
The COMPASS operation code table contains the information that COMPASS requires for interpreting legal operation field entries for COMPASS instructions.

When assembly begins, the operation code table contains these entries:

- Pseudo instructions (except LOCAL)
- CPU symbolic instructions (chapter 8)
- CMU symbolic instructions (chapter 8)
- PPU symbolic instructions (chapter 9)
- System macro and opdef definitions

The MACRO, MACROE, and OPDEF pseudo instructions (chapter 5) cause entries to be made in this table. In addition, the programmer has the capability of creating entries through the following instructions discussed later in this chapter:

- | | |
|-------|---|
| CPOP | CPU operation |
| PPOP | PPU operation |
| OPSYN | Synonymous PPU or pseudo operation or macro |
| CPSYN | Synonymous CPU operation or opdef |

If a new entry redefines an instruction already in the table, the obsolete entry is not physically removed from the table. Instead, it is saved so that the table can be reconstructed between assemblies. COMPASS reconstructs the operation code table using all the original system macros, opdefs, pseudo instructions, and symbolic machine instructions. No programmer-created entry is preserved from assembly to assembly. The number of entries in the table is limited to 4123.

The only pseudo instructions that logically remove entries from the operation code table are PURGMAC and PURGDEF.

Entries in the operation code table are in two distinct formats permitting a logical division of the table. One type of entry permits identification of an instruction by finding a match for the contents of the operation field, thus, it provides mnemonic recognition. The other type of entry is looked at only if the search for a mnemonic operator fails to yield a match during a CPU assembly.

This type of entry provides for recognition of an instruction according to its syntax. COMPASS analyzes the statement to be interpreted, determines the syntax of the operation and variable subfields, and again searches the table.

Instructions are recognized in the mnemonic search and the information provided to the assembler for each instruction are as follows:

| | |
|--------------------------------------|---|
| Pseudo instructions | The entry contains addresses to routines that perform pass one and pass two operations. |
| PPU symbolic instructions | The entry describes the format of the instructions to be assembled. |
| Instructions described through PPOP | The entry describes the format of the instruction to be assembled. |
| Macro instructions | The entry directs the assembler to the location of the saved definition. |
| Instructions described through OPSYN | The entry is a copy of the synonymous entry. |

For a PPU assembly, a failure to find an entry for a mnemonic operator causes an operation code error. For a CPU assembly, however, if the search for the mnemonic operator does not yield a match, COMPASS searches the operation code table again for an entry with a matching syntax. Instructions recognized in the syntactical search and the information provided to the assembler for each instruction are as follows:

| | |
|--------------------------------------|--|
| CPU symbolic instructions | The entry describes the format of the CPU instruction to be assembled. |
| Instructions described through CPOP | The entry describes the format of the CPU instruction to be assembled. |
| Instructions defined through OPDEF | The entry directs the assembler to the location of the definition. |
| Instructions described through CPSYN | The entry is a copy of the synonymous instruction The action taken depends on the synonymous entry. |

If, following the second search of the operation code table, the statement still has not been identified, the assembler takes the following action:

For a PPU assembly, it generates a 24- or 32-bit instruction of which the first 12 or 16 bits are zero.

For a CPU assembly, it generates a 30-bit zero instruction.

Although OPSYN and CPSYN pseudo instructions provide a means of rendering more than one instruction synonymous, only instructions of the same type can become synonymous. The logical division of the table between the two types of entries prevents mnemonically identified instructions from being made synonymous with syntactically identified instructions.

When a MACRO, MACROE, PPOP, or OPSYN creates an entry for a mnemonic name that is already in the table for a different instruction, the new entry takes precedence over the old entry. Similarly, when a OPDEF, CPOP, or CPSYN redescribes a syntax already in the table for a different instruction, the new entry takes precedence over the old entry. As a result, the order of precedence for operation field recognition is, from highest to lowest:

1. Programmer-created entries for mnemonically identified instructions.

2. System macros, pseudo instructions, PPU symbolic machine instructions, and CMU instructions other than the IM instruction.
3. Programmer-created entries for syntactically identified instructions
4. CPU symbolic instructions and the CMU IM instruction

