Internal Maintenance Specifications
CDC CYBER 70 Series COMPASS Version 3.0

6000 and CYBER 70/ Models 72-74
SCOPE 3.4 Operating System

7000 and CYBER 70/ Model 76
SCOPE 2.0 Operating System

© COPYRIGHT CONTROL DATA CORP. 1972

Contained herein are software products copyrighted by Control Data Corporation. A reproduction of the copyright notice must appear on all complete or partial copies.

M664

CDC CYBER 70 Series COMPASS Version 3.0
Internal Maintenance Specifications

Preface

This document describes the internal structure and operation of Version 3.0 of the COMPrehensive ASSembler program, COMPASS, for the Control Data 6000, 7000, and CYBER 70 Series computer systems. COMPASS is written in its own language and is an absolute central processor program whose execution is supervised by the SCOPE 3.4 operating system for the 6000 series and CYBER 70/ Models 72, 73, and 74, or the SCOPE 2.0 operating system for the 7000 series and CYBER 70/ Model 76. The assembler program is identical for both systems; installation parameters control conditional assembly of alternate code sequences where these are made necessary by incompatibilities between the host operating systems. COMPASS can also be used with most other 6000, 7000, and CYBER 70 series operating systems, such as 6000 MACE, KRONOS, and SCOPE 3.3, 7000 SCOPE 1.1 and 1.2, and special customer systems, but minor changes may be needed in some cases.

This internal maintenance specification is intended to be used in conjunction with an assembly listing of COMPASS. The reader is also assumed to be familiar with the relevant portions of the following publications:

60360900	CYBER 70/ COMPASS 3.0 Reference Manual.
60100000	6000 Series Computer Systems Reference Manual.
60258200	7600 Computer System Reference Manual.
60347000	CYBER 70/ Model 72 System Description, Volume 1.
60347200	CYBER 70/ Model 73 System Description, Volume 1.
60347400	CYBER 70/ Model 74 System Description, Volume 1.
60347300	CYBER 70/ Models 72, 73, 74 Instruction Descriptions, Vol. 2.
60347100	CYBER 70/ Extended Core Storage Reference Manual, Vol. 3.
60367200	CYBER 70/ Model 76 Computer System Reference Manual.
60307200	6000 Series SCOPE 3.4 Reference Manual.
60342600	7600 and CYBER 70/ Model 76 SCOPE 2 Reference Manual.
60342500	UPDATE Reference Manual.
60281700	MODIFY Reference Manual.
60344200	LOADER Reference Manual.
60281200	7600 SCOPE 1.1 Reference Manual.
60307300	Record Manager Reference Manual.

The UPDATE and MODIFY manuals contain descriptions of the program library and compile files read by COMPASS. The LOADER manual specifies formats of the binary output written by COMPASS. The SCOPE 1.1 manual includes descriptions of the macros and routines used by COMPASS when it is not using a Record Manager for assembly-time I/O.

Table of Contents

1.0 THE ASSEMBLY PROCESS

The COMPASS assembler assembles in a rather straightforward manner. This chapter describes the principle functions of the process in general terms; the details are examined in later sections.

1.1 Two-Pass Philosophy.

Two-pass assemblers all operate on essentially the same principle. The first pass reads the source file, and performs the following main functions:

- Examines each instruction to determine how much storage is required in the object code by the instruction.
- Defines symbols. When symbol definition operations are requested, such as by the appearance of a symbol in the location field of a machine instruction, or by its use in some pseudo-operation, the symbol is given a definition. The collection of these symbols and their values is known as the Symbol Table.
- Expands the various higher level operations, such as macros and duplications.
- Accumulates literals.

Thus, at the conclusion of the first pass of assembly, the following information is known:

1. The length of each block in the assembly.
2. The value, relative to some USE block, of each symbol.
3. The quantity, value and order of all literals used.

Clearly, with this information, a second pass over the source statements can fill in all valid symbol values, locate literals, and assign block origins.

This two-pass philosophy is followed in COMPASS. The main control of pass 1 consists of the following operations:

1. Reading a statement.

2. Editing the statement to remove micro and concatenation marks.

3. Looking up the operation code and jumping to the appropriate processor.

4. Recording the statement on the intermediate file along with other pertinent information to be passed along for pass 2 processors.

5. Returning to step 1.

Similarly, the main control of pass 2 consists of:

1. Reading a statement from the intermediate file and preparing the line for listings.

2. Jumping to the appropriate processors for assembly, which:

   a. Evaluate the fields of the instruction,

   b. Format the results for listable output,

   c. Output the results for binary output.

3. Returning to step 1.

In addition to the main passes 1 and 2, there are two pseudo-passes 0 and 3. Pass 0 consists of all operations that are performed before the first assembly is begun: scanning the COMPASS control card, setting filenames and internal flags accordingly, allocating file buffer areas, and obtaining system macro, micro, and symbol definitions from system text. Pass 3 is performed at the conclusion of each assembly, and consists of printing the symbol table and cross-references if requested.

Let us now look in more detail at each of the steps outlined above. Omitted for the present are the inter-pass functions which initialize each pass and terminate each assembly. They can be clarified only when the reason for their operations is known.

1.2 Pass 1 Functions.

1.2.1 Reading a Statement.

In COMPASS, the subroutines INPUT1, UCARD, and RNS are all involved in the reading process. Statements may come from the primary input medium, in which case INPUT1 directs RNS to create a statement of cards from the input file. Internally, this statement is generated in a long card area, one character per word. The sequence fields (columns 73-90) are separated and stored elsewhere. The card is recollected for listing purposes only.

There are, however, other sources of input. Cards can come from macro call expansions, duplication expansions, ECHO expansions, remote assembly, and XTEXT sources. All this is accomplished by INPUT1 in conjunction with UCARD. When INPUT1 is called to get a statement, it first examines the STACK. If the stack is empty, then input from the input file is assumed. However, a macro call, a DUP, ECHO, HERE (or an END card), or XTEXT can cause the other parts of the assembler to store packed images of statements internally, and then make a stack entry. Thus, when INPUT1 is called, the stack is non-empty, and the statements are unpacked from the appropriate table area by UCARD.

All information concerning the source of statements is contained in the stack. STACK is a push-down list; i.e., a stack entry can be made even when statements are being derived from internal tables. All information about the source of the next statement is contained in the top-most entry of the stack, and restored there after each statement has been created. Thus, true push-down facilities are realized.

The INPUT1 routine also initializes various control cells for each statement.

1.2.2 Editing the Statement.

While reading a statement, COMPASS scans all the characters of the statement to look for the two editing marks: the micro mark (65B) and the concatenation mark

(64B). If none exists, the editing process is in effect a null operation. If they do exist, however, the statement as it exists with the special marks is conditionally written out on the intermediate file, the intermediate flag (MICFLG) is set, and the statement is edited. Thus, for each statement with at least one of the editing characters present, two statements are recorded on the intermediate file. This is done for listings only, since the unedited line is no longer needed. If this line will not be listed, it is discarded without writing it on the intermediate file. The assembler is no longer cognizant of any editing marks; they are ignored from this point onward.

1.2.3 Determining the Statement Type.

After editing, the statement is broken into parts. This process, for efficiency only, isolates the location field entry and the operation code field entry. It also sets pointers so that the beginning of the operation and address field can be easily located. This solves the free-field problem of COMPASS.

After setup functions, the operation code entry is looked up in the operation code table. At this point, IF-skipping must be considered.

The assembler contains several IF-type operations which permit the programmer to specify that the statements following are not to be assembled. The count of the number of statements to be skipped either is declared on the IF instruction or is declared by the use of the ENDIF operation. During the classification process, operation codes are looked up even if statements are being skipped. However, only when the look up results in an operation of internal type 3 (END, ELSE, and ENDIF) does control transfer to the pseudo-operation processor. In all other cases, during IF-skipping, the card counts are merely reduced, if relevant. If the F list option is selected, the skipped card is written on the intermediate file with the NOAS flag set, otherwise the card is discarded since it is no longer needed. The NOAS flag is interpreted in pass 2 processing to determine if the statement is to be ignored.

1.2.4 Operation Processors.

The real work of the assembler is performed by the operation code processors. The simplest ones are those that analyze machine operations. Note that in the case of central processor instructions, the operation code field entry does not define the instruction. In fact, when there is no match in the operation code table, the assembler assumes that the operation is a central processor or OPDEF instruction (in CP assemblies) and performs a detailed analysis on the statement to determine the actual instruction.

Once the instruction is determined, the operation must be accounted for in terms of its length. An internal position counter contains the current position in a word. This is updated as required. Situations occur that cause the position counter to be reset to beginning of the next word (such as a 30-bit instruction occurring when fewer than 30 bits remain in the current word, the statement contains a location field entry, a force upper is implied from the prior line, or a force upper is implied from this line). All this facilitates the definition of symbols, the allocation of instructions, and definition of block lengths.

The address fields of instructions are also examined for literals. If a literal is found, it is evaluated and searched for in the table and, if not in the table, is added to the growing literal table.

Other operations, such as some pseudo-operations, also affect the internal origin, location and position counters. Consider, for example, BSS and BSSZ, which advance the origin and location counters.

Operations, such as EQU, =, SET, MAX, MIN, and MICCNT, cause entries to be made in the symbol table. MICRO, OCTMIC, DECMIC and sometimes BASE, CODE, and QUAL result in an entry in the micro table. COMPASS evaluates IF-type operations; when code is to be skipped, it sets the control words IFNAME and IFCNT.

The two main classifications of operations remaining are the counter control operations (USE, ORG, ORGC, LOC, POS, USELCM) and the definition operations (MACRO, MACROE, OPDEF, RMT, XTEXT, DUP, ECHO). The counter control

operations rely upon an internal table known as USETAB. Because code may be assembled under different blocks, this table maintains the last known information about a block. A block (or table entry) has the following associated information:

Name of block
Memory type (SCM or LCM)
Block type (local or common)
Current length
Current position
Force upper on next instruction

When assembling under a given block, these values are maintained in the control words ORGCTR and POSCTR. The occurrence of a USE, USELCM, ORG, or ORGC, however, causes these values to be stored in USETAB, and a different set of values is used. ORG and ORGC also reset the length and position values.

The definition operations give COMPASS its higher-level language facilities. When COMPASS encounters an RMT operation, it reads subsequent cards unedited from the input described and packs them unedited into the remote table (RMTAB) or labeled remote table (LRMTAB). The closing RMT operation returns control to the main flow.

For a MACRO, MACROE or OPDEF operation, COMPASS reads subsequent statements from the input, scans for formal parameters and replaces them with parameter markers, and packs the unedited statements into MACDEF, the macro definition table. COMPASS stores the macro name, with appropriate pointers, in the operation code table. Thus, a subsequent macro name appearing in the operation code field will be detected in the operation code table.

For MACROE the list of formal parameter names is also stored into MACDEF for use when the statement is expanded.

For a DUP operation, COMPASS reads the subsequent cards and packs them unedited into TEMTAB. When the duplication range is terminated, the text in TEMTAB is added to the end of DUPTAB and COMPASS makes the stack entry and starts duplication.

A similar operation occurs on XTEXT, but the source of statements is restricted to an external file. INPUT1 is

not used; instead RNS is called directly. The statements are packed unedited into LASTAB, and then COMPASS makes a stack entry directing INPUT1 to read from that table until it is exhausted.

For an ECHO operation, COMPASS reads subsequent statements from the input, scans for formal parameters, replaces them with parameter markers, and packs the unedited statements in TEMTAB. When the duplication range is terminated, the text in TEMTAB is added to the end of ECHTAB, a stack entry is made, and duplication starts.

Operator defining operations, CPOP, PPOP, OPSYN, and CPSYN, define actual machine instructions. Entries are made in the opcode table for these operations.

IDENT actually goes through an END processing sequence. All current storage is allocated, the USE table is relocated, the symbol table and segment table are relocated. The literal table and the entry point tables are bound off and a new USE block group is started.

1.2.5 Writing the Intermediate File.

CCMPASS writes the intermediate file after processing operations. The file contains the following information in addition to the copy of the source statement:

- The result of the operation code lookup.
- Error flags, and other indicator flags.
- Values being transmitted to pass 2.

In COMPASS, the intermediate file is built in storage. When and if storage overflows, it is dumped onto a scratch file. Thus, for small programs, no scratch file is required, speeding up the assembly time.

1.2.6 Returning for Next Statement.

Until the END statement is encountered, control reverts to the read section for the next statement. END processing, which is actually rather complex, is explained in section 2.8.

1.3 Pass 2 Functions.

1. Reading the intermediate file and recreating the statement and its print line (if required).

2. Jumping to appropriate processors.

The Pass 2 control first examines several flags recorded on the intermediate file. If the NOAS flag is set, the line is being IF-skipped, and it is listed depending upon the list controls. If the TXTFLG is set, the statement is part of a definition (RMT, MACRO, ECHO, DUP). Such lines are not assembled, they are included for listing depending upon the listing controls.

If a line is none of the above, the result of the pass 1 operation code lookup is consulted, and the appropriate jump is taken.

For machine operations, the remaining processing includes evaluation of the various fields of the instruction. The instruction and its structure is known from pass 1. Pass 2 substitutes address and register values. The result of the evaluation is listed and delivered to the binary output routines.

For pseudo-operations, processing may or may not be trivial. For operations such as LCC, MICRO, and RMT, all the operations occurred in pass 1. The pass 2 processors jump to the list routines. Counter control operations, however, require an operation similar to that in pass 1. In addition, partial binary words are saved when assembly switches to a new block. Thus, the USETAB is expanded to include the following information about each block:

- Name of block.
- Memory type (SCM or LCM).
- Block type (local or common).
- Current position within the current word.
- Force upper flags.
- Actual origin of the block, as calculated at the start of pass 2.
- Partial binary word.
- Number of relocatable address fields in the partial binary word.
- Block maximum, as calculated at the end of pass 1 based upon the known length of the block.

For a relocatable assembly, RELTAB contains relocation information for the current partial binary word for each USE block.

1.4 Example of Assembly Process.

As an example, let us follow a card through the assembly process starting in pass 1.

Consider that the following statement appears on the input file:

```
ALPHA     SA6       B4-6      SOME COMMENT          00225
```

This statement is read in by INPUT1 and RNS, and arrayed in a long card area. Editing for this statement is a null operation. The setup operation isolates ALPHA as the location symbol, and SA6 as the operation symbol. Pointers to the S and the B facilitate finding the operation and address fields.

In this example, operation code lookup fails to find an operation code entry of SA6, so COMPASS assumes it is a central processor operation code (provided that this card is not being IF-skipped). COMPASS constructs a mask based upon the syntax of the statement, which indicates that this line is a 30-bit instruction of the format SABQ. This format is found in the OPTAB and has the value 51ijK. The location field is processed causing a force upper. The symbol ALPHA is defined as the current value of the location counter. COMPASS examines the address field for literals--this example has none--and writes the line on the intermediate file. Of course, the assembler decrements the position counter by 30 to reflect the room required for this instruction.

When this line is reread in pass 2, control again reverts to the central processor instruction routine which examines the structure of the operation code, and evaluates the registers and address. Again, the location symbol causes a force upper, but the symbol ALPHA is not redefined.

The assembler places the values into the instruction prototype. The result for this example is 51647 77771. This value is placed in the listing area and delivered to the output routines. Again the position counter is decremented by 30.

1.5 Pass Initialization.

The pass 1 initialization consists of presetting all words to their nominal values. This includes clearing the origin counter, location counter, all table sizes, force upper flags, etc. It assumes a central processor program, and sets the word length and position counter to 60. The nominal blocks ABSOLUTE, 0, and LITERALS are preset into the USETAB. COMPASS then begins reading cards, checking for a preliminary IDENT card.

When a second IDENT card or the END card is detected during pass 1, the bind-off functions consist of forcing upper on all blocks. Default symbols are defined. The size of each block is known. The length of the program can be deduced from the sum of the local block sizes. A SEGTAB entry is made of the program length. The END card location symbol, if present, can be defined. The USE, segment, and symbol tables are relocated.

At the start of pass 2, preliminary operations include the following:

- Presetting of listing controls.
- Listing of storage allocation which includes segment addresses and lengths, binary control cards, and block names, sizes and origins.
- Listing of entry points.
- Listing of externals.
- Initialization of the USE table and the binary output routines.
- Dumping of the literal table.

When a second IDENT card or the END card is encountered in pass 2, the terminal functions dump the binary output. For the END card the functions also produce and write the linkage tables, the REP tables, etc. Some more listings are produced: the literals, the default symbols, statistics, and the reference table.

This concludes the assembly of one subprogram. Batching them is rather trivial, and the assembly process is complete.

2.0 COMPASS METHODOLOGY

Various methods employed for table management, macro processing, input stacking, output processing, etc., are the core of the COMPASS assembler. These methods are described in this chapter. Detailed table formats and subroutines involved are listed elsewhere.

2.1 Table Management and Organization.

Practically all of the tables in COMPASS are dynamic. Thus, no fixed length tables exist. Management routines ALC, MTU, MTD, ACL and two vectors of table pointers control these tables. The two vectors are:

ORIGINS — Contains the current first word address of each table.

SIZES — Contains the current length of each table.

For example, the INTER table, the first table in the managed list, can be located by consulting the address at location ORIGINS+0. Its current size is in SIZES+0.

Whenever a table size is to be increased, the assembler calls the ALC routine via a macro:

```
MANAGE    table-name, increment
```

This expands into:

```
SX1       increment       (deleted if increment=X1)
SA0       table-name      (deleted if table-name=A0)
RJ        ALC
```

The first duty of the table manager is to find room if additional space is available in the field length. If room is not available, however, it cannot be said that a particular table overflowed. For example, it may be that the Symbol Table is excessively large due to an error in the programmer's use of symbols, but the overflow occurred while allocating room to the literal table. Thus, a characteristic of dynamic tables is that when one table overflows, all do.

The entire table area starts at the end of the code and I/O buffers currently in use and terminates at 10 less than the original field length. The 10 words at the end are for the many move loops that, for efficiency, fetch entries and then decide not to use them. To prevent address range errors, the ten-word area at the end is never used for table area.

Although COMPASS never makes a MEMORY call to the operating system, such a call is possible to expand the table area. The information required is already present. CP.AFLS contains the current field length. A MEMORY call that increments CP.AFLS by 100B would cause SIZCORE to be incremented by 100B to reflect this change. The table manager discovers that there is more room.

Tables are stored in the order in which they are defined in the allocation vector via the TABLE macro. To add a new table, add a new TABLE macro call. Assume a new table named TAB is to be added. Then:

```
TAB             TABLE
```

accomplishes this feat. TAB will be a table that can be used in both passes. If, however, one table is to overlay another, name sharing can be used. For example, suppose TAB is to overlay the pass 1 table DUPTAB. Then, follow the definition of DUPTAB with:

```
TAB             TABLE           DUPTAB
```

For a new table, the TABLE macro expands to:

```
tnam            EQU             *-ORIGINS
                CON             tnam
                ORG             *-1
O.tnam          CON             BUCKET
                RMT
L.tnam          CON             0
                RMT
```

If a table is to share another's space, the macro expands to:

```
tnam            EQU             eqiv
O.tnam          EQU             O.eqiv
                RMT
L.tnam          EQU             tnam+SIZES
                RMT
```

Thus, the name of the table (tnam) is defined as an ordinal in the allocation vector (of which ORIGINS is the beginning); the symbol O.tnam defines the origin of this table; the symbol L.tnam defines the location in SIZES containing the length of this table. The symbol O.eqiv identifies the equivalent table.

The auxiliary routines used by the management process are of interest to a user also. The MTD and MTU routines pack the tables to low or high core. The routines are used in COMPASS as follows:

MTU (pack to high) when low core limits are changed.

MTD (pack to low) when it is desired to free the maximum amount of storage, for example, for reading the SYSTEXT record as well as sorting the Cross-Reference table.

The routine ACL adjusts the low-core pointer for the tables. This is used when the I/O buffers are managed to obtain the maximum amount of storage to sort the cross-reference table.

ALC allocates storage in a multi-phase operation. That is, it makes successive tries, in the following order.

1. It first determines if enough room is available for the expansion of the table between it and the next higher table. If there is, the SIZES entry is updated.

2. Otherwise, if the sum of the SIZES vector and the amount requested is less than or equal to SIZCORE minus a threshold (1000B) it will try to reshuffle the tables to buy the storage. When this occurs, a re-allocation phase packs all the tables in high core and then re-distributes them in low core so that the growth room is proportional to the lengths of the tables. One-half the available storage is divided equally among the tables, the other one-half is divided proportional to the current table lengths.

3. If this fails, an overflow condition exists. The emergency procedure is pass dependent, as follows:

   a. Pass 0 Overflow Processing.

      Processing of the current system text aborted with the dayfile message INSUFFICIENT STORAGE FOR SYSTEM TEXT.

b. Pass 1 Overflow Processing.

If the intermediate file resides in core, it is dumped from the INTER table onto the file ZZZZZRL. The intermediate table is emptied and the flag INTERIO is set to notify WINTER and RINTER that the intermediate file is on mass storage. The INTER table is not used again.

If the intermediate file is not in core (INTERIO ≠ 0), the job is aborted with the dayfile message PASS 1 TABLE OVERFLOW ASSEMBLING xxxxxxx.

c. Pass 2 Overflow Processing.

If the intermediate file resides in core, it is dumped onto the ZZZZZRL file with an end-of-record write. ZZZZZRL is then rewound, and read into the intermediate buffers. INTERIO is set so that RINTER obtains subsequent information from the I/O file. Space is thus bought back by emptying the INTER table, and processing continues.

If there is still insufficient memory space and the common and external linkage tables are not empty, DLAST is called to write them to the binary output file and release the memory space they occupy, and processing continues.

If there is still insufficient memory space and the cross-reference table still resides in core, it is dumped onto the file ZZZZZRM. The cell REFIO is set so that ENTREF writes references to the file. Space is thus bought back by emptying REFTAB, and processing continues.

If there is still insufficient memory space, the job is aborted with the dayfile message PASS 2 TABLE OVERFLOW ASSEMBLING xxxxxxx.

d. Pass 3 Overflow Processing

Formatting of the cross-reference table for printing uses MEMORY as a temporary table to invent columns and rows. If overflow occurs in pass 3, the MEMORY table is discarded with no message.

2.2 Recursion Methods and Stack Processing.

In COMPASS it is possible to call macros (a) within macros, (b) within DUPs, (c) within ECHOs, (d) within RMTs, and (e) within XTEXT. This is accomplished primarily with the STACK, and secondarily with the other push-down tables MARDIS, MARGS, DUPTAB, ECHTAB, RASTAB, and LASTAB. The formats of these tables are described in chapter 4.

