16. DATA HANDLING

16.1 File Control

The manner of input and output and the equipment used has been described in sections 9 to 15. This section describes the various computer actions which may be utilised in controlling the input and output equipment.

As well as the basic actions for the input and output of information other actions provide a preliminary check to ensure that the equipment is available for use. There are also several facilities for manipulating tape to a specified position.

16.1.1 Testing Route

Before any file manipulation takes place it is advisable to test the route concerned to ensure that abnormal conditions are not present. This is carried out by action 26.1.0 which tests for the following conditions in the order stated.

(i) Route closed
(ii) Route engaged
(iii) 'Doubtful block' condition set
(iv) 'Advance warning of end' condition set

and changes sequence as specified in the action definition below.

Unless the route is tested before an input or output instruction is given, the operator will not be informed automatically (by a comment typed in the log) that an abnormal condition exists, nor, if immediate input or output is impossible, will control be passed to another program.

Action 26.1.0 : TEST ROUTE

Definition: Test the route specified by N1 (literal)

(i) If the route is closed carry on to the next instruction.

(ii) If the route is not closed but is engaged set 'interrupt' condition on that route and skip one instruction.
(iii) If the route is neither closed nor engaged, but the 'doubtful block' condition is set, skip two instructions.

(iv) If the route is neither closed nor engaged, the 'doubtful block' condition is not set, but the 'advance warning of end' condition is set, skip three instructions.

(v) If the route is neither closed nor engaged and neither the 'doubtful block' condition nor the 'warning of end' condition is set, skip four instructions. (The route may be assumed to be available).

Notes:

(i) M specifies in binary the full route number, i.e. \((8 \times \text{channel number}) + \text{route number}\).
16.1.2 Interchanging Areas

During the running of a programme, processing is delayed whenever the input of a block of information is not complete when the information is required. These delays can be reduced by initiating input as far in advance as possible before the information is required. This is permitted by allocating two input areas to each route, and arranging for the programme to initiate input of a block to one area immediately before it starts to process the block already read into the other area.

The same arrangement is used for output. Two output areas are allocated to each route, and the output of a block from one area to the output equipment is initiated as soon as possible after it has been formed up in one area, the programme then forms up the next block in the other area.

Thus the two areas must perform their input (or output) function alternately. Since input and output always take place to or from an annex whose start location is specified by \((64 + R)\) - where \(R\) is the full route number - the start locations of the two areas allocated to route \(R\) must alternate with each other in compartment \(64 + R\). To achieve this, for each route a compartment \(64 + P\) is set aside where \(P\) is a number in the range 0 to 63 not currently being used as a route number. Initially the two area start locations are stacked, one in \(64 + R\), one in \(64 + P\). (This is normally done by the Master Programme).

An action \((1.0.3)\) is provided to interchange the contents of \(64 + R\) and \(64 + P\) each time input or output is initiated. (The 'interchange area addresses' action is placed after the 'test route' action but before input or output in the programme).

Thus \(64 + P\) always contains the start of the area not currently being used for input or output. The programme uses this area by means of modification by the contents of \(64 + P\).

**Action 1.0.3 : INTERCHANGE AREA ADDRESSES**

**Definition:** Interchange \((64 + N1)\) and \((64 + N2)\).

**Notes:**

(i) \(N1\) and \(N2\) specify in binary the values of \(R\) and \(P\) (see above).

(ii) When the Fast Channel control is fitted to the 326 machine, this action is not permissible.
16.1.3 Initiation of File Control Action

When the route has been tested and found available, and areas have been interchanged, the '9 action is used to send a signal to the coordinator to initiate movement of the appropriate file.

Action 19: FILE CONTROL

Definition: Test route N1 engaged. If engaged repeat the instruction unless the address of the instruction in the arithmetic unit is different from that in store, in which case, repeat the previous instruction (which in this case will be 24/0/0, 24/0/1, 24/0/3 or 24/1/2) also.

If not engaged, take action as follows:

Action 19.0.0 OUTPUT one block to route N1

Action 19.0.1 (milliseconds timer) RESET timer (on route N1) to zero.

(Other media) INPUT one block from route N1

Action 19.0.2 RUN BACK to next alignment mark on route N1 (see notes (vi) and (vii))

Action 19.0.3 RUN ON to next alignment mark on route N1 (see notes (vi) and (vii))

Action 19.1.0 STEP BACK one block on route N1 (see note (viii))

Action 19.1.1 REWIND route N1. Set route N1 to manual, rewind tape to position ready to read or overwrite first block, then open route N1.

Action 19.1.2 UNLOAD route N1. Set route N1 to manual, and rewind tape onto upper spool ready to remove from the deck.

These actions may be used only for magnetic tape routes
(magnetic tape)  STEP ON one block on route N1. Input the first word in the block, then step on to the next block end word without further transfer of information. The tape is left in position ready to read or overwrite the next block.

(other media)  SET ROUTE N1 TO MANUAL.

Notes:
   (i)  N1 specifies \((6 \times \text{channel number} + \text{route number})\) in binary.
   (ii) The location of the first long word of the annexe in the store must be specified by bits 1 to 15 of \((64 + N1)\), and the key to be set up in the assembler key register must be held in bits 17 to 20 of \((64 + N1)\) except for assemblers using a fixed location for annexe, e.g. paper tape punch, 24-hour clock and typewriter when \((64 + N1)\) is irrelevant.
   (iii) For assemblers referring to a function word, the location of the function word for route N1 must be given by \((64 + N1)\). The data to be output is held in the words following the function word.
   (iv)  REWIND and UNLOAD do not engage the assembler.
   (v)  A block end character on magnetic tape or paper tape input causes the assembler to cease reading. A block end character on output stops the writing process. For magnetic tape a special block end word is required, held in the last long word in the block. It contains block end characters in the first and fifth positions, while the three middle octets should be empty.

| Block end | Block end |

The three central characters are used to hold a six-bit track sum.

When the block end word is read into the store, the three middle characters are empty unless there is a track sum error, in which case a doubtful block condition is generated.
(vi) Alignment Blocks

Any block containing at least four 'alignment mark' (2/15) characters is an 'alignment block'. These characters should occupy consecutive locations in the same word.

The run on and run back actions stop on the next block end encountered after the four alignment marks. Thus when running on, the tape is correctly aligned to read or overwrite the block following the alignment block. When running back, however, the tape is aligned on the block end preceding the alignment block and is not correctly aligned to read or overwrite any block (see section 12). To read or overwrite the alignment block it is necessary to first step back, then step on before reading or writing (stepping on only, enables the block following the alignment block to be read or overwritten).

