4. THE PHRASE STATEMENT

4.1. Introduction

The next aim is to add the routines necessary for compiling the PHRASE statement. Basically all that is required is the PHRASE routine associated with the Master Phrase:-

PHRASE 218

This, however, requires quite a large number of subroutines which can be divided into basically two classes. The first set consists of general subroutines for dictionary manipulation and printing which will be equally useful later on. The second set consists of routines, dictionaries and syntax definitions closely associated with the PHRASE statement.

4.2. Dictionary Routines

The complete set of routines required are as follows:-

204	Add word to list
205	Add word to nest
206	Withdraw word from nest
207	Delete list
209	Add list to list
210	Copy linear list to chain
259	Add nil branch to dictionary
141	Look up or enter in a double entry dictionary
225	Merge new entry into dictionary
224	General dictionary routine

Pictorial descriptions of all these routines apart from the last two are included in the appendix.

Routine 225 can be used to add an entry to the front of a dictionary or alternatively at the end. Its flow diagram is given in the appendix.

The General Dictionary Routine 224 has three main purposes. To look-up, enter or delete entries from a dictionary. The flow diagram (i) gives the outline of the parts of the routine common to all three types of entry. Flow diagrams (ii) and (iii) give the parts which depend on the types of entries. The first part differentiates between the following 6 cases:-

A	The dictionary is empty (B63 = 4).
B	No match was found between a dictionary entry and the given string. However, there is a dictionary entry which forms the stem of the string to be added (B63 = 2).
C	No match, but there is a dictionary entry which has a stem equal to a stem of the string to be added (B63 = 3).
D	No match at all. Not even a common stem (B63 = 6).
E	A match exists between the dictionary empty and the string to be added (B63 = 0).
F	No match but string to be added is stem of existing entry in dictionary (B63 = 1).

The action required then differs depending on whether look-up (B63 = 1), entry (B63 = 2) or deletion (B63 = 3) is taking place.

4.3. Input of the Phrase Statement

The philosophy adopted in the Compiler Compiler, as far as interface with the subscan and initial input is concerned, is to take all the input required for a particular statement and then to process the statement independent of further access to the subscan level. This is obviously inefficient as far as storage use is concerned unless the amounts of storage required in this way are small and this need not be the case. In the case of the Phrase Statement this, however, is usually so and is probably the simplest and most efficient method of execution.

Two routines are required to read in the whole of the Phrase Statement.

1. READ NEXT SECTION 213
2. DELETE SUPERFLUOUS EOL'S 265

The first routine reads in statements until it reaches a line that can be recognized as the next Master Phrase by the Analysis Routine. All the input up to the next Master Phrase is read and built up into a list pointed at by B61. The next Master Phrase is pointed at by B79. Both strings are stored as conventional circular chains. The B61 chain has a character 128 added at the end.

The second routine removes all redundant end-of-line characters which may have been inserted during the process of building-up the body of the Phrase Statement.

Once these two routines have been accessed the Phrase Statement body in B61 is ready for analysis.

The only input to Routine 213 is the variable B79 pointing at the first line of the Phrase Statement with the characters P,H,R,A,S,E removed.

The Routine 265 enters with B69 pointing at the input to be checked. On exist B69 points to the reduced line.

4.4. Print Routines

The two routines required are:-

177	PRINT NEW FORMAT OR PHRASE
183	PRINT SYMBOL

The first routine will output the words from B68 until B69. Each word is output as a symbol if it has a value less than 128 else the number itself is output surrounded by square brackets. This is followed by the Phrase or Format Index Number which is passed as a parameter stored in B67.

The Routine 183 is a subroutine which prints the value of B69 as a symbol.

4.5. Phrase Routine

The first action of the Phrase Routine, after the complete statement has been input by Routine 213, is to call the Identifier Conversion Routine 230.

Routine 230 is also used by the ROUTINE routine for reducing the body of a routine and, for this purpose, it has to recognize and separate different phrase variables of the same type. In the case of the Phrase Routine entry (differentiated by B78 being non-zero) it has the task of replacing each Phrase Identifier enclosed in square brackets by its position in the index. Characters other than Phrase Identifiers are not changed.

At this point we see the first use of the Analysis Routine as an internal recognizer. This routine has been designed to analyze strings with syntactic structure. The generalized phrase variable falls into this category. As we have not yet implemented the PHRASE statement for defining syntactic structure the syntax has to be built in as ITEM'S. The phrase variable can then be checked by the analysis routine for legality. The syntax of the PHRASE variable is as follows:-

```
    PHRASE [PHRASE VAR] = [[][PHRASE ID] [PHRASE LABEL] [PHRASE INDEX] ]
    PHRASE [PHRASE LABEL] = / [N] , NIL
    PHRASE [PHRASE INDEX] =  (  [ABN]  ) , NIL
```

The [PHRASE ID] recognizer is a Built-In Phrase. This means that instead of a conventional dictionary form of entry used by the Analysis Routine to test for recognition, we have a subroutine which itself looks at the input stream and, if it recognizes anything, will produce an analysis tree. Built-In Phrases will, in general, be used when efficient recognition can be done by this non-standard method. The [PHRASE ID] routine has to recognize two basically different kinds of Phrase Identifier. The commonest will be an open square bracket followed by a set of characters terminating with a close square bracket. The second form is open square bracket, integer, close square bracket. This will be used if, for example, a dictionary or phrase has been put in as an ITEM and has no name entered in the CID dictionary. In this case the integer represents the index position.

