SECTION II INTRODUCTION

The standard software supplied with the FABRI-TEK MP12 consists of the following programs:

- Source Edit Utility
- Assembler
- Binary Loader
- Debugging Utility
- Processor Diagnostic
- Cross-Assembler

The software is distributed in object format on fan-fold paper tape except for the Cross-Assembler which is supplied in source-listing form. Source listings for all programs are provided in the deliverable MP12 documentation package.

MP12 software is designed for stand-alone operation with a teletype, Model 3320-3JA or equivalent. The minimum equipment configuration required to support operation with a teletype includes the following items:

ITEM	PART NUMBER
MP12 Processor with Teletype Interface	999-4090-01
Teletype Modification Kit	117-0320-00
Chassis Assembly	258-0192-00, or equivalent
Power Supply	261-0125-00, or equivalent
ASR 33 Teletype	supplied by user

The standard software also supports operation with the following devices:

- Keyboard/Printer
- High Speed Paper Tape Reader
- High Speed Paper Tape Punch

The programming required to operate these devices presupposes standard MP12 interfaces.

SOURCE EDIT UTILITY

The FABRI-TEK MP12 Source Edit Utility (Edit) facilitates the preparation and modification of symbolic assembly language source tapes. Edit is an interactive program which enables the user to perform the following functions by way of the teletype:

- Construct a symbolic source tape.
- Insert, delete, replace, and modify statements in an existing source program, and obtain a new source tape which incorporates the modifications.
- Obtain a statement-numbered listing of the program being edited.

Editing is performed to update an existing source program punched on paper tape, or to construct a new source tape. In the former case, editing is accomplished with reference to a statement-numbered listing of the program. This listing may be an assembly listing or a statement-numbered listing obtained from Edit.

Edit accepts two kinds of typeins from the teletype keyboard, Statements and Commands. Statements consist of a statement-number followed by the text of the statement. Commands consist of a mnemonic command code and command parameters. All typeins are terminated by typing a carriage-return, Ⓡ; Edit supplies a line-feed and the next statement or command may then be entered. If an error is discovered during a typein, typing a left-arrow deletes the previously typed character; typing ESC reinitializes the line typein.

STATEMENTS

All statements entered from the keyboard are placed into an internal edit buffer. Each statement entered is preceeded by a statement-number that specifies the relative order of the statement in relation to all other statements. Statement-numbers need not be consecutive and the sequence in which statements are entered is immaterial; however, statements are buffered in order of increasing statement number and are always listed and dumped in that order. Statement numbers which preceed each statement entered into Edit may take any of the following forms:

```
p
p.q
0.q
p.0
```

The terms p and q are integers in the range from 1 through 4095. The notation p.q is used to denote the qth statement inserted after statement p. As special cases, p and p.0 are the same, and the notation 0.q denotes the qth statement inserted before the first statement. For example, if statements 12 and 13 had been entered previously, and the user wanted to insert two additional statements between statements 12 and 13, he could type:

12.10 [Statement Text] ®
12.20 [Statement Text] ®

The statements 12 through 13 would then be ordered within the edit buffer as follows:

Statement 12
Statement 12.10
Statement 12.20
Statement 13

An additional statement could be inserted between statements 12.10 and 12.20 by typing:

12.15 [Statement Text] ®

The order of the statements would then be:

Statement 12
Statement 12.10
Statement 12.15
Statement 12.20
Statement 13

Typing a statement-number immediately followed by a carriage-return causes the corresponding statement to be deleted from the edit buffer. For example, typing

12.15 ®

would cause statement 12.15 to be deleted.

A statement within the edit buffer can be replaced by another statement by simply typing the same statement number followed by the text of the new statement. For example, if a statement numbered 12.10 had previously been entered into the edit buffer, typing

12.10 [New Statement Text]®

would cause the existing statement 12.10 to be deleted and replaced by the new statement 12.10.

A special edit feature enables statements to be entered without the need to type a statement number for each statement. This is accomplished by typing a second

carriage-return following the text of a statement. Edit types the next consecutive four-digit statement number and the user may then type the text of the statement.

Example 1, below, illustrates edit statement typeins and is referenced in examples 2 through 5 in the following section.

EXAMPLE 1, Statement Typeins

```
1 ICH,0
2 IØT    037/READ CHARACTER
3 JMP    .-1/
4 SNA    /SKIP IF NØT NULL
5 JMP    ICH+1/NULL, IGNØRE
6 JMP I  ICH/RETURN
5.10 AND    MASK/REMØVE PARITY BIT
8 MASK,0177/CHARACTER MASK
9 $/END
0.10/INPUT CHARACTER SUBRØUTINE
```

COMMANDS

Edit commands are implemented as single, double, or triple letter mnemonics followed by optional command parameters. All commands are terminated by typing a carriage return, ® . Commands may be typed at any time during the edit process in place of entering a source statement. If an error is detected, Edit outputs a question mark (?) and the command is ignored. Typeouts, resulting from a command, may be aborted by depressing the keyboard BREAK key.

In the following command descriptions, m and n represent statement numbers of the general form p.q, where p and q are integers in the range 0 through 4095. Square brackets ([]) enclose optional command parameters.

INITIALIZE

The result of the Initialize command clears the internal edit buffer in preparation for edit input. The format of the Initialize command is as follows:

I ®	Initialize Edit Buffer

LOAD

The Load command causes source statements to be loaded into the edit buffer from the paper tape reader. The format of the Load command is as follows:

Lm ®

The parameter m is the statement number to be assigned to the first source statement entered. Succeeding statements are assigned consecutive statement numbers and entered into the edit buffer. The load function is terminated when a $ statement is encountered; the $ statement is not entered into the edit buffer.

