National Semiconductor

MOLE™

MOLE™ Brain Board

User's Manual

Microcomputer Systems

Publication Number 420408188-001A
Order No. 420408188-001
March 1984

MOLE™

MOLE™ Brain Board
User's Manual

2900 Semiconductor Drive
Santa Clara, California 95051

REVISION RECORD

REVISION	RELEASE DATE	SUMMARY OF CHANGES
A	03/84	First Release. MOLE™ Brain Board User's Manual Publication No. 420408188-001

PREFACE

This manual describes the MOLE™ Brain board, its hardware and firmware. In particular it describes how to connect a Brain board to a MOLE personality board, how to connect the Brain board/personality board to a host system and how to use the communication, PROM programming and diagnostic firmware. It also describes how to operate multi-MOLE systems.

The Brain board is one component of a MOLE development system. The other required component is a MOLE personality board. A host system is strongly recommended.

The personality board firmware adds chip-specific capabilities to the system firmware supplied by the Brain board.

The following National Semiconductor publications provide related study/reference material for using a MOLE system.

- MOLE PDS Systems Software User's Manual for COPS™ Applications
  Order Number 424410088-001.
- MOLE CP/M™ Systems Software User's Manual for COPS Applications
  Order Number 424409479-001
- COPS Microcontrollers Databook or The Micro Databook
- MOLE CP/M Systems Software User's Manual for TMP™ Applications.
  Order Number 424410087-001.
- MOLE COPS CMOS Personality Board User's Manual
  Order Number 420408189-001.
- MOLE TMP Personality Board User's Manual
  Order Number 420408203-001
- NS455 Series TMP Data Sheet.

WARNING

This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instructions manual, may cause interference to radio communications. As permitted by regulation it has not been tested for compliance with the limits for Class A computing devices pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

The information in this manual is for reference only and is subject to change without notice.

CONTENTS

ILLUSTRATIONS

TABLES

Chapter 1

INTRODUCTION/OVERVIEW

1.1 INTRODUCTION-BRAIN BOARD AND MOLE

The MOLE Brain board is one part of a MOLE development system. The National Semiconductor MOLE (Microcontroller On-Line Emulator) is an emulation system meant for the development of programs for NSC microcontrollers such as members of the COPS family or the NS455 Terminal Managment Processor.

1.1.1 MOLE

A MOLE consists of two printed circuit boards. One is common to all versions of MOLE and is referred to as the MOLE Brain board. The other is specific to the chip or chip family being emulated and is called a personality board.

The personality board contains the necessary hardware and firmware for microcontroller emulation. It has also has sockets for connection to the target system. See the appropriate personality board user's manual for information on emulation and debugging functions. The Preface contains a list of currently available personality board user's manuals.

The Brain board contains an NSC800 microprocessor, 64K bytes of read/write memory, and 32K bytes of firmware. It also has three serial RS-232 channels and a PROM programmer. The Brain board firmware supplies programs to perform basic operations such as moving and inspecting data in buffers, communication with a host or other MOLEs, performing diagnostic tests and PROM burning. The personality board firmware appears as additional system programs; these additional programs are only described in the personality board's user's manual.

The MOLE is a stand-alone emulation system. The firmware on the two boards is sufficient for all emulation tasks. However, a host system (such as National Semiconductor's PDS, STARPLEX or any system running CP/M) permits the user to edit and assemble microcontroller assembly language source files. The host's assembler outputs object code which can then be downloaded to the MOLE.

1.1.2 Brain Board Manual

This manual describes:

- How to connect a Brain board to a personality board and power supply
- How to connect the MOLE to a host system or a terminal
- How to operate the communication firmware
- How to use the diagnostic firmware
- The PROM programming functions supplied by the Brain board

This manual is also meant as a general guide to the use of a MOLE system. As such, it refers to particular personality boards in examples, but the features described here apply no matter which personality board is actually used.

The physical description, setup, and initial checkout of the Brain board are described in Chapter 2. Although familiarity with the operation and capabilities of the MOLE will be useful, it is not necessary for the initial part of the check-out since this part is presented in a step-by-step manner. However, to test the complete system the user will want to transfer a file to and from the MOLE and the host; this requires some knowledge of the communication program COMM.

General rules of operation such as the available programs, how to invoke programs, and the command-line editing capabilities are given in Chapter 3.

The firmware programs, EXEC, COMM, PROG and DIAG, are described individually.

EXEC is described in Chapter 4. EXEC is the MOLE's executive program. The user moves from one firmware program to another through EXEC commands. A program can be invoked from EXEC by entering its name; one program can be ended and another begun by using the EXEC command "@". EXEC also provides functions which permit the user to add and subtract both decimal and hexadecimal numbers, compare buffers, erase buffers, move data between buffers and display and change the status of RS-232 parameters and buffers.

COMM is described in Chapter 5. COMM provides for transfer of files and messages between a host and a MOLE. COMM programs are available for several host systems; these COMMs may differ slightly from the COMM operated from the MOLE. Check the host system documentation for details on the host COMM.

Chapter 6 describes PROG, the PROM programming program.

Chapter 7 describes the diagnostic program DIAG. DIAG tests the system RAM, ROM and RS-232 channels J2 and J3.

Chapter 8 describes the rules, setup, commands and possible configurations for multiple-MOLE operation.

Appendix A provides information on RS-232 connections such as common cable arrangements, signals and the terminal-type assignments for the Brain board and several possible host systems.

Appendix B contains a table listing the function and default settings for the jumpers and switches.

Appendix C shows a sample MOLE session.

Appendix D summarizes the input line editing commands originally described in Chapter 3.

Appendix E describes power-on diagnostics.

