17328900

CONTROL DATA CORPORATION

MASS/MPSIM REFERENCE MANUAL

CONTROL DATA®
CYBER 170 SYSTEMS
CYBER 70 COMPUTER SYSTEMS
MODELS 71, 72, 73, 74
6000 COMPUTER SYSTEMS

REVISION RECORD

REVISION	DESCRIPTION
01	Original release describing the general operation of the MP Micro ASSembler (MASS) and providing necessary instruction for preparing programs
(76-05-21)	for assembly.
02	This manual is being reissued to clarify and elaborate on the use of the Micro Assembler (MASS) to generate binary microcode to be used on the
	MP32 and on the Micro Simulator (MPSIM) which is used to debug MP32 microcode.
Publication No. 17328900	

REVISION LETTERS I, O, Q AND X ARE NOT USED

Address comments concerning this manual to:

CONTROL DATA CORPORATION

**215 MOFFETT PARK DRIVE
SUNNYVALE, CALIFORNIA 94086**

or use Comment Sheet in the back of this manual

PREFACE

The MP Micro ASSembler (MASS) is a component of the MP-60 Software package. It is intended to assemble microcode for the CDC Microprogrammable Processor, herein called the MP. The assembler operates under control of the CYBER 170/70L/6000 NOS/BE 1.0 Operating System and the CYBER 170/70L/6000 NOS 1.0 Operating System. A separate version of the MP Micro Assembler executes on the MP-60 Series computer.

This assembler manual is to be used in conjunction with the Basic System Controller Model 65109 Hardware Reference Manual (see list of publications below). This manual will describe the general operation of the assembler and provide necessary instruction for preparing programs for assembly. No attempt is made here to provide a programmer's guide; therefore, examples will be limited. It is assumed that the reader is already familiar with the operation of the MP computer. A detailed description of the MP computer may be found in the above-mentioned Basic System Controller Model 65109 Hardware Reference Manual.

Following is a list of related manuals:

Manuals	CDC Publication No.
NOS 1.0 Reference Manual Vol. I	60435400
NOS 1.0 Reference Manual Vol. II	60445300
NOS/BE 1.0 Reference Manual	60493800
Basic System Controller Model 65109 Hardware Reference Manual	41618400

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features or undefined parameters.

CONTENTS

CONTENTS (Cont'd)

CONTENTS (Cont'd)

CONTENTS (Cont'd)

CONTENTS (Cont'd)

CONTENTS (Cont'd)

INTRODUCTION

The assembler for the CONTROL DATA Microprogrammable Processors provides the mnemonic language necessary for the programmer to write a microprogram. The assembler translates symbolic source program instructions into object machine instructions and provides a listing of assembly results.

The characteristics of the assembler as written for the CYBER 170/70L/6000 and MP-60 Series computers are described. This assembler is based on the MICRO-71 assembler for the MPP computer.

Input to this assembler consists of one or more source programs followed by a FINIS card. Each program begins with an IDENT card and is terminated with an END card. Each program is coded using these basic elements:

- Symbols
- Constants
- Pseudo Instructions
- Mnemonic Instructions

The basic elements are punched into a card in specific fields, always left-justified within the field.

Output from the assembler consists of the following:

- Assembly listing including diagnostics
- Zero location map
- Deadstart object time
- User generate micro binary

LANGUAGE STRUCTURE 2

STATEMENT FORMAT

A source input statement to the MP assembler (MASS) consists of 11 fields, as contained in Table 2-1 and as illustrated in the coding form depicted in Figure 2-1. Of these fields, the Q (qualifier), location, and comment fields are used to improve the documentation of the assembled microinstructions, while the remaining eight fields correspond to the eight fields of the MP microinstruction shown in the Basic System Controller Model 65109 Hardware Reference Manual. The eight fields used on the input form are in the same order that the programmer will tend to use in preparing microinstructions for a microprogram.

Information entered in each field (if anything is entered) is entered left-justified with blank fill. Information which is not entered left-justified will not be processed correctly by the assembler.

Q FIELD

The Q field is column 1 of the input coding form. This field may contain an asterisk (*), dollar sign ($), plus sign (+), minus sign (-), or it may be blank.

An asterisk (*) or dollar sign ($) specifies that the rest of the input source statement is a remark and that the remaining 79 columns contain comments. This qualifier allows the remarks card to be printed on the listing with no effect on the assembler object code output. One line is skipped before and after a single comment card or group of comment cards.

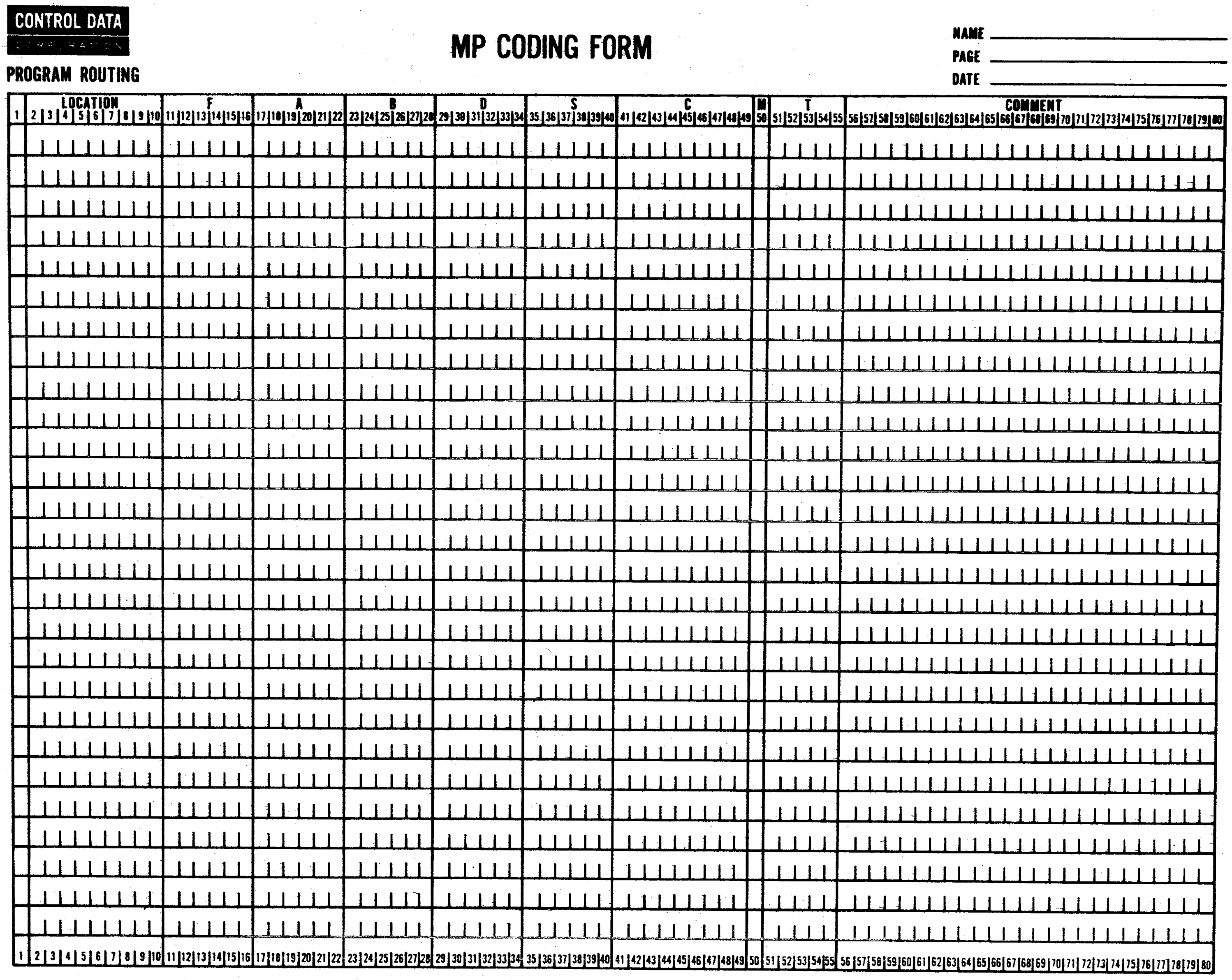
A plus sign (+) in the Q field locates the resulting microinstruction as the upper instruction of a microinstruction pair.

A minus sign (-) in the Q field locates the resulting microinstruction as the lower instruction of a microinstruction pair.

A blank in the Q field locates the resulting microinstruction in the next available half of a microinstruction pair.

TABLE 2-1. SOURCE STATEMENT FIELDS

FIELDS	COLUMNS	COMMENTS
Q	1	The qualifier field may specify whether the (qualifier) statement is a comment, an upper instruction, or lower instruction.
Location	2 through 9	The location field specifies the statement's symbolic address in this program.
F (function)	11 through 16	The function field specifies a logical, arithmetic, shift or scale operation that is performed by the arithmetic and logic unit (ALU) on two sources and placed in a destination.
A	17 through 22	Specifies the A source of the function.
B	23 through 28	Specifies the B source of the function.
D	29 through 34	Specifies the destination of the result of the ALU.
S	35 through 40	The special field provides special instruction modes that either: • Extend the A, B, and D fields, or • provide a special command which is performed in parallel with the data transfers taking place in the ALU.
C (constant)	41 through 49	The constant field specifies another special command that is performed independently of the rest of the instruction; it is executed in parallel with the rest of the instruction.
M	50	The mode field specifies the addressing method for obtaining the next instruction pair: .sequential, jump or return.
T (test)	51 through 55	The test field is the conditional branch of the instruction and specifies which instruction (upper or lower) of the next instruction pair to execute. The test and branch are executed after the rest of the instruction has executed.
Comment	56 through 80	The comment field is used for remarks that are printed as part of the list output.



CONTROL DATA
CORPORATION

PROGRAM ROUTING

MP CODING FORM

NAME ____________
PAGE ____________
DATE ____________

	LOCATION	F	A	B	D	S	C	M	T	COMMENT
1	2 3 4 5 6 7 8 9 10	11 12 13 14 15 16	17 18 19 20 21 22	23 24 25 26 27 28	29 30 31 32 33 34	35 36 37 38 39 40	41 42 43 44 45 46 47 48 49	50	51 52 53 54 55	56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

Figure 2-1. MP Coding Form

Each micromemory location (address) contains space for two microinstructions. These are referred to as the upper and the lower microinstruction. If column 1 is blank, the statement is assigned to the next available microinstruction position (either upper or lower). If column 1 contains a plus (+) sign or any character other than -, $, *, or blank, that statement is assigned to the next available upper microinstruction location. If column 1 contains a minus (-) sign, that statement is assigned to the next available lower microinstruction location. The use of + and - results in possibly skipping an available microinstruction location. The skipped location is assembled as all zeros and not listed on the listing.

LOCATION FIELD

The location field is in columns 2 through 9 of the input coding form. This field may be left blank or it may contain a symbol. If a symbol is included in the field, it must be entered left-justified and follow the definition of a symbol.

A symbol in the location field is assigned the value of the location of the corresponding microinstruction or constant, or the value of the expression starting in columns 17 with a SET or EQU pseudo instruction.

A symbol in the location field of a microinstruction takes on the upper/lower quality of the actual microinstruction location. This quality is used in coding jumps in the C field of a microinstruction.

SYMBOLS

A symbol is a set of characters that identifies a value and its associated qualities. A symbol can be a maximum of eight characters. A symbol must contain a non-numeric character to distinguish it from a constant. A symbol cannot include the following characters:

, = + - * / blank

The process of associating a symbol with a value and qualities is known as symbol definition. This can occur in two ways:

1. A symbol used in the location field (columns 2 through 9) of a symbolic machine instruction or certain pseudo instructions is defined as an address having the current value of the location counter and having an upper or lower quality.
2. A symbol used in the location field of definition pseudo instructions EQU and SET is defined as having the value and quality derived from an expression starting in column 17 of the instruction. The SET pseudo instruction assigns an attribute of redefinability to a symbol. Unless a symbol was previously defined with a SET instruction, a second attempt to define it with a different value produces a duplicate definition fatal error flag.

A symbol may be used in the location field (columns 2 through 9), the S field (columns 35 through 40), or in the C field (columns 41 through 49) of a microinstruction on the input form.

A symbol is undefined when it has never appeared in the location field, or if it is equated to an undefined symbol. The assembler identifies the use of undefined symbols on the assembly listing.

Examples:

HCNYL	Legal
TAG	Legal
1234	Illegal (will be interpreted as a constant)
*12.3	Illegal (contains an asterisk)
XYZ/3P	Illegal (contains a slash)
B=3	Illegal (contains an equal sign)

CONSTANTS

Constants are used to represent numbers and may be used in the S and C fields of the input form. Constants may also be used on the right side of the several pseudo instructions. The MP assembler recognizes three types of numeric constants: decimal, octal and hexadecimal. Decimal constants have no suffix, octal constants have B as a suffix, and hexadecimal constants have X as a suffix. The numeric constant is represented by a string of digits within the number base of the constant and is always a whole integer number. Constants must fit in the field's length except with the VFD pseudo.

DECIMAL CONSTANTS

A decimal constant consists of a string of decimal digits. If the constant is larger than the field width within the microinstruction, the high order bits will be discarded.

Examples:

999	Legal
98A	Illegal (contains an alphabetic character)
12.1	Illegal (contains a decimal point)

OCTAL CONSTANTS

An octal constant consists of a string of octal digits that are suffixed with the letter B.

Examples;

123B	Legal
77B	Legal
019B	Illegal (contains a non-octal digit)

HEXADECIMAL CONSTANTS

A hexadecimal constant consists of a string of hexadecimal digits and is suffixed with the letter X. The hexadecimal digits are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F.

Examples:

77BX	Legal
1GX	Illegal (contains a non-hexadecimal digit)
ABC	Illegal (has no X suffix)
77B	Will be interpreted as an octal 77

PSEUDO INSTRUCTIONS

Pseudo instructions direct the MP assembler to perform specific functions. They do not generate MP instructions. They define assembler control, listing control, data definition, and other operations. Pseudo instructions are defined in Section 3 of this manual.

MNEMONIC INSTRUCTIONS

Mnemonic instructions allow the programmer to use convenient names to specify the binary information to be inserted in each field of MP micro-instruction. The list of mnemonic instructions recognized by the MP assembler for each field of the instruction is given in Table 2-2. Detailed usage of the mnemonic instructions is given in Section 4 of this manual.

Also allowed are user definable opcodes that can be used in place of the assembler opcodes. MASS will try to match the opcodes first from the symbol table and then from the opcode set in MASS.

Whether or not a user-defined opcode is illegal or not is dependent on which field the opcode is to be used. Each field has associated with it a mask to define the bit fields. The masks for the fields are shown as follows:

TABLE 2-2. SUMMARY OF INSTRUCTIONS AND BIT SETTINGS

M	C...1FFF	F	3FFF....	A	.1C..F..	B	..38.FFF	D	...7.F..	T	....E.8F	S	.....F..	C	C...1.FF
R	0.......	-A	00......	F2	.00.....	F2	..00....	NOP	...0....	*L	....0.0.	NOP	.....0..	WPF	....0.20
S	4.......	-A+-B	02......	P	.04.....	N,K	..08...0	P	...1....	U	....2.0.	RD2	.....1..	WSR	....0.21
J	8...0.##	-A+B	04......	I	.08.....	K	..08...4	I	...2....	L	....4.0.	RPT	.....2..	RPF	....0.22
JP	8...1###	ONE	06......	X	.0C.....	N	..08...8	Q	...3....	KZU	....6.0.	RD3	.....3..	RSR	....0.23
K=##	C...0.##	-A.-B	08......	A	.10.....	ZERO	..08...C	F1	...4....	NZU	....8.0.	RD1	.....4..	LWA1	....0.24
N=##	C...1.##	-B	0A......	F	.14.....	BG	..10..##	A	...5....	INTU	....A.0.	L8EA	.....5..	LHA1	....0.25
		-EOR	0C......	F1	.18.....	X	..18....	X	...6....	NU	....C.0.	F2WR	.....6..	LCA1	....0.26
		A+-B	0E......	MEM	.1C.....	Q	..20....	F	...7....	ZL	....E.0.	AP	.....7..	LWA2	....0.28
		-A.B	10......	SM1	.00..7..	F	..28....	IOD	...0.9..	LQL	....2.8.	BP	.....8..	LHA2	....0.29
		EOR	12......	M1	.04..7..	F1	..30....	IOA	...1.9..	K7L	....4.8.	DP	.....9..	LCA2	....0.2A
		B	14......	SM2	.08..7..	MEM	..38....	MMU	...2.9..	OVFL	....6.8.	APDP	.....A..	LWA3	....0.2C
		A+B	16......	M2	.0C..7..	CRTJ	..08.8..	M1	...4.9..	BTU	....8.8#	DPP	.....B..	LHA3	....0.2D
		ZERO	18......			INRD	..10.8..	SM1	...5.9..	ERL	....A.8.	GATEI	.....C..	LCA3	....0.2E
		A.-B	1A......			INRS	..18.8..	M2	...6.9..	COL	....C.8.	HALT	.....D..	CLRK	....0.40
		A.B	1C......			MMU	..20.8..	SM2	...7.9..			RTJ	.....E..	DECK	....0.44
		A	1E......			INTA	..30.8..					CLRNP	.....F..	INCK	....0.45
		A-	20......											CLRN	....0.48
		A-T	22......											DECN	....0.4C
		SUB	28......											INCN	....0.4D
		SUBT	2A......											SETF/#	....0.5#
		SUB-	2C......											CLRF/#	....0.6#
		SUB-T	2E......											RQLXN	....0.70
		ADD	30......											RQR0E	....0.72
		ADDT	32......											RQR1E	....0.73
		ADD+	34......											RL0E	....0.74
		ADD+T	36......											RL1E	....0.75
		A+	38......											RR0E	....0.76
		A+T	3A......											RR1E	....0.77
		AL0E	3C80....											TMA/#	....1.0#
		AL1E	3C88....											TMAK/#	....1.1#
		ALEA	3C90....											GITMAK/#	....1.2#
		AQL0E	3CC0....											TK/#	....1.3#
		AQLEA	3CD0....											TN/#	....1.4#
		AR0E	3D00....											TR/#	....1.5#
		ARSE	3D08....											/K	....1.60
		AREA	3D10....											J/K	....1.61
		AQR0E	3D40....											L/K	....1.62
		AQRSE	3D48....											JLJ/K	....1.63
		AQREA	3D50....											/J	....1.64
		SL0E	3E80....											J/J	....1.65
		SL1E	3E88....											L/J	....1.66
		SLEA	3E90....											JLJ/J	....1.67
		SDL0E	3EC0....											GATEIXT	....1.68
		SDLFA	3ED0....											K=##	C...0.##
														N=##	C...1.##

FIELD	MASK
M	C0001FFF
F	3FFF0000
A	01C00F00
B	00380FFF
D	00070F00
T	0000E08F
S	00000F00
C	C00010FF

The user-defined opcode is usually established with an EQU and the bits must fit within the mask or an error will result.

```
Example:  ROTE   EQU   11110000X -- legal only in the F Field
```

PSEUDO INSTRUCTIONS 3

Pseudo instructions are instructions to the MP assembler and normally do not result in any microcode output (the only exceptions are the HEX, DEC, OCT and VFD pseudo instructions). A pseudo instruction consists of the pseudo operation code, which is coded in the F field of the input form (columns 11 through 16), plus additional information coded in the other fields of the input form. The detailed field usage is given under each pseudo instruction.

ASSEMBLER CONTROL

These pseudo instructions define and control the operation of the MP assembler, but do not generate code in the object program.

IDENT

This pseudo instruction provides program identification and must be used as the first instruction of each program. Text in columns 17 through 71 of the IDENT card is printed as the first line at the top of each page of the output listing. The location field is ignored.

Example:

11	17
IDENT	CDC 844 DISK FILE CONTROLLER EMULATION FOR MP

TITLE

This pseudo instruction provides page heading information and may be used anywhere in the program. The TITLE pseudo causes a page eject and a new heading to be printed at the top of the page. This heading consists of columns 17 through 23 from the IDENT card, followed by one space and the contents of columns 17 through 63 from the TITLE card.

Example:

```
 |11          |17
-+------------+------
 |TITLE       |DISK FILE CONTROLLER EMULATION FOR MP
```

END

The END pseudo instruction signals the end of this program for assembly and must be the last instruction in a program. It causes the assembler to proceed with the 'complete assembly' process. On completion of the assembly process, the assembler is reset and continues reading input information to obtain the next microprogram of a batch to assemble. The total assembly process is completed on detecting a FINIS pseudo instruction.

Example:

```
 |11
-+----
 |END
```

FINIS

The FINIS pseudo instruction signals the completion of a batch of assemblies by the MP assembler and returns control to the host computer operating system. The MP assembler accepts the FINIS pseudo instruction in either columns 10 or 11.

Examples (either of the following):

```
 |10|11
-+--+-----
 |  |FINIS
 |FINIS
```

OPTN

The OPTN pseudo instruction informs MASS of the control and input/output options for this assembly. The OPTN pseudo instruction may appear anywhere in the program. The requested options are separated by commas with the first blank or column 50 terminating the scan. Most of the options once set remain in effect for the complete assembly and, therefore, more than one OPTN pseudo can be specified. Only the first character is used to determine the requested option (L and LIST are equivalent option requests). Any illegal options are flagged as errors and ignored.

The following options may be specified by the OPTN pseudo:

C	The C (CROSS) option requests a cross-reference listing on the list file (OUTPUT).
D	The D (DEAD) option requests a deadstart format file on the object file (PUNCH).
M	The M (MAP) option requests a listing of unused (zero) micro-memory locations on the list file (OUTPUT). This option is available only if either deadstart or simulate output is requested.
N	The N (NOSUM) option requests that no checksums of micromemory be generated. If the checksum is generated, the result is an exclusive OR of all of micromemory and is stored in the first (lowest) available unused (zero) micromemory location.
S	The S (SIMULATE) option requests that a binary file of micro-memory, file 1, file 2 and the symbol table be written on the simulation object output file (SIMFIL for input by the MP Simulator (MPSIM).
L,L=xx	The L (LIST) option requests a listing on the list file (OUTPUT). If the L=xx option is used, and the xx is not equal to 61, no further output listing is produced.
I	The I (INPUT) option requests from a file named INPUT.

The default options are L and I. File names in parenthesis are defaults.

<u>Example</u>:

```
          11        17
          OPTN      CROSS,DEAD,LIST,INPUT
```

FILE

The FILE pseudo is used to place constants in file registers 1 or 2. Card column 17 contains either 1 (or blank) or 2 to specify which register file the succeeding constants go into. The FILE pseudo is used to create constants for loading into the file registers and may appear anywhere in the program.