PRINT

The Print command causes specified statements within the internal edit buffer to be listed on the teletype printer. The following is the format of the Print command:

P[F] [m[,n]] ®   Print

The optional command letter F controls the format of the listing. For the P command, each statement is listed exactly as it was entered. For the PF command, each statement is listed in the following format:

1. The statement number is printed in print positions 1 through 9 in the format p.q.
2. If the first character of the statement is a slash (/), the statement is listed beginning in print position 11.
3. If a symbol consisting of no more than six characters followed by a comma or equal sign is present, the symbol is printed, left justified, in print positions 11 through 17. The remainder of the statement is listed beginning in print position 18.
4. If a slash (/) is encountered prior to print position 34, the slash and the remainder of the statement are listed beginning in print position 34.

The Print command is followed by the optional statement numbers m and n. If m and n are not specified, the entire edit buffer is listed. If m is specified, statement m is listed. If n is also specified, statements m through n, inclusive, are listed.

If multiple lines of output result from the Print command, the statements are listed in the following order:

1. All statements numbered O.p are listed first. Statement O.p precedes statement O.q if p<q.
2. All statements numbered p.q are listed after statement s and prior to statement t if s≤p<t. If q<r, then statement p.q is listed prior to statement p.r.

With reference to Example 1, the following examples illustrate the listings produced by the P and PF commands.

EXAMPLE 2, P Command

```
P5.10
0005.0010  AND   MASK/REMØVE PARITY BIT

P5,6
0005       JMP   ICH+1/NULL, IGNØRE
0005.0010  AND   MASK/REMØVE PARITY BIT
0006       JMP I ICH/RETURN

P
0000.0010 /INPUT CHARACTER SUBRØUTINE
0001       ICH,0
0002       IØT   037/READ CHARACTER
0003       JMP   .-1/
0004       SNA   /SKIP IF NØT NULL
0005       JMP   ICH+1/NULL, IGNØRE
0005.0010  AND   MASK/REMØVE PARITY BIT
0006       JMP I ICH/RETURN
0008       MASK,0177/CHARACTER MASK
0009       $/END
```

EXAMPLE 3, PF Command

```
PF2,4
0002             IØT    037      /READ CHARACTER
0003             JMP    .-1      /
0004             SNA             /SKIP IF NØT NULL

PF
0000.0010 /INPUT CHARACTER SUBRØUTINE
0001      ICH,   0
0002             IØT    037      /READ CHARACTER
0003             JMP    .-1      /
0004             SNA             /SKIP IF NØT NULL
0005             JMP    ICH+1    /NULL, IGNØRE
0005.0010        AND    MASK     /REMØVE PARITY BIT
0006             JMP I  ICH      /RETURN
0008      MASK,  0177            /CHARACTER MASK
0009             $               /END
```

DUMP

The Dump command causes specified statements within the internal edit buffer to be output on the teletype punch. The format of the Dump command is as follows:

D[m[,n]]® Dump

The Dump command parameters are identical to the Print command parameters. The specified statements are sorted and dumped in the same manner as for Print except that statement-numbers are not output. The dump is preceeded and followed by 72 frames of blank tape. The Punch Control switch must be set to the ON position immediately after the Dump command is issued and returned to the OFF position as soon as the dump is complete.

EDIT-PRINT

The Edit-Print command causes the contents of the internal edit buffer to be merged with the statements on an existing source tape and a listing produced which represents the edited source program. The format of the Edit-Print command is indicated below:

EP[F] [P_1[,P_2[,... [,P_k]]] ...] ® Edit-Print

The optional command letter F controls the format of the listing in the same manner as indicated for the PF command. The optional parameters p_1 through p_k are represented in either of the forms m or m-n, where m and n are integers in the range 1 through 4095 representing statement-numbers corresponding to statements on the source tape to be edited. The appearance of a parameter of the form m specifies that the mth statement on the source tape is to be deleted and is not to appear on the listing. A parameter of the form m-n specifies that statements m through n, inclusive, are to be deleted. Examples of Edit-Print commands are shown below.

```
EP3,15-18,23,27-31

EPF2-5,12,23,27-31
```

Prior to issuing an Edit-Print command, the following procedures should be followed:

1. Set the Reader Control switch to the FREE position.
2. Position the source tape to be edited in the reader with blank tape under the read heads.
3. Enter the Edit-Print command.
4. Set the Reader Control switch to the START position.

Edit reads the source tape and produces a listing which is structured as follows:

1. If the edit buffer contains a statement numbered p, then this statement is listed in place of the pth statement on the source tape. The pth statement on the source tape is not listed.
2. All statements within the edit buffer which are numbered p.q are listed prior to the (p + 1)th statement on the source tape. If q<r, then statement p.q is listed prior to statement p.r.
3. Consecutive statement numbers are printed for each statement as indicated below:

```
0001   [First Statement]
0002   [Second Statement]
0003   [Third Statement]
etc.
```

The listing is terminated when a $ statement is encountered on the source tape. The following examples illustrate the listings produced by Edit-Print commands.