1.2 CONVENTIONS

Certain conventions are used in the description of the statement syntax for commands; others are used in examples. The command syntax conventions are described in Section 1.2.1; the example conventions are described in Section 1.2.2.

1.2.1 Command Syntax Conventions

The following notational conventions are used in the syntax descriptions:

- The minimum abbreviation for a command name or option is indicated by the upper-case letters in the name.

  Example:

  Syntax: Help

  The command "HELP" can be abbreviated to "H".

- Pairs of square brackets ([]) are used to indicate optional parts of the syntax. Bracketed items within a bracketed item can only be selected if the outer option is used.

  Example:

  Syntax: Receive <mole buf>[<opt>[<opt>...]]

  The (COMM) RECEIVE command has an optional item specified by:

  [<opt>[<opt>...]]

  Within this item there is another optional item:

  [<opt>...]

  The inner <opt> need not be specified if the outer <opt> is specified; however, it cannot be specified unless the outer <opt> is specified.

- Pairs of angle brackets (<>) are used to enclose descriptive names or phrases. The bracketed item is to be replaced by a specific incidence of the item. For example, in the syntax for the RECEIVE command in the preceding paragraph, the <opt>'s indicate where the user specifies options for a file being received from another system. The bracketed item may be described in the text, or may require its own syntax description.

- A vertical bar (|) is used to separate mutually exclusive options.

  Example:

  Syntax: Chip [2716 | 2732 | 2764 | 2816]

  The user can only select among the chip numbers in the list. (Since the list is enclosed in square brackets, the user can also elect not to specify any chip.)

- Three dots (...) indicate optional repetition of the preceding item.

  In the preceding Receive example, the inner <opt> is followed by "...". There is more than one available option; the syntax indicates that several options can be specified in one command.

1.2.2 Example Conventions

The example conventions are as follows:

- Upper-case letters represent literal screen displays, whether generated by the user or the computer.

- User input is underlined.

- All user-input lines are terminated by a carriage return. The carriage return is not indicated, unless it is the only user input.

- Angle brackets (<>) are used to enclose the name of a control key or non-printing key; e.g. <cr> is used to represent a carriage return, <ctrl-C> represents a control C.

- Comments are introduced into examples by "<----".

Chapter 2

INITIAL INSTALLATION AND TESTING

2.1 INTRODUCTION

This chapter discusses how to get started with the Brain board. In particular it describes:

- What is shipped with the Brain board
- What must be supplied by the user
- The physical layout of the board including the location of connectors and jumpers
- How to connect a power supply to the Brain board and personality board
- The Brain board RS-232 ports
- The initial check-out

2.2 EQUIPMENT

The following items are shipped with each MOLE Brain board:

- One 4-wire power cable (this cable has a 5-position connector at one end)
- Two 25-wire RS-232 cables
- 12 nylon 1/4-inch screws
- 12 nylon 1/2-inch female threaded standoffs
- One set of Brain board schematics

For a minimum check-out, the following additional items will be needed:

- +5 volts DC 8-Amp power supply
- RS-232 terminal device
- A COPS CMOS personality board

2.3 BRAIN BOARD PHYSICAL LAYOUT

Figure 2-1 is a photograph of the Brain Board; Figure 2-2 shows the layout of components and connectors on the board. The main features of the board are:

- Three RS-232 connectors on the upper edge of the board. These are marked J1, J2 and J3, left to right.
- J8--a 5-position right-angle connector on the left-hand side, top of center. This is the power connector.
- SW1--at the lower left-hand corner of the board. This is used to reset the MOLE.
- Above SW1 there is a 28-pin zero-insertion-force socket labelled XU1. This is used for PROM programming.
- Underneath the board, on the right-hand side, are two 36-pin connectors, J4 and J5. These mate with corresponding connectors on the personality board.

2.4 POWER SUPPLIES

Both the MOLE Brain board and the personality board need +5 volts power supplies. A power cable is supplied with each of the boards. The power supply wiring is shown in Figure 2-3.

The cable with the 5-position connector is meant for connection to J8 of the MOLE Brain board. The required connection is:

J8 on Brain Board	Power Cord	Use
Pin 1	Red	+5 volts
Pin 2	Black	Ground
Pin 3	White	+21 volts or +25 volts
Pin 4	Black	Ground
Pin 5	Open	

NOTE: The +21 volts or +25 volts is used for PROM programming. This voltage should be applied to the board only during the actual PROM programming.

