APPENDIX F : Layout of the Intercode and Computer Code Print-Out.

OVERLAY

H.1 Introduction

As explained in Part 1, section 10, if a programme is of sufficient length it may be divided into units called chapters. The programmer must submit a list of the first procedures in the chapters (i.e. a list of the chapter start points). Each working section is given a chapter number by the programmer, and such sections are placed by the Translator in the indicated chapters, after the procedures. (Here, as elsewhere, the word 'procedure' means anything with a three-digit reference, i.e. a procedure, table or constants section). In addition to the programmer's chapters, the Translator creates an extra chapter (see Part 2, section 3.4.15), which is never addressed by the programmer.

When an instruction is translated, Pass 2 determines whether the address refers to a different chapter, and if so, inserts a 24/1/2 instruction into the programme to modify the address by the address of the start point of the other chapter.

H.2 The Overlay Instruction (Action 152)

An overlay instruction indicates to the Translator that the two chapters whose first procedures are given in the instruction are equivalent, i.e. their start points have the same address in the translated programme. When Pass 2 translates an instruction addressing a chapter equivalent to that containing the current instruction, the 24/1/2 instruction normally inserted for an inter-chapter reference is suppressed.

Pass 1 analyses the overlay instructions in the programme to determine which chapters are equivalent. When an overlay instruction is found, the equivalent of each of the two chapters involved is set at the lower number of the two; if either chapter has already been found to be equivalent to another chapter, the equivalentsof all three are set at the lowest value. An index is set up, containing the true chapter numbers and the numbers of the chapters to which they are equivalent. For example, suppose that a given programme has five (programmer's) chapters, and contains three overlay instructions:

(i) overlay 3 by 5;

(ii) overlay 2 by 4;

(iii) overlay 2 by 5.

(i) causes the equivalents of 3 and 5 both to be set at 3, (ii) the equivalents of 2 and 4 at 2, and (iii) the equivalents of 2 and 5 at 2. Since 3 and 5 are equivalent, and 5 and 2 are equivalent, 3 is equivalent to 2. So the index set up is:

True chapter number	Equivalent
1	1
2	2
3	2
4	2
5	2

The end routine of Pass 1 now examines this index to determine how many of the chapters are to be placed in the store when the programme is loaded, assuming that the appropriate overlay instructions will be obeyed to read the others in when they are needed. It does this by selecting the highest equivalent number from the index; if there have not been any overlay instructions the highest equivalent number will be equal to the number of programmer's chapters, and all chapters will be initially loaded. In the example, the highest equivalent number is 2, and so the first two chapters of the programme will be placed in the store; the example illustrates how the Translator deals with a situation in which the programmer has presumably omitted an overlay instruction, for chapter 3 will never be read in.

H.3 Layout of Programme Tape

The Master Programme requires that the last of the chapters initially loaded be the Translator's extra chapter, in order that it may set up its working locations. For this reason the highest equivalent number from the index is increased by one to give the total number of chapters to be initially read in. If the programme does not use the overlaying facility the Translator places the extra chapter after the programmer's chapters on the programme tape; if the programme does use the overlaying facility, the extra chapter is inserted between the chapter whose true number is the highest equivalent number, and the following chapter. In this case the chapters are re-numbered on the tape: in the example in H.2, this would give:

Chapter:	1	2	extra chapter		3	4	5
Position:	1	2		3	4	5	6

The translation of an overlay instruction takes the re-numbering into account when indicating to the Master Programme which chapter to read in.

H.4 Effect on Chapter Lengths

If several chapters are equivalent, Pass 3 will arrange that the space allocated will be sufficient for the longest. In the example, if chapter 5 occupies 6000 locations and all the others 1000 locations, the chapter lengths allocated are 1000 for chapter 1 and 6000 for chapters 2,3,4 and 5.

H.5 The Dummy Overlay Instruction

An overlay instruction can be used to address an area of store previously occupied by a programme chapter as working sections. A procedure containing no instructions must be submitted to the Translator and the procedure number entered on the Section Description Sheet as a chapter start point; the working sections are specified (also on this sheet) as belonging to this chapter, which must not contain any procedures other than the empty one. The corresponding overlay instruction is written in the normal way (but so that it is never obeyed); since it does not actually cause any information to be read from the programme tape it is referred to as a dummy overlay instruction. The start address of the first working section is equal to (8 + address of the first instruction in the chapter which is dummy overlayed). Note that the parameters in the chapter are preserved, and that only those parts of chapters which will not be needed again (e.g. the preparatory routines) may be so overlayed.

