1

Action 55: CONVERT

**Definition:**
If \( N \) is non-zero, convert \((N)\) from radix defined by excess constants in L to radix defined by excess constants in L2, and place in A.
If \( N \) is zero, convert \((A)\) from radix defined by excess constants in L to radix defined by excess constants in L2.

**Notes:**
1. The new radix is set by the action.
2. Register B is left clear.

**Modification type:**
1
**Action 66: Set up for significant digit floating point arithmetic**

**Definition:**
Set $-37$ (binary) in compartment 132 of division 0, and set $-2^{19}$ (binary) in compartment 133 of division 0.

**Notes:**
1. A is left clear
2. Compartments 132 and 133 of division 0 are used by all floating point instructions to specify the mode of arithmetic.
3. Either this action or action 67 must be used before any floating point arithmetic is carried out.
4. The action is only available on computers with relevant computer code facilities.

**Modification type:** Non-modifiable.

---

**Action 67: Set up for normalised floating point arithmetic**

**Definition:**
Set $37$ (binary) in compartment 132 of division 0 and set $2^{19}$ (binary) in compartment 133 of division 0.

**Notes:**
1. After the action A contains $2^{39} \times 37$
2. Compartments 132 and 133 of division 0 are used by all floating point instructions to specify the mode of arithmetic.
3. Either this action or action 66 must be used before any floating point arithmetic is carried out.
4. The action is only available on computers with relevant computer code facilities.

**Modification type:** Non-modifiable.
Group 7: SEQUENCE CHANGES

**Actions 70 - 75: TESTS ON ACCUMULATOR**

**Definition:**
- 70 change sequence to N if (A) is zero.
- 71 change sequence to N if (A) is non-zero.
- 72 change sequence to N if (A) is positive or zero.
- 73 change sequence to N if (A) is negative.
- 74 change sequence to N if (AB) is zero.
- 75 change sequence to N if (AB) is non-zero.

**Modification type:** Non-modifiable.

**Action 76: UNCONDITIONAL SEQUENCE CHANGE**

**Definition:** Change sequence to N.

**Modification type:** Non-modifiable.

**Actions 77 - 79: TESTS ON FLOATING POINT ACCUMULATOR**

**Definition:**
- 77 change sequence to N if \((A^*)\) is equivalent to zero.
- 78 change sequence to N if \((A^*)\) is not equivalent to zero.
- 79 change sequence to N if \((A^*)\) is positive (including the case of the normalised floating point zero).

**Notes:**

1. \((A^*)\) is taken as zero if the exponent is less than \(L\), non-zero if the exponent is greater than or equal to \(L\). If \(L\) is zero the test is made using \((M^*)\).

2. \((A^*)\) is taken as positive if the mantissa is positive. This includes the case of the normalised floating point zero.
iii. Each digit of I is taken to represent one quartet. \((N^C_i)\) is treated as a binary number.

iv. After the action A holds (exponent of \(A^e - \) control value) for actions 77, 78 or the mantissa of \(A^e\) for action 79.

v. The actions are only available on computers with relevant computer code facilities.

Modification type: Non-modifiable.
**Action 81: SET SWITCH(es) AT LITERAL VALUE**

**Definition:**
Set switch(es) **R** at value **L**.

**Notes:**
1. A switch is always referred to by a procedure number which must be the number of some existing procedure, but not necessarily of that containing the switch.
2. Any number of switches may be referred to by the same procedure number and will be set simultaneously at the same value by one instruction.
3. A is left clear.

**Modification types:**
Non-modifiable.

**Action 84: SET SWITCH(es) AT VARIABLE VALUE**

**Definition:**
If no continuation line is present, set switch(es) **R** at (**A**), binary.
If a continuation line is present, set switch(es) **R** at (**NC, A**), binary.

**Notes:**
1. The relevant procedure number must be specified.
2. A is left clear.

**Modification types:**
First line non-modifiable
Continuation line 1

**Action 85: SWITCH **

**Definition:**
Change sequence to the instruction specified by **NC** *n*, where **n** is the value at which the switch is set.

**Notes:**
1. The identity of the switch, which must be a procedure number is specified by **R**. Any number of switches may have the same identity.
Action 66 (Cont'd)

Notes:

ii. Continuation lines 1 - n denote destination locations for switch settings at values 1 - n.

Modification type: Non-modifiable.

Action 66: SELECT SWITCH R

Definition: Select the setting of switch R into A.