Example:

The following example illustrates a special case in which a macro name takes precedence over one form of a machine instruction, i. e. , the form using SB4 as an operation code.

| LOCATION | OPERATION | VARIABLE | COMMENTS |
|----------|-----------|----------|--|
| 1 | 11 | 18 | 30 |
| SB4 | MACRO | P1,P2 | DEFINE MACRO NAMED SB4 |
| | . | | |
| | . | | |
| | ENDM | | |
| | . | | |
| | SR4 | A1+ABLE | CALL TO MACRO. NOT CPU INSTRUCTION |
| | . | | |
| | SR3 | A1+ABLE | MACHINE INSTRUCTION |
| SR4 | OPSYN | NIL | DISABLES MACRO BUT DOES NOT RESTORE NORMAL USE OF SR4 AS AN OPERATION CODE. EVEN IF IT WERE REDEFINED WITH OPDEF IT WOULD NOT BE RECOGNIZED. THE MACRO FORM ALWAYS TAKES PRECEDENCE. |
| | . | | |
| | PURGMAC | SR4 | RESTORES NORMAL USE OF SR4 |

6.1 MNEMONICALLY IDENTIFIED INSTRUCTIONS

Mnemonically identified instructions include all pseudo instructions, macro instructions, and PPU symbolic instructions whether system or programmer defined. PPOP, OPSYN, NIL, and PURGMAC provide the programmer with a means of creating or removing operation code table entries that are in the mnemonically identified format.

6.1.1 PPOP — PPU OPERATION CODE

The PPOP pseudo instruction defines the operation and variable fields of a PPU symbolic machine instruction and creates an operation code table entry for the instruction. COMPASS generates an octal machine instruction of the defined format whenever the PPU instruction described by the PPOP instruction is used. If the operation code table already contains an entry for the name, the new definition takes precedence over the old during assembly of the subprogram or until it is redefined. No error is flagged. Any illegal parameter in PPOP causes COMPASS to ignore the PPOP and issue a 7-type error flag.

Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|----------|-----------|--------------------|
| name | PPOP | ctl, val, type |

name Mnemonic name, 1 through 8 characters

ctl Control of instruction assembly

| <u>ctl</u> | <u>Significance</u> |
|------------|---|
| 0 | Illegal; if used, COMPASS ignores the PPOP |
| 1 | 24-bit instruction with 12-bit address and no indexing |
| 2 | 12-bit instruction with signed relative address or absolute address (e.g., UJN) |
| 3 | 24-bit instruction with 18-bit address (e.g., LDC) |
| 4 | 12-bit instruction with 6-bit address (e.g., LDN) |
| 5 | 24-bit instruction with 12-bit address and optional indexing (e.g., LDM) |
| 6 | 12-bit instruction with signed relative address (e.g., SHN) |
| 7 | 24-bit instruction with 12-bit address and required second field (e.g., IAM) |

val An evaluable expression specifying the octal 4-digit operation code value; usually, only the two leftmost digits are significant. If the assembly base is M, the field is assumed to be octal.

type An evaluable expression specifying an integer value that COMPASS interprets as follows:

| | |
|---|--|
| 6 | Restrict the instruction being defined to the CYBER 180 Series, CYBER 170 Series, CYBER 70/Models 71, 72, 73, and 74; COMPASS sets an error flag if the instruction being defined is used in a CYBER 70/Model 76 PPU assembly. |
| 7 | Restrict the instruction being defined to the CYBER 70/Model 76; COMPASS sets an error flag if the instruction being defined is used in a CYBER 180 Series, CYBER 170 Series, CYBER 70/Models 71, 72, 73, and 74 PPU assembly. |

other or omitted

The instruction is not restricted to either machine type. If the base is M, type is assumed to be octal. If type is omitted, the comma preceding it can be omitted also.