Example:

11	17	
FILE	1	
FILE	2	
FILE		Assumes file 1

ENDF

The ENDF pseudo is used to terminate the assembly of instructions into the register file areas.

Example:

11
ENDF

LISTING CONTROL

The listing output for the MP assembler is controlled by the following pseudo instructions. These pseudo instructions may appear anywhere in the source input between IDENT and END pseudo instructions.

EJECT

The EJECT pseudo instruction causes the listing to eject to the top of the next page and start a new page. The EJECT pseudo instruction card is not printed.

Example:

11
EJECT

SPACE

The SPACE pseudo instruction causes blank lines to be printed. The SPACE pseudo instruction is not printed. The number of blank lines to be listed is defined in the A field of the pseudo instruction. The A field may contain a constant or a predefined symbol.

Example:

2	11	17
NUMBER	EQU	2
	SPACE	6
	SPACE	NUMBER

The first SPACE pseudo instruction would cause six blank lines to be printed. The second would cause two lines to be printed if NUMBER has been defined to be 2 by an EQU or SET pseudo instruction.

BOX

This pseudo instruction is used in conjunction with EBOX to provide emphasis for comments in the listing. This pseudo instruction is not printed; instead, a line of asterisks will be listed. All succeeding cards will have asterisks in columns 1 and 80 to create comment cards. Only an EBOX or END pseudo instruction following a BOX pseudo instruction will be executed. The listing will be spaced one line before printing the first line of asterisks for the BOX command.

NOTE

A BOX command will turn all succeeding micro-instructions to comment cards until EBOX is encountered.

Example:

```
|11      17
+----------
|BOX
```

EBOX

This pseudo instruction causes a line of asterisks to be listed rather than this pseudo instruction. In addition, the automatic assignment of asterisks to columns 1 and 80 started by the BOX pseudo instruction will be terminated. One blank line will be listed after the line of asterisks.

Example:

```
|11      17
+----------
|EBOX
```

NOLIST

The NOLIST pseudo stops further listing until a LIST or END pseudo is encountered, except for cards which cause diagnostics.

Example:

```
|11      17
+----------
|NOLIST
```

LIST

The LIST pseudo starts listing after being stopped by the NOLIST pseudo.

Example:

```
|11      17
+----------
|LIST
```

REM

The REM pseudo is another way to create comments. The entire card is considered a comment card except there is no automatic line space before the REM card, or after unless it immediately follows a comment card.

Example:

1	11	17
THIS	REM	IS A COMMENT

MEMORY MANAGEMENT AND SYMBOL DEFINITION

These pseudo instructions define symbols and provide for controlling the allocation of micromemory for the object code output.

The EQU, SET, VFD, continuals and ORG pseudo instructions require an address expression that begins in card column 17 and may continue through column 50. The expressions are made up of operands separated by operators. The operands may be constants, asterisks or previously defined symbols. The operators are +, -, * and / (add, subtract, multiply and divide). Expressions are evaluated from left to right; there is no hierarchy of operators. Parentheses are not allowed for grouping within an expression. The expression terminates on the first blank character. The results of the expression and any intermediate computations are truncated integers within the range of the host processor; for each EQU and SET, the range is 0 to 2^{20} (19 bits).

Example:

A-B/C*D+E	Legal (if A=8, B=3, C=4, D=6, E=3, value is 9)
F	Legal
*	Legal (current location counter)
**5	Legal (current location counter times 5)
*+5	Legal (current location counter + 5)

All micromemory and file register definitions are defined as 2*value. In an expression, the micromemory/file register definitions are adjusted during multiplication and division.

Note that symbols for file and memory locations are stored in the symbol table at twice the memory location, if upper of the microinstruction pair, and twice the memory location, plus 1, if lower of the microinstruction pair. Symbols for file locations are always twice the file location. This implies that if the label "STUFF" is associated with micromemory location 00C upper, then the instruction LABEL EQU STUFF+5 would generate a hexadecimal 11 for LABEL. If the EQU was changed to ORG, then LABEL would be associated with 011 upper (not 00E lower).

Any symbol appearing as an operand in the expression of an EQU, SET, ORG instruction must have appeared and been defined in the location field prior to its use in the expression.

EQU

The EQU pseudo instruction assigns a value corresponding to the expression beginning in column 17 to the symbol appearing in the location field (columns 2 through 9). The symbol in the location field of a micromemory location or file takes on an upper or lower quality matching that of the expression in the A field. The use of the qualifier field in column 1 of an EQU card has no effect on the quality of the symbol defined by the equate operation.

Example:

2	11	17	
VALUE	EQU	4	VALUE = 4
LABEL	EQU	VALUE*VALUE/2	LABEL = 8
LABEL.1	EQU	LABEL+1*2	LABEL.1 = 18
P.43X	EQU	VALUE/3	P.43X=1
LOCATE	EQU	*	LOCATE = current location counter

SET

The SET pseudo instruction assigns a value corresponding to the expression beginning in column 17 to the symbol appearing in the location field. The symbol in the location field takes on an upper or lower quality matching that of the expression in the A field. The use of qualifier field (column 1) has no effect on the upper or lower quality of the symbol. Unlike the EQU pseudo, a symbol defined by a SET pseudo may be redefined by a later SET. A symbol defined by SET may not appear in the location field of an EQU pseudo.

Examples:

2	11	17	
VALUE	SET	4	VALUE = 4
LABEL	SET	VALUE*VALUE/2	LABEL = 8
LABEL	SET	LABEL+1*2	LABEL = 18
P.43X	SET	VALUE/3	P.43X = 1
LOCATE	SET	*	LOCATE = current location counter
LOCATE	SET	*+5	LOCATE = current location counter +5

ORG

The ORG pseudo instruction is used to assign a starting value to the micromemory location counter. The micromemory location counter provides for automatic allocation of microinstructions to successive upper and lower locations, unless the allocation is changed by the coding of a plus sign or minus sign in column 1 of the microinstruction input card. When ORG is encountered, all instructions and data following the ORG pseudo instruction are assembled in consecutive upper and lower micromemory locations, starting with the upper location of the address specified by the expression beginning in column 17. The ORG may be used as many times as desired. If use of the ORG pseudo instruction causes some instruction to be assembled into a non-zero location (i.e., assembly over an already assembled location), an error is flagged and the number of the card that previously caused the location to be assembled is printed for cross-reference. The most recent instruction does, however, overlay the previously assembled instruction.

Examples:

11	17	
ORG	100+ABC	Set program location counter to upper of 150 decimal if ABC = 50.
ORG	FF3X	Set program location counter to upper of $FF3_{16}$.
ORG	TAG	Set program location counter to the value TAG (provided TAG is defined).

DATA DEFINITION

Three data definition pseudo instructions are provided so the programmer can define 32-bit constants to be inserted in the micromemory at the current location specified by the micromemory location counter. The pseudo commands are DEC, OCT and HEX for decimal, octal and hexadecimal constant generation. The pseudo commands are coded in the F field of the coding form, and a string of digits in the number base is included in columns 17 through 28. Comments may start in any column after 29. The string of digits may include a minus sign (-). Embedded blanks are ignored. The string of digits is converted in its number base to a 32-bit binary number. The result is complemented (one's complement) if a minus sign exists anywhere in the string. A symbol may be placed in the location field for referencing the constant, and the qualifier field may have a plus, a minus or a blank to control the micromemory allocation.

An error is indicated if the string of digits contains any digit not in the number base.

OCT

The OCT pseudo instruction causes the string of digits starting in column 17 to be converted from octal representation to binary and stored at the current micromemory location. A symbol in the location field is optional. Occurrence of any character other than 0 through 7 or minus in the string will cause an error to be indicated. The string is terminated by a blank.

Example:

11	17	
OCT	123	Create 00000000123_8 in the current location
OCT	-123	Create 37777777654_8 in the current location

DEC

The DEC pseudo instruction causes the string of decimal digits in columns 17 through 28 to be converted from decimal representation to binary and stored as a 32-bit number in the current micromemory location. A symbol in the location field is optional. Occurrence of any character, other than 0 through 9, or minus in the string, will cause an error to be indicated. The string is terminated by the first blank.

Example:

11	17	
DEC	10	Create 0000000A (hex) in the current location.
DEC	-10	Create FFFFFFF5 (hex) in the current location.

HEX

The HEX pseudo instruction causes the string of hexadecimal digits in columns 17 through 28 to be converted from hexadecimal to binary representation and stored as a 32-bit number in the current micromemory location. A symbol in the location field is optional. Occurrence of any character other than 0 through 9, A through F or minus in the string will cause an error to be indicated. The string is terminated by the first blank.

Example:

11	17	
HEX	DEAD	Create 0000DEAD (hex) in the current location.
HEX	DEAD-	Create FFFF2152 (hex) in the current location.

VFD

The VFD pseudo instruction is used to create specific bit patterns within a micromemory word.

The VFD pseudo generates one word (32-bits) from one or more specifications in columns 17 through 50. The format of the instruction is:

VFD $s_1,m_1,s_2,m_2, \ldots s_n,m_n$

where s_i is the number of bits of the value of m_i to be included in the word, and

m_i is an expression.

The sum of the s_i's must equal 32. If the number of bits necessary to contain the value of an expression is less than the corresponding size s, the value is right-justified with zero fill. If the number of bits is larger than the size s, bits are truncated from the left.

Like the EQU, SET, ORG pseudo instructions, any symbol appearing as an operand in an expression MUST have been defined in the location field prior to its use in the instruction.

Example:

Result	2	11	17
	A	EQU	10
		ORG	8
E0A0A803	X	VFD	4,-2,8,A,2,X,6,X+2,1,1,11,*-5
00000008		VFD	32,X
FFFFFFFB		VFD	32,-5
00007000		VFD	16,0,4,7,12,0

PROGRAMMING INFORMATION

The programming information pseudo instructions provide the programmer with additional information in the output listing.

The MASS analyzes each microinsruction for its execution time in the variable cycle length of the MP. This timing information is printed immediately preceding the first column of the card listing on the assembler printout. This timing is represented in a cycle count of the 56 nanosecond basic cycle time of the MP. The following pseudo instructions allow the programmer to notify the assembler of the mode of operation for timing purposes.

If no timing pseudo instructions are used, the assembler assumes two's complement operation. In addition, some instructions take a different amount of time to execute, depending on whether they are executed on a 16- or 32-bit machine. See the Basic System Controller Model 65109 Hardware Reference Manual for these differences.

CMP1

The CMP1 pseudo instruction causes the timing information following the pseudo instruction to be listed for each instruction as if the MP were operating in the one's complement mode. CMP1 may be used anywhere and as often as desired in a program.

CMP2

The CMP2 pseudo instruction causes the timing information following the pseudo instruction to be listed for each instruction as if the MP were operating in the two's complement mode. CMP2 may be used anywhere and as often as desired in a program.

CONDITIONALS

Instructions in a program can be skipped or assembled according to the value of a symbol or expression at assembly time by the use of conditionals. In the following conditional pseudos, m is the symbol or expression to be evaluated; any symbols used must be previously defined. The tag is a symbol and is optional; if used, it is separated from m by a comma.

All instructions following the conditional pseudo instruction are assembled when the test condition is true and skipped when the condition is false. Skipping is terminated either by the matching ENDIF or an END pseudo. One blank line is output on the listing before each conditional.

ENDIF

This pseudo is used to terminate the conditional skipping of code to be assembled; otherwise, it has no effect. If there is no tag in the location field, any skipping in effect is terminated. If a tag appears in the location field, only the conditional(s) which specify that tag will be terminated. One blank line is skipped after an ENDIF.

Format:

2	11	17
tag	ENDIF	

IZD

IZD causes the coding lines following the IZD pseudo to be assembled only if the value of m is zero.

Format:

2	11	17
	IZD	m,tag

IND

IND causes the coding following the IND to be assembled only if the value of m is not zero.

Format:

2	11	17
	IND	m,tag

IGD

IGD causes the coding following the IGD to be assembled only if the value of m is greater than or equal to zero.

Format:

2	11	17
	IGD	m,tag

ILD

ILD causes the coding following the ILD to be assembled only if the value of m is less than zero.

Format:

2	11	17
	ILD	m,tag

Example of Conditional:

2	11	17
XYZ	EQU	1
JKL	EQU	0
PQR	EQU	-1
	IZD	XYZ,DWD
	•	
	•	Code Not Assembled
	•	
DWD	ENDIF	
	IGD	XYZ,ART
	•	
	•	Code Assembled
	IND	XYZ,ART
	•	
	•	Code Assembled
	ILD	JKL-XYZ,ART
	•	
	•	Code Not Assembled
ART	ENDIF	Terminates the IGD, IND and ILD
BUFF	EQU	*
	ORG	BUFF+X
	IGD	*-BUFF-30
	•	
	•	Code not assembled if value
	•	of X is greater than 29.
	ENDIF	

MNEMONIC MICRO INSTRUCTIONS 4

All instructions of the MP may be coded in the MASS language using mnemonic codes and symbolic programming techniques. This section describes how machine language microinstructions are expressed in MASS and the relevant requirements and limitations.

MP MACHINE ORGANIZATION

Figure 4-1. gives the organization of the registers and the ALU in the MP, including transforms.

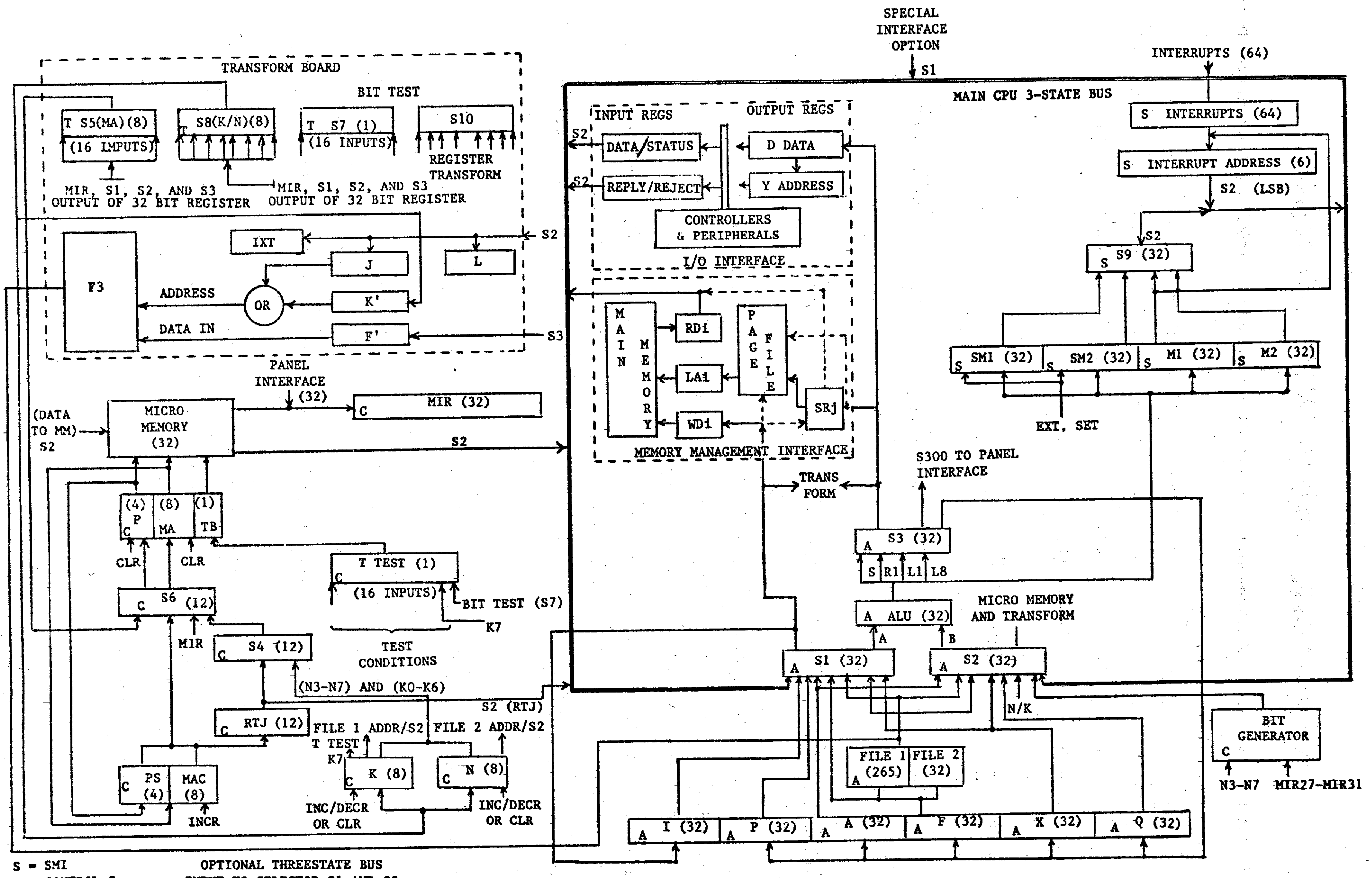
MICROINSTRUCTION FORMATS

Figure 4-2. shows the format of the microinstructions for the MP. Each microinstruction is 32 bits in length. The micromemory is organized into 64-bit micromemory locations; each location contains two microinstructions, an upper microinstruction and a lower microinstruction. The determination of which microinstruction of a pair to execute is specified by the T (test) field of the previously executed microinstruction.

Each microinstruction consists of five major divisions:

- M field, which specifies the mode of selecting the next microinstruction pair and also selects different formats for interpretation of the S and C fields of the microinstruction.

- ALU control, which consists of the F, A, B and D fields; these fields specify two sources, a destination and an ALU operation to be performed. ALU control may be included in all microinstructions.



S = SMI
C = CONTROL 2
A = ALU
T = TRANSFORM

* File 1 is furnished as an option.

Figure 4-1. Central Processing Unit Detailed Block Diagram

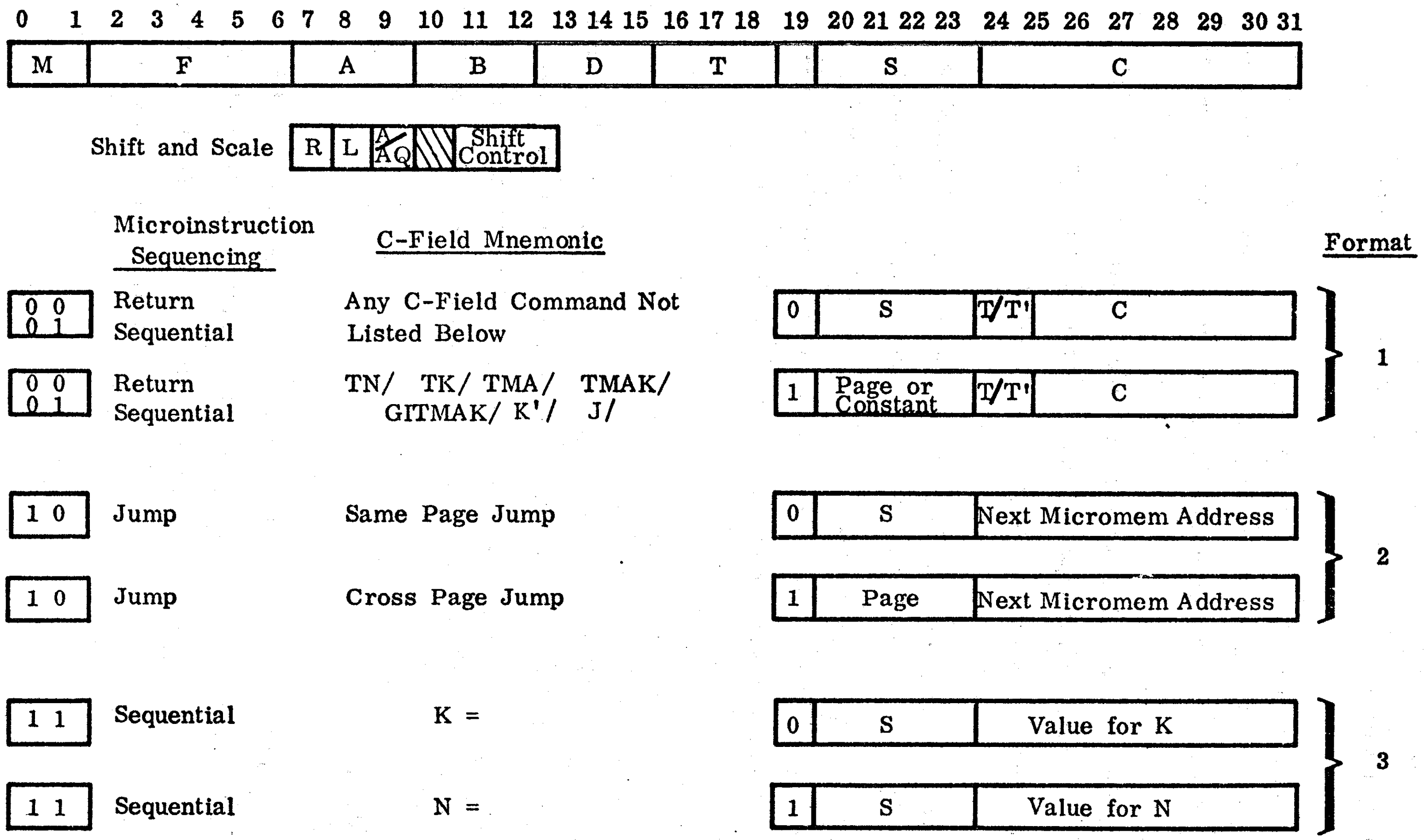


Fig. 4-2. Microinstruction Formats

- T field, which specifies a condition to be tested in selecting the upper or lower of the next microinstruction pair for execution. The T field may also specify an unconditional selection.
- S field, which specifies a special operation to be executed in parallel with ALU control, a page for a microinstruction jump or transform, or extension of the A, B and/or D field coding.
- C field, which specifies additional operations to take place in parallel with other operations, a jump address, or a constant for transfer to the N or K register.

The six microinstruction formats in Figure 4-2. are selected based on the information coded in the M and C fields of the input form. In all cases, the ALU control portion of the microinstruction may be coded regardless of the format selected. Basic test conditions may be coded in all formats; extended tests may only be coded in format 1.

F FIELD

The F field specifies either ALU operations on the output of selector 1 and selector 2 (the A field and the B field respectively), or shift/scale operations in the A or AQ registers. In the case of shift and scale operations, the A and B fields of the microinstruction must not be specified.