EXAMPLE 4, Sample EP Listing

```
0001 /INPUT CHARACTER SUBRØUTINE
0002  ICH,0
0003  IØT   037/READ CHARACTER
0004  JMP   .-1/
0005  SNA   /SKIP IF NØT NULL
0006  JMP   ICH+1/NULL, IGNØRE
0007  AND   MASK/REMØVE PARITY BIT
0008  JMP I ICH/RETURN
0009  MASK,0177/CHARACTER MASK
0010  $/END
```

EXAMPLE 5, Sample EPF Listing

```
0001 /INPUT CHARACTER SUBRØUTINE
0002 ICH,    0
0003         IØT    037        /READ CHARACTER
0004         JMP    .-1        /
0005         SNA               /SKIP IF NØT NULL
0006         JMP    ICH+1      /NULL, IGNØRE
0007         AND    MASK       /REMØVE PARITY BIT
0008         JMP I  ICH        /RETURN
0009 MASK,   0177              /CHARACTER MASK
0010         $                 /END
```

EDIT-DUMP

The Edit-Dump command causes a source tape to be punched which contains the same statements listed via EP, except that the statement-numbers are not output. The format of the Edit-Dump command is as follows:

ED[P_1[,P_2[,...[,P_k]]]...]® Edit-Dump

The optional parameters p_1 through pk are the same as those for Edit-Print. Prior to entering an Edit-Dump command, the following procedures should be followed:

1. Set the Reader Control switch to the FREE position.
2. Position the source tape to be edited in the reader with blank tape under the read heads.
3. Enter the Edit-Dump command.
4. Set the Punch Control switch to the ON position.
5. Set the Reader Control switch to the START position.

The dump is preceded by 72 frames of blank tape and terminated when a $ statement is encountered on the source tape.

STORAGE REQUIREMENTS

Edit resides in memory locations 0000 through 2166_8. Locations 2167_8 through 7577_8 are used for the edit buffer. In case buffer capacity is exhausted during processing, Edit types the message "BUFFER FULL." The user should then dump and initialize the edit buffer prior to entering the next statement.

OPERATING PROCEDURES

The procedures for operating Edit are summarized below.

1. Load the object tape containing Edit with the Binary Loader.
2. When loading is complete, toggle the console Reset switch.
3. Toggle the Run switch. Edit outputs a carriage return to the teletype printer. The user may then enter statements and commands via the teletype keyboard.

At the end of the Edit object tape are two additional records which may be loaded to switch tape reading and punching operations to the high speed reader and high speed punch, respectively. The first record may be loaded by toggling the Run switch after the main portion of the object tape has been loaded. The second record may be loaded by toggling Run after the first record has been loaded.

When the first record is loaded, Edit is configured to accept input from the high speed reader for Load, Edit-Print, and Edit-Dump operations. When the second record is loaded, Edit is configured to use the high speed punch for Dump and Edit-Dump operations.

ASSEMBLER

The FABRI-TEK MP12 Assembler translates symbolic assembly language programs into executable machine programs. Programs are prepared on paper tape, using the MP-12 Source Edit Utility, and then input to the Assembler by way of the teletype reader. The Assembler reads the source tape containing the program and produces (1) a printed listing of the assembled program, and (2) a punched object tape containing the machine version of the program. The object tape may then be loaded using the Binary Loader, and the program executed on the computer.

Three separate passes, or readings of the source program tape, are required to complete the assembly process. The function of each pass is indicated below.

Pass 1 – The Assembler reads the source tape and constructs a table of program symbols for reference during passes 2 and 3.

Pass 2 – The Assembler reads the source tape and punches an object tape containing the machine version of the program.

Pass 3 – The Assembler reads the source tape and prints a listing of the assembled program.

Once pass 1 has been completed, passes 2 and 3 may be selected in any order or repeated as desired. The assembler is completely self-initializing and a pass may be restarted at location 0200 in case it is manually terminated.

ASSEMBLY LISTING FORMAT

A listing of the assembled program is printed during pass 3 of the assembly. The format of the listing is indicated below.

Print position	1:	Error flag
	2:	Blank
	3-6:	Location (octal)
	7:	Blank
	8-11:	Contents of location (octal)
	12:	Blank
	13-16:	Statement number (decimal)
	17:	Blank
	18-72:	Program statement

The text of the program statement is printed in the following format: If the first character of the statement is a slash (/), the statement is listed in print positions 18 through 72. Otherwise, if a label is specified, the label is printed, left justified, in print positions 18-24. The remainder of the statement, beginning with the next non-blank character, is listed beginning in print position 25. If a label is not specified, the statement, beginning with the first non-blank character, is listed beginning in print position 25. In all cases, if a slash is encountered prior to print position 41, the remainder of the statement, beginning with the slash, is listed beginning in print position 41.

The Assembler prints 52 lines per page; each page is numbered in decimal. Eleven inch page separation marks, consisting of six dashed lines, are provided to aid manual page separation. If multiple lines of output result from the same statement, the first line printed contains the statement text in print positions 18 through 72. The remaining lines are truncated at print position 12.

ERROR FLAGS

If an error is detected in processing a statement, the statement is printed along with an error flag in print position 1. The error flags and their meanings are indicated as follows:

ERROR FLAG	MEANING
S	STATEMENT ERROR. An illegal or unexpected character has been encountered in processing the current statement. This error is indicated during assembly pass 3, and also pass 1 if detected in processing an expression following an equal sign (=) or asterisk (*).
D	DUPLICATE DEFINITION. A label or equality symbol has been duplicated in the current statement. The value assigned at the first occurrence is used. This error is only indicated during pass 1.
U	UNDEFINED SYMBOL. A symbol appearing within the current statement has not been defined as either a label, an equality symbol, or an Assembler mnemonic. A value of zero is assumed. This error is indicated during pass 3, and also pass 1 if detected in processing an expression following an equal sign (=) or asterisk (*).
I	ILLEGAL COMBINATION. An illegal combination of operate instructions has been specified in the current statement. This error is indicated during assembly pass 3.
P	PAGE ERROR. A memory reference instruction operand does not lie within either the current page or the base page; the instruction cannot be assembled in the present form. The highest current page address is inserted for the instruction operand address. This error is indicated during assembly pass 3.

F	SYMBOL TABLE FULL. The program symbol table has overflowed available storage. The assembly continues, but no further symbols are stored. This error is indicated during assembly pass 3.

EXTENDED MNEMONIC SET

The following additional mnemonics are recognized by the assembler.