Consider how recursion is accomplished. The method is to retain all the information concerning the source of cards in the push-down list known as the STACK. When INPUT1 is called to get the next card, the top-most entry in STACK is consulted. When that card has been processed, the stack pointers are reset so the next card will be generated from the next address in the packed card image area.

The stack maintains the word address for the card images within one of five tables. It either points to an area in MACDEF for macro expansions, DUPTAB for duplications, ECHTAB for ECHO processing, RASTAB for remote lines, or LASTAB for XTEXT expansions. As COMPASS extracts cards from one of these tables, it resets the stack pointer. When the end of this source is reached, the stack is pushed up to eliminate that top-most entry. Thus, even if a card unpacked from MACDEF is, itself, a macro call, the conclusion of this inner call pushes up the stack; i.e., it eliminates the top-most entry, and returns to the outer macro as the controlling entry.

In addition to word addresses for card images, STACK also maintains pointers to the tables used for substituting macro arguments. Two tables are used for this: MARDIS which contains pointers indexed by parameter number, and MARGS which contains the actual parameter strings.

To expand a macro, a number of words in MARDIS are allocated corresponding to the number of parameters in the macro definition. Then the arguments are scanned, or invented in the case of local parameters, and the relative address of the beginning of the string in MARGS is stored in the corresponding address in MARDIS. This indirect method was selected since lengths of actual arguments vary.

The STACK entry for a macro points to the beginning of the table of pointers in MARDIS. Also, the original length of MARGS is saved so that when the macro is terminated, the

lengths of MARGS and MARDIS can be set to the values they had When the macro was called, and the storage can be released.

The substitution of parameters into an inner call is accomplished through the call rather than through the inner expansion. Presume that a macro call occurs in a macro:

```
OUTER     MACRO     X,Y
INNER     MACRO     A,B
          SA1       X+A
          AX6       X1
          SA6       Y+B
INNER     ENDM
          INNER     3,4
OUTER     ENDM
          .....
          OUTER     GAMMA,DELTA
```

The processing of the OUTER macro call will be as follows:

```
          OUTER  GAMMA,DELTA
```

This is a macro call. OUTER is located and its definition extracted. It has 2 parameters, so two words are allocated in MARDIS and enough words to hold the actual parameters are allocated in MARGS. Assuming this is an outer level macro, these tables were empty. Thus:

LOCATION	OPERATION	VARIABLE SUBFIELDS	MARDIS	MARGS
			0	GAMMA
			1	DELTA

```
INNER     MACRO     A,B
```

This is the first card generated by unpacking the expansion of OUTER. Note that there were no argument substitutions. This is definition, so control goes to PMACRO to define the macro. Reading continues from MACDEF with no change.

	SA1	GAMMA+A	Note the argument substitution. Because A is a formal parameter, it is changed to a parameter marker in the table.
	AX6 SA6	X1 DELTA+B	Again, parameter substitution has occurred.
INNER	ENDM		This terminates the definition of INNER.
	INNER	3,4	This is an inner macro call. Thus, MARGS and MARDIS are update to look as follows:

MARDIS	MARGS
0	GAMMA
1	DELTA
2	3
3	4

	SA1	GAMMA+3	The GAMMA came from the definition, and '3' came because there was a mark for the first parameter. The STACK entry for this inner macro gives '2' as the origin of MARDIS.

And so on. The quantities saved in the STACK are:

- Byte address, pointing to the next card image.
- L.MARGS at the beginning of the process, so that it can be reset when the stack is pushed up.
- L.MARDIS so that the beginning of the descriptors for this stack entry can be found. It is also used to reset the stack.

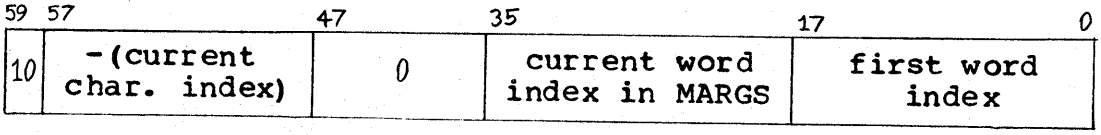
2.3 Definitions of Macros and Opdefs.

The COMPASS assembler recognizes macro and opdef definitions by the occurrence of MACRO, MACROE, or OPDEF in the operation code field. When defining the macros, COMPASS processes this header card to extract the name of the macro, and the names of the formal parameters. The list of formal and local parameters is saved while the definition goes on so that occurrences of parameters in the text can be detected. When such an argument name is found, it is replaced with a 77xx (where xx is the parameter number) and is stored in MACDEF. The operation code table entry points to the first line stored in MACDEF. In this way, the definition is reclaimable, and the formal parameter names are no longer around. Thus truly they are formal names and have no meaning outside this context of definitions. For OPDEF, of course, the process of deriving the operation code name is a bit more complex, but beyond the name extraction, the process is the same. These functions are performed in PMACRO.

For MACROE the formal parameter names are also stored in a list in MACDEF so they can be recognized as keywords when the macro is called.

At expansion time, the count of formal parameters is extracted, and the required number of parameters is generated or scanned. In OPDEF, however, this requires isolating the address elements from the register names, and throwing away the register operators. This is accomplished by means of evaluating and classifying everything in the address field. If it is a register, the register is stored as the actual parameter name. If it is an address, it is saved for later storing in MARGS. In MACROE the formal parameter names are moved to MARDIS. The keywords on the call point to MARGS; any keyword not found points to a null argument.

When an IRP is encountered during expansion, the current MACDEF index is recorded (for later resetting) in the stack entry. The MARDIS entry for this parameter is changed to contain the current character and word position of the MARGS entry and the IRP flag is set so that the MARDIS entry is:



When the terminating IRP is encountered (flagged by a type U) the MARDIS pointers are advanced to the next argument delimiter and the current statement index is reset to the start of the IRP. When the argument list is exhausted, the MARDIS entry is reset and generation continues with the card following the terminating IRP.

2.4 Duplications.

DUP employs stack processing. During the first pass of assembly, when a DUP is encountered, the cards which comprise the definition of the DUP are packed into TEMTAB and then transferred to DUPTAB. A T card is generated as the terminator card, and the STACK is pushed down to cause generation. The duplication controls are:

1. Initial card address (points to the T card).
2. Duplication count (set to one greater than the programmer's count).
3. DUPTAB reset quantity (set to the size of the DUPTAB when the duplication definition started).

Thus, in the standard case, the first card unpacked is the T card. Since this is a DUP entry, the next card pointer is reset to the beginning of the definition, and the iteration count is reduced. If it becomes zero, the duplication is complete, and the length of the duplication table is reset and the stack is pushed up. In this way, the outer controls are restored, even for an outer DUP. If the iteration count is non-zero (and the STOPDUP flag is not set) when the T card is read, then card generation resumes with the first card of the definition set.

2.5 Echoed Lines.

During the first pass of assembly, when an ECHO is encountered, the definition card is scanned and entries are made for the formal parameters in MARDIS and the substituted arguments in MARGS. The cards which comprise the definition of the ECHO are written into TEMTAB substituting parameter markers for the formal parameters. TEMTAB is then

transferred to ECHTAB, a terminal T card is generated as the termination card, and the stack is pushed down to cause generation. The echo controls are:

Initial card address.
Initial MARDIS address.
Initial MARGS address.
Current card address.

The argument pointers in MARDIS point to the start of the argument as well as the current position of the argument. When the T card is encountered each argument pointer is advanced to the next argument delimiter. When any of the argument lists have been exhausted or the STOPDUP flag is set, the echo is terminated by pushing the stack up and returning the space in ECHTAB, MARDIS, and MARGS.

2.6 Remotes.

When COMPASS detects an RMT operation, it starts storing the ensuing cards into RMTAB or LRMTAB. When the closing RMT is found, assembly resumes from the previous source. The actual assembly of the remote lines is deferred until a HERE (or END card) is encountered. For unlabeled remote code when this is found, a terminal card is packed into the RMTAB (a T card in the case of HERE, and an E card in the case of END). The RMTAB is moved to RASTAB, and RMTAB is emptied out. The stack is pushed down so that assembly starts from RASTAB.

For labeled remote code, when a HERE is encountered, the LRMTAB is searched for corresponding text. This text is moved to RASTAB one group at a time and the corresponding text is deleted from LRMTAB. When all the text has been moved, a terminating card is added to RASTAB and the stack is pushed down so that assembly starts from RASTAB.

If RMT coding appears in remotely generated lines, the process is identical. The lines are stored in RMTAB, and pulled out by the next HERE or END. If there are remote lines left over at END time, they are lost with no diagnostic.

2.7 Methods of Binary Output.

As COMPASS generates binary output, it makes calls to the following output routines:

BINOUT	Writes one byte of information.
DWORD	Dumps the accumulated word into BINREC (relocatable assembly) or MEMORY (absolute assembly).
DBSSZ	Dumps accumulated BSSZ coding.
DFIRST	Dumps the preliminary information.
DLAST	Dumps the terminal information.
DDUMP	Dumps the memory image for absolute programs.

As data is generated for a word, it is added to the word's binary output (BINWORD, BINREL) by calls to BINOUT. In pass 2, ZFOUP is called and then DWORD which dumps the accumulated word.

The format of BINWORD is just the obvious amount of data, either 12 or 60 bits. BINREL contains the relocation information.

A call to DWORD may follow each call to BINOUT. When there are USE or ORG statements, the current contents of BINWORD, BINREL are saved in USETAB and RELTAB to be pulled out later. Of course, END card processing cleans all.

REP, REPC, and REPI statements cause output directly to the binary output file.

DWORD can make entries into the link and common tables to be dumped later.

2.8 End of Pass Processing.

Most of the functions of the assembler for the pseudo-operations are rather obvious. The reference manual defines what has to be done. However, the processing to be performed at the end of each pass is not obvious and is discussed here.

2.8.1 Pass 1.

When Pass 1 is complete, COMPASS performs several bookkeeping functions to complete the information needed for pass 2. This involves performing a HERE-type operation which assembles all waiting entries in RMTAB. When this is complete, all blocks are forced upper by the subroutine YFUALL. Next, the default operations of COMPASS are accounted for. This involves examining the entries in SLITS, which contains the names and types of the default symbols. Each one that has never been defined is given a definition as either external or local. In the latter case, the size of the 0 block is increased.

At the end of pass 1, the total size of all blocks is known, and all symbols which are to be defined have been defined. Thus, it is possible to sum up the sizes of the local blocks and define the possible END card symbol. Furthermore, if the intermediate file overflowed to disk, the buffer is emptied and the file is rewound.

Now the USE table is relocated. This involves progressing through the USETAB and assigning origins to local SCM and LCM blocks. At the same time the loader-type relocation is created. Note that during pass 1, relocation is relative to the individual blocks. Now actual origins are assigned to the blocks. Each common block is assigned a relocation number starting at 2. All local blocks have a relocation number of 1. Of course, for an absolute program, all blocks are local with a relocation number of 0.

Because the symbol table is still relative to the pass 1 procedure, all symbols must be relocated. This involves either adding or subtracting the origin from the symbol value, and setting the new relocation. At this time also, redefinable symbols are set undefined so that pass 2 will catch any reference to the symbol prior to its first SET instruction.

The segment table was created during pass 1 so it must also be relocated. The last address of each segment is relocated and the first and last default symbol table, entry point table, and literal table index are set up for each segment.

2.8.2 Pass 2.

At the end of pass 2, the only critical functions are those pertaining to dumping remaining binary output. All blocks are forced upper, which causes a DWORD call for each USE block that is not positioned at a word boundary. The partial binary word and relocation controls, saved in USETAB and RELTAB, are cleaned out and included in the binary output. DBSSZ is called to dump any waiting BSSZ code, DLAST is called to dump the common and external linkage tables for a relocatable assembly, and DDUMP is called to dump the memory image for an absolute assembly.

The actual bulk of END processing pertains to the listings. This includes the listing of literals, the listing of default symbols which COMPASS defined, and the error comments and reference table.

Finally, the binary output is terminated. At this point, if any fatal errors were detected, the binary output is erased. The number of sections written (due to IDENT or SEGMENT operations) is maintained in DKCNT which in turn dictates the number of backspaces to be issued.

The only non-trivial process here is the reference table entry processing which is described below. This processing phase is sometimes referred to as Pass 3.

a. If the reference table overflowed to the scratch file ZZZZZRM, that file is completed, the binary and intermediate buffers are returned, all extra tables are cleared and moved down, all of the assembler code above subroutine PRT is dumped to the end of file ZZZZZRM to maximize the amount of room available for the reference table, the file is rewound, and the references are read back into memory. A count is kept of all references that will not fit in memory at this point.

b. A link of references is created to tie symbols to their references. For this process bits 59 - 42, the hash link fields of the symbol table 9 are first zeroed. Each reference entry defines a

symbol via an index into the symbol table. The link field of that symbol is moved to the reference table entry, and the number (or index) of the reference entry is stored back into the symbol link field. In this way, starting at a symbol equivalent, the link field points to the earliest reference, thence to the next reference, and stops with a zero link.

c. Each symbol that is not to be printed is removed from the symbol table by zeroing its entry. A symbol is removed if any of the following is true:

- its NOREF bit is set,
- it has no references, it is an SST or XTEXT symbol, and LIST T is not active,
- it has no references, it is not an SST or XTEXT symbol, and LIST N is not active.

d. The symbol table is processed to left-justify the symbols at bit 47, keeping the QUAL index in bits 48 - 59, and to squeeze the table down to suppress the zero entries.

e. After presetting the subtitle for the listing to "SYMBOLIC REFERENCE TABLE." and adding a message if references were lost, a shell sort is performed on the symbol table. This groups the qualified symbols together since the qualifier index is in the higher order bits.

f. To generate the references, COMPASS generates the first 40 columns for the definition of the symbol, and perhaps notes a U-error. Then the references for a symbol are converted to display code and added to the table MEMORY. The number of rows required to contain the references for a symbol is computed and an eject is performed if they will not all fit on the current page.

g. After all references are listed, COMPASS is read back into core from the end of file ZZZZZRM and the file is rewound again.

2.9 Defining System Macros.

Initialization code is executed when COMPASS first gains control. This code is overlayed by tables and buffers, so that the code space is reclaimed. The process of loading the system text involves accessing the library for an overlay load of the record whose name is preset in low core. Normally this contains the name 'SYSTEXT' although, with the S-option on the COMPASS call card, this name can be changed. If the G option was specified, the overlay is read from the specified file rather than the system library. In either case, the overlay is read and the tables are loaded into SSYMS, MACDEF, SYSMIC, and entries are made in the opcode table. If the job's ECS/LCM field length is non-zero, as many as possible of these tables are stored there, since they are accessed only once at the start of each assembly.

The resultant length of MACDEF is saved in LSYSMAC so that when programmer macros are discarded at the end of assembly, system macros are saved for the next subprogram.

2.10 Inter-pass Processing.

Before the start of pass 2, some bookkeeping functions must be performed before the actual work can begin.

The storage allocation page is listed if not suppressed. This involves unpacking the binary control cards from IDTAB and listing the USE blocks, entry points and externals.

The ORG counter in the USE table blocks is reset and the intermediate is searched for the IDENT card. After the preliminary loader information and literals have been dumped the real pass 2 processing starts.

2.11 Packed Card Format.

COMPASS includes a SQUEEZE subroutine that condenses a source statement for storing. SQUEEZE is called whenever a statement is to be recorded in either a definition table (MACDEF, LASTAB, RMTAB, LRMTAB, ECHTAB, IDTAB, or DUPTAB), or added to the intermediate file. Squeezed statements are unpacked by UCARD and RINT.

In the format of the packed image, the compression technique involves the recognition of sequential blank columns, colons, and end of statement. Three or more blanks are replaced by the 12-bit code 00xx, where xx is the number minus one of blanks being recorded. Thus:

55	one blank
5555	two blanks
0002	three blanks
0003	four blanks
...	...
0077	64 blanks
007755	65 blanks
00775555	66 blanks
00770002	67 blanks

etc. The code 0001 represents a single 00 character (colon), and 0000 marks the end of the statement. The last word is filled on the right with binary zeros.

When packing a card, at least one column, STYPE, is always recorded verbatim. This is useful in some optimized stack loops which assume a non 00 mark is the first character when they address the packed card formats. Also, the character following a 77 character (semicolon or parameter mark) is recorded verbatim if it is a 55 (blank), so that formal parameter number 55 followed by two or more blanks is recorded as 7755 rather than 7700xx. If the 77 character is followed by a colon (0001) or end of statement (0000), it is packed as 770001 or 770000; this is safe because there is never a parameter number zero.

When SQUEEZE is called, the contents of CARD, relevant up to LASTCOL, are packed into an area SQIMAGE. The number of packed words is in SQLGN. SQLGN is a flag to mark the packing process; when SQUFEZE is called and SQLGN is non-zero, no packing is performed.

2.12 BSSZ Processing.

This discussion considers both the BSSZ instruction, itself, and the blank operation code with a location symbol. To economize on the binary output, BSSZ processing produces information in a non-obvious method. Two controls are on this processing:

ORGBSSZ records the last known origin of BSSZ's;
CNTBSSZ records the count of words to be zeroed.

When a BSSZ is encountered out of sequence with the earlier ones, DBSSZ is called to generate the zeros, and start a new sequence. A new BSSZ in sequence causes CNTBSSZ to be updated.

When DBSSZ is called, the zeroing process occurs. For relocatable output, if CNTBSSZ is less than 5, the number of zero words is generated. If it is 5 or more, then one zero is produced, and a REPI entry is generated.

For absolute output, however, the zeros are always produced explicitly at the time the BSSZ is encountered, so that the loading takes place immediately.

2.13 Listing Controls.

A great deal of effort went into the planning for list options to allow maximum flexibility in their use. COMPASS list selection is two part -- there are list options that force listing of lines that would not normally be listed and the rest of the list options discard lines for one reason or other. This discarding process can occur at several places. It is important to the speed of the assembly process to discard lines as soon as possible so extra steps can be eliminated. At the end of pass 1 processing of a statement, tests are made on statements that require no pass 2 processing to determine if the statements can be discarded.

If the error flag is not set, and

master list is off (LIST -L)

or

external list is off (L = 0 on COMPASS card)

or

LS = 0 and SYSFLG = 1

or

LX = 0 and LIBFLG = 1

or

LM = 0 and MACFLG = 1

or

LE = 0 and ECHFLG = 1

or

LD = 0 and RMTFLG = 1

or

LF = 0 and NOAS = 1

the statement is discarded. In pass 2 when a statement is read from the intermediate file, it is always read into the long card buffer. Additional tests are made at this time to determine if the card will be listed or whether the creation of the print line can be bypassed. Several flags are used in pass 2 to control the various sequences that can occur during the listing process. When a statement is read the following flags are set:

DETFLG = 0	This is the first of possibly several list lines.
CTYPE = 0	This is not a SPACE, TITLE, TTL, or EJECT card.
PLFLG = 0	There is no print line ready to be printed.
NLFLG = 1	If the tests performed on the statements that require no pass 2 processing are met for this statement. (No list flag.)
DLFLG = 1	If the listing of this line is to be deferred until the octal for this line is generated. (Set only for macro calls.)

If the line contains an error, a call is made to LDL to list a deferred line if one is pending and the print line is created. (PLFLG is set to 1 by CPL to indicate that a print line has been created.) After pass 2 processing of this statement, another sequence is gone through to determine if the line should be listed.

For a line that generates no code, the following tests are made to determine if the listing of the line can be bypassed:

If the error flag is off, and

If RINTER found a condition to inhibit listing

or

LD = 0 and DETFLG = 1

or

LC = 0 and CTYPE = 1

or

master list is off (LIST -L)

the line is not printed.

The remaining list options are processed outside those already described since they pertain to listing lines that would normally not be listed.

A-Option: This control is detected inside RINTER. If a card is read that contains MICFLG = 1, and the A-option is set, RINTER calls LDL, CPL, then LISTER to list the line, controlled as outlined above.

B-Option: This control is detected by PLM and controls the listing of the storage allocation for a program. Since this page is printed prior to pass 2 processing, the setting of LB at the end of pass 1 controls the listing of this page.

G-Option: This control is detected when LISTERG is called by those routines that generate binary output. If a deferred line is ready, this line is listed along with the present octal without creating a print line from the statement that generated the octal. If no deferred line is ready and LIST L is on, the flag FLIST is set to force listing of the line and the print line is created by calling CPL.

R-Option: This option controls the setting of SUPREF, which dictates whether or not to accumulate references. The R-option also is checked when the reference table is about to be created.

External list control is also in force. If LISTFG = 0 or the LO option on the control card was used, then the LIST pseudo-operation code is ignored completely.

2.14 Operation Code Recognition Order.

When examining a line, the assembler isolates the operation code field entry and searches for it in the operation code table.

This table contains:

1. CCMPASS pseudo instructions other than LOCAL.
2. PP machine instructions.
3. System and programmer defined MACRO and MACROE names.
4. Programmer defined OPSYN and PPOP instructions.

If a match is found, the operation is identified. However, if no match is found and a CP assembly is underway, an additional analysis is performed. COMPASS syntactically analyzes the entire address field and the operation code field and again searches the operation code table for a match. The format for this search differs from the previous format so that for the purposes of the search another table is being searched that contains:

1. CP instructions.
2. System and programmer defined OPDEF macro names.
3. Programmer defined CPSYN and CPOP instructions.

If this search also fails an operation code error is noted.

When a CPSYN, OPSYN, CPOP, PPOP, MACRO, MACROE, or OPDEF definition takes place, COMPASS first searches the operation code table to see if the op code is already present. In the case of a MACRO, MACROE, OPSYN, or PPOP definition, the macro name is used in the search. In the case of an OPDEF, CPSYN, or CPOP definition, the entry for the search is a "descriptor" of the same format as the descriptions in the op code table of CP machine instructions and other OPDEF descriptions. Thus, if a MACRO, MACROE, OPSYN, or PPOP name matches any other name in the table, a duplicate definition flag is issued, and the new definition will take the place of

the old one. Likewise, if a CPSYN, CPOP, or OPDEF descriptor matches any other descriptor in the table, the same will occur. An OPDEF, CPSYN, or CPOP descriptor will not match any name in the table, so that an OPDEF, CPSYN, or CPOP cannot redefine a MACRO name, a PP machine instruction mnemonic, or a pseudo instruction name. Conversely, because a MACRC, MACROE, OPSYN, or PPOP name cannot match any of the OPDEF or CP mnemonic descriptors in the table, a MACRO, MACROE, OPSYN, or PPOP does not cause duplicate definition of any OPDEF-defined macro or CP mnemonic. For example, a MACRO definition, such as SB4, redefines the machine instruction SB4, because the entry SB4 in the op code table is found before COMPASS tries the syntactic analysis to find a CP mnemonic. In other words, the SB4 macro exists at the same time as SBx or SB.x CP instructions. However, if a MACRO is named SB4, it does not cause a duplicate definition of other SBn CP instructions (where n is not 4). A later OPDEF definitiCn that redefines all instructions of the form SBr+r, is a duplicate definition of all other SBi rj+rk instructions, and the duplicate definition flag is issued. Henceforth, if a SB4 instruction is encountered, the SB4 macro is expanded since it has not been redefined. If a SBi rj+rk (where i is not 4) instruction is encountered, however, the OPDEF definition is expanded since all instructions of the format SBi rj+rk have been redefined by the OPDEF.