(vii) Reliability of Run Back

The reliability of this action is not guaranteed when alignment marks are within eighty words of a block end. In such circumstances the programme should check the alignment.

(viii) Step Back

The step back instruction does not leave the tape correctly aligned for reading/overwriting. To remedy this, it is necessary to step back one block further than required, then step on one block.

(ix) If any 19 action, not applying to magnetic tape, has a discriminant _1, the route is set to manual. If the discriminant is zero, an input or output instruction, as appropriate, is initiated.
16.2 Handling Input Data

16.2.1 Data Representation

There are two main forms in which data streams enter the computer. One form has a fixed number of characters in the stream to each item, giving rise to 'fixed field' data layouts in the store. The other form has unwanted zeros or spaces eliminated and the items separated by a special character (number end) giving rise to 'variable field' data layouts in the store.

Fixed field data is nearly always used on cards and often on magnetic tape. Variable field data is usually used on paper tape and often on magnetic tape. It is very rarely used on cards.

Whichever method is used the block will be stored in consecutive long compartments of the input area, five alpha characters in each compartment. To separate the items and stack them in separate compartments and, if required, to convert them to numeric form, one of the 'unpack' actions is used.

16.2.2 Unpack Actions

Two unpack actions are provided; one for fixed and one for variable field data. Each refers to a table of constants held in consecutive long compartments in the store, which describes the way each item is to be unpacked. Normally one table entry is held for each item in a block and consecutive table entries apply to consecutive items in the block.

Action 26.0.2 : UNPACK FIXED FIELD DATA

Definition: Unpack data held in alpha octet form in a fixed field layout in consecutive long compartments starting at N, according to the table held in consecutive long compartments starting at (A). Stop unpacking when a table entry bearing a 'last table entry' indication has been dealt with.
The layout of each table entry in its long compartment is as follows:

B1
Location in binary where item is to be stacked
B16
Q6 Number of characters in item (if this exceeds the capacity of the destination compartment it is the more significant characters which are lost).

B37 0 = item to be stacked in numeric form (see note on control quartets below) 1 = item to be stacked in alpha form

B38 0 = numeric item to be stacked in short word 1 = numeric item to be stacked in long word
(if B37 = 1 the value of B36 does not affect the action)

B39 0 = numeric item not to be signed 1 = numeric item to be signed
(if B37 = 1 the value of B39 does not affect the action)

S2 0 = not last table entry 1 = last table entry

The values of other bits in the table entry do not affect the action.

Notes:

(i) A and B are left clear.
(ii) All items are aligned by the least significant character after unpacking.
(iii) Interpretation of control quartets (in items to be stacked in numeric form)
   (a) First character of numeric item which is to be signed
      If the control quartet is 4p+1 or 4p+2 the sign bit of the destination is set negative.
      If the control quartet is 4p or 4p+3 it is ignored.
(b) All other characters

If the control quartet is $4p+1$ or $4p+2$
the destination character is set as $10$
or $11$ respectively; the basic quartet is ignored.

If the control quartet is $4p$ or $4p+3$ it is ignored.

(iv) The compartment after that containing the last
word to be unpacked must have a tag such that
access to it during the unpacking process will
not cause lockout (LSO III only).

Procedure:

The action is carried out as follows:

(i) (SC) are stored in compartment 1.

(ii) (A) are copied to SC.

(iii) The first table entry is selected into register
D from the address specified by (SC).

(iv) The words to be unpacked are selected in
sequence into B from the address specified by
(OA). Each time a word is to be selected (OA)
are increased by 2. The number of characters
specified in the table entry are shifted into A
in either alpha or numeric form as required.
If Q6 = 0 sixteen characters are unpacked.
Negative signs and '10' and '11' characters are
inserted where appropriate when the item is to
be stacked in numeric form. When A holds the
desired number of characters they are written
into the location specified in the table entry.

(v) (SC) are increased by 2.

(vi) The next table entry is selected into register
D from the address specified by (SC).

(vii) Steps (iv), (v) and (vi) are repeated until
the unpacking of an item whose table entry
bears a 'last table entry' indication has
been completed.

(viii) The original contents of SC are replaced and
A and B cleared.
Section 16.2 (Cont'd)

Action 28.0.3: UNPACK VARIABLE FIELD DATA

Definition:
Unpack data held in alpha octet form in a variable field layout separated by number ends in consecutive long compartments starting at N, according to the table held in consecutive long compartments starting at A. Stop unpacking when a line end or block end character is reached, or when a table entry bearing a 'last table entry' indication has been dealt with.

Table:
One table entry is normally held for each item in the block (unless bit 40 of a table entry is set - see below). The layout of each table entry in its long compartment is as follows:

B1  Location in binary where item is to be stacked.
B16
B37 0 = item to be stacked in numeric form (control quartets are ignored except in negative sign characters; for these the sign bit of the current destination is set negative).
1 = item to be stacked in alpha form.
B38 0 = numeric item to be stacked in short word form. (If such an item contains more than five digits, the least significant five digits of each set of ten - or part of a set - are stacked in successive short compartments beginning with the one specified; the remaining characters of each set of ten are lost).
1 = alpha item, or numeric item to be stacked in long word form. (If such an item contains more than five or ten digits as appropriate, the following words will be used as necessary).
B40 0 = stack one item.
1 = stack consecutive items in the format specified by this table entry in consecutive compartments until a line end or block end character is reached.
S2 0 = not last table entry.
1 = last table entry

The values of other bits in the table entry do not affect the action.

Amendment No. 8
December 1964
Notes:

(i) At the end of the action A contains the location of the last word unpacked; B is clear.

(ii) Alpha items are aligned by the most significant character after unpacking.

(iii) Numeric items are aligned by the least significant character after unpacking.

(iv) (LSE III) When unpacking alpha to numeric, 5/13 (question mark) is recognised as a 'number end'. A 'line end' is not recognised if it follows ten other characters.

(v) On LSE 326, the characters 13/14, 9/14, 5/14, and 1/14 will be treated as line end; the characters 15/14, 11/14, 7/14, 3/14 will be treated as block end.

(vi) When unpacking (alpha → numeric), 15/13, 11/13, 7/13, and 3/13 characters will be treated as number end, and 9/6 and 1/6 as a sign.

(vii) When unpacking (alpha) only 7/13 will be treated as number end.

Procedure:

The action is carried out as follows:

(i) (SC) are stored in compartment 1.

(ii) (A) are copied to SC.

(iii) The first table entry is selected into register D from the address specified by (SC).

