File Number 1130/1800-21
Order Number GC26-3733-0

IBM Systems Reference Library

IBM 1130/1800 Macro Assembler Programming

This manual describes how to use the Macro Assembler of the 1130 Disk Monitor System, Version 2, and the 1800 Multiprogramming Executive Operating System, Versions 2 and 3. It should be used by programmers who have a basic knowledge of the assembler language and use this language to write programs for these systems.

This publication is supplemental to the 1130 and 1800 Assembler Language manuals and should be used in conjunction with them. This manual describes the 1130/1800 Macro Assembler: the definition and usage of the macro instruction, features of macro and assembler-language programming, and creation of a language for a specific need; the Macro Update Program; and the error messages, error flags, and warning flags of the Macro Assembler and Macro Update Program.

First Edition (June, 1970)

This edition applies to the IBM 1130 Disk Monitor System, Version 2, and the IBM 1800 Multiprogramming Executive Operating System, Versions 2 and 3, and to all subsequent versions and modifications until otherwise indicated in new editions or Technical Newsletters. Significant changes and/or additions to the specifications contained in this publication are being made from time to time; therefore, before using this publication in connnection with IBM systems, consult the latest SRL Newsletter, Order Number GN20-1130 or Order Number GN20-1800, for editions that are applicable and current.

Requests for copies of IBM publications should be made to the IBM Branch Office serving your locality.

Forms are provided at the back of this publication for the reader's comments. If the forms have been removed, comments may be addressed to IBM Corporation, Programming Publications, Department D78, Monterey and Cottle Roads, San Jose, California 95114.

Preface

This publication is a guide for assembler-language programmers of the 1130 Disk Monitor System, Version 2, or the 1800 Multiprogramming Executive Operating System, Version 3. It is supplemental to, and should be used in conjunction with, the 1130 and 1800 Assembler-Language manuals.

The first chapter, "Introduction," discusses the fundamentals of the Macro Assembler: what it is, how it operates, how much main storage it requires, how fast it performs, what types of macros there are, how to incorporate macros into your system; and, briefly, the Macro Update Program and error detection.

The second chapter, "The Macro Instruction," discusses how to define and use a macro instruction.

The third chapter, "Macro Assembler Features," discusses conditional assembly pseudo-operations, the ANOP, SET, and PURG pseudo-operations, automatic name generation, concatenation, optional remarks, indirect parameter substitution, the division operator, and the symbolic tag field. At the end of this chapter is a section on programming techniques. This section includes a sample program and programming tips. The sample program is for 1800 MPX but is also valid for the 1130 (an 1130 DM2 sample program is in Appendix A).

The fourth chapter, "Macro Assembler Language," describes how the Macro Assembler can be used to create a language for a specific purpose. The example is for the 1800 MPX system but is a general illustration that is also valid for 1130 users.

The fifth chapter, "The Macro Update Program," describes how you can set up and maintain your macro libraries through various statements. These statements may refer to whole libraries, macros within the libraries, or statements within the macros.

The sixth chapter, "Errors and Warnings," discusses the various error messages, error flags, and warning flags you may receive when using the Macro Assembler and the error messages you may receive when using the Macro Update Program.

Marginal notes have been included in this publication to allow easy reference to matter within the text.

The coding forms used in this manual are: for Macro Assembler statements, the 1130/1800 Assembler Coding Form, Order Number GX33-8000; for other statements, the General Purpose Card Punching Form, Order Number GX20-8030.

Required Reading

1130 publications:

Assembler Language manual, Order Number GC26-5927

1800 publications:

Assembler Language manual, Order Number GC26-5882

System Introduction, Order Number GC26-3718

Suggested Reading

1130 publications:

Disk Monitor System, Version 2, Programming and Operator's Guide, Order Number GC26-3717, for information on the Disk Utility Program and Macro Assembler control statements and error messages.

1800 publications:

Programmer's Guide, Order Number GC26-3720, for information on the Disk Management Program and Macro Assembler control statements.

Error Messages and Recovery Procedures manual, Order Number GC26-3727, for information on the warning flags, error codes, and error messages.

Contents

Tables

The 1130/1800 Macro Assembler

condensing sequences

The 1130/1800 Macro Assembler allows you to condense a sequence of assembler-language coding that you use over and over again into one instruction, a macro instruction.

generalizing sequences

If you use the Macro Assembler, you can generalize a sequence of coding and then modify it slightly each time it is used. You code the sequence only once, defining macro parameters that cause the appropriate code to be generated for a particular use. The exact code generated when the macro instruction is used is based on the conditional assembly, automatic name generation, and/or parameter substitution facilities of the Macro Assembler.

creating a language

The Macro Assembler also allows you to define a language that is unique to your application; such a language may be simple and/or meaningful enough to be used by a person other than a professional programmer.

THE MACRO INSTRUCTION

A macro instruction, or macro, is a source program statement. When the Macro Assembler encounters a macro, it expands the macro by processing a sequence of assembler-language statements. This sequence must have been defined in a macro definition before it can be used.

defining a macro

When you define a macro, you specify its operation code (macro name), its parameters, and the sequence of assembler-language statements to be processed when the Macro Assembler encounters the macro name in a source program.

types

Using the Macro Assembler, you can define two kinds of macros, temporary macros and stored macros. You should define every macro as a temporary macro until you are sure that it will execute properly. If it does execute properly, and you want to store it, you can include it in a macro library.

TEMPORARY MACROS

use of temporary macros

A temporary macro can be used only during the assembly of the program in which it is defined. This kind of macro isn't saved by the system; if you want to use it in another program, you have to define it again during assembly of that program. You do not have to define stored macros in order to use temporary macros.

N2

If temporary macros are to be defined, you will need an *OVERFLOW SECTORS control statement. Two new parameters, N2 and N3, have been added to this statement. The N1 parameter remains the same. N2 is the number of sectors you allocated for the overflow of macro parameters from main storage to

disk. This parameter can be zero, and space will not be required if the overflow from a macro that is defined or called with another macro definition never exceeds 100 words. The required size of N2 may be estimated by using the following formula:

estimating

$$\text{Number of words} = 3+N+\sum_{i=1}^{N} 1/2\,(m_i+1)$$

N3

N is the number of parameters and m_i is the number of characters per parameter. For example, the call EXPND ALPHA,BETA,C would be computed as 3+3+1/2(5+1)+1/2(4+1)+1/2(1+1)=12 words; the remainders of individual terms are ignored. N3 is the number of sectors you allocated for temporary macro definitions. You can estimate the number of sectors needed by dividing the total number of statements in all macro definitions within the assembly by 40. If you want to retain remarks, you may have to increase N3 to accommodate them. For further information on the *OVERFLOW SECTORS statement, see the 1130 Programming and Operator's Guide, Order Number GC26-3717, or the 1800 Programmer's Guide, Order Number GC26-3720.

estimating

STORED MACROS

use of stored macros

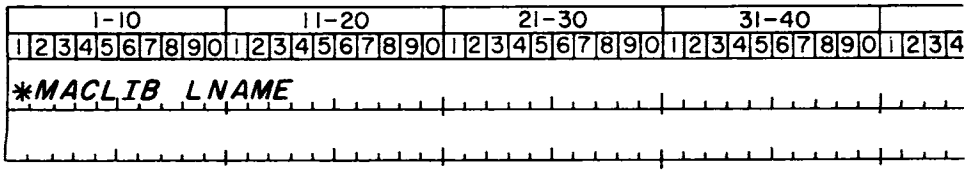
A stored macro, on the other hand, resides on disk in a macro library and can be used by any program. When you assemble a program that uses or defines stored macros, you must specify which macro library you wish to use during the assembly with the *MACLIB Macro Assembler control statement.

The macro library specified on the *MACLIB statement must be defined and initialized before it can be used by the Macro Assembler. For details on how to initialize a macro library, refer to the discussion on initializing disk space in the chapter "The Macro Update Program." The *MACLIB statement enables you to access one library selectively without having to access all of the stored macros. Multiple macro libraries may be accessed in one assembly if you logically concatenate the libraries before the assembly. Additional information on macro libraries may be found in the chapter on the Macro Update Program.

macro libraries

use an *MACLIB statement

You cannot define a stored macro within an assembly unless you have used an *MACLIB statement. If your program attempts to call a stored macro which cannot be found in the specified library, the macro call will be flagged as an illegal operation. The format of the *MACLIB is as follows:



LNAME is the name of the macro library to be used in the current assembly. For further information on the *MACLIB statement, see the 1130 Programming and Operator's Guide, Order Number GC26-3717, or the 1800 Programmer's Guide, Order Number GC26-3720.

PSEUDO-OPERATIONS

definition

Pseudo-operations, or pseudo-ops, extend the capabilities of the assembly process. Pseudo-ops are written like assembler-language statements, but they are used to provide information to the Macro Assembler rather than to generate executable code for the program. They can appear anywhere within an assembler-language program, providing you follow the other rules for their use.

Several new pseudo-ops have been included for use in programs that use the Macro Assembler. These are described in detail under "Features of the Macro Assembler" and perform the following functions:

- Defining the beginning and the end of a macro definition.
- Determining during an assembly whether certain statements are to be processed, based on a specific condition.

uses

- Permitting a program label to be set to two or more different values at different points in a program without a multiple-definition error condition.
- Logically removing a macro from a library.

NESTED MACRO DEFINITIONS AND CALLS

A macro can be defined or called within another macro definition. This process is called nesting. The nesting of macro definitions is limited only by the physical size of your system. You may want to nest definitions to allow the dynamic definition (with decisions) of an inner macro when you are expanding the outer macro. This can also be used to advantage to conserve library space if the same code is being used in both macros (the inner one can be called only by the outer one).

The nesting of macro calls is limited to 20 calls per nest. You may wish to nest calls to macros so that values from one macro can be passed to the other macro.

MAIN-STORAGE REQUIREMENTS

THE 1130

The 1130

In the 1130 Disk Monitor System, Version 2, the Macro Assembler functions as and is fully compatible with the 1130 Disk Monitor System, Version 2, Assembler. However, the Macro Assembler requires 8K words of main storage for the macro capability. The symbol table in main storage has been reduced in size to accommodate approximately 750 words for the resident macro processor. Assemblies that almost

exhaust the symbol table area of the Assembler will probably require symbol table overflow sectors for successful assembly using the Macro Assembler (see the 1130 DM2 Programming and Operator's Guide, Order Number GC26-3717). The pseudo-operations SET, ANOP, AIF, and AGO (discussed later) are also available to the 4K 1130 user.

THE 1800

The 1800

The 1800 MPX Macro Assembler requires a VCORE of 5140 words. It occupies no more main storage than, and is fully compatible with, the 1800 MPX Version 1 Assembler.

MACRO ASSEMBLER PERFORMANCE

If you have a program that has been assembled successfully under the 1130 Disk Monitor Version 2 Assembler or the 1800 MPX Version 1 Assembler, the performance of the Macro Assembler will be approximately equal to that of the earlier Assemblers. Assembly time will be greater for new programs if the Macro Assembler encounters either a macro definition, a macro call, or an invalid op code. In the case of an invalid op code, the Macro Processor of the Macro Assembler will search the temporary macros and/or the specified macro library to determine whether the unrecognized op code is the name of a macro.

Macro Update Program

The Macro Update Program assists you in initializing and maintaining macro libraries. Using this program, you can add or delete macros from your library, alter those that are already in your library, physically or logically join two libraries, and perform other functions necessary to maintaining macro libraries.

Error Messages, Error Flags, and Warning Flag

During the assembly process, the Macro Assembler checks for source program errors. If an error is detected, an error flag or an error code and message are printed. If a questionable instruction is encountered, it is flagged with the warning flag (Q). At the end of each assembly, the Macro Assembler prints a message to indicate the number of errors and warnings it encountered during that assembly. Any errors in the Macro Update Program are detected by the Disk Utility Program (DUP) for the 1130 or the Disk Management Program (DMP) for the 1800.

Defining a Macro Instruction

A macro definition can appear any time after the Macro Assembler has completed processing the control statements. The definition must appear before the first call to the macro.

MAC, SMAC and MEND

The first statement in a macro definition in an assembly must be the MAC (for a temporary macro) or the SMAC (for a stored macro) statement. The last statement of a macro definition must be the MEND statement. The pseudo-op names MAC, SMAC, and MEND must appear in the op code fields of the statements.

No label and no operand are required on these statements. Column 35 of the MAC and SMAC statements can be used to specify that remarks are retained so that they can be printed on listings (see "Optional Remarks"). Note that if you exercise this option, you will need additional disk space to accommodate these remarks. Comment statements (* in column 21) within the definition are always retained and listed within the expansion.

THE DEFINITION PROTOTYPE STATEMENT

The statement immediately following the MAC or SMAC statement is called the definition prototype statement. This statement contains the macro name in its op code field.

macro name

The macro name may be from one to five characters long. You cannot use a period, a comma, and a left or right parenthesis in macro names. A macro name can contain embedded blanks or can consist of all blanks. An invalid macro name is flagged as an illegal op code.

If you define a stored macro with a name that is already in the library named in the *MACLIB statement, it will be flagged as an invalid macro name. A temporary macro of the same name will not be flagged; it will be expanded when the macro is called since the temporary macros are searched before the stored macros. An apostrophe should not be used in the name of a stored macro because once this macro has been stored, it cannot be modified or removed from your library.

The label and operand fields of the definition prototype statement contain the names of parameters which are supplied when the macro is used. A parameter name can be any valid assembler-language symbol; an invalid parameter name is flagged as an op code error. You may use an op code as a

parameter name

parameter, but it must be done carefully because substitution will occur for all uses of the op code. Parameter names in the operand field are separated by commas. Do not include a blank within a parameter name or between parameters, as a blank terminates the parameter list. If the label field is blank, it is ignored when the macro is used. The definition prototype statement must

consist of one source statement; no continuation is allowed. A maximum of 20 parameter names is allowed.

definition

The text of the macro follows the definition prototype statement and is a sequence of assembler-language instructions, calls to other macros, and/or pseudo-ops. The statements in the text may contain the parameter names specified in the definition prototype statement. During assembly, the parameters specified in the call to the macro are substituted positionally for the corresponding parameter names in the text statements (see "An Example of Macro Definition"). Whenever the Macro Assembler prints macro definition prototype statements, five-digit decimal sequence numbers are printed to the left of each statement.

obtaining a listing

The list control pseudo-operations (see the 1130 Assembler Language manual, Order Number GC26-5927, or the 1800 Assembler Language manual, Order Number GC26-5882) can be used within a macro definition to control the listing of a macro call. If you use an 1800, you may inhibit printing by turning sense switch 2 to the ON position.

parameter substitution

You can substitute a parameter into any field or subfield of a text statement in the macro definition. A parameter substituted into the operand field of a text statement may be any valid assembler-language expression. The number of characters in the parameter name has no relationship to the number of characters actually substituted, except in the case of format and tag fields. Both the parameter name and the parameter substituted must be exactly one character long in order for the parameter to be substituted correctly into the format or tag field. Note that the number of characters in a parameter on a call may also be significant. For example, an increase in the number of characters caused by the substitution of longer parameters during the expansion may cause the operand field (columns 35-71) to be exceeded. Any information beyond column 71 will be ignored.

special characters

The slash, comma, period, plus sign, minus sign, and asterisk retain their usual meaning. When a blank occurs in an operand field (except where permitted by the assembler language), the rest of the operand field is ignored. Special characters used in Macro Assembler statements must conform to the character code summaries as listed in the 1130 Assembler Language manual, Order Number GC26-5927, or the 1800 Assembler Language manual, Order Number GC26-5882.

AN EXAMPLE OF MACRO DEFINITION

The temporary macro SUM is completely defined by the following sequence. A definition of the macro SUM is shown below and the call and statements generated by the call are shown in the next section.

macro definition

```
Label          Operation      F  T      Operands & Remarks
21        25   27      30     32 33  35   40   45   50   55
             MAC
NAME         SUM                      X,,COUNT,LIST,STOR
             LDX                 X    -COUNT
             SLA                      16
NAME         A                   L X  LIST+COUNT
             STO                      STOR
             MDX                 X    1
             MDX                      NAME
             MEND
```

Using a Macro Instruction

calling a macro

After a macro has been defined, you call it by using its name as an op code and specifying in the label and/or operand field the parameters to be substituted for the parameter names in the definition prototype statement. The parameter names must follow the order outlined under "The Definition Prototype Statement." If a parameter name is omitted (two commas in a row or a trailing comma), it is ignored when the macro is used.

When the Macro Assembler encounters a macro instruction, it processes the statements in the macro definition text with the parameters you have specified for substitution.

The assembled instructions are listed along with the macro call in the assembly listing. Any statement within a program that is a result of a macro expansion is flagged with a plus sign to the left of the label field of the Macro Assembler listing.

A sample definition of the macro SUM was illustrated in the previous section. Below is an example of a call to that macro, and the code generated by that call.

macro call

code generated

```
Label          Operation      F  T      Operands & Remarks
21        25   27      30     32 33  35   40   45   50   55
LOOP         SUM                      2,,10,FROM,TEMP
             LDX                 2    -10
             SLA                      16
LOOP         A                   L 2  FROM+10
             STO                      TEMP
             MDX                 2    1
             MDX                      LOOP
```

label field parameters

Parameters specified in the call to SUM are substituted for the dummy parameters of the SUM definition prototype statement. LOOP is substituted for NAME, 2 for X, 10 for COUNT, FROM for LIST, and TEMP for STOR. There's nothing special about the parameter in the label field of the call; it's substituted just like any other parameter. In fact, it need not even be used as a label in a macro definition; it can be used as an op code or an operand. For example, the macro DIVBY could be defined as follows (note that a while A appears in the label field of the definition prototype statement, it is used as an operand in the text of the macro):

```
Label   Operation  F  T    Operands & Remarks
21   25 27    30  32 33   35   40   45   50   55
*
*       'DIVIDE BY' MACRO
*       A IS DIVIDED BY B
*       A AND B ARE SINGLE WORD VARIABLES
*
        MAC
A       DIVBY             B
        LD                A
        SRT               16
        D                 B
        MEND
```

omitting parameters

When you call a macro, you may want to leave out one or more of the parameters. If the parameter to be left out would ordinarily be in the label field of the call, just leave it out. If the parameter would ordinarily appear followed by a comma in the operand field, leave out the character string to be substituted, but include the comma that would ordinarily follow the parameter. If the parameter ordinarily comes at the end of the operand field, you may leave it out. If you leave out one or more parameters that ordinarily come at the end, you may leave out all the commas that immediately precede them. If you anticipate omitting parameters that will be parts of AIF or AIFB expressions, refer to the section "Unspecified Parameter Checking" for restrictions on this operation.

The following examples illustrate the omission of parameters.

macro definition

Label (21–25)	Operation (27–30)	F (32)	T (33)	Operands & Remarks (35–55)
	MAC			
	DCS			A,B,C
	DC			A
	DC			B
	DC			C
	MEND			

The macro DCS generates DC statements. For example:

macro call

Label	Operation	F	T	Operands & Remarks
	DCS			1,2,3

would generate

code generated

Label	Operation	F	T	Operands & Remarks
	DC			1
	DC			2
	DC			3

Omitting parameter B in a call

macro call

Label	Operation	F	T	Operands & Remarks
	DCS			6,,1ØØØ

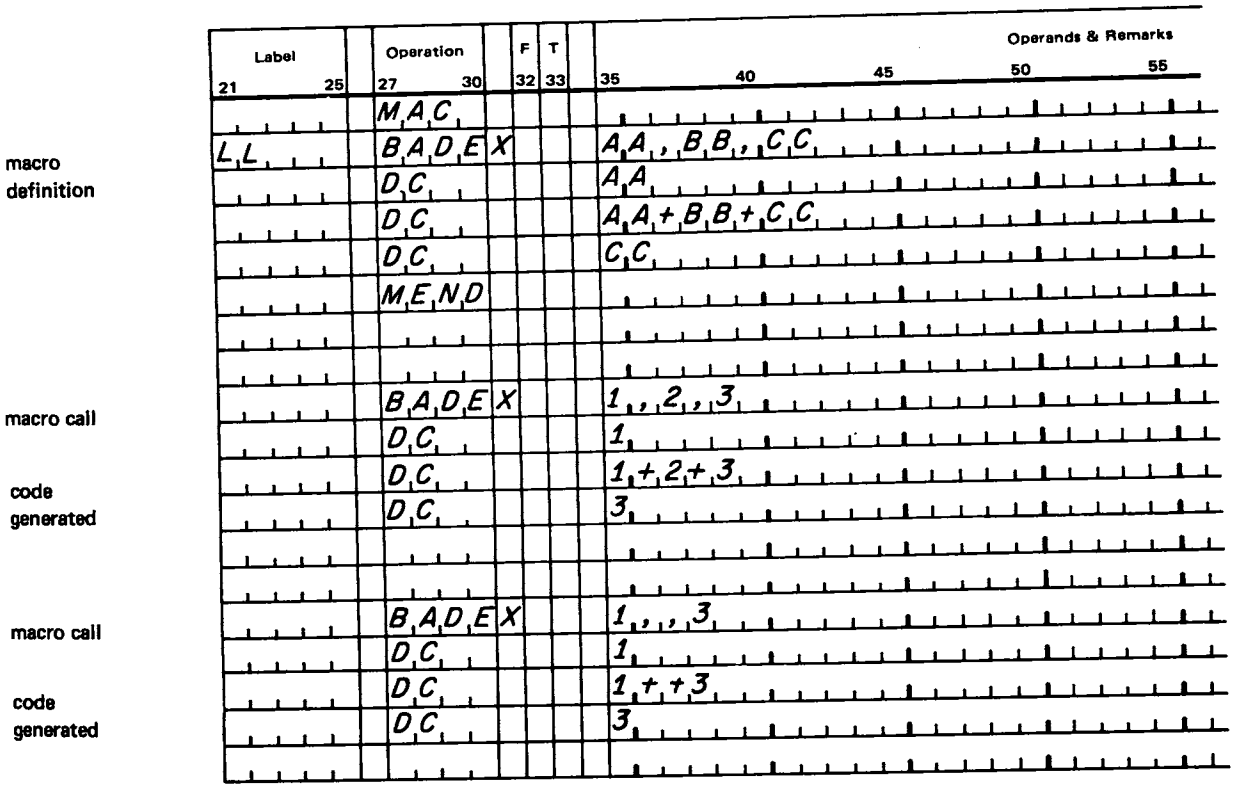
would cause

code generated

Label	Operation	F	T	Operands & Remarks
	DC			6
	DC			
	DC			1ØØØ

to be generated. Note that the operand of the second DC statement is blank because its associated parameter was omitted.

You should be extremely careful when omitting parameters because nothing is substituted for the parameter that is missing. Consider the following example; in the first call, the parameter is not missing and the code is generated properly. In the second call, the second parameter is missing and the generated code is in error.



Substituting a Character String for a Parameter

You can substitute a character string containing embedded blanks and special characters (such as commas, periods, or slashes) for a macro parameter name by enclosing the string in parentheses. This makes it easy to pass a set of parameters to a nested macro call. However, you must be careful in passing character strings to ensure that parameters do not exceed the record length, as any information beyond column 71 will be ignored.

character string substitution

Assume in the example given below that SEE is a previously defined macro with a maximum of three parameters specified in the operand field. If another macro SCAN calls SEE, parameters may be passed to SEE as shown:

macro definition

```
Label    Operation  F  T  Operands & Remarks
21   25  27    30   32 33 35   40   45   50   55
*        'SCAN'           MACRO DEFINITION
         MAC
         SCAN             D,H
         LD               D
         SEE              H
         MEND
*
```

macro call

```
         SCAN             TABLE,(VAL1,,VAL3)
*
```

code generated

```
         LD               TABLE
         SEE              VAL1,,VAL3
```

limits

Without this facility the same SCAN macro would have to be defined as follows in order to generate the above statement sequence. Notice that this second method may be restrictive in passing parameters, because a definition prototype statement may have no more than 20 parameters.

```
Label    Operation  F  T  Operands & Remarks
21   25  27    30   32 33 35   40   45   50   55
*        'SCAN'          MACRO DEFINITION
         MAC
         SCAN             D,H,I,J
         LD               D
         SEE              H,I,J
         MEND
```

in message generation

You can also use character-string substitution to generate messages. An example of this is given below.

	Label 21–25	Operation 27–30	31	F 32	T 33	34	Operands & Remarks 35 …
	*	'MSG	'		M	E	SSAGE MACRO
	*	T IS		T	H	E	CHARACTER CODE TYPE
	*	TEXT		I	S		THE MESSAGE TEXT
		SMAC					
macro definition		MSG					T,,TEXT
		DMES			T		'S'
		DMES			T		TEXT'E
		MEND					
	*						
macro call		MSG					1,,(INVALID CONTROL STATEMENT)
code generated		DMES			1		'S'
		DMES			1		INVALID CONTROL STATEMENT'E

Continuing Calls to Additional Records

A call to a macro may be too long to fit on one record, so the Macro Assembler allows calls to be continued onto one or more additional records. When you call a macro, you indicate that there is another record in the call by placing any character other than a blank in the format field (column 32). You can then continue the macro call beginning in column 35 of the next record. The first 31 columns of the next record are ignored; column 32, if used, specifies continuation of the call onto still another record.

place a character in col. 32

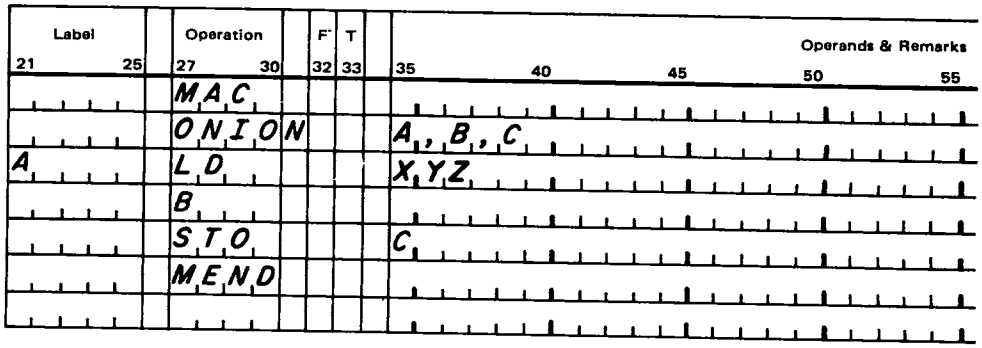
Note that on continued calls, you must be careful in substituting parameters that have more characters than those in the original definition, because you may cause the CALL record or operand field to be exceeded on either the CALL record or any subsequent continuation records. Therefore, some of your values may be lost (see "Macro Parameter Substitution" in the section on programming techniques).

When a macro call is continued onto an additional record, the Macro Assembler will stop its parameter scan of the first record when either there is a blank (not within parentheses) in the operand field or the operand extends to and includes column 71. The next character in your parameter list is assumed to be in column 35 of the next record.

replacing consecutive fields

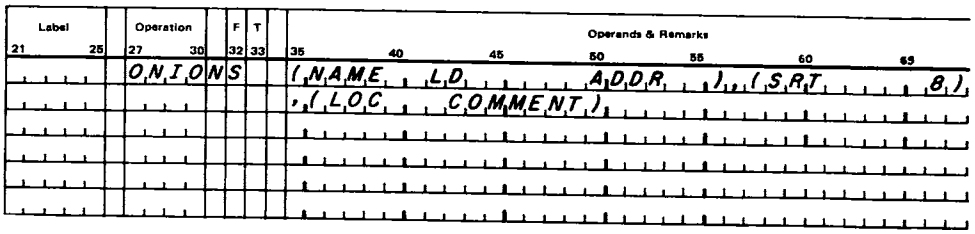
Using parentheses and continuation records, you can replace any number of consecutive fields in a macro definition statement. Look at this example:

macro definition



Here's a call to the ONION macro:

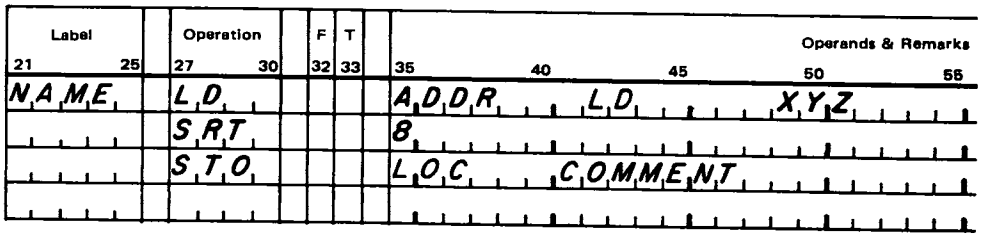
macro call



A punch other than a blank (the S in the above example) in the format field means that the macro call is continued on a second record.

The following code is generated from the above macro call.

code generated



In the macro call, the instruction NAME LD ADDR replaces the label. The remaining fields, LD XYZ, are displaced to the right. The Macro Assembler treats these fields as remarks because in this example there are two blanks after ADDR.