If, for example, we have the Phrase Identifier [ABC] then the analysis tree produced will be & 3+F2 A B C with the ampersand pointing to C. The word containing an F2 flag bit is a Block Word (see Trees and Routines) which indicates that the next 3 characters should be treated as a string and not processed by routines which manipulate analysis records.

The second form of Phrase Identifier [175], for example, produces an analysis record & N where N = 175. The differentiation between the two forms is done by noting whether or not the Block Word is present.

The Index Positions of the above Phrases are:-

171	PHRASE	VAR
172	PHRASE	ID
174	PHRASE	LABEL
176	PHRASE	INDEX

Once the call on the Analysis Routine by Routine 230 has produced a recognition then there are two main paths. If the Phrase Identifier is a simple index position then all that is required is to add this value to the output stream and continue.

If we have a genuine Phrase Identifier, it must be looked up in the CID dictionary. If it is already there then its index position is obtained and proceed as above. If not there, then we must assign an index position to this identifier and add it to the CID dictionary. A record of where this Phrase Identifier has been put is printed, its index position cleared and then proceed as before.

The one complication to the straightforward process defined above is the introduction of * and ? endings to Phrase Identifiers which generate additional Phrase declarations. For example, [ABC *] appearing anywhere implies an additional Phrase declaration

PHRASE [ABC *] = [ABC] [ABC *], [ABC]

Similarly, [ABC ?] implies an additional Phrase declaration:-

PHRASE [ABC ?] = [ABC] , NIL

In the case of a new Phrase Identifier with either ending appearing it is therefore necessary to also insert an entry in CID and assign an index position for the Phrase Identifier without the ending.

Also, the above definitions must somehow be added to the Phrase Definitions presented to the PHRASE routine. This is done by building up a bit of additional Phrase Statements that are generated in this way and then processing these before the Phrase statement we were processing originally. Skeletons for the two types of Phrase definitions to be added are stored in ITEM 139. Processing is done by the call of the Auxiliary Phrase Routine 242.

Once the Identifier Conversion Routine 230 has reduced the Phrase Identifiers to their associated numeric values we can process the Phrase Statement body. Once again the Analysis Routine is used to produce the recognition.

The built in syntax is as follows:-

```
PHRASE [PHRASE BODY] = [CONTRACT RECORD] [ANY PHRASE ID] = [PHRASE TAIL]
PHRASE [CONTRACT RECORD] = (C R) , NIL
PHRASE [PHRASE TAIL] = [PHRASE ALTERNATIVE] [PHRASE TAIL] , [PHRASE
                                                     ALTERNATIVE] [128]
PHRASE [PHRASE ALTERNATIVE] = BUT NOT , [PHRASE ELEMENT] [,] , NIL,
                                                     [PHRASE ELEMENT]
```

PHRASE [ANY PHRASE ID] is built-in and just checks for a Phrase Identifier in the next position of the input. The Analysis Record produced being & N.

PHRASE [PHRASE ELEMENT] is built-in and checks for a sequence of Phrase Identifiers or basic characters terminating on [,] or [128]. The analysis record produced for a string of N such symbols is

$$N+F_2\ C_0\ C_1\ \ldots\ldots\ C_N$$

where the C_i are the characters checked.

The positions of these phrases is as follows:-

PHRASE BODY	151
CONTRACT RECORD	143
PHRASE TAIL	156
PHRASE ALTERNATIVE	157
ANY PHRASE ID	152
PHRASE ELEMENT	158

The remainder of the Phrase Routine consists of building up the Phrase Entry from the sub parts detected by the Analysis Routine using the routines for adding entries to dictionary. The BUT NOT entry has to be added before the first entry, otherwise the process is straightforward. The Contract Record Phrase will have the T_4 bit set in the M word. The K word contains the maximum number of phrase identifiers in any alternative.

4.6. [N], [A] and [B]

These 3 routines are required to recognize the basic variables and constants of the system. They are built-in for efficiency purposes. Although only [N] has been used so far in a positive manner the other two have been used to ensure that a Phrase Index does not appear. It is not essential to add [ABN], [AB], [A] and [B] at this stage, but the routines for [N], [A] and [B] are coded as a single routine with multiple entry points and will, therefore, all be treated now.

In the case of [N] recognition is successful if an integer with less than 9 digits appears. The analysis record consists of just the integer value. The anlaysis record for [B] is exactly the same. Once the B has been checked, the [N] routine is entered.

[A] is recognized in a similar way, although the analysis tree producedd is different. In this case we have to reset B81 to the maximum value of [A] encountered so far. The Analysis Record in this case consists of two words. The first contains $1+J_4+F_2$ and the second B80 + i for A_i.

B80 by this time will have the number of Phrase Identifiers appearing in a particular routine so that B80 + i will be the position of A_i in the LSE (List of Selected Expressions). On execution, the address of A_i is the value in the Analysis Record relative to B72.

4.7. Conclusion

Once the above has been written it will be possible to add Phrase Statements to the Compiler Compiler. Although none will be required at this point it is a convenient point to stop and test before proceeding. The actual definition of Phrases and Formats is not really necessary until after we have introduced the ROUTINE routine.