ALU OPERATIONS

The ALU operations are either logical or arithmetic, and combine two source inputs. The result is routed to a single destination. The two inputs are called the A source and the B source. The A source is referred to as the A input or selector 1 (S1) and the B source as the B input or selector 2 (S2).

LOGICAL OPERATIONS

The logical operations perform bit-by-bit combinations of the A input and B input for delivery to the destination. An example of the use of the logical operations is shown in Figure 4-3.

Assume:	0 0 1 1	A input bit values
	0 1 0 1	B input bit values

F Code	Bit Results of the Operation	Operation
ZERO	0 0 0 0	Zeros output
A.B	0 0 0 1	Logical product of A and B
A.-B	0 0 1 0	Logical product of A and not B
A	0 0 1 1	Transfer A
-A.-B	0 1 0 0	Logical product of not A and B
B	0 1 0 1	Transfer B
EOR	0 1 1 0	EXCLUSIVE OR of A and B
A+B	0 1 1 1	INCLUSIVE OR of A and B
-A.-B	1 0 0 0	Logical product of not A and not B
-EOR	1 0 0 1	Negation of EXCLUSIVE OR of A and B
-B	1 0 1 0	Transfer one's complement of B
A+-B	1 0 1 1	INCLUSIVE OR of A and not B
-A	1 1 0 0	Transfer one's complement of A
-A+B	1 1 0 1	INCLUSIVE OR of not A and B
-A+-B	1 1 1 0	INCLUSIVE OR of not A and not B
ONE	1 1 1 1	One's output

ARITHMETIC OPERATIONS

The arithmetic operations of addition and subtraction are specified in this set of F codes. Two options are provided. The first option provides an arithmetic overflow test if the letter T is included in the F code. This sets the overflow capture bit on the transform logic if an add or subtract operation generates an arithmetic overflow. The overflow condition exists if the signs of the A source and B source are the same and the sign of the result is different (addition), or if the signs of the A and B sources differ and the sign of the result is the same as the B source (subtraction). An example of an overflow condition is shown in Figure 4-4.

The second option provides for a forced carry into the ALU logic unit, coded as a +. This option is used to emulate multiple-precision arithmetic.

The arithmetic operations are performed in one's or two's complement arithmetic and can operate on single-precision operands using the main ALU. Selection of one's complement or two's complement mode (SM101)* is explained in the Basic System Controller Model 65109 Reference Manual.

The arithmetic codes are as follows:

F Code	Operation
SUB	Subtract B input from A input.
ADD	Add A and B inputs.
SUBT	SUB with overflow test.
ADDT	ADD with overflow test.
SUB-	SUB with forced carry-in (generally the same as a SUB).

*SM100 means status/mode register 1, bit position 00. Bit positions are numbered from left to right, with 00 as the left-most bit.

```
CYBER/MP-60 MICRO ASSEMBLER PSD VERSION 1.0 MASS    LOGICAL OPERATIONS EXAMPLE

HEX   HEX       CYCLE   LABEL   F      A      B      D      S      C         MT    COMMENTS
MML   IMAGE     COUNT

                        ***********************************************************************
                        *                                                                     *
                        *         EXAMPLES OF LOGICAL OPERATIONS                              *
                        *                                                                     *
                        ***********************************************************************

019 L 5326 2000   3             EOR    A      Q      X                               X = (Q) EOR (A)
01A U 40C6 0000   3             -A     X             X                               COMPLEMENT X
    L 4A23 2000   3             -B            Q      Q                               COMPLEMENT Q

HEX   HEX       CYCLE   LABEL   F      A      B      D      S      C         MT    COMMENTS
MML   IMAGE     COUNT
```

Figure 4-3. Example of Logical Operations

```
CYBER/MP-60 MICRO ASSEMBLER PSD VERSION 1.0 MASS    EXAMPLES FOR MASS REFERENCE MANUAL

HEX   HEX       CYCLE   LABEL   F      A      B      D      S      C         MT    COMMENTS
MML   IMAGE     COUNT


                        ***********************************************************************
                        *         OVERFLOW AND TIMING EXAMPLES                                *
                        ***********************************************************************

000 U 7265 0000   4             ADDT   P      Q      A                               OVERFLOW SAVED IN S/M
                                CMP1
000 L 7265 2000   5             ADDT   P      Q      A
                                CMP2

                        *       NOTE THAT THE OVERFLOW BIT STAYS SET IN S/M UNTIL EXPLICATLY CLEARED

HEX   HEX       CYCLE   LABEL   F      A      B      D      S      C         MT    COMMENTS
MML   IMAGE     COUNT
```

Figure 4-4. Example of an Overflow Operation and Timing

F Code	Operation
ADD+	ADD with forced carry-in (A+B+1 for two's complement; A+B+1 for one's complement if both numbers are positive; otherwise, A+B for one's complement if one number is negative).
SUB-T	SUB- with overflow test.
ADD+T	Add+ with overflow test.
A-	Subtract 1 from A input.
A-T	Subtract 1 from A with overflow test.
A+	Add 1 to A.
A+T	Add 1 to A with overflow test.

SHIFT OPERATIONS

Shift operations shift the A or the AQ register left (L) or right (R) the number of bit positions specified by the contents of the N register. Additional shift modifier postfixes are as follows:

- 0E Enter zero (right or left) for each shift cycle.
- 1E Enter one (right or left) for each shift cycle.
- SE Extend initial value of bit 0 for each shift cycle.
- EA Perform end-around shift.

The number of bits shifted is determined by the count in N at the start of the shift instruction. If the N register is zero, no shift occurs. The N register can be set in the (same) instruction by placing N=value in the C field; the value set affects the following instruction. The shift operations are various combinations of shift A or A/Q, left or right, end-around or end-off, sign-extended or not sign-extended and entry of a 0 to 1 in the vacated bit position. The A, B and D fields must be left blank.

An example of the shift operation is shown in Figure 4-5.

The shift operation is coded as the F field mnemonic. The D field of the microinstruction normally should be left blank to generate a NOP because the shift is performed in the A or AQ register without use of the ALU. The shift options are coded in the A and B fields of the microinstruction, and these fields of the coding form must be left blank.

The legal shift mnemonics using the above codes are as follows:

F Code	Operation
AROE	Shift A right with zero entry.
ARSE	Shift A right with sign extended.
AREA	Shift A right, end-around.

```
                                                                                         08.18.14. 05/24/76  PAGE
CYBER/MP-60 MICRO ASSEMBLER PSD VERSION 1.0 MASS    SHIFT EXAMPLES
                                                                                                 DIAGNOSTICS
HEX  HEX       CYCLE  LABEL   F    A    B    D    S    C       MT   COMMENTS
MML  IMAGE     COUNT

                     ************************************************************
                     *        SHIFT EXAMPLE, ASSUME (N) IS  6 INITIALLY         *
                     *        THEN THE FOLLOWING INSTRUCTION LEFT SHIFTS        *
                     *        AQ  END AROUND 6 PLACES.  N=FF AFTER              *
                     *        INSTRUCTION COMPLETION.                           *
                     ************************************************************

001 U 7C00 0000 5+N           AQLEA

                     ************************************************************
                     *        SHIFT MOST SIGNIFICANT 8-BIT CHARACTER INTO Q AFTER *
                     *        CLEARING  Q.                                      *
                     ************************************************************

001 L D803 3008  3            ZERO            Q          N=8
002 U 7C00 0000 5+N           AQLEA

                     ************************************************************
                     *        PLACE MOST SIGNIFICANT BIT OF A INTO LSB OF Q AND *
                     *        MAKE ALL OTHER BITS ZERO.                         *
                     ************************************************************

                                                         N=63
002 L D800 303F  3
003 U 7040 0000 5+N           AQROE
                                                                                                 DIAGNOSTICS
HEX  HEX       CYCLE  LABEL   F    A    B    D    S    C       MT   COMMENTS
MML  IMAGE     COUNT
```

Figure 4-5. Examples of Shift Operations

F Code	Operation
AL0E	Shift A left with 0 entry.
AL1E	Shift A left with 1 entry.
ALEA	Shift A left, end-around.
AQR0E	Shift AQ right with 0 entry.
AQRSE	Shift AQ right with sign extended.
AQREA	Shift AQ right, end-around.
AQL0E	Shift AQ left with 0 entry.
AQLEA	Shift AQ left, end-around.

SCALE OPERATIONS

Scale operations are similar to shift operations but are conditioned by the two bits at the scale point being different. Scale examines N. If N = 0, the scale operation is terminated. If N ≠ 0, then the contents of A register bits 0 and 1 are examined. If the bits are different, the scale is stopped. Otherwise, the scale operation then takes place as a left shift of A or AQ and continues until either the bits at the scale point are different or until the contents of N are reduced to 0. At completion, the difference of the initial value of N and the final value of N is the number of bits that A or AQ was shifted. The A, B and D fields of the coding form should be left blank so the assembler can insert the correct values.

The examples depicted in Figure 4-6 assume that a number is positioned in the A/Q registers and must be scaled. In the one's complement example, an end-around scale is used to provide for the propagation of the correct value for the least significant bits. In the two's complement example, a zero entry scale is used.

```
HEX   HEX      CYCLE   LABEL    F     A     B     D     S     C        MT   COMMENTS
MML   IMAGE    COUNT

                      ********************************************************************************
                      *         SCALE EXAMPLE ONES COMPLEMENT ARITHMETIC                             *
                      ********************************************************************************

003 L D800 3040  3                                            N=64          SET MAXIMUM SHIFT
004 U 7ED0 0000 5+N             SDLEA                                       NUMBER OF SHIFTS = N - 64

                      ********************************************************************************
                      *         SCALE EXAMPLE TWOS COMPLEMENT ARITHMETIC                             *
                      ********************************************************************************

004 L D800 3040  3                                            N=64          SET MAXIMUM SHIFT
005 U 7EC0 0000 5+N             SDLOE

HEX   HEX      CYCLE   LABEL    F     A     B     D     S     C        MT   COMMENTS
MML   IMAGE    COUNT
```

Figure 4-6. Examples of Scale Operaticns

Scale operations are as follows:

F Code	Operation
SL0E	Scale A register left with 0 entry.
SL1E	Scale A register left with 1 entry.
SLEA	Scale A register left, end-around.
SDL0E	Scale AQ registers left with 0 entry.
SDLEA	Scale AQ registers left, end-around.

A, B, AND D FIELDS

The A, B and D fields of the microinstruction specify sources and destinations of information in the MP organization. In all three fields, there are basic mnemonics and extended mnemonics. In the object language output, the A, B and D fields occupy three bits each and thus allow only for specifying one of eight different sources and destinations. Since more than eight sources and destinations may be specified in each field, the S field is used also to provide alternate coding interpretation for the 3-bit number in the A, B and D fields. Alternate codes are referred to as A', B', D', D" and DD". The MP assembler accepts any of the specified alternate mnemonics for the fields and provides an automatic S field setting in the object code output. The result of the use of the S field to specify operations, as well as alternate decodings of the A, B and D fields, leads to a possible conflict of mnemonics. The resolution of this conflict is described in Appendix D of this manual. The assembler also provides diagnostic messages if any conflict occurs in the source program.

A FIELD

The A field specifies a source of information to the output of selector 1 and is available for the A input of the ALU.

A Code	Operation
F2	Initial value of N at start of microinstruction specifies address in File 2.*
P	Use contents of P register as A source.
I	Use contents of I register as A source.
X	Use contents of X register as A source.
A	Use contents of A register as A source.
F	Use contents of F register as A source.
F1	Initial value of K at start of microinstruction specifies address in File 1.*
MEM	Contents of main memory are transferred. Data must have been read in a previous microinstruction.

The following A codes (A' codes) are extended mnemonics; the S field must be left blank.

A Code	Operation
SM1	Use contents of SM register 1 as A source.
SM2	Use contents of SM register 2 as A source.
M1	Use contents of interrupt mask register 1 as A source.
M2	Use contents of interrupt mask register 2 as A source.

*RESTRICTION: Value of addressing register (N or K) cannot have been modified by a C field increment or decrement command in the preceding microinstruction. In addition, whichever file is used during current microinstruction, this file cannot have been written in the preceding microinstruction.

B FIELD

The B field specifies a source of information to the output of selector 2 and is available for the B input of the ALU.

B Code	Operation
ZERO	Use all zeros as output of selector.*
N	Use N register as bits 16 through 23 of selector 2 with zeros as rest of selector 2.*
BG	Use single bit set to 1 with rest 0's. The bit to set is controlled by either value of lower five bits of C field of this microinstruction or contents of N register as elected by setting of status/mode bits.
K	Use K register as lower 8 bits of selector 2 with zeros as rest of selector 2.*
N,K	Use N and K registers as lower 16 bits of selector 2 with zeros as rest of selector 2. K fills bits 24-31 and N fills bits 16-23.*
X	Use contents of X register as B source.
Q	Use contents of Q register as B source.
F	Use contents of F register as B source.
F1	Use contents of a word in File 1 as B source input. Value of K at start of microinstruction specifies address in File 1.**
F2	Use contents of a word in File 2 as B source input. Value of N at start of microinstruction specifies address in File 2.**

*Selection of this B code option is controlled by setting bits 28 and 29 of the microinstruction in the C field. Other C field commands may be given if they match in bits 28 and 29. The assembler allows this operation.

**RESTRICTION: In the preceding microinstruction, the addressing register (N or K) cannot have been incremented or decremented and whichever file is read (F1 or F2), cannot have been written.

B Code	Operation
MEM	Contents of main memory is used as B source. Data must have been read in a previous microinstruction. Restriction: If extended A code is used, data from A source is also used as B source, suppressing memory data.

The following B codes (B' codes) are extended mnemonics and the S field must be left blank. If these extended mnemonics are used, it is not possible to use extended mnemonics in the A or D field.

B Code	Operation
CRTJ	Read the complement of the RTJ register in the lower 12 bits of selector 2.
INRD	Read data/status through I/O TTY card from peripheral devices.
INRS	Read responses from I/O TTY card.
MMU	Transfer data from micromemory to X register as a destination. Data is passed through ALU, and F field must make reference to B source. D field, if specified, will also contain the data from micromemory.* Next microinstruction is always next sequential upper instruction.
INTA	Read complement of address of current highest priority active interrupt having its corresponding mask bit set to output of selector 2. Address consists of bit number of mask bit that controls interrupt. An INTU test command must have been given in the preceding microinstruction.

*The address of micromemory is determined by the transform number specified in the C field or NK, and the upper or lower location is specified by the test field of this instruction.

D FIELD

The D field specifies the destination of information from the main organization of the MP. There are four sources of information for delivery to the specified destination. These are:

- The optionally shifted output of the ALU. This shifting occurs in a shifting network (selector 3) that provides the shift on the output of the ALU.
- The direct (unshiftable) output of the ALU.
- The output of selector 1 (input to the selector is specified by the A field).
- The output of selector 2 (input to the selector is specified by the B field).

D Code	Operation
NOP	Do not transfer any information to any destination.
P	Transfer output of ALU as shifted by selector 3 to P register.
I	Transfer output of selector 1 to I register.
Q	Transfer output of ALU as shifted by selector 3 to Q register.
F1	Transfer output of ALU as shifted by selector 3 to F register, and then write in file 1 during first part of next microinstruction. Location in file 1 to receive information is specified by contents of K register on completion of this microinstruction. Note that if next instruction specifies F1 in A or B field, an erroneous result will be received; the information is available in the F register and should be obtained from there.

D Code	Operation
A	Transfer output of ALU as shifted by selector 3 to A register.
X	Transfer output of ALU as shifted by selector 3 to X register.
F	Transfer output of ALU as shifted by selector 3 to F register.

The following extended D (D' codes) may be specified in the same microinstruction as an extended A code. The assembler provides the proper S field setting.

D Code	Operation
MMU	Transfer output of selector 2 to 32-bit micromemory word. Micromemory address is specified by transform corresponding to number contained in C field of this microinstruction. Upper or lower micromemory word is specified by T field of this microinstruction. Next microinstruction is taken from next sequential upper location.
M1	Transfer output of ALU to M1 register.*
M2	Transfer output of ALU to M2 register.*
SM1	Transfer output of ALU to SM1 register*
SM2	Transfer output of ALU to SM2 register.*
IOD	Transfer output of selector 3 (S3) to I/O data register on TTY card.
IOA	Transfer output of selector 3 (S3) via the I/O data register to I/O address register. Destroys contents of I/O data register.

*Outputs to the mask and status/mode registers are not shiftable.

EXAMPLES OF F, A, B, AND D FIELDS

The assembler output listing shown in Figure 4-7 demonstrates the basic use of the fields discussed above. In some cases, the S and C fields will also be used to demonstrate common programming errors that are detected by the assembler.

S FIELD

The special field (S field) is coded in columns 35 through 40 of the assembly coding form. If this field is not used by the assembler to specify alternate translations for the A, B or D fields, it may be used by the programmer to specify a special instruction or it may contain a constant or a programmer defined symbol. The S field mnemonics specifying alternate A, B or D field codings are not normally used by the programmer and are automatically generated as required by the assembler.

These S codes specify operations taking place in parallel with the F and C codes.

S Code	Operation
NOP	No operation specified in S field.
RPT	If contents of N register are not equal to zero, decrement N by one and take current microinstruction pair, using upper/lower decision implied by T field. If contents of N register equals zero, decrement N by one and use microinstruction sequencing specified by M field and T field.
L8EA	Selector 3 is set to provide shift of data left eight bits, end around. This provides shift of ALU output to P, A, F, X and Q registers if used as destinations.

```
HEX   HEX        CYCLE  LABEL   F     A     B     D     S     C          MT   COMMENTS                                DIAGNOSTICS
MML   IMAGE      COUNT

                        *****************************************************************************************
                        *       F, A, B, D  FIELD EXAMPLES                                                      *
                        *****************************************************************************************

406 U 7125 0000  5              ADD   A     Q     A                               A= (A)+(Q)

406 L DE86 2703  4              A     SM2         X           K=3                 X= (S/M REG 2)
                                REM                                               ALSO SET K=3

407 U F1B4 0006  5              ADD   F1    F1    F1          K=6                 F1(6) = 2*F1(3)
    L 7051 201F  5              ADD   P     BG    P           31                  P=(P)+1  32 BIT MP
408 U 7449 000C  5              ADD+  P     ZERO  P                               P=(P)+1  USES C FIELD
    L 5441 2040  3              A+    P           P           CLRK                P=(P)+1  C FIELD OPEN
409 U 4702 0000  3              ONE   A           I                               I=(A)  F HAS NOEFFECT

409 L 5426 2000  3              B           Q     X                               X=(Q)
40A U 5CD6 0003  3              A.B   X     BG    X           3                   X = BIT 3 OF X
    L 5615 2A01  4              A+B   SM1   BG    SM1         1                   SET BIT 1 OF SM1
40B U 5A97 0A04  4              A.-B  SM2   BG    SM2         4                   CLEAR BIT 4 OF SM2

                        *       FOLLOWING CODE IN ERROR

40B L 700D 2700  6              ADD   SM1   CRTJ  A                               A AND B BOTH REQUIRE S                  SAB
40C U F10D 0008  5              ADD   A     N     A           K=31                C FIELD USED BY B FIELD                 CBC
                                                                                                                          DIAGNOSTICS
HEX   HEX        CYCLE  LABEL   F     A     B     D     S     C          MT   COMMENTS
MML   IMAGE      COUNT
```

Figure 4-7. Examples of the Use of F, A, B, and D Fields

S Code	Operation
F2WR	Write contents of the F register into file 2 at address specified by contents of N register at start of this microinstruction. Must not be specified if preceding microinstruction used F2 as a destination code.
GATEI	Transfer output of selector 1 to I register.
HALT	If SM109 (halt bit) is set to 1, complete this microinstruction and halt operation.
RTJ	Transfer address of next sequential microinstruction pair to RTJ register.
CLRNP	Clear N register and set page register in P-MA register so next microinstruction will be executed from page 0. This is normally used with J or R in M field of microinstruction.
RD1 RD2 RD3	Gate data specified by lookahead register 1, 2 or 3 onto the main bus for use by the next instruction at selector 2. MEM in the A or B field would allow the data to be stored in a register specified in the D field.

The following S codes are not normally used by the microprogrammer because they are set by the assembler to handle extended A, B and D field mnemonics.

- AP (for A' codes)
- BP (for B' codes)
- DP (for D' codes)
- APDP (for A' and D' code combination)
- DPP (for D" codes)

EXAMPLES

The code shown in Figure 4-8 will count the number of 1 bits in the A register. Assuming two's complement arithmetic, the total number of 1-bits will be left in the Q register.

The code at HERE adds X to itself to get a left shift of 1, and the COL in the T field checks for carry out of the high order bit if it is a 1. If there is no carry out, the upper instruction at HERE is repeated. If there is a carry out, the lower instruction is performed, which adds one to the A register and jumps back to HERE. Each execution of the instruction at HERE counts N down by one. When N is counted down to zero, control goes to the next sequential upper instruction, since a two's complement adder is assumed, leaving all zeros in the X register after the 32 additions of X to itself.

C FIELD

The C field is a multipurpose field and its interpretation is dependent on the M field.

JUMP INSTRUCTIONS

If the M field is coded with a J, the C field is taken as a jump address. It may contain a constant, which expresses page and location within the page. In this case, the T field must be coded to specify the upper or lower location to be jumped to. The C field may contain a symbol, which identifies a page or location within the page and an upper or lower property. In this case, the T field may be left blank and the assembler will select the correct upper or lower designation. If a T field value is given, it will override the assembler selection.

The page to be jumped to is examined by the assembler, along with the page of the jump instruction. If the jump is within the same page, format 2 with bit 19 clear (refer to Figure 4-2) is selected, allowing use of the S field for special instructions. Otherwise, format 2 with bit 19 set is selected.

```
HEX   HEX      CYCLE   LABEL    F     A     B     D     S     C        MT    COMMENTS
MML   IMAGE    COUNT

                      *********************************************************************************
                      *         S FIELD REPEAT EXAMPLE                                                *
                      *********************************************************************************

016 L D803 3020  3              ZERO              Q           N=32           CLEAR Q, SET RPT COUNT
017 U 5415 201F  3              B           BG    A           31        U    1 TO A

018 U 70DE C280  5    +HERE     ADD   X     X     X     RPT               COL  TEST MSB OF X
    L B123 2018  4              ADD   A     Q     Q           HERE     J     COUNT  A 1 BIT

019 U 5800 0000  3    +                           NOP                        NEXT INSTRUCTION

HEX   HEX      CYCLE   LABEL    F     A     B     D     S     C        MT    COMMENTS
MML   IMAGE    COUNT
```

Figure 4-8. Example of S Field Coding

FLAG SETTING AND CLEARING COMMANDS

The Status/Mode registers contain bits which are individually associated with various status and operation modes. The individual bits in the Status/Mode register which can be turned on/off are called flag bits.