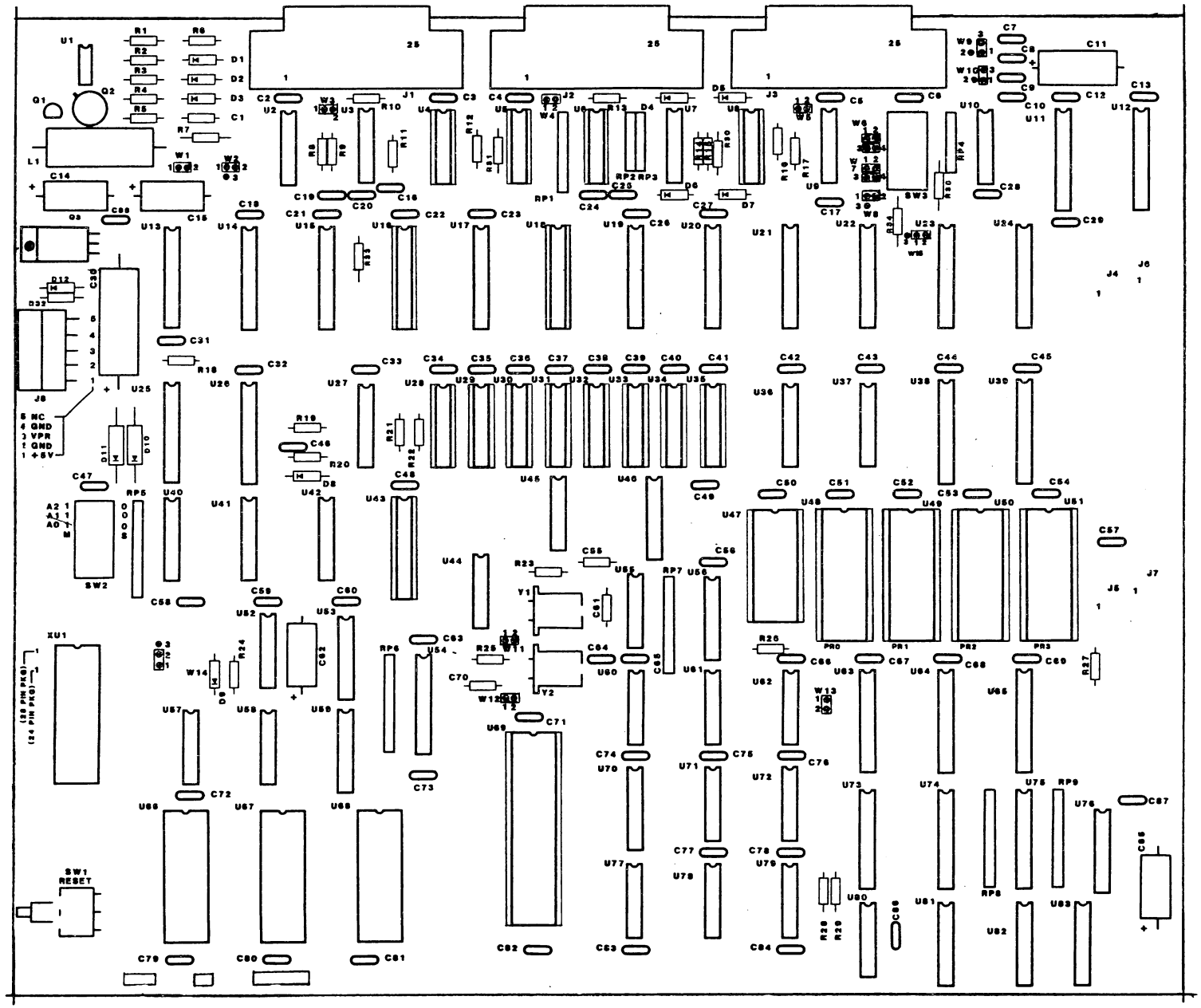


Figure 2-2 MOLE Brain Board Assembly Drawing

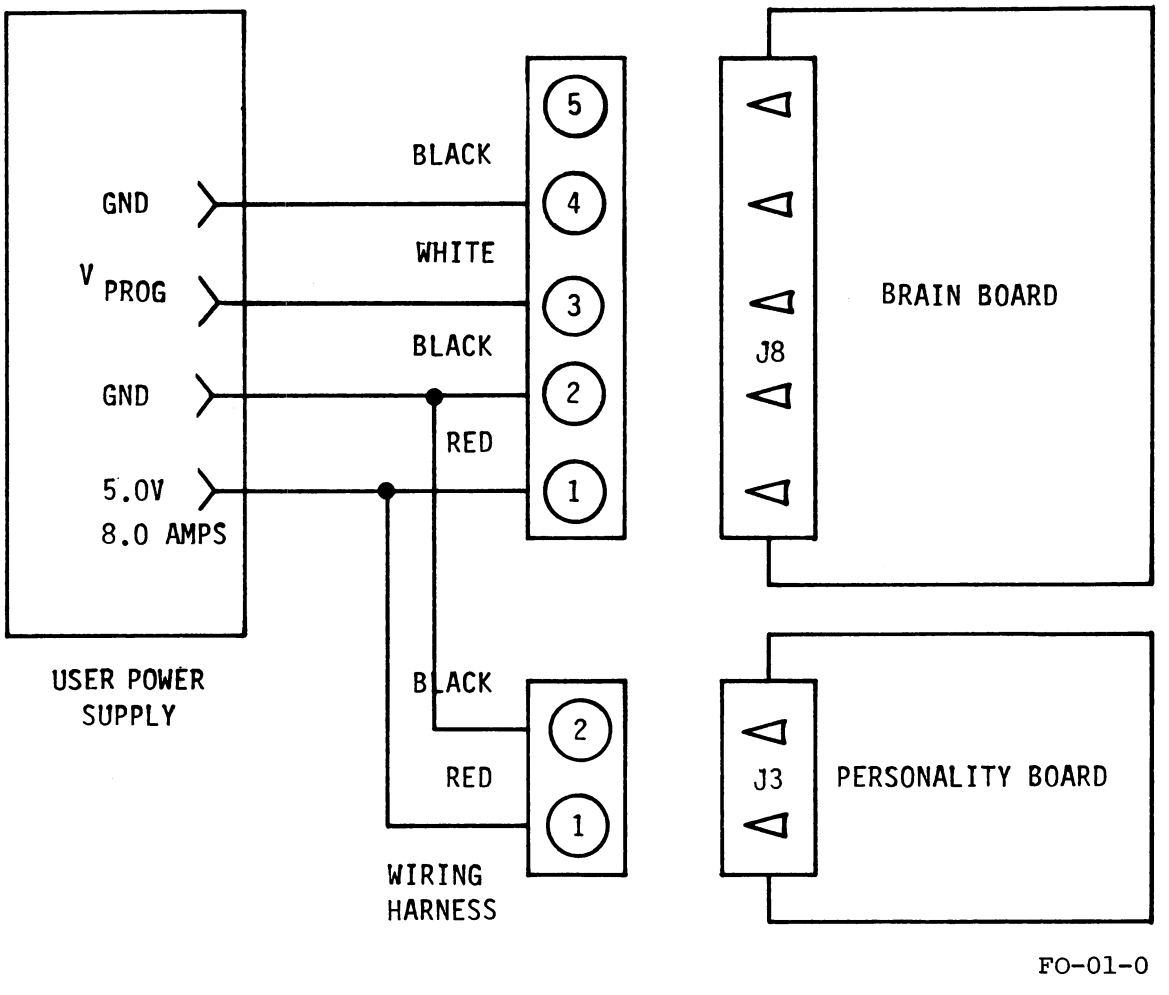


Figure 2-3 Power Supply Connector Wiring

The 2-position power cord is similarly used to supply +5 volts to the personality board; for example, for the 404C Personality board the required connection to J3 is:

J3 on Personality Board	Power Cord	Use
Pin 1	Red	+5 volts
Pin 2	Black	Ground

Total power consumption for the two boards is approximately eight Amps at five volts.

2.5 RS-232 CHANNELS

The MOLE Brain board has three RS-232 serial channels. The firmware also supports three logical communication channels: one for the system console, refered to as CN, one for serial data, refered to as SD, and one for a line printer, referred to as LP. When the MOLE Brain is reset or powered on, it scans all three serial channels for console input. The first channel to send a carriage return is assigned as the system console, CN. The MOLE Brain board determines the console baud rate, number of bits and parity from the carriage return, and sets all three channels up to match. This automatic determination requires that the console baud rate, number of bits and parity values be restricted to a certain set of values. Table 2-1 lists these allowed values.

Once the console channel is assigned, the other two channels are given default assignments. Table 2-2 lists these default assignments.

The device assignments and channel parameters may be changed using the EXEC STAT command described in Section 4.7.

A recommended configuration is:

- J1 and should be connected to a CRT terminal. (J1 is configured as a data set.) This terminal will be used to operate the MOLE and is the MOLE "console".

- J2 should be connected to a serial RS-232 port on the host development system (e.g. STARPLEX™). J2 is configured as a data terminal and is used to download object code from a host system. (The serial port on the host system must be configured as a data set, that is, suitable for connection to a CRT terminal.)

- J3 may be connected to a serial printer. J3 can be configured as either a data set or data terminal. It is shipped as a data set.

Appendix A discusses the RS-232 signals, cables and the port configurations for several devices.

TABLE 2-1 ALLOWED BAUD RATES AND BIT AND PARITY SETTINGS

ALLOWED BAUD RATES
110 300 600 1200 2400 4800 9600 19200
ALLOWED BITS AND PARITY VALUES
8 bits No parity 8 bits Odd parity 7 bits Even parity