H.6 Preservation of Working Locations

The programme tape contains only the non-zero part of the chapters, i.e. the procedures; the associated working locations are not held on the tape, and so when chapter X is read over chapter Y, the locations associated with Y are available for use in X (provided that X is not so long as to overwrite part of them with instructions). When working areas are used in this fashion, they should be associated with the longer chapter, to ensure that they are preserved. (This is the case when a 24/1/2 instruction will be suppressed).

The switch/item +/indirect modification register locations used in the chapter are placed at the end of the longest equivalent chapter so that they are preserved automatically when the chapter is overlayed.

THE SWITCH, ITEM + AND INDIRECT MODIFICATION REGISTER FACILITIES

I.1 Introduction

These facilities resemble each other in that the Translator generates a working location for each one in the object programme; that is to say, a working location is generated for each switch (i.e. the reference of an 85 action), each distinct programmer's section acted upon by either a 51 or 53 action, and each modification register in the range 4 - 20 used by the programmer.

These working locations are placed in the chapter in which the relevant facility is first used, and form the last of the Translator's four extra procedures (see section 3.4.13): i.e. procedure 1004 if this is chapter 1, 1008 if chapter 2, and so on. If the facility is used in another chapter an inter-chapter reference will be made, unless the two chapters are equivalent.

Pass 1 counts the facilities used and forms a list of each of the three types (see section 2.3.2.(e)); for each facility the list gives the number of the containing chapter, the identifier of the facility (e.g. the number of the modification register) and the position within the generated procedure. Pass 2 refers to these lists when access is required to a facility during translation, to note the location to be addressed. Pass 3 determines the position within the chapter of the generated procedure.

I.2 Switches

A switch consists of a working location containing a binary number in the range 1-99. This location is identified by the reference of the 85 action (Switch).

Each 85 action is translated into a 'leave sub-routine' action, modified (indirectly) by the setting of the switch; the return address for the nth continuation line of the switch is given by the least significant 15 bits of the nth location after the 'leave sub-routine' action. A negative switch setting will cause the 'indirect modify' instruction to 'negative search'; i.e. if the setting is n-, the contents of location n will be taken as the switch setting (unless this location carries tag 14, in which case the action will lock out).

I.3 Item + Counters

Each distinct programmer's section (or Section 0, i.e. absolute address) acted upon by either a 51 or 53 action is allocated a working location, which forms the item + counter for this section. The counter is used by the programmer when the data to be unpacked (condensed) contains (is to contain) line end characters; a line end character is a character which indicates the end of a record in the case where one block contains several records. Such a record starts at the most significant character of a word, and occupies an integral number of words (the line end may occur anywhere within the final word of the record).

As explained in Section 3.4.10, the unpack and condense actions leave the address of the next free location (within the transit area) in Register A. When Pass 2 translates a 51 or 53 action, it generates an instruction to copy the contents of A to the appropriate item + counter; when an action using the + facility is translated, Pass 2 generates a sequence of instructions which obtain the location on which the action operates from the relevant counter.

I.4 Indirect Modification Registers

Pass 1 counts the 59, 90, 91, 92 and 93 actions acting on distinct modification registers in the range 4 - 20; Pass 2 then arranges that the register is addressed as a normal store location when it is to be set up or used.

SEQUENCE CHANGES

J.1 Introduction

The destination of a sequence change is a line serial in the Intercode print-out, and thus any instruction may be a destination. The exceptions are the directive lines PROCR and NOTES, the C-lines of instructions, together with tables, constants, PM points and trial data. The source of a sequence change is the line serial of the following instructions:

(a) Actions 70-79 (conditional tests on A,B, A* and unconditional sequence change).

(b) Actions 94, 97 (conditional tests on modification registers).

(c) C-lines of Action 150 (option choices).

(d) C-lines of actions 85, 87 (switch settings).

(e) Machine code instructions 24/1/0 and 27/d/m.

The sub-routine Actions 80 (enter) and 82 (leave) cause changes of sequence, but are processed separately.

A sequence change forward has a destination greater than the source, and a sequence change back has a destination less than the source.

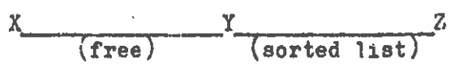
Sequence change destinations are indicated on the Pass 2 print-out of the programme by the source serial followed by an arrow, to the left of the line serial of the destination. Where there are several sources these are listed vertically and the final one is printed opposite the destination.