This page intentionally left blank.

Conditional Assembly Pseudo-Operations

definition

You may want different calls to the same macro to produce different lines of assembled code, depending on some condition to be examined during the assembly. Conditional assembly pseudo-ops allow you to do this. These pseudo-operations do not generate any executable code and do not modify the address counter.

Applications which require slight code modifications to a general technique need be coded only once using conditional assembly pseudo-ops within macro calls. This saves time for the programmer.

AIF, AIFB PSEUDO-OPS

Two conditional-assembly pseudo-ops, the "assemble if" and "assemble if back" pseudo-ops AIF and AIFB, have the following format:

format

- An optional label.
- The op code (AIF, AIFB).
- In the operand field, a left parenthesis, an expression, one or more blanks, a condition, one or more blanks, another expression, a right parenthesis, a comma, and a name.

The two expressions can be any valid assembly expressions. The name should be a valid assembler-language symbol or may be left blank. It may also be any combination of from one to five characters if this combination is used in the label field of one of these pseudo-ops: AIF, AIFB, AGO, AGOB, ANOP, PURG, LIST, EJCT, HDNG, MEND, END, or SPAC. All symbols used within AIF or AIFB statement expressions must have been predefined or the statement will be flagged with a U (undefined symbol). If the name is left blank, the statement will be flagged with a warning flag (Q).

The condition must be one of the following:

conditions

- EQ-Equal to
- GT-Greater than
- LT-Less than
- NE-Not equal to
- GE-Greater than or equal to
- LE-Less than or equal to

AIF function

During assembly, the condition statement between the parentheses is evaluated. If it is true, the AIF statement causes all the following statements to be skipped (and not processed) until the Macro Assembler finds a statement with a label corresponding to the symbol specified in the AIF statement. If the statement between the parentheses is false, the assembly continues with the statement immediately

following the AIF statement. The AIF statement may be used anywhere in a assembler-language program.

AIFB function

The AIFB (AIF back) statement functions as the AIF statement, except that the Macro Assembler returns to the beginning of the current (innermost within a nest) macro definition being expanded (called) before searching for a label. Unlike the AIF statement, the AIFB statement may occur within a macro definition only; it is flagged as an illegal op code if it appears outside of a macro definition. If the search is unsuccessful, the MEND statement will terminate the search and the expansion of that macro.

if symbol subfield is blank

If the name subfield of the AIF or AIFB operand is left blank and the label search is to be performed, statements are skipped until the first statement with no label is encountered, at which time assembly continues. In any case, when a label search is performed, the search can continue until an END statement is encountered; the END statement will be processed and flagged with a Q. If the AIF statement is in a definition, a MEND statement will terminate the search.

Let's look at an example of the use of AIF.

AIF example

Label (21–25)	Operation (27–30)	F (32)	T (33)	Operands & Remarks (35–55)
	AIF			(X+1 EQ Y),SPOOK
	LD	L		PLACE
	STO	L		SPOT
SPOOK	LD			GHOST
	STO			WITCH
	AIF			(X+1 EQ Y)
HOPE	MDX		1	1
WHEE	MDX		2	1
	MDX			SPOOK

The assembly of this code depends on the values of X+1 and Y when the Macro Assembler evaluates them. If they are not equal, all the instructions shown above are processed. If they are equal, only the following three instructions are assembled.

results

Label (21–25)	Operation (27–30)	F (32)	T (33)	Operands & Remarks (35–55)
SPOOK	LD			GHOST
	STO			WITCH
	MDX			SPOOK

Notice that the second AIF has a blank name field. As a result, the first instruction following this statement with a blank label field is assembled.

AGO, AGOB PSEUDO-OPS

Two other instructions are used along with AIF and AIFB to effect conditional assembly. These pseudo-ops, AGO and AGOB, cause unconditional branching and have the following format:

format

- An optional label.
- The op code (AGO, AGOB).
- A valid assembler-language symbol or five blanks in the operand field. If the name field is left blank, the statement will be flagged with a warning flag (Q).

The name should be a valid assembler-language symbol, or may be left blank. It may also be any combination of from one to five characters if this combination is used in the label field of one of these pseudo-ops: AIF, AIFB, AGO, AGOB, ANOP, PURG, LIST, EJCT, HDNG, MEND, END, or SPAC.

AGO function

The AGO statement causes the Macro Assembler to skip (and not process) statements following the AGO statement until it encounters a statement with a label corresponding to the symbol specified in the AGO statement. See the ANOP section of this manual for an example of the use of the AGO instruction. The AGO statement may be used anywhere in an assembler-language program.

AGOB function

The AGOB (AGO back) statement functions as the AGO statement, except that the Macro Assembler returns to the beginning of the current (innermost within a nest) macro definition being expanded before performing the label search. If the search is unsuccessful, the MEND statement will terminate the search. Unlike the AGO statement, the AGOB statement may occur within a macro definition only; it is flagged as an illegal op code if it appears outside of a macro definition.

if symbol subfield left blank

If the operand of the AGO or AGOB statement is left blank and the label search is to be performed, statements are skipped until the first statement with no label is encountered. In any case, when a label search is performed, the search can continue until an END statement is encountered. Like the AGOB statement, if the AGO statement is in a definition, a MEND statement will terminate the search.

UNSPECIFIED PARAMETER CHECKING

The name searching technique used by the AIF and AGO pseudo-ops may be utilized in checking for unspecified parameters.

Assume that the COUNT parameter on the following prototype statement is a count of how many data words are to be moved from one area to another.

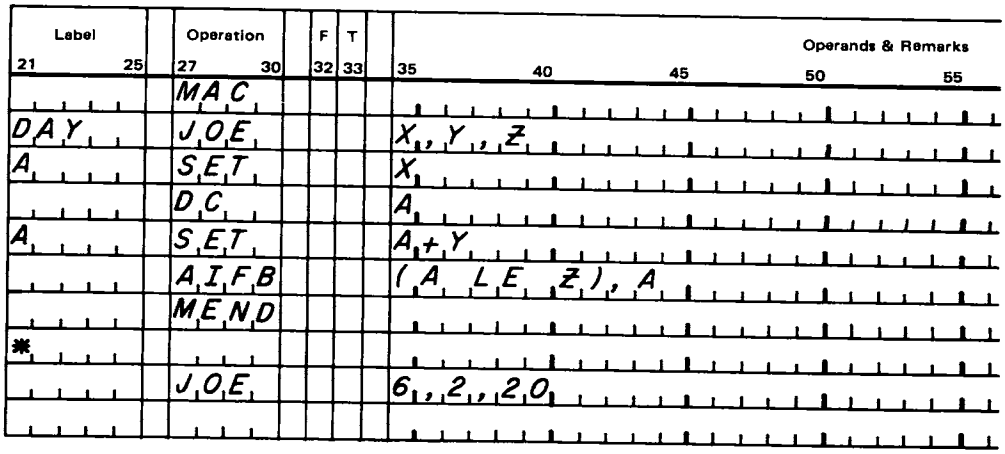
Label	Operation	F	T	Operands & Remarks
21 25	27 30	32	33	35 40 45 50 55
	MAC			
	MOVE			FROM,TO,COUNT
	AGO			COUNT
	ANOP			
X	SET			1
	AGO			BEGIN
COUNT	ANOP			
X	SET			COUNT
BEGIN	ANOP			
	.			
	.			
	.			
	MEND			

If COUNT is not specified in a call to MOVE, the name search prompted by the AGO COUNT statement will be terminated on the ANOP statement that follows immediately, because a blank was substituted for COUNT and the ANOP has a blank label field.

If COUNT is specified, the COUNT that is a label on an ANOP statement will be replaced with the COUNT specified in the call. Thus, the name search prompted by the AGO COUNT statement will terminate on the ANOP statement that has COUNT as a label.

SPECIAL CONSIDERATIONS USING AIFB AND AGOB

Note that if the AIFB or the AGOB causes a second assembly of the same code, multiple label definition errors may occur. It is your responsibility to ensure that the label to be skipped to is either unique or not entered in the symbol table, that is, a label on an AIF, AIFB, AGO, AGOB, SPAC, EJCT, HDNG, LIST, MEND, END, PURG, or ANOP statement. Also note that with the capability of the AIFB and AGOB, you can put the Macro Assembler into a loop. This will occur if the conditions never get changed, thus causing the Macro Assembler to loop between the AIFB and AGOB statements. The call below will cause the AIFB expression to be evaluated always as true (8 LE 20) because the AIFB to A will cause a branch to the first statement labeled A within the macro. Thus, the Macro Assembler will loop interminably between "A SET X" and the AIFB statement.



ANOP Pseudo-Operation

purpose

The purpose of the ANOP pseudo-op is to provide a label which an AIF, AGO, AIFB, or AGOB can reference to resume assembling. Assembling an ANOP label has the same effect as assembling the instruction immediately following it. The label on an ANOP is not placed in the symbol table, so statements other than AIF, AGO, AIFB, and AGOB can't use it as a reference. This is also true of other labels as discussed previously under "Special Considerations Using AIFB and AGOB."

format

The format of the ANOP statement is:

- A label.
- The op code ANOP.

The ANOP pseudo-operation allows you to associate temporary and permanent labels with the same instruction. Thus, the temporary label can be used to clarify a conditional assembly sequence while the permanent label can be used to clarify the instruction sequence.

The following is an example of a way in which ANOP might be used. In this example, A is assumed to have been defined prior to the AIF statement.

ANOP example

```
Label          Operation      F  T   Operands & Remarks
21      25     27      30     32 33  35   40   45   50   55
               AIF            |  |   (A,LT,Ø),SKIP1
SET Ø1         LD             |  |   INTØ1
               AGO            |  |   SKIP2
SKIP1          ANOP           |  |
SETØ1          LD             |  |   INTØ2
SKIP2          ANOP           |  |
               STO            |  |   SWTCH
```

When A is less than 0, the generated code is:

```
SETØ1          LD             |  |   INTØ2
               STO            |  |   SWTCH
```

When A is greater than or equal to 0, the generated code is:

```
SETØ1          LD             |  |   INTØ1
               STO            |  |   SWTCH
```

ANOP is useful when you're using the SET pseudo-op. An example of this usage is given in the "SET Pseudo-Operation" section.

SET Pseudo-Operation

purpose

SET allows you to assign a value to a symbol and, later in the assembly, assign another value to the same symbol without a multiple label definition error resulting. The symbol retains the value of the last SET statement associated with it from the first pass of the Macro Assembler until the Macro Assembler encounters an associated SET in the second pass. You can't use the EQU statement this way because the EQU statement is not processed on the second pass of the Macro Assembler and, consequently, cannot be used to change the value of a symbol during the assembly.

format

The format of the SET statement is:

- A label.
- The op code SET.
- A valid assembler-language expression in the operand field.

The label is set equal to the value of the expression in the operand field. Any symbols used within the expression on a SET statement must have been predefined, or the statement will be flagged with a U (undefined symbol).

Here's an example of the use of SET. Suppose A is the starting address of some data to be sent to disk, and B is the address of the end of the data. Assume we know the data will take up no more than two sectors and we want to set SECT equal to the number of sectors. The Macro Assembler automatically calculates the value of SECT in the following statement sequence.

SET example

```
Label      Operation   F T  Operands & Remarks
21    25   27    30    32 33 35  40  45  50  55
N          EQU         |  |  |  B-A
K          SET         |  |  |  N-320
           AIF         |  |  |  (K LE 0),ONE
SECT       SET         |  |  |  2
           AGO         |  |  |  OK
ONE        ANOP        |  |  |
SECT       SET         |  |  |  1
OK         (next instruction)
```

In the above example, N, the difference between B and A, is compared to 320 by the AIF instruction. If the difference is greater than 320, the first SET following the AIF statement sets SECT to 2. The AGO then causes the assembly to continue around the next SET and the assembly proceeds.

If the difference (N) is less than or equal to 320, AIF causes the assembly to continue at ONE. This is equivalent to a continuation at the second SET following the AIF statement, since ONE is an ANOP instruction. Notice that it is impossible to branch directly to the correct SET instruction, since two SET instructions in the sequence contain the label SECT. If the AIF statement specified a branch to SECT, the Macro Assembler would continue processing with the next statement having SECT in its label field--in this case, the wrong instruction.

Here's another example of a macro that uses SET.

```
Label      Operation   F T  Operands & Remarks
21    25   27    30    32 33 35  40  45  50  55
           MAC         |  |  |
           TAB         |  |  |  A,B,C
X          SET         |  |  |  A
NAME       AIF         |  |  |  (X GE C),ON
           DC          |  |  |  X
X          SET         |  |  |  X*B
           AGOB        |  |  |  NAME
ON         MEND        |  |  |
```

Given variables A and B, the TAB macro defines a constant equal to A times B. It next defines a constant equal to this product times B. It continues this way until the result reaches a specified value, C. Note that if A and B are equal, TAB builds a table of powers of B.

PURG Pseudo-Operation

purpose

The PURG pseudo-operation removes the specified macro name from the macro library associated with the assembly by the *MACLIB control statement. PURG causes operations to occur only on the library associated with the *MACLIB control statement; it does not affect any other library even if it has been concatenated to the associated library (see MUP section on "Joining Macro Libraries Logically").

You can then define another macro with the same name, but the space occupied by the purged macro isn't available for reuse until the next DMP/DUP macro update job is performed on the library (see "Macro Update Program"). The space is reclaimed by any macro update function run on that library.

format

The format of the PURG statement is:

- Optional label (can be used as a target for pseudo-ops).
- Op code PURG.
- Macro name in the operand field.

The macro name must be enclosed in apostrophes (the first apostrophe must be in column 35). If the macro name is not properly formatted, is missing, or cannot be found, the PURG statement will be flagged with a warning flag (Q) and the PURG operation will not be performed.

Automatic Name Generation

purpose

If your macro definition contains a label that isn't a parameter of the macro, and if you call the macro more than once in a given assembly, you'll get a multiple-definition error for that label. You can get around this problem by making all labels used in macros parameters of the macros, but then you have to supply all the labels every time you call a macro. An easier method to use is automatic name generation.

use an apostrophe

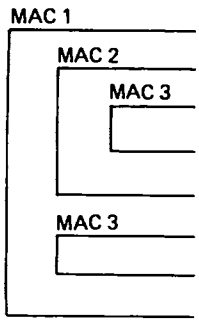
Instead of writing out a complete label in the macro definition, you write from one to four alphameric characters, the first of which must be alphabetic, followed by an apostrophe. Each time a macro is called in a given assembly, the Macro Assembler replaces the apostrophe with a number--a different number each time.

generates a five character label

The number of digits added to your label always causes a five character label to be generated. For example, repeated uses of the label P' result in the labels P0001, P0002, P0003, and so on. Repeated uses of the label PAM' result in PAM01, PAM02, PAM03, and so on.

problems in nested macros

The digits inserted into your label by the above method are determined by a counter maintained by the Macro Assembler. When the Macro Assembler encounters a macro that utilizes the automatic name generation feature, the counter is



incremented by one and remains at that value until the expansion of that macro is complete. This causes a problem when other macro calls are included in the first macro (that is, the nesting of macros) and one of them is called twice. MAC3 in the diagram at the left is called twice within the nest. If MAC3 contains labels in the form of letters followed by an apostrophe, the same set of labels will be generated twice because the counter is not incremented until MAC1 is completed.

use an apostrophe before

The way around this problem is to place an apostrophe before the characters in the label. The Macro Assembler replaces this apostrophe with a different alphabetic character (beginning with A) each time a macro is called within a given nest. Thus, if the label 'RAB is used in a macro and the macro is called four times in the same nest, the labels generated are ARAB, BRAB, CRAB, and DRAB.

begin and end with apostrophe

You can use apostrophes at both the beginning and the end of a label; in fact, this is the simplest way of ensuring that you don't get multiply-defined labels. The label 'C' can result in labels from AC001 through TC999 (T rather than Z because only 20 levels of macro nesting are allowed). The label 'SPY' can result in labels from ASPY1 through TSPY9.

Note that if you alternately call macros with automatic name generation, the numbers of the generated labels will also alternate. For example, if you call two macros alternately and the first has a label C' and the second has a label D', the resulting labels would be C0001, D0002, C0003, D0004, and so on.

The Macro Assembler also automatically generates labels in the remarks field (not on comment records) of assembler-language statements. For this reason, you must be careful when using apostrophes within a remark.

Concatenation

period or ampersand

By concatenating two parts of a field, you can join a parameter to a character string, two parameters, or two character strings. You can use either a period or an ampersand as a concatenator. The ampersand functions as a concatenator only if it appears as the first character in a parameter of a definition prototype statement. Otherwise, it functions as a plus operator. If you use the ampersand as a concatenator, you may use six characters, including the ampersand, for the parameter name.

period example

The following example shows several uses of period concatenation.

macro definition

Label (21–25)	Operation (27–30)	F (32)	T (33)	Operands & Remarks (35–55)
	MAC			
NAM	MOVE			A,B,C,N,M,U
	LDX		N	M
X.NAM	LD.A	L	N	AB.C
	ST.B	L	N	BB.C
	MDX		N	-U
	MDX			X.NAM
	MEND			

Using the above definition with the macro calls below, the code following the calls would be generated.

	Label (21–25)	Operation (27–30)	F (32)	T (33)	Operands & Remarks (35–55)
	*				
	*	'MOVE'			MACRO CALL
	*				
macro call	FIRS	MOVE			,O,2S,1,5,1
code generated		LDX		1	5
	XFIRS	LD	L	1	AB2S
		STO	L	1	BB2S
		MDX		1	-1
		MDX			XFIRS
macro call	ABLE	MOVE			D,D,L1,3,18,2
code generated		LDX		3	18
	XABLE	LDD	L	3	ABL1
		STD	L	3	BBL1
		MDX		3	-2
		MDX			XABLE

Notice that in the above examples, parameter substitution does not occur for A and B in the names AB.C and BB.C. You need to be careful when you are using any op code as a parameter, because substitution will occur for all uses of that op code.

ampersand example

The following examples show several ways in which the ampersand can be used as a concatenator. Notice in the second example that the B in the macro definition is not preceded by an ampersand and, therefore, the B is a parameter, but the &B is not.

example 1

macro definition

macro call

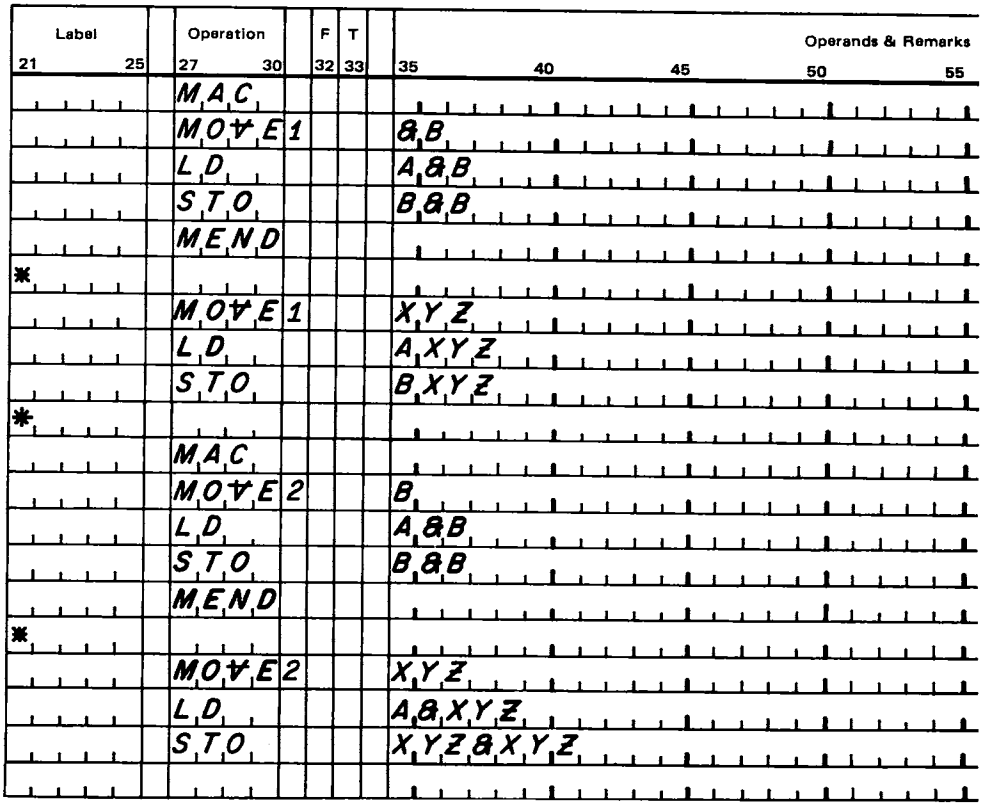
code generated

example 2

macro definition

macro call

code generated



Because a period is used frequently as a decimal point when writing DEC and XFLC statements, the Macro Assembler inhibits period concatenation when writing these statements. The ampersand concatenation feature may still be used.

Optional Remarks

retaining remarks

When you define a macro, you specify whether or not you want remarks on the macro definition statements to be reproduced each time the macro is expanded. If you want to keep the remarks, place any nonblank character in column 35 of the MAC or SMAC statement. The disk space required for the macro is increased according to the amount of space required for your remarks. Comment statements within the definition are always retained and listed within the expansion.

parameters in remarks

If parameters occur in remarks, parameter substitution is performed for the remarks also. This substitution includes automatically-generated names. The Macro Assembler also substitutes parameters into comments statements (asterisk in column 21), but it does not substitute automatically-generated names into comments statements. On such statements, the apostrophe is treated like any other character.

All records are truncated following column 71 of the record, and no error indication is given.

Indirect Parameter Substitution

use a semicolon

The indirect parameter substitution feature allows you to substitute one parameter for another when a macro is expanded. You do this by specifying a semicolon followed by any valid assembler-language expression, instead of the parameter you wish to replace, in the macro call.

determining position

The value of this expression is evaluated by the Macro Assembler and is considered to be the position of the replacement parameter in this macro call. For example, if the expression is evaluated as 3, the parameter in the third position is the replacement parameter. Remember, in determining the position of the replacement parameter, the label field of the call is the first parameter.

The position number and semicolon are counted as one parameter. If the replacement parameter position referenced is not specified in the macro call, an empty parameter is substituted. An indirectly-specified parameter may select as many as 19 other indirectly-specified parameters. If this limit is exceeded, an empty parameter is substituted, and the macro call is flagged with a syntax error indicator. Processing continues with the next statement.

syntax errors

If a symbol within a parameter substitution specification is not defined before its use, the substituted expression is evaluated as zero, the referenced parameter is evaluated as a blank parameter, and processing continues.

examples

The following examples demonstrate the use of indirect parameter substitution.

	Label (21–25)	Operation (27–30)	F (32)	T (33)	Operands & Remarks (35–55)
		MAC			
macro definition	LOC	BRTB			A
	LOC	BSC	L		A
		MEND			
	*				
macro call		BRTB			;4,LOC1,LOC2,LOC3
code generated		BSC	L		LOC2
	*				
	*				
macro call	A	SET			1
	NAME	BRTB			;A+3,LOC1,;A+5,LOC2,LOC3
code generated	NAME	BSC	L		LOC3

In a nested macro call, if the symbol following the semicolon is a parameter of the outer macro, the parameter must be concatenated to the semicolon for recognition by the Macro Assembler. (Concatenation was described previously under "Concatenation.")

Division Operator

a slash

The Macro Assembler interprets any slash in the operand field as a division operator, unless it can be interpreted as a hexadecimal number indicator. A hexadecimal constant is indicated by a slash in column 35. A slash preceded by an operator is interpeted as the hexadecimal indicator. A

preceded by a term

slash preceded by a term is interpreted as a division operator. A division operator may be immediately followed by a + or - to indicate whether the divisor is positive or negative. If no + or - follows the division operator, the divisor is assumed to be positive. A division operator followed immediately by a multiplication operator is flagged as a syntax error (S). Division by the internal address register (IAR) is allowed in an absolute assembly; in a relocatable assembly, it is flagged as a relocation error (R).

Each division operation within each term is performed from left to right. The 16-bit quotient is the result of a division operation; the remainder is lost.

The Macro Assembler performs all operations in an expression algebraically. For example:

```
(1) 3+2*4/2 = 7    (2) 5*2/3+9/-3 = 0
```

In example 1 the entire term 2*4/2 is evaluated from left to right before it is combined with 3. In example 2 the term 5*2/3 is evaluated (left to right) first. Then the term 9/-3 is evaluated and combined with the first term. Note that since the result of a division is always an integer with the remainder ignored, the first term in example 2 if written as 2/3*5 would be evaluated as 0 but 3 would result when the term is written as 5*2/3.

Note that division by zero results in a zero quotient and a warning flag. A relocation error flag (R) will be issued if either the dividend or divisor is relocatable. If two consecutive division operators are found in a single term, the term will be replaced by zero. For example, 27/9/3 will not be correctly evaluated and will be replaced by zero; the statement in which it occurred will be flagged as a syntax error (S).

Symbolic Tag Field

purpose

If you wish to change an index register designation once you have coded that portion of your program, you can do this by using the symbolic tag field feature. You specify the tag field with a one-character symbol which is defined in the assembly by means of an EQU or SET statement (see the 1130 Assembler Language manual, Order Number GC26-5927 or the 1800 Assembler Language manual, Order Number GC26-5882).

in SET pseudo-ops

You may change the value of the tag symbol dynamically when using the SET pseudo-operation to define the tag field. The tag symbol retains the value of the last SET statement associated with it from the first pass of the Macro Assembler until the Macro Assembler encounters an associated SET statement in the second pass. The EQU statement is not read on the second pass of the Macro Assembler and,

consequently, cannot be used to change the value of the tag symbol during assembly.

example

The following example illustrates use of the symbolic tag field in instructions and the code generated by those instructions.

source statements (as on listing)

code listed as if these were the source statements

Label	Operation	F	T	Operands & Remarks
21 25	27 30	32	33	35 40 45 50 55
A	SET			1
	LDX		A	5
LOOP1	LD	L	A	TABL1
	STO	L	A	TABL2
	MDX		A	-1
	MDX			LOOP1
A	SET			2
*				
A	LD	L	A	0
	SET			1
	LDX		1	5
LOOP1	LD	L	1	TABL1
	STO	L	1	TABL2
	MDX		1	-1
	MDX			LOOP1
A	SET			2
	LD	L	2	0

Programming Techniques

The following items should help you in using the Macro Assembler.

CHECKING FOR BLANK PARAMETERS

It is generally desirable to simplify macro calls by defining macros so that parameters (preferably the last parameter in the call) may be optionally omitted. Passing a blank parameter to the macro can cause a special sequence of code or no code to be generated. This blank parameter in the call causes no substitution to occur for the parameter when the macro is expanded. Consider the following examples:

```
// JOB   01 JAN 70 00.504 HRS
// * EXAMPLE 1 SHOWS HOW A BLANK MACRO PARAMETER SUBSTITUTES
// * WHEN USED IN A MACRO CALL
// ASM SAMPL   01 JAN 70 00.505 HRS
*LIST
*OVERFLOW SECTORS ,,1
                              MAC     C          BEGIN MACRO DEFINITION
                              APPLE   GOOD
            00001             DC      GOOD*5     CONSTANT DEFINED
                              MEND               END OF MACRO DEFINITION
0000                  START   EQU     *
                      *
                      *       THE FOLLOWING CALL TO APPLE
                      *       CAUSES A SYNTAX ERROR
                      *
                              APPLE              MACRO CALL
0000 0  0000      S +         DC      *5     CONSTANT DEFINED
0001 30 059C98C0              EXIT
0004    0000                  END     START

001 ERROR(S) AND 000 WARNING(S) IN ABOVE ASSEMBLY.


D46 'NOLD ON
DMP FUNCTION ABORTED
// * EXAMPLE 2 SHOWS HOW A SET INSTRUCTION CAN BE USED
// * TO BYPASS THE PROBLEM OF SYNTAX ERRORS GENERATED BY
// * BLANK PARAMETRS IN A MACRO CALL
// ASM SAMPL   01 JAN 70 00.511 HRS
*LIST
*OVERFLOW SECTORS ,,1
                              MAC     C          BEGIN MACRO DEFINITION
                              APPLE   GOOD
            00001    $APPL    SET     GOOD       BLANK OPERAND SET TO 0
            00002             DC      $APPL*5    CONSTANT DEFINED
                              MEND               END OF MACRO DEFINITION
                      *
                      *       PASSING A BLANK PARAMETER TO APPLE
                      *       WILL CAUSE THE VALUE OF ZERO TO BE
                      *       SUBSTITUTED - SEE CALL BELOW
                      *
0000 0  1000          BEGIN   NOP
                              APPLE              MACRO CALL
0000                 +$APPL   SET          BLANK OPERAND SET TO 0
0001 0  0000         +        DC      $APPL*5   CONSTANT DEFINED
0002 30 059C98C0              EXIT
0004    0000                  END     BEGIN

000 ERROR(S) AND 000 WARNING(S) IN ABOVE ASSEMBLY.
```

The SET statement is used in Example 2 to avoid Macro Assembler errors that can result from passing blank parameters. Note the use of $ in the label field. Use of $, @, and # in macro labels may help prevent conflict with other labels used in the program. These characters may also be used in conjunction with the automatic name generation feature.