TABLE 2-2 DEFAULT CHANNEL ASSIGNMENTS

```
If CN = S1 then SD = S2 and LP = S3
If CN = S2 then SD = S1 and LP = S3
If CN = S3 then SD = S2 and LP = S1


Where: CN = Console device
       SD = Serial data device
       LP = System printer device
       S1-S3 = Physical Brain board
               RS-232 channels J1-J3.
```

2.6 GETTING STARTED WITH MOLE

1) The first step is to connect the Brain and personality boards.

   The MOLE system is designed to operate as two, freestanding, connected boards. Figure 2-4 shows an exploded view of the two boards and indicates the alignment of the standoffs used to hold the boards together and off the work surface. It also shows the signal connectors between the boards.

   The 12-inch by 12-inch (30-cm by 30-cm) personality board is supported by 16 half-inch nylon standoffs. The male ends of the 12 male/female standoffs are inserted from the solder side of the personality board and screwed into the twelve female standoffs screwed to the Brain board. Four screws connect the four female standoffs to the front of the personality board. A total of 16 standoffs hold the personality board off the work surface; a total of 12 hold the Brain board off the personality board.

   To connect the boards the steps are:

   a) Attach the 12 female standoffs to the Brain board using the nylon screws. The standoffs go on the bottom of the board.

   b) Mate the 36-pin connectors on the Brain board (J4 and J5) with the corresponding connectors on the personality board. Secure the boards by using the 12 male/female standoffs through the bottom side of the personality board.

   c) Attach four female standoffs to the lower edge of the personality board with nylon screws.

   There are no mounting procedures for the MOLE system itself, but the working surface should be free of metal tools, pieces of wire, or other metallic objects that might cause shorts to the system.

2) Make sure all the jumpers and switches are in their default (i.e. shipping) configuration. It should not be necessary to alter any of these for a preliminary check-out. The default jumper settings are described in Appendix B.

3) Make the RS-232 connections specified in Section 2.5. For an initial check-out, only the connection to the CRT terminal is necessary. The host system connection can be made later.

4) Make the power supply connections as described in Section 2.4.

5) Turn on the +5 volts supply.