Modification type: Non-modifiable.
**Action 87 : SWITCH ACCORDING TO N**

**Definition:** Change sequence to address specified by $N_n$ where $n$ is specified by $(N)$ in binary.

**Notes:**

(i) Actions 83, 84, 85 cannot be used to set or examine the switch.

(ii) The compartment (s) must contain a binary number greater than zero.

**Modification type:** 2
Section 5 (Cont'd)

Group 9 : MODIFICATION REGISTER CONTROLS

Notes: (i) The value added to the 15-bit address of a modified instruction is the 15-bit binary number in the less significant half of the appropriate modification register. Care should be taken when using absolute addresses in conjunction with computer code instructions.

(ii) A modification register when used for repetition is set initially thus:

| End value + repetition increment | Start value |

(iii) Numbers are held in modification registers in 15-bit binary complement form.

For other details of the modification system see section 11.

Action 90 : SET MODIFICATION REGISTER AT LITERAL

Definition: Place in modification register M the start value L and the (end value + repetition increment) LC,

Notes: (i) The translator converts the literals from decimal to binary and takes care of negative number requirements (see section 11).

(ii) The radix set in C is immaterial.

(iii) If M>3 register A is left clear otherwise it is unchanged.

Modification type: Non-modifiable

Amendment No. 16
September 1963
**Action 51: SET MODIFICATION REGISTER AT VARIABLE**

**Definition:**
If \( n \) is non-zero, place \( (s') \) in modification register \( M \).
If \( n \) is zero, place \( (A) \) in modification register \( M \).

**Notes:**
1. The radix set in \( C \) is immaterial.
2. \((W')\) or \((A)\) is treated as binary.
3. Negative number requirements are not taken care of by the translator.
4. If \( M\overline{0} \) register \( A \) is left clear, otherwise it is unchanged.

**Modification type:**
Non-modifiable.

**Action 52: SET MODIFIER AT VARIABLE**

**Definition:**
If \( n \) is non-zero, place \( (s) \) in the modifier (less significant half) of modification register \( M \).
If \( n \) is zero, place \( (A) \) in the modifier of modification register \( M \).

**Notes:**
1. The radix set in \( C \) is immaterial.
2. \((W)\) or \((A)\) is treated as binary.
3. Negative number requirements are not taken care of by the translator.
4. Register \( A \) is always left clear.

**Modification type:**
Non-modifiable.
**Action 13:** SET (N) as MODIFIER

**Definition:** Convert (N) from decimal to binary and place in the modifier (less significant half) of modification register M.

**Notes:**
1. The radix set in C is immaterial.
2. Negative number requirements (see section 11) are taken care of by the translator.
3. (a) is unchanged.

**Modification type:** Non-modifiable.

**Action 94:** STEP ON AND TEST MODIFICATION REGISTER (LITERAL)

**Definition:** Step on modifier in modification register M by L (converted from decimal to binary); if the modifier is not then equal to the end value, change sequence to N; otherwise obey the next instruction.

**Notes:**
1. If no sequence change address is specified, sequence change is assumed to be to the next instruction.
2. The modulus of the step value may not exceed 4095.
3. The sequence change may not refer to another chapter (see section 10).
4. The radix set in C is immaterial.
5. Negative number requirements (see section 11) are taken care of by the translator.

**Modification type:** Non-modifiable.
Action 95: SELECT MODIFICATION REGISTER

**Definition:**
Select the contents of modification register M into A.

**Notes:**
1. The binary pattern is placed in A.
2. The radix set in C is immaterial.

**Modification type:**
Non-modifiable.

Action 96: SELECT MODIFIER

**Definition:**
Select the modifier of modification register M into A.

**Notes:**
1. The binary pattern is placed in A.
2. The radix set in C is immaterial.

**Modification type:**

Action 97: STEP ON AND TEST MODIFICATION REGISTER (VARIABLE)

**Definition:**
Step on modifier of modification register M by (N), binary; if the modifier is not then equal to the end value, change sequence to NO; otherwise obey the next instruction.

**Notes:**
1. If no continuation line is present, sequence change is assumed to be to the next instruction.
2. The modulus of the step value may not exceed 4095.
3. The sequence change may not refer to another chapter (see section 10).
4. The radix set in C is immaterial.
5. Negative number requirements (see section 11) are not taken care of by the translator.
**Action 58: ADD MODIFICATION REGISTER TO A**

**Definition:** Add the contents of the word containing modification register M to register A.

**Note:** Discriminant 0 indicates modifier only. Discriminant 1 indicates modifier and end value.

**Modification type:** Non-modifiable.

---

**Action 59: SUBTRACT MODIFICATION REGISTER FROM A**

**Definition:** Subtract the contents of the word containing modification register M from register A.

**Note:** Discriminant 0 indicates modifier only. Discriminant 1 indicates modifier and end value.

**Modification type:** Non-modifiable.
Section 5 (Cont'd)

Intercode Actions 100 to 131

These are computer code actions, specified in Intercode by the addition of 100 to the computer code action number (see Volume I of the users' manual for action list).