RESTRICTIONS ON AIF, AIFB, AND SET PSEUDO-OPERATIONS

We have said that symbols used either in the SET operand field or in the AIF or AIFB expression must have been defined before the referencing SET, AIF, or AIFB statement is processed. Since symbols used as above in AIF, AIFB, and SET are evaluated at the time they are first encountered, they are flagged with the U error even if the symbols are defined later in the program. This error flagging has been implemented to help ensure that code generated by SET, AIF,

or AIFB statements is the code intended by the user. See Example 3.

```
// JOB   01 JAN 70 00.517 HRS
// * EXAMPLE 3 ILLUSTRATES THE RESTRICTION ON THE AIF OPERATION
// * THAT A SYMBOL MUST BE DEFINED PRIOR TO ITS USE IN
// * THE SET, AIF, OR AIFB OPERATIONS
// ASM SAMPL   01 JAN 70 00.517 HRS
*LIST
*OVERFLOW SECTORS ,,2
                         MAC     C          BEGIN MACRO DEFINITION
                         TABLE
            00001        AIF     (A LE 0),EXIT EXIT MACRO IF A LE 0
            00002   CONT ANOP
            00003        DC      /00F0+A    CONSTANT DEFINED
            00004   A    SET     A+1        INCR A
            00005        AIFB (A LE /F),CONT  END TABLE IF A GT 15
                    EXIT MEND               END MACRO DEFINITION
                    *
                    *    IF A IS NOT DEFINED PRIOR TO THE CALL
                    *    TO TABLE, NO TABLE WILL BE GENERATED
                    *    SINCE A IS EVALUATED AS ZERO
                    *
0000 0  1000        BEGIN NOP
                         TABLE              MACRO CALL
                   U+    AIF     (A LE 0),EXIT EXIT MACRO IF A LE 0
0001 30 059C98C0         EXIT
0004    0000             END     BEGIN

001 ERROR(S) AND 000 WARNING(S) IN ABOVE ASSEMBLY.
```

LABEL AND BLANK PARAMETER CHECKING USING AGO

The Macro Assembler does not have an explicit method for character string comparison. However, the AGO and ANOP pseudo-operations provide a means of label checking both within and outside of a macro. Consider Example 4.

```
// JOB   01 JAN 70 00.525 HRS
// * EXAMPLE 4 ILLUSTRATES THE USE OF THE AGO AND ANOP OPERATIONS
// * TO HANDLE BLANK MACRO PARAMETERS WITHOUT CAUSING ASSEMBLER
// * ERRORS TO OCCUR
// DUP   01 JAN 70 00.525 HRS
*DELET    D          MACRO *****
DMP FUNCTION COMPLETED
*DFILE               MACRO 0007
WILL RESERVE AT SCTR ADDR   038D
DMP FUNCTION COMPLETED
*MACRO UPDATE
BUILD 'MACRO'
                     038D 0005 ** LIBRARY END **
                     0000
ENDUP
                          UPDATE COMPLETED
// ASM SAMPL   01 JAN 70 00.531 HRS
*MACLIB MACRO
*LIST
                         SMAC    C          BEGIN MACRO DEFINITION
                    LABEL DISK   FUN1,ARE1,ERR1
            00001   LABEL LIBF   DISKN      CALL TO DISKN SUBR
            00002         AIF    (FUN1 EQ 1),READ  TEST FOR READ FUNC
            00003         AIF    (FUN1 EQ 3),WRITE  TEST FOR WRITE FUN
            00004         AIF    (FUN1 EQ 0),TEST  TEST FOR TEST FUNC
            00005         LIST   ON
            00006   *     ILLEGAL REQUEST
            00007         LIST
            00008         AGO    EXIT       EXIT MACRO
            00009   READ  ANOP
```

```
                    00010          DC       /1000      READ FUNC CODE
                    00011          AGO      AREA       GO ASSEMBLE I/O AREA ADDR
                    00012    WRITE ANOP
                    00013          DC       /3000      WRITE FUNC CODE
                    00014          AGO      AREA       GO ASSEMBLE I/O AREA ADDR
                    00015    TEST  ANOP
                    00016          DC       /0000      TEST FUNC CODE
                    00017          AGO      ARE1       IF BLANK,ASM NEXT STMNT
                    00018          DC       *+2        AUTOMATIC TEST I/O ADDR
                    00019          MDX      *-4        CONT TO TEST BUSY
                    00020          MDX      *+2        BYPASS DISK HEADER
                    00021          BSS      2          AUTOMATIC DISK HEADER
                    00022          AGO      EXIT       EXIT MACRO
                    00023    ARE1  ANOP
                    00024          DC       ARE1       USER SPECIFIED I/O ADDR
                    00025          MDX      *-4        CONT TO TEST BUSY
                    00026          AGO      EXIT       EXIT MACRO
                    00027    AREA  ANOP
                    00028          DC       ARE1-1     I/O AREA ADDR
                    00029          AGO      ERR1       IF BLANK,ASM NEXT STMNT
                    00030          DC       *+1        ERROR ENTRY ADDR
                    00031          MDX      *+3        BYPASS ERROR SUBR
                    00032          DC       *-*        ENTER HERE ON ERROR
                    00033          BSC   I  *-3        RETURN TO RETRY OPERATION
                    00034          AGO      EXIT       EXIT MACRO
                    00035    ERR1  ANOP
                    00036          DC       ERR1       ERROR SUBR ADDR
                           EXIT    MEND                END MACRO DEFINITION
                           START   DISK     1,BUFFR    READ SCTR OFF DISK
0000 20 04262495           +START  LIBF     DISKN      CALL TO DISKN SUBR
                           +       AIF      (1 EQ 1),READ  TEST FOR READ FUNC
                           +READ   ANOP
0001 0  1000               +       DC       /1000      READ FUNC CODE
                           +       AGO      AREA       GO ASSEMBLE I/O AREA ADDR
                           +AREA   ANOP
0002 1  0019               +       DC       BUFFR-1     I/O AREA ADDR
                         Q +       AGO           IF BLANK,ASM NEXT STMNT
0003 1  0005               +       DC       *+1        ERROR ENTRY ADDR
0004 0  7003               +       MDX      *+3        BYPASS ERROR SUBR
0005 0  0000               +       DC       *-*        ENTER HERE ON ERROR
0006 01 4C800005           +       BSC   I  *-3        RETURN TO RETRY OPERATION
                           +       AGO      EXIT       EXIT MACRO
                                   DISK     0          TEST BUSY
0008 20 04262495           +       LIBF     DISKN      CALL TO DISKN SUBR
                           +       AIF      (0 EQ 1),READ  TEST FOR READ FUNC
                           +       AIF      (0 EQ 3),WRITE  TEST FOR WRITE FUN
                           +       AIF      (0 EQ 0),TEST  TEST FOR TEST FUNC
                           +TEST   ANOP
0009 0  0000               +       DC       /0000      TEST FUNC CODE
                         Q +       AGO           IF BLANK,ASM NEXT STMNT
000A 1  000D               +       DC       *+2        AUTOMATIC TEST I/O ADDR
000B 0  70FC               +       MDX      *-4        CONT TO TEST BUSY
000C 0  7002               +       MDX      *+2        BYPASS DISK HEADER
000D    0002               +       BSS      2          AUTOMATIC DISK HEADER
                           +       AGO      EXIT       EXIT MACRO
                                   DISK     3,BUFFR    WRITE SCTR TO DISK
000F 20 04262495           +       LIBF     DISKN      CALL TO DISKN SUBR
                           +       AIF      (3 EQ 1),READ  TEST FOR READ FUNC
                           +       AIF      (3 EQ 3),WRITE  TEST FOR WRITE FUN
                           +WRITE  ANOP
0010 0  3000               +       DC       /3000      WRITE FUNC CODE
                           +       AGO      AREA       GO ASSEMBLE I/O AREA ADDR
                           +AREA   ANOP
0011 1  0019               +       DC       BUFFR-1     I/O AREA ADDR
                         Q +       AGO           IF BLANK,ASM NEXT STMNT
0012 1  0014               +       DC       *+1        ERROR ENTRY ADDR
0013 0  7003               +       MDX      *+3        BYPASS ERROR SUBR
0014 0  0000               +       DC       *-*        ENTER HERE ON ERROR
0015 01 4C800014           +       BSC   I  *-3        RETURN TO RETRY OPERATION
                           +       AGO      EXIT       EXIT MACRO
0017 30 059C98C0                   EXIT
0019 0  0140                       DC       320        WORD COUNT
001A 0  0100               BUFFR   DC       /0100      SCTR 100
001B    0140                       BSS      320        ALLOCATE DATA AREA
015C    0000                       END      START

000 ERROR(S) AND 003 WARNING(S) IN ABOVE ASSEMBLY.
```

Note that if a macro parameter, which may be a character string greater than one character, is used in an AIF statement to check for character values, a syntax flag will be generated by such a statement and the value of the substituted expression will be zero. See Example 5.

```
// JOB   01 JAN 70 00.552 HRS
// * EXAMPLE 5 ILLUSTRATES PROBLEMS WHICH CAN BE
// * ENCOUNTERED WITH CHARACTER STRING COMPARISON
// ASM SAMPL   01 JAN 70 00.552 HRS
*LIST
*OVERFLOW SECTORS ,,5
                          MAC     C          BEGIN MACRO DEFINITION
                          TEST    FUNC
            00001         AIF     (.FUNC EQ .R),READ  TEST FOR R
            00002         AIF     (.FUNC EQ .W),WRITE  TEST FOR W
            00003         LIST    ON
            00004   *     ILLEGAL REQUEST
            00005         LIST
            00006         AGO     EXIT       EXIT MACRO
            00007   READ  DC      /1000      READ FUNCTION
            00008         AGO     EXIT       EXIT MACRO
            00009   WRITE DC      /3000      WRITE FUNCTION
                    EXIT  MEND               END MACRO DEFINITION
                          TEST    R          GENERATE FUNCTION CODE
                    +     AIF     (.R EQ .R),READ  TEST FOR R
0000 0  1000        +READ DC      /1000      READ FUNCTION
                    +     AGO     EXIT       EXIT MACRO
                          TEST    READ       GENERATE ILLEGAL REQUEST
                   S+     AIF     (.READ EQ .R),READ  TEST FOR R
                   S+     AIF     (.READ EQ .W),WRITE  TEST FOR W
                    +*    ILLEGAL REQUEST
                    +     AGO     EXIT       EXIT MACRO
0001 30 059C98C0          EXIT
0004    0000              END     READ

002 ERROR(S) AND 000 WARNING(S) IN ABOVE ASSEMBLY.
```

MACRO PARAMETER SUBSTITUTION

When special features such as call continuation and indirect parameter substitution are used in nested macro calls you must account for such special instances as you code your macro. If, for example, you anticipate that a nested macro call may be a continued call, you must pass the continuation indicator and ensure that any parameter string from the first call is completed on each nested continuation record. See Example 6. If you wish to pass a symbolic value to a nested macro call and an indirect substitution expression and that symbol appears next to the semicolon, you must concatenate the symbol to the semicolon. See Example 7.

```
// JOB   01 JAN 70 00.561 HRS
// * EXAMPLE 6 ILLUSTRATES THE METHOD FOR HANDLING CONTINUATION OF
// * NESTED MACRO CALLS
// ASM SAMPL   01 JAN 70 00.562 HRS
*LIST
*OVERFLOW SECTORS ,,5
                              MAC     C          BEGIN MACRO DEFINITION
                              ABLE    A,X
            00001             AIF     (A GT 0),GEN  TEST TO GENERATE MSG
            00002             AGO     END        EXIT MACRO
            00003   GEN       ANOP
            00004             MSG     X
                    END       MEND               END MACRO DEFINITION
                              MAC                BEGIN MACRO DEFINITION
                              MSG     A
            00001             EBC     A          GENERATE EBC MESSAGE
                              MEND               END MACRO DEFINITION
0000 0  1000          BEGIN   NOP
0001 30 059C98C0              EXIT
                              ABLE X  1,((.THIS IS MACRO GENERATED MSG NO.
                                      10.))
                      +       AIF     (1 GT 0),GEN  TEST TO GENERATE MSG
                      +GEN    ANOP
                      +       MSG     (.THIS IS MACRO GENERATED MSG NO. 10.)
0003    0022          +       EBC     .THIS IS MACRO GENERATED MSG NO. 10.
0014    0000                  END     BEGIN

000 ERROR(S) AND 000 WARNING(S) IN ABOVE ASSEMBLY.
```

```
// JOB   01 JAN 70 00.571 HRS
// * EXAMPLE 7 ILLUSTRATES A METHOD OF SPECIFYING INDIRECT
// * PARAMETER SUBSTITUTION IN NESTED MACRO CALLS
// ASM SAMPL   01 JAN 70 00.571 HRS
*LIST
*OVERFLOW SECTORS ,,5
                              MAC     C          BEGIN MACRO DEFINITION
                              BEAN    A
            00001             AIF     (A GT 2),BRANC  GENERATE BR IF A GT 2
            00002             AGO     END        EXIT MACRO
            00003   BRANC    ANOP
            00004             BR       .A,BR1,BR2,BR3  GENERATE BRANCH
                    END       MEND               END MACRO DEFINITION
                              MAC     BEGIN MACRO DEFINITION
                              BR      LOCN
            00001             BSC  L  LOCN       GENERATE LONG BRANCH
                              MEND
                      *
0000 0  1000          BEGIN   NOP
                              BEAN    3
                      +       AIF     (3 GT 2),BRANC  GENERATE BR IF 3 GT 2
                      +BRANC  ANOP
                      +       BR       3,BR1,BR2,BR3  GENERATE BRANCH
0001 01 4C000003      +       BSC  L  BR1        GENERATE LONG BRANCH
0003 30 059C98C0       BR1    EXIT
0006    0000                  END     BEGIN

000 ERROR(S) AND 000 WARNING(S) IN ABOVE ASSEMBLY.
```

; in col. 35 not listed by printer, but present in source code

SAMPLE PROGRAM

The following sample program illustrates three macros: their definitions, calls to them, and the code generated.

```
// JOB   01 JAN 70 00.310 HRS
// *
// *  MACRO ASSEMBLER SAMPLE PROGRAM
// *
// ASM SAMPL   01 JAN 70 00.310 HRS
*LIST
*OVERFLOW SECTORS 0,0,1                                          (1)
```

a move macro

```
                *****************************************************
                *     DEFINE A MACRO, 'MOVE', DESIGNED TO MOVE      *
                *     DATA FROM ONE AREA TO ANOTHER.                *
                *****************************************************
                      MAC
                LABEL MOVE    FROM,TO,CNT                           (3)
00001                 LIST    OFF
00002                 AIF     (CNT LE 0),NOMVE                      (2)
00003                 LIST
00004           LABEL LDX  L1 -CNT       INITIALIZE LOOP COUNTER
00005           LOOP' LD   L1 FROM+CNT   GET WD TO BE MOVED
00006                 STO  L1 TO+CNT     MOVE IT
00007                 MDX   1 1          BUMP MOVE (LOOP) COUNTER
00008                 MDX     LOOP'      LOOP UNTIL MOVE COMPLTED
00009           NOMVE LIST
                      MEND
```

macro definition (4)

a message macro

```
                *****************************************************
                *     DEFINE A MACRO, 'MSG', DESIGNED TO SET UP     *
                *     THE PRINTING OF A MESSAGE VIA A DMES STMNT.   *
                *****************************************************
                      MAC
                      MSG     TEXT
00001                 DMES    'S TEXT  'E
                      MEND
```

a parameter checking macro

```
                *****************************************************
                *     DEFINE A MACRO 'VKUS' TO CHECK FOR AN         *
                *     UNSPECIFIED PARAMETER IN A MACRO CALL.        *
                *     #VKUS IS SET TO 0 IF THE PARAMETER WAS        *
                *     UNSPECIFIED. #VKUS IS SET TO 1 IF THE         *
                *     PARAMETER WAS SPECIFIED.                      *
                *****************************************************
                      MAC
                      VKUS    PARAM
00001                 AGO     PARAM
00002                 ANOP
00003           #VKUS SET     0
00004                 AGO     QUIT
00005           PARAM ANOP
00006           #VKUS SET     1
                QUIT  MEND
```

macro definition

DC--generating macros

```
                *****************************************************
                *     DEFINE TWO MACROS, 'DCS' AND 'DCB', TO        *
                *     GENERATE DC STATEMENTS. ONLY 'DCS' IS         *
                *     REFERENCED BY THE USER. 'DCB' IS CALLED       *
                *     INTERNALLY BY 'DCS'.                          *
                *****************************************************
                      MAC

                LABEL DCS     CNT,A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q
00001                 LIST    (6) OFF
00002                 VKUS    LABEL
                (5)   AIF     (#VKUS EQ 0),BLANK
00004                 LIST    (6)
00005           LABEL EQU     *
00006           BLANK LIST    (6) OFF
00007           Z     SET     CNT
00008           X     SET     3
00009           RETRN AIF     (Z EQ 0),QUIT
00010                 DCB     ;X,A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q
00011           X     SET     X+1
00012           Z     SET     Z-1
00013                 AGOB    RETRN
00014           QUIT  LIST    (6)
                      MEND
```

VKUS macro call

; in col. 35 not listed by printer, but present in source code

```
                    MAC
                    DCB     Y
00001               LIST
00002               DC      Y
00003               LIST    OFF
                    MEND

                    ********************************************************
                    *       FIRST-  GENERATE A VARIABLE LENGTH TABLE
                    *
                (8)→TABLE DCS     10,0,1,2,3,4,5,6,7,8,9
0000                +TABLE EQU     *
0000 0  0000        +       DC      0
0001 0  0001        +       DC      1
0002 0  0002        +       DC      2
0003 0  0003    (7)→+       DC      3
0004 0  0004        +       DC      4   (9)
0005 0  0005        +       DC      5
0006 0  0006        +       DC      6
0007 0  0007        +       DC      7
0008 0  0008        +       DC      8
0009 0  0009        +       DC      9
                (10)→DCS    5,/1000,/2000,/3000,/4000,/5000
                  Q +       AGO
000A 0  1000        +       DC      /1000
000B 0  2000        +       DC      /2000
000C 0  3000        +       DC      /3000
000D 0  4000        +       DC      /4000
000E 0  5000        +       DC      /5000

                    *
                    *       NOW MOVE TABLE TO ANOTHER AREA
                    *
                (11) START  MOVE    TABLE,NEW,15
000F 00 6500FFF1    +START  LDX   L1 -15
0011 01 C500000F    +LOOP1  LD    L1 TABLE+15
0013 01 D5000028    +       STO   L1 NEW+15
0015 0  7101        +       MDX    1 1
0016 0  70FA        +       MDX      LOOP1


0017 30 059C98C0            EXIT
0019    000F          NEW   BSS     15
                            MSG  (12)('THE DATA HAS BEEN MOVED)
0028    001C          +       DMES    'S THE DATA HAS BEEN MOVED  'E
0036    000F                  END     START

000 ERROR(S) AND 001 WARNING(S) IN ABOVE ASSEMBLY.
```

DCS macro call

code generated

DCS macro call

code generated

move macro call

code generated

message code generated

---

(1) The 1 indicates the overflow sector necessary to house temporary macro definitions in this assembly.

(2) If CNT is less than or equal to 0, the definition is not expanded.

(3) Sequence numbers of definition statements are printed for easier visual perception.

(4) The move loop is generated with your program's parameters.

(5) This AIF checks the call to the DCS macro to determine if you want a label to be associated with the first DC.

(6) You can significantly alter a printed listing when a macro is called through use of the LIST pseudo-op.

(7) All statements resulting from a macro expansion are flagged with a plus in column 20.

(8) This is a call to DCS with the label specified.

(9) Note how use of the LIST pseudo-op caused this expansion to be printed.

(10) This is a call to DCS without the label specified.

(11) LOOP1 illustrates the automatic name generation feature of the Macro Assembler. LOOP1 was named as LOOP' in the macro definition.

(12) By using parentheses, the parameter TEXT (see the MSG definition) is replaced by a string of characters into a DMES statement.

NOTES

This page intentionally left blank.

The features of the Macro Assembler permit you to create a language that can help programmers who are new to your installation or who don't understand the more detailed aspects of certain operations. This specialized language may also be used by programmers who know the operation and want to simplify their job or who need to interact with persons other than programmers. Such a specialized language may also be used to help others learn to write simple programs for your system.

Consider the following 1800 MPX example.

Mr. Jones is a programmer who works for a sports information service center and is in charge of all baseball statistics. He has been assigned the following project in response to a customer requisition.

SPORTS INFORMATION SERVICE CENTER

programmer: A.B. Jones
date needed: September 10, 1972
date submitted: September 7, 1972
customer: I26–3381–07

Problem:

1. Read a batch of data cards having the following format:
   columns 1–20 Player's name
   25–28 Number of at-bats
   30–33 Number of singles
   35–38 Number of doubles
   40–43 Number of triples
   45–48 Number of home-runs
2. Compute the player's batting average, slugging percentage, and home-run ratio.
3. Print all the information mentioned in 1 and 2.
4. Terminate execution of the program when a data card containing an asterisk (*) in column 1 is read.

Mr. Jones knows that in order to answer this request, he will have to use some macros from the general purpose library, use some macros from his own library, and use the following computations in his solution.

Batting average=number of hits number of times at bat.
For example, if hits=13 and at-bats=64, 13 ÷ 64=.203
The batting average would be .203.

Slugging percentage=(number of singles + 2 x number of doubles + 3 x number of triples + 4 x number of home runs) ÷ number of at-bats.
For example, if at-bats=64, singles=8, doubles=3, triples=1, and home runs=1, the slugging percentage would be [8+2(3)+3(1)+4(1)] ÷ 64 or 21 ÷ 64 or .328

Home run ratio=number of at-bats ÷ home runs.
For example, if at-bats=64 and home runs=4, the home run ratio would be 64 ÷ 4 or 16.

Mr. Jones then looks in the guide to general purpose macros and determines which ones he wants to use. The guide lists the macros in a library that have been built by all the programmers who work with Mr. Jones. These macros are ones that are used frequently by the programmers, such as, read a card and print a line.

The following listing demonstrates how this library of macros was created. In this case the library has been named SYSTM.

```
// JOB   00 JAN 00 00.235 HRS
// DMP   00 JAN 00 00.235 HRS
*DFILE    0          SYSTM 10
WILL RESERVE AT SCTR ADDR   04C0
DMP FUNCTION COMPLETED
*MACRO UPDATE
 BUILD 'SYSTM'
 NAME LABEL,INPUT,OUTPT
 ADD 'DCBIN'
                    04C0 0005 DCBIN
                x   00001  LABEL CALL     DCBIN
                x   00002         DC      INPUT
                x   00003         DC      OUTPT
                    00004  ** MACRO END **
 ADD 'BINDC'
                    04C0 0019 BINDC
                x   00001  LABEL CALL     BINDC
                x   00002         DC      INPUT
                x   00003         DC      OUTPT
                    00004  ** MACRO END **
 NAME LABEL,CNTRL,AREA,WDCNT
 ADD 'LLIST'
                    04C0 002D LLIST
                x   00001  LABEL DC       0
                x   00002         DC      0
                x   00003         BSS     4
                x   00004         DC      0
                x   00005         DC      CNTRL
                x   00006         DC      AREA
                x   00007  AREA   DC      WDCNT
                x   00008         BSS     WDCNT
                    00009  ** MACRO END **
 NAME LABEL,LIST
 ADD 'RDCRD'
                    04C0 0054 RDCRD
                x   00001  LABEL CALL     CARON
                x   00002         DC      LIST
                x   00003         LD   L  LIST
                x   00004         BSC  L  *-4,Z
                    00005  ** MACRO END **
 ADD 'PRINT'
                    04C0 006F PRINT
                x   00001  LABEL CALL     PRNTN
                x   00002         DC      LIST
                x   00003         LD   L  LIST
                x   00004         BSC  L  *-4,Z
                    00005  ** MACRO END **
 NAME LABEL,CNTRL,INPUT,OUTPT,CHRCT
 ADD 'HOLPR'
                    04C0 0D8A HOLPR
                x   00001  LABEL CALL     HOLPR
                x   00002         DC      A'
                x   00003         DC      INPUT
                x   00004         DC      OUTPT
                x   00005         DC      A'+1
                x   00006         MDX     *+2
                x   00007  A'     DC      CNTRL
                x   00008         DC      CHRCT
                    00009  ** MACRO END **
 NAME LABEL,CNTRL,AREA,LABL1,LABL2

 ADD 'SLIST'
                    04C0 0D86 SLIST
                x   00001  LABEL DC       0
                x   00002         DC      0
                x   00003         BSS     4
                x   00004         DC      0
                x   00005  LABL1  DC      CNTRL
                x   00006  LABL2  DC      AREA
                    00007  ** MACRO END **
                    04C0 0DD5 ** LIBRARY END **
                    0000
 ENDUP
                          UPDATE COMPLETED
```

```
          21        27        35
Format:   LABEL     DCBIN     INPUT,OUTPUT
```

This macro calls the DCBIN subroutine which converts a decimal value in 5 IBM card-coded characters and 1 sign character to a 16-bit binary word.

```
          21        27        35
Format:   LABEL     BINDC     INPUT,OUTPUT
```

This macro calls the BINDC subroutine which converts a 16-bit binary number to its IBM card-coded decimal equivalent.

```
          21        27        35
Format:   LABEL     LLIST     CNTRL,AREA,WDCNT
```

CNTRL = control parameter
AREA = address of the area to be worked upon
WDCNT = word count of the I/O area
This macro ("long list") generates a nine word I/O list which may be used with the various MPX I/O subroutines.

```
          21        27        35
Format:   LABEL     RDCRD     LIST
```

LIST = address of the associated 9-word I/O list
This macro, used in conjunction with the LLIST macro or the SLIST macro which is shown below, calls the CARDN subroutine and also performs a busy test.

```
          21        27        35
Format:   LABEL     PRINT     LIST
```

LIST = address of the associated 9-word I/O list
This macro, used in conjunction with the LLIST or SLIST macro, calls the PRNTN subroutine and performs a "busy" test.

```
          21        27        35
Format:   LABEL HOLPR CNTRL,INPUT,OUTPT,CHRCT
```

CNTRL = 0 if 1053 code is desired; 1 if 1442 is desired
INPUT = address of the area to be converted
OUTPT = address of the area which will contain the converted data
CHRCT = number of characters to be converted
This macro calls the HOLPR subroutine which converts an area from IBM card code (Hollerith) to either 1443 or 1053 code.

```
          21        27        35
Format:   LABEL     SLIST     CNTRL,AREA,LABL1,LABL2
```

CNTRL = control parameter
AREA = address of the area to be worked upon
LABL1 = label (optional) of the control parameter DC
LABL2 = label (optional) of the area parameter DC
This macro ("short list") is similar to the LLIST in that a nine word I/O list is generated. It is different in that the word count and I/O area are not allocated and, also, that labels, if desired, may be placed on the control and area parameter DC statements. The SLIST macro would probably be used when it is desired at various points in a program to modify the control and/or area parameters.

He then determines which macros from his library, which he has named JONES, are required. This library contains macros that he has written because he finds them useful in his job of providing baseball statistics.