A duplicate macro definition flag is produced when:

1. A MACRO, MACROE, OPSYN, or PPOP name is the same as:

   a. A previous MACRO, MACROE, OPSYN, or PPOP name (system or programmer defined).

   b. A PP machine instruction (if PP assembly).

   c. A pseudo instruction.

2. An OPDEF, CPSYN, or CPOP description is the same as:

   a. A CP instruction.

   b. A previous OPDEF, CPSYN, or CPOP description (system or programmer defined).

2.15 Hash Tables.

Hashing provides a very fast entry and lookup technique for the symbol table and operation code table. Since these operations are central to the assembly process, maximum speed is essential. The hashing method implemented in COMPASS is described below.

Initially, each table is allocated its base size of 2**k entries and cleared to zeros. In each two-word entry, a field (12 bits for the opcode table, 18 bits for the symbol table) is reserved to contain a hash link. Whenever a symbol or opcode name is to be entered or looked up, it is hashed: the name is treated as a binary integer and multiplied by some constant, and then a k-bit field is extracted from the product. This hash value, between 0 and 2**k-1, is used as an index into the base table. The hash link field of each entry points to the next entry having the same hash value; the last entry of each such chain has 0 in its hash link field.

As an example, consider the entry whose name is A. Its value, 2000 0000 0000 0000 0001 octal, is multiplied by the hashing constant, 2000 2525 0010 0100 1001 octal, giving the product xxxx 2525 0010 0100 1001 octal (exponent irrelevant). If the base table contains 256 (=2**8) entries, the high-order 8 bits of the product coefficient are extracted as an initial index, 125B. Thus, the chain begins at entry number 125B of the table, or words 252B and 253B. Assuming that this is the first entry with this hash value, the entry is stored into these two words of the table, with 0 in the hash link field.

Now, suppose an entry named D is being made. The multiplication will give xxxx 2524 0040 0400 4004 octal as the product. Extracting the 8-bit hash value again gives 125B as the initial index. When words 252B and 253B of the table are consulted, an entry is found, A, which is not the same as D. The actions performed now depend on which table is involved.

For the symbol table, two words are allocated at the end of the base table, the new entry is stored there with 0 in its hash link field, and the former last entry in the chain is made to point to the new entry. If this is the first collision, the new entry, D, is placed in words 2**k and 2**k+1 of the table, and a 1 is placed in the hash link field of entry A.

For the operation code table, however, the reverse is true. Two words are allocated at the end of the base table, but the cld entry (A) is moved there and the new entry (D) is placed in the base table entry. This is done only because in the opcode table, entries with the same name are permitted. It is desirable, in order to preserve the lookup order, to have the most recent entry available first. The search stops on the first match or at end of chain, whichever occurs first.

The advantages of the hash-table method are:

1. Extremely fast lookup. In effect, one is searching sequentially through only 1/(2**k) of the table for each entry.

2. Entry of a symbol into the table is quite simple.

The disadvantages of the hash-table method are:

1. If the table is ever sorted, the hashing must be removed.

2. A zero entry can never be made.

3. Base table positions for unused hash values are not used, causing some memory space wastage for small programs.

Examination of the other table handling methods (sequential, sequential then sorted for binary searches, ordered upon entry by merging) shows that all appear to demand the expenditure of more time for large programs than the hash method.

2.16 Source Input Formats.

COMPASS can read source input files in any of five internal formats. No control card parameter is needed to tell COMPASS which format to expect. Instead, COMPASS determines the format by inspection, and sets the control word CP.IFORM accordingly. The CP.IFORM values and corresponding formats are as follows:

CP.IFORM=0: Normal source input. Each card image is one to ten words in length, with display code characters packed ten per word. The last word of a card has two or more 00 characters in the low-order end of the word (the word may be all 00 characters); these trailing 00 characters comprise an end-of-card mark rather than data characters. For Record Manager I/O, COMPASS uses a GET macro for each card and expects the record length, in characters, in the FIT; COMPASS does not look for trailing 00 characters. Reading stops at end of section.

CP.IFORM=+1: MODIFY compressed compile file. This is the format of a compile file written by the symbolic program library maintenance program, MODIFY, when its A option is specified. The first word of the file is

7700 0016 0000 0000 0000

octal, and this word is followed by any number of compressed card images. The first word of each card image is a header word with the following format:

Bits 59-18: Deck or modification identifier in display code, left-justified with 00 fill.

Bits 17-00: Sequence number as a binary integer.

This card header word is followed by one to eight words containing a compressed card image. All non-blank characters are represented in display code. Strings of consecutive blanks are represented as follows:

55	one blank
00nn	nn+1 blanks, nn ≠ 00.

The end of card image is marked by 12 zero bits, and the last word is filled on the right with zero bits. For Record Manager I/O, COMPASS treats the entire input section as one SCOPE logical record (RT=S), and uses a GETP macro for each word of the file. Reading stops at end of record.

CP.IFORM=+2: UPDATE compressed compile file. This is the format of a compile file written by the symbolic program library maintenance program, UPDATE, when its X option is specified. The first word of the file is

7700 0000 0000 0000 0000

octal, and this word is followed by any number of compressed card images. Each card begins with two header words in the following format:

Word 1, bits 59-00: First ten characters of sequence field (i.e., card columns 74-83) in display code (column 73 is always blank).

Word 2, bits 59-18: Last seven characters of sequence field (columns 84-90) in display code.

Word 2, bits 17-00: Word count of compressed card image, as a binary integer.

These card header words are followed by the indicated number of words containing a compressed card image. All characters other than blank (55) and colon (00) are represented in display code. Strings of consecutive blanks are represented as follows:

55	one blank
5555	two blanks
00nn	nn+1 blanks, nn ≠ 00 or 01.

The code 0001 represents a single 00 character (colon). If the last word of the compressed card image is not full, it is filled on the right with zero bits. No 0000 terminator is needed, because the word count is specified in the second header word.

For Record Manager I/O on the 6000 series, COMPASS treats the entire input section as one SCOPE logical record (RT=S), and uses a GETP macro to read the N words of a card image & the two header words of the next card; reading stops at end of record. On the 7000, COMPASS uses one or two GETP macros per card, ignoring W record boundaries, and reading stops at end of section.

CP.IFORM=-2: MODIFY common deck. This format is expected when XTEXT statement specifies a common deck in a MODIFY program library file. The common deck consists of a SCOPE logical record containing a prefix table, a correction identifier table, and one or more compressed card images, some of which may be inactive. The prefix table begins with the word

77nn nnnn 0000 0000 0000

octal, which is followed by nnnnnnB words. The prefix table must be present but is ignored.

The correction identifier table begins with the word

7002 0000 0000 00nn nnnn

octal, which is followed by nnnnnnB words. The correction identifier table must be present but is ignored.

Each card image begins with one or more modification history words. The card is inactive and ignored if bit 59 of its first modification history word is 1. The last modification history word has zeros in bits 15-00, and is followed by a compressed card image in the same format as for CP.IFORM=+1 (see above), but without the header word. For Record Manager I/O, reading is done in the same way as for CP.IFORM=+1.

CP.IFORM=-1: UPDATE common deck. This format is expected when an XTEXT statement specifies a common deck in an UPDATE program library file. On the 6000 series, file must be a SCOPE random file, and the common deck is a SCOPE logical record containing one or more compressed card images, some of which may be inactive. The first active card must be a *COMDECK card, which must be present but is ignored. Each card image begins with one or more correction history words.

The card is inactive and ignored if bit 58 in its first correction history word is 1. The last correction history word has a 1 in bit 59, and is followed by a compressed card image in the same format as for CP.IFORM=+2 (see above), but without the header words. The number of words in the compressed card image is in bits 53-36 of the first correction history word. For Record Manager I/O on the 6000 series, COMPASS uses a GETP macro for each correction history word and another for the N-word card image; reading stops at the end of the SCOPE logical record. On the 7000, the file must be an unblocked sequential file with W records, each common deck comprising a section. COMPASS uses a GETP macro for each correction history word and one or two more for the N-word card image, ignoring W record boundaries and stopping at end of section.

The main source input file may have any of the first three formats described above (CP.IFORM = 0, +1, or +2); its format is determined by inspecting the first word of the file. When COMPASS is called by a compiler, the first word of the file is not available for inspection so COMPASS uses the value of CP.IFORM set by the calling compiler.

The file designated by an XTEXT statement may have any of the five formats described above, and need not be the same as the main source input file. If the XTEXT variable field is empty, the file is rewound and CP.IFORM is set to 0, +1, or +2 by inspecting the first word of the file; the file is then read until end of section or of SCOPE logical record, or until a COMPASS END statement is encountered.

On the 6000, if the XTEXT variable field is not empty, the file is OPENED to see whether it is a SCOPE random or sequential file and to load its index record if random. If SCOPE says the file is sequential, it must be a MODIFY program library file. The last SCOPE logical record is assumed to be an end of file (level 17), and the next-to-last record must be a program library directory. It begins with an optional prefix (77B) table, followed by a word containing

7000 xxxx xxxx xxxx xxxx

octal. Each subsequent two words comprise an index entry, with a record name in the first word and a mass storage record address in the second word. When the specified entry is found, CP.IFORM is set to -2 and the designated record is read until end of SCOPE logical record or a COMPASS END statement is encountered.

If the file is random, it may be an UPDATE program library file or a user random file. In the former case, the first word of the index is

7000 xxxx xxaa aaaa aaaa

octal, where aaaaaaaaaaB is the mass storage address of the record containing the deck list. The deck list is read and searched for the specified entry, CP.IFORM is set to -1, and the designated record is read until end of SCOPE logical record or a COMPASS END statement is encountered. The first card (*COMDECK) is skipped.

For a user random file, the first word of the index is

6xxx xxxx xxxx xxxx xxxx

octal, and each subsequent two words contain an index entry, with a record name in the first word (display code left justified with 00 fill) and a mass storage record address in the second word. When the specified entry is found, the designated record is read. CP.IFORM is set to 0, +1, or +2 by inspecting the first word, and then the record is read until end of SCOPE logical record or a COMPASS END statement is encountered.

On the 7000 without Record Manager I/O, processing is as described above. When the Record Manager is used, an XTEXT with a non-empty variable field is processed as follows. The file must be unblocked with type W records. The POSITION macro is used to set word addresses. If the last record of the file is at least three words in length and its first word contains 7LDIRECT$, the file is an UPDATE random program library. The second word of the last record contains the word address of the deck list section. The deck list is read and searched for the specified entry, CP.IFORM is set to -1, and the designated record is read until end of section or a COMPASS END statement is encountered. The first card (*COMDECK) is skipped. If the file is not an UPDATE random program library, it must be a record indexed file; see below.

For both 6000 and 7000 but only when a Record Manager is used for I/O, COMPASS can access a "Record Indexed" file. Such a file can be created by a FTN 4.0 program by the library subroutines OPENMS, WRITMS, CLOSMS with a name index, or by a COBOL 4.0 program declaring ORGANIZATION IS STANDARD, SYMBOLIC KEY IS data-name. On the 6000, such a file appears as a sequential file to SCOPE, but is accessed through the Record Manager as a word addressable file with C blocking and W records. On the 7000, it is a word addressable file, unblocked, with W records. The first record in the file, with its W control word (WCW) at word address 1, is one word in length and contains

7nnn nnnn naaa aaaa aaaa

octal, where aaaaaaaaaaaB is the word address of the WCW for the master index, and nnnnnnnnB is its length in words. The first word of the master index record must have its sign bit set, to indicate a name index, and the rest of the word is ignored. The rest of the index record contains two-word entries in which the first word contains the symbolic key, left justified with 00 fill, and the second word contains, in its rightmost 33 bits, the word address of the WCW for the designated record. COMPASS reads this record, determines the format and sets CP.IFORM accordingly, and then reads sequentially until end of section (or partition) or a COMPASS END statement is encountered.

2.17 Compiler Calls to COMPASS.

Any compiler may call the COMPASS assembler as a level {1,0} overlay from the operating system library. A communication area in the calling compiler's level {0,0} overlay contains all information passed between the compiler and COMPASS. This communication area is defined by a common deck, COMPCOM, contained in the COMPASS program library file. The control card sequence

```
REQUEST{PL2,MT}
UPDATE{P=PL2, N=COMPCOM,A}
RETURN{PL2}
```

in conjunction with the COMPASS statement

```
COMPCOM XTEXT COMPCOM
```

within the compiler, will cause the COMPASS interface definition to be included in the compiler. The following conditions must be established before the XTEXT statement.

For an absolute assembly, the origina dn location counters must be at 101B. For a relocatable assembly, the origin and location counters must be at the beginning of the first common block, if any, otherwise at the beginning of the main program. In any case, COMPASS expects the communication area to begin at absolute address RA + 101B.

The following symbols and micros must be defined.

BUFL micro - minimum CIO buffer length, noramlly ^1001B^; changed to ^0^ by COMPCOM if program is assembled to use 7RM for I/O.

CP.ABORT micro - indicates whether processor should abort after total source program errors. ^0^ = no, ^1559^ = yes.

CP.F= micro - complement of value of special symbol *F in COMPASS. ^0^ = COMPASS, ^-1^ = RUN 2.3, ^-2^ = RUN 3.0 or FTN, etc.

CP.LISTF micro - indicates whether processor should write source listing. ^0^ = no, ^1^ = yes.

CP.PAGE micro- indicates whether procesor should propagate listing page numbering across subprograms. ^0^ = no, ^1559^ = yes.

CP≡RM symbol - indicates whether processor is assembled to use a Record Manager for compiler-time I/O. 0 = no, 1 = yes. For the 6000, COMPCOM will include all of the 6RM routines for sequential I/O, obtaining them from the 6RM program library by XTEXT statements.

IBUF symbol - first word address of the CIO buffer for the source input file. Need not be previously defined.

LISTRM micro - indicates whether the I/O routines within COMPCOM are to be listed. null = yes, ▽-▽ = no.

MIN.FL symbol - minimum SCM field length needed by processor. Need not be previously defined.

MODEL micro - the CYBER-70 series model number on which the processor will run. ≠ MODEL ≠ < ▽75▽ means 6000 series, ≠ MODEL ≠ ≥ ▽75▽ means 7000 series.

OBUF symbol - first word address of the CIO buffer for the listing output file. Need not be previously defined.

OBUFL symbol - length of the CIO buffer for the listing output file. Need not be previously defined.

STOP symbol - address to which COMPASS jumps to return to the calling compiler. Need not be previously defined.

≡PLRM≡ micro - name of random program library file from which COMPCOM obtains the 6RM common decks when CP≡RM = 1 and ≠ MODEL ≠ < ▽75▽.

FET macro - needed only when CP≡RM =0, i.e., no Record Manager is used. COMPCOM uses this macro to define File Environment Tables for the course input, listing output, and binary output files. The macro call is: locsym FET lfn,buf, size . When CP≡RM = 1, COMPCOM defines its own FET macro.

Some of the above parameters describe the operating system environment. Examples of possible combinations are as follows:

Operating System	≠MODEL≠	CP≡RM
6000 SCOPE 3 with CIO	74	0
6000 SCOPE 3 with 6RM	74	1
6000 KRONOS	76	0
7000 SCOPE 1	76	0
7000 SCOPE 2	76	1

COMPCOM defines the communication area, presets all of the interface words, and defines the symbols and micros listed below.

BUFL micro - changed to "0" if CP≡RM = 1 and ≠MODEL≠ ≥ "75".

IBUFL symbol - length of the CIO buffer for the source input file, defined equal to ≠BUFL≠.

CP.NAME micro - name of the COMPASS level {1,0} overlay called by the compiler. At present, this is "COMP2⇳".

CP.OVLIB micro - name of the library containing the COMPASS overlay. Null means the global library set {established by LIBRARY control cards, if any} and the operating system nucleus library.

CP.ORG symbol - load address for the COMPASS level {1,0} overlay. This is the address at which the overlay header word is loaded. All information, including source input and listing output CIO buffers, to be preserved when COMPASS is called must be below this address. At present, the possible values of CP.ORG are as follows:

{a}	2777B	if	CP≡RM = 0, regardless of ≠MODEL≠.
{b}	7777B	if	CP≡RM = 1 and ≠MODEL≠ < "75".
{c}	07777B	if	CP≡RM = 1 and ≠MODEL≠ ≥ "75".

Case {a} provides room for CIO buffers in the caller's {0,0} overlay when no Record Manager is used.

Case {b} provides for CIO buffers and 6RM sequential I/O code modules in the {0,0} overlay.

Case {c} is small because the {0,0} overlay includes no Record Manager code nor CIO buffers when 7RM is used for compile-time I/O.

CP.BASE symbol - one less than the base address of the communication area. After relocation, if any, CP.BASE must be 100B.

CP.BATCH symbol - this word contains the complement of the value of the *F special symbol. Its sign tells COMPASS whether it was called by a compiler or by a COMPASS control card. The word is preset to ≠CP.F=≠, q.v., and should not be changed.

CP.ABORT symbol - this word contains a flag indicating whether the processor should abort at the end of a batch in which fatal source program errors were detected. It is preset to ≠CP.ABORT≠, q.v., and may be changed by the control card A option.

CP.ERRCT symbol - this word contains a flag in the sign bit and a cumulative fatal error count in the remaining 59 bits. Both are preset to zeros. The sign bit may be changed to 1 by the control card D option, indicating that binary output is to be written even when fatal errors are found in the source program.

CP.LISTF symbol - this word contains a flag indicating whether the processor is to write a source listing. It is preset to ≠CP.LISTF≠, q.v., and may be changed by the control card L option.

CP.PAGE symbol - this word contains a flag in the sign bit indicating whether the processor is to propagate listing page numbering across subprograms, and the remaining 59 bits contain the last page number as a binary integer. The word is preset to ≠CP.PAGE≠, q.v., and may be changed by the control card P option. If the sign bit is 1, pagination is even propagated across COMPASS calls, and COMPASS updates the page number before returning to the calling compiler.

CP.CPU symbol - this word contains the CYBER-70 series model number for the machine on which the processor is executing. It is preset to 2L≠MODEL≠.

CP.MAXFL symbol - this word contains the largest of the amounts of SCM used for all of the compilations and assemblies in the batch, i.e., the minimum SCM field length in which the batch could be performed, not including internal tables that can overflow to scratch files. It is preset to MIN.FL, q.v.

CP.XNAME symbol - this word contains the file name, left justified with 00 fill, to be used by COMPASS for an XTEXT statement whose variable field is blank. It is preset to 0L0LDPL and may be changed by a control card X or XT option.

CP.IFORM symbol - this word contains an integer representing the format of the source input file. It is preset to 0. The possible values and meanings are given in section 2.16 of this document.

CP.STOP symbol - this word contains a jump instruction to STOP. Each compiler that calls COMPASS must have a STOP routine in its level {0,0} overlay. This routine tests whether COMPASS encountered end of source input {see CP.CARD}. If so, STOP issues appropriate dayfile messages and terminates the run. Otherwise, STOP reloads the compiler's level {1,0} overlay and continues its processing.

CP.UNU1 symbol - this word is unused at present.

CP.LINE symbol - this is the first of four words that are preset to all blanks. It may be used as the left margin when source cards are printed directly from CP.CARD.

CP.CARD symbol - equal to CP.LINE=4. This is the first of 16 words that contain the next source card to be processed. When the compiler calls COMPASS, either {a} the first card image of the subprogram to be assembled {i.e., the IDENT card} begins in CP.CARD, or {b} word CP.CARD contains +0, in which case COMPASS will open the course input file, determine its format and set CP.IFORM accordingly, and read the first card. When COMPASS returns to its caller by jumping to CP.STOP, either {a} the first card image of the next non-COMPASS subprogram begins in CP.CARD, or {b} word CP.CARD contains +0, indicating that COMPASS encountered end of section or greater on the source input file, i.e., there is no more source input to process. In the latter case, COMPASS has already closed the source input, listing output, and binary output files.

CP.STEXT symbol - this word contains the number of system texts specified by control card S and G or GT options. It is preset to zero. The next 7 words contain system text overlay names in the following format:

```
VFD    42/0L ovlname, 17/0, 1/gflag
```

where gflag is 0 for a system text to be loaded from a library {S option} or 1 for a system text to be loaded from a non-library file {G or GT option}. In the latter case, ovlname may be null, in which case COMPASS loads the first system text overlay it finds on the file. The first of these 7 words is preset to

```
VFD    42/0LSYSTEXT, 18/0
```

since this is the default system text.

CP.LIB symbol - this word contains the value to be used as the number of system texts when word CP.STEXT contains zero. Word CP.LIB is preset to 1, and should be changed to 0 by the processor when it finds an S=0 option on the control card. The next 7 words contain the library or file names for the corresponding CP.STEXT words. Each name is left justified with 00 fill. For a CP.STEXT word with gflag=0, the corresponding CP.LIB word may be zero {null library name}, indicating the global library set.

CP.AFLS symbol - this word contains the actual SCM field length.

CP.NFLS symbol - this word contains the nominal SCM field length, i.e., the amount of SCM that COMPASS may use. It may be less than or equal to CP.AFLS. If less than, COMPASS will preserve the information from {CP.NFLS} to {CP.AFLS} whenever it changes the SCM field length.

CP.AFLL symbol - this word contains the actual ECS/LCM field length.

CP.NFLL symbol - this word contains the nominal ECS/LCM field length. Usage of these two words is similar to the preceding two words.

CP.MODL symbol - this word contains the value of the builtin MODLEVEL micro, left justified with 00 fill. It is present to zero {null} and may be changed by a control card ML options. If it is null, COMPASS uses ≠JDATE≠.

CP.PCOM symbol - this is the first of three words that contain the value of the builtin PCOMMENT micro, left justified with blank fill. It is preset to 30 blanks and may be changed by a COMPASS control card PC option. A compiler should store here whatever it writes in the ▽processor comment▽ field in the PRFX tables in its binary output.

CP.IFET symbol - first word of the File Environment Table for the source input file. The file name is preset to INPUT, and the buffer pointers to IBUF, IBUFL.