Example:

| Code Generated | | LOCATION | OPERATION | VARIABLE | COMMENTS |
|----------------|-----------|----------|----------------|-----------|----------|
| | | I | II | 18 | 30 |
| | 0=0 | | PERIPH BASE | 0 | |
| | | | . | | |
| | | | . | | |
| | 15 | LA | FQJ | 15 | |
| | 40 | C | FQJ | 40 | |
| | | STM | PPOP | 5,5400+LA | |
| | | | . | | |
| | | | . | | |
| | | | . | | |
| 7311 | 5415 0040 | | STM | 0 | |

6.1.2 OPSYN — SYNONYMOUS MNEMONIC OPERATION

The OPSYN pseudo instruction makes a name in the location field of the OPSYN synonymous with the macro, pseudo instruction or PPU mnemonic name specified in the variable field. The size of the operation code table is the only limit to the number of instructions that can be made synonymous.

Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|-------------------|-----------|--------------------|
| name ₁ | OPSYN | name ₂ |

The name in the variable subfield must be previously defined as a standard instruction code. After an OPSYN, either name produces equivalent results. If the location field specifies a previously defined macro or operation code, the new definition takes precedence over the old without notification. Thus, a macro defined by a name that is subsequently used in an OPSYN location field is not called when the macro name is used in the operation field. The instruction actually called is the instruction named in the variable subfield of the OPSYN. On the other hand, the old macro definition is not lost and can be restored by purging the new definition with PURGMAC.

Example:

1. An operation named CALL is synonymous with RJM.

| LOCATION | OPERATION | VARIABLE | COMMENTS |
|----------|-----------|----------|---|
| I | II | 18 | 30 |
| CALL | OPSYN | RJM | |
| | . | | |
| | . | | |
| | . | | |
| | CALL | =XSUBR= | PRODUCES SAME RESULTS AS IF IT WERE AN RJM |

The assembler interprets each call to MACK as a NIL instruction. TAG is not defined because it becomes the location field symbol for NIL when the statement is assembled.

6.1.4 PURGMAC—PURGE MACROS

The PURGMAC pseudo instruction provides a means of disabling operation code entries for the named instructions for the duration of the current assembly.

Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|----------|-----------|---|
| | PURGMAC | name ₁ , name ₂ , . . . , name _n |

name_i Names of mnemonic operation codes for macro definitions, pseudo instructions, or PPU instructions.

A location field symbol if present is ignored.

6.2 SYNTACTICALLY IDENTIFIED INSTRUCTIONS

Syntactically identified instructions apply to CPU assemblies only. The CPOP and CPSYN pseudo instructions create operation code table entries for instructions that are to be identified through recognition of their syntax, rather than through the contents of the operation field only.

6.2.1 CPOP — CPU OPERATION CODE

The CPOP pseudo instruction describes the syntax of a new CPU symbolic machine instruction and creates an operation code table entry for the instruction. An instruction of the defined format is generated whenever the CPU instruction described by the CPOP instruction is used. If the operation code table already contains an entry for the instruction, the new definition takes precedence over the old during assembly of the subprogram. Any illegal parameter in CPOP causes COMPASS to ignore the CPOP and issue an error flag.

Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|----------|-----------|---------------------|
| sytx | CPOP | ctl, val, reg, type |

sytx The syntax consists of a mnemonic operator and variable field descriptors. The mnemonic operator consists of two characters. The first can be any character except blank. The second character can be a register designator: A, B, or X, in which case, the operation field of the instruction is recognized as cAn, cXn, or cBn, (c is a unique character; n is 0-7); or the second character can be any other character except blank, in which case the operation field of the instruction is recognized simply by a two-character mnemonic, such as EQ.

The variable field descriptors define the order of appearance of all registers, expressions, and subfield separators that comprise the variable field of the instruction being described. It consists of none, one, two, or three of the following 22 subfield descriptors. Q represents an expression. An r represents a register letter (A, B, or X). A comma separates two descriptors; a blank terminates the syntax.

| | |
|------------|-------------|
| void | Q |
| r | rQ |
| -r | -rQ |
| r_1+r_2 | r_1+r_2Q |
| $-r_1+r_2$ | $-r_1+r_2Q$ |
| r_1*r_2 | r_1*r_2Q |
| $-r_1*r_2$ | $-r_1*r_2Q$ |
| r_1/r_2 | r_1/r_2Q |
| $-r_1/r_2$ | $-r_1/r_2Q$ |
| r_1-r_2 | r_1-r_2Q |
| $-r_1-r_2$ | $-r_1-r_2Q$ |