The following listing demonstrates how his library was created.

```
*OFILE    0         JONES 10
WILL RESERVE AT SCTR ADDR   04CA
DMP FUNCTION COMPLETED
*MACRO UPDATE
 BUILD 'JONES'
 NAME LABEL,FROM,TO,COUNT
 ADD 'MOVE'
                    04CA 0005 MOVE
                  X 00001         LDX  L1 FROM
                  X 00002         LDX  L2 TO
                  X 00003         LDX   3 COUNT
                  X 00004  'A'    LO    1 0
                  X 00005         STO   2 0
                  X 00006         MDX   1 1
                  X 00007         MDX   2 1
                  X 00008         MDX   3 -1
                  X 00009         MDX     'A'
                    00010  ** MACRO END **
NAME LABEL,X,COUNT,LIST,STORE
ADD 'SUM'
                    04CA 0034 SUM
                  X 00001  LABEL LDX   X -COUNT
                  X 00002        SLA     16
                  X 00003  B'    A    LX LIST+COUNT
                  X 00004        STO  L  STORE
                  X 00005        MDX   X 1
                  X 00006        MDX     B'
                    00007  ** MACRO END **
NAME LABEL,AB,HITS,BAVG
ADD 'BA'
                    04CA 0056 BA
                  X 00001         LO   L  HITS
                  X 00002         M       0'
                  X 00003         D    L  AB
                  X 00004         STO  L  BAVG
                  X 00005         MDX     *+1
                  X 00006  0'     DC      1000
                    00007  ** MACRO END **
ADD 'CDEQU'
                    04CA 0076 CDEQU
                  X 00001  CC1    EQU     LABEL+1
                  X 00002  CC5    EQU     LABEL+5
                  X 00003  CC10   EQU     LABEL+10
                  X 00004  CC15   EQU     LABEL+15
                  X 00005  CC20   EQU     LABEL+20
                  X 00006  CC25   EQU     LABEL+25
                  X 00007  CC30   EQU     LABEL+30
                  X 00008  CC35   EQU     LABEL+35
                  X 00009  CC40   EQU     LABEL+40
                  X 00010  CC45   EQU     LABEL+45
                  X 00011  CC50   EQU     LABEL+50
                  X 00012  CC55   EQU     LABEL+55
                  X 00013  CC60   EQU     LABEL+60
                  X 00014  CC65   EQU     LABEL+65
                  X 00015  CC70   EQU     LABEL+70
                  X 00016  CC75   EQU     LABEL+75
                    00017  ** MACRO END **
NAME LABEL,NUMBR,BY,PROD
ADD 'MPY'
                    04CA 00F9 MPY
                  X 00001  LABEL LD    L  NUMBR
                  X 00002        M       C'
                  X 00003        SLT     16
                  X 00004        STO  L  PROD
                  X 00005        MDX     *+1
                  X 00006  C'    DC      BY
                    00007  ** MACRO END **
NAME LABEL,AB,HR,RSULT
ADD 'HRATE'
                    04CA 011A HRATE
                  X 00001  LABEL LD    L  AB
                  X 00002        RTE     16
                  X 00003        SLA     16
                  X 00004        D    L  HR
                  X 00005        STO  L  RSULT
                    00006  ** MACRO END **
NAME LABEL,TEXT,X
ADD 'MSG1'
                    04CA 0135 MSG1
                  X 00001  LABEL DC      'Z'-'Y'
                  X 00002  'Y'   DMES  X TEXT
                  X 00003  'Z'   BES     0
                    00004  ** MACRO END **
NAME LABEL,TEXT1,TEXT2,X
ADD 'MSG2'
                    04CB 000E MSG2
                  X 00001  LABEL DC      'Z'-'Y'
                  X 00002  'Y'   DMES  X TEXT1
                  X 00003        DMES  X TEXT2
                  X 00004  'Z'   BES     0
                    00005  ** MACRO END **
```

---

21 27 35
Format: LABEL MOVE FROM,TO,COUNT
FROM = address of the data to be transferred
TO = address at which the data will be transferred
COUNT = number of words to be transferred
The MOVE macro transfers a block of data from one area to another.

---

21 27 35
Format: LABEL SUM X,COUNT,LIST,STORE
X = index register to be used
COUNT = number of words to add
LIST = address of the first word to be added
STORE = address where the sum shall be placed
The SUM macro calculates the sum of a block of contiguous words and places the result into a word designated by a STORE parameter.

---

21 27 35
Format: LABEL BA AB,HITS,BAVG
AB = address of the word which contains the number of at-bats
HITS = address of the word which contains the number of hits
BAVG = address where the batting average shall be placed
The BA macro calculates a batting average and places it into a word designated by the BAVG parameter.

---

21 27
Format: LABEL CDEQU
The CDEQU macro generates a series of EQU statements tailored for the processing of data cards.
LABEL = address of the word count word which precedes the buffer into which a data card will be read
Since EQU statements do not increase the size (in words) of a program, it is not wasteful if all of the labels generated by CDEQU are not used.

---

21 27 35
Format: LABEL MPY NUMBR,BY,PROD
NUMBR = address of one number to be multiplied
BY = second number (not an address) to be multipliad
PROD = address of the word where the product will be stored
The MPY macro multiplies two given numbers and stores the result into a word designated by the PROD parameter.

---

21 27 35
Format: LABEL HRATE AB,HR,RSULT
AB = address of the word which contains the number of at-bats
HR = address of the word which contains the number of home-runs
RSULT = address where the home-run ratio will be stored
The HRATE macro calculates a home-run ratio and places the result into a word designated by the RSULT parameter.

---

21 27 35
Format: LABEL MSG1 TEXT,X
TEXT = text, using DMES syntax, of the message
X = 0 if 1053 message, 1 if 1443 message
The MSG1 macro facilitates the printing of a message. A word count and DMES statement are generated.

---

21 27 35
Format: LABEL MSG2 TEXT1,TEXT2,X
The MSG2 macro is the same as the MSG1 macro except that two DMES statements (for longer messages) are generated.

```
NAME LABEL,CHK,TSTCH,LOC
ADD 'CHECK'
                    04CB 002B CHECK
                  X 00001  LABEL LD    L  CHK
                  X 00002        EOR      A'
                  X 00003        BSC   L  LOC,+-
                  X 00004        MDX      *+1
                  X 00005  A'    DC       TSTCH
                    00006  ** MACRO END **
```

```
                    21      27      35
Format:  LABEL   CHECK   CHK,TSTCH,LOC
CHK = address of the word to be checked
TSTCH = value of the word against which the check
        is made
LOC = "branch to" address for when the check shows
      the words are equal.
The CHECK macro compares a word against a test
character and branches to LOC if the two words are
identical.
```

```
NAME LABEL,CHAR,WITH,START,COUNT
ADD 'SPRSS'
                    04CB 0049 SPRSS
                  X 00001  LABEL LDX   1 COUNT
                  X 00002        LDX   L2 START
                  X 00003  A'    LD    2 0
                  X 00004        EOR      B'
                  X 00005        BSC   L  D',Z
                  X 00006        LD       C'
                  X 00007        STO   2 0
                  X 00008        MDX   2 1
                  X 00009        MDX   1 -1
                  X 00010        MDX      A'
                  X 00011        MDX      0'
                  X 00012  B'    DC       CHAR
                  X 00013  C'    DC       WITH
                  X 00014  D'    EQU      *
                    00015  ** MACRO END **
```

```
                    21      27      35
Format:  LABEL   SPRSS   CHAR,WITH,START,COUNT
CHAR = the value (character) to be suppressed
WITH = the value (character) to replace the suppressed
       charecter
START = address where suppression shall begin
COUNT = number of contiguous words to check for
        suppression
The SPRSS macro is used primarily to suppress leading
zeros in a numeric field about to be printed.  For
example, it would fecilitate printing 63 instead of
000063.
```

```
CONCAT 'SYSTM'
                    04CB 0091 ** LIBRARY END **
                    04C0
ENDUP
                          UPDATE COMPLETED
```

This allows Mr. Jones to have all the macros in his library available along with the system library, SYSTM.

Mr. Jones has now selected all the macros he is going to use, and the next step is for him to write the coding. The following is a list of the source coding Mr. Jones decided to use. Note that he could have further reduced the coding by using more complex macros. (In order to aid understanding, the macros in this sample are not complex; therefore, the coding required by Mr. Jones is far more extensive than would be needed in actual applications.)

```
START PRINT   PRLST     PRINT FIRST LINE (MES1)
      LDX  L1 MES2      GET ADDRESS OF MES2
      STX  L1 PR000     SET UP I/O AREA PARAMETER
      PRINT   PRLST     PRINT OUTPUT HEADINGS
READ  LDX   1 80
      STX  L1 CDBUF
      RDCRD   CDLST     READ A DATA CARD
      CHECK   CC1,/4220,END
      MOVE    CC25,WKBUF+2,4
      DCBIN   WKBUF,ATBAT
      MOVE    CC30,WKBUF+2,4
      DCBIN   WKBUF,SNGLS
      MOVE    CC35,WKBUF+2,4
      DCBIN   WKBUF,DBLES
      MOVE    CC40,WKBUF+2,4
      DCBIN   WKBUF,TRPLS
      MOVE    CC45,WKBUF+2,4
      DCBIN   WKBUF,HOMRS
      SUM     1,4,SNGLS,HITS
      BA      ATBAT,HITS,BTAVG
      HRATE   ATBAT,HOMRS,RATIO
      MPY     DBLES,2,DBLES
      MPY     TRPLS,3,TRPLS
      MPY     HOMRS,4,HOMRS
      SUM     1,4,SNGLS,HITS
      BA      ATBAT,HITS,SPCT
      BINDC   BTAVG,WKBUF
      MOVE    WKBUF+2,CC50,4
      BINDC   SPCT,WKBUF
      MOVE    WKBUF+2,CC55,4
      BINDC   RATIO,WKBUF
      MOVE    WKBUF+2,CC60,4
      SPRSS   /2000,0,CC25,3
      SPRSS   /2000,0,CC30,3
      SPRSS   /2000,0,CC35,3
      SPRSS   /2000,0,CC40,3
      SPRSS   /2000,0,CC45,3
      SPRSS   /2000,0,CC50,1
      SPRSS   /2000,0,CC55,1
      SPRSS   /2000,0,CC60,3
      HOLPR   1,COBUF+1,COBUF+1,80
      LDX  L1 CDBUF     GET ADDRESS OF CDBUF
      STX  L1 PR000     SET UP I/O AREA PARAMETER
      LDX   1 40        PRINT WORD COUNT
      STX   1 CDBUF     STORE PRINT WORD COUNT
      PRINT   PRLST     PRINT DATA
      BSC  L  READ      BR TO PROCESS NEXT CARD
CDLST LLIST   /1000,CDBUF,80 GENERATE CARD I/O LIST
CDBUF CDEQU             GENERAL CARD EQUATES
WKBUF DC      /A000
      DC      /2000
      BSS     4         WORK AREA FOR DCBIN SUBR
```

```
ATBAT DC      *-*       NUMBER OF AT-BATS
SNGLS DC      *-*       NUMBER OF SINGLES
DBLES DC      *-*       NUMBER OF DOUBLES
TRPLS DC      *-*       NUMBER OF TRIPLES
HOMRS DC      *-*       NUMBER OF HOME RUNS
HITS  DC      *-*       NUMBER OF HITS
BTAVG DC      *-*       BATTING AVERAGE
SPCT  DC      *-*       SLUGGING PERCENTAGE
RATIO DC      *-*       HOME RUN RATIO
PRLST SLIST   /2100,MES1,,PROOO PRINT I/O LIST
MES1  MSG1    (**EXECUTE SAMPLE PROGRAM**'E),1
MES2  MSG2 X  (NAME'21SAB'3S1B'3S2B'3S3B'3SHR' )
              ('2SBAVG'SSPCT'SHRR'E),1
MES3  MSG1    (***END OF JOB***'E),1
END   LDX  L1 MES3      GET ADDRESS OF MES3
      STX  L1 PROOO     SET UP I/O AREA PARAMETER
      PRINT   PRLST     PRINT END OF JOB MESSAGE
      CALL    EXIT      END EXECUTION
      END     START     END OF ASSEMBLY
```

Mr. Jones then submitted the coding punched on cards to his system operator and requested that the job be performed and a listing of the operation supplied. The following listing shows the assembly and execution of his program. (The explanation of the coding is given to the side of the listing; the macro instructions are enclosed in rectangles.)

```
// ASM SAMPL   00 JAN 00 00.267 HRS
*LIST
*MACLIB JONES
                START PRINT   PRLST      PRINT FIRST LINE (MES1)
0000 30 176558D5 +START CALL  PRNTN
0002 1  01D6     +      DC    PRLST
0003 01 C4000106 +      LD  L PRLST
0005 01 4C20D003 +      BSC L *-4,Z
0007 01 650001ED        LDX L1 MES2      GET ADDRESS OF MES2
0009 01 6D00010E        STX L1 PROOO     SET UP I/O AREA PARAMETER
                PRINT   PRLST            PRINT OUTPUT HEADINGS
0008 30 176558D5 +      CALL  PRNTN
000D 1  01D6     +      DC    PRLST
000E 01 C4000106 +      LD  L PRLST
0010 01 4C20000E +      BSC L *-4,Z
0012 0  6150     READ   LDX   1 80
0013 01 6D000176        STX L1 CDBUF
                RDCRD   CDLST            READ A DATA CARD
0015 30 03059115 +      CALL  CARDN
0017 1  016D     +      DC    CDLST
0018 01 C400016D +      LD  L CDLST
001A 01 4C200018 +      BSC L *-4,Z
                CHECK   CC1,/4220,END
001C 01 C4000177 +      LD  L CC1
001E 0  F003     +      EOR   A0001
001F 01 4C1A0216 +      BSC L ENO,+-
0021 0  7001     +      MDX   *+1
0022 0  4220     +A0001 DC     /4220

                MOVE    CC25,WKBUF+2,4
0023 01 6500018F +      LDX L1 CC25
0025 01 660001C9 +      LDX L2 WKBUF+2
0027 0  6304     +      LDX   3 4
0028 0  C100     +AA002 LD    1 0
0029 0  D200     +      STD   2 0
002A 0  7101     +      MDX   1 1
002B 0  7201     +      MDX   2 1
002C 0  73FF     +      MDX   3 -1
002D 0  70FA     +      MDX     AA002
                OCBIN   WKBUF,ATBAT
002E 30 040C2255 +      CALL  DCBIN
0030 1  01C7     +      DC    WKBUF
0031 1  01CD     +      DC    ATBAT

                MOVE    CC30,WKBUF+2,4
0032 01 65000194 +      LDX L1 CC30
0034 01 660001C9 +      LDX L2 WKBUF+2
0036 0  6304     +      LDX   3 4
0037 0  C100     +AA003 LD    1 0
0038 0  0200     +      STD   2 0
0039 0  7101     +      MDX   1 1
003A 0  7201     +      MDX   2 1
003B 0  73FF     +      MDX   3 -1
003C 0  70FA     +      MDX     AA003
                OCBIN   WKBUF,SNGLS
003D 30 040C2255 +      CALL  DCBIN
003F 1  01C7     +      DC    WKBUF
0040 1  01CE     +      DC    SNGLS

                MOVE    CC35,WKBUF+2,4
0041 01 65000199 +      LDX L1 CC35
0043 01 660001C9 +      LDX L2 WKBUF+2
0045 0  6304     +      LDX   3 4
0046 0  C100     +AA004 LD    1 0
0047 0  0200     +      STD   2 0
0048 0  7101     +      MDX   1 1
0049 0  7201     +      MDX   2 1
004A 0  73FF     +      MDX   3 -1
004B 0  70FA     +      MDX     AA004
                DCBIN   WKBUF,DBLES
004C 30 040C2255 +      CALL  DCBIN
004E 1  01C7     +      DC    WKBUF
004F 1  01CF     +      DC    DBLES
```

All macros in SYSTM and JONES may be used in this assembly.

Print the line **EXECUTE SAMPLE PROGRAM**.

Change the print I/O area parameter to point to the word count of the next line to be printed.

Print the second line: NAME,AB,1B, etc.

Assure that the word count preceding the card input buffer is 80. This is not necessary for the first card to be read, but is necessary for the reading of all ensuing cards since a 40 will be stored at CDBUF before printing a data card.

Read a data card into CDBUF.

Check the card just read for an asterisk in column 1 (* = /4220 in IBM card code) and branch to END if an asterisk is found.

WKBUF and WKBUF+1 contain an IBM card code (Hollerith) plus and zero rexpectively. The four columns (25 through 28) for at-bats are moved into the four-word buffer following WKBUF.

Converts the number of at-bats to a 16-bit binary value and stores the result into location ATBAT. (See the 1800 MPX Subroutine Library manual, Order Number GC26-3724,for further information on DCBIN.)

Move and convert the four columns for singles (30 through 33) and store the result into location SNGLS.

Move and convert the four columns for doubles (35 through 38) and store the result into location DBLES.

```
                        MOVE    CC40,WKBUF+2,4
0050 01 6500019E  +     LDX  L1 CC40
0052 01 660001C9  +     LDX  L2 WKBUF+2
0054 0  6304      +     LDX  3  4
0055 0  C100      +AA005 LD    1  0
0056 0  0200      +     STO  2  0
0057 0  7101      +     MDX  1  1
0058 0  7201      +     MDX  2  1
0059 0  73FF      +     MDX  3  -1
005A 0  70FA      +     MDX     AA005
                        OCBIN   WKBUF,TRPLS
005B 30 040C2255  +     CALL    OCBIN
0050 1  01C7      +     DC      WKBUF
005E 1  0100      +     DC      TRPLS
                        MOVE    CC45,WKBUF+2,4
005F 01 650001A3  +     LDX  L1 CC45
0061 01 660001C9  +     LDX  L2 WKBUF+2
0063 0  6304      +     LDX  3  4
0064 0  C100      +AA006 LD    1  0
0065 0  0200      +     STO  2  0
0066 0  7101      +     MDX  1  1
0067 0  7201      +     MDX  2  1
0068 0  73FF      +     MDX  3  -1
0069 0  70FA      +     MDX     AA006
                        OCBIN   WKBUF,HOMRS
006A 30 040C2255  +     CALL    OCBIN
006C 1  01C7      +     DC      WKBUF
006D 1  0101      +     DC      HOMRS
                        SUM     1,4,SNGLS,HITS
006E 0  61FC      +     LDX  1  -4
006F 0  1010      +     SLA     16
0070 01 850001D2  +B0007 A     L1 SNGLS+4
0072 01 D4000102  +     STO  L  HITS
0074 0  7101      +     MDX  1  1
0075 0  70FA      +     MDX     B0007
                        BA      ATBAT,HITS,BTAVG
0076 01 C40001D2  +     LD   L  HITS
0078 0  A005      +     M       00008
0079 01 AC0001CD  +     D    L  ATBAT
007B 01 D40001D3  +     STO  L  BTAVG
007D 0  7001      +     MDX     *+1
007E 0  03E8      +D0008 DC     1000
                        HRATE   ATBAT,HOMRS,RATIO
007F 01 C40001CD  +     LD   L  ATBAT
0081 0  18D0      +     RTE     16
0082 0  1010      +     SLA     16
0083 01 AC0001D1  +     D    L  HOMRS
0085 01 D40001D5  +     STO  L  RATIO
                        MPY     DBLES,2,DBLES
0087 01 C40001CF  +     LD   L  DBLES
0089 0  A004      +     M       C0009
008A 0  1090      +     SLT     16
008B 01 D40001CF  +     STO  L  DBLES
008D 0  7001      +     MDX     *+1
008E 0  0002      +C0009 DC     2
                        MPY     TRPLS,3,TRPLS
008F 01 C4000100  +     LD   L  TRPLS
0091 0  A004      +     M       C0010
0092 0  1090      +     SLT     16
0093 01 D40001D0  +     STO  L  TRPLS
0095 0  7001      +     MDX     *+1
0096 0  0003      +C0010 DC     3
                        MPY     HOMRS,4,HOMRS
0097 01 C4000101  +     LD   L  HOMRS
0099 0  A004      +     M       C0011
009A 0  1090      +     SLT     16
009B 01 D40001D1  +     STO  L  HOMRS
009D 0  7001      +     MDX     *+1
009E 0  0004      +C0011 DC     4
                        SUM     1,4,SNGLS,HITS
009F 0  61FC      +     LDX  1  -4
00A0 0  1010      +     SLA     16
00A1 01 850001D2  +B0012 A     L1 SNGLS+4
00A3 01 D40001D2  +     STO  L  HITS
00A5 0  7101      +     MDX  1  1
00A6 0  70FA      +     MDX     B0012
                        BA      ATBAT,HITS,SPCT
00A7 01 C40001D2  +     LD   L  HITS
00A9 0  A005      +     M       D0013
00AA 01 AC0001CD  +     D    L  ATBAT
00AC 01 D40001D4  +     STO  L  SPCT
00AE 0  7001      +     MDX     *+1
00AF 0  03E8      +D0013 DC     1000
                        BINDC   BTAVG,WKBUF
00B0 30 02255103  +     CALL    BINDC
00B2 1  01D3      +     DC      BTAVG
00B3 1  01C7      +     DC      WKBUF
                        MOVE    WKBUF+2,CC50,4
00B4 01 650001C9  +     LDX  L1 WKBUF+2
00B6 01 660001A8  +     LDX  L2 CC50
00B8 0  6304      +     LDX  3  4
00B9 0  C100      +AA014 LD    1  0
00BA 0  0200      +     STO  2  0
00BB 0  7101      +     MDX  1  1
00BC 0  7201      +     MDX  2  1
00BD 0  73FF      +     MDX  3  -1
00BE 0  70FA      +     MDX     AA014
                        BINDC   SPCT,WKBUF
00BF 30 02255103  +     CALL    BINDC
00C1 1  01D4      +     DC      SPCT
00C2 1  01C7      +     DC      WKBUF
                        MOVE    WKBUF+2,CC55,4
00C3 01 650001C9  +     LDX  L1 WKBUF+2
00C5 01 660001AD  +     LDX  L2 CC55
00C7 0  6304      +     LDX  3  4
00C8 0  C100      +AA015 LD    1  0
00C9 0  0200      +     STO  2  0
00CA 0  7101      +     MDX  1  1
00CB 0  7201      +     MDX  2  1
00CC 0  73FF      +     MDX  3  -1
00CD 0  70FA      +     MDX     AA015
```

Move and convert the four columns for triples (40 through 43) and store the result into location TRPLS.

Move and convert the four columns for homers (45 through 48) and store the result into location HOMRS.

Compute SNGLE+DBLS+TRPLS+HOMRS and store the result into location HITS.

Compute the batting average and store the result into location BTAVG.

Compute the home-run ratio and store the result into location RATIO.

In preparing for the slugging percentage calculation, multiply DBLES by 2 and store the result in location DBLES.

Multiply TRPLS by 3 and store the result in TRPLS.

Multiply HOMRS by 4 and store the result in HOMRS.

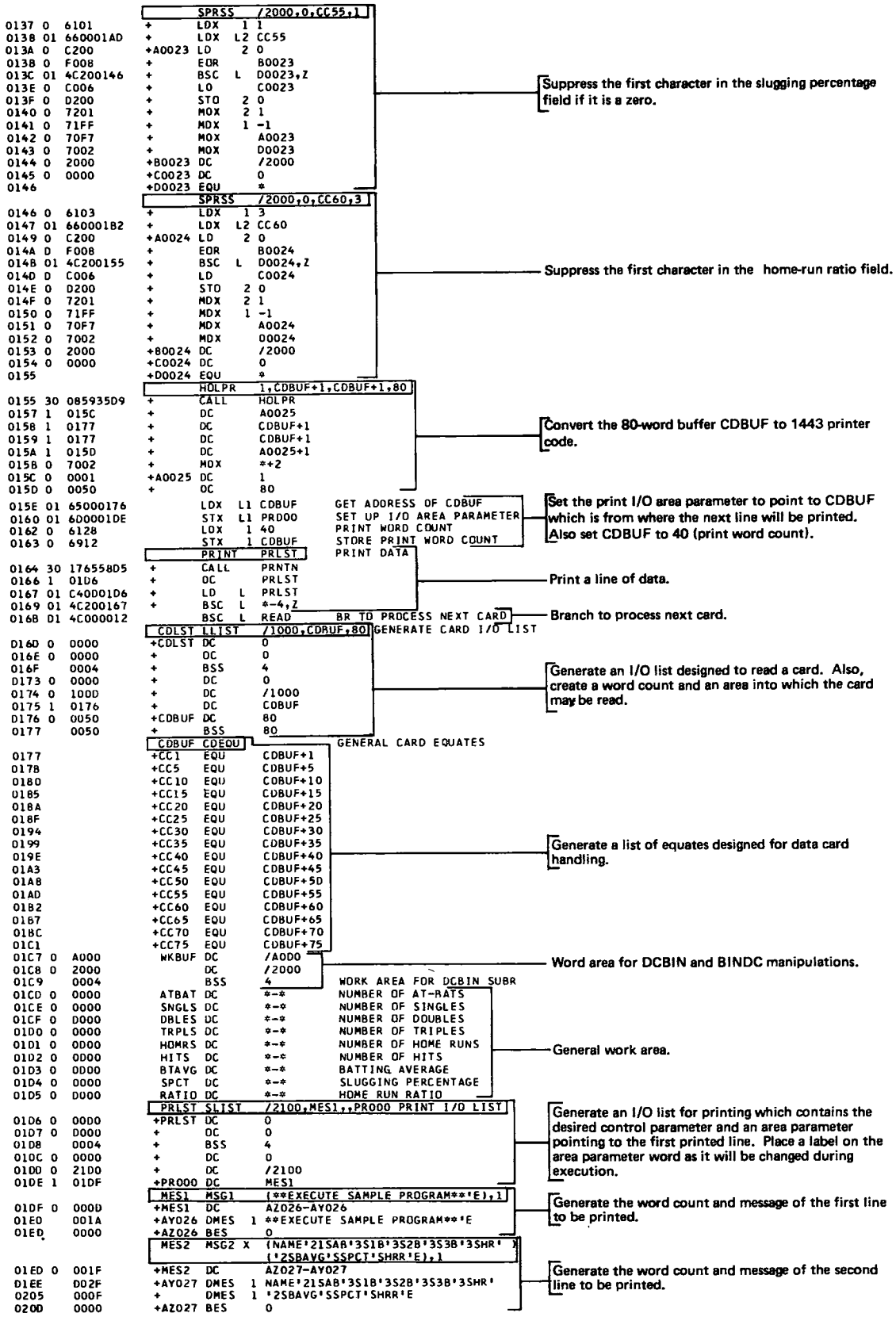
Compute SNGLS+DBLES+TRPLS+HOMRS and store the result into location HITS.

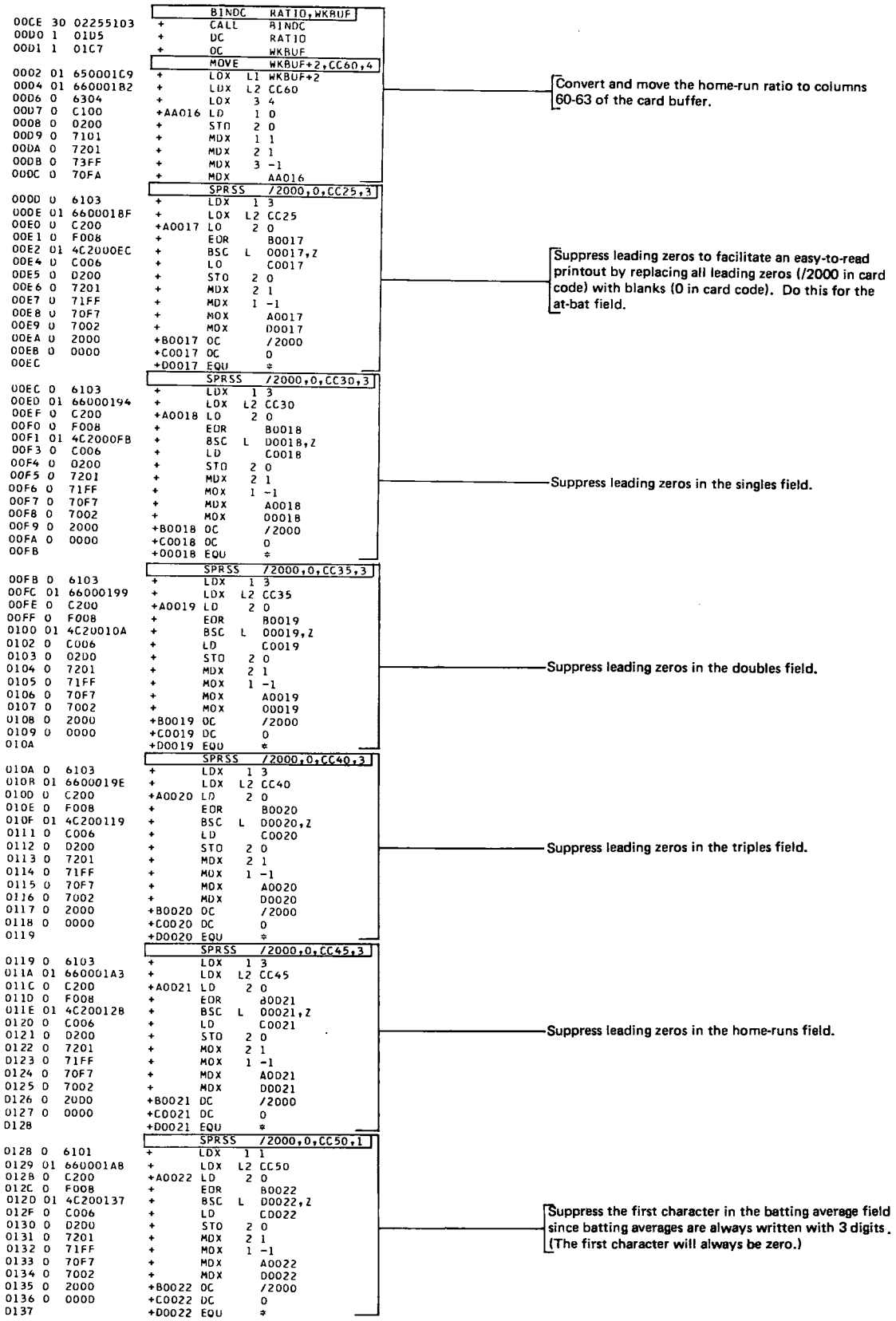
Compute the slugging percentage and store the result into location SPCT.