(iv) The words to be unpacked are selected in sequence into B from the address specified by QA. Each time a word is to be selected QA is increased by 2. Characters are shifted into A in alpha or numeric form as specified by the table entry. Whenever a character with control quartet 1, 2, 6, 9, 10, 13, 14 is discovered on unpacking numeric items, A is set negative and the basic quartet ignored. When a number end, line end or block end character is reached, (A) is written into the location specified by the table entry (after realigning to the more significant end if the data is unpacked in alpha form), step (v), (vi) or (vii) is then obeyed. A is then cleared.

(v) If a number end, line end or block end character was not discovered in step (iv) the table entry in D is increased by 2 and step (iv) repeated.
(vi) If a number end was discovered in step (iv), action is taken as follows:

(a) If the table entry in D has bit 40 set the table entry is increased by 2 and step (iv) repeated.

(b) If the table entry has a 'last table entry' indication (except in case (a) above), the original contents of SC are replaced, (OA) are copied to A, B is cleared and the action ends.

(c) If neither case (a) nor case (b) applies, (SC) are increased by 2 and the next table entry selected into D from the address specified by (SC). Step (iv) is then repeated.

(vii) If a line end or block end was discovered in step (iv), the original contents of SC are replaced, (OA) are copied to A, B is cleared and the action ends.

**Action 29.0.3 : EXPAND**

*Available on LEO 326 only*

**Definition:** Expand data in the store from 'condensed' form into its original form, according to the table held in consecutive long compartments starting at (A). Stop expanding when a last table entry or a block end character has been encountered.

**Table:** Layout of table entries is as given for CONDENSE (section 16.3.2).

**Notes:**

(i) The data is in alphanumeric form with number end characters separating successive words, and may have line end characters separating successive sub-records.

(ii) When a line end character is encountered, the next table entry is obtained. The data specified by that table entry is then unpacked in the normal manner.
16.3 Preparing Data for Output

16.3.1 Data Representation

As in input, data for output may be in either fixed or variable field form, and must be formed up by programme using either 'edit' or 'condense' actions. Fixed field form is always used for punched cards, usually for printer, and often for magnetic tape, but rarely for paper tape.

Variable field form is usually used for paper tape, often for magnetic tape, rarely for printer and punched cards.

16.3.2 Edit and Condense Actions

The following actions are the output equivalent of the two unpack actions. They use tables of a similar type to those used by the unpack actions.

**Action 26.1.2 : EDIT FIXED FIELD FORMATS**

**Definition:** Edit items into a fixed field format in alpha sextet form in consecutive compartments starting at N, according to the table held in consecutive long compartments starting at (A). Stop editing when a table entry bearing a 'last table entry' indication has been dealt with.

**Table:** The layout of each table entry in its long compartment is as follows.

<table>
<thead>
<tr>
<th>B1</th>
<th>Location in binary of item to be edited.</th>
</tr>
</thead>
<tbody>
<tr>
<td>B16</td>
<td></td>
</tr>
</tbody>
</table>
| B17 | 0 = normal
|     | 1 = if last item edited has significant digits, suppress 1 non-significant zero for this item, otherwise suppress all non-significant zeros for this item. (The contents of B21 to B24 must be the same as those of B33 to B36. The facility is used for e.g. shillings). |
| Q6  | Number of characters of item to be edited. |
Section 16.3.2 (Cont'd)

Q7  Number of space characters to be inserted before item.

Q8  Number of characters of word containing item to be discarded (counting from more significant end).

Q9  Number of characters of item for which non-significant zeros are to be replaced by space characters.

B37  0 = item is in numeric form.
     1 = item is long word in alpha form (the sign bit will be ignored).

B38  0 = numeric item is short word.
     1 = numeric item is long word.
     (If B37 = 1 the value of B38 does not affect the action).

B39  0 = alpha item, or numeric item for which no sign is required.
     1 = numeric item which is to be signed.

B40  0 = alpha item or numeric item which is to be signed in variable field notation (i.e., with a negative sign or space character following the last numeric character). This notation is also used for printer output.
     1 = numeric item which is to be signed in fixed field notation (i.e., with a ' in the control quartet of the first character of negative items). This notation is not used for printer output.

S2  0 = not last table entry.
    1 = last table entry.

The values of the other bits in the table entry do not affect the action.

Notes:

(1) To ensure correct alignment of the last word stacked, the total number of characters edited by this action (including sign characters if any) must be a multiple of 5. (If it is not it will be made up to a multiple of 5 by the introduction of spurious characters at the more significant end of the last word).
Section 16.3.2 (Cont'd)

(ii) Block ends are not inserted automatically by
the action.

Procedure: The action is carried out as follows:

(i) (SC) are stored in compartment 1.

(ii) (A) are copied to SC and A is cleared.

(iii) The first table entry is selected into register
D from the address specified by (SC).

(iv) As many space characters are inserted into A as
specified by Q7 of D. The item is then read into
B (short word items are read into the most
significant five quartets of B), and as many
digits or characters are discarded from the most
significant end of B as are specified by Q8 of D.
As many characters are then shifted into A in
alpha sextet form as are specified by Q6 of D.
If Q6 = 0 and B37 = 0, 16 characters are edited.
If Q6 = 0 and B37 = 1 no characters are edited.
For numeric items all control quartets become
zero except for significant zeros for which the
control quartet is 3. The introduction of
significant zeros begins either as soon as the
number of consecutive zero digits shifted into A
equals the number specified by Q9 or after the
first non-zero digit is shifted into A, whichever
is the sooner.

Negative signs are inserted as appropriate. (A)
are written to the address specified by (QA)
whenever A holds 5 characters and (QA) are
increased by 2.

(v) (SC) are increased by 2.

(vi) The next table entry is selected into register D
from the address specified by (SC).
Section 16.3.2 (Cont'd)

(vii) Steps (iv), (v) and (vi) are repeated until the editing of an item whose table entry bears a 'last table entry' indication has been completed.

(viii) The original contents of SC are replaced and A and B cleared.

**Action 28.1.3 : CONDENSE**

**Definition:** Condense items into a variable field layout in alpha sextet form in consecutive compartments starting at N, inserting number end characters after each item, according to the table held in consecutive long compartments starting at (A). Stop condensing when a table entry bearing a 'last table entry' indication has been dealt with.