For example, to describe $-X3*B1$, the descriptor, $-r_1*r_2$, would be written as $-X*B$ whereas, to describe $B2+ALPHA$, the descriptor rQ would be written as BQ .

ctl Control of instruction assembly.

| <u>ctl</u> | <u>Significance</u> |
|------------|---|
| 0 | 15-bit instruction |
| 1 | 30-bit instruction |
| 2 | 15-bit instruction, force upper before assembly |
| 3 | 30-bit instruction, force upper before assembly |
| 4 | 15 bit instruction, force upper after assembly |
| 5 | 30-bit instruction, force upper after assembly |
| 6 | 15-bit instruction, force upper before and after assembly |
| 7 | 30-bit instruction, force upper before and after assembly |

val An evaluable expression specifying a 9-bit operation code; if the base is M, val is assumed to be octal.

reg Three octal digits specifying the order from left to right into which register numbers are to be inserted into the i, j, k portions of a 15-bit instruction, or into the i and j portions of a 30-bit instruction. If the assembly base is M, reg is assumed to be octal.

- 1 Register number obtained from operation field
- 2 Number of second register or only register in variable field
- 3 Number of first two registers in variable field
- 0 Set field to 0

type An evaluable expression specifying an integer value that COMPASS interprets as follows:

6 Restrict the instruction being defined to the 6000 Series, CYBER 180 Series, CYBER 170 Series, and CYBER 70/Models 71, 72, 73, and 74; COMPASS sets an error flag if the instruction being defined is used when MACHINE 7 has been specified.

7 Restrict the instruction being defined to the 7600 or the CYBER 70/Model 76; COMPASS sets an error flag if the instruction being defined is used when MACHINE 6 has been specified.

other or omitted The instruction is not restricted to a machine type.

If base is M, type is assumed to be octal. If type is omitted, the comma preceding it can be omitted also.

Example:

| Code Generated | LOCATION | OPERATION | VARIABLE | COMMENTS |
|----------------|----------|-----------|-------------|-------------------|
| | I | II | IB | 30 |
| | SAX+B | CFOP | 0,530B,132B | DEFINES SA1 XJ+BK |
| | SXXG | CFOP | 1,720B,120B | DEFINES SXI XJ+K |
| | | . | | |
| | | . | | |
| 53731 | | SA7 | X3+B1 | |
| 722 7231000003 | IAG | SX3 | X1+3 | |

6.2.2 CPSYN — SYNONYMOUS CPU INSTRUCTION

The CPSYN pseudo instruction renders an instruction with the syntax given in the location field synonymous with the instruction having the syntax specified in the variable field. The only limit to the number of CPU instructions that can be made synonymous is the size of the operation code table (4123 entries).

Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|-------------------|-----------|--------------------|
| sytx ₁ | CPSYN | sytx ₂ |

sytx₁ Syntax of a CPU instruction (see CPOP for legal forms). If this syntax is already in the operation code table, the table entry for sytx₂ takes precedence over the old table entry for sytx₁ without notification.

sytx₂ Syntax of a CPU instruction for which there must be an entry in the operation code table. Following the CPSYN, an instruction in either sytx₁ or sytx₂ produces an octal instruction of the format described by the entry for sytx₂.

6.2.3 PURGDEF — PURGE CPU OPERATION CODE

The PURGDEF pseudo instruction provides a means of disabling syntactically-identified operation code entries for the duration of the current assembly.

Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|----------|-----------|--------------------|
| | PURGDEF | sytx |

sytx Syntax of a CPU instruction (see CPOP for legal forms).

A location field symbol, if present, is ignored.

The COMPASS micro capability enables the programmer to symbolically refer to a defined character string. When used in conjunction with IFC, DUP, STOPDUP, and SET pseudo instructions, micro strings provide for varied manipulation of character strings -- testing for a particular character, counting characters, concatenation of strings, etc.

Use of a micro definition requires two steps: definition of the character string, and substitution. In this discussion, substitution rather than definition is discussed first so that the reader has a better understanding of how a definition is used when it is described.