Convert the 16-bit binary value for BTAVG (batting average) to its IBM card-coded equivalent. Store the 6-word result into location WKBUF. (See 1800 MPX Subroutine Library manual, Order Number GC26-3724).

Move 4 words beginning at WKBUF+2 to words 50-53 of the card buffer. This puts the batting average into the card buffer which will soon be converted to 1443 code and printed.

Convert and move the slugging percentage to columns 55-58 of the card buffer.


SPRSS /2000,0,CC55,1
Suppress the first character in the slugging percentage field if it is a zero.
SPRSS /2000,0,CC60,3
Suppress the first character in the home-run ratio field.
HOLPR 1,CDBUF+1,CDBUF+1,80
Convert the 80-word buffer CDBUF to 1443 printer code.
LDX L1 CDBUF GET ADDRESS OF CDBUF
STX L1 PRD00 SET UP I/O AREA PARAMETER
LDX 1 40 PRINT WORD COUNT
STX 1 CDBUF STORE PRINT WORD COUNT
Set the print I/O area parameter to point to CDBUF which is from where the next line will be printed. Also set CDBUF to 40 (print word count).
PRINT PRLST PRINT DATA
Print a line of data.
BSC L READ BR TO PROCESS NEXT CARD
Branch to process next card.
CDLST LLIST /1000,CDBUF,80 GENERATE CARD I/O LIST
Generate an I/O list designed to read a card. Also, create a word count and an area into which the card may be read.
CDBUF CDEQU GENERAL CARD EQUATES
Generate a list of equates designed for data card handling.
WKBUF DC /A000
Word area for DCBIN and BINDC manipulations.
ATBAT DC *-* NUMBER OF AT-BATS
SNGLS DC *-* NUMBER OF SINGLES
DBLES DC *-* NUMBER OF DOUBLES
TRPLS DC *-* NUMBER OF TRIPLES
HOMRS DC *-* NUMBER OF HOME RUNS
HITS DC *-* NUMBER OF HITS
BTAVG DC *-* BATTING AVERAGE
SPCT DC *-* SLUGGING PERCENTAGE
RATIO DC *-* HOME RUN RATIO
General work area.
PRLST SLIST /2100,MES1,,PR000 PRINT I/O LIST
Generate an I/O list for printing which contains the desired control parameter and an area parameter pointing to the first printed line. Place a label on the area parameter word as it will be changed during execution.
MES1 MSG1 (**EXECUTE SAMPLE PROGRAM**'E),1
Generate the word count and message of the first line to be printed.
MES2 MSG2 X (NAME'21SAB'3S1B'3S2B'3S3B'3SHR')
('2SBAVG'SSPCT'SHRR'E),1
Generate the word count and message of the second line to be printed.


00CE 30 02255103 + BINDC RATIO,WKBUF
00D0 1 01D5 + CALL BINDC
00D1 1 01C7 + DC RATIO
+ DC WKBUF
+ MOVE WKBUF+2,CC60,4
0002 01 650001C9 + LDX L1 WKBUF+2
0004 01 660001B2 + LDX L2 CC60
00D6 0 6304 + LDX 3 4
00D7 0 C100 +AA016 LD 1 0
0008 0 0200 + STO 2 0
00D9 0 7101 + MDX 1 1
00DA 0 7201 + MDX 2 1
00DB 0 73FF + MDX 3 -1
00DC 0 70FA + MDX AA016
+ SPRSS /2000,0,CC25,3
00DD 0 6103 + LDX 1 3
00DE 01 6600018F + LDX L2 CC25
00E0 0 C200 +A0017 LD 2 0
00E1 0 F008 + EOR B0017
00E2 01 4C2000EC + BSC L 00017,Z
00E4 0 C006 + LD C0017
00E5 0 D200 + STO 2 0
00E6 0 7201 + MDX 2 1
00E7 0 71FF + MDX 1 -1
00E8 0 70F7 + MDX A0017
00E9 0 7002 + MDX D0017
00EA 0 2000 +B0017 DC /2000
00EB 0 0000 +C0017 DC 0
00EC +D0017 EQU *
+ SPRSS /2000,0,CC30,3
00EC 0 6103 + LDX 1 3
00ED 01 66000194 + LDX L2 CC30
00EF 0 C200 +A0018 LD 2 0
00F0 0 F008 + EOR B0018
00F1 01 4C2000FB + BSC L D0018,Z
00F3 0 C006 + LD C0018
00F4 0 0200 + STO 2 0
00F5 0 7201 + MDX 2 1
00F6 0 71FF + MDX 1 -1
00F7 0 70F7 + MDX A0018
00F8 0 7002 + MDX 00018
00F9 0 2000 +B0018 DC /2000
00FA 0 0000 +C0018 DC 0
00FB +D0018 EQU *
+ SPRSS /2000,0,CC35,3
00FB 0 6103 + LDX 1 3
00FC 01 66000199 + LDX L2 CC35
00FE 0 C200 +A0019 LD 2 0
00FF 0 F008 + EOR B0019
0100 01 4C20010A + BSC L 00019,Z
0102 0 C006 + LD C0019
0103 0 0200 + STO 2 0
0104 0 7201 + MDX 2 1
0105 0 71FF + MDX 1 -1
0106 0 70F7 + MDX A0019
0107 0 7002 + MDX 00019
0108 0 2000 +B0019 DC /2000
0109 0 0000 +C0019 DC 0
010A +D0019 EQU *
+ SPRSS /2000,0,CC40,3
010A 0 6103 + LDX 1 3
010B 01 6600019E + LDX L2 CC40
010D 0 C200 +A0020 LD 2 0
010E 0 F008 + EOR B0020
010F 01 4C200119 + BSC L D0020,Z
0111 0 C006 + LD C0020
0112 0 D200 + STO 2 0
0113 0 7201 + MDX 2 1
0114 0 71FF + MDX 1 -1
0115 0 70F7 + MDX A0020
0116 0 7002 + MDX D0020
0117 0 2000 +B0020 DC /2000
0118 0 0000 +C0020 DC 0
0119 +D0020 EQU *
+ SPRSS /2000,0,CC45,3
0119 0 6103 + LDX 1 3
011A 01 660001A3 + LDX L2 CC45
011C 0 C200 +A0021 LD 2 0
011D 0 F008 + EOR B0021
011E 01 4C200128 + BSC L D0021,Z
0120 0 C006 + LD C0021
0121 0 D200 + STO 2 0
0122 0 7201 + MDX 2 1
0123 0 71FF + MDX 1 -1
0124 0 70F7 + MDX A0021
0125 0 7002 + MDX D0021
0126 0 2000 +B0021 DC /2000
0127 0 0000 +C0021 DC 0
0128 +D0021 EQU *
+ SPRSS /2000,0,CC50,1
0128 0 6101 + LDX 1 1
0129 01 660001A8 + LDX L2 CC50
012B 0 C200 +A0022 LD 2 0
012C 0 F008 + EOR B0022
012D 01 4C200137 + BSC L D0022,Z
012F 0 C006 + LD C0022
0130 0 D200 + STO 2 0
0131 0 7201 + MDX 2 1
0132 0 71FF + MDX 1 -1
0133 0 70F7 + MDX A0022
0134 0 7002 + MDX D0022
0135 0 2000 +B0022 DC /2000
0136 0 0000 +C0022 DC 0
0137 +D0022 EQU *
Convert and move the home-run ratio to columns 60-63 of the card buffer.
Suppress leading zeros to facilitate an easy-to-read printout by replacing all leading zeros (/2000 in card code) with blanks (0 in card code). Do this for the at-bat field.
Suppress leading zeros in the singles field.
Suppress leading zeros in the doubles field.
Suppress leading zeros in the triples field.
Suppress leading zeros in the home-runs field.
Suppress the first character in the batting average field since batting averages are always written with 3 digits. (The first character will always be zero.)

```
                MES3  MSG1      (***END OF JOB***'E),1
020D 0  0008    +MES3 DC        AZ028-AY028
020E    0010    +AY028 DMES   1 ***END OF JOB***'E
0216    0000    +AZ028 BES      0
0216 01 6500020D  END  LDX  L1 MES3      GET ADDRESS OF MES3
0218 01 6D00010E       STX  L1 PR000     SET UP I/O AREA PARAMETER
                       PRINT   PRLST     PRINT END OF JOB MESSAGE
021A 30 17655805    +  CALL    PRNTN
021C 1  0106        +  DC      PRLST
021D 01 C40001D6    +  LD   L  PRLST
021F 01 4C200210    +  BSC  L  *-4,Z
0221 30 059C98C0       CALL    EXIT      END EXECUTION
0224    0000           END     START     END OF ASSEMBLY

000 ERROR(S) AND 000 WARNING(S) IN ABOVE ASSEMBLY.
SAMPL
DMP FUNCTION COMPLETED
// XEQ SAMPL L   00 JAN 00 00.387 HRS
*CCEND


MPX, BUILD SAMPL

CORE LOAD  MAP
TYPE NAME  ARG1  ARG2

*COM TABLE 8002  0012
*FIO TABLE 8014  001E
*CNT TABLE 8032  0004
MAIN SAMPL 8036
CLNT SAMPL 8034
CALL CARDN 825A
CALL DCBIN 83E4
CALL BINDC 8446
CALL HOLPR 849A
CALL PRT   8570
CORE       85BC  7A44
MPX, SAMPL LD XQ
// XEQ SAMPL
*CCEND
TOM BAIRD              0642 0048 0003 0009 0026
ED BATTLES             0132 0027 0017 0000 0002
MAMIE BEARD            0614 0113 0006 0007 0004
AL BERGLUND            0599 0173 0021 0003 0009
ED CAMPBELL            0032 0011 0000 0001 0000
JIM CROSSLEY           0535 0156 0009 0000 0002
KEVE GABBERT           0587 0169 0003 0001 0010
STEVE GRAHAM           0602 0183 0011 0003 0017
MORRIS GROVE           0649 0233 0025 0001 0032
DON HADERLE            0492 0138 0006 0002 0007
BURT HANNAY            0545 0176 0007 0002 0005
WILLIE HATHORN         0584 0178 0008 0007 0009
BILL HO                0614 0111 0013 0000 0000
RON HOLMES             0476 0138 0008 0000 0014
MILT KHOOBYARIAN       0623 0178 0012 0004 0006
GENE LESTER            0369 0003 0001 0000 0000
MARILYN MARPLES        0062 0017 0004 0000 0001
BOB MAY                0542 0155 0006 0001 0002
TOM PIERCE             0207 0054 0002 0001 0000
RALPH PIPITONE         0613 0169 0011 0004 0003
MARILYN TAGHON         0612 0193 0009 0005 0006
ROBERT TARBUTTON       0575 0169 0002 0000 0012
*END OF DATA CARDS
**EXECUTE SAMPLE PROGRAM**
NAME                    AB   1B   2B   3B   HR  BAVG SPCT HRR
TOM BAIRD              642   48    3    9   26  133  288   24
ED BATTLES             132   27   17    0    2  348  522   66
MAMIE BEARD            614  113    6    7    4  211  263  153
AL BERGLUND            599  173   21    3    9  343  434   66
ED CAMPBELL             32   11    0    1    0  375  437    0
JIM CROSSLEY           535  156    9    0    2  312  340  267
KEVE GABBERT           587  169    3    1   10  311  371   58
STEVE GRAHAM           602  183   11    3   17  355  468   35
MORRIS GROVE           649  233   25    1   32  448  637   20
DON HADERLE            492  138    6    2    7  310  373   70
BURT HANNAY            545  176    7    2    5  348  396  109
WILLIE HATHORN         584  178    8    7    9  345  429   64
BILL HO                614  111   13    0    0  201  223    0
RON HOLMES             476  138    8    0   14  336  441   34
MILT KHOOBYARIAN       623  178   12    4    6  321  382  103
GENE LESTER            369    3    1    0    0  010  013    0
MARILYN MARPLES         62   17    4    0    1  354  467   62
BOB MAY                542  155    6    1    2  302  328  271
TOM PIERCE             207   54    2    1    0  275  294    0
RALPH PIPITONE         613  169   11    4    3  305  350  204
MARILYN TAGHON         612  193    9    5    6  348  408  102
ROBERT TARBUTTON       575  169    2    0   12  318  384   47
***END OF JOB***
```

Generate the word count and message of the last line to be printed.

Change the print I/O area parameter to point to the word count of the next line to be printed.

Print the last line.

Terminate execution.

These are the data cards that Mr. Jones was given.

From this example you should realize how the use of macros with meaningful names helped Mr. Jones and his fellow workers make efficient use of their system. Similarly, you can design your macros and libraries to aid your programmers and others who must work with the programmers.

This page intentionally left blank.

The Macro Update Program (MUP) assists you in maintaining macro libraries. It performs the following functions:

- Initializes disk space for macro libraries.
- Adds macros to libraries.
- Deletes macros from libraries and reclaims the space they occupied.
- Joins macro libraries, physically or logically.
- Renames macros.
- Obtains a listing of the contents of macro libraries by macro name or by macro name and statement.
- Inserts or deletes statement(s) within a macro.
- Provides macro definition source statements on cards.

calling MUP

To call the Macro Update Program, you must first load the Disk Utility Program (for DM2) or the Disk Management Program (for MPX) into main storage. After it has been loaded, you use an *MACRO UPDATE control statement to call MUP. Following this statement, you should use MUP control statements to indicate the functions you want to perform.

field specifications

The function field must begin in column one, and at least one blank is required to separate it from the operand field. If you leave the first column blank, the statement will be ignored. If you want to specify more than one operand, you must separate the operands by commas and leave no blanks within or between them. Any unused columns in MUP control statements are reserved for system use. With the exception of the NAME function (described later in this chapter), control statement continuation is not allowed in MUP.

Note that once a Macro Update function has been started, it should not be aborted, because an incompleted modification may cause the library to be unusable.

All special characters used in the Macro Update Program must conform to the character code summaries as listed in the 1130 Assembler Language manual, Order Number GC26-5927, or the 1800 Assembler Language manual, Order Number GC26-5882.

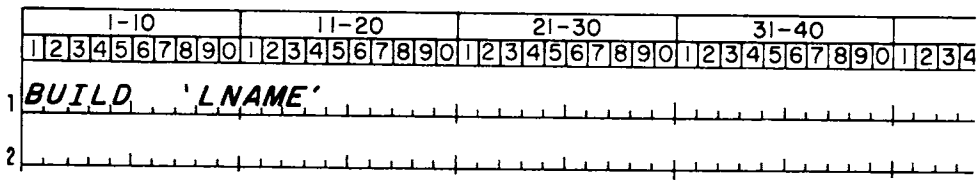
Initializing Disk Space

reserve space

Before initializing disk space, you must reserve a data file to serve as your macro library. This can be accomplished by means of an *DFILE statement or an *STOREDATA statement. For a detailed description of *DFILE and *STOREDATA, see the Programming and Operator's Guide (for DM2), Order Number GC26-3717, or the Programmer's Guide (for MPX), Order Number GC26-3720.

To initialize disk space for a macro library, you must use a BUILD statement. The library name in the BUILD statement must be the same as the one defined in the *DFILE or *STOREDATA statement. After you have named a library, you must use that library's name in all LIB, BUILD, JOIN, or CONCAT statements (described later) that refer to that library. You must initialize disk space before requesting the Macro Assembler or the Macro Update Program to operate on a specific library. The format of a BUILD statement is as follows:

BUILD format

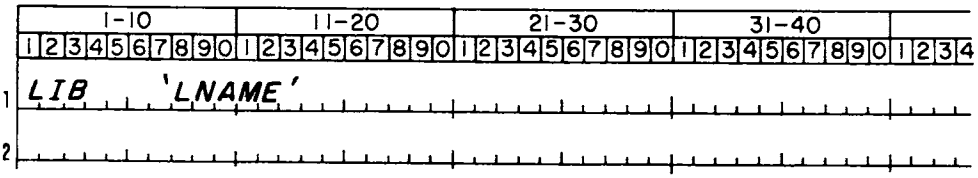


LNAME is the name of the macro library that was reserved by the *DFILE or *STOREDATA statement (discussed above). LNAME is a 1-5 character name for the macro library. The apostrophes are delimiters and, as such, are required by MUP for the names of all macros and libraries. You can use alphabetic characters A-Z and the characters 0-9, #, @, and $ within your library name. The digits 0-9 may not be used as the first character. A library name is considered a symbol, and therefore, it must conform to the rules for symbols as stated in the Assembler Language manual, Order Number GC26-5927 (for the 1130) or Order Number GC26-5882 (for the 1800). If LNAME applies to a previously initialized library containing macros, the function purges the library and reinitializes it.

Specifying the Macro Library

To specify the macro library to be operated on, you use the LIB statement. The format of a LIB statement is as follows:

LIB format



LNAME (discussed above) is the name for the macro library.

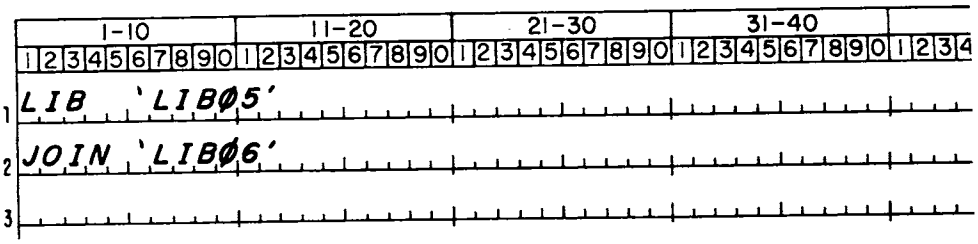
Joining Macro Libraries Physically

If you want to physically join a macro library to the end of the library specified in the last BUILD or LIB statement,

use JOIN

you use a JOIN statement. For example, if you wanted to physically join LIB06 to LIB05, you would use the following statements.

format



contents unchanged

These statements cause the contents of LIB06 to be added (physically copied) to the end of the contents of LIB05. This does not change the contents of library LIB06.

notes

If the first library cannot accommodate the library being joined to it, the JOIN operation is suppressed. Neither library is changed and processing continues with the next LIB or BUILD statement. If you specify a new library name in a BUILD and then join an existing library to it, you physically copy the existing library.

Joining Macro Libraries Logically

use CONCAT

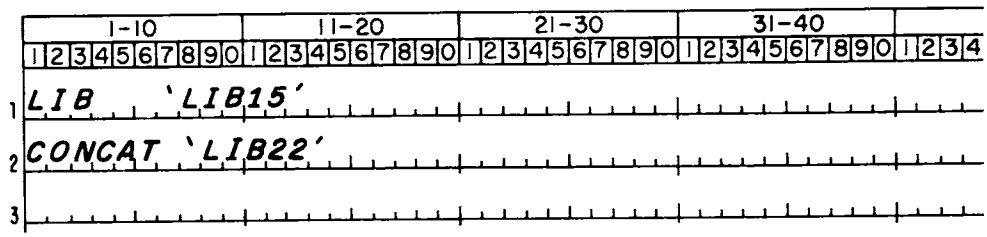
If you want to logically join a macro library to the library specified in the last BUILD or LIB statement, you use a CONCAT statement. This statement allows you to maintain

purpose

individual libraries, and then unite them for assembly purposes without using additional disk space.

For example, if you wanted to logically join LIB22 to LIB15, you would use the following statements.

format



multiple concaten-ation

These statements cause both libraries to be available for assembly purposes when LIB15 is referenced by the Macro Assembler. However, in any Macro Update Program operation, only the library named on the last LIB or BUILD statement is operated upon.

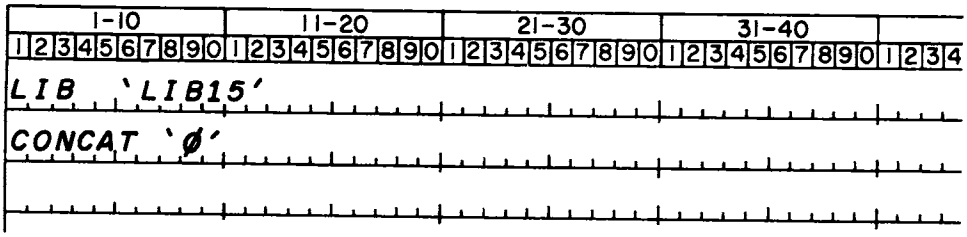
It is possible to concatenate a multiplicity of libraries, so that several libraries may be available to the Macro Assembler even though the assembly references only one. You

would perform this multiple concatenation by concatenating library B to library A, library C to B, library D to C, library E to D, and so on until all the libraries that you wanted were linked together. You can concatenate only one library to any other library, but you can concatenate up to a total of 16 libraries, making a total of up to 17 available for assembly purposes. If the Macro Assembler does not find a macro and has searched through 17 libraries, the statement containing the name of the macro will be flagged as an op code violation and the assembly will continue.

disconnect

If you want to disconnnect a library that has been concatenated to another library, you can use the following statements.

format

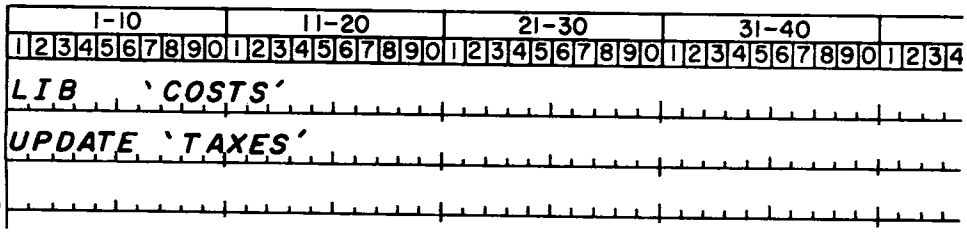


The CONCAT statement would cause the library (LIB22) that has been concatenated to library LIB15 to be disconnected. The physical contents of the libraries remain unchanged.

Updating a Macro in a Library

If you want to alter a macro that has been stored in the library specified in the last BUILD or LIB statement, you specify that macro with the UPDATE function. An example of specifying the macro TAXES might be as follows.

use UPDATE



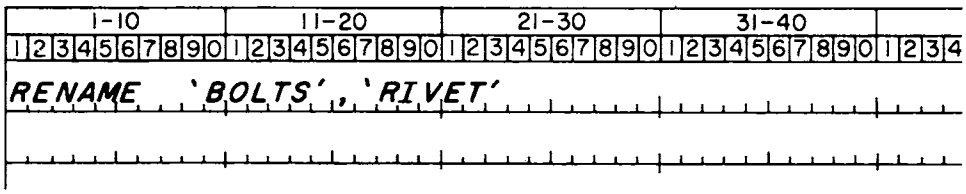
The UPDATE statement is normally followed by an INSERT or DELETE statement. Descriptions of these two statements are discussed under "Inserting Statements in a Macro" and "Deleting Statements From a Macro."

Renaming a Macro in a Library

use RENAME

If you want to rename a macro in a library, you use the RENAME statement. Once you have renamed a macro, the original name is lost, and the macro can be referenced only by its new name. The format for the RENAME statement is as follows:

format

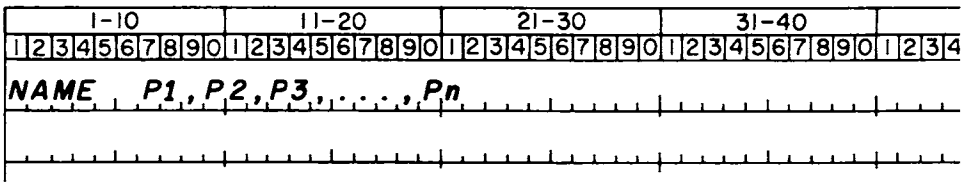


This statement would cause the macro BOLTS to be renamed RIVET. BOLTS represents the name of the current macro; it must be enclosed in apostrophes and separated from the new name by a comma. RIVET represents the new name for the macro. The macro must now be referenced by this new name. BOLTS can now be used as a name for another macro. The RENAME statement may be followed by INSERT or DELETE statements.

Defining a Macro During a Macro Update Run

You can define a macro during a macro update run by using essentially the same method you use during an assembly. The differences are that you don't use MAC or SMAC and MEND statements, and you don't use a definition prototype statement. Instead, you use a NAME statement to name the parameters to be used in a subsequent definition and an ADD statement to name the macro (discussed below). The format of the NAME statement is as follows:

NAME format



P1 is the symbol for parameter 1, P2 is the symbol for parameter 2, and so on.

Note that parameter 1 is the parameter used in the label field of the macro call. Parameter 2 would be the first parameter in the operand field. If you need more than one record for all your parameters (limited to 20 possible), use as many NAME records as needed immediately following the first, and continue the parameters on these records. In

such a case, a comma must not follow the last parameter on each record.

If your NAME statement does not have sufficient parameter names for the parameters in the macro being processed, one of the following will occur.

- On input, the additional parameters will be assumed to be standard Macro Assembler variables instead of macro parameters.

- On output, the operation will be aborted and a D117 error message printed. This also occurs if a parameter that is used in the format or tag fields is given a name that is greater than one character.

- On listing, extra parameters will be replaced by // N where N is the number (1-20) of the parameter. A D117 error message will be printed. Note that the parameter number may be truncated if it exceeds the field length.

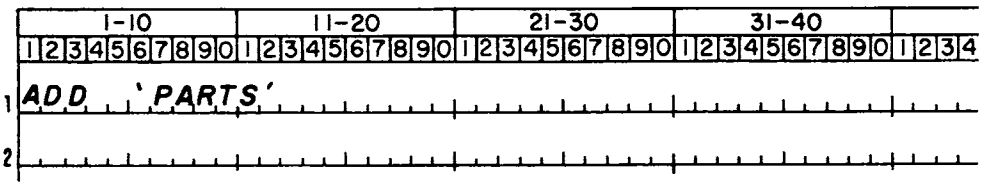
The names specified in the NAME statement are used in all subsequent operations until the next LIB or BUILD statement is encountered or another NAME statement is read.

ADD STATEMENT

then ADD statement

After the NAME statement, you use the ADD statement. The ADD statement adds the macro to the library specified in the last LIB or BUILD. The ADD statement in conjunction with the NAME statement performs the same function as the definition prototype statement of the Macro Assembler definition. The format of the ADD statement is as follows:

format



This statement causes the macro PARTS to be added to the end of the macro library specified in the last BUILD or LIB statement.

Following the ADD statement, you place the assembler-language source statements that you want to include in the definition. The macro being thus defined by the ADD function is terminated by the occurrence of a MUP control statement. An example of how to define a macro during a macro update run is as follows:

```
                1-10        11-20       21-30       31-40
      1234567890 1234567890 1234567890 1234567890 1234
1     *MACRO UPDATE        call to Macro Update Program
2     LIB 'QUALI'          specifies library
3     NAME LABLE,T1,T2     specifies parameters (described above)
4     ADD 'TESTS'          adds and names macro
5                     LD        T1
6                     STO       T2
7     ENDUP                terminates MUP run
8
```

lack of room

If there is not sufficient space in the library to accommodate the macro you want to store, the macro is not added. Processing continues with the next LIB or BUILD statement; the library is not changed by this occurrence.

If you define another macro within a macro definition, then the MAC or SMAC and MEND statements of the nested macro are included in the definition of the outer macro. Thus, if a macro is defined with an ADD statement and its source statements include a MAC or SMAC and MEND statement, then every time it is called in an assembly, a macro definition is generated.

The definition prototype statement cannot be used in a macro defined by the ADD function except in conjunction with a MAC or SMAC statement. If one is present, no error is diagnosed. Note also that macro source statements stored during a MUP run are not diagnosed for errors.

Deleting a Macro From a Library

use PURGE

If you want to delete a macro from the library named in the last LIB or BUILD statement, you use a PURGE statement. This function deletes the macro that is specified, and automatically reclaims the space it occupied. The format of a PURGE statement is as follows:

format

```
                1-10        11-20       21-30       31-40
      1234567890 1234567890 1234567890 1234567890 1234
1     PURGE 'WAGES'
2
```

This statement causes the macro WAGES to be deleted from the library named in the last LIB or BUILD statement and reclaims the space WAGES occupied.

PURGE vs. PURG

The Macro Update Program PURGE statement should not be confused with the Macro Assembler PURG statement. The Macro Assembler PURG statement does not automatically reclaim the space occupied by the macro named in that statement. Instead, the space is reclaimed by running the Macro Update Program. Any macro update run affecting a particular library will reclaim the space occupied by a macro deleted from that library with the Macro Assembler PURG statement. The PURG and PURGE statements cause operations to occur in regard to the specific library named in the last LIB, BUILD, or *MACLIB statement. They do not affect any other library.