**Table:** The layout of each table entry in its long compartment is as follows:

B1 Location in binary of first item to which this entry applies
B20 Location in binary of last item to which this entry applies
B37 0 = item is in numeric form
     1 = item is in alpha form (space characters are not deleted unless they fill a complete word)
B36 0 = numeric item is in short word
     1 = alpha or numeric item is in long word
B39 0 = item is to be without sign
     1 = item is to be signed
S2 0 = not last table entry
    1 = last table entry

The value of S40 in the table entry does not affect the action.
Notes:
(i) At the end of the action A contains the location of the next free compartment in the destination area. and B and C are clear.
(ii) The action does not delete non-significant spaces or zeros from alpha items unless a whole word consists of space characters.
(iii) On LEO III, overflow occurs if $0$ is condensed. On LEO 326, it will be condensed as a negative sign character and a number end (if sign requested in the table entry) and B will be clear at the end of the action.
(iv) If the contents of B1 to 20 are greater than those of B21 to 36, no condensing takes place for that table entry.

Procedure:
The action is carried out as follows:
(i) (SC) are stored in compartment 1.
(ii) (C) are stored in compartment 32.
(iii) (A) are copied to C.
(iv) The first table entry is selected into register D from the address specified by (C). The 'first item address' part of the table entry is copied to SC.

The contents of the first compartment to be condensed are selected from the location specified by (SC).

If the item is numeric all non-significant zeros are discarded and the item shifted into A in alpha sextet form. If the item is negative and a sign is required a negative sign is formed and shifted into A. A number end character is then formed and shifted in A.

If the item is alpha it is tested to see if it consists of space characters only. If not the whole item is shifted into A. A number end is then formed and shifted into A as for numeric items.

The contents of A are written to the address specified by (0A) whenever A holds 5 characters, and (0A) increased by 2.

Amendment No. 8
December 1964
Section 16.3.2 (Cont'd)

(v) (SC) are increased by 1 or 2 according to the setting of bit 38 or (D).

(vi) Steps (iv) and (v) are repeated until an item has been dealt with whose location (as specified by (SC)) equals the 'last item address' in the table entry (in register D).

(vii) (C) are increased by 2.

(viii) The next table entry is selected into register D from the address specified by (C).

(ix) Steps (iv), (v), (vi), (vii) and (viii) are repeated until the condensing of items whose table entry bears a 'last table entry' indication has been completed.

(x) The original contents of SC are replaced, (OA) are copied to A, and B is cleared.

Note:
If the last item condensed is negative, register B is set at -0 instead of 0 at the end of the action. If register B is actually required to be zero (e.g. for double length shifting), a 1/0/1/0 order must be inserted after the 'condense'.
Section 16.4

16.4 General Purpose Output Assembler Requirements

16.4.1 Because of the simplified way in which the General Purpose output assembler operates, information for punched card or printer output must be formed up by programme in a special way. This is done by using one of the special actions 280/0 or 280/1 after an ordinary edit actions 281/2.

16.4.2 General Purpose Output Actions

Two actions are provided - 290/0 for card output and 290/1 for high speed printer output.

Action 290.0 : EDIT FOR HOLLERITH OUTPUT

Definition: Clear long compartments N' to (N'+50) inclusive, and fill them with information for output as follows:

(i) Transfer the long word from compartment (A) to compartment N' (this is a function word).

(ii) Set bits in compartments N+4' to N+50' to correspond to the value of the most significant character of the compartment (A)+2', in accordance with the table below.

(iii) Repeat (ii) for the remaining characters of compartment (A)+2', and for successive characters of (A)+4' onwards.

(iv) Terminate (iii) when 80 characters have been dealt with, or when a block end character is encountered.

Table of Equivalent Values

The action treats compartments N+4' to N+50' as forming a matrix of 80 by 12 bits thus:

<table>
<thead>
<tr>
<th>Bit No.</th>
<th>80-</th>
<th>40-</th>
<th>12-</th>
</tr>
</thead>
<tbody>
<tr>
<td>N+ 6'</td>
<td>N+ 4'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N+10'</td>
<td>N+ 8'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N+50'</td>
<td>N+48'</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Successive pairs of long compartments are treated as containing 80 bit positions, as shown.

Equivalent values of the characters in compartments (A) to (Z) onwards are formed in this matrix according to the following table:

<table>
<thead>
<tr>
<th>Value of control quartet of nth character</th>
<th>Value 1 placed in nth bit position of:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No entry made</td>
</tr>
<tr>
<td></td>
<td>(N + 5') / (N + 4')</td>
</tr>
<tr>
<td>1</td>
<td>(N + 10') / (N + 8')</td>
</tr>
<tr>
<td>2</td>
<td>(N + 14') / (N + 12')</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value of basic quartet of nth character if</th>
<th>Value 1 placed in nth bit position of:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No entry made</td>
</tr>
<tr>
<td></td>
<td>(N + 18') / (N + 16')</td>
</tr>
<tr>
<td>1</td>
<td>(N + 22') / (N + 20')</td>
</tr>
<tr>
<td>2</td>
<td>(N + 26') / (N + 24')</td>
</tr>
<tr>
<td>3</td>
<td>(N + 30') / (N + 28')</td>
</tr>
<tr>
<td>4</td>
<td>(N + 34') / (N + 32')</td>
</tr>
<tr>
<td>5</td>
<td>(N + 38') / (N + 36')</td>
</tr>
<tr>
<td>6</td>
<td>(N + 42') / (N + 40')</td>
</tr>
<tr>
<td>7</td>
<td>(N + 46') / (N + 44')</td>
</tr>
<tr>
<td>8</td>
<td>(N + 50') / (N + 48')</td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

**Notes:**

(i) Characters 0/10 and 4/10 are edited as 1/0
Characters 0/11 and 4/11 are edited as 2/0

(ii) For LEO III/1, 2, 4 and 5 only, any character with basic quartet 9 causes overflow. When overflow occurs for this reason, the bit corresponding to the control quartet will already have been placed in the matrix. The block end character is the exception as it stops the editing but is not itself edited.

(iii) Overflow occurs if there are more than 80 characters before a block end character is encountered.
(iv) The sign bits of the source words are ignored.
(v) No entry is made in the matrix for space characters.
(vi) On completion of the action, B is clear and A contains the location of the last word edited.

Action 29.0.1: EDIT FOR ANLEX OUTPUT

Definition: Transfer the function word from long compartment (A) to compartment N'. Examine the function word in long compartment (A). Then:

(a) If bit 37 is zero:
   (i) Clear compartments N+2' to N+126'
   (ii) Set bits in compartments N+2' to N+126' to correspond to the value of the most significant character of compartment (A)+2, in accordance with the table below.
   (iii) Repeat (ii) for the remaining characters of (A)+2', and for successive characters of (A)+4' onwards until 40 characters have been dealt with.

(b) If bit 37 is set at 1, leave compartments N' to N+126' undisturbed.