MNEMONIC	OCTAL CODE	OPERATION
STL	7120	Set Link to 1.
GLK	7204	Copy Link into AC bit 11.
STA	7240	Set each bit of the AC to 1.
CIA	7041	Two's complement the AC.
LAS	7604	Load AC from switch register.

STORAGE REQUIREMENTS

The assembler occupies memory locations 0 through 3175_8. Locations 3176_8 – 7577_8 are used for symbol table storage.

OPERATING PROCEDURES

The procedures for operating the assembler are described as follows:

1. Load the assembler object tape using the Binary Loader.

2. When loading is complete, toggle the console Reset switch.

3. Enter the selected pass number into console switches 10-11 as follows:

   01_2 – Pass 1
   10_2 – Pass 2
   11_2 – Pass 3
   Switches 0-9 are ignored.

4. Ready the teletype and position the tape containing the source program in the reader with blank tape under the read heads. For passes 1 and 3, the teletype punch should be turned OFF. For pass 2, the teletype punch must be turned ON.

5. Toggle the Run switch to commence processing. The assembler begins the selected pass and executes a halt when an end-of-program statement (the $ statement) is encountered. A halt is executed at location 0177_8 with 0000_8 displayed in the AC. The user may then select the next pass and repeat steps 3 through 5 above.

At the end of the Assembler object tape are two additional records which may be loaded to switch tape reading and punching operations to the high speed reader and high speed punch, respectively. The first record may be loaded by toggling the Run switch after the main portion of the object tape has been loaded. The second record may be loaded by toggling Run after the first record has been loaded.

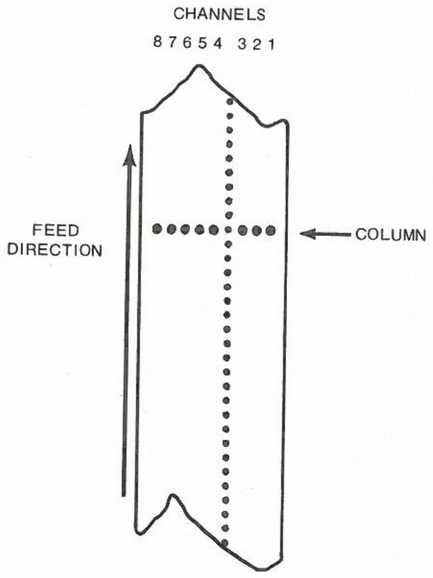
When the first record is loaded, the assembler is configured to accept source input from the high speed reader for passes 1, 2, and 3. When the second record is loaded, the assembler is configured to punch an object tape on the high speed punch during pass 2.

BINARY LOADER

The FABRI-TEK MP12 Binary Loader is used to load binary object tapes obtained from the Assembler or Debugging utility. Each tape is made up of a number of load records which contain address data, object data, and checksum data. The Loader reads the object tape via the teletype reader and processes each record sequentially by placing the object data into the memory locations specified by the address data.

PAPER TAPE

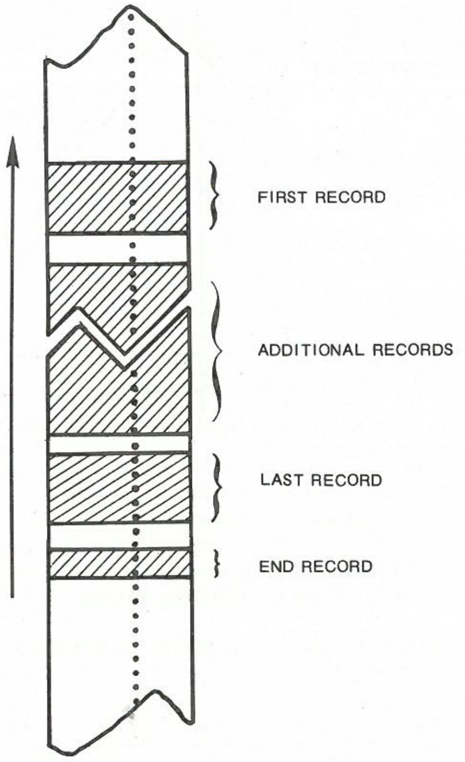
Data is recorded on paper tape by groups of holes arranged in binary format along the length of the tape. The tape is divided into columns which run the length of the tape, and channels which extend across the width of the tape; as illustrated below.



A column can represent eight bits (one byte) of binary information; a channel punch within a column denotes a binary one, and an unpunched channel represents a binary zero. For example, a column with a punch in channels 2, 4, and 8 would represent the binary quantity 10001010, or 212_8.

TAPE FORMAT

A binary object tape consists of one or more records which are organized in a sequential fashion along the length of the tape. Each record consists of a maximum of 67, and a minimum of 1, consecutive tape columns. Blank tape may appear before, between, or after records. The arrangement of records on a binary object tape is illustrated as follows:

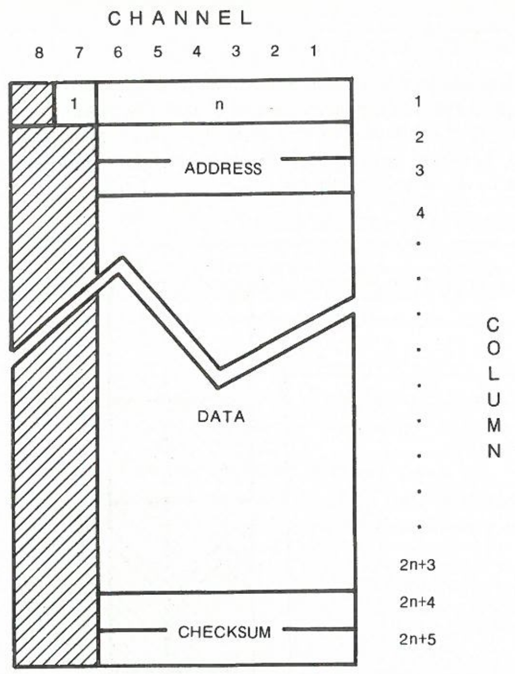


RECORD ARRANGEMENT

RECORD FORMAT

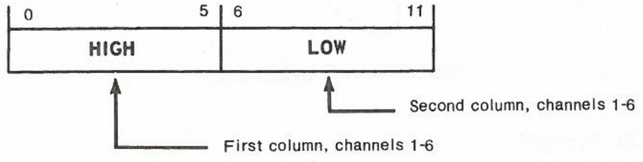
The following illustrates the format of a binary load record.