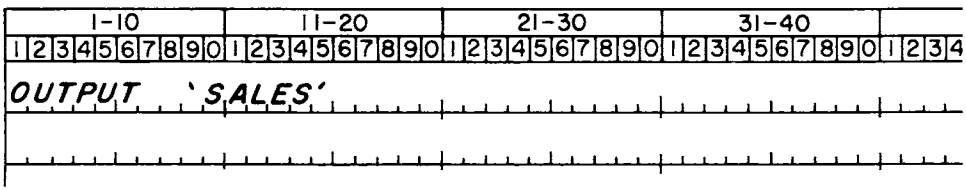
Punching Source Statements

use OUTPUT

If you want to punch the source statements of a specified macro definition, you can use the OUTPUT statement. This statement must have been preceded by a NAME statement. Blank cards must be available for punching as soon as the OUTPUT statement is read.

Care should be taken in preparing the NAME statement since the OUTPUT function will be aborted if a parameter is defined incorrectly or is left undefined (see NAME statement described previously). The definition to be punched must be part of the library specified in the last LIB or BUILD statement. The format of the OUTPUT statement is as follows:

format



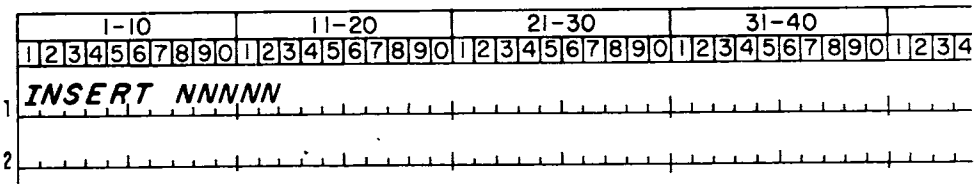
SALES is the name of the macro definition to be punched. This statement also causes an ADD 'SALES' statement to be punched prior to the source statement to facilitate loading of the definition at a later time. The source statements will contain an identification (first three characters of the macro name) and a sequence number in columns 73-80.

Inserting a Statement in a Macro

use INSERT

If you want to insert additional macro definition source statements into a macro, you can use the INSERT statement. This statement must be preceded by a RENAME or UPDATE statement and a NAME statement (discussed previously) that specifies the macro to be modified and its parameters, and it must be followed by the source statements to be inserted. The format of an INSERT statement is as follows:

format



NNNNN is a decimal integer (maximum of 32767) up to five digits long. It references a macro definition source statement sequence number. The source statements are inserted after NNNNN; so, if you want the statements inserted before any other statements in that macro, you must specify NNNNN as zero.

Whenever MUP or the Macro Assembler prints macro definition source statements, five-digit decimal sequence numbers are printed to the left of each statement. These sequence numbers are referenced by INSERT and DELETE statements. Any statements inserted into a library by the ADD, INSERT, or DELETE (described in the next section) function are flagged with an X when the definition is printed. Macro definition source statements are automatically sequenced when the ADD function is used, and resequenced when the INSERT or DELETE functions are used.

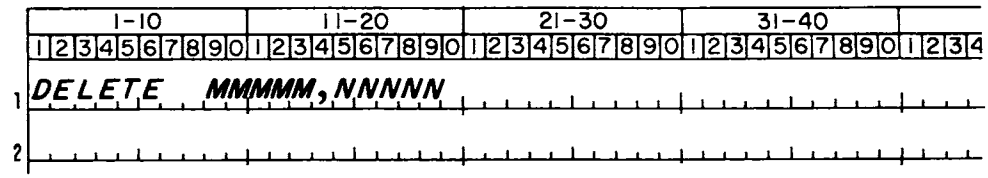
notes

You can insert only as many statements as the library has room for. In other words, if you have 25 statements to add to a macro, and there is enough space to accommodate only 15 of those statements, those fifteen statements will be added. A D103 LIBRARY OVERFLOW message will be printed; processing will continue with the next LIB or BUILD statement. If you want to include the ten statements left out of your macro after regaining enough space to accommodate them, you will have to alter your INSERT control statement because of the resequencing of the statements in the macro library that has occurred.

Deleting a Statement From a Macro

If you want to delete one or more statements from a macro, you can use the DELETE statement. The DELETE statement must be preceded by a RENAME or UPDATE statement that specifies the macro to be altered and a NAME statement (discussed previously) that specifies parameter names used for alteration. The format of a DELETE statement is as follows:

format



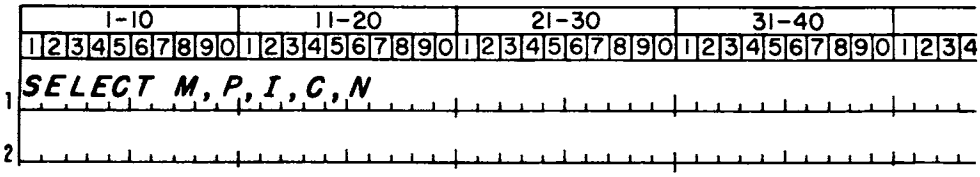
MMMMM is the sequence number of the first source statement to be deleted and NNNNN is the sequence number of the last source statement to be deleted. This statement would cause the statements from MMMMM through NNNNN inclusive to be deleted. If you want to delete a single source statement, you can leave the comma and second parameter off or specify the first sequence number again.

A DELETE statement may be followed by source statements to be inserted in place of the deleted statements. You do not need to use an INSERT statement to perform this function. You may insert more or fewer statements than you deleted. If you insert statement(s), you can insert only as many as the library has room for. The Macro Update Program will insert as many as will fit, print an error message, and continue processing with the next LIB or BUILD statement.

Obtaining a Listing of Macro Libraries by Statements or Macros

The SELECT control statement is used to control the Macro Update Program printed output and remark inclusion. The output is always printed on the list (principal) printer, and MUP control statements are always printed. You can also specify in a SELECT statement that remarks are to be included with any macro text statement being placed on disk. An example of the print format may be seen in the sample program that is at the end of this chapter.

The format of the SELECT statement is as follows:



no parameters--suppresses printing of macro headers and macro text.

M--causes headers to be printed.

P--causes text and headers to be printed. If you select this option, it should have been preceded by a NAME statement (discussed previously). On an OUTPUT operation with this option, the 1800 will not list the macro text. Note that if a LIB, BUILD, or ENDUP statement is encountered and no name statement is available, SELECT P will be reset and the text will not be printed. On the 1800, if sense switch 2 is ON, printing will be suppressed.

I--causes headers to be printed except during an ADD, INSERT, or DELETE operation, in which case, the macro headers and text are printed. If you do not specify a SELECT statement, I is assumed.

On the 1800, if sense switch 2 is ON, printing will be suppressed.

C--indicates that remarks are to be included with any macro text statement being placed on disk.

SELECT N

N--indicates that both of the following conditions are true (see also the section following):

- You want to update a nested definition.
- The statement(s) inserted uses the automatic name generation feature.

You may use any combination of parameters in a SELECT statement; however, I will override M and P will override either I or M, or both. Each two consecutive parameters must be separated by a comma, with no embedded blanks. If you use more than one SELECT statement, the latest is assumed, and the prior SELECT statements are overridden.

SPECIAL REQUIREMENTS ON THE USE OF AUTOMATIC NAME GENERATION IN NESTED DEFINITIONS

When automatic name generation is used with nested definitions, the indicators for automatic name generation (leading and/or trailing apostrophes) must be suppressed until the call (expansion) of the outer macro occurs. Otherwise, the automatic name generation feature will not function properly. To suppress this feature, the statements in a nested definition must be stored as data. In a nested definition, the MAC statement(s) of the inner nested definitions indicates that the Macro Update Program (on an ADD statement) or the Macro Assembler (on an SMAC statement) should store the statements between the MAC and its associated MEND.

When inserting statements that use automatic name generation in a nested definition, you must precede the INSERT or DELETE statements with a SELECT statement that includes N as one of its parameters. This should be done to indicate the Macro Update Program should store these inserted statements as data. The SELECT N option should be followed by another SELECT option (without N as a parameter) when all inserts to the nested definition are completed. If suppressing the automatic name generation feature by SELECT N is not done properly, the consequences will not be observed until an assembly with the call (expansion) of the nested macro is attempted.

Consider the following example:

```
LIB 'LIB01'
SELECT P
NAME LABEL,A,B,C

                    14BO 0005 MAC09
                    00001  LABEL LD    L  A
                    00002        STO   L  B
                    00003        MAC
                    00004        MOVE     CNT,FROM,TO
                    00005        LDX      -CNT
                    00006  A'    LD    L1 FROM+CNT
                    00007        STO   L1 TO+CNT
                    00008        MDX    1 1
                    00009        MDX      A'
                    00010        MEND
                    00011        BSC   L  C
                    00012  **MACRO END**
                    14BO 0049  **LIBRARY END**
                    0000
ENDUP
                              UPDATE COMPLETED
```

The macro MOVE is a nested definition and uses the automatic name generation feature. Hence, a SELECT N statement must precede any updates (statements for insertion into the macro MOVE) that use the automatic name generation feature. Failure to do so will result in the symbol actually being expanded at this time, which is not desired.

Note that any updates using automatic name generation outside the macro MOVE should not be preceded by a SELECT N statement, since the automatic name generation will be suppressed. In the above example, an insert that uses automatic name generation made between statement number 00003 and statement number 00010 should be preceded by a SELECT N statement. An insert made elsewhere in the macro MOVE that does not use automatic name generation should not be preceded by a SELECT N.

Designating Comments

use a period

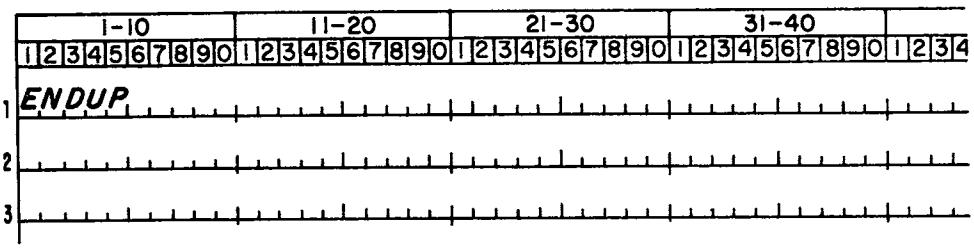
A period in column one of any record designates that record as a MUP comment and it is printed on the list (principal) printer. If, however, an error has occurred, and MUP is ignoring all statements until the next LIB or BUILD statement, comments will also be ignored and not printed. Note also that MUP comment records are never included in your macro library.

Terminating a Macro Update Run

use ENDUP

To terminate a Macro Update Program run properly, you must use an ENDUP statement. This statement must be the last MUP control statement used, or else none of the statements that follow it will be processed. You don't specify any parameters in an ENDUP statement; the format is as follows:

format



The UPDATE COMPLETED message (see the following sample program, note 9) indicates that the Macro Update run has been properly terminated. This does not imply that all the operations requested were successful.

Sequencing MUP Control Statements

The following table illustrates the required order of MUP control statements. Each x in the matrix indicates that the function at the top of the column must precede the statement to the left side of the x. MUP statements not included in this table have no precedence requirements.

	BUILD or LIB	RENAME or UPDATE	NAME
LIBRARY FUNCTIONS			
JOIN	X		
CONCAT	X		
NAME STATEMENT	X		
MACRO FUNCTIONS			
UPDATE	X		
ADD	X		X
PURGE	X		
RENAME	X		
OUTPUT	X		X
STATEMENT FUNCTIONS			
INSERT		X	X
DELETE		X	X
PRINT FUNCTIONS			
ANY SELECT P			X

Note that a new LIB or BUILD function removes the previous NAME statement and a new one must be used. On any function where you use a SELECT P option, you must have a NAME statement available to SELECT P or else the print or punch operation will not be as you wanted it.

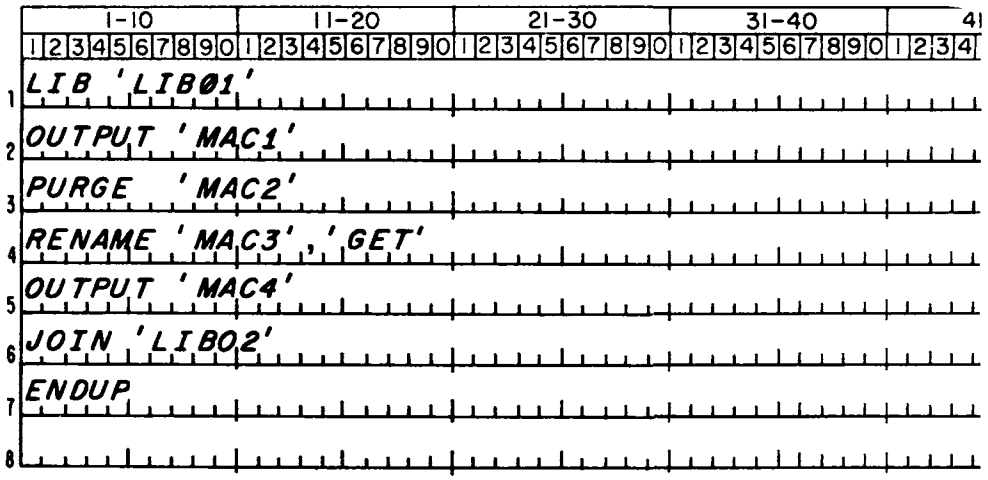
Making Efficient Use of the Macro Update Program

The Macro Update Program searches sequentially through the library it is operating on for the macro specified in a RENAME, OUTPUT, UPDATE, or PURGE statement. It joins

libraries and adds macros to the end of each library. Consequently, the printing may be voluminous while the Macro Update Program is positioning for the requested function. Therefore, to make the most efficient use of the program, you should request the macro functions in the order in which the macros appear in the library, and the JOIN and ADD functions after all other functions for that library have been performed.

request functions in order

For example, suppose LIB01 contains four macros: MAC1, MAC2, MAC3, and MAC4, respectively and you want to perform an OUTPUT function on MAC1 and MAC4, a PURGE function on MAC2, a RENAME function on MAC3, and a JOIN function on LIB02. The following statement sequence is the most efficient to perform these functions.



The concatenate function does not cause library positioning, and thus, its position within a MUP run is not important.

A Sample Macro Update Program

The following sample program illustrates a sample Macro Update Program run.

```
// JOB         X   00 JAN 00 00.012 HRS
// *      SAMPLE USAGE OF MACRO UPDATE PROGRAM (MUP)
// OMP    00 JAN 00 00.012 HRS
*MACRO UPOATE
*
* BUILD A MACRO LIBRARY CALLED 'LIB01'
(1)BUILD 'LIB01'
*
* ADD A MACRO TO PLACE THE SPECIFIED INDEX REGISTER'S CONTENTS IN THE ACC
* CALLING FORMAT IS
*                 LABEL NXACC   X
* WHERE 'LABEL' IS ANY VALID LABEL AND 'X' IS THE INDEX REGISTER WHOSE
* CONTENTS ARE TO BE PLACED IN THE ACCUMULATOR
(2)NAME LABEL,X
 ADD 'NXACC'
                  (4)1480 0005 NXACC
                (5)X 00001  LABEL STX   X *+2
              (6)  X 00002         LD      *+1
                   X 00003         MDX     *+1
                   X 00004         DC      *-*
*
* ADD A MACRO TO SPACE LISTING 1 LINE WHEN BLANK OP CODE APPEARS
                  (7)00005  ** MACRO END **
 ADD '     '
                  (4)1480 001F
                (6)X 00001         SPAC    1
              (8)
*
* BUILD A MACRO LIBRARY CALLED 'LIB02'
                  (7)00002  ** MACRO END **
                (3)1480 0029 ** LIBRARY END **
                  (8)0000
(1)BUILD 'LIB02'
*
* ADD A MACRO TO SPACE LISTING 1 LINE WHEN BLANK OP CODE APPEARS
(2)NAME
 ADD '     '
                  (4)14B1 0005
                (5)X 00001         SPAC    1
              (6)
*
* ADD A MACRO TO FILL CORE STORAGE WITHIN THE LIMITS SPECIFIED WITH THE
* WORD AT THE LOCATION SUPPLIED USING INDEX REGISTER 1
* CALLING FORMAT IS
*                 LABEL FILL    FROM,TO,WORD
* WHERE 'FROM' IS THE STARTING ADDRESS, 'TO' IS THE ENDING ADDRESS, AN
* 'WORD' IS THE ADDRESS OF THE FILL WORD. 'FROM' MUST BE LESS THAN OR
* EQUAL TO 'TO'.
                  (7)00002  ** MACRO END **
(2)NAME LABEL,FROM,TO,WORD
 ADD 'FILL'
                  (4)14B1 000F FILL
                (5)X 00001  LABEL STX   1 XR1'
              (6)  X 00002         LD      AD2'
                   X 00003         S       AD1'
                   X 00004         STO     *+1
                   X 00005         LDX  L1 *-*
                   X 00006         MDX   1 1
                   X 00007         LD   L  WORD
                   X 00008  LP'    STO  L1 FROM-1
                   X 00009         MDX   1 -1
                   X 00010         MDX     LP'
                   X 00011         LDX  L1 *-*
                   X 00012  XR1    EQU     *-1
                   X 00013         MDX     *+2
                   X 00014  AD1'   DC      FROM
                   X 00015  AD2    DC      TO

*
* BUILD A MACRO LIBRARY CALLED 'LIB03'
                  (7)00016  ** MACRO END **
                (3)14B1 0066 ** LIBRARY END **
                  (8)0000
(1)BUILD 'LIB03'
*
* ADD A GENERAL CORE MOVE MACRO USING INDEX REGISTER 1
* CALLING FORMAT IS
*                 LABEL MOVE    FROM,TO,COUNT
* WHERE 'FROM' IS THE SENDING FIELD STARTING ADDRESS, 'TO' IS THE RECEIVING
* FIELD STARTING ADDRESS, AND 'COUNT' IS THE NUMBER OF WORDS TO MOVE.
(2)NAME LABEL,FROM,TO,COUNT
 ADD 'MOVE'
                  (4)14B7 0005 MOVE
                (5)X 00001  LABEL LDX  L1 COUNT
              (6)  X 00002         LD   L1 FROM-1
                   X 00003         STO  L1 TO-1
                   X 00004         MDX   1 -1
                   X 00005         MDX     *-6

*
* ADD A MACRO TO PLACE THE ACC CONTENTS IN THE INDEX REGISTER SPECIFIED
* CALLING FORMAT IS
*                 LABEL ACCNX   X
* WHERE 'X' IS THE INDEX REGISTER TO BE LOADED WITH THE ACC CONTENTS.
                  (7)00006  ** MACRO END **
(2)NAME LABEL,X
 ADD 'ACCNX'
                  (4)14B7 0024 ACCNX
                (5)X 00001  LABEL STO     *+1
              (6)  X 00002         LDX  LX *-*
```

(1) The library named in a BUILD statement must have been defined previously with an *DFILE control statement.

(2) The NAME statement is required and may specify parameter names to be used in a subsequent definition.

(3) **LIBRARY END** is printed each time the Macro Update Program encounters the end of the library. The two numbers indicate (1) the logical drive number and sector address of the last sector of the library currently used, and (2) the relative address of the last word on that sector which is used.

By inspecting these two words, you can determine how much of your library file has been filled.

(4) The name of the macro being operated upon is printed. The two numbers preceding the macro name indicate where on the disk the subject macro was built. Word one indicates the logical drive number and sector address of the definition and word two indicates the relative location of where that definition begins on that sector.

(5) An X is printed with each statement that is added to a library or macro.

(6) Sequence numbers are printed whenever a macro text is printed.

(7) **MACRO END** is printed whenever the Macro Update Program is through processing a definition.

(8) If the library being worked upon is concatenated to another library, this number indicates the logical drive number and sector address of the concatenated library. If no concatenation exists, this number is zero.

(9) The Macro Update Program prints the macro names it encounters when scanning through a library. The two numbers in front of the macro name indicate information as described in (4)

(10) The library named in a LIB, JOIN, or CONCAT statement must have been defined previously with an *DFILE control statement and initialized with a BUILD statement.

(11) This message is printed when the Macro Update run has been completed. It does not imply successful completion; messages prior to this one may indicate error conditions.

```
. ADD A MACRO TO MOVE THE CONTENTS OF A SPECIFIED INDEX REGISTER TO
. ANOTHER SPECIFIED INDEX REGISTER
. CALLING FORMAT IS
.                LABEL LOADX   FROMX,TOX
. WHERE 'FROMX' IS THE INDEX REGISTER WHOSE CONTENTS ARE MOVED TO THE
. INDEX REGISTER 'TOX'.
                (7)00003  ** MACRO END **
(2)NAME LABEL,X,Y
ADD 'LOADX'
                (4)14B7 0034 LOADX
              (5)X 00001  LABEL STX   X *+1
             (6)  X 00002        LDX  LY *-*
. PHYSICALLY JOIN LIBRARY 'LIB03' TO LIBRARY 'LIB01'
                (7)00003  ** MACRO END **
              (3)14B7 0044 ** LIBRARY END **
                (8)0000
(10)LIB 'LIB01'
                (9)14B0 0005 NXACC
                (9)14B0 001F
(10)JOIN 'LIB03'
                (9)14B0 0029 MOVE
              (9)14B0 0048 ACCNX
                (9)14B0 0058 LOADX
.
. UPDATE THE 'MOVE' MACRO TO SAVE AND RESTORE INDEX REGISTER 1
UPDATE 'MOVE'
                (3)14B0 0068 ** LIBRARY END **
              (8)0000
                (9)14B0 0005 NXACC
              (9)14B0 001F
                (9)14B0 0029 MOVE
(2)NAME LABEL,FROM,TO,COUNT
DELETE 1
              (5)X 00001  LABEL STX   1 *+9
              (5)X 00002        LDX  L1 COUNT
INSERT 5
             (6)
                   00003        LD   L1 FROM-1
                   00004        STO  L1 TO-1
                   00005        MDX   1 -1
                   00006        MDX     *-6
              (5)X 00007        LDX  L1 *-*
.
. RENAME THE 'MOVE' MACRO AS 'MOVE1'
                (7)00008  ** MACRO END **
RENAME 'MOVE','MOVE1'
                (9)14B0 0053 ACCNX
              (9)14B0 0063 LOADX
              (3)14B0 0073 ** LIBRARY END **
              (8)0000
                (9)14B0 0005 NXACC
              (9)14B0 001F
                (4)14B0 0029 MOVE
                          ABOVE MACRO RENAMED AS
                   14B0 0029 MOVE1
. ONLY THE NAME 'MOVE1' CAN NOW BE USED TO REFERENCE THE MACRO
. FORMERLY NAMED 'MOVE'.
.
. PUNCH THE MACRO NOW CALLED 'MOVE1'
OUTPUT 'MOVE1'
                (9)14B0 0053 ACCNX
              (9)14B0 0063 LOADX
              (3)14B0 0073 ** LIBRARY END **
              (8)0000
                (9)14B0 0005 NXACC
              (9)14B0 001F
                (9)14B0 0029 MOVE1
              (9)14B0 0053 ACCNX
                (9)14B0 0063 LOADX
                (3)14B0 0073 ** LIBRARY END **
                (8)0000
ENDUP
                        (11)UPDATE COMPLETED
```

(1) The library named in a BUILD statement must have been defined previously with an *DFILE control statement.

(2) The NAME statement is required and may specify parameter names to be used in a subsequent definition.

(3) **LIBRARY END** is printed each time the Macro Update Program encounters the end of the library. The two numbers indicate (1) the logical drive number and sector address of the last sector of the library currently used, and (2) the relative address of the last word on that sector which is used.

By inspecting these two words, you can determine how much of your library file has been filled.

(4) The name of the macro being operated upon is printed. The two numbers preceding the macro name indicate where on the disk the subject macro was built. Word one indicates the logical drive number and sector address of the definition and word two indicates the relative location of where that definition begins on that sector.

(5) An X is printed with each statement that is added to a library or macro.

(6) Sequence numbers are printed whenever a macro text is printed.

(7) **MACRO END** is printed whenever the Macro Update Program is through processing a definition.

(8) If the library being worked upon is concatenated to another library, this number indicates the logical drive number and sector address of the concatenated library. If no concatenation exists, this number is zero.

(9) The Macro Update Program prints the macro names it encounters when scanning through a library. The two numbers in front of the macro name indicate information as described in (4)

(10) The library named in a LIB, JOIN, or CONCAT statement must have been defined previously with an *DFILE control statement and initialized with a BUILD statement.

(11) This message is printed when the Macro Update run has been completed. It does not imply successful completion; messages prior to this one may indicate error conditions.

During the assembly process, the Macro Assembler checks for source program errors. If an error is detected, an error flag or an error code and message will be printed. If a questionable instruction is encountered, it is flagged with a warning flag, Q. Errors in the Macro Update Program are detected by the DM2 Disk Utility Program or the MPX Disk Management Program.

Macro Assembler Sign-Off Message

At the end of each assembly, the Macro Assembler indicates the number of errors and warnings it encountered during that assembly. The message reads:

XXX ERROR(S) AND XXX WARNING(S) IN ABOVE ASSEMBLY.

XXX represents a three-digit decimal number.

Macro Assembler Warning Flag

If the source program contains certain questionable instructions, the Macro Assembler will interpret and process them and flag them with a Q. For example, the statement MDX L PLACE is assembled as MDX L PLACE,0 and is flagged with the warning indicator. Warning flags are not counted as errors and will not prevent the execution of the object program. It is suggested that you check to make sure that the Macro Assembler has performed the task that you really wanted it to do on each statement flagged with a Q.

Macro Assembler Error Detection Codes

During the assembly process, the Macro Assembler checks the source program for errors. Error and warning flags are printed to the left of the label field of each source statement that is in error or is questionable. For a complete description of the listing format of an assembly, see the 1130 DM2 Programming and Operator's Guide, Order Number GC26-3717, or the 1800 Error Messages manual, Order Number GC26-3727.

See Table 1 for an explanation of the error flags and Table 2 for a listing of the Macro Assembler error messages and their meanings.

Macro Update Program Error Messages

When the DM2 Disk Utility Program or the MPX Disk Management Program encounters an error in the Macro Update Program, one of the error messages in Table 3 is printed.

Table 1. Error Flags

Flag	Cause and Macro Assembler Action
A	• Address Error. Attempt made to specify, directly or indirectly, a displacement field outside range of –128 to +127. Displacement is set to zero. • Invalid reference to CE Core in a long form instruction. Address field is set to zero. (1800 only)
C	Condition Code Error. Character other than +, –, Z, E, C, or O detected in first operand of short branch or second operand of long BSC, BOSC, or BSI statement. Displacement is set to zero.
F	Format Code Error. Character other than L, I, X, or blank detected in column 32, or L or I format specified for an instruction valid only in short form, or I format specified when not allowed. Instruction processed as if L format were specified, unless that instruction is valid only in short form, in which case it is processed as if the X format were specified.
L	Label Error. Invalid symbol detected in label field. Label is ignored.
M	Multiply Defined Label Error. Same symbol encountered in label fields of two or more statements. First occurrence of symbol in label field defines its value; subsequent occurrences of symbol in label field are ignored and cause an M error flag to be printed.
O	Op Code Error. • Invalid op code. Statement is ignored and address counter is incremented by 2. • ABS used when *COMMON is used to define FORTRAN common table. Statement is ignored. (1800 only) • ISS, ILS, ENT, LIBR, SPR, EPR, or ABS incorrectly placed. Statement is ignored.
Q	Warning. A statement whose syntax is questionable was encountered.
R	Relocation Error. • Expression does not have a valid relocation. Expression is set to zero. • Non-absolute displacement specified. Displacement is set to zero. • Absolute origin specified in relocatable program. Origin is ignored. • Non-relocatable operand in END statement of relocatable mainline program. Columns 9–12 are left blank; entry is assumed to be relative zero. • Non-absolute operand specified in BSS or BES. Operand is assumed to be zero. • ENT operand is non-relocatable. Statement is ignored. • Invalid reference to CE Core. Address field is set to zero. (1800 only) • Invalid reference to a symbol defined in a COMMON area. Address field is set to zero. (1800 only)
S	Syntax Error. • Invalid expression, for example, invalid symbol, adjacent operators, or illegal constant. Expression is set to zero. • Illegal character in record. If illegal character appears in expression, label, op code, format, or tag field, additional errors may be caused. • Main program entry point not specified in END operand. Columns 9–12 are left blank; entry is assumed to be relative zero. • Incorrect syntax in EBC statement (such as no delimiter in column 35, or zero character count). Columns 9–12 are not punched; address counter is incremented by 17. • Invalid label in ENT or ISS operand. Statement is ignored.
T	Tag Error. Column 33 contains character other than blank, 0, 1, 2, or 3 in instruction statement. Tag of zero is assumed.
U	Undefined symbol. Undefined symbol in expression. Expression is set to absolute zero.
1130 ONLY	
W	X or Y coordinate, or both, not within the specified range, or invalid operand. Operand set to zero.
X	Character other than R or I in column 32, or character other than D or N in column 33. Field set to zero.
Z	Invalid condition in a conditional branch or interrupt order. Condition bits in first word set to zero.