(c) Repeat (a) or (b) according to the values of bits 38, 39 and 40 of the function word and where:
   bit 38 controls compartments N+130' to N+254'
   bit 39 controls compartments N+258' to N+382'
   bit 40 controls compartments N+386' to N+510'
   taking source characters from the successive compartments after those already dealt with in (a) (ii) and (a) (iii).
(d) Terminate the action when compartments N+384' to N+510' have been treated as in (a) or (b) or when a line end or block end character is encountered in the source.

**Table of Equivalent Values**

The action treats compartments N+2' to N+125', N+130' to N+254', N+258' to N+382', N+386' to N+510' as a matrix 160 bits by 63 bits thus:

<table>
<thead>
<tr>
<th>Bit No.</th>
<th>160-</th>
<th>-121</th>
<th>120-</th>
<th>-81</th>
<th>80-</th>
<th>-41</th>
<th>40-</th>
<th>-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>N+366</td>
<td>N+256'</td>
<td>N+130'</td>
<td>N+2'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N+368</td>
<td>N+260'</td>
<td>N+132'</td>
<td>N+4'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N+510</td>
<td>N+382'</td>
<td>N+254'</td>
<td>N+126'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sets of four long compartments are treated as containing 160-bit positions as shown.

Equivalent values of the characters in compartments (A)+2': onwards are formed in this matrix according to the following table:

<table>
<thead>
<tr>
<th>Value of control and basic quantity of nth character</th>
<th>Value 1 placed in mth bit position of:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0/0</td>
<td>No entry made</td>
</tr>
<tr>
<td>0/1, 0/2</td>
<td>N+366'/N+258'/N+130'/N+2'</td>
</tr>
<tr>
<td></td>
<td>N+386'/N+260'/N+132'/N+4'</td>
</tr>
<tr>
<td>0/15, 1/0</td>
<td>N+414'/N+286'/N+158'/N+50'</td>
</tr>
<tr>
<td></td>
<td>N+416'/N+288'/N+160'/N+52'</td>
</tr>
<tr>
<td>1/1</td>
<td>N+446'/N+350'/N+190'/N+62'</td>
</tr>
<tr>
<td></td>
<td>N+448'/N+352'/N+192'/N+64'</td>
</tr>
<tr>
<td>2/0</td>
<td>N+476'/N+350'/N+222'/N+94'</td>
</tr>
<tr>
<td></td>
<td>N+480'/N+352'/N+224'/N+96'</td>
</tr>
<tr>
<td>3/0</td>
<td>N+510'/N+382'/N+254'/N+126'</td>
</tr>
</tbody>
</table>
Section 16.4.2 (Cont'd)

Where \( m = n \) if none of bits 37 to 40 of the function word are set. Otherwise:

If \( n \) is in range 1-40

- \( m = n \) if bit 37 = 0
- \( m = n + 40 \) if bit 37 = 1 and bit 38 = 0
- \( m = n + 80 \) if bits 37, 38 = 1 and bit 39 = 0
- \( m = n + 120 \) if bits 37, 38, 39 = 1 and bit 40 = 0

If \( n \) is in range 41-80

- \( m = n \) if bits 37, 38 = 0
- \( m = n + 40 \) if one of bits 37, 38 = 1 and bit 39 = 0
- \( m = n + 80 \) if two of bits 37, 38, 39 = 1 and bit 40 = 0

If \( n \) is in range 81-120

- \( m = n \) if bits 37, 38, 39 = 0
- \( m = n + 40 \) if one of bits 37, 38, 39 = 1 and bit 40 = 0

If \( n \) is in range 121-160

- \( m = n \) if bits 37, 38, 39, 40 are all 0

In all other cases no bits are inserted in the matrix as the action will have ended before the nth character is reached.

**Notes:**

(i) The sign bits in the source words are ignored.

(ii) No entry is made in the matrix for space characters.

(iii) On completion of the action, B is clear and A contains the location of the last word edited.
16.5 Data Held in Integral Word Form

16.5.1 'Integral word' is the name given to a way of holding data in which each item occupies an integral number of 5-character words. Data on magnetic tape is sometimes held in this form.

To avoid the necessity to unpack this type of data an action is provided to copy data in bulk from one part of the store to another, with conversion between alpha and numeric forms if desired.

Associated with this action is one which clears specified areas of the store.

16.5.2 Bulk Copy and Clear Actions

The various versions of action 28 with modifier 0 or 1 provide these facilities and are defined below.

**Action 28.0/1 0/1 : BULK COPY**

(Bit 38 of A = 0)

**Definition:** Bulk copy to compartments starting at \( N \) the items specified by the table entry \( \{A\} \).

**Action 28.0.0**

COPY short numeric words without change of form.

**Action 28.0.1**

COPY short numeric words to alpha form. \( N \) must be \( D+1 \) where \( D \) is the first (long) destination compartment.

**Action 28.1.0**

COPY alpha words to short numeric form. See note for \( B! \) to 15 of \( \{A\} \) below.

**Action 28.1.1**

COPY alpha or long numeric words without change of form.

**Table:**

Register \( A \) must contain the following 'table entry':

\( B! \) First source location (in binary) plus one

\( B!5 \) for action 28.1.0.
Section 16.5 (Cont'd)

E21  (in binary) Actions 26.0.0 and 26.0.1;

to number of short source words to be copied with

E33 a maximum of 4095.

Actions 26.1.0 and 26.1.1; twice the number of
long source words to be copied with a maximum
of 8190 (i.e. 4095 long words).

E38 Zero (1 = bulk clear, see below).

The values of the other bits of (A) do not affect
this action.

Notes:

(i) Alpha-numeric transfers obey the same rules as
described for 'long discriminant and odd address'
in section 5.4.2. In particular, when converting
from numeric to alpha, all zeros are treated as
space characters.

(ii) When copying using Action 26.1.1 the sign bit
of the less significant half of each word is set
at zero in the destination.

(iii) A and B are left clear.

(iv) If the contents of A are negative, then more or
less locations than anticipated will be copied,
or the action will enter an endless loop (INO
III only).

Procedure:

See Bulk Clear.

Action 26.0/1, 0/1: BULK CLEAR
(Bit 36 of A = 1)

Definition:

Bulk clear compartments starting at N as specified
by the table entry (A).

Action 26.0.0

CLEAR short compartments starting at N.

Action 26.1.1

CLEAR long compartments starting at N.

Table:

Register A must contain the following 'table entry';

B21  (in binary) Action 26.0.0: number of short

to compartments to be cleared with a maximum of

B33  4095.