Flag Number	SM Reg. No.	Bit No.	Function
0	1	00	Double Precision
1	1	01	One's Compliment
2	1	02	Bit Generation from N Register
3	1	03	Adder Split
4	1	16	Lock Memory
5	1	17	MP60 Bit Register
6	1	18	Monitor (Program) Mode
7	1	19	Reserved
8	2	00	Enable Auto Data Transfer
9	2	01	Read I/O Strobe
10	2	02	Write I/O Strobe
11	2	03	Sterm I/O Strobe
12	2	16	Test/XMC MPM (MPM Test Mode)
13	2	17	Isolate BSC (BSC Test Mode)
14	2	18	Reserved
15	2	19	Page Address From S3

The commands are written as follows:

```
        SETF/m              CLRF/m
```

where m is a constant or a symbol. m is converted to binary and the least significant four bits are used to specify the flag to be set (set to 1) or cleared (set to 0).

REGISTER SETTING INSTRUCTIONS

The register setting instructions set the K and the N register and are coded as follows:

- K=value
- N=value

where value may be a constant or a symbol. The constant or symbol is expressed in binary, and the lower eight bits of the binary value are inserted in the C field of the microinstruction as the value to be set into the specified register.

If the S field contains the CLRNP command, the result of a K= or N= in the C field will be zero in the specified register, along with a zero in the N register.

REGISTER CONTROL COMMANDS

The register control commands provide for incrementing, decrementing, and clearing the K or the N register. The commands are as follows:

C Code	Operation
INCK	Increment K register by 1.
INCN	Increment N register by 1.
DECK	Decrement K register by 1.
DECN	Decrement N register by 1.
CLRK	Clear K register.
CLRN	Clear N register.

If the T field specifies a test of the register, the test is made on the initial value of the register before the control command.

SELECTOR 3 (SHIFT COMMANDS)

The following C field commands specify a left or right shift one bit position of data transferred from the ALU through selector 3 destined to P, A, F or the X register. The contents of the Q register may be combined with the destination register to make a double length register, with Q register as lower order bits. These shifts are end off with provision for entry of 0, 1 or the inverse of the sign of the ALU output. The Q register may not be used as a destination register in any of the following commands and obtain correct results. Also, these commands cannot be performed when S field contains L8EA.

First Part	Direction	Entry To Open Bit	
R (shift destination register)	L (left)	0E	Enter zero
RQ (shift destination and Q)	R (right)	1E	Enter one
		XN	Enter complement of ALU sign

The allowable mnemonics for the shift operations are as follows:

C Code	Operation
RQLXN	Shift destination and Q left with complement of ALU sign entered into lowest bit position of Q. This command is used in divide iteration.
RQR1E	Shift destination and Q right and enter 1 in sign position of destination register.
RQR0E	Shift destination and Q right and enter 0 in sign position of destination register.

C Code	Operation
RL0E	Shift destination left and enter 0 in lowest bit position.
RL1E	Shift destination left and enter 1 in lowest bit position.
RR0E	Shift destination right and enter 0 in sign bit position.
RR1E	Shift destination right and enter 1 in sign bit position.

TRANSFORMS

Transforms are used to provide a means of forming micromemory addresses or constants from any pattern of bits from selectors, registers or data paths of the processor. The addresses may be used to sequence micromemory or to set the N or K registers.

The transform instructions in the C field are written as follows:

TMA/j   TK/j   TN/j   TR/j   TMAK/j   GITMAK/j

where j is a symbol or a transform number. The symbol or number is expressed in binary, and the lower four bits of the number are inserted in the microinstruction. The S field may contain a symbol or a constant to specify the page used in the MA transform. For the TN/ and TK/ instructions, the S field must contain a symbol or a constant to avoid an informative diagnostic from MASS.

C Code	Operation
TMA/j	The micromemory transform begins execution at one of the 256 microinstructions (of the current micromemory page) specified by either S2 or the IXT register. The transform number j specifies the register and the bit field as follows:

TRANSFORM NUMBER j	OPERATION
0	Not Defined
1	Micro Addr = Bits 0-5 of S2
2	Micro Addr = Bits 1, 4-10 of IXT
3	Not Defined
4	Not Defined
5	Not Defined
6	Not Defined
7	Micro Addr = Bits 24-31 of S2

TK/j
TN/j
Load the K or N register and the K' register with data from S2, the CPU state register/IXT register, the MIR or the F' register. The following table describes the various fields that are loaded into the K or N register from bits 0-7 of S8. Bits MSB-7 of S8 are used to load the K' register and the K' register is always loaded whether the N or the K register is specified. The following chart summarizes the bits that are loaded into the K and K', or the N and K' register.

The symbols under the transform number denote whether the transform is wirewrapped (Δ), clad (○) or unassigned (OPEN). The number under each of the bit numbers of S8 denotes the register and the bit number which the N or K is loaded. Example: Bit 7 of transform 0 shows S231. This means bit 31 of S2 is loaded into the least significant bit of K, K' or N, K' register.

Bit 11 of Status Mode register 1 is =0 if File 3 is to be addressed and =1 if File 1 is to be addressed.

Generally, transforms 2, 3, 4, 5 and 7 are used to assist in decoding MP-60 type assembly instruction (see MP-60 Reference Manual, Publication No. 14306500A, pages 6-1, 6-4). Transform 2 is used to decode the B field; Transform 3,

XFORM NO	SELECTOR S8 I/O													
	MSB	E	D	C	B	A	0	1	2	3	4	5	6	7
0 ⬡	SMI11	S219	S220	S221	S222	S223	S224	S225	S226	S227	S228	S229	S230	S231
1 △	SMI11	GND	GND	GND	CPU SR27	CPU SR28	CPU SR29	CPU SR30	CPU SR31	**S211**	**S212**	**S213**	**S214**	**S215**
2 △	SMI11	GND	GND	GND	CPU SR27	CPU SR28	CPU SR29	CPU SR30	CPU SR31	IXT16	IXT17	IXT18	IXT19	IXT20
3 △	SMI11	GND	GND	GND	CPU SR27	CPU SR28	CPU SR29	CPU SR30	CPU SR31	IXT06	IXT07	IXT08	IXT09	IXT10
4 △	SMI11	GND	GND	GND	CPU SR27	CPU SR28	CPU SR29	CPU SR30	CPU SR31	GND	IXT11	IXT12	IXT13	IXT14
5 △	SMI11	GND	GND	GND	CPU SR27	CPU SR28	CPU SR29	CPU SR30	CPU SR31	GND	GND	IXT11	IXT12	IXT13
6 ⬡	SMI11	KP00	KP01	KP02	KP03	KP04	MIR24	MIR25	MIR26	MIR27	MIR28	MIR29	MIR30	MIR31
7 △	SMI11	GND	GND	GND	CPU SR27	CPU SR28	CPU SR29	CPU SR30	CPU SR31	IXT27	IXT28	IXT29	IXT30	IXT31

XFORM NO	SELECTOR S8 I/O													
	MSB	E	D	C	B	A	0	1	2	3	4	5	6	7
8 OPEN														
9 OPEN														
A OPEN														
B △	SMI11	GND	GND	GND	GND	GND	GND	GND	SMI(B) • FP26	FP27	FP28	FP29	FP30	FP31
C OPEN														
D OPEN														
E OPEN														
F ⬡	SMI11	S227	S228	S229	S230	S231	GND	GND	GND	GND	GND	GND	GND	GND

K/N TRANSFORMS

the sub-opcode field; Transform 4, the index field for half-word addressing; Transform 5, the index field for character addressing; and Transform 7, the C field decode.

TR/j	The register transform is used to route data directly into File 3 from either S3 or from another register. Presently, only transform 0 is available, which puts bits 0-31 of S3 into File 3.
TMAK/j	Transfer data to the K register as specified by the wiring of the K/N transform j and then perform a TMA/j.
GITMAK/j	Gate the output of selector 1 to the IXT register and then perform a TMAK/j transform.

OTHER C CODES

C Code	Operation
LWAn	Load word address into lookahead register n from page file and S3. If write memory mode is selected, the data in the register in S1 (A field) is used.
LHAn	Load half-word address into lookahead register n.
LCAn	Load character address into lookahead register n.
LBAn	Load bit address into lookahead register n.
WPF	Load the page file (specified by address in S3) with the data word from S1.
RPF	Gate the contents of the page file (specified by address in S3) to the main CPU bus for use in S1 or S2 during the following instruction cycle.
WSR	Load the state register specified by S3 with data from S1.

RSR Gate the contents of the state register (specified by address in S3) onto the main CPU bus for use in S1 or S2 during the following instruction cycle.

EXAMPLE OF C FIELD CODING

If the M field is coded with a J, the C field is used as the address of the micromemory jump destination. The examples in Figure 4-9 show legal and illegal use of the S field in conjunction with the C field.

Examples of multiply and divide codes implemented in an MP which uses one's complement arithmetic are shown in Figures 4-10 and 4-11.

M FIELD

The M field is coded in column 50 of the assembler coding form. The mnemonics in the M field specify the method of selecting the next microinstruction pair (upper and lower) to be executed on completion of the current microinstruction.

If the M field is left blank, MASS assumes that a sequential (S) specification is required.

The code selected in the M field may be overridden in two cases:

- If the T field has the code *L, the lower microinstruction of the current microinstruction pair is selected regardless of the M field code.

- If the C field specifies the transforms TMA/, TMAK/ or GITMAK/, then the transform addressing takes precedence over the M field code.

```
HEX   HEX        CYC.E   LAB3L    F      A      B     D     S      C          MT   COMMENTS                         DIAGNOSTICS
HML   IMAGE      COUNT

                                  IDENT  MASS
000 U 9F06 2000   3               A      A            X     HALT   400X       J    JUMP TO OTHER BANK, S             CRP
                                  REM                                                HOLDS PAGE INFORMATION
                                  A      SM1                       400X       J                                      CRP
000 L 9E00 2700   4                                                                  BOTH SHIFT SAME RESULT
001 U 5800 0573   3                                         L8EA   RQR1E             S FIELD USE INHIBITED             CSC
    L 5800 2E00   5                                         RTJ    TMA/2
                                                                                                                      DIAGNOSTICS
HEX   HEX        CYCLE   LAB3L    F      A      B     D     S      C          MT   COMMENTS
HML   IMAGE      COUNT
```

Figure 4-9. Examples of S and C Field Conflicts

```
HEX  HEX        CYCLE  LABEL   F      A     B     D     S     C        MT     COMMENTS
MML  IMAGE      COUNT

                       ****************************************************************************
                       *       MULTIPLY A BY X, PRODUCT TO AQ.                                    *
                       *       MULTIPLY WORKS ON POSITIVE NUMBERS SO PROVIDE LEAD IN TO           *
                       *       CALCULATE SIGN OF NEGATIVE INPUTS AND CORRECT AT END.              *
                       *       CODE IS WRITTEN FOR ONES COMPLEMENT INPUT NUMBERS.                 *
                       ****************************************************************************

                               CMP1                                             INDICATE ONES COMPLEMENT

030                            ORG    30X

030 U 0F03 C000   4    MULT    A      A           Q           K=0      SNU    CHECK SIGN

031 U 4A23 0045   3    +       -B           Q     Q           INCK            COMP Q FOR POSITIVE
    L 5EC0 C000   4            A      X                                SNU    CHECK SIGN OF X

032 U 40C6 0044   3    +       -A     X           X           CECK            GET POS X AND SIGN IN K
    L D805 301F   3            ZERO               A           N=31            CLEAR A, SET TIMES COUNT
033 U 5800 20F2   3                                           RQR0E    SLQL   MAKE FIRST STEP TEST

034 U 5F05 22F2   3    +       A      A           A     RPT   RQR0E     LQL   MULTIPLY ITERATION LOOP
    L 711D 22F2   5    -       ADD    A     X     A     RPT   RQR0E     LQL   MULTIPLY ITERATION LOOP

035 U 4105 6000   3            -A     A           A                    SKZU   EXIT ON POS SIGN TEST REM

036 U 8105 2059   3    +       -A     A           A           EXIT     J      POS RESULT RECOMP A
    L 8A23 2059   3    -       -B           Q     Q           EXIT     J      NEG RESULT, COMP Q
037                    EXIT    SET    *                                       NEXT INSTRUCTION

HEX  HEX        CYCLE  LABEL   F      A     B     D     S     C        MT     COMMENTS
MML  IMAGE      COUNT
```

Figure 4-10. Example of Multiply Coding

```
HEX   HEX          CYCLE   LABEL    F      A      B      D      S      C         MT      COMMENTS
MML   IMAGE        COUNT
                           ***********************************************************************
                           *           DIVIDE AQ BY CONTENTS OF X. USES ONES COMPLEMENT REP      *
                           *           THIS ROUTINE MAKES OVERFLOW TEST AND SETS BIT IN SM1 IF PRESENT *
                           *           F REGISTER IS USED TO CALCULATE SIGN OF QUOTIENT          *
                           ***********************************************************************

04E U 5F07 C000    4       DIVIDE   A      A             F                       SNU     CHECK SIGN

04F U 4105 2000    3       +        -A     A             A                       SU        IF NEG, COMPLEMENT A
    L 5800 4000    3       -                                    NOP              SL        IF POS, LEAVE ALONE

050 U 4A23 0000    3       +        -B            Q      Q                               WAS NEG, COMPLEMENT Q
    L D2EF 301D    3                EOR    X      F      F             N=29              QUOT SIGN IN F, SET CNTR
051 U 5EC2 C061    4                A      X             I             CLRF/1    SNU     SAVE X IN I, SET 2S COMP
                                    CMP2                                                 INDICATE THOS COMPLEMENT
051 L 40C6 2000    3                -A     X             X                               GET POSITIVE X

                           *        CHECK FOR OVERFLOW

052 U 5F05 0070    3                A      A             A             RQLXN             SHIFT AQ LEFT 1
    L 691D C0F0    5                SUB    A      X      A             RQLXN     SCOL    TEST DIVIDE OVERFLOW

053 U 711D C070    5       +        ADD    A      X      A             RQLXN     SNU
    L 5615 2A0E    4       -        A+B    SM1    BG     SM1           14                SET BIT 14 DIVIDE OVERFLO

                           *        DIVIDE ITERATION LOOP

054 U 711D C270    5       +        ADD    A      X      A      RPT    RQLXN     NU
    L 691D C270    5       -        SUB    A      X      A      RPT    RQLXN     NU

                           *        END CORRECTION A IS 1 LEFT AND MAY BE 1 TO MANY SUBTRACTS

055 U 711D 2000    4       +        ADD    A      X      A                       SU
    L 5F05 4076    3       -        A      A             A             RROE      SL

056 U 711D 0000    4       +        ADD    A      X      A

                           *        CHECK SIGN OF QUOTIENT

056 L 5F40 C051    4                A      F                           SETF/1    SNU     SET ONES COMP
                                    CMP1                                                 INDICATE ONES COMPLEMENT
057 U 4A23 0000    3       +        -B            Q      Q                               COMPLEMENT QUOTIENT

                           *        CHECK SIGN OF REMAINDER AND CORRECT A

057 L 52A8 C000    4       -        EOR    I      F                              SNU     CHECK REMAINDER SIGN

058 U 8105 2037    3       +        -A     A             A             EXIT      J       DONE
    L 9800 2037    3       -                                           EXIT      J       DONE NO CHANGE
059                        EXIT     SET    *
HEX   HEX          CYCLE   LABEL    F      A      B      D      S      C         MT      COMMENTS
MML   IMAGE        COUNT
```

Figure 4-11. Example of Divide Coding

M Code	Operation
S or blank	Sequential addressing. If T field is left blank, MASS will select the next sequential microinstruction either from from current pair or from next microinstruction pair. If the T field is not blank and does not contain a *L (select current lower instruction), the upper instruction of the next sequential microinstruction pair within current page of micromemory will be selected. (Note that location 00_{16} in current page follows location FF.)
J	Jump addressing. Next microinstruction pair will be selected from micromemory location specified by symbol or constant coded in C field of current microinstruction. MASS will select the correct type of jump based on current program address and jump destination. If the jump address is in the current page, a within-page jump will be performed and the S field is available for coding. If the jump address is in another page, the assembler will use the S field to specify the page in a page-jump instruction. If T field is left blank, MASS will select correct upper or lower T value for the symbol.
R	Return addressing. Next microinstruction pair is selected from location in current page specified by contents of RTJ register. The four MSB bits specify the page and and the remaining eight bits specify the address within the page. The RTJ register must have been previously set up by an RTJ in the S field. If the current S field contains CLRNP, the next microinstruction pair is selected from page 0 as specified by contents of RTJ register. T field must be filled in with an upper/lower selection.

T FIELD

The mnemonics in the T field specify the selection of the upper or the lower microinstruction from the next microinstruction pair for execution on completion of the current microinstruction, except *L. The final letter of the mnemonic specifies the upper (U) or the lower (L) microinstruction to be selected if the condition specified by the first letters of the mnemonic is true.

When micromemory is being read or written as an operand, the T field is used to address the referenced micromemory location and the upper instruction in the next sequential microinstruction pair is always selected.

The following T field mnemonics may be included in any microinstruction:

T Code	Operation
*L	Select the lower of current microinstruction pair. This operation overrides M field addressing mode.
U	Select upper of next pair.
L	Select lower of next pair.
KZU	If initial contents of the K register is zero, select upper of next pair. Cannot be used with C field INCK.

T Code	Operation
NZU	If initial contents of the N register is zero, select upper of next pair. Cannot be used with C field INCN.
NU	If output of ALU is negative on completion of current microinstruction, select upper of next pair.
ZL	If output of the ALU is zero on completion of current microinstruction, select lower of next pair.

The following T field mnemonics are usable only in format 1 as shown in Figure 4-2. The use of these mnemonics with other formats, such as a jump or N= or K= in the C field, will cause an assembler diagnostic.

T Code	Operation
LQL	If at start of this microinstruction the least significant bit of Q is 1, take the lower of next microinstruction pair.
INTU	If there is an interrupt, and the corresponding mask bit in the mask register is set, execute the upper of the next microinstruction pair.
BTU	If the bit selected by the least significant four bits of the C field is set, execute the upper of the next microinstruction pair. Presently, only the following bits are selectable:

C Field Value	Test
2	Bit Register
5	Selective skip bit of FCR
9	Monitor/Program Mode Bit
F	Macro RUN/STOP Bit

T Code	Operation
COL	If there is a carryout of ALU from arithmetic operation, take lower of next pair.
K7L	If the least significant bit of K register is set, execute lower of next microinstruction pair; if clear, execute upper of next microinstruction pair.
OVFL	If overflow exists, execute the lower of the next microinstruction pair; if not, execute upper of the next microinstruction pair.
ERL	If any memory fault occurred, take the lower of the next microinstruction pair.

ASSEMBLER PROCESSING OF M AND T FIELDS

SEQUENTIAL ADDRESSING

If the M and T fields are left blank, the assembler will assume sequential addressing mode and will choose a T code in the object code output to execute the next sequential microinstruction. If the current microinstruction is an upper, the *L code will be inserted for the T field. If the current microinstruction is a lower, a U code will be inserted in the T field.

The example in Figure 4-12 shows assembly output of two sequences of code to show two ways of specifying sequential addressing.

In Figure 4-13, the instructions with NOP coded in the D field are not executed, but the other instructions with coding are executed. This example shows how it is possible to interleave two paths of program flow through one set of micromemory locations.

JUMP ADDRESSING

If a J is coded in the M field, the C field is interpreted as the jump destination address. The C field may contain a symbol or a constant. A symbol is carried as an actual micromemory location address and has an upper or lower property as well. A constant is interpreted as an upper micromemory address. If no T field value is specified, the assembler provides the correct T field value for the location addressed. However, the default T field selection is overriddened if the programmer specifies a value in the T field.

The assembler compares the page of the jump destination address with the page of the current microinstruction. If the page numbers are the same, a within-page jump is coded and the S field may be used for additional instructions. If the pages are different, a page jump is coded and the page number is extracted from the constant or symbol value and inserted

```
HEX   HEX      CYCLE   LABEL    F     A     B     D     S     C        MT    COMMENTS
MML   IMAGE    COUNT

                      ********************************************************************************
                      *         ASSEMBLER GENERATED SEQUENTIAL ADDRESSING                            *
                      ********************************************************************************

006 U 5F06 0000  3    +         A     A           X
    L 5425 2000  3              B           Q     A
007 U 5EC3 0000  3              A     X           Q

                      ********************************************************************************
                      *         PROGRAMMER  SEQUENCE.  NOTE SAME MICRO CODE GENERATED.               *
                      ********************************************************************************

008 U 5F06 0000  3    +         A     A           X                    S*L
    L 5425 2000  3    -         B           Q     A                    SU

009 U 5EC3 0000  3    +         A     X           Q                    S*L
HEX   HEX      CYCLE   LABEL    F     A     B     D     S     C        MT    COMMENTS
MML   IMAGE    COUNT
```

Figure 4-12. Example of Sequential Addressing

```
HEX   HEX      CYCLE   LABEL    F     A     B     D     S     C        MT    COMMENTS
MML   IMAGE    COUNT

                      ********************************************************************************
                      *         FURTHER SEQUENTIAL ADDRESSING                                        *
                      ********************************************************************************

00A U 5F06 2000  3    +         A     A           X                    SU
    L 5800 2000  3                                      NOP

00B U 5425 4000  3    +         B           Q     A                    SL
    L 5800 2000  3                                      NOP

00C U 5800 0000  3    +                                 NOP
    L 5EC3 2000  3              A     X           Q                    SU

00D U 5806 0000  3    +         ZERO              X                     *L
HEX   HEX      CYCLE   LABEL    F     A     B     D     S     C        MT    COMMENTS
MML   IMAGE    COUNT
```

Figure 4-13. Further Examples of Sequential Addressing

in the S field for the object code. The location within a page is coded in the C field of the object code. If the programmer has used a value in the S field and a page jump is coded, a diagnostic will be generated.

The example in Figure 4-14 shows four methods of arriving at a specific microinstruction.

RETURN ADDRESSING

Return addressing causes control to be returned to the microinstruction pair specified by the contents of the RTJ register. The programmer must specify a value in the T field to get a correct return location (upper or lower of the microinstruction pair). The RTJ register may be set any time by placing the mnemonic RTJ in the S field of a microinstruction. The address stored into the RTJ register is that of the next sequential micromemory word following the instruction with RTJ in the S field. Both page and address within a page are stored in the RTJ register.