LOAD RECORD FORMAT



The first column of a load record contains a control punch in channel 7, followed by a count (n) of the number of data words in channels 1 through 6. This count excludes the address word and checksum word, and specifies a maximum of 31 (37_8) data words. A count of zero indicates an end record and causes the loader to discontinue processing. An end record is characterized by a tape column containing a channel 7 punch only; the binary quantity 001000000, or 100_8.

Address, data, and checksum words are formed from pairs of consecutive tape columns as illustrated below.



The control column of a load record is followed by an address word which gives the starting address of the data words to be loaded. If the record count is represented by n and the address by a, then the n data words following the address word are inserted into memory locations a through a + n - 1, inclusive. Each data word is assembled from two successive tape columns and placed into the proper memory location. The data words within the record are followed by a checksum word which

represents the arithmetic sum of all columns in the record, excluding the checksum columns. The loader computes a checksum while loading the record and compares the computed checksum with the record checksum after the record has been loaded. In case of disparity, the loader halts processing and displays an error code in the AC console display.

OPERATING PROCEDURES

The procedures for operating the MP12 Binary Loader are summarized below.

1. If the Binary Loader is not resident in memory locations 7702_8 through 7777_8, refer to the Loading the Binary Loader information.
2. Position the object tape to be loaded in the teletype reader with blank tape under the read heads. Insure that the unit is powered, on-line, and the reader switch is set to the START position.
3. Toggle the console Reset switch and enter 7777_8 into the PC via the console switches.
4. Toggle the Run switch. The reader should begin reading tape; the loader will halt when the load is complete, or an error condition is encountered, with one of the following codes present in the AC console display:

HALT CODE	MEANING
0000_8	END OF LOAD. The user may place a new tape in the reader and toggle Run to continue loading, if desired. Otherwise the program just loaded may be executed by setting the PC to its starting address and toggling Run.
0200_8 0300_8	RECORD ERROR. The first column of a record does not contain a punch in channel seven, or contains a punch in channel eight. Possible causes of this error are a wrong format tape, a punch error, or a read error. Toggling Run causes the loader to process the next record.
7777_8	CHECKSUM ERROR. The last record loaded contained a checksum error and as a result, the loaded data does not correspond to the tape data. Possible causes of this error are a punch error or a read error. The record may be reread by manually repositioning the tape at the beginning of the last record and toggling the console Run switch. Otherwise, toggling Run causes the loader to process the next record.

LOADING THE BINARY LOADER

The Binary Loader occupies memory locations 7702_8 through 7777_8. In the event that the loader must be reloaded, a 14 word bootstrap program is used to load a special tape containing the loader. The bootstrap program resides in memory locations 7662 through 7677 as listed below.

LOCATION	CONTENTS
7662	7700
7663	1262
7664	3261
7665	6032 or 6012 for High Speed Reader
7666	7106
7667	7006
7670	7006
7671	6031 or 6011 for High Speed Reader
7672	5271
7673	6034 or 6014 for High Speed Reader
7674	7420
7675	5265
7676	3661
7677	2261

The procedures for loading the tape containing the binary loader are summarized as follows:

1. Insure that the bootstrap program is installed in memory locations 7662_8 through 7677_8. If not, manually reload via the console switches.
2. Position the tape containing the loader in the teletype reader with blank tape under the read heads. Insure that the unit is powered, on-line, and that the reader switch is set to the START position.
3. Toggle the console Reset switch and enter 7662_8 into the PC via the console switches.
4. Toggle the Run switch. The teletype should begin reading tape. The bootstrap program will execute a halt with the PC set to 7702_8 when the loader is loaded. Toggling Run will cause the binary loader to begin execution.

Once loaded, the Binary Loader is configured to use the teletype reader. The input device may be switched to the high speed reader by manually changing the contents of location 7755_8 from 6037_8 to 6017_8.

DEBUGGING UTILITY

The FABRI-TEK MP12 debugging utility (Debug) is an interactive debugging tool which aids program development and testing. Debug normally operates in conjunction with a user program undergoing checkout, and provides a number of functions which support the checkout activity. These functions are summarized below.

- Set and clear execution breakpoints for a program under test.
- Transfer control to a program under test.
- Display and alter registers for a program under test.
- Trace program execution.
- Display and alter specified memory locations.
- Print memory contents between specified locations.
- Dump memory contents, between specified locations, on paper tape in binary load format.
- Fill memory, between specified locations, with a specified value.
- Search memory, between specified locations, for a specified pattern.

COMMANDS

Commands to Debug are implemented as single letter mnemonics followed by a maximum of four octal command parameters separated by commas. All commands are terminated by typing a carriage-return, ®. Command parameters are unsigned octal integers; if more than four digits are typed, only the last four are used. If an error is detected during command processing, Debug types a question mark (?) under the command line and the command is not executed. In case an error is discovered during command typein, depressing the keyboard ESC key causes the command to be ignored. After a command has been processed, Debug positions the teletype carriage at the beginning of the next line and a new command may then be entered.