Action 26.1.1: twice the number of long
compartments to be cleared with a maximum of
8190 (i.e. 4095 long words).

E38 1 (0 = bulk copy, see above).

Amendment No. 8
December 1964
Section 16.5 (Cont'd)

The values of the other bits of \( (A) \) do not affect this action.

**Notes:**

(i) When clearing long compartments (action 26.1.1) both sign bits are cleared.

(ii) \( A \) and \( B \) are left clear.

**Procedure for Bulk Copy and Bulk Clear:**

The actions are carried out as follows:

(i) \( (SC) \) are copied to compartment 1.

(ii) \( (\text{Bits 1 to 15 of } A) \) are copied to \( SC \).

(iii) **Bulk Copy only:** A word is selected from the compartment specified by \( (SC) \), long or short as specified by the discriminant. If \( (SC) \) are odd and discriminant = 1 alpha-numeric conversion will take place.

\( (SC) \) are increased by 1 or 2 according to whether the discriminant = 0 or 1.

(iv) The word selected in step (iii) (Bulk Copy) or zero (Bulk Clear) is copied to the compartment specified by the modifier. If \( (OA) \) are odd and modifier = 1 numeric + alpha conversion will take place.

(v) \( (OA) \) are increased by 1 or 2 according to whether the modifier = 0 or 1.

(vi) Steps (iii), (iv) and (v) are repeated until the required number of compartments have been copied or cleared.

(vii) The original contents of \( SC \) are replaced and \( A \) and \( B \) are cleared.
16.6 Sorting

16.6.1 The Sorting Process

Data in random order may have to be sorted into some specified order. To enable this to be done each item must be given a key number such that by sorting items, into ascending order of key number, they are sorted into the required order. The key must be the most significant part of each item.

The sorting process is done in the following stages:

(i) 'Strings' of data (i.e. sequences of sorted data) are formed in the store.

(ii) Pairs of strings are 'merged' into larger strings - reducing the total number of strings to half.

(iii) Stage (ii) is repeated until one long string is formed containing all the data.

The process is described in detail in Volume V (Part 2) of this manual. This section describes the computer actions provided to aid sorting.

16.6.2 Initial Formation of Strings

This may be carried out in several ways. One method is to make use of the 'table look up' action which may be used to search for the position of a new item in a sequential list of items already sorted.

Action 1.d.0 : TABLE LOOK-UP

Definition: Search successive compartments N onwards until a compartment L is found such that (L) > (A), and then place L in B. (Comparison between compartments and (A) is in binary).

Notes:

(i) Overflow occurs if the contents of A plus the reflection of the last word in the search exceed \(2^{40} - 1\).

(ii) If (A) or any word in the table is negative, overflow may occur. This does not apply to LEO 326.

16.6.3 Merging of Strings

When it is required to merge two strings of data into one, Action 8 is used. There are two versions - one for fixed length items whose length is specified by the programme and one for variable length items, in which case the length of each must be stated in the data.
'Keys' must be of constant length and must contain fixed field information in each case. They must be the most significant part of each item.

**Action 8.0.0/8.1.0: MERGE**

**Definition:**
Merge strings of items whose locations are specified by modification registers 1 and 2 to form a string whose location is specified by modification register 3 of the current modification group.

Compare items by binary subtraction from the two 'input' strings, word by word. As soon as a difference occurs copy the item with the smaller word to form the next item of the 'output' string. Stop merging when either of the two input strings becomes exhausted or when the output string is full.

**Action 8.0.0**

MERGE CONSTANT LENGTH ITEMS as above. Item length in short words is specified in binary by (N).

**Action 8.1.0**

MERGE VARIABLE LENGTH ITEMS as above. Item length in short words is specified in binary in alpha form in the word preceding each item.

**Notes:**
(i) The modification registers must be set with the first (long) location of the string in the less significant half and the last (long) location +2 in the more significant half.

(ii) If two items are identical as far as the word of the shorter one, that in the string indicated by modification register 1 is stacked first into the destination string.

(iii) The 'item length' associated with variable length items specifies the number of compartments occupied by the item plus the item length word.

(iv) In action 8.1.0, the contents of the compartment specified by the address are read from store but are not used.

(v) Both merge actions read (but do not use for comparison) one extra word from each 'input' area at the end of the action.

(vi) At the end of the action A contains (modifier-end value) of modification register 3 and B is clear.

Amendment no. 6

December 1964
Section 16.6 (Cont’d)

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

(viii) (c) are unchanged at the end of the action.

(ix) If a word in either string is negative, overflow may occur (LEO III only).

(x) If the item length is specified as one or zero, overflow will occur, and in LEO III the action may enter an endless loop.

Action 16.1.C : COMPARE
(available on LEO 326 only)

Definition: Compare strings S1 and S2 of alpha data, where S1 starts at N and S2 at (A). Successive pairs of words are compared until a difference is found or the end of one of the strings is reached.

If S2 > S1, the next instruction is obeyed.
If S2 = S1, one instruction is skipped.
If S2 < S1, two instructions are skipped.

Amendment No. 3
December 1964
17. FACILITIES FOR CONCURRENT RUNNING OF PROGRAMMES

This section describes facilities which aid concurrent running of programmes on LEO III.

They are:

(i) The interrupt facility
(ii) The store reservations system
(iii) Actions used for communication with the operators.

The use made of these facilities is described in Volume IV, Part 1 of this manual (Master Programme Description).

17.1 Interruption

The purpose of the interruption facility is to allow the Master Programme to transfer control from one programme to another when a programme which has been held up becomes able to proceed ('external interruption') and also to allow the Master Programme to assume control when 'overflow' or 'lockout' occurs ('internal interruption').

The effect of interruption is to cause change of sequence to a fixed compartment in the store. This compartment contains a sequence change to an appropriate part of the Master Programme.

17.1.1 Conditions Causing Internal Interruption

Internal interruption will take place at the end of any action, if at that time all the following conditions are satisfied:

(i) 'overflow' or 'lockout' conditions set.
(ii) 'halt' condition not set (see section 5.6.1).
(iii) Instruction just executed not 24/0/0, 24/0/1, 24/0/3 or 24/1/2, ('modify next instruction address').

17.1.2 Occurrence of Conditions for Internal Interruption

The overflow condition, which is automatically set whenever the addition of two numbers of the same sign results in a sum of the opposite sign, is represented by a flip-flop in the coordinator. It can be set, reset, and tested by micro-programme and is left set at the ends of those actions which did not have the overflow corrected.