Lists of sequence change sources and destinations are kept for two reasons, firstly so as to print the source with the destination, secondly so that the address of the destination may be noted, to be placed as the address of the source when it is translated.

J.2 Work done in Pass 1

As Pass 1 outputs each Intercode instruction, it is tested to be a sequence change; if so, it is tested to be a sequence change back. For this, Pass 1 must evaluate the address (destination) in the light of the current amendments, so that the list of adjust details/unique

references/old-new procedures is examined to determine what address (destination) Pass 2 will give this instruction. When the address is computed, it is compared with the current line serial. Sequence changes forward are ignored by Pass 1, and the sequence changes back are sorted into a list. The list contains one word per sequence change, the most significant half being the destination serial and the least significant half being the source serial, both in decimal, and sorted into ascending destination. For coding ease, the list has a fixed end and floating start, and appears as follows:



Location Z contains an infinity word as the list end point, and Y moves toward X as the list fills.

The Pass 1 end routine copies the list from one end of the section to the other, such that the list begins at X and terminates at Y'. The space from Y' to Z will be used during Pass 2.

J.3 Work done in Pass 2

This is done in two stages, and each stage is sub-divided into back and forward sequence changes. In the course of translation, there are two areas of store holding sequence changes. One of these areas is the X-Z region used in Pass 1, which becomes sub-divided into 3 lists.

J.3.1 Stage 1 : Translation of a sequence change destination

When the next instruction is input from Pass 1, it is tested to be the destination of a sequence change. Firstly, the line serial of the instruction is compared with the smallest destination in the sequence change back list set up during Pass 1; if there is a match, the source is printed and the next destination examined for further entries to the same serial. A report is made if the smallest destination is less than the current serial, and the test is repeated until a destination is found, the word containing line destination + line source is replaced by a word containing line source + location of destination in current chapter, i.e. the current value of the space counter. The position of the smallest unmatched destination is noted for the tests on subsequent instructions.

The second part of stage 1 determines if the current instruction is the destination of a sequence change forward; stage 2 has made a list of sequence changes forward whose source has been translated but whose destination has not, and this list is now searched to determine if the current instruction is a destination. The list is in ascending destination, and the smallest destination is compared with the current serial to determine a match; if it is earlier, a report is made. All matches are printed as entry points, and the destination address is stored in the B-Z region of the list such that the nth location from Z contains the destination of the nth sequence change forward in the programme. The sequence change forward list contains the counter n with each destination/source, with n = 0 for all sequence changes forward to the start of a procedure.

The X-Z region now contains 4 parts, as follows:

X_____X_____B_____C_____Z

X-A Sequence changes back whose source has not been translated but whose destination has; the destinations are held with each source in random source order.

A-B Sequence changes back whose destination has not yet been translated.

B-C Free

C-Z Addresses of sequence changes forward, other than to the start of a procedure, whose destination has been translated. Not all of the locations will be occupied as they are in source order and consequently random destination order, until the end of Pass 2.

The points A, B, C will move as Pass 2 progresses, such that B approaches A and C approaches B. The distance X-A will increase/decrease continually.

J.3.2 Stage 2 : Translation of a sequence change source

When a sequence change instruction is translated, its address (destination) is recovered from the sequence change back list, built up in stage 1, or it is placed in a list of sequence changes forward whose source has been translated but whose destination has not.

(i) The X-A region is examined to see if any of the stored source addresses match with the source of the current sequence change. This list is in random source order, and must be completely scanned. If no match is found, the sequence change must be forward. If a match is found, the destination address is copied as the address of the current instruction, and if the source were found at point A', the list A' - B is moved one word up to delete the completely processed sequence change back.

(ii) If no match with the source was found in the X-A region, the sequence change is forward and is inserted into a separate list. A counter is kept of sequence changes forward, and the list contains:

Line destination + line source + counter n sorted into order of ascending destination. The list has a fixed start and floating end. When stage 1 finds a match in this list, the list is moved up to overwrite the completely processed sequence change forward. The sequence change instruction is specially marked so that Pass 3 may extract the appropriate address from the C-Z region when it processes the instruction. A sequence change to a procedure start has n = 0 as the address can be deduced immediately from the procedure start index.

At the end of Pass 2, the points X, A, B will coincide and the region C-Z will be completely filled. In the course of Pass 2, the point C may move to overwrite the initial position of point B.

J.4 Work done in Pass 3

All sequence changes forward not to a procedure start bear a special indicator. Pass 3 keeps a counter to select the nth location from the C-Z region when processing the nth instruction with the special indicator, which is then placed in the instruction address.