The example in Figure 4-15 shows use of return addressing. In this example, a jump is made to the routine SUB which tests the value in the A register. Return is to the lower of the following microinstruction pair if the value in the A register is negative, and returns to the upper of the pair if the value is positive.

```
HEX  HEX         CYCLE  LABEL    F     A     B     D     S     C         MT     COMMENTS
MML  IMAGE       COUNT

                        ****************************************************************************
                        *        JUMP  ADDRESSING                                                  *
                        ****************************************************************************

400 L 9800 2003   3                                          LOCA      J
401 U 9800 2003   3                                          LOCAA     J
    L 9800 2003   3                                          LOCA      JU
402 U 9800 2003   3                                          LOCB      JU

403 U 5800 0000   3     +LOCA                          NOP
    L 5800 2000   3     -LOC3                          NOP

404 U 9800 4003   3                                          LOCB      J
    L 9800 4003   3                                          LOCBB     J
405 U 9800 4003   3                                          LOCB      JL
    L 9800 4003   3                                          LOCA      JL
403                      LOCAA   EQU   LOCA
403                      LOCBB   EQU   LOCB

HEX  HEX         CYCLE  LABEL    F     A     B     D     S     C         MT     COMMENTS
MML  IMAGE       COUNT
```

Figure 4-14. Examples of Jump Addressing

```
HEX  HEX        CYCLE  LABEL   F    A     B    D    S     C       MT   COMMENTS                          DIAGNOSTICS
MML  IMAGE      COUNT

                       ******************************************************************
                       *        JUMP EXAMPLES                                           *
                       *---------------    PAGE OR BLANK   -----------------------------*
                       ******************************************************************

200                            ORG  200X

200 U 9F06 4501  3             A    A          X    L8EA  LOCN0   J    JUMP IN BANK
    L 9800 3400  3                                        LOCN2   J    JUMP OTHER BANK
201 U 9F06 2500  3             A    A          X    L8EA  LOCN2   J    JUMP OTHER BANK (S ERROR)         CRP
    L 9800 2000  3     LOCN0                              200X    JU   JUMP THIS BANK
202 U 9800 3400  3                                        400X    JU   JUMP TO 00 IN PAGE 4
    L 9800 2000  3                                  NOP   400X    JU   JUMP TO PAGE 4 (S ERROR)          CRP

400                            ORG  400X                               START PAGE 4

400 U 5800 0000  3     LOCN2                        NOP                INSTRUCTION IN PAGE 4
```

Figure 4-14. Examples of Jump Addressing (Cont.d)

```
HEX  HEX        CYCLE  LABEL   F    A    B    D    S    C      MT   COMMENTS
MML  IMAGE      COUNT

                       ****************************************************************
                       *        RETURN ADDRESSING MODE                                *
                       ****************************************************************

00E U 9800 2E12  3     +                           RTJ  SUB1   J
    L 5800 2000  3     -

00F U 5800 0000  3     +                      NOP                   RETURN LOCATION UPPER
    L 5800 2000  3     -                      NOP                   RETURN LOCATION LOWER

                       *

010 U 9800 2E14  3     +                           RTJ  SUB2   J
    L 5800 2000  3     -                                            RETURN LOCATION (A -)

011 U 5800 0000  3     +                                            RETURN LOCATION (A +)
    L 5800 2000  3     -

                       *

012 U 5F00 C000  4      SUB1   A    A                          NU   TEST A

013 U 1800 4000  3     +                                       RL   RETURN LOWER IF A IS -
    L 1800 2000  3     -                                       RU   RETURN UPPER IF A IS +

014 U 1F00 C000  4      SUB2   A    A                          RNU  RETURN TO UPPER IF A IS -
                               REM                                    ELSE RETURN TO LOWER
```

Figure 4-15. Examples of Return Addressing

MAIN MEMORY INTERFACE 5

The Memory Interface performs the following basic functions:

- Converts a programmer address to a physical address.
- Decouples microprocessor and memory timing, using look-ahead.
- Formats address and data for full-word, half-word and character operations.

MEMORY ACCESSING

Supporting up to 4 million words of main memory and providing capabilities for virtual memory, memory protection and dynamic allocation is achieved by logically dividing memory into blocks, or pages. Each page consists of 2048 contiguous words. To access a maximum memory of 4 million words requires 2048 pages and an address of 22 bits. The required 22-bit address is divided into an upper and lower portion of eleven bits each. The upper portion selects one of the 2048 possible pages, while the lower portion selects one of the 2048 words within the selected page. The lower portion comes from 11 word address bits of the macroinstruction. The upper portion is supplied from an entry in the page index file.

Figure 5-1 shows the creation of the actual physical address from the macroinstruction operand fields and the page index file.

PAGE INDEX FILE

The Page Index File is a 1024, 17-bit bipolar random access memory. The Page Index File is micro-accessible (see WPF, RPF instructions) and is logically divided into 32 groups of 32 entries each. The CPU Program State Register, when in State Mode (or 5 bits from the macroinstruction, when in Direct Mode), selects one of the 32 groups. Five bits from the upper part of the word address in the macroinstruction selects one of the 32 entries in the group. This was done to help assist in the emulation of the MP-60 instruction set.

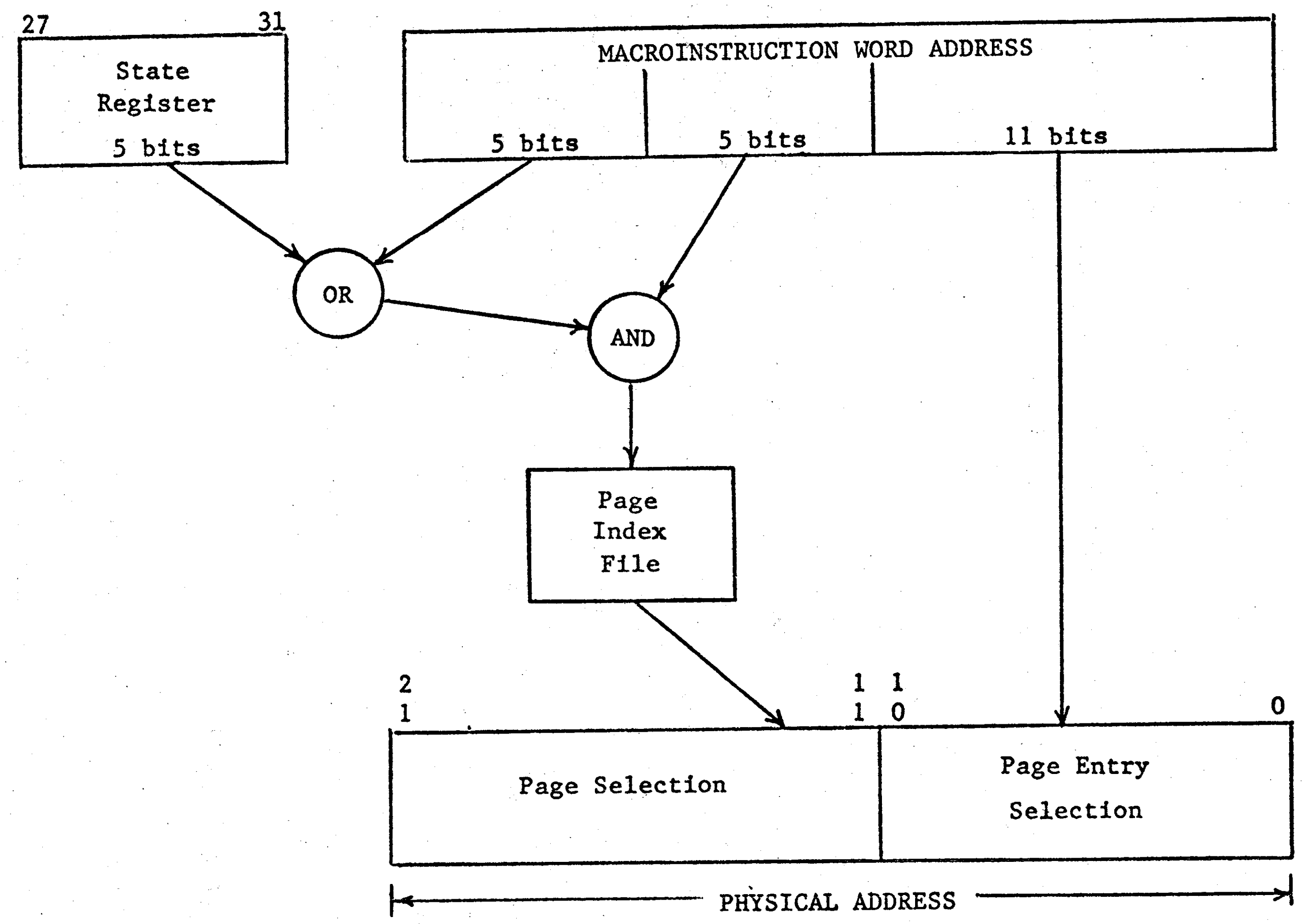


Figure 5-1. Creation Of The Actual Physical Address

Each of the 32 groups represents an individual state. Each of the entries in that group represents a logical page of 2048 words, so a state has a logical maximum size of 65K words.

Along with providing page addresses, the Page Index File supplies odd parity information to the memory protect logic.

MEMORY PAGE ADDRESSING

As shown in Figure 5-2., each Page Index File entry contains an 11-bit address for a particular page of main memory. Together with 11 bits from the word address in the macroinstruction, this provides addressing for over 4 million words of main memory. Although the Page Index File can hold only 1024 of the 2048 possible page addresses at any one time, the file can be changed by microprogram instructions to allow access of the maximum memory present. Actual memory address generation is discussed under MEMORY ACCESSING.

The page address parity logic generates an odd parity bit on the 11-bit page address as it is loaded from selector 1. This bit is stored at bit position 15 when the page address is stored in the page index file. When this page file entry is accessed, the parity bit is sent with the page address to the holding register.

PAGE STATUS

Bit positions 17 and 20 hold the MODIFIED and ACCESSED status bits respectively. The page status logic monitors activity of the Page Index File. When a Page Index File entry is accessed for a read operation, the page status logic sets the ACCESSED bit in that entry. On a write operation, both the ACCESSED and MODIFIED bits are set in the corresponding Page Index File entry.

MEMORY PROTECT LOGIC

Bits 16, 18 and 19 specify the protect state associated with each page. These bits are monitored by the protect logic to ensure that no illegal memory operations are allowed to occur. These bits are used as follows:

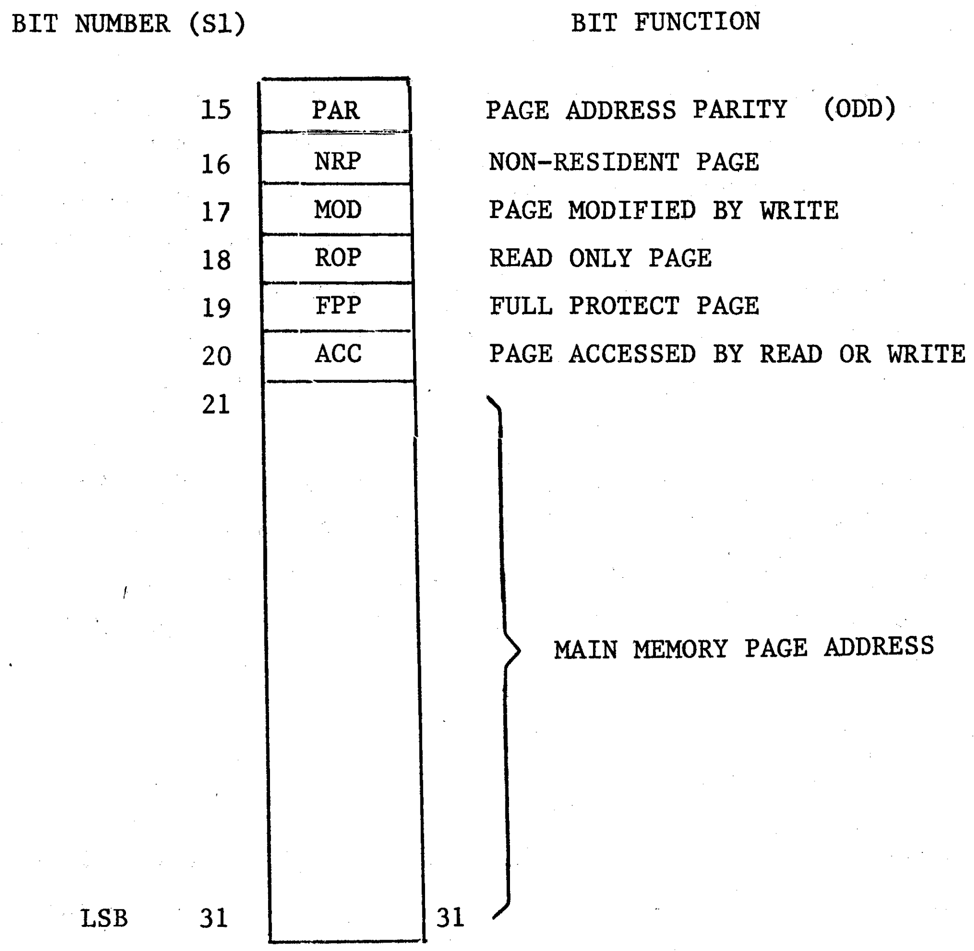


Figure 5-2. Page Index File Entry Bit Assignments

- Non-Resident Page - Bit 16 indicates that the page is physically or logically not present in the system and cannot be accessed.
- Read Only Page - Bit 18 indicates that the page may only be accessed by read instructions.
- Full Protect Page - Bit 19 indicates that the page is not available for any access.

PROGRAM STATE REGISTERS

The memory interface contains one (1) CPU State Register and four (4) DMA State Registers, one for each DMA port. These state registers perform the following functions:

- Provide the upper half of the Page Index File address for main memory addressing.
- Provide mode control.
- Report fault conditions.

CPU STATE REGISTER

The CPU State Register is accessible by using the WSR, RSR mnemonics in the C field. The CPU State Register bit assignments are shown in Figure 5-3 and are used as follows:

- Mode Control - Bits 20-22 are mode control bits for the three lookahead address/data register pairs of the CPU. Setting a bit puts the corresponding lookahead register pair in the write mode. Clearing a bit puts the corresponding lookahead register pair in the read mode.
- Addressing - Bits 27-31 form the upper portion of the Page Index File address, selecting one of 32 groups of 32 Page Index File entries. Refer to MEMORY ACCESSING section for a discussion of address generation. Bits 27-31 are also routed to the backpanel for use by the transform module. These bits are undefined after a master clear.

- Fault Reporting - Bits 15-26 are used to report faults that have occurred during CPU memory operations. The bits are set by the fault detection hardware and may be cleared by using a read-modify-restore microcoded sequence.

- Bits 13-14 - These bits are constantly updated and have no significance until a DMA error occurs (bit 23 is set). When this happens, they contain the number of the DMA port which caused the error. Bit 23, when set, prevents any changes to bits 13-14 until a WSR (Write State Register) function is issued to the CPU State Register. These bits are undefined after a Master Clear.

- Bit 15 - This bit will be set when a timeout error occurs, along with one of the lookahead error bits (bits 23-26) indicating which one generated the timeout error. This bit is cleared with a WSR instruction or a Master Clear.

- Bits 16-19 - These bits, all of which could be set simultaneously, indicate a parity error on the corresponding data character(s). One of the lookahead bits (bits 23-26) would be set at the same time, indicating which lookahead register set caused the error and, at the same time, blocking any further setting of data parity error flags by subsequent errors. A Master Clear or a WSR instruction will clear these bits and remove the inhibit update signal.

- Bits 23-26 - These error flags are set on any error to indicate which lookahead register set generated the error. In addition, these bits prevent any further update of any other error information until the CPU clears these bits with a WSR instruction or a Master Clear. Any bit set will inhibit the updating of state register bits 16-19 (Data Parity bits). Bit 23 will inhibit the updating of state register bits 13-14 (DMA port number in error). Presently, only lookahead 1 is testable using T field logic. ERL in the T field would force execution to start in the lower of the next microinstruction pair on a fault.

BIT NUMBER	BIT ASSIGNMENT	
13-14	DMA PORT NUMBER IN ERROR	
15	TIMEOUT ERROR	
16	CHAR 0	DATA PARITY ERROR
17	CHAR 1	DATA PARITY ERROR
18	CHAR 2	DATA PARITY ERROR
19	CHAR 3	DATA PARITY ERROR
20	LOOKAHEAD 1	READ/WRITE BIT
21	LOOKAHEAD 2	READ/WRITE BIT
22	LOOKAHEAD 3	READ/WRITE BIT
23	DMA (LOOKAHEAD 0)	ERROR FLAG
24	LOOKAHEAD 1	ERROR FLAG
25	LOOKAHEAD 2	ERROR FLAG
26	LOOKAHEAD 3	ERROR FLAG
27-31	UPPER 5-BIT PAGE INDEX FILE ADDRESS PORTION	

DMA STATE REGISTERS

The DMA State Registers are used to perform two functions for the CPU: they provide Page Index File information for main memory addressing and transfer mode control (half word/full word) information to the DMA port logic. The selection of the DMA State Register to be read or written is determined by the least significant two bits from S3. All mode bits and set of address bits are written to a Data State Register with a WSR instruction. All mode bits and one set of address bits are read back with each RSR instruction. Bit assignments are shown below.

BIT NUMBER	BIT ASSIGNMENT
23	TRANSFER MODE BIT -- PORT 0
24	TRANSFER MODE BIT -- PORT 1
25	TRANSFER MODE BIT -- PORT 2
26	TRANSFER MODE BIT -- PORT 3
27-31	UPPER 5-BIT PAGE INDEX FILE REGISTER PORTION

- Bits 23-26 - These are transfer mode bits selecting either half word (0) or full word (1) operation for the corresponding DMA port.

- Bits 27-31 - Upper 5-bit Page Index File address portion.

CODING EXAMPLES FOR MEMORY INTERFACE

Figure 5-4 shows coding for reading and writing the contents of a Page Index File entry. Reading and writing the program and DMA state registers are shown in Figure 5-5. Reading physical location 00 using state 2 and Page Index File entry 34 are shown in Figures 5-6 and 5-7. Placing the address of a main memory word within the word for the first 65,536 locations is shown in Figure 5-8. Figure 5-9 is an example of using a TK transform and lookaheads to perform a main memory cycle. Figure 5-10 shows a TMA transform and the processing of interrupts.

```
HEX   HEX      CYCLE   LABEL    F     A     B     D     S     C        MT    COMMENTS
MML   IMAGE    COUNT

                      *********************************************************************************
                      *         WRITING A PAGE FILE ENTRY,    X-CONTAINS ADDRESS, A-DATA              *
                      *********************************************************************************

037 U 5518 002C  3    +         B     A     X                 WPF

                      *********************************************************************************
                      *         READING A PAGE FILE ENTRY,    X-CONTAINS ADDRESS                      *
                      *********************************************************************************

037 L 5418 202E  3              B           X                 RPF            PAGE FILE TO
038 U 5FC5 0000  3              A     MEM         A                          A-REGISTER

                      *********************************************************************************
                      *         WRITING THE PAGE FILE ENTRY WITH ITS ADDRESS                          *
                      *********************************************************************************

01F                    ONE      SET   31
015                    C1024    EQU   21
038 L 5806 2000  3              ZERO              X                    U

039 U 5EC0 002C  3     LOAD     A     X                       WPF            PF=X
    L 7006 201F  5              ADD   X     BG    X           ONE            X=X+1
03A U 5C06 E015  4              A.B   X     BG    X           C1024    ZL    DOES X = 1024

03B U 5800 2000  3    +                                                U     DONE
    L 9800 2039  3                                            LOAD     J     LOOP

HEX   HEX      CYCLE   LABEL    F     A     B     D     S     C        MT    COMMENTS
MML   IMAGE    COUNT
```

Figure 5-4. Examples of Reading and Writing a Page Index File

```
HEX   HEX        CYCLE   LABEL    F      A      B      D      S      C         MT     COMMENTS
MML   IMAGE      COUNT

                         *****************************************************************************
                         *        WRITING THE STATE REGISTER,   X-CONTAINS ADDRESS, A-DATA            *
                         *****************************************************************************

03C U 5518 002D   3               B      A      X                    WSR

                         *****************************************************************************
                         *        READING THE STATE REGISTER,                                         *
                         *****************************************************************************

03C L 5800 202F   3                                                  RSR
03D U 5FC5 0000   3               A      MEM           A

                         *****************************************************************************
                         *        READING THE CPU STATE REGISTER, MOVING IT TO DMA 3                  *
                         *        STATE AND INCREMENT IT BACK INTO THE CPU STATE REGISTER             *
                         *****************************************************************************

01D                      FOUR     SET    29
01F                      ONE      SET    31
03D L 5415 201D   3               B             BG     A             FOUR              A=4
03E U 5800 002F   3                                                  RSR
    L 553E 202D   3               B      A      MEM    X             WSR               X=CPU STATE
03F U 70D6 001F   5               ADD    X      BG     X             ONE               X=X+1
    L 58C0 202D   3               ZERO   X                           WSR               CPU STATE=X
HEX   HEX        CYCLE   LABEL    F      A      B      D      S      C         MT     COMMENTS
MML   IMAGE      COUNT
```

Figure 5-5. Examples of Reading and Writing State Registers

```
HEX   HEX      CYCLE   LABEL    F     A     B     D     S     C        MT    COMMENTS
MML   IMAGE    COUNT

                      ********************************************************************************
                      *         READ PHYSICAL MEMORY LOCATION ZERO USING CPU STATE 0.                *
                      ********************************************************************************

040 U 5805 0000  3              ZERO              A                          CLEAR CPU
    L 5906 2020  3              ZERO  A           X           WSR            STATE REGISTER

                      *

041 U 5518 002C  3              B     A     X                 WPF            CLEAR PAGE

                      *                                                      FILE ENTRY
                      *

041 L 5418 2000  3              B           X                 LWA1           LOAD ADDRESS
                                REM                                            REGISTER 1

                      *
                      *         ANY DELAY OR PROCESSING WOULD GO HERE
                      *

042 U 5800 0000  3                                      RO1                  INFORM AMI-NEED DATA
    L 5FC5 2000  3              A     MEM         A                          DONE, DATA TO A-REG.

HEX   HEX      CYCLE   LABEL    F     A     B     D     S     C        MT    COMMENTS
MML   IMAGE    COUNT
```