CP.IFIT symbol - first word of the File Information Table for the source input file. Defined only when CP≡RM = 1.

CP.OFET symbol - first word of the File Environment Table for the listing output file. The file name is preset to OUTPUT, and the buffer pointers to OBUF, OBUFL.

CP.OFIT symbol - first word of the File Information Table for the listing output table. Defined only when CP=RM = 1.

CP.BFET symbol - first word of the File Environment Table for the binary output file. The file name is preset to LGO, and the buffer pointers to OBUF, OBUFL. The processor must change the buffer pointers when it defines a CIO buffer area for this file.

CP.BFIT symbol - first word of the File Information Table for the binary output file. Defined only when CP=RM = 1.

CP.LCOM symbol - length of the communication area.

For all system environments except 7000 with Record Manager I/O, COMPCOM includes the following subroutines.

SYS=	process system request.
RCL=	place program on periodic recall.
WNB=	wait not busy (autorecall).
MSG=	send dayfile message.

These are the standard system communication routines from 6000 KRONOS 2.0 and 7000 SCOPE 1.1 common deck COMCSYS, except that MSG= has been modified for 6000 SCOPE 3.x.

For the 6000 with Record Manager I/O, COMPCOM also includes Record Manager routines with the following entry points.

CHEK.RM	CHECK processor - check previous operation.
CLSF.RM	CLOSE ... FILE processor.
ENDF.SQ	ENDFILE processor.
GETP.RM	GET partial record processor.
GET.RM	GET record processor.
PUTP.RM	PUT partial record processor.
PUT.RM	PUT record processor.
REW.RM	REWIND processor.
SKBL.RM	SKIPBL processor.
WEOR.SQ	WEOR processor - write end of SCOPE record or W section.

These routines can only handle sequential files with BT=C and RT=S, U, W, Z, or BT=I and RT=W.

Following COMPCOM, the compiler's level (0,0) overlay should contain the following.

1. Resident routines , including the STOP routine.
2. The listing output file CIO ubffer area OBUF.
3. The source input file CIO buffer area IBUF.
4. Initialization routines, including the control card processor, overlaying the I/O buffers.

The end of the input buffer must not extend past CP.ORG.

The {0,0} overlay does not have room for a CIO buffer for the binary output file. Each processor must allocate such a buffer in its higher overlay. Except for the 7000 with Record Manager I/O, this buffer must be flushed whenever it is overlaid.

COMPASS does not use any of the low core cells RA+0 through RA+100B, except that RA+1 is used for system communication and RA+64B through RA+67B are used when loading overlays.

2.18 Debugging Facilities.

A PATCH/SNAP debugging facility is included in COMPASS by assembling it with DEBUG = 1. During initialization (pass 0), the file named PATCHES is rewound and PATCH and SNAP directive cards are read until end of SCOPE logical record. Each card is copied to the file named SNAPPER.

PATCH directives provide for changing code in COMPASS without reassembling. A PATCH area is provided starting at ORGOVER. The PATCH directive card format is as follows:

Cclumns	1-6	- *PATCH
Columns	7-10	- Blanks
Columns	11-80	- Two octal numbers separated by a comma (blanks are ignored).

The first number is an address. The second number is the value to be stored at that address.

Example:

```
*PATCH    14106,51100 00173 73617 75001
```

SNAP directives provide for dumping registers and memory areas dynamically. Snapshots are formatted and written to the file named SNAPPER during COMPASS execution. After assembly, rewind SNAPPER and COPYCF to a print file. The SNAP directive card format is as follows:

Columns	1-5	- *SNAP
Columns	6-10	- Blanks
Columns	11-80	- One or more octal numbers separated by commas (blanks are ignored).

The first number is an address. The snapshot is taken whenever the first instruction at that address is about to be executed. The referenced word may contain any instructions. If it contains a return jump instruction (RJ), the word must not be followed by parameter words, and the called subroutine must not contain snapshots. COMPASS does not check for violation of these rules. The remaining numbers (if any) on the *SNAP card are pairs of values, each pair giving the first word address and word count of a memory area to be dumped. An asterisk after a number causes indirect addressing - the value is taken from bits 17-0 of the specified word at SNAP time.

Each snapshot is written as one SCOPE logical record containing a dump of the first 72 characters of the CARD area, a dump of all registers, and dumps of the specified memory areas. In the memory dumps, all-zero lines are omitted.

Example:

```
*SNAP          01774,216*,10
```

will dump eight words starting at the address in the word at location 216B.

Snapping saves and restores all registers except B1. If B1 ≠ 1, a jump out of bounds occurs. Otherwise, a 1 is regenerated.

3.0 CONTROL WORDS

Discussed here are the various assembler flags and control words as well as their function, use, and the routines or codes that modify and look at them. All are listed in alphabetic order. For those items that are more than one word in length, the item name is followed by a number in parentheses giving the item length in CM words.

ABASE - Base Character.

This word contains one right-justified character representing the current number base. This is preset to "D" at the beginning of each pass and is changed by the BASE pseudo-operation.

ABSFG - Absolute Flag.

This flag marks an absolute assembly. It contains either 0 (for relocatable) or 1 (for absolute). It is initialized to 0 and changed to 1 by PERIPH, PPU, and ABS. This flag is important: it indicates the following when it is 1.

1. The "zero" block is the absolute block rather than the second block.
2. When the blocks are relocated, all blocks are assigned absolute origins, and considered local blocks.
3. The operations EXT, REP, REPI, and ICC are illegal; BSSZ of 5 or greater does not cause a REPL entry.
4. The operations SEG and IDENT are legal.
5. Binary information is retained in MEMORY instead of being put out on an as-you-go basis.
6. The values of ORGBASE and SEGEPT have meaning.

ATIME - Assembly CPU Time.

This word is set, by a TIME request to the operating system, at the beginning of pass 1 and the end of pass 2, and the difference is printed in the ASSEMBLY TIME message at the end of each assembly listing.

B (13) - Binary Output FET or FIT.

When no Record Manager is used for I/O, B is equivalenced to CR.FET in COMPCOM. When a Record Manager is used, B is equivalenced to CP.BFIT in COMPCOM.

BADLOC - Bad Location Flag.

This word is set non-zero by SETUP when a location symbol of greater than 8 characters is encountered. It is examined by routines that use the location field for definitions. It does not apply for OPDEF, CPSYN, or CPOP names for which very long location fields are possible.

BASEMIC (Pass 1 only) - BASE Micro Value.

This word contains the character (normally D), left justified with 00 fill plus a character count of 1, that is the current value of the BASE built in micro. It is set and used by the BASE pseudo-op processor and is also used by TLUMIC when evaluating a reference to the BASE micro and the MICTAB lookup fails.

BASESTK - BASE Pushdown Stack

This area is a pushdown stack, with room for up to MSTACK entries, that is pushed by a BASE letter statement and pulled by a BASE * statement.

BBUF (BBUFL) - Binary Output Buffer.

This is the circular buffer area used in pass 0 for debugging directive input, in pass 1 for XTEXT input, and in passes 1 and 2 for binary output.

BINREC (18) (Pass 2 only) - Binary Record.

BINREC holds the relocatable binary text table currently being assembled. The first word holds the current origin address and word count, the second word holds the relocation bits, and the remaining words hold text. RESORG resets the first word to the origin and its relocation, and sets the word count to zero. DWORD adds words and relocations. When RESORG is called again, it dumps the table and starts again if the word count is non-zero.

The first word is organized as follows:

Bits 59-48: Zero during creation. However, when the actual image is dumped, 4000B or 3700B is added to bits 59-48.

Bits 47-36: Count of data words in the record. This is one less than the final word count produced.

Bits 35-34: Zero.

Bit 33: Conditional load flag, if ORGC is in effect.

Bits 32-00: Relocation and origin address.

If the origin address is less than 400000B, bits 59-48 will be 4000B (TEXT) and bits 32-00 are as follows:

Bits 32-27: Zero.

Bits 26-18: Relocation.

Bits 17-00: Origin address.

Otherwise, bits 59-48 will be 3700B (XTEXT) and bits 32-00 are as follows:

Bits 32-24: Relocation

Bits 23-00: Origin address.

BINREL (3) (Pass 2 only) - Binary Relocation.

This area contains the relocation indicators associated with BINWORD. BINOUT adds to it when it adds bytes of relocatable information into the current word. It is processed into actual relocation by DWORD. When switching USE blocks, the first word of BINREL is saved in the USETAB entry and the other two words are saved in the corresponding RELTAB entry. The first word of BINREL contains the number (0 to 4) of relocatable fields in the binary word thus far, and the other two words contain four 30-bit descriptors, numbered 0 to 3 from left to right. See RELTAB table description for details.

BINWORD (Pass 2 only) - Binary Word.

This word contains the partial binary word being assembled. It is OR'ed into by BINOUT when bytes of information are added to the current word, and is dumped into the output stream by DWORD.

USER saves BINWORD in the fourth word of a USETAB entry and restores it when the block is extracted for subsequent assembly. For a PP assembly, this word never contains more than 12 low-order bits of information.

BTEMP, BTEMPA, BTEMPB (Pass 2 only) - Scratch Storage.

These words are used as temporary storage by the binary output routines.

BUCKET - Beginning of Managed Table Area.

This is the address at which the managed table area nominally begins (and extends up to field length - 10). It follows all of the file buffers. However, these buffers are not necessarily all needed, so the managed table area may actually begin at a lower address.

C (8) - COMPASS Image FET.

This is the File Environment Table for the file in which COMPASS is saved to make room for the symbolic reference table (REFTAB) during pass 3. The file is used only when REFTAB is large enough to require this drastic action. The FET points to the buffer area CBUF.

CARD (71*NCARDS+30) - Statement Area.

CARD is a long area in which the columns of statements are expanded, one character per word, in the low-order 6 bits of each word. The length of this area is designed to accommodate the 71*NCARDS+1 words that can come from NCARDS cards, plus some overflow area used if EDIT increases the length of the statement with micro substitution or PDC replaces one-character formal parameter names with two-character codes of the form 77xx.

Non-blank data exists only out to LASTCOL. Blanks fill the remainder of the card buffer.

CBUF (approx. 15000B) - COMPASS Image.

This is the portion of COMPASS that is saved on a scratch file when necessary during pass 3. It consists of all the code from the end of the pass 3 processor (PRT) to the beginning of the REFTAB buffer area (RBUF).

CCOL - Comment Column.

This word contains the number of the column where a comment may start if a card contains no variable field. This word is set at the beginning of passes 1 and 2 to the default value (30). It is used by SETUP when the card is read, and changed by the COL pseudo-operation.

CCT - Card Count.

This word records the actual number of cards that comprise the current statement. It is set in pass 1 by RNS, EDIT, or UCARD, depending upon the source of the statement, and in pass 2 by RINTRD. It is critical for the listing process, for it records the number of lines which must be recreated for proper listings. It also dictates the number of words or pairs of words from SEQ which will be written on the intermediate file, depending on the source input file format.

CHAR - Character.

This word contains the current character as last returned by GETCH.

CLF - Conditional Load Flag.

This word is zero normally, but the sign bit is set whenever an ORGC pseudo-op is in effect. It is set by ORGC and is saved in the USETAB entry when blocks are switched. It is used by RESORG in writing binary TEXT tables, by DWORD in making COMTAB and LNKTAB entries that will later become binary FILL and LINK tables, and by BSSZ when it sets ORGBSSZ+1 for what will be written as TEXT and REPL tables.

CODEMIC (Pass 1 only) - CODE Micro Value.

This word contains the character (normally D), left justified with 00 fill plus a character count of 1, that is the current value of the CODE builtin micro. It is set and used by the CODE pseudo-op processor and is also used by TLUMIC when evaluating a reference to the CODE micro and the MICTAB lookup fails.

CODESTK - CODE Pushdown Stack.

This area is a pushdown stack, with room for up to MSTACK entries, that is pushed by a CODE letter statement and pulled by a CODE * statement.

COL (2) - Column Markers.

Two words, COL and COL+1, contain the column numbers of the blank columns immediately preceding the operation and variable fields, respectively. They are established in SETUP and examined in many places. They permit the GETCH subroutine to be positioned at the beginning of the field to be scanned.

COLUMN - Current Column Number.

This word contains the number of the column currently under examination. When GETCH is called, the following occurs:

COLUMN is updated by one.
CHAR and X1 and X6 are set to the contents of that card column located at CARD+COLUNN-1.

If COLUMN exceeds LASTCOL, however, it is not incremented and blanks are returned.

CP.ABORT - Abort Flag (in COMPCOM).

This word is normally zero and is changed to 1559 by the control card A option. COMPASS does an ABORT, NODUMP at the end of an assembly batch if any of the assemblies had a fatal error and CP.ABORT is non-zero.

CP.AFLL - Actual LCM Field Length (in COMPCOM).

This word contains the ECS/LCM field length that was in register X0 upon entry to COMPASS.

CP.AFLS - Actual SCM Field Length (in COMPCOM).

This word contains the CM/SCM field length that was in register A0 upon entry to COMPASS. If it is less than MIN.FL, COMPASS assumes the job is in REDUCE mode and arbitrarily does a MEMORY request to set the field length to 50000B.

CP.BATCH - Batch Mode (in COMPCOM).

The sign of this word tells COMPASS how it was called: positive= COMPASS control card, negative = compiler call. The absolute value is stored in FMODE and is the value of the *F special symbol. CP.BATCH is set by the F option on the COMPASS control card.

CP.BFET - Binary Output File Environment Table (in COMPCOM).

CP.BFIT - Binary Output File Information Table (in COMPCOM).

CP.CARD - Source Card Image (in COMPCOM).

This area always contains the card, read from the source input file by RNC, to be processed next by RNS. This look ahead technique is used because RNS must look at the next card to determine whether it is a continuation card for the current statement. When COMPASS is called by a compiler, CP.CARD contains the IDENT card image that caused the call, or +0 if COMPASS is to open the file and read the first card. When COMPASS returns control to the compiler, CP.CARD contains the non-IDENT card that was read when an IDENT card was expected, or +0 to signal the compiler that there is no more source input and COMPASS has closed all files. This communication technique eliminates any need for backspacing of the source input file.

The format of the card image in CP.CARD depends on the value of CP.IFORM. For normal source input (CP.IFORM=0), the card image is stored ten characters per word beginning in CP.CARD and ends with a zero byte. For a MODIFY compressed compile file (CP.IFORM= +1), CP.CARD is not used, the sequence word is in CP.CARD+1, and the compressed card image begins in CP.CARD+2 and ends with a zero byte. For an UPDATE compressed compile file (CP.IFORM=+2), the two header words (sequence field and word count) are in CP.CARD and CP.CARD+1, the compressed card image begins in CP.CARD+2 and ends with a zero byte, and the next two words contain the header words for the next card. For a MODIFY or UPDATE common deck (CP.IFORM=-2 or -1), CP.CARD contains the first correction history word, CP.CARD+1 contains the last correction history word if there is more than one, and the compressed card image begins in CP.CARD+2 and ends with a zero btye.

CP.CPU - Central Processor Model (in COMPCOM).

This word contains the CYBER 70/ model number of the CPU on which COMPASS is running, as two characters left justified with 00 fill. It is set by CTM and not used at present.

CP.ERRCT - Batch Error Count (in COMPCOM).

Bit 59 of this word is set to 1 by the control card D option; this causes writing of binary output even when fatal source program errors are detected. Bits 58-00 contain a cumulative error count as a binary integer. It is incremented by 1 for each fatal error detected in the source program and is not cleared at the start of each assembly. At the end of the batch, COMPASS aborts the jobs if CP.ERRCT is positive non-zero and the control card A option was used.

CP.IFET - Source Input File Environment Table (in COMPCOM).

CP.IFIT - Source Input File Information Table (in COMPCOM).

CP.IFORM - Source Input Format (in COMPCOM).

This word directs the activities of subroutines RNC and RNS. The possible values of CP.IFORM are:

+2	UPDATE compressed compile file.
+1	MODIFY compressed compile file.
+0	Normal source input.
-1	UPDATE common deck.
-2	MODIFY common deck.

CP.LIB (8) - System Text Library/File Names (in COMPCOM).

The first word of this area contains the number of system text overlays to be loaded when CP.STEXT is zero. CP.LIB is initially 1 and is set to 0 by the control card S=0 option. The remaining seven words contain the library names specified by control card S options, and file names specified by G options. They are stored in the order in which S and G options are found in a left-to-right scan of the control card. Each name is left justified with 00 fill. A zero word results from an S option with no library name.

CP.LINE - Print Line Image (in COMPCOM).

Not used by COMPASS. This area is provided in COMPCOM for use by compilers that write source listings in the first pass.

CP.LISTF - Long Listing Flag (in COMPCOM).

This word is preset to 1 and is set to 0 by the control card L=0 option. When this word is 0, COMPASS does not write a long listing, but may write an error listing unless it also is suppressed by the control card 0=0 option.

CP.MAXFL - Maximum Field Length (in COMPCOM).

This word contains the largest of the minimum field lengths for all of the assemblies in the batch, including Pass 0 processing. This value is used in the dayfile message SCM USED at the end of the batch. It is preset to MIN.FL and is updated by ATS.

CP.MODL - Value of MODLEVEL Micro (in COMPCOM).

This word contains the character string that is the value of the builtin micro MODLEVEL. This value is up to nine characters left justified with 00 fill, and is set by the control card ML option. The word is preset to zero. If there is no ML option, COMPASS uses ≠JDATE≠ as the value of MODLEVEL.

CP.NFLL - Nominal LCM Field Length (in COMPCOM).

This word contains the amount of ECS/LCM available for COMPASS to use. A calling compiler may set CP.NFLL less than or equal to the actual field length in CP.AFLL; if less than, COMPASS will preserve the information between the two addresses whenever it changes the ECS/LCM field length.

CP.NFLS - Nominal SCM Field Length (in COMPCOM).

This word contains the amount of CM/SCM available for COMPASS to use. A calling compiler may set CP.NFLS less than or equal to the actual field length in CP.AFLS; if less than, COMPASS will preserve the information between the two addresses whenever it changes the CM/SCM field length.

CP.OFET - Listing Output File Environment Table (in COMPCOM).

CP.OFIT - Listing Output File Information Table (in COMPCOM).

CP.PAGE - Page Number (in COMPCOM).

This word controls pagination for the long listing file. If bit 59 is a 1, listing page numbers start with 1 at the beginning of each assembly, and this word is not altered (the page number is kept in PGCNT instead). Otherwise, CP.PAGE contains the current long listing page number as a binary integer, and is not reset at the beginning of an assembly. CP.PAGE is prest to 1S59 and is changed to +0 by the control card P option.

CP.PCOM (3) - Value of PCOMMENT Micro (in COMPCOM).

This area contains the 30-character value of the builtin micro PCOMMENT. It is preset to 30 blanks, and can be set by the PC option on the COMPASS control card. When a compiler calls COMPASS, the compiler should store here the "processor comment" information that is normally writes in words 8-10 of each PRFX table in its binary output.

CP.STEXT (8) - System Text Overlay Names (in COMPCOM).

The first word of this area contains the number of system text overlays specified by control card S and G options. It is preset to 0. If it is still 0 after control card processing, CP.LIB is used instead. Each of the reamining 7 words after CP.STEXT contains a system text overlay name left justified with 00 fill, and a flag in the rightmost bit. If this flag is 0, the system text is loaded from the library whose name is in the corresponding word of CP.LIB; a null library name means the global library set and the system nucleus library. If this flag is 1, the system text is loaded from the file whose name is in the corresponding word of CP.LIB; a null overlay name means the first system text overlay on the file. The first word following CP.STEXT contains the default system text name, SYSTEXT.

The effects of the various control card S and G options are shown below. N represents the integer in CP.STEXT.

No S or G options:	N = 0, CP.STEXT+1 = 0LSYSTEXT+0, CP.LIB = 1, CP.LIB+1=0.
S = 0:	CP.LIB = 0.
S:	N = N+1, CP.STEXT+N = 0LSYSTEXT+0, CP.LIB+N=0
S = overlay:	N = N+1, CP.STEXT+N = 0L overlay+0, CP.LIB+N=0
S = library/overlay:	N = N+1, CP.STEXT+N = 0L overlay+0, CP.LIB+N = 0L library.
G = 0:	Ignored.
G:	N=N+1, CP.STEXT+N = 1, CP.LIB+N = 0LSYSTEXT.
G = file:	N=N+1, CP.STEXT+N=1, CP.LIB+N = 0L file.
G = file/overlay:	N=N+1, CP.STEXT+N = 0L overlay +1, CP.LIB+N = 0L file.

CP.XNAME - Default XTEXT File Name (in COMPCOM).

This word contains the file name, left justified with 00 fill, to be used by an XTEXT pseudo instruction in which the location field is blank. It is preset to 0LDPL and can be changed by the control card X option.

CT - Character Type

This word contains a shift count used by SCD when creating binary values for a character string. It is preset to 0 at the beginning of passes 1 and 2 and is reset by the CODE pseudo-operation. It takes on the following values at the current time:

0	Display code data.
6	External BCD data.
12	Internal BCD data.
18	ANSCII subset data.

CTYPE (Pass 2 only) - Control Card Type.

This word is set for the pseudo-operations TITLE, TTL, SPACE, and EJECT and controls the listing of such cards, in conjunction with the C list option.

D (8) - Debugging Output FET or FIT.

This is the File Environment/Information Table for the debugging output file. It points to the circular buffer area DBUF. The FET and FIT are omitted when COMPASS is assembled with DEBUG=0.

DATE - Current Date.

This word contains the current date in display code, and is set by a DATE request to the operating system at the beginning of each assembly. It is used in the title line as well as in the preset micro °DATE° and the prefix (77B) tables in binary output.

DBUF (DBUFL) - Debugging Output Buffer.

This is the circular buffer area for the debugging output file.

DETFLG (Pass 2 only) - Detail Flag.

This is the detail flag which, in conjunction with the D list option, controls detail listings. DETFLG is set by the LISTER routine after listing each line, and is cleared whenever a line is read in RINTER. In this way, the second and subsequent lines generated from one intermediate read are under the control of DETFLG and the D list option.

DI (2) - Default Symbol Index.

In pass 1, DI contains the base of the default symbol or symbol literal table (SLITS) relative to the table's origin. It is zero unless a second IDENT card has been processed, in which case it contains the length of SLITS at the time the index in SLITS and the last index+1 in SLITS for the current segment. These indexes are recorded in the segment table (SEGTAB). In relocatable assemblies they are always 0 and equal to the length of SLITS, respectively. They are used for defining default symbols in pass 1 for printing them in pass 2.

DKCNT (Pass 2 only) - Deck Count.