Discriminants 0, 1 or 2 may be used, but the 'modification register' column may refer only to registers 1, 2 or 3. The programme will only be allocated a modification group if some Intercode action therein refers to registers 1 to 3.

A zero literal indicates that the action is to be translated as given, where the programmer is responsible for correct addressing of other chapters, transit areas, etc.

A non-zero literal indicates that the address of the instruction is to be obtained by a computer code expansion if necessary, as if the action were modification type 2. Q2, Q3 of the literal will also be interpreted as a decimal modification register, taking values from 0 to 9, whose contents are added to the register referred to in the 'mod' column.

The following instructions should be avoided, as being operationally undesirable besides interfering with the running of the Master Programme:

100/0/0  HALT COMPUTER (Intercode action 150 should be used instead).
101/0/3  INTERCHANGE AREA ADDRESSES (assumes tag '4 set in the arithmetic unit)
119/4/m  INPUT/OUTPUT (assumes tag 14 in arithmetic unit, and can invalidate the Master Programme records of 'state of device')
123/0/2  ENTER SUBROUTINE, SETTING TAG 14 IN ARITHMETIC UNIT (virtually ignoring lockout checks)
123/0/3  SELECT TAG OF N (assumes tag 14 in arithmetic unit)
123/1/m  SET TAG OF N AT (A) (assumes tag 14 in arithmetic unit)
124/1/1  SET MODIFICATION GROUP (of no value to the programmer, as he can never be allocated more than 1 modification group)
124/1/3  All operate on the indicator register, whose use is reserved exclusively for the Master Programme. All settings of the register by programme (except 25/1/4 4096 - set non-interruptible)
125/1/2  will be interpreted as operator commands, and all operator commands will first be intercepted by the Master Programme and hence cleared before accessed by programme. The action 125/1/2 0 does not use the indicator register, and is occasionally used by software for clearing register A.

Amendment No. 28
November 1964
Section 5 (Cont'd)

126/0/2 ENTER PRIORITY CONTROL (a form of 'enter subroutine', of no value to the programmer.

126/0/3 LEAVE SUBROUTINE, RESTORING PROGRAMME TAG (LEAVE MASTER PROGRAMMS). This normally operates on the subroutine link formed by the 25/0/2 action, and with that action is used in the Intercode 123 action expansion and some other software.
Group 14: DOUBLE LENGTH ARITHMETIC (only available on computers with relevant computer code facilities)

Note: In this group numbers such as \((N+2'), (N')\) in the store are manipulated as 20-digit numbers in sign and modulus form. The sign must be given by both the sign bit of \(N+2'\) and the sign bit of \(N'\), e.g., as set by actions 143 and 145. \((AB)\) is considered as a 20-digit number in sign and complement form. The sign must be given by the sign bit of \(A\) and the sign bit of \(B\) must be zero. For all the actions (except 140) the radix set in \(C\) must be uniform.

Action 140: CLEAR AB

Definition: Clear the double register AB.

Note: The radix set in \(C\) is immaterial.

Modification type: Non-modifiable

Action 141: DOUBLE LENGTH ADD

Definition: Add \((N+2'), (N')\) to \((AB)\)

Modification type: 1

Action 142: DOUBLE LENGTH SUBTRACT

Definition: Subtract \((N+2'), (N')\) from \((AB)\).

Modification type: 1

Action 143: DOUBLE LENGTH COPY

Definition: Copy \((AB)\) to \(N+2', N'\), setting the sign bits of both \(N+2'\) and \(N'\) equal to that of \(A\).

Note: \((AB)\) is unchanged.