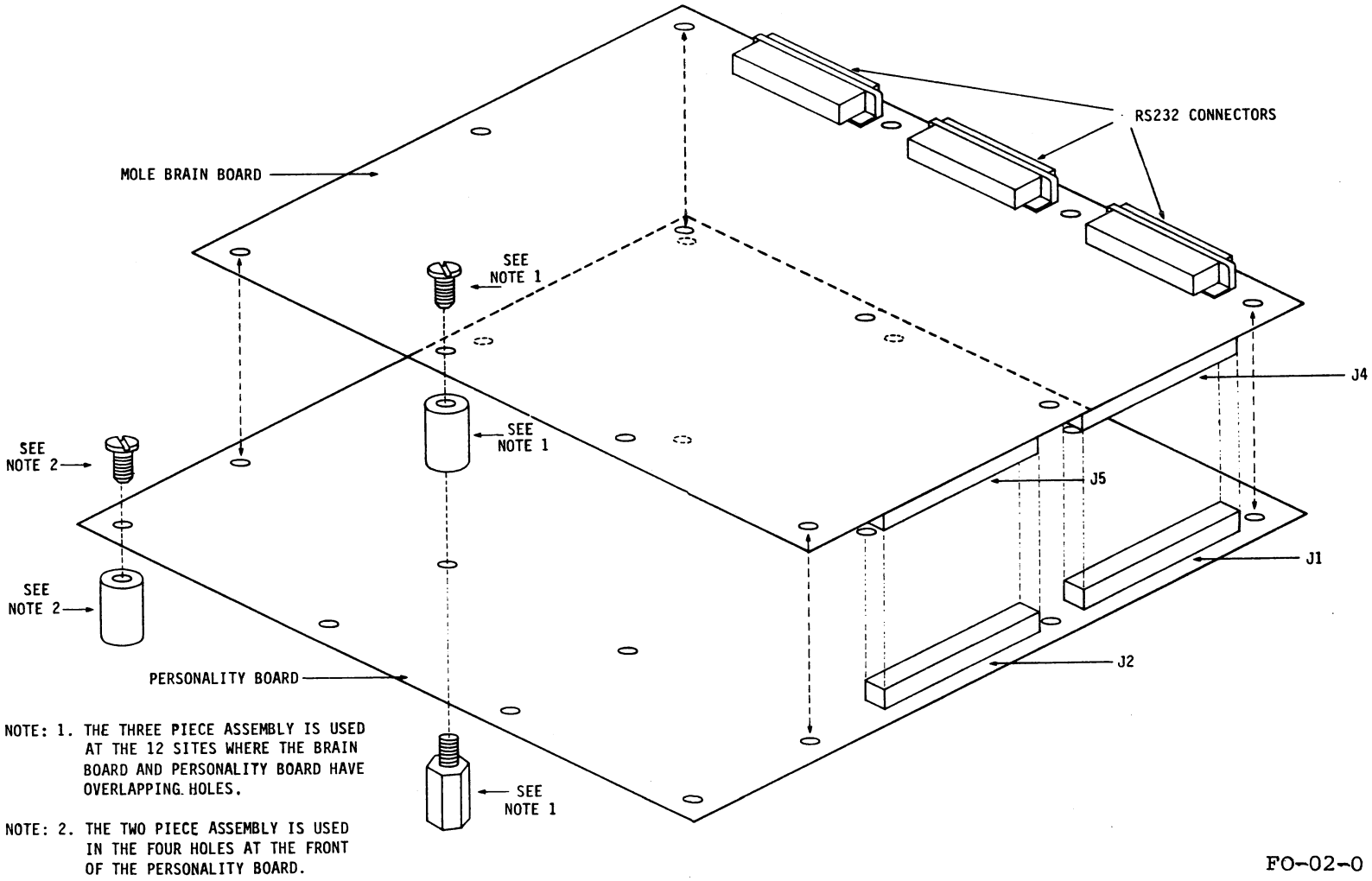


Figure 2-4 Brain Board and Personality Board, Exploded Veiw

When the power supply is on, the MOLE performs two diagnostic tests. These tests check the on-board RAM and the firmware PROMS. If there are any errors, the tests will issue messages. If there are no errors, the tests issue no messages. The tests are described more fully in Appendix E.

6) Press the carriage return on the CRT terminal.

The MOLE system will respond with a message on the CRT:

```
EXEC,REV:x,<date>
X>
```

If this does not happen, push the reset switch (SW1), and try the carriage return again.

The "X>" prompt indicates that you are currently in the MOLE EXEC, which is the "top level" program from which the other programs are invoked.

At this point the initial check-out is complete. A more complete check-out involves invoking and quitting the different programs. The following programs can be invoked from the EXEC:

COMM	-	File transfer program
DIAG	-	Diagnostics
MONITOR	-	Emulation monitor (for emulating COPS, TMP etc.)
PROG	-	Prom programmer

To invoke any of these, type at least the first letter of the program name.

For example:

```
X>PR                    <---- Type "PR" followed by a carriage return.
PROG,REV:A,15 SEP 83

P>                      <---- PROG responds with its signon line, and its
                              prompt, viz., "P>".
```

To exit any of these programs and return to the EXEC, type "END" or "@", followed by a carriage return. For example:

```
P>END                   <---- "P" indicates you are in PROG.  Type "END".

X>                      <---- "X" indicates you are back in the EXEC.
```

All the programs have a HELP command. This will display a summary of the relevant commands for that program.

The user will also want to become familiar with the input-line editing commands described in Chapter 3 and summarized in Appendix D.

Once the MOLE itself has been exercised, the next step is to transfer a file from a host system. It is first necessary to become familiar with the communication program, COMM - both the MOLE version and the host system version. COMM on MOLE is described in Chapter 5 of this manual. Also refer to the chapter on COMM in the appropriate host system manual.

The recommended setup for the first attempt is the configuration shown in Section 5.1.1. This is referred to as Configuration 1 in Section 5.1.4. Also see the sample MOLE session in Appendix C.

Chapter 3

USING MOLE SOFTWARE

3.1 INTRODUCTION

This chapter describes what the user needs to know about the MOLE buffers and serial channnels to effectively operate MOLE firmware. Additionally, there are general rules of operation such as how to invoke programs, and the syntax for entering numbers. This chapter also explains the facilities available for correcting and modifying input lines.

3.2 SYSTEM BUFFERS

The MOLE Brain board firmware maintains a set of variable length buffers in memory. Additionally, personality boards may add buffers. Section 3.2.1 describes these buffers, with an emphasis on the Brain board buffers. Section 3.2.2 describes error messages that may result from the use of the buffers.