This word records the number of decks (SCOPE logical records or sections of binary output) produced from one assembly. It is cleared in pass 2 initialization, incremented by LCC or whenever DFIRST is called, and is used to determine the number of back-spaces to issue, when erasing binary output.

DKNAM (Pass 2 only) - Deck Name.

This word contains the name of the current deck. It is initially the same as IDNAM, and is the reset by each SEGMENT statement and each IDENT statement with a non-blank variable field. It is delivered in the calling sequence to the routine DFIRST that sets it. It is then used in the prefix (77B) table at the beginning of the deck.

DLFLG (Pass 2 only) - Deferred List Flag.

This word indicates that a macro line has been encountered and is ready to be printed. It is set to a 1 by ZMACALL in pass 2 and is tested by LDL to determine if a deferred line is to be listed.

DPBA (15) - Binary Prefix Table Buffer.

DPBA is a 15-word buffer that contains the PRFX table dumped as preliminary binary information by DFIRST.

E (8) - Error FET or FIT.

This is the File Environment/Information Table for the short listing file. It points to the circular buffer area EBUF if long and short listings are both requested and they are directed to different files; otherwise it points to OBUF.

EBUF (EBUFL) - Error Buffer.

This is the circular buffer area for the error output file. Its length is changed to zero when FET E is changed to point to OBUF instead of EBUF.

ECHFLG - Echo Flag.

This flag is included in IND written out on the intermediate file. ECHFLG is set to 1 by DUP and ECHO processing when the duplication definition is complete and the stack is pushed down to start the duplication. It is cleared when the end of the duplication is encountered and the stack is pushed up.

This flag is used in the STOPDUP operation to make sure that there is a DUP or ECHO in effect. Otherwise, the STOPDUP operation is ignored.

ECHFLG is used to control listings in pass 2 based upon list option E.

EDITFG - Edit Flag.

In pass 1, this word denotes that the current statement may have micros or concatenation marks in it. The indication is accurate when statements are derived from the source input file, because the occurrence of these characters has been tested. When internally generated statements are created (via UCARD), however, the flag is always set.

The flag is considered on if the sign bit is set. When this flag is positive, the subroutine EDIT does not examine the card. When it is negative, EDIT makes a character-by-character test of the line to determine if there are any editing marks in it.

This flag is a time saver only, enabling the EDIT routine to bypass a time-consuming scan in most cases.

EDITM (Pass 1 only) - Edit Mask.

This word contains ones in bit positions 60-MICMARK and 60-CONCAT, and zeros in all remaining bit positions. It is used by RNS to set EDITFG if the current source statement contains any editing characters.

EFLG - Error Flag.

This word contains 0 if all of the statement error flags (xERR and WnERR) are zeros; it is set to 1 whenever an error flag is set. It is tested by LISTER. If it is 0, there is no need to look at or clear the individual error flags.

EI (2) - Entry Point Index.

In pass 1, EI contains the base of the entry point table (EPTAB) relative to the table's origin. It is zero unless a second IDENT card has been processed, in which case it contains the length of EPTAB at the time the IDENT card occurred. In pass 2, EI and EI+1 contain the base index in EPTAB and the last index+1 in EPTAB for the current segment. These indexes are recorded in the segment table (SEGTAB). In relocatable assemblies they are always 0 and equal to the length of EPTAB, respectively. In absolute assemblies they are used to create the multiple entry point overlays.

ELCNT (Pass 2 only) - Error Line Count.

This word records the number of lines produced on a given page of the error list file. The LEL subroutine checks this word. If the addition of the newly requested line would cause ELCNT to exceed PAGESIZ, LHD produces a page eject code along with the titling.

ELEXT - Element External.

This word contains the external number of the address element scanned by EVITEM. If zero, the element is non-external. Note that not more than one of ELEXT, ELREG, and ELREL can be non-zero.

ELOP - Element Operator.

This word contains a code representing the operator that precedes the current address expression element, as follows:

1	none or + or - or ^
2	*
3	/

ELREG - Element Register.

This word is set non-zero by EVITEM when it scans a register designator in a central processor assembly. In this case,

the value of ELREG is an integer rrnB where rr is either 01, 02, or 03 for A, B, or X respectively, and n is the register number. If the address expression element is not a register designator in a central processor assembly, EVITEM returns zero in ELREG.

ELREL - Element Relocation.

This word contains the relocation for the address element scanned by EVITEM. The representation is the same as in EXREL, q.v.

ELVAL - Element Value.

This word contains the value of the address element scanned by EVITEM, with sign extension through all 60 bits. It is +0 if the element is an undefined symbol, and is -0 if the element is empty or is a register designator in a central processor assembly.

ENDP - Value of END Statement Location Symbol.

This word is set during pass 1 END statement processing. Its value is the sum of the lengths of all of the local USE blocks in the assembly. It is used in the pass 2 END statement processor as the location address for listing the statement, and in DFIRST as the program length in the first word of the binary PIDL table in a relocatable assembly.

ENDSEQ (2*NCARDS) - END Statement Sequence Fields.

When the END statement is first encountered in pass 1, its sequence fields (from SEQ) are saved in ENDSEQ, the END statement is added to the remote text definition table (RMTAB), and a HERE statement is generated. When the END statement is encountered again, its sequence fields (which now contain *RMT*.1) are restored to SEQ from ENDSEQ.

EOFINP - End of Input Flag.

This word normally contains zero. It is set non-zero bv RNC when it encounters an end of data on the file it is reading (source or XTEXT input file RNC also generates the card image

//////// END CARD MISSING.

so that COMPASS processing can terminate more or less gracefully. After an assembly is completed, COMPASS terminates the run, or returns to the calling compiler, if EOFINP is non-zero.

EPCNT (Pass 2 only) - Error Page Count.

EPCNT records the current error page number. It is converted to display code for inclusion in the title line by LHD. It is initialized at the beginning of pass 2 and is incremented for each page eject on the error list file.

ERCNT - Assembly Error Count.

This word is updated in pass 2 by the listing routine to reflect the total number of fatal errors. If it is non-zero:

1. The binary output is suppressed at many places so that a shorter backspace is possible at the end.
2. The binary output is erased at the end of the assembly.
3. A dayfile message is produced with the error count.
4. The contents of ERCNT is added to CP.ERRCT to form a cumulative error count.

ERDIR (5*LEFLG) (Pass 2 only) - Error Messages.

This table parallels ERFLAGS and contains the text of the error messages. Each message is 5 words in length.

ERFLAGS (LERFLAGS) - Statement Error Flags.

This table contains one word for each error flag for the current statement. Each word contains 0 if the error was not detected or 1 if it was. The words are named xERR for fatal errors and WnERR for warning errors. The last word is EFLG, q.v. All of these flags are included in IND written on the intermediate file. They are cleared by INPUT1 and LISTER.

ERRLETS (LEFLG) (Pass 2 only) - Error Flag Letters.

This table parallels ERFLAGS and contains one word for each error flag. Each word contains the character, in display code right justified with binary zero fill, to be printed in the left margin of the listing line when the corresponding word in ERFLAGS contains a 1.

EXERR - Expression Error Flag.

This flag is set to 0 for each statement by INPUT1 and LISTER, and is set to 1 by SCAD, SCD, EVITEM, and TLUSYM whenever any error is detected. When EXERR is 1, SCAD stores +0 into EXREG and EXVAL, and SCD returns with an all-zero value, one word in length.

EXEXT - Expression External.

EXEXT follows EXVAL and contains the external number of the address expression. If zero, the expression is non-external. Note that EXREL and EXEXT cannot both be non-zero.

EXLGN - Expression Field Length.

This word contains the length, in bits, of the field that will contain the value of the expression being scanned by SCAD.

EXREG - Expression Register.

This word contains a record of the register values encountered during an address scan. To be valid, any expression that contains a register must be in a machine operation which was extensively checked for syntax in pass 1. Thus, no checking is performed by the address scan routine for valid register usage.

If no registers appear in an expression, EXREG is zero. If one register appears in an expression, it is located in the lower 9 bits of EXREG, exactly as EVITEM returned it: rrnB,

where rr is either 01, 02 or 03 for A, B, or X respectively, and n is the register number. If two registers appear, the first is in bits 17-9 and the second in bits 8-0.

Note that since the register classes determine the operation code in pass 1, and the register values are all that are needed in pass 2, the class is irrelevant in pass 2.

EXREL - Expression Relocation.

This word, which follows EXVAL, contains the relocation of an address field. If the subfield evaluated by SCAD is absolute, EXREL is zero. Otherwise, it contains the 9-bit relocation encoded in a pass-dependent manner as follows:

Pass 1:	000 - absolute
	002 - positive relative to the second block
	402 - negative relative to the second block
Pass 2:	000 - absolute
	001 - positive program
	401 - negative program
	002 - positive relative to the first common block
	etc.

EXSTOP - Expression Terminator.

This word contains a value (0 for blank or 1 for comma) indicating the character that stopped scanning of an expression by SCAD. It is used by many routines to determine whether another expression follows.

EXVAL - Expression Value.

SCAD sets EXVAL to the expression value after an address scan. At other times, it is used as a scratch cell.

FLAG - General 60-bit word carried on Intermediate.

This word is carried on the intermediate file for each statement. Since the word is normally not used, it is zeroed at the beginning of each statement (INPUT1) and is written explicitly on the intermediate file only if it is non-zero when WINTER is called.

For USE and ORG, the FLAG word is set up:

Bits 59-24: Old block number.
Bits 23-00: New block number.

The operations SEGMENT and END use the word as follows:

Bits 59-24: Old block number (or current one).
Bits 23-00: A "1".

The setting occurs in USER for USE and ORG, and in YFUALL for SEGMENT and END.

FLIST (Pass 2 only) - Force List Flag.

When FLIST is non-zero, the current line is listed by LISTER regardless of other tests. It is set by LISTL, and LISTER clears it when done. This forces the listing of header and trailer lines produced by COMPASS.

FMODE - FORTRAN Mode.

This word contains a binary integer indicating which subsystem called for a COMPASS assembly. For a COMPASS control card call, FMODE is set by the F parameter. When COMPASS is called by a compiler, the twos complement of {CP.BATCH} is stored in FMODE. It can be tested by a program by use of the special symbol *F. The following will be returned as the value of this symbol:

Program called from	Value of *F
COMPASS	0
RUN	1
FTN	2

FNAME (NFNAME) - F Parameter Names.

This is a table of names that may be used in the F parameter on the CCMPASS control card, and is used by SFV to convert a name to the numeric value to be stored into FMODE.

G {8} - System Text FET or FIT.

This is the File Environment/Information Table used in pass 0 for reading a system text overlay from the file specified by the G parameter on the COMPASS control card. It points to the circular buffer area SBUF.

HASH - Hash Constant.

This word contains the floating-point value by which a symbol or operation code name is multiplied to form its hash value. The bit pattern is designed to randomize the name and give extra weight to the first two characters. It will usually produce different hash values for names that differ by only one character.

HTYPE - Hardware Dependencies.

This word contains zeroes in bits 59-54, and hardware dependency characters, if any, from the MACHINE pseudo instruction in bits 53-00, left justified with blank fill. This word, with the deck type ORed into bits 59-54, is stored into word 7 of each PRFX table in the binary output.

I {13} - Input FET or FIT.

When no Record Manager is used for I/O, I is equivalenced to CP.IFET in COMPCOM. When a Record Manager is used, I is equivalenced to CP.IFIT in COMPCOM.

IBUF {IBUFL} - Input Buffer.

This is the circular buffer area for the source input file. If the long listing is suppressed {L=0 on COMPASS control card}, the length of IBUF is increased by all but 101B words of the output buffer OBUF.

IDNAM - Ident Name.

This word contains the name extracted from the IDENT card. It is used to establish the first deck name, and for the error comment in the case of assembly errors. If no IDENT card appears, the name is set to ****** or to ******** for peripheral and central routines respectively.

IFCDGP - First Card Group Flag.

This word controls the first card group. When it is zero, the group 5 pseudo-operations ABS, PERIPH, PPU, MACHINE, and STEXT are permitted. It is set non-zero by the first occurrence of a machine operation or any group 2 pseudo-operation. When ABS, PERIPH, PPU, MACHINE, or STEXT is encountered and the cell is non-zero, the assembler notes operation code error. It is processed in each pass.

IFCNT (Pass 1 only) - IF Count.

This word records the number of statements remaining to be skipped due to IF-skipping. It is set in IFXXN0 to contain either the actual number of statements to be skipped, or -1 if the skip is controlled by an ENDIF only. It is examined in pass 1 master control and updated to reflect skipping. While this word is non-zero, only ENDIF, ELSE, and END are recognized. In addition, the ENDIF processor ignores the statement if IFCNT is positive. While this word is non-zero, the flag cell N0AS is set to 1 by WINTER to signify an IF-skipped card. Listing of such statements in pass 2 is controlled by the F list option.

IFNAME (Pass 1 only) - IF-skipping Bracket Name.

This word contains the bracket name as extracted from an IF-type or ELSE statement which initiates skipping. It is checked by ENDIF and ELSE processing to determine that the skipping bracket name is encountered.

IND - Indicator Word.

This word is a temporary storage word for the flags and indicators used during the reading and writing of the intermediate file. Included in this category are all flags appearing in the IND word of the intermediate file.

Flags (described elsewhere):

NOAS	No-assembly Flag (IF-skipping)
TXTFLG	Text Flag (definition card)
MICFLG	Micro Substitution Flag
SYSFLG	System Macro Flag
MACFLG	Programmer Macro Flag
ECHFLG	Echo (duplication) Flag
RMTFLG	Remote Assembly Flag
LIBFLG	XTEXT Flag

And also the error flags (see ERFLAGS).

INTERIO - Intermediate I/O Flag.

This word is initially 0. When non-zero, it flags that the intermediate is on the file ZZZZZRL instead of in memory. It is set by the table manager routine when the intermediate is dumped. WINTER and RINTER examine it to determine the destination or source of the intermediate information.

INVENT - Invented Local Symbols.

This word is initialized at the start of pass 1 to contain the symbol 000000 in display code, right justified with binary zero fill. As each invented symbol is needed, the word is updated. An 8-character symbol is used so that it can be detected in pass 2 and the system can delete references and listings of the symbol in the reference table.

IOP - Operation Code Field Entry.

Contains the right-adjusted operation code field entry. It is set in the routine SETUP. If the operation code field is missing this word contains 5555B.

KADFLAG - Address Term Flag.

This word is set to +0 upon entry to SCAD and is changed to 0 when a non-register element is found in the epxression. It is used at the end of SCAD to set EXVAL to +0 if the expression is vacuous or contains only register designators, and to the actual expression value (possibly -0) otherwise.

LASTCOL - Last Column Number of Statement.

This word contains the column number of the last relevant column of an input statement. It is set by RNS, UCARD, RINT, and EDIT. If set by RNS, it is 72 for a 1-card statement, 143 for a 2-card statement, etc. When set by the other routines, it reflects the number of characters in a statement after parameter substitution, concatenating, and micro substitution have been performed.

LASTCOL is used by the character scanning routines to determine when the end of a card is encountered. GETCH tests it to determine when to force blanks. SQUEEZE uses it to determine the limit of the card to be packed. Routines use it to determine how much of the prior statement must be cleared out. This saves time since all 71*NCARDS+30 words in CARD need not be cleared out. Routines that scan deliminted character strings use it to store a copy of the delimiter just past the end of the statement in case the source statement does not contain a terminating delimiter.

LCCT (Pass 2 only) - List Card Count.

LCCT dictates the number of lines listed for a statement. It is set by RINTER based upon the card count recorded in the OPTYPE word for each intermediate record. Internally generated lines occur, in which case LCCT controls the listings.

LCM - Large Core Memory Length.

This word is used by RUT to keep the current value of the amount of LCM used. It is used to allocate local LCM blocks.

LCMEND - Next Available LCM Address.

This word contains the first word address of unallocated ECS/LCM space. It is initially 200B, since the first 100B words contain zeros for rapid clearing of SCM areas, and the next 100B words are reserved for use in fast moving of data within SCM. Pass 0 moves each of several tables to ECS/LCM starting at the address in LCMEND and then updates LCMEND, provided the difference between CP.NFLL and LCMEND is not less than the size of the table.

LCMMAC - System Macro Table Pointer

If the job's ECS/LCM field length is sufficient, Pass 0 stores the system macro table there and sets LCMMAC to point to it:

Bits 59-30 Word Count.
Bits 29-00 First word address in ECS/LCM.

Otherwise, LCMMAC is zero. When it is non-zero, Pass 1 initialization reloads the MACDEF table from ECS/LCM, and the Pass 1 END processor destroys the SCM MACDEF table.

LCMMIC - System Micro Table Pointer.

This word is similar to LCMMAC, but for the system micro table, SYSMIC.

LCMOPC - Opcode Table Pointer.

This word is similar to LCMMAC, but for the opcode table, OPTAB.

LCMSYM - System Symbol Table Pointer.

This word is similar to LCMMAC, but for the system symbol table, SSYMS.

LI (2) - Literal Table Index.

In pass 1, LI contains the base of the literal table (LITAB) relative to the table's origin. It is zero unless a second IDENT card has been processed, in which case it contains the length of LITAB at the time the IDENT card occured. In pass 2, LI and LI+1 contain the base index in LITAB and the last index+1 in LITAB for the current segment. These indexes are recorded in the segment table (SEGTAB) and are used in pass 2 to determine the extent of the literals to dump to binary for each segment, as well as to limit the area of LITAB that is searched for a literal match. In relocatable assemblies LI and LI+1 are always 0 and equal to the length of LITAB, respectively.

LIBFLG - XTEXT Source Flag.

This flag is included in IND on the intermediate file. LIBFLG is set by CTEXT or by XTEXT processing when the stack has been pushed down to start unpacking statements from LASTAB. Thus, this flag indicates statements derived from CTEXT or XTEXT sources. It is used in pass 2 to control listings based upon list option X.

LINE (9*CARDS) (Pass 2 only) - Line Image.

This area contains the re-creation of card images for a statement. It is created in RINTER where the card is unpacked and the sequence fields are re-combined with the card image. Whenever LISTER is called, the line area up to LLINE is blank filled. The amount of LINE to list is dictated by LCCT, listing card count.

LISTOPS (28) - Listing Options.

This is a table of paired words. The first word contains the name and nominal value of each list option, as follows:

Word 1	Bits 59-54	20B.
	Bits 53-48	The list option letter in display code.
	Bits 47-00	The nominal value; 0 if off, 1 if on.
Word 2	Bits 59-00	The current value; 0 if off, 1 if on.

In this table, the second word is set from the first word during initialization of each pass.

This table reflects the current status of each list option. and has a name "Lx", where 'x' is the list option. The options currently in this list are:

A, B, C, D, E, F, G, L, M, N, R, S, T, X.

The actual list processing is controlled by a table in LISTER which checks the option and the condition it controls.

LISTSTK - LIST Pushdown Stack.

This area is a pushdown stack, with room for up to MSTACK entries, that is pushed by a LIST options statement and pulled by a LIST= statement.

LLB - Local LCM Block Relocation.

This word contains, in bits 32-24, the loader relocation code for the local LCM block (if any) of the program being assembled. This is the concatenation of the local LCM blocks specified in the source program by USELCM statements with no slashes around the block names. LLB is set by RUT and used by PBN, PEP, /PASS2/REP, DFIRST, and PRT.

LLINE (Pass 2 only) - Last Print Line Used.

This word contains the address+1 of the last word used in the LINE buffer. It is set by CPL and other routines that set up print lines. LISTER uses it to determine the area of LINE that must be cleared after listing a line. It has no logical function other than to save assembly time in clearing an area that is already clear.

LOCCTR (2) - Location Counter.

Like ORGCTR, this two-word area contains the current value of the location counter. This word thus contains the current value of * or *L. It is used to define location symbols. It is updated in parallel to ORGCTR, and is set whenever ORGCTR is set. It may be set independently only by a LOC pseudo instruction.

This value is also used as the reference address for symbol references, and for the value listed on the listings in the assembler output.

LOCORE - Low Core Address of Managed Table Area.

This word contains the base address of the managed table area. It is set upon loading to the end of the assembler and is adjusted by the routine ACL. The managed tables always lie between LOCORE and LOCORE+SIZCORE.

LOCSYM - Location Symbol.

Contains the right-justified location field entry of a statement. It is set up in the routine SETUP. It is examined extensively in many routines (those that look at the location symbol). It may contain invalid location symbols, even those that are longer than 8 characters. Thus, the location BADLOC is reserved to mark long symbols.

LOSTREF (Pass 2 only) - Lost References.

LOSTREF records the number of reference accumulations that had to be discarded due to a table area conflict. The word is zeroed in pass 2 initialization, incremented in end processing and if non-zero, written in the title line of the cross reference table as well as output to the dayfile.

LPCNT (Pass 2 only) - Line Count.

This word records the number of lines produced on the current page of the long listing. The LISTER subroutine LHD checks this word. If the addition of the current statement, (LCCT) lines, would cause LPCNT to exceed PAGESIZ, LHD ejects the page and produces titling. A page eject can be forced by setting LPCNT to some quantity greater than PAGESIZ.

LPGM - Length of Program.

LPGM is set from SEGTAB by SBL at the beginning of each segment. Program length is derived by summing all local block lengths after all default symbols have been processed in the case where SEG or SEGMENT have not been used. It is derived from the value of the origin counter if SEG or SEGMENT is used. It is used in pass 2 for the production of the binary PIDL table, as well as for MEMORY allocation for an absolute program.

LSYSMAC - Length of System Macro Definitions.

At the conclusion of system macro definition in pass 0, the length of the macro definition table (MACDEF) is saved in LSYSMAC. It is used at the end of pass 1 to reset the length of MACDEF to discard all programmer definitions but retain the system definitions. A value of 0 is possible.

LWORD - Length of Word.

This cell is initialized to 60 and is changed to 12 by PERIPH and PPU. It is used to establish the upper value for POSCTR, and in SCD to position character data values.

MACFLG - Programmer Macro Flag.

This flag is included in IND on the intermediate file. MACFLG is set in MACALL when a programmer macro is called. It gets cleared by macro pushups. A programmer macro is detected because the type-field in OPTYPE for the entry in the operation code table contains a 7 instead of a 6 which marks system macros. MACFLG is used in pass 2 to control listings based on the M list option.

MACHINE - Type of Assembly.

This cell contains 0 during a central processor assembly or 1 during a peripheral processor assembly. It is initialized to 0 and is changed to 1 by PERIPH and PPU. Its value must be either 0 or 1 because calculations are often performed on it. For example, the octal output for a DATA statement has its rightmost digit in print position 36-11*MACHINE.

MAXCORE - Maximum Amount of Storage Used.