Modification type: 1
Section 9 (Cont'd)

**Action 145 : DOUBLE LENGTH TRANSFER**

**Definition:** Transfer (AB) to N+2', N', setting the sign bits of both N+2' and N' equal to that of A

**Note:** AB is left clear.

**Modification type:** 1

**Action 146 : DOUBLE LENGTH MULTIPLY**

**Definition:** Multiply (N') by (NQ'), and place in AB.

**Notes:**
(i) (N') and (NQ') must both be in the uniform radix set in C.
(ii) (N') may be up to 9 digits.

**Modification type:** Both lines 1

**Action 147 : DOUBLE LENGTH MULTIPLY AND ADD**

**Definition:** Multiply (N') by (NQ') and add to (AB).

**Notes:**
(i) (N') and (NQ') must both be in the uniform radix set in C.
(ii) (N') may be up to 9 digits.

**Modification type:** Both lines 1

**Action 148 : DOUBLE LENGTH MULTIPLY AND SUBTRACT**

**Definition:** Multiply (N') by (NQ') and subtract from (AB).

**Notes:**
(i) (N') and (NQ') must both be in the uniform radix set in C.
(ii) (N') may be up to 9 digits.

**Modification type:** Both lines 1
Section 5 (Cont'd)

**Group 15 : MASTER ROUTINE CONTROLS**

**Note:** For all actions in this group the radix set in C is immaterial.

**Action 150 : OFFER OPTIONS**

**Definition:** Display the alarm number L. (For each alarm, the operator has one or more options). When option k has been chosen, change sequence to the instruction specified by NO. If L is zero then the alarm number is given by (M) decimal; N must be in the same chapter as the 150 action.

**Notes:**

(i) The alarm number L must be in the range 1 to 99 inclusive.

(ii) The number of options, and therefore the number of continuation lines, must be in the range 1 to 9 inclusive. Choice by the operator of option 10 allows him to delay his decision. In this case the options will be re-offered after 3 minutes.

(iii) The double register AB is left clear.

**Modification type:** Non-modifiable

**Action 151 : UNLOAD PROGRAMME**

**Definition:** Unload the programme from the computer.

**Note:** This must be the last instruction obeyed by any programme as it enables the store space and peripheral equipment to become available again, after operator action.

**Modification type:** Non-modifiable

Amendment No. 18
September 1963
Section 5 (Cont'd)

Action 152 : OVERLAY

Definition: Read in the chapter which starts with procedure R, and stack it in place of the chapter which starts with procedure R.

Notes:
(i) For description of chapters see section 10.
(ii) If a chapter which has already been in the store is being recalled using this action, it must start at the same point which it started at before.
(iii) In the programme as it is presented to the Translator, chapters to be placed in the store initially must precede those to be called later with the overlay action.
(iv) Chapters containing input/output sections may not be overlayed.

Modification type: Non-modifiable

Action 153 : SELECT DATE TO A, TIME TO B

Definition: The date appears as quartet values OOOOHHNNNY, i.e. 2 digits each for day, month, year in decimal; the time appears as quartet values OOOOHHNNNX, i.e. 2 digits for hours, 2 digits for minutes and one digit for tenths of minutes (stepped by 4 every 6 seconds) in decimal.

Notes:
(i) The radix in C is immaterial.
(ii) The time is taken from the Saxby clock (also used for log time) and is only available on computers where this is fitted; on other computers the Master Programe uses the compartments selected to B for a simulated clock, but this is in binary and bears no relationship to real time.
Action 154: COMMENT IN LOG

Definition: Write up to 60 characters of comment in log.

Notes:

(i) R, I indicate the start of an annexe containing up to 60 characters of comment. R may be a 3-digit reference or a section, and need not be a transit area.

(ii) The last word of the annexe must have its two least significant characters each "line end" (calling carriage return and line feed on the typewriter), occurring not later than the 59th and 60th characters in the annexe; if this is not so, the Master Program will replace the twelfth word of the annexe by 5 LE characters.

(iii) The Master Program will print the time and tabulate before writing the comment, so that all comments will have a similar format. The programmer must place sufficient information in his comment to enable the source program to be identified.

(iv) The characters NS, LE, BS and alignment mark are reserved for Master Program use (see the typewriter code in Volume 1 of User's Manual). All other characters may be typed, but the typewriter code may only correspond to the printer code for A-2 and A-9 depending upon the installation.
Section 9 (Cont'd)

Action 154 (Cont'd)

(v) The Master Program copies the anne translated word by word from the program area to its own area, for a maximum of 12 words. A request for use of the typewriter is set up, and control returns to the program. Typing proceeds concurrently with program running, and terminates when a word whose length is at least 15 significant characters is available, has been output.

Should the typewriter not be free when the Master Program is entered, the program is closed until the consent can be transferred.

(vi) Although sixty characters of log consent can be made, it is recommended that the consent be kept as short as possible, lest the machine become typewriter dependent.