3.2.1 The Buffers

There are a total of four buffers; called B0, B1, B2 and B3. They are meant for temporary data storage and as PROM programming buffers. The total space available to the buffers is approximately 55K bytes.

When the buffers are used for downloading object code from a host system, their size is flexible. In this situation, the buffers are best thought of as mini disk files. A buffer will grow as long as there is available space.

When the buffers are used for PROM programming, they function as fixed-size blocks of memory. The buffers can be displayed and altered by the PROM programming utility, PROG.

In addition to the Brain board's own buffers, the personality boards have at least one block of memory on them which functions as a buffer. For example, the COPS CMOS personality board has 2K bytes of shared memory. This block of memory is also considered a buffer by the Brain board firmware. The COMM and PROG utilities have access to this memory, which is referred to as SH. The main difference between SH and the system buffers B0..B3 is that SH has a fixed size.

3.2.2 Buffer Related Error Messages

Commands which access, modify or address MOLE buffers, the Brain board buffers or a personality board buffer, may issue "error" messages. Although the message

indicates some problem with the command, it does not necessarily mean the command failed. For example, assume buffer B0 is 8K bytes in size and that the personality board buffer SH is 2K bytes in size, then the command

```
X>CMPARE B0,,SH
```

will produce the error message:

```
ILLEGAL ADDRESS (BUFFER SH):  0800
```

This is because the COMPARE command starts at both buffers' address 0H and proceeds up. Buffer B0 has a valid address 0800H, but SH does not, (its maximum address is 07FFH) so the command "fails" at that point. The results of the comparison, however, are still valid.

If the command had been entered as:

```
X>CMPARE SH,,B0
```

the same comparison would have been done, but no error message would have been issued.

The messages and their likely cause are:

- BUFFER <bufname> IS EMPTY

  An attempt was made to read an empty buffer.

  Example:

  If buffer B0 is empty (size=0) and the command

  ```
  X>MOVE B0,,B1
  ```

  is entered, the buffer empty message is issued.

- BUFFER <bufname> NOT AVAILABLE

  The buffer is not accessible. As an example, on the NS455 TMP personality board, the buffer SH can be either on the personality board or in off-board memory. If the command

  ```
  X>CMPARE SH,,B0
  ```

  is issued, and SH is off-board, the buffer not available message is issued.

- ILLEGAL ADDRESS (BUFFER <bufname>): aaaa

  This means that an attempt was made to access the address "aaaa" and that that address is outside the address range of the buffer <bufname>.

- OUT OF SPACE (BUFFER <bufname>)

  This message occurs only when writing to one of the Brain board buffers (B0 through B3). The message is issued when there is not enough buffer space available to complete the write. Buffers which are no longer needed can be erased and the command repeated.

- WRITE ERROR (BUFFER <bufname>) AT ADDR: aaaa. DATA EXPECTED/READ=ww/rr

  Buffer writes are followed by reads to verify the write. The write error message is issued when the value read differs from the value written. If the message is issued for a Brain board buffer, then there is a problem with either the Brain board RAM or firmware. If the message is issued for a personality board buffer, the problem may be with the buffer RAM or firmware; alternately the buffer has been assigned to memory on the prototype hardware, but the memory is not actually present on the prototype board.

3.3 RS-232 CHANNELS

The three serial channels are referred to as S1, S2 and S3. S1 is on the Brain board J1 connector, S2 is on J2 and S3 is on J3.

The channel used to log on after power-on or reset becomes the console device (CN). The other two channels are referred to as the serial data and printer channels (SD and LP). The default assignments of these channels are described in Section 2.4.

The significance of the assignment is that:

- The serial channel (SD) is the channel used by the communications utility, COMM, to download object code files from a host system. (COMM can also transfer files over the console channel, but using the serial channel may be more convenient. A complete description of the possible configurations is given in Chapter 5.)

- The firmware assumes the printer channel (LP) is attached to a line printer, and will be used for hardcopy output. This is discussed in Section 3.4.

3.4 GENERAL RULES FOR OPERATING MOLE

- EXEC is the central program. All other programs are either invoked as EXEC commands or are invoked from other programs with EXEC commands. See Chapter 4 for details on EXEC.

- Upper-case characters are used everywhere.

- All input lines are terminated by pressing the Return key.

- Numbers must be entered in hexadecimal or decimal as required by the syntax of the particular command being entered. With one exception, it is not necessary to precede a hexadecimal number with a 0. The exception is a hexadecimal number entered into CALC and which otherwise would start with 1-9; e.g. 9A. It is never legal to use X'nn or nnH notation. For example, to enter the hexadecimal number BF:

```
  BF   is  OK
 0BF   is  OK (leading zeroes are ignored)
X'BF   is  illegal
  BFH  is  illegal
```

  In general, numbers are entered in their "natural" format. For example, addresses and data are always in hexadecimal. Numbers such as time delays and baud rates are entered in decimal.

- Line editing is possible when entering an input line on the MOLE. For example, a backspace character (ctrl-H) will move the cursor back one position. The complete set of line editing control functions is described in Section 3.5.

- When the MOLE is sending characters to the console, output can be temporarily frozen by typing a ctrl-S. Type ctrl-Q to resume output. Pressing any key, except ctrl-S, will abort the output.

- The output of any command can be redirected to the LP device by typing "LP:" at the end of the command line. The "LP:" must be the last entry on the line prior to pressing the Return key.

  For example, the "list" command of the monitor can be used to list the first 256 bytes of shared memory:

```
M>L 0/FF
```

  To get this printed on the LP device, enter the line as:

```
M>L 0/FF LP:
```

- "@" and "@@" can be used as prefixes to move between programs. Their use is described in Section 3.6.