This word records the amount of storage used by the tables. It is incremented in the manager routine whenever the total

amount of used storage has increased. The size of INTER (the intermediate file) and REFTAB (the cross-reference table) are not counted since they can overflow to disk. The word is output at the end of assembly to produce the STORAGE USED figure. It is reinitialized at this time so that the figure is independent for each assembly.

MAXORG (Pass 2 only) - Maximum Origin Within Current Block.

This word contains the maximum origin within the block under which assembly is currently proceeding. It is set by USER when a new block is started. Pass 2 initialization also sets it. MAXORG is set according to the current block as follows:

Blank common block: zero.
Other common block: length of block.
Local program block: last word address + 1 of block.
Any LCM block in an absolute assembly = Zero.

It is used in the binary output routines to check the origin values for output, as well as in BSSZ processing. When BINOUT or BSSZ detects a value or ORGCTR which is greater than or equal to MAXORG, it suppresses the output and notes an R-error.

MBASE - Nominal Mixed Base.

This word normally contains a 10, defining the number base of numeric data items used in card counts and bit counts to be decimal. However the BASE pseudo-operation modifies this to be either 8 for O or 10 for D or M. It is initialized to 10 at the beginning of each pass. It is used by SCD for interpreting a number that has no B, D, or O as a radix modifier, when it occurs in a context in which the assumed radix is 10 when BASE M is in effect.

MICFLG - Micro Flag.

This flag is included in IND on the intermediate file. MICFLG is set by the R= pseudo and by EDIT whenever it encounters a statement in which there are micro marks or concatenation symbols. The R= operation is designed so that the original line is written on the intermediate with MICFLG = 1. Then, MICFLG is cleared, and the R= pseudo goes to normal CP op processing. In EDIT the micros and concatenation marks are removed (or substituted) and EDIT exits for normal assembly. Thus, the intermediate record for this statement consists of two records, the first with the

R= before substitution or with the editing marks still present and MICFLG = 1, and the second edited to the actual statement the assembler should assemble, with MICFLG = 0. Pass 1 processing is aware of this process only in R= and EDIT where the first of the double writes are performed.

During pass 2, when RINTER encounters a statement for which MICFLG is set, one of two operations follows, depending upon list option A. When the A list option is off, the R= line or the line with the editing marks is formatted for listings and the second record, containing the actual assembly line, is set into CARD. Thus, the line as assembled differs from the line listed. One line of listing results.

If the A option is on, however, the first line with the editing marks is force-listed. This line has no information on it other than the source line. Then the next record is read in and CARD and the print line are re-created. Thus, the print line shows the substituted images.

MINORG (Pass 2 only) - Minimum Origin for Binary Output.

This word contains the minimum origin within the block under which assembly is currently proceeding, thus the origin of the block. It is set by USER when a new block is started. Pass 2 initialization also sets it. It is used in the binary output routines to check the origin values for output as well as in BSSZ processing. When BINOUT or BSSZ detects that ORGCTR is less then MINORG, it suppresses the output, and notes an R-error.

MTYPE - Object Processor Model Type.

This word indicates the type of processor that will execute the object program, as follows:

MTYPE=0	Any
MTYPE=1	6000 Series or CYBER 70/ Model 72,73,74
MTYPE=2	7000 Series or CYBER 70/ Model 76.

MTYPE is set to 0 by Pass 1 initialization and may be set non-zero in Pass 1 by first card group instructions as follows:

```
MACHINE 6 sets MTYPE = 1
MACHINE 7 sets MTYPE = 2.
PERIPH sets MTYPE = 1    }   unless MTYPE already set
PPU    sets MTYPE = 2    }   non-zero by MACHINE
```

The machine instruction processing routines set the opcode error flag (OI) if MTYPE ≠0, the opcode table entry is restricted to one machine type, and they do not agree.

NBASE - Nominal Number Base.

This word normally contains 10, defining the number base of numeric data items to be decimal. However, the BASE pseudo-operation modifies this to either 8 for O or M, or 10 for D. It is initialized to 10 at the beginning of each pass. It is used by SCD for interpreting a number that has no B, D or O as radix modifier, when it occurs in a context in which the assumed radix is 8 when BASE M is in effect.

NBLOCKS - Number of Common Blocks.

This word records the number of common blocks. It is set by RUT, and is used in DFIRST to determine the number of words in the binary PIDL table.

NCHARS - Number of Characters per Word.

This word contains either 10 or 2, depending upon the assembly. It is used as a divisor in DIS and SCD. It is initialized to 10 and changed to 2 by PERIPH and PPU.

NFOUP - Force Next Upper.

This word contains the force upper flag. It is initially off, but is set to 1 in the following cases:

1. A central processor operation of the type PS, RE, WE, XJ is encountered which requires a force upper.

2. After assembling a line which forces next upper, such as JP, PS, EQ (unconditional), ZR (unconditional), RJ, XJ, ES, and MJ.

When a force upper is performed, the cell is cleared. It is saved and restored in the USE processing. The permissible values are 0 and 1. It is looked at in the routine PRLOC.

NOAS - No Assembly Flag.

This flag is included in IND on the intermediate file. It is set to 1 by WINTER in pass 1 whenever IFCNT is non-zero, indicating that IF-skipping is in progress. Thus, pass 2 need not re-evaluate IF-type statements, but merely examines the NOAS flag. Listing of such lines in pass 2 is controlled by the F list option. For the sake of speed, WINTER does not write a statement to the intermediate file when NOAS = 1 and the F list option is off, since such statements are completely ignored by pass 2 anyway.

NLFLG (Pass 2 only) - No List Flag.

This word is set to 1 if RINTER determines that the card should not list. This is one of the flags that LISTER examines to determine whether to list a particular card.

NOLFG - No Label Flag.

This word controls the suppression of the 77B IDENT table (NOLFG ≠ 0), as well as the suppression of the PP header word, the CP 50B overlay header word and the CP 51B multiple entry point overlay table (NOLFG = 1). It is cleared at the beginning of pass 1 and is set by the NOLABEL pseudo-operation.

0 (13) - Output FET or FIT.

When no Record Manager is used for I/O, 0 is equivalenced to CP.OFET in COMPCOM. When a Record Manager is used, 0 is equivalenced to CP.OFIT in COMPCOM.

OBUF (approx. 1500B) - Output Buffer.

This is the circular buffer area for the long listing file. If the long listing is suppressed (L=0 on COMPASS control card), all but 101B words of OBUF is added to the input buffer IBUF, and the FET for the short listing file (E) is changed to point to the remaining 101B words of OBUF rather than EBUF.

OCTAL (40) (Pass 2 only) - Octal Print Area.

OCTAL contains the first 40 columns of the print image. Whenever LISTER is called, these 40 words are packed into 4 words and listed. The area is cleared after each print line.

This method was selected since the octal output of the assembler is produced column-by-column, by such routines as PACKO etc. Thus, it is easier to generate it in separate words, and then pack the words when listings are produced.

OPADS (7) - Operation Decomposition.

This is a set of temporary storage cells used in analyzing machine instruction syntax.

OPTYPE - Operation Code Equivalent.

This word most frequently contains the result of the last call to TLUOP during pass 1, and a value extracted from the intermediate file during pass 2. The format of the word is described in the chapter on assembler tables.

This word is artificially set by central operation code scanning in OPSYN, PURGDEF, PURGMAC and also by macro expansions (MACALL) to signal pass 2 processing that a macro call was encountered, and to pass along a flag as to the type of call (whether or not the location field needs processing).

OPVAL (Pass 2 only) - Operation Code Value

OPVAL is a general sort of word that is usually used to record the value of the operation code. This usage is

followed for machine operation codes and VFD processing. However, the binary output routines and list routines rely upon values supplied in the calling sequences rather than values in OPVAL.

ORGBASE - Base of Absolute Output.

This word is used only for absolute binary output. It contains the base address of the particular overlay being constructed, hence the bias address for MEMORY. It is used in the binary output routines as an additional low-origin check address. It is set from the second parameter on the IDENT card, and from the first parameter on the SEGMENT card. It also dictates in conjunction with LPGM the amount of output to dump when absolute output is created from MEMORY.

ORGBSSZ (2) (Pass 2 only) - Origin of BSSZ.

This pair of words records the last known origin of BSSZ coding, in the same format as ORGCTR, except that the sign bit of the second word is set if ORGC (conditional loading) is in effect. The number thereof is recorded in CNTBSSZ. When BSSZ code is encountered, a check is made to see whether the new BSSZ falls in sequence with the old. If so, the CNTBSSZ cell is incremented and the actual output is deferred until there is a change in sequence or the DBSSZ routine is called.

This method is used so that successive BSSZ statements can be generated as a single REPL entry employing the rule for relocatable output that if there are less than 5 BSSZ words, a zero is generated for each, and that for 5 or more, a zero is generated and an immediate REPI entry produced. This speeds up the BSSZ output process.

ORGCTR (2) - Origin Counter.

This two-word area contains the current value of the origin counter. During pass 1, the second word contains the block number under which the assembly is proceeding. During pass 2, it contains the relocation of the origin counter, with values 1,2,3... The first word contains a 21-bit integer, right justified with zero fill, which is initialized to 0 at the start of each pass. It is updated by UPLOC and FOUP, and is saved and reset by USE processing. On a given statement, the value of the origin counter in the two passes must be equal after relocation.

ORIGINS (NTABLES) - Origins of Managed Tables.

This vector contains one word for each managed table containing the current first word address of that table. Entries in ORIGINS are indexed by table number. ORIGINS parallels SIZES.

OVLHDR - Overlay Header Word.

This word contains the overlay header word (type 50B or 51B) when a CPU absolute overlay is assembled. It immedaitely follows the PRFX table image so that both can be written with one PUT request. It is set and used by DFIRST.

P (8) (Debugging only) - Patches FET or FIT.

This is the File Environment/Information Table for the file from which patch/snap debugging directives are read in pass 0 if COMPASS was assembled with DEBUG = 1. It points to the circular buffer area BBUF.

PAGENO - Page Number.

This is the thirteenth word of TITBUF and contains the decimal page number with 4 trailing blanks. It is updated by incrementation for each page ejected.

PASS - Assembler Pass Number.

This word contains either a 0, 1, 2, or a 3, depending upon the pass of assembly.

Pass 0 is considered that section involved with system text definition. Pass 3 is considered that section involved with cross-reference table listing.

The flag is used only in the manager routine to determine the appropriate action to take when the tables overflow.

PCC - Prefix Character Count.

This word contains the length, in characters, of the PRFX table. It is set and used by DFIRST when a Record Manager is used for assembly-time I/O

PGCNT - Page Count in Binary.

PGCNT records the current long listing page number, and is used only in recording the page number for the error directory. It parallels PAGENO, which is the decimal version

of the page number. It is inititalized to 0 in pass 2 initialization (unless page number propagation is selected by the control card P option), and is incremented by one for each page eject.

PLFLG (Pass 2 only) - Print Line Ready Flag.

This word is used by CPL to indicate that a print line has been created. It is cleared by RINTER each time a card is read. This flag is used so several calls can be made to CPL and only one print line will be assembled.

POSCTR - Position Counter.

This word contains the number of available bits remaining in a word. It is incrementally updated by UPPOS, and reset to LWORD (length of word) in FOUP. It contains values between 60 and 00. A value of 60 marks the assembly at the top of a word, and 00 is a transient value, which will be changed to 60 when FOUP is entered. When the address element $ is used, one less than the value of POSCTR is returned. A value of 00 is impossible for POSCTR at this time, since the position counter is updated after a field is assembled.

PPJUMP - PP Jump Flag.

This word is normally zero. It is set non-zero by a PERIPH or PPU pseudo-operation with a J in the variable field. ZPP examines this word to determine how to evaluate xJN type PP instructions. If this word is zero, ZPP evaluates the address expression and tests this expression to determine if it falls between -32 and +32. If this test fails, the location counter is subtracted from the expression for the value of the relative jump. However, if PPJUMP is not zero, only the second test is made. That is, the value of the relative jump is always the expression minus the location counter.

PPTYPE - PERIPH/PPU Switch.

This word is initially 0 and is set to 1 by a PPU pseudo-operation in pass 1. It is irrelevant in a central processor assembly. For a peripheral processor assembly, it specifies the format expected for the first IDENT statement in pass 2 and the format of the header word in the binary output written by DFIRST. PPTYPE = 0 designates 6000 SCOPE 3 formats; PPTYPE = 1 designates 7000 SCOPE 1 formats.

PRFX (15) - Binary Prefix Table Image - same as DPBA.

PRFXC (7) - Comment Field in PRFX Table Image.

This area contains the comment text from COMMENT pseudo instructions. It is set and used by GSM and DFIRST.

PSIZE - Page Size.

This word contains the nominal number of lines per page. It normally contains PAGESIZ+5, but is set to 0 by the N parameter on the COMPASS control card. Various pass 2 routines accomplish a page eject by adding PSIZE to LPCNT. If the N parameter was not specified, this makes LPCNT greater than PAGESIZ so that the next line printed will be at the top of the new page. But, if the N parameter was specified, PSIZE is zero and adding it to LPCNT has no effect.

P1TEMP, P1TEMPA,...,P1TEMPE (Pass 1 only) - Temporaries.

These words are used for temporary storage by many pass 1 routines.

QNAME (Pass 2 only) - Qualifier Name.

This word contains the current qualifier name, left justified with blank fill, to be printed along with the subtitle. It reset each time the QUAL pseudo-operation is called.

QUALMIC (Pass 1 only) - QUAL Micro Value.

This word contains the characters (normally none), left justified with 00 fill plus a character count (normally 0), that is the current value of the QUAL builtin micro. It is set and used by the QUAL pseudo-op processor and is also used by TLUMIC when evaluating a reference to the QUAL micro and the MICTAB lookup fails.

QUALSTK - QUAL Pushdown Stack.

This area is a pushdown stack, with room for up to MSTACK entries, that is pushed by a QUAL name statement and pulled by a QUAL * statement.

QVAL (2) - Symbol Qualifier Value.

The first word contains in bits 59-48 an index into the QVTAB that indicates the symbol qualifier currently in use. A value of zero indicates global symbols. The second word is a scratch cell that holds the current qualifier when a new temporary qualifier is being used. For example, EVITEM resets the first word when it comes across symbols of the form/qualifier/symbol. QVAL is set by the QUAL pseudo-operation and is cleared at the beginning of both passes.

R {8} - Reference FET.

This is the File Environment Table that is used for writing overflow from the symbolic reference table {REFTAB} to a scratch file during pass 2. It points to the circular buffer area RBUF.

RA.CCD {8} - Control Card Image {defined by SYSCOM macro}.

This area contains the operating system control card that invoked COMPASS, and is also used for reading continuation control cards.

RA.LDR - Loader Communication Word {defined by SYSCOM macro}.

COMPASS clears this word whenever it requests the operating system to load an overlay, and waits until the system makes the word non-zero to signal completion of the load operation.

RBUF {RBUFL} - Reference Buffer.

This is the circular buffer area used in pass 2 for writing REFTAB overflow to a scratch file. Its length is changed to zero if the long listing is suppressed {L=0 on a COMPASS control card}, since symbol references are not collected in this case.

REFIO {Pass 2 only} - Cross Reference Overflow Flag.

This word is initially 0. When non-zero, it flags that the reference table REFTAB has overflowed to the scratch file SCR1. It is set by the table manager routine ALC when the table overflows. ENTREF examines it to determine the destination of cross-references and PRT examines it to determine the source of the cross-reference table.

REFLET {Pass 2 only} - Cross Reference Usage Letter.

This word is normally blank. It is set non-blank when a special character is to be appended to the cross-reference entry to indicate usage of the sumbol. Currently, the usage letters are:

L Symbol appears in the location field of a machine instruction, a data-generating of BSS or END pseudo instruction, or a call to a type 1 macro.

D Symbol is defined by a SET, EQU, =, MAX, MIN, or MICCNT operation.

E Symbol appears in an ENTRY statement.

F Symbol is used in an IF-type operation with LIST F on.

S Symbol is used for storage {SA6, or SA7 instruction}.

X Symbol appears in an EXT statement.

blank symbol is referenced in any other way.

RELVEC (256) - Relocation Vector.

This area has several uses, but only one at a time. In the subroutine SCAD it records the relocation coefficients of each term. Because there can be 255 blocks in an assembly, this large vector is needed. Only the first USECNT words of the array are used, and thus cleared. This is purely a time saving technique.

While processing MACRO, MACROE, OPDEF, and ECHO definitions, and calls to macros defined by MACROE, the first few words are used to store the names of the formal parameters. They are placed into RELVEC as each new unique name is found in PMACF and searched for when looking for substitutable references to parameters.

While processing a call to a macro that was defined by MACROE, the complete list of formal parameter names (keywords) is moved from MACDEF to the words beginning at RELVEC+64, so that the MARDIS entries can be arranged in the proper order.

While processing an ECHO statement or a call to a macro that was defined by MACRO or MACROE, each argument value is packed into the words beginning at RELVEC+128 and is then added to MARGS, so that the table manager is called only once per argument rather than for each ten characters of an argument.

While processing a call to a macro that was defined by OPDEF, the characters comprising the address field are stored in the words beginning at RELVEC while the register substitutions are placed directly into MARGS. At the end of the subfield, the address is moved from RELVEC to MARGS. This is necessary since the address field is being re-arranged.

Whenever a micro is {re} defined {by pass 1 initialization, BASE, CODE, DECMIC, OCTMIC, QUAL, or MICRO}, the micro's new value and name are packed into RELVEC before EMT is called to enter the micro table.

In pass 1 processing of the LCC pseudo instruction, the loader directive card image is packed into RELVEC and written from there to the binary output file.

In pass 1 processing of the SST pseudo instruction, the symbols listed in the variable field are stored in RELVEC.

The pass 2 subroutine RINTER reads each record from the intermediate table or file into RELVEC and distributes it from there to OPTYPE, IND, FLAG, SEQ, and CARD.

REQC - R= Switch.

This word is set to zero at the beginning of both passes. It is used for the R= pseudo-operation to indicate that either the B1=1 or B7=1 pseudo-operation has been called. It contains the table address for substituting instructions for the R= pseudo.

RIFA (Pass 2 only) - Intermediate Header Word.

When the intermediate table has overflowed to a scratch file, RINTER uses this word to hold the one-word header that contains the length of the intermediate record. For speed, COMPASS reads the header word of the next record along with the text of the current record. This allows COMPASS to obtain the length required for the next read without making an access to the circular buffer.

RMTFLG - Remote Flag.

This flag is included in IND on the intermediate file. RMTFLG is set by HERE (and END) processing at the point labeled HEREPK when the contents of RMTAB are moved to RASTAB and the stack is pushed down to start unpacking the remote assembly lines. It is also used in pass 1 processing by the HERE and END processors to detect when the original HERE or END cards are re-encountered during the unpacking process. RMTFLG is used to control listings in pass 2 based on the D list option.

S (8) - Scratch FET or FIT.

This is the File Environment/Information Table used for writing overflow from the intermediate table (INTER) to a scratch file in pass 1 and reading it in pass 2. It points to the circular buffer area SBUF.

SBNAME (Pass 2 only) - Subroutine Name.

This word contains the current subroutine name, left-justified with blank fill, for use in the subtitle line. This name is taken from SUBNAME each time the title line is printed.

SBUF (SBUFL) - Scratch Buffer.

This is the circular buffer area for the scratch file that contains INTER overflow.

SEGEPT (pass 2 only) - Entry Point of Segment.

This word is used for absoute CP and 7000 PP output only. It is set from the third argument on the IDENT card, and the second argument on the SEGMENT card. It is used to create the 50B header word that precedes CP overlays and the 52B header word that precedes 7000 PP overlays.

SEQ (2*NCARDS) - Card Sequencing Information.

This area is used to record the contents of columns 73-90 of each card. Only 2*CCT words of SEQ are used for each statement. When a card is read in pass 1, the trailing columns are removed from the card, the card is spread into CARD and the contents of columns 73-90 are placed into SEQ. Normally, there are two words in SEQ for each card of the statement, with the following format:

Word 1	Bits 59-48	Zeros.
	Bits 47-00	Card columns 73-80.
Word 2	Bits 59-00	Card columns 81-90.

When the statement came from the source input file (L.STACK 0) and CP.IFORM = +1 (MODIFY compressed compile file), there is only one word in SEQ for each card of the statement, with the following format:

Bits 59-18	Identifier name in display code left justified with 00 fill.
Bits 17-00	Sequence number as a binary integer.

When the card is read in pass 2, the card and sequence information are reconstructed for listings. There are always two words in SEQ for each card of the statement in pass 2. For the second case described above, the first word is zero. CPL blank-fills the identifier name and converts the sequence number to decimal.

If the statement is internally generated, INPUT1 places the name from the stack (fourth word) and the level number into the sequence array. Thus, the listings of those internally generated lines contain the identification placed there and the level number. For macros, it is the macro name. For OPDEF it is the contents of the operation field. For DUP, ECHO, and unlabeled HERE (or END), it is the words "*DUP*", "*ECHO*", or "*RMT*". For a labeled HERE, it is the remote text group name. In the case of XTEXT, it is the file name.

SEQMIC {2} {Pass 1 only} - SEQUENCE Micro Value.

These words contain the current value of the SEQUENCE builtin micro. The first word contains card columns 73-82, and the second word contains card columns 83-90 and a character count of 8, from the first card of the statement most recently read from the source input file. It is set by INPUT1 and is used by TLUMIC when evaluating a reference to the SEQUENCE micro and the MICTAB lookup fails.

SI - Segment Index.

This word contains the index into the segment table SEGTAB of the current segment. During pass 1, it is equal to L.SEGTAB if there have been no SEG or SEGMENT cards since the last IDENT card. At the beginning of pass 2 it is used by the routines that print the storage allocation page.

SIZCORE - Size of Managed Table Area.

This word contains the length of the table pool area. It is initialized by COMPASS initialization to be {CP.NFLS}-BUCKET-10. It is adjusted by ACL whenever the low core limit is adjusted. The managed tables always lie in the area starting at LOCORE and are contained entirely in SIZCORE amount of storage. This cell is used extensively in the table management routines.

SIZES {NTABLES} - Sizes of Managed Tables.

This vector contains one word for each managed table containing the current length in words of that table. Entries in SIZES are indexed by table number. SIZES parallels ORIGINS.

SNA (8) (Debugging only) - Snapshot A Registers.

The A registers are saved in this area when the snapshot routine is entered, so they can be printed and later restored.

SNAPBUF (100) (Debugging only) - Snap Descriptor Buffer.