7.1 MICRO SUBSTITUTION

Wherever a micro name between micro marks (\neq) occurs in a statement other than a comment line (* in column 1), the assembler substitutes the micro before it interprets the statement. If column 72 of the last statement read is exceeded as a result of micro substitution, the assembler creates up to a maximum of 9 continuation statements, beyond which it discards excess characters without notification on the listing. No replacement takes place if the micro name is unknown or if one of the micro marks has been omitted. If the micro name is unknown, the assembler flags a nonfatal assembly error. If the micro name is null (that is, the two micro marks are adjacent), then

1. Both micro marks are deleted, and
2. No error flag is set

Example:

A micro identified as NAM is defined as the 7 characters:

ADDRESS

A reference to NAM is in the variable field of a line:

| | LOCATION | OPERATION | VARIABLE | COMMENTS |
|---|----------|-----------|----------------------|----------|
| 1 | | 11 | 18 | 30 |
| | LOC | SA1 | \neq NAM \neq +4 | |

However, before the line is interpreted, COMPASS substitutes the definition for NAM producing the following line:

| | LOCATION | OPERATION | VARIABLE | COMMENTS |
|---|----------|-----------|-----------|----------|
| 1 | | 11 | 18 | 30 |
| | LOC | SA1 | ADDRESS+4 | |

NOTE

Unless the A option of the LIST pseudo instruction is enabled, the listing depicts the instruction as it was before the substitution took place.

7.2 MICRO DEFINITION

Pseudo instructions specifically designed for the purpose of defining micros are: MICRO, OCTMIC and DECMIC. In addition, the following pseudo instructions optionally define micros: BASE, CODE, and QUAL. Also, system or built-in micros are automatically defined by COMPASS at the start of each subprogram assembly.

7.2.1 MICRO — DEFINE MICRO

The MICRO pseudo instruction defines a character string and assigns a name to that string.

Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|----------|-----------|----------------------|
| micname | MICRO | $n_1, n_2, dstringd$ |

| | |
|----------|---|
| micname | Name by which definition is called; 1-8 characters |
| n_1 | Absolute evaluable expression specifying starting character in string; when the base is M, COMPASS assumes that n_1 is decimal. |
| n_2 | Absolute evaluable expression specifying number of characters; when the base is M, COMPASS assumes that n_2 is decimal. |
| dstringd | Delimited character string. The delimiter d is a character not used in the string. |

Counting the first character after d as character 1, the assembler forms the string by extracting n_2 characters starting with character n_1 . If the second delimiting character occurs before count n_2 is exhausted, the defined string terminates at that point. If n_1 is greater than zero and n_2 is omitted, zero, or negative, the defined string includes all the characters from n_1 to the closing delimiter (see second example).

If n_1 is omitted, zero, or negative, the defined string is empty; no substitution takes place when the micro name is referred to. That is, n_2 and the character string are ignored.

A previously defined micro can be a part of a micro definition; one micro can be defined as a substring of another (see third example).

A micro can combine previously defined micros or can be a subset of another. Also, a micro defined originally as one character string can be redefined subsequently with a different character string. After the redefinition, the original character string is inaccessible.

If n_1 or n_2 is negative, the assembler generates a 7-type error.

Examples:

- The following MICRO defines NAME as the 19 characters beginning with A and ending with G.

| | LOCATION | OPERATION | VARIABLE | COMMENTS |
|---|----------|-----------|--------------------|----------|
| I | | 11 | 18 | 30 |
| | NAME | MICRO | 1,19,*ALPHANUMERIC | STRING* |

2. This example illustrates a blank character count. The defined string begins with A and is terminated by the closing delimiter.

| LOCATION | OPERATION | VARIABLE | COMMENTS |
|----------|-----------|--------------------------|----------|
| 1 | 11 | 18 | 30 |
| MICKY | MICRO | 1,,*ALPHANUMERIC STRING* | |

3. One micro can be defined as a substring of another.

| LOCATION | OPERATION | VARIABLE | COMMENTS |
|----------|-----------|--------------|------------------------------------|
| 1 | 11 | 18 | 30 |
| NAM1 | MICRO | 1,25,*\$AJC* | ALPHANUMERIC STRING* |
| . | . | . | |
| . | . | . | |
| . | . | . | |
| NAM2 | MICRO | 7,,*#NAM1* | SAME STRING AS IN EXAMPLES 1 AND 2 |