Table 2. Macro Assembler Error Codes and Messages

Error Code and Message		Cause
A01 (1130)	MINIMUM W.S. NOT AVAILABLE ASSEMBLY TERMINATED	Working Storage (for DM2) or Batch Processing Working Storage (for MPX) available is less than the number of overflow sectors specified plus one.
A01 (1800)	MINIMUM W.S. NOT AVAILABLE... ASM TERM.	
A02 (1130)	SYMBOL TABLE OVERFLOW ASSEMBLY TERMINATED	Number of sectors of symbol table overflow is greater than the number of overflow sectors specified.
A02 (1800)	SYMBOL TABLE OVERFLOW... ASM TERM.	
A03	DISK OUTPUT EXCEEDS W.S.	Intermediate output in pass 1 or final disk system format output in pass 2 is larger than Working Storage (for DM2) or Batch Processing Working Storage (for MPX) minus the number of overflow sectors specified.
A04	SAVE SYMBOL TABLE INHIBITED	With *SAVE SYMBOL TABLE option specified: • Program in relocatable assembly. • Program contains an assembly error. • Source program causes more than 100 symbols to be present in the System Symbol Table.
A05 (1130)	XXX ERRONEOUS ORG, BSS, OR EQU STATEMENTS IN ABOVE ASSEMBLY	XXX is the number of ORG, BSS, BES, and/or EQU statements that were undefined in pass 1. At the end of pass 1, these erroneous statements are printed on the list printer. If the error was due to forward referencing, it will not be detected during pass 2.
A05 (1800)	ERRONEOUS STMNT ON PASS ONE.	Errors were detected during pass one. This warning message is printed at the end of pass one.
A06	LOAD BLANK CARDS	A card containing a nonblank column in columns 1-71 has been read while punching the symbol table (as a result of an *PUNCH SYMBOL TABLE control statement).
A07 (1130)	ABOVE CONTROL STATEMENT INVALID	An invalid control statement has been read by the Macro Assembler. The control statement is ignored and the assembly is continued.
A07 (1800)	INVALID CTRL STATEMENT	
A08	MACLIB UNDEFINED	The Macro Assembler has been asked to process a SMAC statement and either no *MACLIB control statement was previously read, or the name on the *MACLIB statement is not found in a disk area search. The library named in the *MACLIB statement has a library concatenated to it and this library could not be found.
A09 (1130)	PARAMETER LIST OVERFLOW ASSEMBLY TERMINATED	A call to a macro exceeds the space specified in the N2 field of the *OVERFLOW SECTORS control statement used with this assembly. If the *OVERFLOW SECTORS control statement was not used or if the N2 field was not specified, the Macro Assembler assumes the value of the N2 field to be zero.
A09 (1800)	PARAM LIST OVFLO	
A10 (1130)	MACRO AREA OVERFLOW ASSEMBLY TERMINATED	An attempt was made to define a temporary macro and either the N3 field of the *OVERFLOW SECTORS control statement was not specified, or the the space specified by N3 was not large enough, or the macro library was exceeded.
A10 (1800)	MACRO LIBR OVFLO	
A12	NEST LEVEL EXCEEDS 20 ASSEMBLY TERMINATED	A macro call exceeded the allowable nest level limit of 20.
A21	*LEVEL CONTROL STATEMENT MISSING	The program listed above was assembled as an ISS subroutine without the required *LEVEL control card. (1130 only)
A22	INVALID LIST DECK OPTION ASSEMBLY TERMINATED	An attempt was made to punch macro statements in two pass mode. (1130 only)
A40	MAIN PROG NO NAME	Mainline program being assembled has no name specified on a // ASM statement. (1800 only)
A41	// CARD READ	A Supervisor control statement has been read by the Macro Assembler. The Macro Assembler has passed this statement along to the Supervisor before terminating the assembly. Loading and DMP operations are inhibited. (1800 only)
A42	ABSOLUTE REENTRANT PROG	A non-relocatable program has been specified as reentrant. (1800 only)
A43	INVALID SFRLE FILE	When loading the source deck via *SFRLE, the file was truncated due to insufficient sector allocation. (1800 only)
A44	LIT TBL OVFLO	The size of the literal table as specified on the fourth parameter of the *OVERFLOW SECTORS control statement was too small. (1800 only)
A46	XREF DATA OVFLO	VCORE was too small to sort and merge the data for building the cross-reference table. (1800 only)

Table 3. Macro Update Program Error Messages

Error Code and Message		Cause and Corrective Action
D100	LIBRARY NOT FOUND	Librery named on a LIB, BUILD, JOIN, or CONCAT statement could not be found on drives currently in use. If LIB, BUILD, or JOIN statement, all statements are ignored until the next LIB, BUILD, or ENDUP statement is encountered. If CONCAT statement, processing continues with the next control statement. Correct the name field in the statement in error, or change the // JOB statement to include the drive on which the named library resides, or define the macro library using an *DFILE or *STOREDATA control statement.
D101	INVALID SUBFIELD COL XX	If on an INSERT or DELETE statement, the sequence number was incorrectly specified, that is, it was negative, or non-numeric, or sequence numbers were reversed. If on a SELECT statement, an incorrect parameter was specified. Processing continues with the next control statement. If on a NAME statement, an invelid parameter was detected. Processing continues with next LIB, BUILD, or ENDUP statement. XX indicates the column in which the error was found. Correct the error and rerun the portion of the job affected.
D102	ILLEGAL REQUEST	An invalid control statement was detected, an INSERT or DELETE statement was not preceded by an UPDATE or RENAME statement, or on an 1130, request was made for output to paper tape or to a pack configured for paper tape. Processing continues with the next control statement. Correct the error and rerun the portion of the job affected.
D103	LIBRARY OVERFLOW	The library last specified by a LIB or BUILD statement does not have enough room to perform the operation. If on a JOIN or an ADD statement, the operation is suppressed and the library is restored to its previous state. If on en INSERT statement, the stete-ments listed prior to the message were the only ones that could be included. Processing continues with the next LIB, BUILD, or ENDUP statement. Additional space can be obtained in the current library by purging unneeded macros or deleting unneeded statements. If this is not possible, define a larger library using an *DFILE or *STOREDATA control statement, join the old library to a new one, and delete the old library. Once the eddltional space has been obtained, rerun the portion of the job affected. If on an INSERT, it may be necessary to alter your INSERT control statement as the statements in the macro library may have been resequenced.
D104	MACRO NOT FOUND	The macro name specified on an OUTPUT, PURGE, RENAME, or UPDATE statement could not be found in the library being processed. Processing continues with the next control statement. Correct the macro neme on the statement in error, or specify the correct macro library and rerun the portion of the job affected.
D105	SEQUENCE NUMBER NOT FOUND	The sequence number on an INSERT or DELETE statement was out of the range of the macro and could not be found or the sequence numbers on multiple INSERT and/or DELETE statements for the same macro were out of order. Processing continues with the next control statement. Place a correct sequence number on the stetement in error and rerun the portion of the job affected.
D106	LIBRARY NOT SPECIFIED	An attempt was made to operete on a macro without specifying a macro library. Processing continues with the next LIB, BUILD, or ENDUP statement. Place a LIB or BUILD statement before the statement in error and rerun the portion of the job affected.
D107	SPILL OVERFLOW	Macro text insertions have caused the capecity of Working Storage spill to be exceeded. Processing continues with the next LIB, BUILD, or ENDUP statement. Correct the sequence numbers in the unprocessed INSERT statements, if necessary, and rerun these statements. It may be necessary to define additional disk drives to provide adequete Working Storage.
D108	CONTROL STATEMENT READ	An * or // statement has been read and the MUP run is termineted. On the 1800, control is passed to the Supervisor which will begin processing with the next // JOB statement. On the 1130 control is returned to the Supervisor for a // statement or to DUP for an * statement.
D109	NAME STATEMENT NOT FOUND	The operation attempted requires a NAME statement and one has not been processed following the last LIB or BUILD statement. Processing continues with the next LIB, BUILD, or ENDUP statement. Insert a NAME statement and rerun the part of the job that was affected.

Table 3. Macro Update Program Error Messages (Continued)

Error Code and Message		Cause and Corrective Action
D110	INVALID NAME	The name field on a LIB, BUILD, JOIN, CONCAT, UPDATE, ADD, PURGE, RENAME, or OUTPUT statement was left blank, the name specified was invalid, or the apostrophes are improperly placed. If on a LIB, BUILD, or JOIN statement, processing continues with the next LIB, BUILD, or ENDUP statement. If on a CONCAT, UPDATE, ADD, PURGE, RENAME, or OUTPUT statement, processing continues with the next control statement.
D112	NONBLANK CARD READ. ENTER BLANK CARDS	A nonblank card has been read by a 1442-6 or -7 during a punch operation. Remove stacked input from hopper, NPRO nonblank cards, place blank cards followed by NPRO cards end stacked input in the hopper, and press reader START and for the 1130 press program START.
D116	LIBRARY NOT INITIALIZED	Library named on a LIB, JOIN, or CONCAT statement has not been initialized previously. If on a LIB, or JOIN statement, processing continues with the next LIB, BUILD, or ENDUP statement. If on a CONCAT statement, processing continues with the next control statement. Initialize the library with a BUILD statement, and rerun the portion of the job affected. If on a BUILD statement, the library specified was not a data file. Correct the BUILD statement and rerun the portion of the job affected.
D117	INVALID PARAMETER	On a NAME statement, more than 20 parameters were specified. If during the processing of a macro, a paremeter has been detected which was not defined in the NAME statement or a parameter greater than one character was used in the format or tag field. If during an OUTPUT operation, the operation is aborted and processing continues with the next control statement. If during a listing operation, this is a warning message and the invalid parameter is printed as //N where N is 1-20. (Note: N may be truncated if the field size is exceeded.) If on OUTPUT, correct the NAME statement and rerun the portion of the job affected.

This page intentionally left blank.

Appendix A: General Examples of Macros and 1130 DM2 Macro Assembler Features

The following group of macros and the examples of their use are intended to demonstrate how macro instructions can be used to simplify assembler-language programming. If all these macros were defined in your system, then you could use two new statements--the READ statement and the WRITE statement--to accomplish all the programming normally required to effect input and output on an 1130 system having a disk, a 1442 card read punch, and a 1403 printer. When you issue a READ or WRITE macro, you need specify only the name (DISK, CARD, or PRINT) of the device you want to use, the name of the I/O area, and, if you want, the name of your error-handling program. This system of macros then issues calls to the appropriate I/O control subroutines, handles data conversion and blocking, and, at your option, handles error checking and retries.

You could, of course, expand this set of macros to include all I/O devices supported by the 1130 system; you could also write a similar set of macros to simplify I/O programming on the 1800 system.

Refer to the 1130 Subroutine Library manual, Order Number GC26-5929, for a complete description of the 1130 I/O control subroutines (DISKN, CARD1, and PRNT3) referred to in these macros.