Snapshot descriptors are constructed and stored in this area by the pass 0 routine RDD when it reads a *SNAP directive. The specified instruction word is saved in SNAPBUF and the instructions

```
+    RJ   /DEBUG/SNAPPER
-    VFD  30/n
```

are stored in the original word; n is the relative address within SNAPBUF of the saved instruction word. The word in SNAPBUF is then followed by one word for each memory area to be dumped. These words have the following format:

```
VFD  1/end,11/0,1/i,17/fwa,12/0,1/j,17/wdct

     end = 1 if this is the last descriptor for this
           snapshot,
     i   = 1 if fwa is indirect address,
     fwa = first word address of memory area to dump,
     j   = 1 if wdct is indirect address,
     wdct = word count of memory area to dump.
```

SNAPPTR (Debugging only) - Snap Descriptor Pointer.

This word contains the address of the descriptor word within SNAPBUF that is currently being processed.

SNB (8) (Debugging only) - Snapshot B Registers.

The B registers are saved in this area when the snapshot routine is entered, so they can be printed and later restored.

SNINST (Debugging only) - Saved Instruction Word.

This word contains the instructions that were replaced by the RJ SNAPPER.

SNLINE (14) (Debugging only) - Snapshot Line.

This area contains the current print line image for snapshot output.

SNP (Debugging only) - Snapshot P Register.

This word contains the location of the word containing the RJ SNAPPER. It is printed as the P register in the register dump.

SNTEMP (Debugging only) - Temporary.

This word is used for temporary storage by the snapshot routine.

SNX (8) (Debugging Only) - Snapshot X Registers.

The X registers are saved in this area when the snapshot routine is entered, so they can be printed and later restored.

SQIMAGE (71*NCARDS/10+3) - Squeezed Image.

SQUEEZE uses this area for packing a card. When SQUEEZE is called, the statement in CARD up to LASTCOL is packed into SQIMAGE, and the length of the packed image is stored in SQLGN.

SQLGN - Squeezed Length.

SQLGN contains the length of the packed image after SQUEEZE has been called. Also this cell serves as a flag that the card has been packed.

SSTCNT - System Symbol Count.

This word contains the number of entries that have been moved from the system symbol table (SSYMS) to the assembly symbol table (SYMTAB) and is set by the SST pseudo-instruction in pass 1. It is used by the END processor in pass 2 to restore the SSYMS entries.

STACKPTR (6) - Stack Pointers.

This is a list of stack pointers, terminated by a zero word, used by CPS to clear the pushdown stack areas (BASESTK, CODESTK, LISTSTK, QUALSTK, and USESTK) to their initial conditions at the start of each assembly.

STCNT - Statement Count.

This word is initialized to 0 at the beginning of pass 1. It is incremented by WINTER each time it is called, and thus does not include those statements screened out by CWI. It is used by the pass 2 END processor for inclusion in the first line following the END statement in the assembly listing.

STYPE - Statement Type.

This word must immediately precede the CARD area. It is the first column of every statement, and may be regarded as column 0. When a statement is squeezed for table storage or for the intermediate file, STYPE is always the first character stored. Its possible values are:

blank	For a normal statement.
*	For a card in which column 1 is an asterisk or comma.
T	For a terminator statement (end of macro definition or text defined by DUP, ECHO, RMT, or XTEXT). This is a transient value since INPUT1 will immediately decide whether to push up the stack and get another statement, or in the case of an unsatisfied DUP or ECHO, to reset the card pointer and iterate. Thus, the T card never appears outside INPUT1.
U	For an IRP statement. When an IRP is found while packing a macro definition into MACDEF, it is followed by a dummy statement in which STYPE is U and column 1 contains the formal parameter number (01-77 octal). When the second IRP of a pair is found, it is preceded by a dummy statement in which STYPE is U and column 1 contains the 00 character. Thus, the two U cards delimit the text to be expanded iteratively when the macro is called. Each U card is a transient value that directs INPUT1 processing; no U card ever appears outside INPUT1.

E — For the final END statement. This is stored in pass 1 with the END statement that is packed into RMTAB and then read, after all unprocessed unlabeled remote text, by the automatically generated HERE statement. Thus, in both passes, the real END statement is distinguished from those generated internally.

SUBNAME (Pass 2 only) - Sub-Subtitle Name.

This word contains the current sub-subtitle, up to 10 characters in display code, right justified with 00 fill. It is initially set to 0, and is subsequently set from the location field of each CTEXT (if the X list option is on), EJECT, SPACE, TITLE, or TTL statement or central processor 000 instruction (such as PS). It is also set to the current qualifier by PRT and to the constant LITERALS by PLT. It is used by LHD at the start of each listing page, when it is stored left justified with blank fill into SBNAME so that it prints in the second title line.

SUBTIT (10) (Pass 2 only) - Subtitle Line.

This area contains the second title line to be printed at the top of each listing page. It contains SUBTITL, SBNAME, UNAME, and QNAME.

SUBTITL (6) (Pass 2 only) - Subtitle Text.

This area contains the text from the variable field of the most recent TITLE (except the first) or CTEXT (if the X list option is on) statement.

SUPREF (Pass 2 only) - Suppress References Flag.

This flag is set by RINTER, for each statement, to the logical value:

(list R off) or ((list X off) and (LIBFLG = 1))

so that SUPREF is 0 if symbolic references are to be accumulated for this statement or 1 if they are not. ENTREF does nothing when it is called while SUPREF is 1.

SYMCNT - Symbol Count.

This word is initialized to 0 and is incremented by ENTSYMT each time it makes a new entry in the symbol table (SYMTAB).

It is used only by the pass 2 END processor, which prints the symbol count in the first line following the END statement.

SYNAME - System Text Generation Name.

This word contains the name of the system text overlay to be generated at the end of pass 1. It is set to zero at the beginning of pass 1, and is set by the STEXT pseudo-operation. If both binary and systext overlay generation is requested, STEXT zeros this word in pass 2 so the binary will not be suppressed.

SYSFLG - System Macro Flag.

This flag is included in IND on the intermediate file. SYSFLG is set in MACALL when a system macro is called. It gets cleared by macro pushups. A system macro is detected because the type-field in OPTYPE for the entry in the operation code table contains a 6 instead of a 7 which marks programmer macros. SYSFLG is used in pass 2 to control listings based on the S list option.

T (8) - Reference FET.

This is the File Environment Table that is used, at the beginning of pass 3, to read the symbolic reference table (REFTAB) if it overflowed to a scratch file during pass 2. It points to the circular buffer area TBUF.

TARGET - Target Processor Type.

This word contains two characters representing the type of processor for which the program being assembled is optimized. It is normally two blanks and can be set otherwise by the MACHINE pseudo instruction. It is used by DFIRST in bits 29-18 of word 7 of the PRFX table.

TBUF (RBUFL) - Reference Buffer.

This is the circular buffer used for reading REFTAB. It overlays the portion of COMPASS that is saved on another scratch file at the beginning of pass 3.

TECOE - Term Coefficient.

This word contains the coefficient for the address expression term being evaluated by SCAD. At the beginning of a term, TECOE is set to 1 if the first element in the term is relocatable; otherwise, it is set to the value of the first element.

TFEXT - Term External.

This word contains the external number for the current address expression term.

TEOP - Term Operator.

This word contains a code representing the operator that precedes the current address expression term, as follows:

1	none
2	+
3	-
5	^

TEREL - Term Relocation.

This word contains the relocation for the current address expression term. The representation is the same as in EXREL, q.v.

TEVAL - Term Value.

This word contains the value of the current address expression term, with sign extension through all 60 bits.

TIME - Current Time.

This word contains the current time of day in display code, and is set by a CLOCK request to the operating system at the beginning of each assembly. It is used in the title line as well as in the preset micro "TIME".

TITBUF (7) - Title Buffer.

This area is used for the page eject title on the listings. It is preset to the information from the IDENT card, so that if no TITLE pseudo-operation appears, a meaningful title will result. The first word is also preset to a 1 in the leftmost character for page ejects.

TITFG - Title Flag.

This flag records the first occurrence of a TITLE pseudo-operation. When the first one occurs, the text of the title is moved to the area TITBUF and the flag is set on. Subsequent occurrences of TITLE in pass 1 have no effect.

In pass 2, this flag is again used so that the first occurrence of TITLE is ignored. Subsequent occurrences set the subtitle into SUBTIT.

TLINE (2) - Source Machine Type.

These words contain °MODEL xx ASSEMBLY° where xx may be 72, 73, 74, or 76. It is set by CTM to the type of machine in which COMPASS is running, and is included in the second line following the END statement for each assembly.

TXTFLG - Text Definition Flag.

This flag is included in IND on the intermediate file. TXTFLG is set for those statements which are being read in during pass 1 as part of a definition. These include the following:

1. All statements between a MACRO, MACROE, or OPDEF and its ENDM.

2. Those statements between RMTs being saved.

3. Those statements under the control of DUP or ECHO being saved for duplication.

During pass 1 of course, all statements for which TXTFLG is set are not edited. This is because all those routines which store definitions do not call EDIT. The statements will be edited later when they are actually assembled.

During pass 2, all statements for which TXTFLG is set are merely bypassed in the assembly process, but are listed normally.

In the above, the RMT, ECHO, MACRO, MACROE, and OPDEF operations are included in the set of cards for which TXTFLG is set. The DUP statement, however, is not included. It is assembled in pass 2 merely so that the iteration count can be listed.

UI (2) - USE Table Index.

The first word, UI, contains the index into the USETAB for the first block. It is zero unless the IDENT pseudo-operation was used. The second word, UI+1, contains the number of the first or absolute block. This normally is one unless the IDENT pseudo-operation is used. These words are set from SEGTAB in pass 1 and are used tc make the secondary overlays appear to be new programs.

UNAME (pass 2 only) - USE Block Name.

This word contains the name of the current USE block, left-justified with blank fill, for use in the subtitle line. It is cleared at the beginning of pass 2 and is set by the USE, ORG, and USELCM pseudo-operations.

USECNT - USE Block Count.

This word contains the number of USE blocks recorded in USETAB. It is equal to L.USETAB/4, but is maintained incrementally. It is used in searching USETAB, and also in deciding how many entries of RELVEC need be cleared prior to address calculation. It is initialized to 3 at the start of assembly to reflect the first three blocks.

USESTK - USE Pushdown Stack.

This area is a pushdown stack, with room for up to MSTACK entries, that is pushed by a USE name, USELCM name, ORG, or ORGC statement and pulled by a USE * or USELCM * statement.

VALID - Valid Processor Type.

This word is normally zero and can be changed to 2R6: or 2R7: by the MACHINE pseudo instruction, representing the type of processor on which the program being assembled can be executed successfully. It is used by DFIRST to form bits 17-06 of word 7 of the PRFX table.

VALUES {NLITS} - Data Values.

This area contains the data values, converted to binary, during processing of DATA, DIS, and LIT pseudo-instructions and evaluation of literals in address expressions. It is also used as a next card buffer when reading XTEXT input; in this case, it is used in the same way that CP.CARD is used for the main source input file.

WECNT - Warning Error Count.

This word contains the number of warning errors in the current assembly. It is initialized to 0 at the beginning of each assembly, incremented by LISTER, and if non-zero it is issued in a local dayfile message by the pass 2 END processor.

X {8} XTEXT Input FET or FIT.

This is the File Environment/Information Table used by the XTEXT pseudo-op processor for reading alternate source input files. It points to the circular buffer area BBUF.

XLIST - External List Control.

This word is used to flag that the L0 option has been used on the COMPASS control card. If this word is non-zero, the LIST pseudo-operation is ignored.

XR - Reference Type.

The content of this word specifies the format of the symbolic reference table listing as follows:

-1 Page/line only.
-0 Address only.
+1 Both page/line and address.

It is initialized to -1 at the beginning of each assembly and can be set to any of the above values by XREF in pass 1. It is used by the pass 3 routine PRT.

4.0 MANAGED TABLES

Most of the tables used by COMPASS are variable in size. Details on the table management methodology are given in section 2.1. The managed tables are in four categories: global, assembly, pass 1, and pass 2. All of the tables are described below, listed in alphabetic order within each category. In all of the tables that contain packed source statements, the format is as described in section 2.11. Most of these tables have no maximum size, other than the limit on the total of their sizes imposed by the job's field length. Tables INTER and REFTAB have no maximum size at all, since these tables can overflow to mass storage scratch files. OPTAB has a maximum size imposed by the length of its hash link field. USETAB has a maximum size imposed by the length of the block number field in various other tables. The STACK table has an arbitrary maximum size so that COMPASS can provide diagnostic messages more informative than TABLE OVERFLOW IN PASS 1 when runaway recursion, a fairly common programmer error, occurs.

4.1 GLOBAL TABLES

The global tables are created before the first assembly is begun, and continue to exist throughout the batch of assemblies performed in one COMPASS run.

4.1.1 ENDTAB - End Table.

Dummy table used by the table manager. It is always the last ten words of the job's central memory field length.

4.1.2 MACDEF - Macro Definition Table.

Stores the definitions of the macros and opdefs from system text and programmer sources. Only the first (LSYSMAC) words are retained between assemblies.

4.1.3 OPTAB - Operation Code Table.

Contains the name and information about every operation code. The maximum size of OPTAB is NOPCT+4095 entries. At the conclusion of each assembly, all programmer-defined entries are removed, and all entries defined by COMPASS and system text are restored.

Entry = 2 words.

Word 1 (PPU, pseudo, or defined by PPOP, OPSYN, MACRO, or MACROE):

Bits	59-48	Link field for hashing.
Bits	47-00	Operation name right justified with 00 fill.

Word 1 (CPU or defined by CPOP, CPSYN, or OPDEF):

Bits	59-48	Link field for hashing.
Bits	47-36	Two-character mnemonic.
Bits	35-28	N1.
Bits	27-20	N2.
Bits	19-12	N3.
Bits	11-00	0055B.

Where N1, N2, and N3 are as follows:

Bit	7	Leading sign: 0 - plus. 1 - minus.
Bits	6-5	Register name: 0 - blank. 1 - A. 2 - B. 3 - X.
Bits	4-3	Operator: 0 - blank or plus (+). 1 - minus (-). 2 - multiply (*). 3 - divide (/).
Bits	2-1	Register name: 0 - blank. 1 - A. 2 - B. 3 - X.
Bit	0	Constant (Q-field).

Word 2 (CPU or defined by CPOP or CPSYM):

Bits	59-57	0.
Bits	56-48	Value - upper 9 bits of opcode.
Bit	47	Programmer-defined flag.
Bits	46-32	Unused.
Bits	31-30	Machine type: 0 - Both. 1 - 6000 only. 2 - 7000 only.
Bit	29	Force upper after instruction.
Bit	28	Force upper before instruction.
Bit	27	30-bit instruction
Bits	26-24	Source of I-field.
Bits	20-18	Source of K-field.
Bits	17-00	Unused.

The source of a register number field is specified by one of the following codes:

0	None, use 0.
1	Opcode field.
2	Second or only address field register.
3	First of two address field registers.

Word 2 (PPU or defined by PPOP):

Bits	59-57	1.
Bits	56-48	Unused.
Bit	47	Programmer-defined flag.
Bits	46-32	Unused.
Bits	31-30	Machine type: 0 - Both. 1 - 6000 only. 2 - 7000 only.
Bits	29-27	CTL: 0 - Unused. 1 - 24-bit instruction with 12-bit address and no indexing (none). 2 - 12-bit instruction with signed relative or absolute address (UJN). 3 - 24-bit instruction with 18-bit address (LDC). 4 - 12-bit instruction with 6-bit address (LDN). 5 - 24-bit instruction with 12-bit address optional indexing (LDM). 6 - 12-bit instruction with signed relative address (SHN). 7 - 24-bit instruction with 12-bit address and required second field (IAM).
Bits	26-12	Unused.
Bits	11-00	Value.

Word 2 (Pseudo):

Bits	59-57	Pseudo operation type: 2 - Can not occur in first card group. 3 - Process while IF skipping. 4 - Can occur anywhere. 5 - First card group only.
Bits	56-48	Unused.
Bit	47	Programmer-defined flag.
Bits	46-36	Unused.
Bits	35-18	Pass 1 pseudo address.
Bits	17-00	Pass 2 pseudo address.

Word 2 (MACRO, MACROE, or OPDEF):

Bits	59-57	Macro operation type: 6 - System macro. 7 - Programmer macro.
Bits	56-39	Word count of macro definition.
Bit	38	Flag set and used by GSM.
Bits	37-32	Unused.
Bit	31	MACROE flag.
Bits	30-25	Count of formal + local parameters.
Bits	24-19	Count of local parameters.
Bit	18	Location argument flag.
Bits	17-00	Index in MACDEF of start of macro.

4.1.4 SSYMS - System Symbols.

Stores the symbols defined by the SST pseudo operation. Symbols come from system text.

Entry = 2 words.

Word 1: Symbol right adjusted with 00 fill.

Word 2:

Bits	59-39	Unused.
Bits	38-36	System text ordinal.
Bits	35-21	Unused.
Bits	20-00	Symbol value.

4.1.5 SYSMIC - System Micros.

Stores the portion of the micro table defined by system text.

Entry = N words {N = 1 if micro value is null}.

Words 1-(N-1): Value of micro, as a character string packed ten characters per word. In last word (word N-1), bits 59-06 contain 0-9 characters left adjusted with 00 fill and bits 05-00 contain character count for this word.

Word N:

Bits	**59-48**	**2000B + N (for floating unpack).**
Bits	**47-00**	**Micro name right adjusted with 00 fill.**

4.2 ASSEMBLY TABLES

The assembly tables exist throughout an assembly and are cleared at the conclusion of each assembly.

4.2.1 EPTAB - Entry Point Table.

Names of symbols appearing in ENTRY statements.

Entry = 1 word.

Bit	59	Conditional {ENTRYC} flag.
Bits	58-00	Symbol right justified with 00 fill.

4.2.2 EXTAB - External Symbol Table.

Records each external symbol.

Entry = 1 word.

Bits	59-00	Symbol name right justified with 00 fill in Pass 1, left justified with 00 fill in Pass 2.

4.2.3 IDTAB - IDENT Card Table.

Used to hold the text of binary control cards for listing at the start of Pass 2.

Entry = M words.

Word 1:

Bits	59-48	Qualifier index in effect (QVAL).
Bits	47-18	Unused.
Bits	17-00	Number base in effect (NBASE).

Words 2-N: Compressed text of binary control card.
Words N-M: Text of COMMENT cards.

4.2.4 INTER - Intermediate File.

INTER is used to contain the intermediate file if it will fit in core.

Entry = 3 words, sequence numbers, and compressed text.

Word 1:

Bits	59-48	A copy of bits 59-48 of OPTYPE. This is the same as the operation code table entry.
Bit	47	Unused and zero.
Bits	46-45	SEQ flag. If this is 00, then the sequence fields of this statement are blank and are not recorded on the intermediate file. If this is 01, the sequence fields are in MODIFY format, one word per card image. If this is 10, the sequence field is two words and is the same for all cards in the statement (e.g. macro generated), so the two-word sequence field is recorded only once on the intermediate file. If this is 11, the intermediate file contains a two-word sequence field for each card in the statement.

Bit	44	FLAG flag. If this is 0, then the FLAG word is zero, and is not included on the intermediate file. The FLAG word is non-zero only for some pseudo instructions.
Bit	43	IND flag. If this is 0, then the IND word is zero, and is not included on the intermediate file. The IND word contains error flags and other indicators.
Bit	42	Unused.
Bits	41-34	Word count of this INTER entry, including this word.
Bits	33-30	CCT - card count, i.e., number of cards which comprise this statement.
Bits	29-00	Copy of bits 29-00 of OPTYPE.

Word 2: Present explicitly if the IND bit in word 1 is 1. If it is 0, then word 2 can be assumed to have a zero value.

Bits	59-30	These contain a record of the indicators which were set. A 1-bit indicates that the corresponding indicator was on.
Bits	29-00	These contain a record of the error flags which were set. A 1-bit indicates that an error flag was on. The exact order of these error bits depends upon the order in which the errors are listed in the error list (see ERFLAGS).

Word 3: Present explicitly if the FLAG bit in word 1 is 1. If it is 0, then word 3 can be assumed to have a zero value.

Bits	59-00	Contents of FLAG.

Words 4-N: Sequence number field if SEQ in word 1 is not 00. The length of this entry depends on the value of SEQ as follows:

SEQ = 00	0 words.
SEQ = 01	(CCT) words.
SEQ = 10	2 words.
SEQ = 11	2*(CCT) words.

Word 4 - IF SEQ = 01 (MODIFY compressed compile file):

Bits	59-18	Identifier name left justified with 00 fill.
Bits	17-00	Sequence number in binary.

Word 4 - SEQ = 10 or 11:

Bits	59-48	Zero.
Bits	47-00	Columns 73-80 of card image.

Word 5 - SEQ = 10 or 11:

Bits	59-00	Columns 81-90 of card image.

Words N-M - Compressed card text terminated by at least 12 zero bits.

4.2.5 LITAB - Literal Table.

Literals defined during Pass 1.

4.2.6 MEMORY - Memory Table.

Used to hold system text during Pass 0, XTEXT random file index during Pass 1, binary (core image) for absolute programs during Pass 2, and cross-reference entries during Pass 3.

4.2.7 QVTAB - Qualifier Name Table.

Names of qualifiers as they occur. There is no entry for the blank qualifier. The first entry is counted as entry number 1.

Entry = 1 word.

Bit	59	No reference flag.
Bits	58-48	Unused and zero.
Bits	47-00	Qualifier name right adjusted with 00 fill.

4.2.8 SEGTAB - Segment Table.

Records all relevent information about each segment or partial segment.

Entry = 4 words.

Word 1:

Bits	59-30	Unused.
Bits	29-21	Relocation of LWA of segment.
Bits	20-00	Relative LWA of segment.

Word 2:

Bits	59-36	Unused.
Bits	35-18	USE table index.
Bits	17-00	IDTAB index.

Word 3 (Pass 1):

Bits	59-54	Unused.
Bits	53-36	SLITS index.
Bits	35-18	EPTAB index.
Bits	17-00	LITAB index.

Word 3 (Pass 2):

Bits	59-54	Unused.
Bits	53-36	SLITS FWA index.
Bits	35-18	EPTAB FWA index.
Bits	17-00	LITAB FWA index.

Word 4 (Pass 2):

Bits	59-54	Unused.
Bits	53-36	SLITS LWA index.
Bits	35-18	EPTAB LWA index.
Bits	17-00	LITAB LWA index.

4.2.9 SLITS - Symbol Literals.

Names of symbol literals.

Entry = 1 word.

Bit	59	Set to 1 if symbol defined by COMPASS.
Bits	58-57	Type: 1 - =S type symbol. 2 - =X type symbol. 3 - =S and =X type symbol.
Bits	56-48	Current qualifier index.
Bits	47-00	Symbol name right adjusted with 00 fill.