In the following command descriptions, the letters x, y, m, and n represent user-entered octal values. Optional command parameters are enclosed in square brackets ([ ]). The symbol ® denotes a typed carriage return.

SET BREAKPOINT

The Set Breakpoint command is issued with respect to a program under test and causes a breakpoint to be set into a specified memory location. When the program attempts to execute the instruction at the breakpoint location, control is transferred back to Debug. Debug clears the breakpoint; preserves the contents of the accumulator, link, and PC for the executing program; and types out the breakpoint number, breakpoint location, and the contents of the accumulator and link. The format of the Set Breakpoint command is indicated below.

Bn, x ®   Set Breakpoint

The parameter n is the breakpoint number assigned by the user and may range from 0 to 7, enabling eight possible breakpoints to be set simultaneously. The parameter x specifies the location at which the breakpoint is to be inserted. Setting a breakpoint causes any previous breakpoint with the same breakpoint number to be cleared. The following example illustrates the setting of a breakpoint and the typeout which occurs when the breakpoint location is executed. Note that the execution of a breakpoint location does not result in the execution of the instruction at that location; the typeout represents conditions existing prior to the execution of the breakpoint instruction.

```
B7,403® ←SET BREAKPØINT 7 AT LØCATIØN 0403
G400®←──GØ TØ LØCATIØN 400
  BP     PC    AC    L   {TYPEØUT WHEN BREAK-
 0007   0403  1121   1   {PØINT 7 IS EXECUTED
```

Debug uses memory location 0 for breakpoint processing. Programs executing under Debug must not use this location.

CLEAR BREAKPOINT

This command results in the clearing of previously set breakpoints. The format of the Clear Breakpoint command is indicated as follows:

C[n] ® Clear Breakpoint

The optional parameter n is the number of the breakpoint to be cleared. If n is not specified, all breakpoints are cleared.

EXAMPLE:

```
C7® ←CLEAR BREAKPØINT 7
C® ←──CLEAR ALL BREAKPØINTS
```

GO TO PROGRAM

This command causes the accumulator and link, preserved from the last breakpoint execution, to be restored and control to be passed to the test program. The format of the Go command is indicated below.

G[x] ®   Go to program location

The optional parameter x is the address to which control is to be passed. If x is not specified, control is passed to the location at which the last breakpoint was executed. The Go command is usually preceeded by a Set Breakpoint command; otherwise no return linkage to Debug is maintained.

EXAMPLE:

```
BO,215® ←SET BREAKPØINT 0 AT LØCATIØN 0215
G210® ←──GØ TØ LØCATIØN 0210
  BP     PC    AC    L   {TYPEØUT WHEN BREAK-
 0000   0215  3172   1   {PØINT 0 IS EXECUTED
BO,220® ←SET BREAKPØINT 0 AT LØCATIØN 0220
G® ←─────GØ TØ LAST BREAKPØINT LØCATIØN
  BP     PC    AC    L   {TYPEØUT WHEN BREAK-
 0000   0220  0172   0   {PØINT 0 IS EXECUTED
```

REGISTER DISPLAY AND ALTER

These commands enable the contents of the accumulator and link, preserved from the last breakpoint execution, to be displayed and optionally changed. The formats of the register display commands are indicated below.

A   Display accumulator
L   Display link register

When the command letter is typed, Debug types the contents of the specified register. The user may then type a carriage return to terminate the display, or a new value to replace the existing value, followed by a carriage-return. The link register may only be set to the value zero or one.

EXAMPLE:

```
A  6543® ←CØNTENTS ØF AC ARE 6543
L  0001 0®←CØNTENTS ØF L ARE 1, CHANGE TØ 0
L  0000® ←CØNTENTS ØF L ARE NØW 0
A  6543 6523®←CHANGE CØNTENTS ØF AC TØ 6523
A  6523® ←───CØNTENTS ØF AC ARE NØW 6523
```

TRACE PROGRAM EXECUTION

The Trace command causes snapshot typeouts to occur as program instructions are executed. The format of the trace command is indicated below.

T[x] ®   Trace program

The optional parameter x is the address to which control is to be passed and the trace is to begin. If x is not specified, the trace begins at the location at which the last breakpoint was executed, or the location at which the last trace was completed. Each trace typeout provides the following information:

1. The location of the instruction, i.e., the contents of the program counter (PC).
2. The instruction executed, i.e., the contents of the instruction register (IR).
3. The contents of the accumulator after execution of the instruction (AC).
4. The contents of the link register (L) after execution of the instruction.
5. For memory reference instructions, the effective address of the instruction operand, i.e., the contents of the memory address register (MA).
6. For memory reference instructions, the contents of the effective address after execution of the instruction, i.e., the contents of the memory data register (MD).

In contrast to breakpoint execution, the trace typeout represents conditions existing after the execution of an instruction. An execution trace, once started, may be terminated by depressing the keyboard BREAK key. The following conventions are adopted with respect to trace processing:

1. Input/Output instructions are not traced and result in trace termination at the location containing the instruction.
2. Tracing is terminated whenever a breakpoint instruction is encountered.
3. HLT instructions result in trace termination at the location containing the instruction.
4. Auto-index addressing through locations 10-17 is legal.

When a trace is terminated by Debug, the value of the program counter is typed following the last line of trace typeout. The instruction at this location is not executed.

The following is a sample execution trace.

EXAMPLE:

```
T600® ←INITIATE TRACE AT LØCATIØN 0600
   PC     IR     AC    L    MA     MD
  0600   7010   3251   1
  0601   7620   0000   1
  0603   4511   0000   1   1500   0604
  1501   1022   0002   1   0022   0002
  1502   7010   4001   0
  1503   7620   0000   0
  1504   5700   0000   0   0604   4515
  0604
```

DISPLAY AND ALTER MEMORY

This command enables the contents of a specified memory location to be displayed and optionally changed to a specified value. The format of the Display and Alter Memory command is indicated below.