3.5 LINE EDITING

The MOLE has a line editing function which uses control characters to add, delete and insert characters on a command line. The complete set is as follows:

<ctrl-A> Insert characters at current cursor position. The insertion mode is exited when a control key or carriage return is entered.

<ctrl-B> Backspace the cursor to the beginning of the previous word.

<ctrl-D> Truncate line at current cursor position. Characters after the cursor are ignored and the line gets entered into the system as if the Return key had been pressed.

<ctrl-E> Move the cursor to the end of the line. The characters in the line are not affected. If a <ctrl-E> is the first character entered on a line, it will recall the previous command line. This line can then be edited and entered.

<ctrl-F> Forward space the cursor to the beginning of the next word.

<ctrl-H> Backspace one position. The character backspaced over is not removed from the input line buffer.

<ctrl-L> Forward space the cursor by one position. The character moved over is not affected.

<ctrl-Q> Abort the line.

<ctrl-S> Move the cursor to the start of the line. The characters in the line are not affected.

<ctrl-W> Forward space one character beyond the next character typed.

<ctrl-X> Delete the character at the current cursor position.

<ctrl-Z> Forward space to the next occurrence of the next character typed.

Some examples illustrating the line editing capability are given below. A caret (↑) is used to indicate the position of the cursor. The input line is shown as it appears on the terminal. "M>" is the MOLE prompt and is not entered by the user.

Example 1

Suppose the desired line is

```
M>AL 23,4,5,6
```

but gets entered as

```
M>AL 234,4,5,6
          ↑
```

The cursor is at the end of the line.

Enter backspace characters (crtl-H) until the cursor is at the first "4".

```
M>AL 234,4,5,6
      ↑
```

Now remove the "4" by typing a ctrl-X After typing ctrl-X, the line will look like

```
M>AL 23,4,5,6
      ↑
```

Enter the line by pressing the Return key. It is not necessary to move the cursor to the end of the line.

Example 2

Suppose the desired line is

```
M>AL 23,4C,DF,76,FF,20,12
```

But gets typed as
(A "2" was missed).

```
M>AL 3,4C,DF,76,FF,20,12
                        ↑
```

Bring the cursor to the "3". One way is to backspace repeatedly, but it is quicker to move to the start of the line by entering a crtl-S. After typing a crtl-S, the line looks like

```
M>AL 3,4C,DF,76,FF,20,12
  ↑
```

Now, type crtl-L 3 times. The cursor will advance 3 positions, and the line will be

```
M>AL 3,4C,DF,76,FF,20,12
     ↑
```

To insert a "2", first type in a ctrl-A The MOLE is now ready to insert characters at the current cursor position. There is no visible difference in the display.

Type the character to be inserted, which in this case is a "2". The display is now

```
M>AL 23,4C,DF,76,FF,20,12
                        ↑
```

The editing is complete. Just press the Return key.

Example 3

This example shows how the last entered command can be recalled.

Suppose the desired command was

```
M>DE  44,200/3FF
```

but the following line was typed, and the Return key was pressed.

```
M>DE  44,200/3F
```

This particular command is illegal (3F is less than 200) and the monitor will respond

```
ILLEGAL VALUE
M>
 ↑
```

It is not necessary to retype the line. Before typing anything else, type <ctrl-E>. This recalls the last command line and the display will be

```
M>DE  44,200/3F
             ↑
```

Now the line can be edited. In this case, a character has to be appended to the end of the line. So just type "F". The display is now

```
M>DE  44,200/3FF
              ↑
```

Press the Return key to execute the command.

3.6 THE "@" AND "@@" PREFIXES

1) The "@" prefix is used to move from one MOLE program to another. For example, to invoke COMM from PROG, type

```
P>@C        <---- "P" indicates currently in PROG. Enter "@C".

C>          <---- COMM has been invoked, and responds with its
                  prompt.
```

This is quicker than ending PROG and then invoking the program from EXEC. Just entering "@" will return control to the EXEC.

2) The "@@" prefix is used to execute an EXEC command while in another MOLE program. For example, to execute the EXEC command STAT while in PROG, type "@@ST".

Example:

```
P>@@ST          <---- While in PROG, enter the required EXEC command,
                      preceded by "@@".

BUFFER SIZES: B0=0000  B1=0400  B2=0000  B3=0000
BYTES AVAILABLE = 54528   (213 BLOCKS)

P>              <---- The STAT command was executed.  Control remains
                      with PROG.
```

Chapter 4

EXEC COMMANDS

4.1 INTRODUCTION

There are three sets of EXEC commands.

The first set is just the names of the MOLE programs:

COMM	-	Invoke Communication program
DIAG	-	Invoke Diagnostic program
MONITOR	-	Invoke Personality board emulation monitor
PROG	-	Invoke PROM programming program

Section 4.2 describes how to use this set. The COMM program is discussed in Chapter 5. PROG and DIAG are described in Chapter 6 and 7 respectively. MONITOR varies from personality board to personality board, therefore it is described in each personality board's user's manual.

The second set of EXEC commands is:

CALC	-	Adds/subtracts decimal and hex numbers
CMPARE	-	Compares one buffer with another
ERASE	-	Used to erase all or part of a buffer
HELP	-	Prints a summary of EXEC commands
MOVE	-	Moves data from one buffer to another
STATUS	-	Display status of buffers; display and alter RS-232 parameters

This set of EXEC functions is described in detail in Sections 4.2 through 4.7. Any of the second set of commands may be executed from COMM, DIAG, MONITOR or PROG by prefixing the command with "@@".

Set 3 of the EXEC commands is used exclusively in multiple MOLE configurations and is described in Chapter 8.