4.2.10 SYMTAB - Symbol Table.

Symbols defined during an assembly.

Entry = 2 words.

Word 1:

Bits	59-48	Qualifier index.
Bits	47-00	Symbol name, right adjusted with 00 fill.

Word 2:

Bits	59-42	Link field for hashing.
Bits	41-39	Unused.
Bits	38-36	Systext ordinal.
Bit	35	No reference flag.
Bit	34	XTEXT flag.
Bit	33	Redefinition flag.
Bit	32	SST flag.
Bit	31	External flag.
Bit	30	Defined flag.
Bits	29-21	Relocation or external number.
Bits	20-00	Value of symbol.

4.2.11 USETAB - Program Block Table.

Counters for program blocks. The first entry is counted as entry number 1. The maximum size of USETAB is 255 entries.

Entry = 4 words.

Word 1:

Bits	59-00	Block name right justified with 00 fill. Special block names are: LCM block - complemented name.

	Pass 1	Pass 2
Block 1	ABSOLUTE*	PROGRAM* or ABSOLUTE*
Block 2	one blank	PROGRAM*
Block 3	LITERALS*	LITERALS*

The name of block 1 is PROGRAM* in a PPU or absolute CPU assembly, or ABSOLUTE* in a relocatable assembly.

Word 2:

Bits	59	Conditional load flag (Pass 1 only)
Bits	58-42	Current RELTAB halfword index (BINREL).
Bits	41-30	Unused.
Bits	29-24	Current value of position counter.
Bit	23	Value of NFOUP flag (force next upper).
Bits	22-21	Unused.
Bits	20-00	Value of origin counter.

Word 3 (Pass 1):

Bits	59-54	Unused.
Bits	53-33	Maximum origin of block.
Bit	00	Common flag.

Word 3 (Pass 2):

Bits	59-33	Unused.
Bits	32-24	Relocation of block.
Bits	23-21	Unused.
Bits	20-00	Origin of block.

Word 4 (Pass 1):

Bits	59-21	Unused.
Bits	20-00	Maximum origin of block.

Word 4 (Pass 2):

Bits	59-00	Partial binary word (BINWORD).

4.3 PASS 1 TABLES.

The Pass 1 tables exist only during the first pass of an assembly, and are cleared at the conclusion of Pass 1. Some of them share pointer words (ORIGINS and SIZES entries) with Pass 2 tables.

4.3.1 DUPTAB - Duplication Table.

Used to hold DUP compressed text during assembly.

4.3.2 ECHTAB - ECHO Table.

Used to hold ECHO compressed text during assembly.

4.3.3 LASTAB - Library Assembly Table.

Used to hold XTEXT compressed text during assembly.

4.3.4 LRMTAB - Labeled Remote Table.

Used to hold labeled remote names and compressed text.

Entry = N words.

Word 1: Remote name, right justified with 00 fill.

Words 2-N: Compressed text.

4.3.5 MARDIS - Macro Argument Discriptors.

Contains pointers into MARGS for the actual parameters of a macro expansion or ECHO.

Entry = 1 word.

Non-iterative form:

Bits	59-18	Zero.
Bits	17-00	Index into MARGS table.

Iterative form:

Bits	59-58	10.
Bits	57-48	-(character index).
Bits	47-36	Unused.
Bits	35-18	Index into MARGS table of current start of argument.
Bits	17-00	Index into MARGS table of start of argument.

4.3.6 MARGS - Macro Arguments.

Used to hold the character strings of macro arguments. Each character string starts in a new word and terminates with at least cne 00 character.

4.3.7 MICTAB - Micro Table.

Records the names and current definition of micros. If table is non-empty, first word is used as scratch during table lookup.

Entry = N words (N = 1 if micro value is null).

Words 1-(N-1): Value of micro, as a character string packed ten characters per word. In last word (word N-1), bits 59-06 contain 0-9 characters left adjusted with 00 fill and bits 05-00 contain character count for this word.

Word N:

Bits	59-48	2000B + N (for floating unpack).
Bits	47-00	Micro name right adjusted with 00 fill.

4.3.8 RASTAB - Remote Assembly Table.

Used to hold remote compressed text during assembly.

4.3.9 RMTAB - Remote Code Table.

Used to hold remote compressed text during assembly.

4.3.10 STACK - Recursion Stack.

Control of assembler input sources. The stack is pushed down by each macro/opdef call and DUP, ECHO, END (implied HERE), HERE, or XTEXT statement, and is pushed up when the current input source is exhausted. The maximum size of STACK is ≠LIMRECUR≠ entries.

Entry = 4 words.

Word 1:

Bits	59-54	Period in display code.
Bits	53-18	Recursion level in decimal, left justified with blank fill.
Bits	17-00	Relative address of next card to be unpacked.

Word 2:

Bits	59-56	Type of stack entry: 1 - Macro expansion. 2 - Duplication expansion. 3 - Remote expansion. 4 - XTEXT expansion. 5 - ECHO expansion.
Bits	55-36	A record of indicators set when stack was pushed down.
Bits	35-18	Length of MARGS at start of expansion.
Bits	17-00	Length of MARDIS at start of expansion.

Word 3:

Bits	59-36	Unused.
Bits	35-18	DUP - Iteration count. XTEXT - Length of LASTAB at start of definition. ECHO - Length of ECHTAB at start of definition. Macro - Relative address of first card in range of current IRP, or zero if no IRP is in progress.
Bits	17-00	DUP - Length of DUPTAB at start of definition. Macro - Relative address of MARDIS entry for current IRP parameter, or zero if no IRP is in progress.

Word 4:

Bits	59-48	Unused.
Bits	47-00	Name of macro, or words *DUP*, *ECHO*, *RMT*, or file name for XTEXT, or group name for labeled remote text. In each case, the name is in display code left justified with blank fill.

4.3.11 TEMTAB - Temporary Table.

Temporary table used to hold compressed text during definition operations.

4.4 PASS 2 TABLES

The Pass 2 tables exist only during the second pass of an assembly, and are cleared at the conclusion of Pass 2 (except REFTAB, which is used by Pass 3). All of them share pointer words (ORIGINS and SIZES entries) with Pass 1 tables.

4.4.1 COMTAB - Common Linkage Table.

Used in Pass 2 to record the common linkages in the binary output. The first word is used as scratch by DLAST. Entries in any or all of three formats may be intermixed; DLAST sorts them out.

Basic format - FILL (4200B) Table.

Bits	59-48	Zeros.
Bits	47-39	Unconditional loading: zeros. Conditional loading: same as bits 26-18.
Bits	38-30	Common block number, starting with 3 for the first common block.
Bit	29	1 for later table construction.
Bits	28-27	Address field position: 2 - Upper address (bits 47-30). 1 - Middle address (bits 32-15). 0 - Lower address (bits 17-00).
Bits	26-18	Relocation of reference address, in loader relocation: 0 - Absolute. 1 - Plus program. 3 - First common block. 4 - Second common block. Etc.
Bits 17-00		Reference address.

Extended format - XFILL (4100B) table, unconditional loading.

Bits	59-57	Zeros.
Bits	56-48	Common block number.
Bits	47-39	Relocation of reference address.
Bits	38-33	Zeros.
Bits	32-12	Reference address.
Bits	11-06	Low-order bit position of address field.
Bits	05-00	Length, in bits, of address field.

Extended format - XFILL (4100B) table, conditional loading.

Bits	59-57	001.
Bits	56-48	Relocation of reference address.
Bits	47-42	Zeros.
Bits	41-21	Reference address.
Bits	20-15	Low-order bit position of address field.
Bits	14-09	Length, in bits, of address field.
Bits	08-00	Common block number.

4.4.2 ERRTAB - Error Directory Table.

Used to record page occurrences of errors.

Entry = 1 word.

Bits	59-30	Error type (index into ERFLAGS).
Bits	29-00	Page number.

4.4.3 LNKTAB - External Linkage Table.

Contains all references to external symbols in the binary output. The first (L.EXTAB) words are used as temporary storage by DLAST; the remaining words each contain a one-word reference entry.

Entries in any or all of three formats may be intermixed; DLAST sorts them out.

Entry = 1 word.

Basic format - LINK (4400B) table.

Bits	59-48	Zeros.
Bits	47-39	Unconditional loading: zeros. Conditional loading: same as bits 26-18.
Bits	38-30	External symbol ordinal (1 for the first).
Bit	29	1 for later table construction.
Bits	28-27	Address field position: 2 - Upper address (bits 47-30). 1 - Middle address (bits 32-15). 0 - Lower address (bits 17-00).
Bits	26-18	Relocation of reference address, in loader relocation: 0 - Absolute. 1 - Plus program. 3 - First common block. 4 - Second common block. Etc.
Bits	17-00	Reference address.

Extended format - XLINK (4500B) table, unconditional loading.

Bits	59-57	Zeros.
Bits	56-48	External symbol ordinal.
Bits	47-39	Relocation of reference address.
Bits	38-33	Zeros.
Bits	32-12	Reference address.
Bits	11-06	Low-order bit position of address field.
Bits	05-00	Length, in bits, of address field.

Extended format - XLINK (4500B) table, conditional loading.

Bits	59-57	001.
Bits	56-48	Relocation of reference address.
Bits	47-42	Zeros.
Bits	41-21	Reference address.
Bits	20-15	Low-order bit position of address field.
Bits	14-09	Length, in bits, of address field.
Bits	08-00	External symbol ordinal.

4.4.4 REFTAB - Symbolic Reference Table.

Records the information required for generating the reference table at the end of assembly.

Entry = 1 word.

Bits	59-42	Index of the symbol in SYMTAB.
Bits	41-25	Location counter.
Bits	24-13	Page number.
Bits	12-06	Line number.
Bits	05-00	Usage letter.

4.4.5 RELTAB - Relocation Indicators.

For a relocatable assembly, RELTAB stores the relocation indicators for the current partial binary word for each USE block; it is parallel to USETAB and has the same number of entries. For a PPU or absolute CPU assembly, RELTAB is unused and has zero length.

Entry = 2 words, comprising four 30-bit fields.

Bit	29	External flag.
Bits	28-12	Relocation base or external number.
Bits	11-06	Low-order bit position of address field.
Bits	05-00	Length, in bits, of address field.

5.0 FILES

The files used by COMPASS for all of its input/output are described below, in alphabetic order by FET name.

FET Name	Normal Filename	Coded/ Binary	Buffer Name	Buffer Size	File Usage
B	LGO	binary	BBUF	BBUFL	Binary output file; used by Pass 1 for LCC and system text output and by Pass 2 for all other binary output. Filename set by B parameter on COMPASS control card. This file is never rewound.
C	ZZZZZRM	binary	CBUF	15000B (approx.)	COMPASS image. When the symbolic reference table (REFTAB) has overflowed to file ZZZZZRM during Pass 2, most of COMPASS is dumped to the end of file ZZZZZRM at the beginning of Pass 3 to make room for the reference table, and is restored at the conclusion of Pass 3. This file is EVICTed before and RETURNed after use.
D	SNAPPER	coded	DBUF	DBUFL	Debugging output file. When COMPASS is assembled with DEBUG = 0, the FET is omitted and the buffer size is zero. This file is never rewound.

FET Name	Normal Filename	Coded/ Binary	Buffer Name	Buffer Size	File Usage
E	OUTPUT	coded	EBUF	EBUFL	Error output file (short listing). Filename set by O parameter on COMPASS control card. The filename and buffer size are changed to zero if the long and short listings are directed to the same file. If the long listing is suppressed (L=0 on COMPASS control card), the EBUF size is changed to zero and the FET is changed to point to OBUF instead. This file is never rewound.
G	SYSTEXT	binary	SBUF	SBUFL	System text input; used only by pass 0. Filename set by G parameter on COMPASS control card. This file is rewound before reading.
I	INPUT	coded	IBUF	IBUFL	Source input file. Filename set by I parameter on COMPASS control card. If the long listing is suppressed (L=0 on COMPASS control card), all but 101B words of OBUF is added to IBUF. This file is never rewound. Reading stops at end of SCOPE logical record or section.

FET Name	Normal Filename	Coded/ Binary	Buffer Name	Buffer Size	File Usage
O	OUTPUT	coded	OBUF	1500B (approx.)	Print output file (long listing). Filename set by L parameter on COMPASS control card. If L=0 is specified, all but 101B words of OBUF is added to IBUF and the remaining 101B-word buffer is used for the short listing file instead of EBUF. This file is never rewound.
P	PATCHES	coded	BBUF	BBUFL	Debugging directive input, used only by Pass 0 and only if COMPASS is assembled with DEBUG = 1. The file is rewound before reading. Reading stops at end of SCOPE logical record.
S	ZZZZZRL	binary	SBUF	SBUFL	Overflow from intermediate text table (INTER); written by Pass 1, rewound, and read by Pass 2. This file is rewound before each use. At the end of a batch of assemblies, the file is RETURNed if it is mass storage or rewound otherwise.

FET Name	Normal Filename	Coded/ Binary	Buffer Name	Buffer Size	File Usage
R and T	ZZZZZRM	binary	RBUF and TBUF	RBUFL	Overflow from symbolic reference table (REFTAB). FET R is used for writing the file in Pass 2, and FET T is used for reading the file at the beginning of Pass 3. If the long listing is suppressed (L=0 on COMPASS control card), the RBUF size is changed to zero. This file is rewound before and after use.
X	OLDPL	binary	BBUF	BBUFL	XTEXT input file; used only by Pass 1. Filename set by X parameter on COMPASS control card or by location field of XTEXT pseudo-instruction. This is the only FET that can have different filenames during an assembly. The random access flag (bit 47 in the second word of the FET) is always set to 1. This file is rewound before each use. It is OPENed for each XTEXT card with a non-blank variable field.

Some buffers are shared by more than one file, but never at the same time. When a buffer size is changed to zero, the released memory space is added to the managed table area by adjusting LOCORE and SIZCORE.

6.0 ASSEMBLY PARAMETERS

Discussed here are the various assembly parameters (symbols and micros) used in COMPASS. Each is shown with the value defined in the system as distributed. All are listed in alphabetic order.

BBUFL = ≠BUFL≠ - Binary Buffer Length.

This is the length of BBUF, the buffer area that is used in Pass 0 for debugging directive input, in Pass 1 for XTEXT input and LCC and system text output, and in Pass 2 for binary output.

BUFL - ▽1001B▽ or ▽0▽ - Normal Buffer Length.

This is the size of most CIO buffer areas. It is initially ▽1001B▽ but is changed to ▽0▽ by COMPCOM when COMPASS is assembled for 7000 SCOPE 2 (i.e., ≠MODEL≠ ≥ 75 and CP≡RM = 1).

COMCOL = 30 - Assumed Comment Column.

This is the value stored into CCOL at the start of each pass of each assembly, which can then be changed by a COL pseudo-instruction.

CONCAT = 65B - Concatenation Mark.

This is the display code value of the character that is squeezed out by subroutine EDIT. The character prints as a right arrow (print 1) or an underscore (print 2).

CP.ABORT = ▽0▽ - Abort Option.

This is the value preset in word CP.ABORT in COMPCOM. Changing it to ▽1559▽ would make COMPASS assume the presence of the control card A option.

CP.F = ▽0▽ - Value of ⋈F Special Symbol.

This is the value preset in word CP.BATCH in COMPCOM. Changing it to ▽n▽ would make COMPASS assume the control card F=n option.

CP.LISTF = ▽1▽ - Long Listing Option.

This is the value preset in word CP.LISTF in COMPCOM. Changing it to ▽0▽ would make COMPASS assume the control card L=0 option.

CP.NAME = ▽COMP2$▽ - COMPASS Overlay Name.

This is the name of the level (1,0) overlay called by compilers for assembling COMPASS subprograms. It is defined by COMPCOM.

CP.ORG = 777B, 2777B, or 7777B - COMPASS Overlay Origin.

This is the address at which the overlay header word (5000B) for the COMPASS level (1,0) overlay is loaded when it is called. It is defined by COMPCOM; its value depends on CP≡ RM and ≠MODEL≠.

CP.OVLIB = ▽▽ - COMPASS Overlay Library Name.

This is the name of the library from which the COMPASS level (1,0) overlay is loaded. It is defined by COMPCOM. A null value represents the job's global library set and the operating system nucleus library.

CP.PAGE = ▽1S59▽ - Pagination Option.

This is the value preset in word CP.PAGE in COMPCOM. Changing it to ▽0▽ would make COMPASS assume the control card P option.

CP≡RM = 0 or 1 - Record Manager Usage.

This symbol controls whether COMPASS is assembled to use a Record Manager for assembly-time I/O. 0 = no, 1 = yes.

DBUFL = ≠BUFL≠ ⋈DEBUG - Debugging Buffer Length.

This is the length of DBUF, the buffer area that is used for debugging output. It is either 0 or 1001B depending on the value of DEBUG.

DEBUG = 0 - Debugging Option.

This symbol controls conditional assembly of the patch/snap facility used for debugging the assembler. It may be 0 to omit the feature of 1 to include it. Setting DEBUG = 1 increases the size of the assembler by about 2000B words and forces OVERLAY = 0.

EBUFL = ≠BUFL≠ - Error Buffer Length.

This is the length of EBUF, the buffer area that is used for error output (short listing) when both long and short listings are requested and they are directed to different files. In all other cases, the length of EBUF is changed to zero.

IBUFL = ≠BUFL≠ - Input Buffer Length.

This is the length of IBUF, the buffer area that is used for source input. The buffer length is increased by all but 101B words of OBUF when the long listing is suppressed. The symbol IBUFL is defined by COMPCOM.

INTMUL = 0 - Integer Multiply Usage.

This symbol controls whether COMPASS uses the hardware integer multiply instruction at assembly time. If it is 0, COMPASS uses an OPDEF for integer multiplication. If it is 1, COMPASS uses the hardware instruction, which cause slight reductions in assembly-time memory space and CPU time requirements.

LIMRECUR = ▽400▽ - Maximum Recursion Depth.

The value of this micro is the maximum number of entries in the source input push-down table (STACK). A micro is used, rather than a symbol, for convenience in generating the dayfile message that is issued when this limit is exceeded.

LISTRM = ▽-▽ - List Record Manager Routines.

This micro controls whether the assembly listing of COMPASS includes the Record Manager routines that are obtained from the Record Manager Program Library and incorporated into the assembler via XTEXT statements.

MICMARK = 64B - Micro Mark.

This is the display code value of the character that encloses a micro name that is replaced with the micro's value by subroutine EDIT. The character prints as a not equal sign (print 1) or a quotation mark (print 2).

MODEL = ▽7x▽ - CYBER 70/ Model Number.

This is the model number of the CYBER 70 series computer on which COMPASS runs. Many operating system incompatibilities are resolved by conditional assembly based on whether ≠MODEL≠ is < 75 {i.e., 6000 series} or ≥ 75 {i.e., 7000 series}.

MSTACK = 50 - Maximum Depth of Pushdown Stacks.

This is the number of entries allocated for the pushdown stack areas for the BASE, CODE, LIST, QUAL, and USE pseudo instructions.

NCARDS = 10 - Number of Cards per Statement.

This is the maximum number of cards in a COMPASS source statement; i.e., a statement is contained in an initial card followed by up to NCARDS-1 continuation cards: any subsequent continuation cards are treated as comments. This value determines the lengths of the CARD, LINE, SEQ, and SQIMAGE working storage areas.

NLITS = 100 - Number of Literals.

This is the maximum number of data value words that may be specified in any variable subfield of a DATA statement, in the entire variable field of a DIS or LIT statement, and in any literal in an address expression. It is the length of the VALUES working storage area.

NOPCT = 128 - Number of Operation Code Table Entries.

This is the base number of entries in OPTAB. It must be a power of 2.

NSYMT = 256 - Number of Symbol Table Entries.

This is the base number of entries in SYMTAB. It must be a power of 2.

OSNAME = °SCOPE° - Operating System Name.

This is the operating system name recorded in word 4 of each PRFX table written in COMPASS binary output.

OSVER = °3.4°, °1.2°, or °2.0° - Operating System Version.

This is the operating system version recorded in word 4 of each PRFX table written in COMPASS binary output.

OVERLAY = 1 - Overlay Option.

This symbol dictages whether COMPASS is assembled as one overlay or two. If OVERLAY = 0, the entire assembler is a single level (0,0) overlay; there is no level (1,0) overlay. This would be used when COMPASS is not to be called by a compiler or is to be loaded from a file rather than the system library. OVERLAY = 0 is forced when DEBUG = 1. If OVERLAY = 1, COMPASS is assembled as two overlays.

PAGESIZ = 60 - Number of Lines per Page.

≡PLRM≡ = ▽▽ - Record Manager Program Library.

This is the file name used in XTEXT statements, when ≠MODEL≠ < 75 and CP≡RM = 1, for obtaining Record Manager modules. Blank means OLDPL.

RBUFL = ≠BUFL≠ - Reference Buffer Length.

This is the length of RBUF, the buffer area that is used in Pass 2 for writing overflow from the symbolic reference table {REFTAB} to a scratch file. If the long listing is suppressed {L=0 on COMPASS control card}, the length of RBUF is changed to zero.

SBUFL = 2**≠BUFL≠ - Scratch Buffer Length.

This is the length of SBUF, the buffer area that is used in Pass 0 for reading system text from a file {G parameter on COMPASS control card}, and in Passes 1 and 2 for writing and reading overflow from the intermediate text table {INTER} to and from a scratch file.

SHIFTQ = 8 - Length of Symbol Table Hash Value.

When a symbol is to be located in the symbol table {SYMTAB}, the symbol is multiplied by a hashing constant and the high-order SHIFTQ bits to the lower half of the double-precision product comprise the hash value for that symbol. SHIFTQ is defined by the LOG2 macro such that NSYMT = 2**SHIFTQ.

SPY = 0 - Timing Analysis Option.

This controls whether COMPASS includes routines for calling the PPU program SPY to monitor the P-register at assembly time. Changing it to 1 makes COMPASS include these routines. They are executed only when the control card W option is present.

TIMEMSG = 1 - Assembly Time Message Option.

This controls whether COMPASS issues a dayfile message giving the CPU time at the end of a batch of assemblies. Changing it to 0 suppresses the message.

TLU0PSHF = 7 - Length of Operation Code Table Hash Value.

Same as SHIFTQ but for the operation code table {OPTAB}. TLU0PSHF is defined such that N0PCT = 2**TLU0PSHF.

VERSION = ▽3.≠MODLEVEL≠▽ - Assembler Version.

The value of this micro is placed in the main title line of every page of listing output produced by the assembler, to identify the version of the assembler that processed the program.