Mx ®   Display memory location x

The parameter x is the location to be displayed. The location and its contents are typed on the next line as illustrated below.

EXAMPLE:

```
M32® ←——————————————DISPLAY LØCATIØN 0032
  0032   1356       CØNTENTS ØF 0032 ARE 1356
M400® ←—————————————DISPLAY LØCATIØN 0400
  0400   7000       CØNTENTS ØF 0400 ARE 7000
```

After the contents of a location have been displayed, typing a carriage-return terminates the display function. In place of a carriage-return, the user may type a new value to replace the existing contents, followed by a carriage-return. This procedure is illustrated as follows:

EXAMPLE:

```
M400® ←——————————————DISPLAY LØCATIØN 0400
  0400   7000  7777® CØNTENTS ARE 7000, CHANGE
                     TØ 7777
M400® ←——————————————DISPLAY LØCATIØN 0400
  0400   7777        CØNTENTS ARE NØW 7777
```

In place of typing a carriage-return after a location has been displayed or the contents of a location changed, the user may type one of the following characters:

,	Display next location
.	Display location again
/	Display last location
I	Display indirect. Replace location by its contents and display.

The following example illustrates the use of the display control characters:

EXAMPLE:

```
M400®  ←————————————————DISPLAY LØCATIØN 0400
  0400    7777 ,←————————CØNTENTS ARE 7777, TYPE
                         "," TØ DISPLAY NEXT
  0401    7200 7600 .←———CØNTENTS ØF LØCATIØN 0401
                         ARE 7200, CHANGE TØ 7600
                         AND TYPE "." TØ DISPLAY
                         AGAIN
  0401    7600 ,←————————CØNTENTS ØF LØCATIØN 0401
                         ARE NØW 7600, TYPE "," TØ
                         DISPLAY NEXT LØCATIØN
  0402    1073 I←————————CØNTENTS ØF LØCATIØN 0402
                         ARE 1073, TYPE "I" TØ
                         DISPLAY INDIRECT
  1073    3275 /←————————CØNTENTS ØF LØCATIØN 1073
                         ARE 3275, TYPE "/" TØ
                         DISPLAY PREVIØUS LØCATIØN
  1072    1144® ←————————CØNTENTS ØF LØCATIØN 1072
                         ARE 1144, TYPE CARRIAGE-
                         RETURN TØ TERMINATE DISPLAY
```

PRINT MEMORY CONTENTS

This command causes the contents of memory, between two specified locations, to be listed on the printer. The format of the Print command is indicated below.

Px,y ®   Print memory contents between location x and y

The listing originates on an even octal boundary and terminates at the specified end address. The listing is printed with nine octal values per line, the first value being the starting address of eight consecutive memory locations which contain the next eight values listed. When in progress, the listing may be terminated by depressing the BREAK key.

EXAMPLE:

```
P403,424®
  0400   7006   0120   1146   7450   5302   7001   7650   5267
  0410   4467   5264   4501   1036   1175   7640   5233   4473
  0420   4500   1036   1162   7650   5227
```

DUMP MEMORY CONTENTS

This command causes the contents of memory, between two specified locations, to be dumped in binary load format on the teletype punch. The format of the Dump command is indicated below.

Dx,y ®   Dump memory between locations x and y

The dump is preceeded and followed by 72 blank tape columns. Immediately after the dump starts, the user should depress the punch ON button and tear off any non-blank tape. When the dump stops, the user should depress the punch OFF button and tear off the punched tape. The tape should then be marked for future reference. The punched tape received via the Debug Dump command may be loaded using the binary loader.

FILL MEMORY

The Fill Memory command enables the contents of memory, between two specified locations, to be set to a specified value. The format of the Fill Memory command is indicated below.

Fx,y[,m] ®   Fill memory between locations x and y

The optional parameter m, if specified, is the fill value. If m is not specified, zero is assumed.

EXAMPLE:

```
F4000,4037®
F4010,4027,7777®
P4000,4037®
  4000   0000   0000   0000   0000   0000   0000   0000   0000
  4010   7777   7777   7777   7777   7777   7777   7777   7777
  4020   7777   7777   7777   7777   7777   7777   7777   7777
  4030   0000   0000   0000   0000   0000   0000   0000   0000
```

SEARCH MEMORY

This command enables memory, between two specified locations, to be searched for a specified pattern. The format of the Search Memory command is indicated below.

Sx,y[,n[,m]] ®   Search memory between locations x and y

The optional parameter n, if specified, is the search value. If n is specified, the optional parameter m is a mask to which the contents of each search location are to be logically AND'ed prior to comparison with the search value. If n is not specified, 0 is assumed; if m is not specified, octal 7777 is assumed. Thus any of the following forms of the search command are legal:

Sx,y	Search memory locations x through y for zero
Sx,y,n	Search memory locations x through y for the value n
Sx,y,n,m	Search memory locations x through y for a match to the value n under the mask m

Whenever a match is detected during the search, the following typeout occurs:

xxxx yyyy

where xxxx is the location at which a match was found and yyyy is the contents of location xxxx. The following example illustrates the use of the Search command to locate all JMP instructions (instructions with an op code of 5) in the range of 7200 through 7400.

EXAMPLE:

```
S7200,7400,5000,7000®
  7224  5210
  7226  5600
  7227  5770
  7244  5247
  7266  5270
  7267  5277
  7271  5261
  7276  5242
  7305  5770
  7312  5310
  7313  5706
  7341  5714
  7345  5742
```

STORAGE REQUIREMENTS

Debug occupies memory locations 5600_8 through 7577_8. Base page location 0 is used for breakpoint processing.