Figure 5-6. Example of Reading Location 0 Using State 0

```
HEX  HEX          CYCLE  LABEL   F      A      B      D      S      C          MT      COMMENTS
MML  IMAGE        COUNT

                         ********************************************************************************
                         *          READ PHYSICAL MEMORY LOCATION ZERO USING                            *
                         *                CPU STATE 2 AND PAGE FILE ENTRY 34.                           *
                         ********************************************************************************

01E                       TWO    SET    30                                                BIT FOR BG VALUE OF 2
01A                       C32    SET    26                                                BIT FOR BG VALUE OF 32
043 U 5415 001E   3              B             BG     A             TWO                   A=2
    L 5900 202D   3              ZERO   A                           WSR                   STATE=2
044 U 5715 001A   3              A+B    A      BG     A             C32                   A=32+2
                         *

044 L 5900 202C   3              ZERO   A                           WPF                   PAGE (34) =0
                         *

012                       C8192  SET    18                                                BIT FOR BG VALUE OF 8192
045 U 5416 0012   3              B             BG     X             C8192                 X=0001000000000000
    L 5418 2000   3              B             X                    LWA1                  LOAD ADDRESS
                                 REM                                                        REGISTER 1
                         *
                         *       ANY DELAY OR PROCESSING WOULD GO HERE
                         *
                                                             RD1                          INFORM AMI-NEED DATA
046 U 5800 0000   3              A      MEM           A                                   DONE, DATA TO A-REG.
    L 5FC5 2000   3
HEX  HEX          CYCLE  LABEL   F      A      B      D      S      C          MT      COMMENTS
MML  IMAGE        COUNT
```

Figure 5-7. Example of Reading Location 0 Using State 2 and Page Index File

```
HEX   HEX        CYCLE   LABEL    F      A      B      D      S      C          MT    COMMENTS
MHL   IMAGE      COUNT

                         ****************************************************************************
                         *        WRITE PHYSICAL LOCATIONS WITH THEIR ADDRESS (FIRST 65K)            *
                         ****************************************************************************

01F                      ONE      SET    31
00F                      C65536   SET    15                                               BIT FOR BG VALUE OF 65536
047 U D805 1010  3                ZERO                 A             N=16                 CLEAR CPU
    L 5906 202D  3                ZERO   A             X             WSR        U           STATE REGISTER

                         *

048 U 5F00 002C  3       +        A      A                           WPF                  INITIALIZE
    L 7115 221F  5                ADD    A      BG     A      RPT    ONE        U           PAGE REGISTER

                         *

049 U 5EC0 0020  3       +BEGIN   A      X                           HRFW                 WRITE MEMORY
    L 7006 201F  5                ADD    X      BG     X             ONE                  X=X+1
04A U DCCF 200F  3                A.B    X      BG                   C65536     ZU        TEST COMPLETE

                         *

04B U 9800 204C  3       +                                           DONE       J         DONE
    L 9800 2049  3                                                   BEGIN      J         LOOP
04C                      DONE     EQU    *

HEX   HEX        CYCLE   LABEL    F      A      B      D      S      C          MT    COMMENTS
MHL   IMAGE      COUNT
```

Figure 5-8. Example of Placing a Location's Address in the Location

```
CYBER/MP-60 MICRO ASSEMBLER PSD VERSION 1.0       TK TRANSFORM / LOOKAHEAD EXAMPLE              10.52.58. 77/08/02. PAGE    1

HEX  HEX      CYCLE   LABEL    F     A     B     D     S     C        MT     COMMENTS                                DIAGNOSTICS
MML  IMAGE    COUNT

                               IDENT       TK TRANSFORM / LOOKAHEAD EXAMPLE

                      ********************************************************************************
                      *         THIS EXAMPLE WAS TAKEN OUT OF AN EXTENDED VERSION OF THE MP60         *
                      *         EMULATOR AND SOMEWHAT SIMPLIFIED.  THIS CODE IS DESIGNED TO           *
                      *         EXECUTE AN ASSEMBLY INSTRUCTION WHICH AN INCLUSIVE-OR OF EACH         *
                      *         ELEMENT OF TWO ARRAYS.  THE ASSEMBLY INSTRUCTION IS ASSUMED TO        *
                      *         BEEN READ FROM MAIN MEMORY AND IS RESIDING IN THE IXT REGISTER.       *
                      *         THE A FIELD (BITS 11-15) OF THE ASSEMBLY INSTRUCTION SPECIFIES THE    *
                      *         REGISTER WHICH CONTAINS THE ADDRESS OF ONE OF THE ARRAYS AND THE      *
                      *         FINAL DESTINATION ARRAY.  THE B FIELD (BITS 16-20) SPECIFIED THE      *
                      *         REGISTER WHICH CONTAINED THE ADDRESS OF THE SECOND ARRAY.  THE        *
                      *         C FIELD (BITS 27-31) SPECIFIED WHICH REGISTER CONTAINS THE LOOP       *
                      *         COUNT (I.E. HOW MANY ARRAY ELEMENTS TO INCLUSIVE-OR).  ESSENTIALLY    *
                      *         THE CODE DOES THE FOLLOWING:                                          *
                      *                                                                               *
                      *                     DO 10 I=1,CF                                              *
                      *               10    IARRAY(AF)=IARRAY(AF).OR.IARRAY(BF)                       *
                      *                                                                               *
                      *         IT IS ASSUMED THAT THE FOLLOWING HAS ALREADY OCCURRED:                *
                      *                                                                               *
                      *                     LWA1=A ARRAY ADDRESS                                      *
                      *                     X REG=ADDRESS OF A ARRAY                                  *
                      ********************************************************************************

002                   BF       EQU   2     B FIELD DECODE
007                   CF       EQU   7     C FIELD DECODE
000 U 5800 1032  3                                             TK/BF          K=B FIELD REG NUMBER
    L 5F80 202C  3             A     F1                        LWA3           FETCH B ARRAY ADDRESS
001 U 5800 1037  3                                             TK/CF          K=C FIELD REG NUMBER(LOOP COUNT)
    L 6185 F032  6             A-    F1          A             TK/BF    ZL    DROP LOOP COUNT, NEXT LOWER IF ZERO

                      *  A=LOOP COUNT
                      *  K=B FIELD REG NUMBER
                      *  X=LWA1=A ARRAY ADDRESS
                      *  LWA3=B ARRAY ADDRESS

002 U 78C7 4400  4    +AOR.10  A+    X           F     RD1              L     BUMP A ARRAY ADDR
    L 5800 2400  3                                     RD1                    READ LAST A ARRAY ELEMENT

003 U 95C2 2006  3                   MEM         I             AOR.20   J     I=A ARRAY ELEMENT,GO FINISH OFF

                      *  A=LOOP COUNT
                      *  K=B FIELD REG NUMBER
                      *  X=LWA1=A ARRAY ADDRESS
                      *  LWA3=B ARRAY ADDRESS
                      *  F=A ARRAY ADDRESS+1

003 L 55EA 2024  3             B     MEM   F     I             LWA1           I=A ARRAY ELEMENT,LWA1=A ARRAY ADDR
004 U 7984 032C  4             A+    F1          F1    RD3     LWA3           BUMP A ARRAY ADDR
    L 568F 2000  3             A+B   I     MEM   F                            INCLUSIVE-OR A,B ARRAY ELEMENTS

HEX  HEX      CYCLE   LABEL    F     A     B     D     S     C        MT     COMMENTS                                DIAGNOSTICS
MML  IMAGE    COUNT
```

Figure 5-9. TK Transform/Lookahead Example

```
CYBER/MP-60 MICRO ASSEMBLER PSD VERSION 1.0       TK TRANSFORM / LOOKAHEAD EXAMPLE                     10.52.58. 77/08/02. PAGE    2

HEX   HEX        CYCLE   LABEL    F       A      B      D      S      C          MT    COMMENTS                                  DIAGNOSTICS
MML   IMAGE      COUNT

005 U 5558 0028  3                B       F      X                    LWA2             WRITE RESULT INTO A REG
    L A105 E002  6                A-      A             A             AOR.10     JZL   DECREMENT LOOP COUNT

                         *  LOOPING COMPLETED

006 U 5E00 0300  3       AOR.20   A                            RD3                     READ IN B ARRAY ELEMENT
    L 56BF 2000  3                A+B     I      MEM    F                              INCLUSIVE-OR LAST A,B ARRAY ELEMENT
007 U 5558 0028  3                B       F      X                    LWA2             WRITE RESULT INTO A ARRAY
                                  END

HEX   HEX        CYCLE   LABEL    F       A      B      D      S      C          MT    COMMENTS                                  DIAGNOSTICS
MML   IMAGE      COUNT

*********************************** NO ERRORS           *********************************** .
```

Figure 5-9. TK Transform/Lookahead Example (Cont'd)

```
CYBER/MP-60 MICRO ASSEMBLER PSD VERSION 1.0 TMA / INTERRUPT EXAMPLE                    10.52.32. 77/08/02. PAGE    1
                                                                                                 DIAGNOSTICS
HEX   HEX        CYCLE   LABEL     F      A      B      D      S      C          MT    COMMENTS
MML   IMAGE      COUNT

                                   IDENT TMA / INTERRUPT EXAMPLE

                         **********************************************************************
                         *         THIS IS AN EXAMPLE OF PROCESSING INTERRUPTS USING A TMA    *
                         *         TRANSFORM TO VECTOR TO THE CORRECT INTERRUPT PROCESSING    *
                         *         ROUTINE                                                    *
                         **********************************************************************

                                   FILE  2
000   0003 0000 F2       MSK.REAL  HEX   30000         REAL-TIME INTERRUPT 01-00 ENABLE MASK
                                   ENDF

000 U D800 0000   3      INT.EXPL                                   K=0              SET K TO INDEX FILE 2
    L 5F84 2900   3                A      F1           M1                            ENABLE REALTIME INT 03-01
001 U 5800 A000   3      INT.CHK                                                INTU CHK FOR PEND INTERRUPT
    L 9800 2001   3                                                 INT.CHK    J     GO BACK ^ CHK INTERRUPTS

002 U 5430 3807   5      +         B            INTA                TMA/7      U     JMP TO INTERRUPT ADDR

030                                ORG   30X

030                      REAL.00   EQU   *
                         *         :     :     REAL-TIME 00 PROCESSING CODE
                         *         :     :
                         *         :     :

040                                ORG   40X

040                      REAL.01   EQU   *
                         *         :     :     REAL-TIME 01 PROCESSING CODE
                         *         :     :
                         *         :     :
                                   END
                                                                                                 DIAGNOSTICS
HEX   HEX        CYCLE   LABEL     F      A      B      D      S      C          MT    COMMENTS
MML   IMAGE      COUNT
*********************************** NO ERRORS                 ***********************************
```

Figure 5-10. TMA Transform/Interrupt Example

MPSIM MICROCODE SIMULATOR 6

MPSIM is an interactive simulator which simulates the microcode execution of the CDC Microprogrammable Processor (MP) and the Basic System Controller, Model 65109. It is designed to run interactively using a remote terminal, or in batch mode, on a CDC CYBER 170/70/6000 computer running the NOS/BE or KRONOS operating system.

MPSIM provides a tool to the microprogrammer which allows him to check his coding without actually running it on the MP. Features are provided which make debugging much easier using MPSIM rather than the MP. Some of these features are:

- Communication with MPSIM using symbols from the MASS assembly listing is possible.
- Tracing of instruction execution ranging from full trace, to memory writes, to no tracing may be selected.
- Multiple breakpoints may be set.
- Extensive trapping facilities are available.
- A history of all interactive operations may be captured and printed.
- Execution may be suspended and the current state of the simulation saved on a file.

MPSIM OPERATION

While MPSIM may be run as a batch job, it is strongly recommended that the user become familiar with MPSIM by interactive operation before batch use is attempted. In interactive mode, a question mark (?) is displayed on the terminal any time MPSIM is ready for user input and displays are always sent to the terminal unless DUPONLY is used. In batch mode, "display" is equivalent to "print".

First, a microprogram must be available for simulation. The microprogram is assembled on the CYBER, using the MASS assembler. The MASS option SIMULATE must be specified on the OPTN pseudo instruction to produce a binary simulation file (SIMFIL). Once this file has been created, the user logs in on an interactive terminal, attaches or generates SIMFIL and executes MPSIM. The first command to MPSIM is typically LOAD. LOAD reads SIMFIL into MPSIM which initializes micromemory and MPSIMs symbol table. Breakpoints may be set to return control to the user when a particular location in the microcode is reached. Traps may be set to cause various displays when specified microcode locations are executed. Normally, a trace of all instructions executed is given, along with accumulated simulated time. Further commands as described below may be given to control the running of the simulation. See the EXAMPLE RUN given later in this document.

Using the TLIM command, the user can establish a time limit for microcode "execution" after which control is returned to the user. Otherwise, the time limit is disabled. The execution time is the accumulated microinstruction execution time since the first RUN command or since a later RUN that specified an address.

INTERACTIVE COMMANDS

The following notation is used in command descriptions:

Brackets [] indicate optional elements which may be omitted or included as desired.

Braces {} indicate possible elements, one of which must be chosen and the other omitted.

Ellipsis .... indicate a variable number of similar elements may be used.

Words in UPPER CASE indicate specific terms.

Words in lower case represent information the user is to supply.

Values and addresses are always in hexadecimal, unless noted.

The term mmadr represents a micromemory address of the form paaU or paaL where:

p	= micromemory page
aa	= micromemory address
U/L	= Upper / Lower

A file or main memory address is indicated by adr.

The term symbol is a location field symbol from the MASS assembly listing.

All commands may be abbreviated by the fewest number of characters that make the command unique. If the abbreviation does not make the command unique, or if the command is not valid, "COMMAND ERR" is displayed.

LOAD COMMAND:

$$\text{LOAD}\left[,\ [\text{MP32}]\ \left[,\ [\text{preset}]\ \ [,\text{lfn}]\right]\right]$$

MP32	machine type
preset	hexadecimal value which is put into all micromemory words which are not loaded.
lfn	name of file which contains the binary simulation output created by the MASS Assembler; file is rewound before loading.
Default:	LOAD,MP32,0,SIMFIL
EXAMPLE:	LOAD,MP32,FFFF,LDBIN

Load the coding to be simulated from a file produced by the MASS Assembler. The machine type is selected, "lfn" is rewound and then loaded. Any micromemory locations which are not loaded are set to the "preset" hexadecimal value.

Micromemory size is initialized to the loaded microcode length rounded up to the next multiple of 512; FILE 1 size is 256, FILE 2 size is 32. The assembler symbol table is also loaded for symbol identification. Simulated main memory for macro data and instructions is not initialized. Dynamic CYBER memory management and allocation is used to ensure that minimum memory is used for execution.

RUN COMMAND:

$$\text{RUN}\left[,\ \left\{\begin{matrix}\text{symbol}\\ \text{mmadr}\end{matrix}\right\}\right]$$

Default: Use address from previous execution or 000U if first time.

EXAMPLE: RUN,12L

Start the simulation from the indicated micromemory location, the symbolic address or continue the simulation. If an address is specified, the cumulative simulated execution time is reset to zero.

SET COMMAND:

$$\text{SET,} \left\{ \begin{array}{l} \text{SHOW} \\ \text{ALL} \\ \text{reg} \end{array} \left[, \left\{ \begin{array}{l} \text{value} \\ \text{symbol} \end{array} \right\} \right] \right\}$$

"reg" designation may be any one of the following:

A,Q,X,F,P,I,N,K,RTJ,SM1,SM2,M1,M2,IN1,IN2,Y,D,RS,RD

SHOW lists present contents of all registers.

ALL sets the A,Q,X,F,P,I,N,K,RTJ,SM1,SM2,M1,M2,IN1,IN2,Y,D,RS,RD registers to the value specified or to the symbol addresds. Value is the next hexadecimal value to be entered into the register(s); default = 0. Symbol is symbol address to be entered into the register(s).

EXAMPLE: SET,F,02B3

Set the specified or all the register(s) to "value". The above register mnemonics are standard register names, except RS (status) and RD (data) which are pseudo registers which hold the status or data to be retrieved when INRS or INRD instructions are executed.

TRACE COMMAND:

```
         ( SHOW                                                     )
         ( OFF                                                      )
TRACE,  {  ALL                  [ {        ,TRAP                } ] }
         ( type-1, . . . type-n [ { [,lim-1, lim-2]    [,out ]  } ] )
```

SHOW	lists current trace or trap settings.
OFF	disables all tracing or trap display.
ALL	enables all trace types for normal or trap.
TRAP	specifies type of display when traps are encountered.
"type"	may be any of the following:

TIME	Display microcode execution time and addresses of just completed and next microinstructions. TIME trace type is also enabled with all other trace types.
REGS	Display registers which have changed.
FILES	Display FILE 1 and FILE 2 entries which have changed.
READS	Display results of main memory reads.
WRITES	Display results of main memory writes.

"lim-1" and "lim-2" are micromemory addresses (mmadr or symbols) and define the normal (not TRAP) trace limits; default is "lim-1" = 0 and "lim-2" = maximum micromemory. OUT specifies trace only outside the limits.

Defaults: TRACE, REGS, TIME, FILES, READS, WRITES, MM beginning, MM end.

EXAMPLE: TRACE,REGS,WRITES,030U,50L

TLIM COMMAND:

```
TLIM    [,num]
```

EXAMPLE: TLIM,295

Set/clear the microcode execution time limit. If "num" is not present, the time limit will be disabled. "num" is a decimal integer number of microseconds.

BKP COMMAND:

```
      ( SHOW                 )
BKP,  ) ALL, CLEAR           (
      ) symbol  [          ] (
      ( mmadr   [ , CLEAR  ] )
```

SHOW lists current breakpoint settings.

ALL,CLEAR clears all breakpoints.

A BKP, mmadr, CLEAR or BKP, symbol, CLEAR clears that breakpoint.

EXAMPLE: BKP,TAG2

Set or clear breakpoints. Up to 5 breakpoints may be set. Breakpoints always stop the simulation just before execution of the address specified and returns control to the user; to continue, use RUN.

DUMP COMMAND:

```
         ⎧ MM   [{,mmadr  }]  ⎫
         ⎪      [{,symbol }]  ⎪
DUMP,    ⎨ MEM  [          ]  ⎬  [,words]
         ⎪ F1   [{,symbol }]  ⎪
         ⎪ F2   [{,adr    }]  ⎪
         ⎪ PF   [          ]  ⎪
         ⎩ ST                 ⎭
```

MM	dump micromemory.
MEM	dump main memory.
F1	dump FILE 1 entries.
F2	dump FILE 2 entries.
PF	dump page index file entries.
ST	dump program and 8 DMA state registers.
words	hexadecimal number of words to dump, default is 1.

Starting location or entry (mmadr, symbol, or adr) default is 0.

EXAMPLE: DUMP,MEM,22,1F

Dump either memory locations or file entries starting at the address given and continuing for the specified number of words.

TRAP COMMAND:

```
         location      ┌──────────── Control ────────────┐ ┌──────────────── dump ────────────────┐
                                                            ┌                                      ┐
                                                            │  ⎧ MM   ┌⎧,mmadr ⎫┐ ⎫                │
                                                            │  ⎪      └⎩,symbol⎭┘ ⎪                │
TRAP, ⎧symbol⎫ ┌,first ┌,every ┌,times ┌⎧,GO  ⎫┐┐┐┐          │  ⎪ MEM  ┌          ┐ ⎪                │
      ⎩mmadr ⎭ │       │       │       └⎩,STOP⎭┘┘┘┘         │ ,⎨ F1   │⎧,adr    ⎫│ ⎬ [,words]      │
               └       └       └                            │  ⎪ F2   │⎩,symbol ⎭│ ⎪                │
                                                            │  ⎪ PT   └          ┘ ⎪                │
                                                            │  ⎩ ST                ⎭                │
                                                            └                                      ┘
```

or,

TRAP,SHOW

first	number of times to execute the address before the trap occurs; i.e., 3 indicates trap before the third execution of the address. "first" is a decimal integer. Default is 1.
every	when trapping has started, trap "every" executions of the address; i.e., 2 indicates trap every other execution of the address. "Every" is a decimal integer. Default is 1.
times	total number of traps to do. Trap will be removed after the 'times' time through the trap. "times" is a decimal integer. Default is 100.
GO	do not stop when trap encountered.
STOP	stop when trap encountered. (TRAP operates as a breakpoint).

Dump parameters are the same as for the DUMP command.

SHOW displays information about currently set traps.

Default: TRAP,adr,1,1,100,GO

EXAMPLE: TRAP,LOOP2,10,2,20,STOP,MM,20L,5

A trap is set at the given location to either display information defined by dump parameters or to return to the operator; control parameters specify when and how often the display or return is to be done. Up to 8 traps may be set. Trapping occurs before execution of the address of the trap. The ordinal number of the trap is displayed for use with the UNTRAP command. See TRACE command for defining information other than dump to be displayed when the trap is encountered.

UNTRAP COMMAND:

$$\text{UNTRAP,} \left\{ \begin{matrix} \text{num} \\ \text{ALL} \end{matrix} \right\}$$

EXAMPLE: UNTRAP,2

Remove a specific trap or all traps. "Num" is the ordinal number displayed when the trap is set, and is also displayed with TRAP,SHOW.

PATCH Command:

```
            ⎛ MM  ⎡ ⎧mmadr ⎫⎤ ⎞
            ⎜     ⎢,⎨symbol⎬⎥ ⎟
            ⎜     ⎣ ⎩*     ⎭⎦ ⎟
PATCH,      ⎜ MEM ⎡          ⎤⎟  ,value-1,...value-n
            ⎨ F1  ⎢ ⎧adr   ⎫ ⎥⎬
            ⎜ F2  ⎢,⎨symbol⎬ ⎥⎟
            ⎜ PF  ⎣ ⎩*     ⎭ ⎦⎟
            ⎝ ST                ⎠
```

Value-i   is hexadecimal value to be placed in indicated or subsequent location.

Other parameter definitions are listed under DUMP.

EXAMPLE: PATCH,MM,30U,FF2,345678F2,0,5AB43

Set values into memory locations or file entries. If * is used for the address, the first value will be stored in the next sequential address following the last value stored on the previous patch command.

When patching main memory, addresses are absolute.

ALTIN Command:

```
ALTIN [,lfn [,NOECHO]  [,NOREW]]
```

Default: ALTIN,ALTIN

EXAMPLE: ALTIN,INFILE,NOECHO

Read alternate input file lfn until EOR is encountered; lfn is rewound before reading unless NOREW is specified.

