6. THE ROUTINE STATEMENT

6.1. Introduction

So far the implementation has been primarily concerned with the syntactic analysis of statements. The ROUTINE statement provides the means for defining the semantics associated with a particular statement. Once this has been incorporated together with some Basic Statements, the compiler will be in its minimum workable form. Once this point has been reached, extension of the Compiler can be achieved by using its own definitional facilities.

A CC ROUTINE can be divided into two parts: the heading followed by the body. The main purpose of the heading is to define to which Index position this routine is to be attached. This can be done by either specifying the Index position or providing a heading which can be matched against a FORMAT already defined. In this second method parameters can be specified as input parameters to the ROUTINE. In the FORMAT statement for a particular instruction will be Phrase Identifiers which define the syntactic phrases expected at certain points in the FORMAT. In the ROUTINE heading, Phrase Variables will appear at positions corresponding to the Phrase Identifiers in the FORMAT statement. When a particular statement of this type is recognized, then the Phrase Variables in the Routine heading will be given as values the analysis trees produced during the recognition of the associated Phrase Identifiers in the FORMAT.

The FORMAT statement, therefore, defines the types of phrases expected while the ROUTINE heading states which Phrase Variables are assigned the analysis tree values when a particular statement is recognized.

The ROUTINE body consists of statements in any of the three Format classes: - BS, AS and SS. To initialize the bootstrapping process some basic statements must, of course, be provided external to the general system. These are the BASIC STATEMENTS to be defined as ITEM routines which will be described later.

6.2. Identifier Conversion

Like the PHRASE and FORMAT statements, the whole of the ROUTINE is input initially, and the Phrase Variables are converted using the Identifier Conversion Routine 230. In the case of the ROUTINE statement Phrase Variables rather than Identifiers appear. These may have labels and indices attached, and the output from the routine is more complex than the Index Position produced in the case of the PHRASE and FORMAT statements. This alternative path is marked by B78 being set to zero on entry. The additional part of the Identifier Conversion Routine is given in flow diagram (iii).

The output for each Phrase Variable is now as follows:-

1. FIRST WORD. 'A' field set to Index Position of Phrase Identifier. If the Phrase Variable is the * type, then the Index Position is that of the Identifier with the * deleted.

2. SECOND WORD. 'A' field set to position of Phrase Variable in the L.S.E. A double entry list is kept of the Phrase Variables as they appear. Each new one is allocated the next position in the L.S.E. and entered in the list. (See Trees and Routines.)

   'Z' field set to numeric value of index

   'S' field set to 0, 1, 2 depending on whether index is [N], [A] or [B] type.

For example, the Phrase Variable [ZY*/7(A5)] where

a = Index Position [ZY*]
b = Index Position [ZY]
i = Position of [ZY*/7] in L.S.E.

would produce as output:-

1. A field set to b
2. A field set to i
   Z field set to 5
   S field set to 1

A Phrase Variable without a label is assumed to have a label value of zero.

6.3. Routine Heading

The Routine Heading is recognized by the Analysis Routine where the built-in syntax can be described as follows:-

```
PHRASE [ROUTINE HEADING] = SMALL R [N], R [N], [COMPILE] [PI] [EQV] [RESOLVED-P]
PHRASE [COMPILE]         = (COMPILER), NIL
PHRASE [RESOLVED-P]      = [SET P'] [ANY PI] [RESET P]
```

The Index Positions associated with these are as follows:-

254	ROUTINE HEADING
249	COMPILE
181	PI
185	RESOLVED - P
152	ANY PI
244	SET P'
250	RESET P

The first two alternatives are straightforward. The SMALL Routine is defined as one which only needs the simple form of entry mechanism and cannot have statements in its body which cannot be compiled. It is entered using the simplified non-stacking entry via the DOWN routine.

The [COMPILE] option, if present, states that this version of the routine is designed to produce compiled code for the statement rather than interpret the Analysis Record. More will be said of COMPILE routines later.

The phrase [PI] is built-in and will recognize any Phrase Identifier. Its action is given in the flow diagram. Note that it modifies the syntax for [GENERATED-P] and also [RESOLVED-P]. The latter modification changes [ANY PI] to the particular Phrase Identifier recognized by [PI]. This ensures that the analysis depends on the initial [PI]; that is the remainder of the heading is analyzed with respect to this Format Class.