The lockout condition is described in section 17.2.2.
Section 17.1.3

17.1.3 Conditions Causing External Interruption

External interruption will take place at the end of an action if at that time the following conditions are satisfied simultaneously:

(i) 'external interruption permitted' condition set.
(ii) 'external interruption requested' condition set.
(iii) 'overflow' and 'lockout' conditions not set.
(iv) 'halt' condition not set (see section 5.8.1)
(v) Instruction just executed is not 24/0/0, 24/0/1, 24/0/3 or 24/1/2,
    ('modify next instruction address').

17.1.4 Occurrence of Conditions for External Interruption

The 'external interruption permitted' condition is represented by
bit 13 of the indicator register. It is automatically set by the leave
Master Programme action 26/0/3 (see section 16.1.7) and is automatically
reset when interruption occurs. The condition can also be set, reset,
and tested by the indicator control actions described in section 16.3.

The 'external interruption requested' condition is represented by
a flip-flop in the coordinator and is generated whenever the 'interrupt'
condition of a route is set and that route is available. It is also
generated when the 'manual interrupt' condition is set, or when the
'millisecond timer interrupt' condition is set. These conditions occur as follows:

(i) Corresponding to each route there is a condition which is referred
to as the 'interrupt' condition of that route. The interrupt
conditions of all the routes of a given channel are held in the
assembler for the channel. The interrupt condition of a route
becomes set if the route is tested by the 36/1/0 action at a time
when the route is engaged.

(ii) The 'manual interrupt' condition is part of the operating control
logic of the coordinator. It can be set by pressing the 'stack
indicators' button on the operators control desk (see section 17.3)
and so provides a means by which the operator can request
interruption.
The `millisecond timer interrupt' condition is set whenever the most significant digit of the timer (the 'seconds' digit - see section 10.4) has one of the values, 1, 3, 5, 7 or 9.

### 17.1.5 The Interruption Process

Interruption causes the following to take place:

(i) The action 26/0/2 (see section 17.1.6) is inserted in the instruction register instead of the next programme instruction. The address of the action is dependent on the type of interrumpion as follows:

(a) Lockout condition set - address is 16

(b) Overflow condition set and lockout condition not set - address is 24

(c) External interruption (lockout and overflow conditions not set) - address is 8.

(ii) The 'external interruption permitted' condition is reset.

(iii) All the 'route interrupt' conditions, the 'manual interrupt' condition, and the 'millisecond timer interrupt' condition are reset.

(iv) In the case of internal interruption, the overflow and lockout conditions are reset.

### 17.1.6 Interruption Action

On interruption the following action (26/0/2) is automatically inserted into the instruction register. It will cause a sequence change into the Master Programme which will deal with the cause of interrumpion.

The action can also be used in a programme to enter the Master Programme when it is not desired to return to the next instruction (see note (ii) to the action definition). In the following definitions \( T \) represents the arithmetic unit key number register - see section 17.2.2.

**Action 26.0.2: ENTER PRIORITY CONTROL**

**Definition:** Place \( (S8) \) in bits 1 to 15 of \( N \) and \( (T) \) in bits 17 to 20 of \( N \) and change sequence to No.1. Set \( (T) = 14 \) before selecting next instruction.

**Notes:**

(i) Tag 14 is set as the Master Programme has this tag.
(ii) SC is placed in B at the beginning of the instruction and is not stepped on by I as it normally would be. Thus the address stored in N is dependent on the way the action occurred.

(a) If the action was inserted on interruption:
SC will have been set up ready to select the instruction immediately before which interruption occurred. This selection was suppressed by the interruption and the 26/0/2 instruction obeyed instead. The address stored is therefore that of the next instruction to be obeyed when a return is made after the interruption has been dealt with, and may be used as a subroutine link.

(b) If the action was reached as part of a programme in the normal way: In this case SC will have been set up to select this instruction and will not have been stepped on; hence the address stored is that of this instruction. If a return to this part of the programme is desired the stored address must be stepped on by I before it may be used as a subroutine link.

17.1.7 Access to Master Programme

If it is desired to enter the Master Programme, and later return to the next instruction, action 23/0/2 is used. Action 26/0/3 is provided to return control to a programme previously left for entry to the Master Programme.

The actions are used in a similar manner to the subroutine actions described in section 8.2.

Action 23.0.2: ENTER MASTER PROGRAMME

Definition: Place (location of this instruction +1) in bits 1 to 15 of N. Set (T) = 14 and place the previous contents of T in bits 17 to 20 of N. Change sequence to N+1.
Section 17.1.6

Action 26.0.7 : LEAVE MASTER PROGRAMME

Definition: Replace (M) by bits 17 to 20 of (N), set bit 13 of I equal to 1 (external interruption permitted) and change sequence to the location specified by bits 1 to 15 of (N).

17.1.6 It is necessary when interruption occurs during the operation of a programme for the Master Programme to store the contents of registers, and to replace them when the programme is re-entered, so that no information is lost. This is done using the special actions 0.1.0 and 0.1.1 to save time and space.

Action 0.1.0 : STORE (A), (B) and (C)

Definition: Copy (A), (B), (C) to N', N+2', N+4 respectively.

Notes:
(i) No change is made between sign and modulus and sign and complement forms.
(ii) If N is odd, C is copied to N+4; the sign bit and least significant five quartets of registers A and B respectively are copied (from numeric to alpha forms) into words (N-1)', and (N+1)', respectively.

Action 0.1.1 : REPLACE (A), (B) and (C)

Definition: Replace (A), (B), (C) by (N'), (N+2'), (N+4) respectively.

Notes:
(i) No change is made between sign and modulus and sign and complement forms.
(ii) If N is odd, the contents of (N+1)', are copied to B. The contents of (N-1)', and (N+3)', are copied (from alpha to numeric form) into registers A and C respectively.

In LBO 326, the contents of (N-1)', (N+1)', and (N+3)', are copied (from alpha to numeric form) into A, B, and C respectively.
17.2 Store Reservations and Lockouts (optional facility).

The system provides the following facilities:

(a) Any attempt by one programme to corrupt information in areas of the store assigned to other programmes is detected and may be prevented if required. This is done by a system of interlocks which are set up by the Master Programme. The Master Programme can itself over-ride the interlocks, e.g. to clear the store when unloading a programme.

(b) Any attempt by an input assembler to over-fill its assigned annexe area is detected and may be prevented if required. This is done by placing special 'guard words' at the end of annexe areas (this is carried out by the Master Programme).

There are two modes of operation of the system, 'error detection' and 'error prevention'. The difference between these is described in section 17.2.2.