4. One micro can combine others.

| LOCATION | OPERATION | VARIABLE | COMMENTS |
|----------|-----------|----------------------|------------------------|
| 1 | 11 | 18 | 30 |
| NAM1 | MICRO | 1,12,\$ALPHANUMERIC* | |
| NAM2 | MICRO | 1,7,X STRING X | |
| NAM3 | MICRO | 1,,+#NAM1#X#NAM2# | COMBINES NAM1 AND NAM2 |

5. A micro name can be redefined.

| LOCATION | OPERATION | VARIABLE | COMMENTS |
|----------|-----------|--------------------------|-----------------------------------|
| 1 | 11 | 18 | 30 |
| MSG | MICRO | 1,6,*STRING* | |
| . | . | . | |
| . | . | . | |
| . | . | . | } CODE USING FIRST DEFINITION |
| MSG | MICRO | 1,10,*ALPHANUMERIC #MSG* | |
| . | . | . | |
| . | . | . | } CODE USING SECOND DEFINITION. |
| . | . | . | FIRST DEFINITION IS INACCESSIBLE. |

6. Micro substitution takes place before a line is assembled or examined for syntax. Thus, the following is possible.

| LOCATION | OPERATION | VARIABLE | COMMENTS |
|----------|-----------|------------|---------------|
| 1 | 11 | 18 | 30 |
| NAM | MICRO | 1,25,* LOC | SA1 ADDRESS** |
| . | . | . | |
| . | . | . | |
| #NAM#1 | SA1 | ADDRESS+1 | |

7.2.2 DECMIC – DECIMAL MICRO

Using a decimal conversion, the DECMIC pseudo instruction converts the expression into a character string to be saved under the name specified.

Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|----------|-----------|--------------------|
| micname | DECMIC | aexp, n |

micname Name by which definition is called; 1-8 characters

aexp Absolute evaluable expression

n Optional absolute evaluable expression specifying number of characters in the defined string. The defined string is a maximum of 10 characters regardless of the magnitude of n. When base is M, COMPASS assumes that n is decimal

If n is omitted or has a zero value, the micro contains the number of characters indicated by the conversion to a maximum of 10 characters. If the converted expression has more than n (or 10) digits, the most significant digits are truncated. If the value has fewer than n digits, the string is right justified and filled with leading zeros. All numbers are treated as positive.

Example:

B has the value 1024 decimal or 2000 octal before conversion.

| | LOCATION | OPERATION | VARIABLE | COMMENTS |
|---|----------|-----------|------------|-----------------|
| I | | II | IB | 30 |
| | V | DECMIC | B,6 | |
| | SYMBL | MICRO | 1,,*#V# | STORAGE NEEDED* |
| | SYMBL | MICRO | 1,,*001024 | STORAGE NEEDED* |

7.2.3 OCTMIC – OCTAL MICRO

Using an octal conversion, the OCTMIC pseudo instruction converts the value of the expression into a character string to be saved under the name specified.

Format:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|----------|-----------|--------------------|
| micname | OCTMIC | aexp, n |

micname Name by which definition is called; 1-8 characters

aexp Absolute evaluable expression

n Optional absolute evaluable expression specifying number of characters in the string. The defined string is a maximum of 10 characters regardless of the magnitude of n. When base is M, COMPASS assumes n as a decimal. If n is omitted or has a zero value, the micro contains the number of characters indicated by the conversion to a maximum of 10 characters.

If the converted expression has more than n (or 10) digits, the most significant digits are truncated. If the value has fewer than n digits, the string is right justified and filled with leading zeros. All numbers are treated as positive.

Example:

B has the value 1024 decimal or 2000 octal before conversion.

| LOCATION | OPERATION | VARIABLE | COMMENTS |
|----------|-----------|------------|----------------------------|
| I | II | IB | 30 |
| V1 | OCTMIC | B,6 | |
| S1 | MICRO | 1,,*#V1# | ADDITIONAL STORAGE NEEDED* |
| S1 | MICRO | 1,,*002000 | ADDITIONAL STORAGE NEEDED* |

7.3 PREDEFINED MICRO NAMES

Several standard micros are predefined by the COMPASS assembler. They are available for every assembly. The programmer simply writes the micro reference as desired.