Modification type: Non-modifiable

Action 155: SELECT PROGRAM DETERMINE TO A

Definition: The serial is selected as a 5-character alpha word, with the significant zeros as the character 5/C, i.e. suitable for printing.

Action 156: RELEASE ROUTE(S) ALLOCATED TO FILE R

Definition: This action, used when a file is no longer required in a program, causes the route or routes allocated to that file to be made available for allocation to another program. It may be given at any stage of file control, i.e. before opening, before closing, or after closing.

Notes:

(i) Any subsequent attempt by the program to use the released route will cause the program to be abandoned, with appropriate log consent.

(ii) This action can only be used when the program is running under a Master Program containing the facility (it is optional for each installation). Its primary use is for 80-column card input.

Amendment No. 34
October 1966
Action 157 : EXPAND

This is a variation of 'unpack variable', but with more superficial use of table entries. The data to be expanded is assumed to be in alpha form, and may include line end or block end characters (i.e. the data was assembled by using the 'condense' action).

**Definition:**
Unpack data held in alpha form in a variable field format in long locations starting at N, as specified by the table starting at RC, until either the last table entry is reached or a block end character is reached.

**Notes:**
(i) The form of the table is as for 'condense', i.e. the relevant columns are first reference, first item, second reference, second item, 'short/long/alpha' indicator, last table entry indicator. The setting of the 'sign' column does not affect the action.

(ii) The logic of the action is to unpack successive data separated by number ends into successive addresses beginning at the address specified by 'first reference/item', using the first table entry. The second table entry will be called if more data is to be unpacked than can be accommodated between the 'first' and 'last' addresses of the first table entry. When an L8 character is encountered this is unpacked according to the next table entry, and the subsequent table entry is used to unpack the data following the L8 character.

(iii) Where any item is too long for its destination compartment, the excess digits at the most significant end are lost.

(iv) Both alpha and numeric items are right-aligned in the destination compartment.

(v) The radix set in C is immaterial. At the end of the action, register B is clear and register A contains either:

(a) 2 if the action terminated on a block end character

or:

(b) the absolute location of the word after the last word from which data was unpacked if the action terminated at the 'last table entry' indicator.

Amendment No. 20
March 1965
Action 157 (Cont'd)

(vi) The 'item +' facility can be used in the source address, and at the end of the action the appropriate 'item +' counter is updated (as for actions 57, 53).

(vii) This action is available only on machines with the relevant computer code facility (action 29/0/3).

Modification Type: First line: 2
Continuation line: 2

Action 158: COMPARE

This action exists in two forms, and the literal is used to distinguish them.

(a) I=0

Definition: Compare sets $S_1$ and $S_2$ of alpha data; where $S_1$ starts at $N$ and register $A$ holds in QI-4 the absolute start of $S_2$. The comparison is performed by testing corresponding words of $S_1$ and $S_2$ in their pattern form (cf. merge actions 57, 58) if the start addresses are even. If one or both of the addresses is odd, the numeric quartets of the corresponding alpha data are used in the comparison. The action ends when a difference is found, and control is returned to the programme as follows:

Location Action
x 158
x+1 $76 \rightarrow S_2 > S_1$, i.e. obey next instruction.

x+2 $76 \rightarrow S_2 = S_1$, i.e. skip 1 computer code instruction

x+3 (as needed) $S_2 < S_1$, i.e. skip 2 computer code instructions

The number of words of $S_1$ and $S_2$ to be compared is given by Q6 of register A in binary, and hence cannot exceed 15. If it is zero, the action assumes that $S_1 = S_2$. Note that if the action ends when the counter in A is exhausted, this is also taken to signify that $S_1 = S_2$.

Amendment No. 30
March 1965
Section 5 (Cont'd)

Action 158 (Cont'd)

Notes:
(i) This action is only available on machines with the relevant computer code facility (Action 16/1/0).
(ii) In the example above, the Intercode instructions in locations x+1 and x+2 must have a 1-for-1 expansion, i.e. sequence changes to a position in the current chapter.
(iii) The quartets of A other than Q1-4 and Q6, do not affect the action.
(iv) The setting of register C is immaterial, and at the end of the action register D is clear and register A is unchanged.
(v) Overflow may occur if the alpha words of S1 or S2 are negative.

(b) Case 1:
This action is not yet completely defined. The locations of S1 and S2 are indicated by modification registers of the current modification group, and (b) indicates the amount of data to be compared. The exits from the action are different from case (a). It will be the computer code Action 16/1/1.

Amendment No. 30
March 1965