Once the ROUTINE heading has been recognized a check is made that a match at the top level between the Phrase Identifiers appearing in the Format and the Phrase Variables appearing in the ROUTINE heading. For example, it woudl be illegal to have:-

FORMAT [AS] = GET [ABN]
ROUTINE [AS] ≡ GET [AB/1]

The Phrase Variable must be of type [ABN]. This check is easily made as the top level should consist of P words. In the case of a COMPILE routine we must allocate an Index position for the routine and enter it in the Double Entry List 256 containing Routine Positions and the corresponding positions of the COMPILE versions.

6.4. ROUTINE Body

The compilation of the ROUTINE body is mainly done by the Routine 253 (Compile Body of Routine).

The compilation of Label declarations and uses is achieved by using two lists. Each time a Label is declared, its numerical value is inserted into the B52 double entry list, together with its address in the ROUTINE body. Each time a Label is used, the Label's numeric value is inserted in the address part of the instruction and the address of the instruction is added to the B53 nest.

Once Routine 253 has compiled the body of the routine, the ROUTINE routine inserts into the first two words of the routine:-

1. Length of L.S.E.
2. Length of routine up to Label Directory.

The Label Directory is added at the end of the routine. It consists of a direct look-up table containing the addresses of the labels relative to the start of the routine. The length of the table being the maximum numeric value of the labels. The directory is made by scanning the B52 double entry list.

6.5. Compile Body of Routine 253

Unless a Compile version of the routine associated with a statement exists, a statement in a routine body is interpreted. The 'compilation' of the statement then consists of storing a call of the TRANSPLANT routine followed by the Analysis Record of the statement.

The first action of the routine is to analyze the input to test for the end of the ROUTINE body, the presence of a label or a statement.

If the end has been reached, the address fields of instructions referring to labels must be set to their correct value. The label uses are withdrawn from the B53 nest and looked up in the B52 double entry list to find the correct address. If the label does not appear in the list, then it is undefined and a diagnostic is output.

If a label is declared, its address is added to the B52 double entry list together with the label number.

If neither of the above are present then the next statement must be analyzed, and this can belong to one of the three Format Classes, [BS], [AS] or [SS] and analysis is tried against these in that order. If a [BS] statement has the same FORMAT as one of the statements in the other two classes these can be recognized by adding an '*' before the statement.

The built-in syntax can be described as follows:-

```
PHRASE [LABEL OR END] = [SEP*?] [EOS], [SEP*?] [LABL]
PHRASE [LABL] = [N]), NIL
PHRASE [SEP] = [COMMA], [EOL] built in and Contract Record
PHRASE [STATEMENT] = [BS], [*?] [AS], [*?] [SS]
PHRASE [*?] = *, NIL
```

The Index positions are as follows:-

144	LABEL OR END
146	LABL
148	SEP*?
147	STATEMENT
182	*?

Once a [STATEMENT] has been recognized it may be possible to compile the statement if a Compile version of the Routine associated with the statement exists. The Compile routine is called as long as no parameters exist in the statement. This is done by checking the Analysis Record for P' words. P' words will appear whenever a Phrase Variable is recognized in a [RESOLVED-P]. This can be seen in the flow diagram (ii) for the Analysis Routine 215 (lower left). If a Phrase Variable is recognized then the Analysis Record produced consists of the second word produced by the Identifier Conversion Routine (see 6.2) with an F field set to 2 (P word). If in [RESOLVED-P] then it is a parameter P' in which case the J4 bit is set. This is achieved by [SETP'] setting B44 to J4 + F3 and [RESET P] resetting B44 to F3.

If no P' words are present the COMPILE routine for the statement is entered if it exists and an attempt is made to compile the statement. If it is unable to compile the routine (normally due to the complexity of the statement) then it returns with B54 set to -1. Note that this would be an error if the ROUTINE heading defined a SMALL routine.

If no COMPILE routine exists for the statement, then the 'interpretation' is achieved by calling:-

217 CONVERT ABSOLUTE &'s to RELATIVE &'s

This routine stores first the call of the TRANSPLANT routine followed by the Analysis Record for the statement. As the Analysis Record is added, the ampersands are changed so that all pointers are relative to the front of the routine.