OPERATING PROCEDURES

The procedures for operating Debug are summarized below.

1. If Debug is not resident in memory, load the paper tape containing Debug using the Binary Loader.

2. Toggle the console Reset switch and enter 5600_8 into the PC via the console switches.
3. Toggle the Run switch, Debug outputs a carriage return to the teletype printer. The user may then enter commands via the teletype keyboard.

At the end of the Debug object tape is an additional record which may be loaded to switch Dump operations to the high speed punch. This record may be loaded by toggling the Run switch after the main portion of the object tape has been loaded.

If it is necessary to restart Debug, the user should be aware that any breakpoints which may be set are automatically cleared. The contents of the accumulator, link, and program counter, preserved for the program under test, are not altered. Initially, the values of the PC, AC, and link are set as follows:

PC:	200_8
AC:	0
LINK:	0

PROCESSOR DIAGNOSTIC

The FABRI-TEK MP12 Processor Diagnostic is used to verify MP12 Microprocessor operation. It is organized into 23 groups of test routines as indicated below.

1. SKP Test Group
2. SZA Test Group
3. SPA Test Group
4. SMA Test Group
5. SZL Test Group
6. DCA-TAD Test Group
7. TAD Test Group
8. IAC Test Group
9. RAR Test Group
10. RTR Test Group
11. RAL Test Group
12. RTL Test Group
13. Group I Operate Test Group
14. OSR Test Group
15. NOP Test Group
16. Group II Operate Test Group
17. AND Test Group
18. JMP Test Group
19. DCA Test Group
20. ISZ Test Group
21. JMS Test Group
22. Indirect Addressing Test Group
23. Auto-Index Test Group

Each test group consists of a series of individual tests for specific processor functions that fall under the general group heading. Each test function is identified and documented by a comments block on the program listing. Tests are executed by the processor in the order indicated above.

STORAGE REQUIREMENTS

The Processor Diagnostic occupies memory locations 0_8 – 7546_8 with an initial starting address of 200_8.

OPERATING PROCEDURES

The procedures for operating the Processor Diagnostic are summarized below.

1. Load the Processor Diagnostic object tape using the binary loader.
2. When loading is complete, set the Switch Register to 7777_8, toggle the console Reset switch, and toggle the Run switch. The program will cycle continuously unless an error is detected. In the event of an error, the test routine detecting the error will execute a halt instruction. The location at which the HLT was executed may be used to reference the program listing for specific information regarding the error.

CROSS-ASSEMBLER

The FABRI-TEK MP12 Cross-Assembler executes under IBM DOS and permits programs written for the MP12 computer to be assembled on IBM 360/370 series computers. Input to the assembler is in card form from the system unit SYS005. A program listing is written on SYS007 and an object deck output on SYS006. SYS002 is used as an intermediate storage file. The standard assignments for the system units are as follows:

SYS002 –	2314 DASD
SYS005 –	Card Reader
SYS006 –	Card Punch or Magnetic Tape
SYS007 –	Line Printer

All system units must be assigned for an assembly. In case an object deck is not required, the statement

```
//ASSGN SYS006,IGN
```

must be specified. A sample input deck is depicted below and assumes that the assembler has been previously cataloged in the core image library under the name "XMP12".

```
//JOB ASSEMBLE
//ASSGN SYS005,--
//ASSGN SYS006,--
//ASSGN SYS007,--
//EXEC XMP12

{Source deck terminated by a $ card}

/*
/&
```

Object data is output on SYS006 in the following format:

1. The first four character field of each card contains the beginning octal address, in hollerith code, of the data field which follows.

2. The address field is followed by a data field consisting of a maximum of 18 four-character data values; each data value represents an octal data word in hollerith card code.

3. The data field is terminated by a blank column.

APPENDIX A
FABRI-TEK MP12 INSTRUCTION SET

MEMORY REFERENCE INSTRUCTIONS

MNEMONIC SYMBOL	OPERATION CODE	EXECUTION TIME*	OPERATION DESCRIPTION
AND Y	0	Direct-3.0 Indirect-4.5	LOGICAL AND. This instruction generates the logical product of the contents of memory location Y and the contents of the accumulator. The result replaces the previous contents of the accumulator. The bit stored in the link register is not affected by the LOGICAL AND operation.
TAD Y	1	Direct-3.0 Indirect-4.5	TWO'S COMPLEMENT ADD. This instruction generates the arithmetic sum of the contents of memory location Y and the contents of the accumulator. The result replaces the previous contents of the accumulator. If the operation produces a carry from the most significant bit position, the link bit is complemented.
ISZ Y	2	Direct-3.0 Indirect-4.5	INCREMENT AND SKIP IF ZERO. This instruction adds one to the contents of memory location Y. If the result is zero the next instruction in sequence is skipped. The contents of the link register and accumulator are not affected by the INCREMENT AND SKIP IF ZERO operation.
DCA Y	3	Direct-3.0 Indirect-4.5	DEPOSIT AND CLEAR ACCUMULATOR. This instruction copies the contents of the accumulator into memory location Y and then clears the accumulator to zero. The bit stored in the link register is not affected by the DEPOSIT AND CLEAR ACCUMULATOR operation.
JMS Y	4	Direct-3.0 Indirect-4.5	JUMP TO SUBROUTINE. This instruction copies the address of the next instruction in sequence into memory location Y and transfers program control to location Y+1. The contents of the link register and accumulator are not affected by JUMP TO SUBROUTINE operation.
JMP Y	5	Direct-1.5 Indirect-3.0	JUMP. This instruction transfers program control to location Y. The contents of the link register and accumulator are not affected by the JUMP operation.

*Time referenced in microseconds.