These micros are automatically defined at the beginning of each assembly, and have the default values specified below until they are redefined by the programmer; thereafter, the programmer's definition holds until the start of the next assembly.

7.3.1 DATE

The DATE micro contains the current date in 10 characters in one of the following forms as obtained from the operating system:

Δ yr/mo/dy. or Δ mo/dy/yr.

The micro reference is #DATE#.

7.3.2 JDATE

The automatic value of the JDATE micro is five digits yyddd, where yy is the year and ddd is the day of year at the time of assembly. Thus, JDATE is the Julian date form of DATE.

The micro reference is \neq JDATE \neq .

7.3.3 TIME

The TIME micro contains the current time of day in 10 characters in the following form as obtained from the operating system:

Δ hr.min.sec.

The micro reference is \neq TIME \neq .

Example:

| LOCATION | OPERATION | VARIABLE | COMMENTS |
|----------|-----------|---|----------|
| 1 | 11 | 18 | 30 |
| | TITLE | PROGRAM ASSEMBLED ON \neq DATE \neq AT \neq TIME \neq | |

7.3.4 BASE

The automatic value of the BASE micro is a single letter D, M, or O, corresponding to the number base currently in effect (specified by the most recent BASE pseudo instruction); it is initially D.

The micro reference is \neq BASE \neq .

7.3.5 CODE

The automatic value of the CODE micro is a single letter A, D, E, O, or I, corresponding to the character code currently in effect (specified by the most recent CODE pseudo instruction); it is initially D.

The micro reference is \neq CODE \neq .

7.3.6 QUAL

The automatic value of the QUAL micro is 0 to 8 characters comprising the qualifier symbol currently in effect (specified by the most recent QUAL pseudo instruction); it is null initially and whenever the blank qualifier is in effect.

The micro reference is \neq QUAL \neq .

7.3.7 SEQUENCE

The automatic value of the SEQUENCE micro is 18 characters comprising the sequence field (columns 73-90) of the first line of the COMPASS source statement most recently read from the main source input file. Thus, if the current statement was read from the main source input file, SEQUENCE is the sequence field of the first line of the statement. However, if the current statement is generated (i. e., part of a macro call expansion, DUP expansion, etc.) or is read from a different file via the XTEXT pseudo instruction, then SEQUENCE is the sequence field of the first line of the statement most recently read from the main source input file.

The micro reference is ~~#SEQUENCE#~~.

7.3.8 MODLEVEL

The automatic value of the MODLEVEL micro is the value (up to 9 characters) specified by the ML parameter on the COMPASS control statement. If no ML parameter is present, the automatic value of the MODLEVEL micro is equal to that of the JDATE micro. When COMPASS is called by a compiler to process embedded COMPASS subprograms, the automatic value of the MODLEVEL micro is supplied by the calling compiler. The MODLEVEL micro is intended to be used when assembling a compiler (or COMPASS itself), to provide the compiler modification level to be placed in word 6 of each PRFX table in the binary output written by the compiler.

The micro reference is ~~#MODLEVEL#~~.

7.3.9 PCOMMENT

The automatic value of the PCOMMENT micro is the value specified by the PC parameter on the COMPASS control statement, with characters truncated from the right or blanks appended to the right, as necessary, so that the micro's length is exactly 30 characters. If no PC parameter is present, the automatic value of the PCOMMENT micro is 30 blanks. When COMPASS is called by a compiler to process embedded COMPASS subprograms, the automatic value of the PCOMMENT micro is supplied by the calling compiler. The PCOMMENT micro is intended to be used in a COMMENT pseudo instruction to specify words 8 through 10 of the PRFX table in the binary output. It may also be used, in conjunction with the *F special symbol, to determine compiler options (debug mode, rounded arithmetic, etc.) in effect at the time of assembly.

The micro reference is ~~#PCOMMENT#~~.