If running interactively and NOECHO is specified, input from the alternate file will not be displayed on the terminal. Output resulting from the input will be displayed, however.

DUPOUT Command:

```
DUPOUT  [,lfn]

EXAMPLE:    DUPOUT,000
```

Enables/disables duplication of the interactive commands and displays from the screen onto a file. The file thus written may later be routed to a printer for a history of interactive simulation activity. If "lfn" is omitted, duplication will be disabled. This command is only active when the simulator is operated interactively.

DUPONLY Command:

```
DUPONLY
```

This command enables output to the terminal for the duration of the output from the next command. Output goes only to the DUPOUT file if a DUPOUT command has been previously given. The DUPONLY command allows long outputs like DUMPs to be saved without the information being sent to the interactive display.

SNAP COMMAND:

```
SNAP [,lfn]
```

Default: SNAP,SNAP

EXAMPLE: SNAP,HOLDF

Capture the current state of the simulation on file lfn. File lfn is rewound before it is written.

UNSNAP COMMAND:

```
UNSNAP [,lfn]
```

Default: UNSNAP,SNAP

EXAMPLE: UNSNAP,HOLDF

Restore the state of a previous simulation saved with SNAP from file lfn. File lfn is rewound before it is read.

HELP COMMAND:

```
HELP [,command]
```

EXAMPLE: HELP,LOAD

Display an example of a particular command to prompt correct use. If "command" is omitted or invalid, a complete list of command examples will be displayed.

CPUTIME COMMAND:

```
CPUTIME
```

Display the ratio of CYBER cpu time to simulated execution time. Timing for ratio determination is re-initialized at each RUN command.

END COMMAND:

```
END
```

Exit from the simulator

DISPLAY MNEMONICS

This section describes the mnemonics used in display output of the registers and other intermediate data.

Hardware registers: A,Q,X,F,P,I,N,K,RTJ

Other Registers:

SM1	=	STATUS/MODE	1
SM2	=	STATUS/MODE	2
M1	=	MASK REGISTER	1
M2	=	MASK REGISTER	2

IN1	=	INTERRUPT REGISTER 1
IN2	=	INTERRUPT REGISTER 2
BSA	=	(A) BEFORE SHIFT OR SCALE
BSQ	=	(Q) BEFORE A-Q SHIFT OR SCALE
Y	=	PERIPHERAL ADDRESS REGISTER
D	=	PERIPHERAL DATA REGISTER
RS	=	DATA FOR INPUT BY INRS INSTRUCTION
RD	=	DATA FOR INPUT BY INRD INSTRUCTION

Read Operations:

R1 = DATA RETRIEVED WITH RD1 S-FIELD CODE

R2 = DATA RETRIEVED WITH RD2 S-FIELD CODE

R3 = DATA RETRIEVED WITH RD3 S-FIELD CODE

Address Operations for Main Memory:

Format is: ATO

where:	A	is	1, 2, or 3 for address register.
	T	is	type of access - W (word), H (halfword), C (character), or B (bit).
	O	is	operation - R (read) or W (write).

WPF	=	PAGE FILE ADDRESS FOR WRITE
RPF	=	PAGE FILE ADDRESS FOR READ
WSR	=	STATE REGISTER ADDRESS FOR WRITE
RSR	=	STATE REGISTER ADDRESS FOR READ

CONTROL CARD CALL

MPSIM

For interactive operation, there are no parameters associated with the control card. Input is obtained from the connected terminal and output is directed to the connected terminal (exceptions are ALTIN and DUPOUT commands).

MPSIM may be operated in batch mode. No parameters are read from the control card. Input is taken from file INPUT and output goes to file OUTPUT. Input directives are the same as in interactive operation. Output is not broken into screen-size blocks as done for interactive operation.

ERROR MESSAGES

Message	Command or State
LINE LEN NOT = 80	Initialization

Due to output blocking, interactive terminal line length must be 80 columns.

Message	Command or State
COMMAND ERR	All
INVALID REG	SET
ERROR IN TRACE LIMITS. LIMITS SET TO MAX MM.	TRACE
MAX BKPTS ALREADY SET	BKP
CANT FIND BKP	BKP, CLEAR
INVALID MEM TYPE	DUMP or PATCH
ADR EXCEEDS FL, DUMP TERMINATED	DUMP
MAX TRAPS ALREADY SET	TRAP
PARAM ERROR	UNTRAP
ADR EXCEEDS MEM	PATCH
MEM TYPE CHANGE W/* ILLEGAL	PATCH
INVALID MACHINE TYPE	LOAD
FILE EMPTY	LOAD
MMADR EXCEEDS MM SIZE, SIM STOPPED	Execution
ABSOL MEM ADR > MEM INITIALIZED, SIM STOPPED	Execution

Message	Command or State
UNKNOWN SYMBOL OR BAD HEX DIGIT	All
MM ADR ERROR	All
MPSIM - GO TO ERR, ABORT	All or Execution
FTN computed goto error processing is not used. This message results from a computed goto error.	
MPSIM RECOVERED	Execution
Recovery is included so that simulation may be stopped with %a (NOS/BE) or BREAK(NOS) and so that execution errors will not abort the simulation. An additional message describing the error is generated if the recovery was not caused by %a. The program is aborted without recovery if running in batch mode.	

INTERACTIVE EXECUTION EXAMPLE

Figure 6-1 shows interactive terminal input and output for a MASS assembly. Figure 6-2 shows the same program being run under MPSIM and the resulting output. In both figures, words in lower case are operator entries and upper case words are computer generated.

The following lines show terminal output for an example MASS and MPSIM run.

```
COMMAND- attach,demo,id=gbf.  get the microcode source.
 PFN IS
 DEMO
 PF CYCLE NO. = 001
COMMAND- attach,mass,id=colt. get the micro assembler.
 PFN IS
 MASS
 PF CYCLE NO. = 051
COMMAND- connect,output.
COMMAND- screen,132.
COMMAND- mass,demo,output.    assemble the source.
CYBER/MP-60 MICRO ASSEMBLER PSD VERSION 1.0 SETPAGE
HEX   HEX      CYCLE   LABEL    F     A     B     D     S     C        MT      COMMENTS
MML   IMAGE    COUNT

                                IDENT SETPAGE
                                OPTN  CROSS,L,I,MAP,SIMULATE,D

                      *

01F                    ONE      EQU   31    LSB OF WORD = ARITH 1
015                    ONEK     EQU   21    BIT GENERATOR GENERATES $400 = 1024D

                      *

000 U 5807 0000   3   +BEGIN    ZERO              F                                INITALIZE FLAGS
    L 5806 2000   3             ZERO              X                                INITALIZE CHCKSUM
001 U 5408 002D   3             B     X     X                 WSR                  SET CPU STATE = 0
    L 5805 2000   3             ZERO              A                                INITALIZE (A)
002 U 5528 002C   3    PRLOOP   B     A     F                 WPF                  (A) TO PAGE REG (
    L 7115 201F   4             ADD   A     BG    A           ONE                  INCREMENT A
003 U 5F07 0000   3             A     A           F                                SETUP ADDRESS
    L 5010 E015   5             A.B   A     BG                ONEK      ZL         JPR NEXT IF THRU
004 U 9828 2005   3   +         ZERO        F                 START    J           RE-INITALIZE FLAG
    L 9800 2002   3                                           PRLOOP   J           BACK TO LOOP

                      *

005 U D800 1000   3   +START                                  N=0                  INDEX TO F2

                      *

                                END
HEX   HEX      CYCLE   LABEL    F     A     B     D     S     C        MT      COMMENTS
MML   IMAGE    COUNT

*************** NO ERRORS                          ***************
     MICRO MEMORY MAP FOR SETPAGE

     UNUSED  0005 L   00FF L     501
     CROSS REFERENCE LISTING FOR SETPAGE
NAME       TYPE/VALUE/PAGE
BEGIN      MA-C/0000 /  1
ONEK       MA-E/0015 /  1    BTU/0003L/  1
ONE        MA-E/001F /  1    BTU/0002L/  1
PRLOOP     MA-C/0002 /  1   JJMP/0004L/  1
START      MA-C/0005 /  1   JUMP/0004 /  1
      STOP  FINIS
        .562 CP SECONDS EXECUTION TIME
COMMAND- files.
--LOCAL FILES--
 *DEMO      *MASS     $OUTPUT   *TAPE55    SIMFIL
  PUNCH      TAPE54
```

Figure 6-1. Assembly Of DEMO Program Via Terminal

```
COMMAND- attach,mpsim,id=colt.  get the micro simulator.
 PFN IS
 MPSIM
 PF CYCLE NO. = 052
COMMAND- mpsim.   run the micro simulator.
LINE LEN NOT =80
COMMAND- screen,80.  oops, MPSIM needs 80 col screen.
COMMAND- mpsim.      run the micro simulator again.
MPP/MPXX(AMI) SIMULATOR OF 03/22/76
FIELD LEN   0200673
?load
FIELD LEN   0454768
FIELD LEN   0211113
?run,0u
     0.168 000U 000L A =00000000 Q =00000000 X =00000000 F =00000000 P =00000000
                     I =00000000 N =00000000 K =00000000 RTJ00000000 A*=00000000
                     X*=00000000 Q*=00000000 SM100000000 SM200000000 M1=00000000
                     M2=00000000 IN100000000 IN200000000 Y =00000000 D =00000000
                     RS=00000000 RD=00000000
     0.336 000L 001U
     0.503 001U 001L S1=00000000 S2=00000000 ALU00000000 WSR00000000
     0.671 001L 002U
     0.839 002U 002L S1=00000000 S2=00000000 ALU00000000 WPF00000000
     1.063 002L 003U S1=00000000 S2=00000001 A =00000001
     1.231 003U 003L S1=00000001 F =00000001
?r
     1.455 003L 004L S1=00000001 S2=00000400 ALU00000000
     1.623 004L 002U
     1.791 002U 002L S1=00000001 S2=00000001 ALU00000001 WPF00000001
     2.015 002L 003U S1=00000001 S2=00000001 A =00000002
     2.183 003U 003L S1=00000002 F =00000002
     2.407 003L 004L S1=00000002 S2=00000400 ALU00000000
     2.575 004L 002U
     2.743 002U 002L S1=00000002 S2=00000002 ALU00000002 WPF00000002
     2.967 002L 003U S1=00000002 S2=00000001 A =00000003
     3.135 003U 003L S1=00000003 F =00000003
     3.359 003L 004L S1=00000003 S2=00000400 ALU00000000
?trace,show
  REGS  FILES  READS  WRITES  TIME  LIM1=000U BEG-MM  LIM2=200U END-MM
?bkp,3u
?bkp,show
       003U
?r
     3.527 004L 002U
     3.695 002U 002L S1=00000003 S2=00000003 ALU00000003 WPF00000003
BKPT ENCOUNTERED
     3.919 002L 003U S1=00000003 S2=00000001 ALU00000004 A =00000004 Q =00000000
                     X =00000000 F =00000003 P =00000000 I =00000000 N =00000000
                     K =00000000 RTJ00000000 A*=00000000 X*=00000000 Q*=00000000
                     SM100000000 SM200000000 M1=00000000 M2=00000000 IN100000000
                     IN200000000 Y =00000000 D =00000000 RS=00000000 RD=00000000
?r
     4.087 003U 003L S1=00000004 F =00000004
     4.311 003L 004L S1=00000004 S2=00000400 ALU00000000
     4.479 004L 002U
     4.647 002U 002L S1=00000004 S2=00000004 ALU00000004 WPF00000004
BKPT ENCOUNTERED
     4.871 002L 003U S1=00000004 S2=00000001 ALU00000005 A =00000005 Q =00000000
                     X =00000000 F =00000004 P =00000000 I =00000000 N =00000000
                     K =00000000 RTJ00000000 A*=00000000 X*=00000000 Q*=00000000
                     SM100000000 SM200000000 M1=00000000 M2=00000000 IN100000000
                     IN200000000 Y =00000000 D =00000000 RS=00000000 RD=00000000
?dump,pf,0,8
000  00000000 00000401 00000402 00000003 00000404 00000000 00000000 00000000
?end
COMMAND-
```

Figure 6-2. Interactive Execution Of The DEMO Program By The Simulator

MASS CONTROL CARD A

The MASS control card causes the MASS Assembler to be loaded and executed under control of the MPX operating system. This appendix describes the parameter options available to the programmer.

The card format is as follows:

```
*MASS(I=u,L=u,P=u,X=u)
```

Parameter definitions:

- I=u

  Source input is from logical unit or file u. If =u is absent, standard output is assumed. Input must be specified.

- L=u

  Assembly listing is on logical unit or file u. If =u is absent, standard output will be used. If this parameter is absent, no listing is generated; however, a listing of error lines will be output to standard output.

- P=u

  Deadstart binary object output is on logical unit or file u. If =u is absent, standard punch is assumed. If P is absent, no binary output is generated.

- X=u

  Pure binary output is put on logical unit or file u. If =u is absent, standard load and go file is assumed. If X is absent, no binary is generated. u, if present, must specify a disk file or magnetic tape unit.

EXECUTION ON A CYBER 70/70L/6000 SERIES COMPUTER

The following program call card causes execution of the MASS Assembler on a CYBER system under NOS or NOS/BE. It assumes the MASS Assembler is part of the system library.

```
MASS,p1,p2,p3,p4.
```

p_1	is the name of the file on which the microprogram source resides. The default is INPUT.
p_2	is the name of the file on which the assembler writes the source listing. The default is OUTPUT.
p_3	is the name of the file on which the assembler writes the object output. The default is PUNCH.
p_4	is the name of the file on which the assembler writes the simulation object output. The default is SIMFIL.

Example:

For source cards in the job deck, punch on file DWD, no simulation object output, standard listing, the program call card is:

```
MASS,,,DWD.
```

FORMAT OF DEADSTART OUTPUT DECK B

The deadstart deck contains both the data to be loaded into micromemory and control characters to direct the loading. The data is binary generated by MASS of a source program and is grouped 4, 32 bit words per card and punched in every other column in hollerith.

The control characters of the deadstart deck are directed to the Function Control Register (FCR), which has access to the CPU similar to the way switches on a conventional panel have access to the CPU. Table B-1 shows the control characters that are issued to the FCR to initialize the files, the registers and micromemory. See the BASIC SYSTEM CONTROLLER MODEL 65109 HARDWARE REFERENCE MANUAL for additional and more detailed information on the FCR and the operation of the panel interface.

The typical load sequence to enter a deadstart deck into the MP32 is to put the deadstart deck into the card reader, do a Master Clear and begin loading by enabling bit 4 of SM2 (usually connected to a switch).

TABLE B-1. DEADSTART DECK FORMAT

TYPE	CARDS	MEANING
1	K1008900G	Set up FCR register. Select N register for display 0. Select FCR for display 1. Select MICRO mode. Suppress console transmit. Enable micromemory write.
2	L0000G	Clear N register.
3	J02G	Select K register for display 0.
4	L0000G	Clear K register.
5	J0CG	Select micromemory for display 0.
6	L	Begin load of micromemory.
7	Data	Micromemory data in blocks of 256 32-bit micro-instructions. Each instruction is terminated with a G. The number of instructions per card depends on the spacing.
8	J01G	Select N register for display 0.
9	LXXXXG	Set N register to the number of 256 instruction blocks loaded.

Card types 5 through 9 are repeated until all full blocks are output. Data for the remaining partial block is then punched using cards 5 through 9, where card type 7 will contain less than 256 instructions.

10	K9A088000G	Set up new FCR register. After all data has been punched: Select RTJ register in display 0. Select SM2 register in display 1. Select MICRO mode. Suppress console transmit. Disable micromemory write.
11	K00000000G	Clear SM2. This card stops the deadstart hardware from reading more cards. Even though the MP16 has only 16 bits in the SM2 register, 32 bits of zeros (eight characters) are punched on the card to ensure total clearing of SM2 in case the machine is an MP32.

DEFAULT CODES GENERATED FOR BLANK FIELDS C

Each field of the assembly input may be left blank, with the following values substituted by MASS:

Field	Default Value	
F	A	If A field contains a mnemonic and B field blank.
	B	If A field is blank and B field contains a mnemonic.
	ZERO	If both A and B fields are blank.
A	X	
B	X	
D	NOP	
S	NOP	
C	00	
M	S	If C field is not K= or N=.
	00	If C field is K= or N=.
T	Selected to provide correct sequencing or jumping.	
	*L	If upper instruction and M field is S or blank.
	U	If lower instruction and M field is S or blank.
	U	If M field is R.
	U	If M field is J and C field is constant.
	U	If M field is J and C field is a symbol with upper quality.
	L	If M field is J and C field is a symbol with lower quality.

SELECTING NONCONFLICTING MNEMONICS D

The proper selection of a mnemonic for a field depends on the mnemonics used in the other fields (Table D-1. lists permissible combinations or codes). This is an inherent characteristic of the MP and is a result of maximizing the information content in the instruction repertoire. This means that the most frequently used operands require less space than the less frequently used operands.

TABLE D-1. LEGAL F, A, B, D AND S COMBINATIONS

F FIELD	A FIELD	B FIELD	D FIELD	S FIELD
Arithmetic or logical	A	B	D	Unused
	A'	B	D	AP
	A	B'	D	BP
	A	B	D'	DP
	A'	B	D'	APDP
	A	B	D"	DPP
	A	B	DD	DD
Shifts or scale	b	b	b	Unused
	b	b	NOP	Unused

Where:	b	is a blank field
	A, A', B, B', D, D", DD	are types of A, B and D field codes
	AP, BP, DP, APDP, DPP, DD	are values supplied by the assembler

Each instruction also consists of a maximum of four independent and concurrently executed functional operations; this further reduces the effective instruction execution time. Their associated fields are also dual, in the sense that they can be used to specify less frequently used operands for other fields. The assembler flags ambiguously coded instructions with a diagnostic. See Figure D-1 for examples of conflicting mnemonics.

```
HEX  HEX        CYCLE  LABEL   F     A     B     D     S      C        MT    COMMENTS            DIAGNOSTICS
MML  IMAGE      COUNT

                       ****************************************************************
                       *       EXAMPLES OF CONFLICTING MNEMONIC SELECTION AND ASSEMBLER ERROR CODES *
                       *       BLANK FIELDS ARE AVALIABLE FOR USAGE                      *
                       *       A B D AND S FIELD CONFLICTS                               *
                       ****************************************************************

059 U 7042 0300   5    *       ADD   P     F2    I     HALT                  LEGAL
    L 7042 2700   6            ADD   M1    F2    I     HALT                  ILLEGAL             SAS
05A U 7042 0700   6            ADD   M1    F2    I                           LEGAL
    L 5F0E 2800   4            A     A     CRTJ  X                           LEGAL
05B U 5F0E 0800   4            A     A     CRTJ  X     L8EA                  ILLEGAL             SAS
    L 5F1C 2900   3            A     A     X     M1                          LEGAL
05C U 5F1C 0900   3            A     A     X     M1    GATEI                 ILLEGAL             SAS
    L 5C25 2A00   4            A.B   SM1   Q     SM1                         LEGAL
05D U 5C25 0A00   4            A.B   SM1   Q     SM1   RTJ                   ILLEGAL             SAS

                       ****************************************************************
                       *       T AND C FIELD CONFLICTS                                   *
                       ****************************************************************

05D L D800 3021   3                                           N=33     U     LEGAL

05E U D800 3021   3                                           N=33     LQL   ILLEGAL             TIL
    L 5800 2080   3    LAB1                                   256      LQL   LEGAL

05F U 9800 205E   3                                           LAB1     JLQL  ILLEGAL             TIL

                       ****************************************************************
                       *       B AND C FIELD CONFLICTS                                   *
                       ****************************************************************

05F L 5E08 2008   3                        N                  31             ILLEGAL             CBC
060 U 5E08 0008   3                        N                  10B            LEGAL

                       ****************************************************************
                       *       SHIFT FUNCTION AND A OR B FIELD CONFLICT                  *
                       ****************************************************************

060 L 7C90 2000 5+N            ALEA  A                                       ILLEGAL             ASF
061 U 7CA0 0000 5+N            ALEA        Q                                 ILLEGAL             BSF

HEX  HEX        CYCLE  LABEL   F     A     B     D     S      C        MT    COMMENTS            DIAGNOSTICS
MML  IMAGE      COUNT
```

Figure D-1. Examples of Conflicting Mnemonic Selection and Assembler Error Codes

MASS OUTPUT LISTING E

The output listing produced by MP MASS consists of a side-by-side listing of micromemory location, machine language, source card image, error codes and sequence number.

The information is assigned to the following columns of the listing.

Column	Contents and Meaning
1	Standard print format control.
2-4 (MML)	Micromemory location of assembled instruction, or value specified in EQU or ORG command. This is a three-digit hexadecimal number, with the first digit specifying micromemory page and the next two digits specifying location within the page.
5	Micromemory overlap indicator. This column is blank if micromemory location is used only once in microprogram. This column contains 1 if micromemory location had contained information previous to assembly of the line of code causing overlap. A count of the number of micromemory overlaps is included on the last page of assembly listing for information.
6	Upper/lower indicator. U specifies that micro-instruction is assembled to the upper micro-instruction of location specified in columns 2 through 4. L specifies that the microinstruction is assembled to the lower microinstruction.

Column	Contents and Meaning
8-16 (IMAGE)	Assembled microinstruction in hexadecimal code.
19 (CYCLE COUNT)	Timing information. MASS calculates execution time for each microinstruction and indicates time as the number of minor cycles to execute the statement.
24-103	Source card image.
104-113	Sequence number supplied by COSY package (if applicable.
115	Diagnostic error codes generated by MASS for source statement.

MASS MAP

MASS will generate a micromemory assignment map if the M(AP) option on the OPTN card is specified. The map will show blocks of 256 micromemory words (1 micromemory page) where each micromemory word can hold 2 instructions (an upper and a lower instruction). Asterisks (*) will be printed out for each micromemory location assigned and be put in either the upper or lower position. Remember also that unless a NOSUM option is specified, the checksum of micromemory is placed in the lowest vacant memory location, but would not show on the map.