17.2.1 Tags and Key Numbers

Each long compartment in the store has 4 'tag bits' associated with it. When any instruction obtains store access the tag bits of the compartment concerned are compared with the contents of a special key register in the arithmetic unit. If a difference is detected the lockout condition is set and interruption requested.

The contents of the arithmetic unit key register are set by the enter and leave Master Programme instructions (see sections 17.1.6 and 17.1.7).

A key number is allocated to a programme when it is loaded under Master Programme control, and is in the range 1 to 13. This number is placed (in binary) in the tag bits of all compartments to be used by that programme. No two programmes in the store at one time are given the same key number. The Master Programme then arranges that whenever a programme is entered, the key number associated with that programme is placed in the arithmetic unit key register.

It is arranged that the tag of a compartment can be set up only by a programme which has key number 14; consequently the Master Programme is given this key number (with the exception mentioned below).
Each input or magnetic tape assembler also has a 4-bit key register. This key is set up as follows:

(i) Input assemblers and magnetic tape assemblers: Compartment 64+R for a route R (which contains the 'annexe start location' in bits 1 to 15) holds in bits 17 to 20 the key which is to be set up in the assembler key register. Both of these are placed in the assembler when input is initiated by action 19/O/1 (see section 16.7.3). The key will normally be the same as that of the programme using the assembler.

(ii) Output assemblers: These assemblers always have key 14 irrespective of the contents of bits 17 to 20 of (64+R). In order to prevent an assembler from over-filling an input annexe area, the compartment following the annexe area is given tag 15. Such compartments are said to contain a 'guard word'.

To allow guard words to be cleared when a programme is unloaded, the Unloader Routine of the Master Programme is given key number 15.

Any programme or assembler can use only those areas of the store which have the same tag as that set up in the arithmetic unit or assembler key register, with the following exceptions:

(i) A compartment with tag 0 can be used by any programme or assembler.

(ii) A programme, part of a programme or assembler with key 14 can use any compartment of the store except those with tag 15.

(iii) A programme or part of a programme with key 15 can use any compartment of the store.

17.2.2 Lockout

'Lockout' occurs whenever a request for store access by the main frame or an assembler fails to satisfy any of the following conditions:

(i) Contents of key register in arithmetic unit or assembler = tag of store compartment, or
(ii) Tag of store compartment = 0, or
(iii) Contents of key register in arithmetic unit or assembler = 14 and tag of store compartment ≠ 15, or
(iv) Contents of key register in arithmetic unit or assembler = 15.

If lockout occurs on transfer of information from store, the transfer is allowed to take place since this will not affect the contents of the compartment. If lockout occurs on transfer to store, the transfer is allowed to take place if the system is in the 'error detect' mode, but if it is in the 'error prevent' mode the transfer does not take place and the store compartment is not corrupted.

17.2.3 Lockout on Arithmetic Unit Access

The computer stops when the transfer of information is completed (if it is permitted) and the 'lockout' condition is set. The assemblers are not affected.

An engineer can then investigate the lockout. If he finds it was not due to a computer fault (i.e., it was due to a programme error), he can press the lockout restart key. This causes an attempt to be made to select the next instruction, but since the lockout condition is set interruption will occur (see section 17.1.1).

17.2.4 Lockout on Assembler Access

The channel is closed. The lockout condition is not set, and the computer is not stopped. Interruption will thus not occur.

17.2.5 Lockout Actions

Lockout is a form of interruption and uses the 26/0/2 action described in section 17.1.6. It is not dependent on the 'external interruption permitted' condition.

The tag of a compartment may be set up or selected for inspection by using the actions 23/0/3 and 23/1/m. These actions are used only by the Master Programs and their attempted use by a programme with key other than 14 or 15 causes lockout.
The key of the arithmetic unit is set by the actions 23/0/2 and 26/0/3 described in sections 17.1.7 and used to enter or leave the Master Programme. It is also set at 14 by interruption action 26/0/2 described in section 17.1.6.

The key register in the arithmetic unit is referred to in the following definition as T.

**Action 23.0.3 : SELECT TAG OF N**

**Definition:** Select tag of N into the four least significant bits of A and clear the remainder of A.

**Note:** Lockout is caused if (T) is not equal to 14 or 15.

**Action 23.1.0 : SET TAG OF N**

**Definition:** Copy the four least significant bits of (A) into the tag of N.

(i) (A) is unchanged.

(ii) The remainder of (N) is unchanged.

(iii) Lockout is caused if (T) is not equal to 14 or 15.

(iv) If m = 0, access is made to compartment 0, 8, 16 or 24 according to the modification group setting. Lockout will occur if tags are incorrect.

17.3 Communication with the Operator

The Master Programme uses the typewriter (described in section 14) for communications to the operator. Communications from the operator to the Master Programme are made via bits 1 to 12 of the indicator register. (Bit 13 of this register specifies the 'external interruption permitted condition, bits 14 and 15 specify the current modification group).

The contents of bits 1 to 12 of this register may be set by the operator using the 12 indicator setting keys on the control desk (see Appendix C). The required pattern is set up on the keys and when the stock indicator key is depressed the pattern is transferred to the register and interruption is requested.

Actions are provided for the Master Programme to examine bits 1 to 13 of the indicator register, and also to set and reset any
combination of these bits.

**Action 24.1.1 : STORE INDICATORS**

- **Definition:** Copy (I) to \( \bar{N} \)
- **Note:** (I) is unchanged.

**Action 25.1.0 : SET INDICATOR**

- **Definition:** For bits of \( \bar{N} \) (literal) equal to 1 set the corresponding bits in I equal to 1.
- **Note:** For bits of \( \bar{N} \) equal to 0 the corresponding bits in I are unchanged.

**Action 25.1.1 : CLEAR INDICATORS**

- **Definition:** For bits of \( \bar{N} \) (literal) equal to 1 set the corresponding bits in I equal to 0.
- **Note:** For bits of \( \bar{N} \) equal to 0 the corresponding bits in I are unchanged.

**Action 25.1.2 : INTERROGATE INDICATORS**

- **Definition:** Collate (I) with \( \bar{N} \) (literal) and copy the result to A.
- **Note:** (I) is unchanged.

**Action 25.1.3 : CONDITIONAL HALT**

- **Definition:** Collate (I) with \( \bar{N} \) (literal) and halt the computer if the result is non-zero.
- **Note:** (I) is unchanged.
17.4 Halting the Computer

The computer may be halted by one of two actions - action 25/1/3 (see section 17.3) and action 0/0/0 (described below).

Action 0.0.0 : HALT

Definition: Set 'halt' condition (see section 5).

Note: The computer will halt before selecting the next instruction.