The first part of this sample program (until the *MACRO UPDATE statement) is a FORTRAN program that builds a one sector file of one-word integers having the value 1 through 320. This is necessary to handle the data file in the sample program that follows it.

```
// JOB                                                                  SMAC0010

LOG DRIVE   CART SPEC   CART AVAIL  PHY DRIVE
  0000        0578        0578        0000

V2 M06   ACTUAL 32K  CONFIG 32K

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * SMAC0020

// *  CREATE A DISK DATA FILE NAMED 'FILE1' AND FILL IT WITH          *  SMAC0030

// *  320 INTEGER VALUES 1 TO 320 VIA FORTRAN.                        *  SMAC0040

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * SMAC0050

// DUP                                                                  SMAC0060

*DELETE             FILE1                                             SMAC0070
D 26 NAME NOT FOUND IN LET/FLET


*STOREDATA  WS  UA  FILE1    1                                        SMAC0080
CART ID 0578   DB ADDR  3220   DB CNT   0010

// FOR                                                                  SMAC0090
*IOCS(DISK)                                                           SMAC0100
*LIST ALL                                                             SMAC0110
```

```
*ONE WORD INTEGERS
      DEFINE FILE 1(320,1,U,K)                                          SMAC0120
      K = 1                                                             SMAC0130
      IVAL = 1                                                          SMAC0140
      DO 100 I = 1,320                                                  SMAC0150
      WRITE (1'K) IVAL                                                  SMAC0160
      IVAL = IVAL + 1                                                   SMAC0170
  100 CONTINUE                                                          SMAC0180
      CALL EXIT                                                         SMAC0190
      END                                                               SMAC0200
VARIABLE ALLOCATIONS                                                    SMAC0210
     K(I )=0008       IVAL(I )=0009          I(I )=000A

STATEMENT ALLOCATIONS
 100  =0028

FEATURES SUPPORTED
 ONE WORD INTEGERS
 IOCS

CALLED SUBPROGRAMS
 SDFIO   SDWRT   SDCOM   SDI

INTEGER CONSTANTS
     1=000C    320=000D

CORE REQUIREMENTS FOR
 COMMON      0  VARIABLES     12  PROGRAM     38

 END OF COMPILATION

// XEQ       L  1                                                       SMAC0220

*FILES(1,FILE1)                                                         SMAC0230
FILES ALLOCATION
    1 0322  0001  0578 FILE1
STORAGE ALLOCATION
R 41  7AFC (HEX) WDS UNUSED BY CORE LOAD
LIBF TRANSFER VECTOR
 PAUSE  04D8
 SDCOM  02A5
 SDI    025E
 SDWRT  02DA
 SDFIO  02DF
SYSTEM SUBROUTINES
 ILS04  00C4
 ILS02  00B3
      020C (HEX) IS THE EXECUTION ADDR


// JOB                                                                  SMAC0260

LOG DRIVE   CART SPEC   CART AVAIL  PHY DRIVE
  0000        1111        1111        0002

V2 M07   ACTUAL 32K  CONFIG 32K

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * SMAC0270

// *  DEFINE A DISK FILE AND INITIALIZE IT FOR A MACRO LIBRARY.      *  SMAC0280

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * SMAC0290

// DUP                                                                  SMAC0300

*DELETE             PURGE                                               SMAC0310
CART ID 1111   DB ADDR  2620   DB CNT   0050

*DFILE          UA  PURGE 0005                                          SMAC0320
CART ID 1111   DB ADDR  2620   DB CNT   0050

*MACRO UPDATE                                                           SMAC0330
```

```
BUILD 'PURGE'                                                          SMAC0340
                    0262 0005 ** LIBRARY END **
                    0000
ENDUP                                                                  SMAC0350
                    UPDATE COMPLETED

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * SMAC0360

// *  DEFINE 11 MACROS FOR THE MACRO LIBRARY NAMED 'PURGE'           *  SMAC0370

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * SMAC0380

// ASM                                                                 SMAC0390
*MACLIB PURGE                                                          SMAC0400
*LIST                                                                  SMAC0410
                    *  * * * * * * * * * * * * * * * * * * * * * * *  SMAC0420
                    *  ILLRQ MACRO                                    *  SMAC0430
                    * PRINTS ILLEGAL REQUEST MESSAGE WHEN CALLED      *  SMAC0440
                         SMAC                                          SMAC0450
                         ILLRQ              LLEGAL REQUEST GENERATOR   SMAC0460
          00001          LIST    ON         FORCE LISTING              SMAC0470
          00002     *    ILLEGAL REQUEST    PRINT MESSAGE              SMAC0480
          00003          LIST               RESTORE LIST CONDITION     SMAC0490
                         MEND                                          SMAC0500



                    *  * * * * * * * * * * * * * * * * * * * * * * *  SMAC0520
                    *  AUTO ERROR                                     *  SMAC0530
                    * GENERATES A DEFAULT ERROR ROUTINE WHEN ERROR    *  SMAC0540
                    * PARAMETER WAS 0 OR NOT PASSED.                  *  SMAC0550
                    *  * * * * * * * * * * * * * * * * * * * * * * *  SMAC0560
                         SMAC                                          SMAC0570
                         AUTOE              AUTOMATIC ERROR GENERATOR  SMAC0580
          00001          DC      *+1        CALL SEQ-ERROR ENTRY ADDR  SMAC0590
          00002          MDX     *+3        SKIP AROUND ERROR SUBR     SMAC0600
          00003          DC      *-*        ENTER HERE ON ERROR        SMAC0610
          00004          BSC  I  *-3        RETURN TO RETRY OPERATION  SMAC0620
                         MEND                                          SMAC0630



                    *  * * * * * * * * * * * * * * * * * * * * * * *  SMAC0650
                    *  READ MACRO                                     *  SMAC0660
                    * SETS FUNCTION CODE AND CALLS SPECIFIC DEVICE    *  SMAC0670
                    * MACRO.                                          *  SMAC0680
                    *     LABEL = LABEL                              *  SMAC0690
                    *     DEVC  = DISK,CARD, OR PRINT                *  SMAC0700
                    *     AREA  = I/O AREA ADDRESS OR LABEL          *  SMAC0710
                    *     ERROR = ADDRESS OF USERS ERROR ROUTINE     *  SMAC0720
                    *             THIS PARAMETER IS OPTIONAL         *  SMAC0730
                    *             IF 0 OR BLANK,THE MACRO GENERATES  *  SMAC0740
                    *             AN ERROR ROUTINE                   *  SMAC0750
                    *  * * * * * * * * * * * * * * * * * * * * * * *  SMAC0760



                         SMAC               GENERAL READ MACRO         SMAC0770
                    LABEL READ    DEVC,AREA,ERROR                      SMAC0780
          00001     LABEL DEVC    1,AREA,ERROR                         SMAC0790
                         MEND                                          SMAC0800



                    *  * * * * * * * * * * * * * * * * * * * * * * *  SMAC0820
                    *  WRITE MACRO                                    *  SMAC0830
                    *     LABEL = LABEL                                  SMAC0840
                    *     DEVC  = DISK,CARD, OR PRINT                *  SMAC0850
                    *     AREA  = I/O AREA ADDRESS OR LABEL          *  SMAC0860
                    *     ERROR = ADDRESS OF USERS ERROR ROUTINE     *  SMAC0870
                    *             THIS PARAMETER IS OPTIONAL         *  SMAC0880
                    *             IF 0 OR BLANK,THE MACRO GENERATES  *  SMAC0890
                    *             AN ERROR ROUTINE                   *  SMAC0900
                    *  * * * * * * * * * * * * * * * * * * * * * * *  SMAC0910
```

```
         SMAC              GENERAL WRITE MACRO          SMAC0920
      LABEL WRITE   DEVC,AREA,ERROR                    SMAC0930
00001 LABEL DEVC    3,AREA,ERROR                       SMAC0940
         MEND                                          SMAC0950


         *  * * * * * * * * * * * * * * * * * * * * * * * *  SMAC0970
         *  DISK MACRO                                    *  SMAC0980
         * GENERATES A LIBF CALL TO DISKN.                *  SMAC0990
         * TO TEST FOR DISKN BUSY YOU MUST CALL THE DISK  *  SMAC1000
         * MACRO DIRECTLY WITH A FUNC CODE OF 0.          *  SMAC1010
         *     LABEL = LABEL                              *  SMAC1020
         *     FUNC  = 1 FOR READ,3 FOR WRITE AND 0 FOR TEST SMAC1030
         *     AREA  = I/O AREA ADDRESS OR LABEL          *  SMAC1040
         *     ERROR = OPTIONAL USERS ERROR ROUTINE       *  SMAC1050
         *  * * * * * * * * * * * * * * * * * * * * * * * *  SMAC1060
         SMAC              DISK CALL GENERATOR          SMAC1070
      LABEL DISK    FUNC,AREA,ERROR                     SMAC1080
00001 LABEL LIBF    DISKN     CALL DISK SUBR            SMAC1090
00002       AIF     (FUNC EQ 1),READ  TEST FOR READ FUNC SMAC1100
00003       AIF     (FUNC EQ 3),WRITE TEST FOR WRITE FUNC SMAC1110
00004       AIF     (FUNC EQ 0),TEST  TEST FOR TEST FUNC SMAC1120
00005       ILLRQ             ILLEGAL REQUEST,ABORT CALL SMAC1130
00006       AGO     END        TERMINATE MACRO          SMAC1140
00007 READ  ANOP                                        SMAC1150
00008       DC      /1000     READ FUNC CODE            SMAC1160
00009       AGO     AREA      GO ASSEMBLE I/O AREA ADDR SMAC1170
00010 WRITE ANOP                                        SMAC1180
00011       DC      /3000     WRITE FUNC CODE           SMAC1190
00012       AGO     AREA      GO ASSEM I/O AREA ADDR    SMAC1200
00013 TEST  ANOP                                        SMAC1210
00014       DC      /0000     TEST FUNC CODE            SMAC1220
00015       DC      *+2       I/O AREA ADDR             SMAC1230
00016       MDX     *-4       BRANCH TO CONTINUE BUSY TES SMAC1240
00017       MDX     *+2       BRANCH AROUND I/O AREA    SMAC1250
00018       BSS     2         DUMMY I/O AREA            SMAC1260
00019       AGO     END       EXIT MACRO                SMAC1270
00020 AREA  ANOP                                        SMAC1280
00021       DC      AREA-1    I/O AREA ADDRESS          SMAC1290
00022 QERR  SET     ERROR      CK FOR DEFAULT ERROR     SMAC1300
00023       AIF     (QERR EQ 0),AUTOE                   SMAC1310
00024       DC      ERROR     USER SPECIFIED ERROR PARAM SMAC1320
00025       AGO     END       EXIT MACRO                SMAC1330
00026 AUTOE ANOP                                        SMAC1340
00027       AUTOE             GENERATE ERROR SUBR       SMAC1350
00028 END   ANOP                                        SMAC1360
         MEND                                          SMAC1370


         *  * * * * * * * * * * * * * * * * * * * * * * * *  SMAC1390
         *  CARD MACRO                                    *  SMAC1400
         * GENERATES A LIBF CALL TO CARD1 SUBROUTINE.     *  SMAC1410
         * TO TEST FOR CARD1 BUSY YOU MUST CALL THE CARD  *  SMAC1420
         * MACRO DIRECTLY WITH A FUNC CODE OF 0.          *  SMAC1430
         *     LABEL = LABEL                              *  SMAC1440
         *     FUNC  = 1 FOR READ,3 FOR WRITE AND 0 FOR TEST SMAC1450
         *     AREA  = I/O AREA ADDRESS OR LABEL          *  SMAC1460
         *     ERROR = OPTIONAL USERS ERROR ROUTINE       *  SMAC1470
         *  * * * * * * * * * * * * * * * * * * * * * * * *  SMAC1480
         SMAC              CARD CALL GENERATOR          SMAC1490
      LABEL CARD    FUNC,AREA,ERROR                     SMAC1500
00001 LABEL LIBF    CARD1     CALL CARD SUBR            SMAC1510
00002       AIF     (FUNC EQ 1),READ  TEST FOR READ FUNC SMAC1520
00003       AIF     (FUNC EQ 3),WRITE TEST FOR WRITE FUNC SMAC1530
00004       AIF     (FUNC EQ 0),TEST  TEST FOR TEST FUNC SMAC1540
00005       ILLRQ             ILLEGAL REQUEST,ABORT CALL SMAC1550
00006       AGO     END        TERMINATE MACRO          SMAC1560
00007 READ  ANOP                                        SMAC1570
00008       DC      /1000     READ FUNC CODE            SMAC1580
00009       AGO     AREA      GO ASSEMBLE I/O AREA ADDR SMAC1590
00010 WRITE ANOP                                        SMAC1600
```

```
00011          DC       /2000      WRITE FUNC CODE              SMAC1610
00012          AGO      AREA       GO ASSEM I/O AREA ADDR       SMAC1620
00013    TEST  ANOP                                              SMAC1630
00014          DC       /0000      TEST FUNC CODE               SMAC1640
00015          MDX      *-3        BRANCH TO CONTINUE BUSY TES  SMAC1650
00016          AGO      END        EXIT MACRO                   SMAC1660
00017    AREA  ANOP                                              SMAC1670
00018          DC       AREA-1     I/O AREA ADDRESS             SMAC1680
00019    QERR  SET      ERROR       CK FOR DEFAULT ERROR        SMAC1690
00020          AIF      (QERR EQ 0),AUTOE                       SMAC1700
00021          DC       ERROR      USER SPECIFIED ERROR PARAM   SMAC1710
00022          AGO      END        EXIT MACRO                   SMAC1720
00023    AUTOE ANOP                                              SMAC1730
00024          AUTOE               GENERATE ERROR SUBR          SMAC1740
00025    END   ANOP                                              SMAC1750
               MEND                                             SMAC1760

               *  * * * * * * * * * * * * * * * * * * * * * * *  SMAC1780
               *  PRINT MACRO                                   *  SMAC1790
               * GENERATES A LIBF CALL TO PRNT3 SUBROUTINE      *  SMAC1800
               * TO TEST FOR PRNT3 BUSY YOU MUST CALL THE PRINT *  SMAC1810
               * MACRO DIRECTLY WITH A FUNC CODE OF 0.          *  SMAC1820
               *      LABEL = LABEL                             *  SMAC1830
               *      FUNC  = 3 FOR PRINT AND 0 FOR TEST        *  SMAC1840
               *      AREA  = I/O AREA ADDRESS OR LABEL         *  SMAC1850
               *      ERROR = OPTIONAL USERS ERROR ROUTINE      *  SMAC1860
               *  * * * * * * * * * * * * * * * * * * * * * * *  SMAC1870
               SMAC                PRINT CALL GENERATOR         SMAC1880
               LABEL PRINT   FUNC,AREA,ERROR                    SMAC1890
00001          LABEL LIBF    PRNT3      CALL PRINT SUBR          SMAC1900
00002          AIF      (FUNC EQ 3),WRITE TEST FOR WRITE FUNC   SMAC1910
00003          AIF      (FUNC EQ 0),TEST  TEST FOR TEST FUNC    SMAC1920
00004          ILLRQ               ILLEGAL REQUEST,ABORT CALL   SMAC1930
00005          AGO      END         TERMINATE MACRO             SMAC1940
00006    WRITE ANOP                                              SMAC1950
00007          DC       /2000      WRITE FUNC CODE              SMAC1960
00008          AGO      AREA       GO ASSEM I/O AREA ADDR       SMAC1970
00009    TEST  ANOP                                              SMAC1980
00010          DC       /0000      TEST FUNC CODE               SMAC1990
00011          MDX      *-3        BRANCH TO CONTINUE BUSY TES  SMAC2000
00012          AGO      END        EXIT MACRO                   SMAC2010
00013    AREA  ANOP                                              SMAC2020
00014          DC       AREA-1     I/O AREA ADDRESS             SMAC2030
00015    QERR  SET      ERROR       CK FOR DEFAULT ERROR        SMAC2040
00016          AIF      (QERR EQ 0),AUTOE                       SMAC2050
00017          DC       ERROR      USER SPECIFIED ERROR PARAM   SMAC2060
00018          AGO      END        EXIT MACRO                   SMAC2070
00019    AUTOE ANOP                                              SMAC2080
00020          AUTOE               GENERATE ERROR SUBR          SMAC2090
00021    END   ANOP                                              SMAC2100
               MEND                                             SMAC2110


               *  * * * * * * * * * * * * * * * * * * * * * * *  SMAC2130
               *  CONVERT MACRO                                 *  SMAC2140
               * THIS MACRO HANDLES                             *  SMAC2150
               *   HOLLERITH TO PRINTER CODE VIA 'HOLPR'        *  SMAC2160
               *   1 BINARY TO 6 HOLLERITH CHARS VIA 'BINDC'    *  SMAC2170
               *   6 HOLLERITH TO 1 BINARY CHAR VIA 'DCBIN'     *  SMAC2180
               * WHERE                                          *  SMAC2190
               *      AREA  = INPUT AREA  (OUTPUT FOR BINDC)    *  SMAC2200
               *      RT    = CONVERSION RT. HOLPR,BINDC OR DCBIN*  SMAC2210
               *      +PRNTR= OUTPUT  CODE FOR HOLPR            *  SMAC2220
               *              0 FOR CONSOLE, 1 FOR 1403 PRINTER *  SMAC2230
               *      OUTPT = OUTPUT AREA FOR HOLPR             *  SMAC2240
               *      CHCT  = CHARACTER COUNT FOR HOLPR         *  SMAC2250
               *  * * * * * * * * * * * * * * * * * * * * * * *  SMAC2260
               SMAC                CODE CONVERSION CALL         SMAC2270
               LABEL CNVRT   AREA,RT,+PRNTR,OUTPT,CHCT          SMAC2280
00001          LABEL LIBF    RT                                 SMAC2290
00002          AGO      RT         GEN APPROPRIATE CODE         SMAC2300
00003    HOLPR ANOP                STOP HERE FOR HOLPR          SMAC2310
00004          DC       /+PRNTR    CONVERSION CODE              SMAC2320
00005          DC       AREA       INPUT                        SMAC2330
```

```
00006          DC       OUTPT       OUTPUT                          SMAC2340
00007          DC       CHCT        CHAR COUNT                      SMAC2350
00008          AGO      END                                         SMAC2360
00009    RT    ANOP     STOP HERE FOR BINDC AND DCBIN               SMAC2370
00010          DC       AREA         I/O AREA                       SMAC2380
         END   MEND                                                 SMAC2390


         *  * * * * * * * * * * * * * * * * * * * * * * * * * *    SMAC2410
         *  BLOCK MACRO                                          *  SMAC2420
         * GENERATE A BLOCK OF CONSTANTS.                        *  SMAC2430
         *      A      = CONSTANT TO FILL BLOCK WITH             *  SMAC2440
         *      B      = NUMBER OF CONSTANTS TO GENERATE         *  SMAC2450
         *  * * * * * * * * * * * * * * * * * * * * * * * * * *    SMAC2460
               SMAC                 CONSTANT DATA BLOCK           SMAC2470
         LABEL BLOCK    A,B         GENERATOR                     SMAC2480
00001          LIST     OFF                                        SMAC2490
00002    COUNT SET      B                                          SMAC2500
00003    DATA  ANOP                                                SMAC2510
00004          LIST                                                SMAC2520
00005          DC       A                                          SMAC2530
00006          LIST     OFF                                        SMAC2540
00007    COUNT SET      COUNT-1                                    SMAC2550
00008          AIFB     (COUNT GT 0),DATA                          SMAC2560
00009          LIST                                                SMAC2570
               MEND                                               SMAC2580


         *  * * * * * * * * * * * * * * * * * * * * * * * * * *    SMAC2600
         *  INREG MACRO                                          SMAC2610
         * LOAR AN INDEX REGISTER WITH A VALUE                   *  SMAC2620
         *      A = THE VALUE, B = THE REGISTER NUMBER           *  SMAC2630
         *  * * * * * * * * * * * * * * * * * * * * * * * * * *    SMAC2640
               SMAC                 SET REGISTER B TO VALUE A     SMAC2650
         A     INREG    B                                          SMAC2660
00001          AIF      (A GT 127),LONG                            SMAC2670
00002          AIF      (A LT -128),LONG                           SMAC2680
00003          LDX    B A                                          SMAC2690
00004          AGO      END                                        SMAC2700
00005    LONG  ANOP                                                SMAC2710
00006          LDX   LB A                                         SMAC2720
00007    END   ANOP                                                SMAC2730
               MEND                                               SMAC2740


         *  * * * * * * * * * * * * * * * * * * * * * * * * * *    SMAC2760
         *  DECR MACRO                                           *  SMAC2770
         * DECREMENT A COUNTER FOR LOOP CONTROL.                 *  SMAC2780
         *      A      = STORAGE ADDRESS OF COUNTER, IF ANY      *  SMAC2790
         *      B      = REGISTER NUMBER                            SMAC2800
         *      C      = THE DECREMENT VALUE                     *  SMAC2810
         *  * * * * * * * * * * * * * * * * * * * * * * * * * *    SMAC2820
               SMAC                 DECREMENT COUNTER             SMAC2830
         A     DECR     B,C                                        SMAC2840
00001          AIF      (B EQ 0),LONG  TEST FOR INDEXING           SMAC2850
00002          MDX    B -C             INDEXED COUNTER            SMAC2860
00003          AGO      END                                        SMAC2870
00004    LONG  ANOP                                                SMAC2880
00005          MDX   L  A,-C           NON-INDEXED COUNTER        SMAC2890
00006    END   ANOP                                                SMAC2900
               MEND                                               SMAC2910
```

```
                         *  * * * * * * * * * * * * * * * * * * * * * * * * * * * *  SMAC2930


                         *            SAMPLE PROGRAM                                 SMAC2950
                         *      PROGRAM WHICH                                        SMAC2960
                         *       A- READS ONE SECTOR DATA FILE WHICH CONTAINS        SMAC2970
                         *          INTEGER DATA FROM DISK                           SMAC2980
                         *       B- READS EVERY 5TH ENTRY FROM CARD                  SMAC2990
                         *       C- PUNCHES EVERY 3RD ENTRY TO CARD                  SMAC3000
                         *       D- ADDS ONE TO EVERY OTHER ENTRY BEGINNING          SMAC3010
                         *          WITH SECOND ENTRY                                SMAC3020
                         *       E- WRITES MODIFIED FILE TO PRINTER,5 ENTRIES        SMAC3030
                         *          PER LINE                                         SMAC3040
                         *                                                           SMAC3050


                         *  * * * * * * * * * * * * * * * * * * * * * * * * * * * *  SMAC3070
                         *                                                           SMAC3090
                         *   A - READ THE DISK DATA FILE                         *   SMAC3100
                         *                                                           SMAC3110
                         STEPA READ    DISK,FILE,0  READ FILE                       SMAC3120
                        +STEPA DISK    1,FILE,0
0000 20 04262495        +STEPA LIBF    DISKN
                        +      AIF     (1 EQ 1),READ
                        +READ  ANOP
0001 0  1000            +      DC      /1000
                        +      AGO     FILE
                        +FILE  ANOP
0002 1  0061            +      DC      FILE-1
0000                    +QERR  SET     0
                        +      AIF     (QERR EQ 0),AUTOE
                        +AUTOE ANOP
                        +      AUTOE
0003 1  0005            +      DC      *+1
0004 0  7003            +      MDX     *+3
0005 0  0000            +      DC      *-*
0006 01 4C800005        +      BSC  I  *-3
                        +END   ANOP
                         316   INREG   2          SET LOOP CONTROL                 SMAC3130
                        +      AIF     (316 GT 127),LONG
                        +LONG  ANOP
0008 00 6600013C        +      LDX  L2 316
                        +END   ANOP


                         *                                                           SMAC3150
                         *   B - READ DATA CARDS AND STO IN EVERY 5TH ENTRY          SMAC3160
                         *                                                           SMAC3170
                         STEPB READ    CARD,DATA    READ DATA FROM CARD             SMAC3180
                        +STEPB CARD    1,DATA,
000A 20 03059131        +STEPB LIBF    CARD1
                        +      AIF     (1 EQ 1),READ
                        +READ  ANOP
000B 0  1000            +      DC      /1000
                        +      AGO     DATA
                        +DATA  ANOP
000C 1  01A4            +      DC      DATA-1
0000                    +QERR  SET
                        +      AIF     (QERR EQ 0),AUTOE
                        +AUTOE ANOP
                        +      AUTOE
000D 1  000F            +      DC      *+1
000E 0  7003            +      MDX     *+3
000F 0  0000            +      DC      *-*
0010 01 4C80000F        +      BSC  I  *-3
                        +END   ANOP
                               CARD    0          BUSY TEST                        SMAC3190
0012 20 03059131        +      LIBF    CARD1
                        +      AIF     (0 EQ 1),READ
                        +      AIF     (0 EQ 3),WRITE
```

```
                    +       AIF     (0 EQ 0),TEST
                    +TEST   ANOP
0013 0  0000        +       DC      /0000
0014 0  70FD        +       MDX     *-3
                    +       AGO     END
                    +END    ANOP
                            CNVRT   DATA,DCBIN CONVERT  1ST 5 CARD COLUM  SMAC3200
0015 20 040C2255    +       LIBF    DCBIN
                    +       AGO     DCBIN
                    +DCBIN  ANOP    STOP
0016 1  01A5        +       DC      DATA
0017 01 D6000062            STO  L2 FILE        SAVE AS 5TH CHARACTER     SMAC3210
                            DECR    2,5        DECREMENT LOOP CONTROL     SMAC3220
                    +       AIF     (2 EQ 0),LONG
0019 0  72FB        +       MDX   2 -5
                    +       AGO     END
                    +END    ANOP
001A 0  70EF                B       STEPB       CONTINUE STEP B           SMAC3230
                     *                          PRIME 1442 FOR PUNCHING   SMAC3240
                            READ    CARD,DATA   READ ONE BLANK CARD       SMAC3250
                    +       CARD    1,DATA,
001B 20 03059131    +       LIBF    CARD1
                    +       AIF     (1 EQ 1),READ
                    +READ   ANOP
001C 0  1000        +       DC      /1000
                    +       AGO     DATA
                    +DATA   ANOP
001D 1  01A4        +       DC      DATA-1
0000                +QERR   SET
                    +       AIF     (QERR EQ 0),AUTOE
                    +AUTOE  ANOP
                    +       AUTOE
001E 1  0020        +       DC      *+1
001F 0  7003        +       MDX     *+3
0020 0  0000        +       DC      *-*
0021 01 4C800020    +       BSC  I  *-3
                    +END    ANOP
                     319    INREG   2           SET LOOP CONTROL          SMAC3260
                    +       AIF     (319 GT 127),LONG
                    +LONG   ANOP
0023 00 6600013F    +       LDX  L2 319
                    +END    ANOP


                     *                                                    SMAC3280
                     *   C - CONVERT AND PUNCH EVERY 3RD ENTRY            SMAC3290
                     *                                                    SMAC3300
0025 01 C6000061     STEPC LD   L2 FILE-1    CONVERT INTEGER TO           SMAC3310
                           CARD    0        BUSY TEST                    SMAC3320
0027 20 03059131    +       LIBF    CARD1
                    +       AIF     (0 EQ 1),READ
                    +       AIF     (0 EQ 3),WRITE
                    +       AIF     (0 EQ 0),TEST
                    +TEST   ANOP
0028 0  0000        +       DC      /0000
0029 0  70FD        +       MDX     *-3
                    +       AGO     END
                    +FND    ANOP
                            CNVRT   DATA,BINDC CARD CODE                 SMAC3330
002A 20 02255103    +       LIBF    BINDC
                    +       AGO     BINDC
                    +BINDC  ANOP    STOP
002B 1  01A5        +       DC      DATA
                            WRITE   CARD,DATA    PUNCH DATA WORD         SMAC3340
                    +       CARD    3,DATA,
002C 20 03059131    +       LIBF    CARD1
                    +       AIF     (3 EQ 1),READ
                    +       AIF     (3 EQ 3),WRITE
                    +WRITE  ANOP
002D 0  2000        +       DC      /2000
                    +       AGO     DATA
                    +DATA   ANOP
002E 1  01A4        +       DC      DATA-1
0000                +QERR   SET
                    +       AIF     (QERR EQ 0),AUTOE
```

```
                    +AUTOE ANOP
                    +      AUTOE
002F 1  0031        +      DC      *+1
0030 0  7003        +      MDX     *+3
0031 0  0000        +      DC      *-*
0032 01 4C800031    +      BSC  I  *-3
                    +END   ANOP
                           DECR    2,3       DECR LOOP CONTROL            SMAC3350
                    +      AIF     (2 EQ 0),LONG
0034 0  72FD        +      MDX   2 -3
                    +      AGO     END
                    +END   ANOP
0035 0  70EF               B       STEPC      CONTINUE STEP C              SMAC3360
 319                INREG   2          SET LOOP CONTROL                    SMAC3370
                    +      AIF     (319 GT 127),LONG
                    +LONG  ANOP
0036 00 6600013F    +      LDX  L2 319
                    +END   ANOP



                     *                                                     SMAC3390
                     *   D - ADD ONE TO EVERY OTHER ENTRY                *  SMAC3400
                     *                                                     SMAC3410
0038 01 C6000062     STEPD LD   L2 FILE      ADD 1 TO EVERY OTHER         SMAC3420
003A 0  8025               A       ONE       DATA ENTRY STARTING WITH     SMAC3430
003B 01 D6000062           STO  L2 FILE      THE 2ND ENTRY                SMAC3440
                           DECR    2,2                                    SMAC3450
                    +      AIF     (2 EQ 0),LONG
003D 0  72FE        +      MDX   2 -2
                    +      AGO     END
                    +END   ANOP
003E 0  70F9               B       STEPD     CONTINUE STEP D              SMAC3460
 -320               INREG   1          SET OUTER LOOP COUNT                SMAC3470
                    +      AIF     (-320 GT 127),LONG
                    +      AIF     (-320 LT -128),LONG
                    +LONG  ANOP
003F 00 6500FEC0    +      LDX  L1 -320
                    +END   ANOP
 5                  INREG   2          SET A TO 5                          SMAC3480
                    +      AIF     (5 GT 127),LONG
                    +      AIF     (5 LT -128),LONG
0041 0  6205        +      LDX   2 5
                    +      AGO     END
                    +END   ANOP

                     *                                                     SMAC3500
                     *   E - WRITE FILE TO 1403                          *  SMAC3510
                     *                                                     SMAC3520
0042 01 C50001A3     STEP2 LD   L1 FILE+321  FETCH DATA WORD              SMAC3530
                           CNVRT   DATA+7,BINDC  CONVERT DATA TO 1403     SMAC3540
0044 20 02255103    +      LIBF    BINDC
                    +      AGO     BINDC
                    +BINDC ANOP    STOP
0045 1  01AC        +      DC      DATA+7
                           CNVRT   DATA+7,HOLPR,1,PRINT,6 CODE            SMAC3550
0046 20 085935D9    +      LIBF    HOLPR
                    +      AGO     HOLPR
                    +HOLPR ANOP
0047 0  0001        +      DC      /1
0048 1  01AC        +      DC      DATA+7
0049 1  01F6        +      DC      PRINT
004A 0  0006        +      DC      6
                    +      AGO     END
0049                OUTPT  EQU     *-2                                    SMAC3560
004B 01 74040049           MDX  L  OUTPT,4   ADJUST BUFFER ADDR           SMAC3570
                    STEPF  DECR    2,1       DECR LINE COUNT              SMAC3580
                    +      AIF     (2 EQ 0),LONG
004D 0  72FF        +      MDX   2 -1
                    +      AGO     END
                    +END   ANOP
004E 0  700E               B       STEPG     CONTINUE TO FILL BUFFER      SMAC3590
                    OUTPT  DECR    0,20      RESET BUFFER ADDR,NEW LINE   SMAC3600
                    +      AIF     (0 EQ 0),LONG
                    +LONG  ANOP
```

```
004F 01 74EC0049    +       MDX   L  OUTPT,-20
                    +END    ANOP
                            PRINT   0          BUSY TEST                    SMAC3610
0051 20 176558F3    +       LIBF    PRNT3
                    +       AIF     (0 EQ 3),WRITE
                    +       AIF     (0 EQ 0),TEST
                    +TEST   ANOP
0052 0  0000        +       DC      /0000
0053 0  70FD        +       MDX     *-3
                    +       AGO     END
                    +END    ANOP
                            WRITE   PRINT,PRINT    PRINT LINE            SMAC3620
                    +       PRINT   3,PRINT,
0054 20 176558F3    +       LIBF    PRNT3
                    +       AIF     (3 EQ 3),WRITE
                    +WRITE  ANOP
0055 0  2000        +       DC      /2000
                    +       AGO     PRINT
                    +PRINT  ANOP
0056 1  01F5        +       DC      PRINT-1
0000                +QERR   SET
                    +       AIF     (QERR EQ 0),AUTOE
                    +AUTOE  ANOP
                    +       AUTOE
0057 1  0059        +       DC      *+1
0058 0  7003        +       MDX     *+3
0059 0  0000        +       DC      *-*
005A 01 4C800059    +       BSC  I  *-3
                    +END    ANOP
                     5      INREG   2          RESET LINE COUNT             SMAC3630
                    +       AIF     (5 GT 127),LONG
                    +       AIF     (5 LT -128),LONG
005C 0  6205        +       LDX   2 5
                    +       AGO     END
                    +END    ANOP
005D 0  7101         STEPG  MDX   1 1          INCR DATA POINTER            SMAC3640
005E 0  70E3                MDX     STEP2     CONTINUE TO PRINT            SMAC3650
005F 0  6038         EXIT   EXIT               EXIT-END OF JOB              SMAC3660
                     *  * * * * * * * * * * * * * * * * * * * * * * * * *  SMAC3670
                     *                 CONSTANTS AND BUFFERS                SMAC3680
                     *                                                      SMAC3690
0060 0  0001         ONE    DC      1          CONSTANT 1                   SMAC3700
0062                 FILE   EQU     *+1        DATA FILE SECTOR ADDRESS     SMAC3710
0061 31 06253171            DSA     FILE1                                   SMAC3720
0064    0140                BSS     320        DATA AREA                    SMAC3730
01A4 0  0050                DC      80                                      SMAC3740
01A5    0050         DATA   BSS     80          CARD BUFFER                 SMAC3750
01F5 0  0014                DC      20                                      SMAC3760
01F6                 PRINT  EQU     *                                       SMAC3770
                            BLOCK   /7F7F,20   BLANK PRINT BUFFER           SMAC3780
01F6 0  7F7F        +       DC      /7F7F
01F7 0  7F7F        +       DC      /7F7F
01F8 0  7F7F        +       DC      /7F7F
01F9 0  7F7F        +       DC      /7F7F
01FA 0  7F7F        +       DC      /7F7F
01FB 0  7F7F        +       DC      /7F7F
01FC 0  7F7F        +       DC      /7F7F
01FD 0  7F7F        +       DC      /7F7F
01FE 0  7F7F        +       DC      /7F7F
01FF 0  7F7F        +       DC      /7F7F
0200 0  7F7F        +       DC      /7F7F
0201 0  7F7F        +       DC      /7F7F
0202 0  7F7F        +       DC      /7F7F
0203 0  7F7F        +       DC      /7F7F
0204 0  7F7F        +       DC      /7F7F
0205 0  7F7F        +       DC      /7F7F
0206 0  7F7F        +       DC      /7F7F
0207 0  7F7F        +       DC      /7F7F
0208 0  7F7F        +       DC      /7F7F
0209 0  7F7F        +       DC      /7F7F
020A    0000                END     STEPA                                   SMAC3790

   000 OVERFLOW SECTORS SPECIFIED
   000 OVERFLOW SECTORS REQUIRED
```

```
   014 SYMBOLS DEFINED
    NO ERROR(S) AND   NO WARNING(S)  FLAGGED IN ABOVE ASSEMBLY

// XEQ       L  1 N                                                    SMAC3800

*FILES(1,FILE1)                                                       SMAC3810
R 41  761E (HEX) WDS UNUSED BY CORE LOAD
LIBF TRANSFER VECTOR
 PRTY   095C
 HOLL   090C
 PRNT3  07D8
 HOLPR  0774
 BINDC  072C
 DCBIN  06D4
 CARD1  05CA
 DISKN  00F9
SYSTEM SUBROUTINES
 ILS04  00C4
 ILS02  00B3
 ILS00  09AD
      03C0 (HEX) IS THE EXECUTION ADDR
-00319  +00319  +00319  +00317  +00317
-00315  +00315  +00313  +00313  +00311
-00309  +00309  +00309  +00307  +00307
-00305  +00305  +00303  +00303  +00301
-00299  +00299  +00299  +00297  +00297
-00295  +00295  +00293  +00293  +00291
-00289  +00289  +00289  +00287  +00287
-00285  +00285  +00283  +00283  +00281
-00279  +00279  +00279  +00277  +00277
-00275  +00275  +00273  +00273  +00271
-00269  +00269  +00269  +00267  +00267
-00265  +00265  +00263  +00263  +00261
-00259  +00259  +00259  +00257  +00257
-00255  +00255  +00253  +00253  +00251
-00249  +00249  +00249  +00247  +00247
-00245  +00245  +00243  +00243  +00241
-00239  +00239  +00239  +00237  +00237
-00235  +00235  +00233  +00233  +00231
-00229  +00229  +00229  +00227  +00227
-00225  +00225  +00223  +00223  +00221
-00219  +00219  +00219  +00217  +00217
-00215  +00215  +00213  +00213  +00211
-00209  +00209  +00209  +00207  +00207
-00205  +00205  +00203  +00203  +00201
-00199  +00199  +00199  +00197  +00197
-00195  +00195  +00193  +00193  +00191
-00189  +00189  +00189  +00187  +00187
-00185  +00185  +00183  +00183  +00181
-00179  +00179  +00179  +00177  +00177
-00175  +00175  +00173  +00173  +00171
-00169  +00169  +00169  +00167  +00167
-00165  +00165  +00163  +00163  +00161
-00159  +00159  +00159  +00157  +00157
-00155  +00155  +00153  +00153  +00151
-00149  +00149  +00149  +00147  +00147
-00145  +00145  +00143  +00143  +00141
-00139  +00139  +00139  +00137  +00137
-00135  +00135  +00133  +00133  +00131
-00129  +00129  +00129  +00127  +00127
-00125  +00125  +00123  +00123  +00121
-00119  +00119  +00119  +00117  +00117
-00115  +00115  +00113  +00113  +00111
-00109  +00109  +00109  +00107  +00107
-00105  +00105  +00103  +00103  +00101
-00099  +00099  +00099  +00097  +00097
-00095  +00095  +00093  +00093  +00091
-00089  +00089  +00089  +00087  +00087
-00085  +00085  +00083  +00083  +00081
-00079  +00079  +00079  +00077  +00077
-00075  +00075  +00073  +00073  +00071
-00069  +00069  +00069  +00067  +00067
```

```
-00065  +00065  +00063  +00063  +00061
-00059  +00059  +00059  +00057  +00057
-00055  +00055  +00053  +00053  +00051
-00049  +00049  +00049  +00047  +00047
-00045  +00045  +00043  +00043  +00041
-00039  +00039  +00039  +00037  +00037
-00035  +00035  +00033  +00033  +00031
-00029  +00029  +00029  +00027  +00027
-00025  +00025  +00023  +00023  +00021
-00019  +00019  +00019  +00017  +00017
-00015  +00015  +00013  +00013  +00011
-00009  +00009  +00009  +00007  +00007
-00005  +00005  +00003  +00003  +00001
```

INPUT DATA CARDS

```
-     5                                                    SMAC3820
-    10                                                    SMAC3830
-    15                                                    SMAC3840
-    20                                                    SMAC3850
-    25                                                    SMAC3860
-    30                                                    SMAC3870
-    35                                                    SMAC3880
-    40                                                    SMAC3890
-    45                                                    SMAC3900
-    50                                                    SMAC3910
-    55                                                    SMAC3920
-    60                                                    SMAC3930
-    65                                                    SMAC3940
-    70                                                    SMAC3950
-    75                                                    SMAC3960
-    80                                                    SMAC3970
-    85                                                    SMAC3980
-    90                                                    SMAC3990
-    95                                                    SMAC4000
-   100                                                    SMAC4010
-   105                                                    SMAC4020
-   110                                                    SMAC4030
-   115                                                    SMAC4040
-   120                                                    SMAC4050
-   125                                                    SMAC4060
-   130                                                    SMAC4070
-   135                                                    SMAC4080
-   140                                                    SMAC4090
-   145                                                    SMAC4100
-   150                                                    SMAC4110
-   155                                                    SMAC4120
-   160                                                    SMAC4130
-   165                                                    SMAC4140
-   170                                                    SMAC4150
-   175                                                    SMAC4160
-   180                                                    SMAC4170
-   185                                                    SMAC4180
-   190                                                    SMAC4190
-   195                                                    SMAC4200
-   200                                                    SMAC4210
-   205                                                    SMAC4220
-   210                                                    SMAC4230
-   215                                                    SMAC4240
-   220                                                    SMAC4250
-   225                                                    SMAC4260
-   230                                                    SMAC4270
-   235                                                    SMAC4280
-   240                                                    SMAC4290
-   245                                                    SMAC4300
-   250                                                    SMAC4310
-   255                                                    SMAC4320
-   260                                                    SMAC4330
-   265                                                    SMAC4340
-   270                                                    SMAC4350
-   275                                                    SMAC4360
-   280                                                    SMAC4370
-   285                                                    SMAC4380
```

```
-  290                                                          SMAC4390
-  295                                                          SMAC4400
-  300                                                          SMAC4410
-  305                                                          SMAC4420
-  310                                                          SMAC4430
-  315                                                          SMAC4440
-  320                                                          SMAC4450
PUNCHED OUTPUT
+00003
+00006
+00009
+00012
-00015
+00018
+00021
+00024
+00027
-00030
+00033
+00036
+00039
+00042
-00045
+00048
+00051
+00054
+00057
-00060
+00063
+00066
+00069
+00072
-00075
+00078
+00081
+00084
+00087
-00090
+00093
+00096
+00099
+00102
-00105
+00108
+00111
+00114
+00117
-00120
+00123
+00126
+00129
+00132
-00135
+00138
+00141
+00144
+00147
-00150
+00153
+00156
+00159
+00162
-00165
+00168
+00171
+00174
+00177
-00180
+00183
+00186
+00189
```

```
+00192
-00195
+00198
+00201
+00204
+00207
-00210
+00213
+00216
+00219
+00222
-00225
+00228
+00231
+00234
+00237
-00240
+00243
+00246
+00249
+00252
-00255
+00258
+00261
+00264
+00267
-00270
+00273
+00276
+00279
+00282
-00285
+00288
+00291
+00294
+00297
-00300
+00303
+00306
+00309
+00312
-00315
+00318
+00602 +00307
```

IBM 1130/1800 Printed in U.S.A. GC26-3733-0

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

READER'S COMMENT FORM

IBM 1130/1800 Macro Assembler Programming — Order Number GC26-3733-0

- Please comment on the usefulness and readability of this book, suggest additions and deletions, and list specific errors and omissions (give page numbers). All comments and suggestions become the property of IBM. If you want a reply, be sure to give your name and address.

Name _____ Occupation _____

Address _____

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

YOUR COMMENTS, PLEASE...

This publication is one of a series which serves as reference sources for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note: Requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

fold fold

FIRST CLASS
PERMIT NO. 2078
SAN JOSE, CALIF.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

POSTAGE WILL BE PAID BY . . .

IBM Corporation
Monterey & Cottle Rds.
San Jose, California
95114

Attention: Programming Publications, Dept. D78

fold fold

IBM 1130/1800 Printed in U.S.A. GC26-3733-0

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

READER'S COMMENT FORM

IBM 1130/1800 Macro Assembler Programming — Order Number GC26-3733-0

- Please comment on the usefulness and readability of this book, suggest additions and deletions, and list specific errors and omissions (give page numbers). All comments and suggestions become the property of IBM. If you want a reply, be sure to give your name and address.

Name _____ Occupation _____

Address _____

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

GC26-3733-0

YOUR COMMENTS, PLEASE...

This publication is one of a series which serves as reference sources for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note: Requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

fold fold

FIRST CLASS
PERMIT NO. 2078
SAN JOSE, CALIF.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

POSTAGE WILL BE PAID BY . . .

IBM Corporation
Monterey & Cottle Rds.
San Jose, California
95114

Attention: Programming Publications, Dept. D78

fold fold

IBM 1130/1800 Printed in U.S.A. GC26-3733-0

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

READER'S COMMENT FORM

IBM 1130/1800 Macro Assembler Programming — Order Number GC26-3733-0

- Please comment on the usefulness and readability of this book, suggest additions and deletions, and list specific errors and omissions (give page numbers). All comments and suggestions become the property of IBM. If you want a reply, be sure to give your name and address.

Name _____ Occupation _____

Address _____

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

YOUR COMMENTS, PLEASE...

This publication is one of a series which serves as reference sources for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note: Requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

fold fold

FIRST CLASS
PERMIT NO. 2078
SAN JOSE, CALIF.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

POSTAGE WILL BE PAID BY . . .

IBM Corporation
Monterey & Cottle Rds.
San Jose, California
95114

Attention: Programming Publications, Dept. D78

fold fold

IBM 1130/1800 Printed in U.S.A. GC26-3733-0

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

IBM **Technical Newsletter**

File Number	1130/1800-21 (MPX Version 3)
Re: Order Number	GC26-3733-0
This Newsletter Number	GN26-0610
Date	July 20, 1970
Previous Newsletter Numbers	None

IBM 1130/1800 MACRO ASSEMBLER PROGRAMMING

This technical newsletter provides pages for IBM 1130/1800 Macro Assembler Programming (Order Number GC26-3733-0). Pages to be replaced are listed below:

iii-iv
63-66

A change to the text is indicated by a vertical line in the left margin. A revised illustration is indicated by a bullet to the left of the caption.

Summary of Amendments

The Macro Assembler Error Flags table and Error Codes and Messages table have been updated to reflect the changes made in Version 3 of the 1800 Multiprogramming Executive Operating System.

Please put this cover letter at the back of the manual to provide a record of changes.

IBM Corporation, Programming Publications, Dept. D78, San Jose, Calif. 95114

PRINTED IN U.S.A.