MASS CROSS-REFERENCE

MASS will generate a cross-reference table if the C(ROSS) option on the OPTN card is specified. The cross-reference listing will contain an alphabetic listing of all the labels that occurred within the program. Each label will have a set of entries describing what its use was, the value associated with it and the page that it appeared on. The label use is shown as follows:

MA-C	MICRO ADDRESS	-	CODE
MA-D	MICRO ADDRESS	-	DATA
MA-O	MICRO ADDRESS	-	ORG
MA-E	MICRO ADDRESS	-	EQUATE
MA-S	MICRO ADDRESS	-	SET
F1-C	FILE 1	-	CODE
F1-D	FILE 1	-	DATA
F1-O	FILE 1	-	ORG
F1-E	FILE 1	-	EQUATE
F1-S	FILE 1	-	SET
F2-C	FILE 2	-	CODE
F2-D	FILE 2	-	DATA
F2-O	FILE 2	-	ORG
F2-E	FILE 2	-	EQUATE
F2-S	FILE 2	-	SET
COND	CONDITIONAL		
UNKN	UNKNOWN		
SPCE	SPACE		
SFLD	S FIELD		
JUMP	JUMP		
K=	K REGISTER =		
N=	N REGISTER =		
CFLD	C FIELD		
BTU	BIT TEST UPPER		
BG	BIT GENERATOR		
CLO	CLEAR LOW ORDER		

SAMPLE LISTING INCLUDING MAP AND CROSS REFERENCE

The following notes are keyed to the fields (columns) of the sample listing:

Note	Contents
A	Assembler identification; host machine type (CYBER 32/6000/CYBER 70L/CYBER 170), assembler version number.
B	Value (in hexadecimal) of the expression on an EQU or an ORG card.
C	Micromemory location (in hexadecimal) assigned to this microinstruction. The HEX MML column contains three digits. The first is the page address; the second two are the micromemory address within the page. The next column specifies U for upper-half word or L for lower-half word.
D	The contents in hexadecimal of the location.
E	A number indicating the length of time in cycles required to execute this instruction. (See Basic System Controller Model 65109 Hardware Reference Manual.)
F	Card image. The fields on the card are indicated by the notations: LABEL (location), F, A, B, D, S, C, MT and COMMENTS.
G	If the assembler detects an error in the information coded on the source card, the error code(s) is (are) printed on this part of the listing. There is room to print up to four error codes on the listing.

```
A CYBER/MP-60 MICRO ASSEMBLER PSD VERSION 1.0 EXAMPLE       PROGRAM WITH MAP AND CROSS REFERENCE      08.21.37. 05/24/76

 HEX    HEX       CYCLE  LABEL    F     A     B     D     S     C        MT    COMMENTS                     DIAGNOSTICS
 MML    IMAGE     COUNT
                                                                                                                 ^
                                IDENT EXAMPLE       PROGRAM WITH MAP AND CROSS REFERENCE                         |
                                OPTN  CROSS,DEAD,MAP,NOSUM,SIMU,L,I                                              |
                                                                                                                 |
                         ***********************************************************************                |
                         *      MULTIPLY A BY X, PRODUCT TO AQ.                                 *               |
                         *      MULTIPLY WORKS ON POSITIVE NUMBERS SO PROVIDE LEAD IN TO        *               |
                         *      CALCULATE SIGN OF NEGATIVE INPUTS AND CORRECT AT END.           *               |
                         *      CODE IS WRITT N FOR ONES COMPLEMENT INPUT NUMBERS.              *               |
                         ***********************************************************************                |
                                                                                                                 |
                                CMP1                                     INDICATE ONES COMPLEMENT                |
  030                    START  EQU   30X                                                                        G
B       D         E      <------------------------------ F ------------------------------->                     |
  030                           ORG   START                                                                      |
                                                                                                                 |
C 030 U 0F03 C000  4     MULT   A     A           Q           K=0      SNU   CHECK SIGN                         |
                                                                                                                 |
  031 U 4A23 0045  3     +      -B          Q     Q           INCK           COMP Q FOR POSITIVE                |
      L 5EC0 C000  4            A     X                                SNU   CHECK SIGN OF X                    |
                                                                                                                 |
  032 U 40C6 0044  3     +      -A    X           X           DECK           GET POS X AND SIGN IN K            |
      L 0805 301F  3            ZERO              A           N=31           CLEAR A, SET TIMES COUNT           |
  033 U 5800 20F2  3                                          RQROE    SLQL  MAKE FIRST STEP TEST               |
                                                                                                                 |
  034 U 5F05 22F2  3     +      A     A           A     RPT   RQROE    LQL   MULTIPLY ITERATION LOOP            |
      L 711D 22F2  5     -      ADD   A     X     A     RPT   RQROE    LQL   MULTIPLY ITERATION LOOP            |
                                                                                                                 |
  035 U 4105 6000  3            -A    A           A                    SKZU  EXIT ON POS SIGN TEST REM          |
                                                                                                                 |
  036 U 8105 2037  3     +      -A    A           A           EXIT     J     POS RESULT RECOMP A                |
      L 8A23 2037  3     -      -B          Q     Q           EXIT     J     NEG RESULT, COMP Q                 |
  037                    EXIT   SET   *                                      NEXT INSTRUCTION                   v
                                END

 HEX    HEX       CYCLE  LABEL    F     A     B     D     S     C        MT    COMMENTS                     DIAGNOSTICS
 MML    IMAGE     COUNT

  ************************************** NO ERRORS          **************************************
```

Figure F-1. Sample Listing

```
     CROSS REFERENCE LISTING FOR EXAMPLE          PROGRAM WITH MAP AND CROSS REFERENCE  11.36.12. 77/08/01.PAGE    2

NAME      TYPE/VALUE/PAGE
EXIT      MA-S/0037 /  1   JUMP/0036 /  1   JUMP/0036L/  1
MULT      MA-C/0030 /  1
START     MA-E/0030 /  1   MA-O/0000 /  1




     MICRO MEMORY MAP FOR EXAMPLE          PROGRAM WITH MAP AND CROSS REFERENCE  11.36.12. 77/08/01.PAGE    2

PAGE

        0               1               2               3               4               5               6               7
        0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
      U                                                 *******
      L                                                  ** * *
0
      U
      L
        8               9               A               B               C               D               E               F
        0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
```

Figure F-2. Map And Cross Reference

ASSEMBLY DIAGNOSTICS

The assembler prints error codes to flag and diagnose incorrect assembly statements. When a statement is in error, one to four error codes are listed to the right of the statement describing the problem. Figure G-1. is an example of a listing containing error codes.

If any errors occur on a page, the last line printed on the page and all subsequent pages of coding contains the message "PREVIOUS ERROR(S) ON PAGE n ", where n is the page number of the most recent page to contain errors. The final coding page contains the message " n LINES CONTAIN ERROR(S)" where n is the total number of all lines containing errors, or "NO ERRORS" if no errors were detected.

MASS ASSEMBLY ERRORS

The following error codes are generated by MASS:

Error Code	Meaning
ASF	Mnemonic must not be included in A field because A field is set by F field (shift or scale).
AU	A field mnemonic is undefined.
BSF	Mnemonic must not be included in B field because B field is set by F field (shift or scale).
BU	B field mnemonic is undefined.
CBC	C, B field conflict. B field is N, K, ZERO, or N,K; C field must match B field requirement for bits 28 and 29.

Error Code	Meaning
CER	Constant error. Illegal character in HEX, DEC or OCT pseudo instruction.
CMC	C and M field conflict. Command desired in C field cannot be expressed with instruction sequencing mode specified in M field.
CRP	Cannot reach page requested. S field must be left blank so page jump may be encoded in instruction.
CSC	C and S field conflict. The C field requires the S field which has already been set by an A, B, D or S code or a combination.
CU	C field contains undefined symbol.
DDS	Symbol has been defined more than once, or with other than SET pseudo.
DSC	D field requires use of S field, which is already set due to a mnemonic in A or B field.
DU	D field mnemonic is undefined.
ELU	Location field in EQU or SET pseudo instruction is blank.
EXP	Illegal character, symbol or operator in expression, or missing operator.

Error Code	Meaning
FBG	Incorrect number of bits specified in VFD.
FOV	This file location is out of range for the current file or was previously defined. If previously defined, this instruction does not replace the previous one.
FU	F field mnemonic is undefined.
MEM	Micromemory overflow.
MOV	Micromemory overlap. This micromemory location was defined previously. This instruction replaces the previous one. This error indication appears only if binary output has been selected.
MU	M field is undefined.
OPN	Illegal option specified on the OPTN card.
SAB	Both A and B fields require use of S field.
SAS	Mnemonic coded for S field cannot be assembled because S field is already set by A, B or D field.
SU	S field contains undefined mnemonic or symbol.
TIL	T field code is illegal for jump command or for use in K= or N= command.
TNS	T field must be specified for M field mnemonic R.
TU	T field mnemonic is undefined.

OTHER DIAGNOSTICS AND INFORMATIVE MESSAGES

The following messages may appear in the job's dayfile of a CYBER machine:

```
REQUESTED FL EXCEEDS USER FL
BINARY SUPPRESSED -- INCREASE FL =42000 + 1000 * HIGHEST PAGE
```

The amount of central memory needed for MASS to produce binary output has exceeded the amount that can be requested by the program. Set the CM parameter on the job card to the value of 42000, plus the result of 10008 times the sum of the highest micromemory page number (of 512 words) plus one.

```
HEX   HEX        CYCLE   LABEL    F     A     B     D     S     C        MT    COMMENTS                             DIAGNOSTICS
MML   IMAGE      COUNT

                                IDENT DIAGNOSTIC EXAMPLE FOR THE MP ASSEMBLER

                      **********************************************************************************
                      *                                                                                *
                      *         DIAGNOSTIC EXAMPLE                                                     *
                      *                                                                                *
                      **********************************************************************************

                                OPTN  DEAD
FFF                    TROUBLE  EQU                                          UNDEFINED EQUATE                     EXP
000                             EQU   5                                      UNDEFINED LABEL                      ELU

00A                             ORG   AX

00A U 5800 0000  3     SYM                                                   FIRST SYMBOL DUPLICATE
    L 5800 2000  3                                NOP
00B U 5800 0000  3     SYM                                                   DUPLICATE SYMBOL
    L 9800 200A  3                                            SYM       J    USE OF DUPLICATE SYMBOL              DDS
00C U 5800 0000  3     LOC                              NOP                  GOOD STATEMENT
    L 5800 2000  3              MP                                           UNDEFINED FUNCTION                   FU
00D U 5800 0000  3                    Q                                      CAN≠T USE Q AS A INPUT               AU
    L 5800 2000  3                          A                                CAN≠T USE A AS B INPUT               BU
00E U 5800 0000  3                                XM1                        UNDEFINED DESTINATION                DU
    L 5800 2000  3                                      RAP                  UNDEFINED S CODE                     SU
00F U 5800 0000  3                                            JOE            UNDEFINED C FIELD                    CU
    L D7FF 3FFF  3                                                      X    ILLEGAL M CODE                       MU
010 U 5800 0000  3                                                      BTL  ILLEGAL T CODE                       TU
    L 7C80 2000 5+N             ALOE  X                                      A,B MUST BE BLANK                    ASF

011 U 7C98 0000 5+N             ALOE        X                                A,B MUST BE BLANK                    BSF
    L 5E30 2700  4                    SM1   INTA                             A,B CAN≠T BOTH BE PRIME              SAB
012 U 5400 0700  4                    SM1               HALT                 A≠ S CONFLICT                        SAS
    L 5E10 2800  4                          INRD        L8EA                 B≠ S CONFLICT                        SAS
013 U 5801 0300  3                                IOA   GATEI                D≠ S CONFLICT                        SAS
    L 5E08 2008  3                          N                 DECK           B AND C CONFLICT                     CBC
014 U D800 F022  3                                            N=35      R    C AND M FIELD CONFLICT               CMC
    L 9800 2E00  3                                      RTJ   100X      J    NEED S FIELD FOR PAGE                CRP
015 U D800 300F  3                                            N=15      LQL  ILLEGAL T FOR N=                     TIL
    L 9800 200C  3                                            LOC      JLQL  ILLEGAL T FOR JUMP                   TIL

016 U 9800 200C  3                                            LOC       JU   LEGAL T FOR JUMP

00A                             ORG   10

00A U 565D 0000  3              A+B   P     X     A                          ORG OVER EXISTING CODE               MOV

388

                                END
HEX   HEX        CYCLE   LABEL    F     A     B     D     S     C        MT    COMMENTS                             DIAGNOSTICS
MML   IMAGE      COUNT

************************************ PREVIOUS ERROR(S) ON PAGE    0 ************************************
************************************   23 LINES CONTAIN ERROR(S)    ************************************
```

Figure G-1. Assembler Diagnostics Example

MICROINSTRUCTION TIMING H

The basic CPU microinstruction execution time is 168 nanoseconds. Some microinstructions have longer execution times to allow certain operations to be completed. The microinstructions have been grouped according to execution times as requiring 3, 4, 5, 6, 5 + n and 9 56-nanosecond cycles.

The classification of microinstructions is shown in Figure H-1. This figure provides execution time by types for all legal combinations of microcommands which extend the basic cycle time.

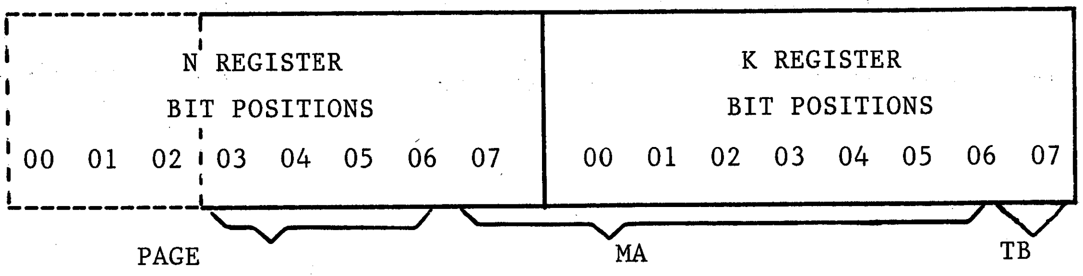
Exceptions to the execution time, as listed in Figure H-1, are:

- The combination of one's complement arithmetic with ADD+ or ADD+T arithmetic operation requires four 56-nanosecond cycles.
- The MMU command is included under read/write micromemory operand commands; therefore, it requires nine, rather than four, cycles.

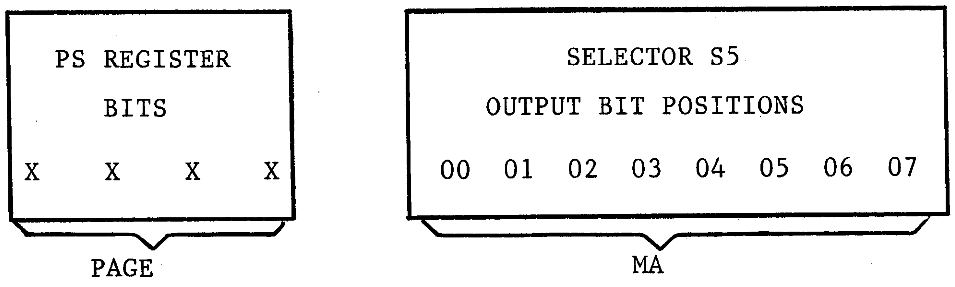
Analysis of a microprogram for execution time starts by classifying each of the microinstructions. This is done by using the microinstruction classification table or by examining the assembler output listing.

Two addressing modes are available for operand references via status mode bit 113 (SM113) as follows:

- SM113 = 0 - The contents of the combined N/K registers are used to reference operands as indicated. The least significant bit of K (bit 07) determines which 32-bit half-word is referenced via T' code 010.



- SM113 = 1 - An MA transform via S5 determines the 64-bit word. The 32-bit half-word to be referenced must be selected by a T field test.



XXXX = Page in which referencing microinstruction resides.

READ/WRITE MICROMEMORY OPERANDS	SHIFT OR SCALE	TMA, TMAK, GITMAK, ZL, NU, COL,	ADD OR SUBTRACT OR	A', B', NU, ZL, COL,	ONE'S COMPLEMENT	INSTRUCTION CYCLES
N	N	N	N	N	N	3
N	N	N	N	N	Y	3
N	N	N	N	Y	N	4
N	N	N	N	Y	Y	4
N	N	N	Y	N	N	4
N	N	N	Y	N	Y	5
N	N	N	Y	Y	N	5
N	N	N	Y	Y	Y	6
N	N	Y	N	N	N	5
N	N	Y	N	N	Y	5
N	N	Y	N	Y	N	5
N	N	Y	N	Y	Y	5
N	N	Y	Y	N	N	5
N	N	Y	Y	N	Y	5
N	N	Y	Y	Y	N	6
N	N	Y	Y	Y	Y	6
N	Y	X	X	X	X	5 + n where n = number of shifts
Y	X	X	X	X	X	9

N = No
Y = Yes
X = Don't Care

Figure H-1. Microinstruction Classification

GLOSSARY

Item	Definition
A	A register. General purpose register that may be shifted individually or in conjunction with the Q register.
A field	Field of MASS input form for specifying input to selector 1.
ALU	Arithmetic logic unit. Capable of performing addition and subtraction, as well as logical operations on output of selector 1 and selector 2.
A/Q	A register, Q register or the combined A/Q register. The A and Q registers are shift registers.
A source	Register or data item selected as output of selector 1.
B field	Field of MASS input form for specifying input to selector 2.
BG	Bit generator. Specialized circuitry that generates a value consisting of single bit set to 1 with rest of bits set to 0.
B source	Register or data item selected as output of selector 2.
Bit test	Output of selector 7 (located on transform board), which allows any specified condition or bits to be used for selection of next upper or lower microinstruction.
C field	Field of MASS input form for specifying instructions, constants or jump address.
Decr	Decrement to reduce contents of a register by one. May affect N or K register.

Item	Definition
F	F register. General purpose, word length register. Note that the contents of F will be destroyed when File 1, 2 is accessed.
F'	Holds data for File 3 write.
F field	Field of MASS input form for specifying ALU operation or shift or scale operation.
F1	File 1. 256-word register file, which is addressed by contents of K register.
F2	File 2. 32-word register file, which is addressed by contents of N register.
F3	File 3. High-speed micromemory which is addressed by the K' or J register.
I	I register. General purpose register; is loaded with output of selector 1.
Incr	Increment to increase contents of a register by one. May affect N or K register.
INP	Input. One of possible paths of transferring information from external devices to selector 4 and thus to selector 2 input to ALU.
Input bus	Means of transferring information to selector 1 or selector 2.
Interrupt	A condition that is tested by the microinstruction by use of interrupt system, consisting of mask registers and interrupt address generator.
IXT	Temporary holding register used to decode macroinstructions.

Item	Definition
J	J register. Used to address File 3.
K	K register. An 8-bit counter that may be set, incremented, decremented and tested for zero. Also used to address file 1.
K'	Contains image of the K register in the lower 8 bits and is also used to address File 3.
L	Holding register for constant increment for J.
L1	Shift input to a selector; specifies left shift one bit position to generate output.
L8	Shift input to a selector; specifies left shift eight bit positions to generate output.
LAi	Main memory lookahead register; i = 0 if DMA lookahead, 1 to 3 if CPU lookahead.
LHW	Shift input to a selector; specifies left shift a half word, end around, to generate output.
M field	Field of MASS input form for specifying mode of obtaining next microinstruction pair.
M1	Mask register. Word-sized register to control recognition of interrupts. An interrupt is recognized if its corresponding mask bit of M1 or M2 is set to 1.
M2	Mask register. Word-sized register to control recognition of interrupts. An interrupt is recognized if its corresponding mask bit of M1 or M2 is set to 1.
MA	Micromemory address -- used to address micromemory.

Item	Definition
MAC	Micromemory address counter. Counter that is always updated to contain address of next sequential microinstruction pair.
MIR	Microinstruction register. Register that holds current microinstruction being executed.
MM	Micromemory.
N	N register. An 8-bit counter that may be set, incremented, decremented and tested for zero. Used as shift and scale counter, used to address file 2 and used to control repeat operation.
One's Complement	The radix-minus-one complement in binary notation.
P	P register. General purpose register named P (not a program counter).
P-MA	Page-micromemory address register. 12-bit register that contains 4-bit page and 8-bit micromemory address of current microinstructions.
Page	Unit of micromemory containing 256 locations of 512 microstructions.
PS	Page number saved of the previous microinstruction executed.
PTR	Paper tape reader.
Q	Q register. General purpose register that may also be used in conjunction with A register in shifting.

Item	Definition
R1	Shift input to a selector; specifies right shift one bit position to generate output.
RDi	Read from main memory holding register; i = register number from 1 to 3.
RTJ	Return register. 12-bit register used to hold a micromemory address for returning from micro-subroutine.
S field	Field of the MASS input form for specifying special operation taking place in parallel with other portions of microinstruction.
S1	Selector 1. Selects A input to ALU.
S2	Selector 2. Selects B input to ALU.
S3	Selector 3. Provides shifting of ALU output going to P, A, F, X, Q, F1 or F2 destinations.
S4	Selector 4. Selects one of four external inputs to selector 6.
S5	Selector 5. Used in performing MA transforms.
S6	Selector 6. Used in selecting source of next microinstruction.
S7	Selector 7. Selects one of the 16 possible bit tests.
S8	Selector 8. Used to select values in K and N transform.
S9	Selector 9. Selects status/mode or mask register for input to selector 1.

Item	Definition
SM1	Status/mode 1 register. Status and mode control.
SM2	Status/mode 2 register. Status and mode control.
SRj	State register; j = DMA state registers 0 to 3 and CPU state register.
SW	Console switches.
T field	Field of MASS input form for specifying selection criteria for next microinstruction.
TB	Test bit -- used to identify upper or lower of micro-instruction pair.
Transform	Facility provided by selector 5 to generate micromemory addresses and selector 8 to load K and N registers.
Two's Complement	The radix complement in binary notation.
WDi	Write-to-main-memory holding register; i = register number from 1 to 3.
X	X register. General purpose word length register.

COMMENT SHEET

TITLE: MASS/MPSIM REFERENCE MANUAL

PUBLICATION NO. 17328900-02 REVISION

This form is not intended to be used as an order blank. Control Data Corporation solicits your comments about this manual with a view to improving its usefulness in later editions.

Applications for which you use this manual.

Do you find it adequate for your purpose?

What improvements to this manual do you recommend to better serve your purpose?

Note specific errors discovered (please include page number reference).

General comments:

CUT ON THIS LINE

FROM NAME: __________ POSITION: __________

COMPANY NAME: __________

ADDRESS: __________

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

FOLD ON DOTTED LINES AND STAPLE

STAPLE

STAPLE

FOLD

FOLD

FIRST CLASS
PERMIT NO. 8241

MINNEAPOLIS, MINN.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION

Publications and Graphics Division

215 Moffett Park Drive

Sunnyvale, California 94086

CUT ON THIS LINE

FOLD

FOLD

STAPLE

STAPLE