4.2 INVOKING OTHER PROGRAMS FROM EXEC

The four Brain board programs which may be invoked from the EXEC are: COMM, DIAG, MONITOR and PROG. In all four cases, it is sufficient to type the first character of the program name to specify the program.

Example:

```
X>C                    <---- While in EXEC, type "C".

C>                     <---- COMM has been invoked.
```

NOTES: 1. The first time a program is invoked after power-on or reset, it will print a signon message:

<program name>,REV:x,<date>

This indicates that the variables associated with the program have been initialized and the program is starting "afresh".

2. When a program is subsequently re-entered, it will only print its prompt character. This indicates that the program variables have not been initialized; the program variables will still have whatever values they had when the program was exited.

3. As described in Section 3.6, the programs can be invoked directly from another program by prefixing the program name with "@". Therefore it is not necessary to execute an END command and then invoke the program from the EXEC.

4.3 CALC COMMAND

Syntax: CALC <num> < + | - > <num> ...

This is used to add and subtract numbers. Numbers which begin with "0" or with "A"-"F" are interpreted as hexadecimal. Numbers beginning with "2"-"9" are interpreted as decimal numbers. The result is displayed in both hexadecimal and decimal. This command performs unsigned 16-bit arithmetic.

Example:

```
X>CA  23+010            <---- Adds decimal 23 and hexadecimal 10.

027   (39 DEC)

X>
```

4.4 CMPARE COMMAND

Syntax: CMPARE <buf1>,[<buf1 addr range>],<buf2>[,<start addr>]

This compares one part of <buf1> with <buf2>.

<buf1>	<---- Name of first buffer
<buf1 addr range>	<---- <addr1>/<addr2>
	<---- <addr1>/L ("L" is shorthand for the Last address of buf1).
<buf2>	<---- Name of second buffer
<start addr>	<---- Address of <buf2> where comparision should start. If not specified, this parameter defaults to 0.

<addr1>,<addr2>,<start addr> are entered in hexadecimal.

<buf1 addr range> defines the region of <buf1> to be compared. If not specified, this defaults to the entire buffer.

Examples:

X>CMP B0,100/1FF,B1 <---- Compares locations 100/1FF of B0 with B1. Comparision starts at location 0 of B1. Thus location 100 of B0 is compared with location 0 of B1, 101 of B0 with 1 of B1, and so on, until the address range is exhausted.

X>CMP B0,,B1 <---- Compares all of B0 with all of B1.

X>CMP B0,200/L,B1 <---- Compares B0 with B1, starting at location 200 of B0 and location 0 of B1. Comparision continues until the last location of B0.

The CMPARE command will print out locations which do not match. The displayed line(s) are of the form:

A:<buf1 addr> <buf1>:<buf1 data> <buf2>:<buf2 data>

Examples:

1) X>CMP B0,0/1FF,B1

```
A:0023  B0:04  B1:78
COMPARE DONE

X>
```

<---- This indicates that location 023 of B0 contains 04, whereas the corresponding location (023) of B1 contains 78.

2) X>CMP B0,100/2FF,B2,200 <---- Here comparision starts with B0 location 100 being compared with location 200 of B2. Locations 157 through 150 of B0 are not the same as locations 257 through 259 of B2.

```
A:0157  B0:0C  B2:0D
A:0158  B0:1C  B2:1D
A:0159  B0:3F  B2:FF

COMPARE DONE X>
```

3) X>CMP B0,,B3 <---- No output appears, and therefore all locations of B0 match those of B3. Note that if B3 is larger than B0, the excess locations are not checked, nor is the difference in size noted by the CMPARE command.

```
COMPARE DONE

X>
```

4.5 ERASE COMMAND

Syntax: ERASE <buffer name>[,<start addr>]

where <start addr> is in hexadecimal.

This command will erase all or part of the specified buffer, <buffer name>. If <start addr> is not specified, it defaults to 0 and the entire buffer is erased.

If <start addr> is specified, then only locations <start addr> onwards are erased. Locations less than <start addr> are not affected. If <start addr> is greater than the maximum address of the buffer, this command has no effect.

If <buffer name> refers to a system buffer (B0..B3), then the erased locations are actually deleted from the buffer and the buffer size is reduced. If <buffer name> refers to a personality board memory (e.g. SH - personality board shared memory), then the erased locations are filled with zeroes.

Examples:

1) X>E B2 <---- Buffer B2 is completely erased and all data is removed. B2 is now empty.

2) X>E B2,100 <---- Locations 100 onwards of B2 are erased (i.e. deleted). If B2 had at least 100 locations before the command it now has exactly 100, i.e., the maximum address is 0FFH.

4.6 MOVE COMMAND

Syntax: MOVE <srcbuf>,[<src addr range>],<destbuf>[,<start addr>]

This moves one part of <srcbuf> to <destbuf>.

<srcbuf>	<---- Name of source buffer
<src addr range>	<---- <addr1>/<addr2>
	<---- <addr1>/L ("L" is shorthand for the last address of srcbuf).
<destbuf>	<---- Name of destination buffer
<start addr>	<---- Destination buffer address. Data from the source buffer is moved here. If this parameter is not specified it defaults to 0.

If the <src addr range> parameter is not specified, it defaults to the entire source buffer. "L" indicates the last location of the buffer.

Examples:

1) X>MOV B0,0/FF,B1 <---- Moves locations 0 through 0FF of B0 to locations 0 through 0FF of B1.

X>

2) X>MOV B0,0/FF,B1,100 <---- Moves locations 0 through 0FF of B0 to locations 100 through 1FF of B1.

X>

3) X>MOV B0,100/L,B1 <---- Starts moving data from location 100 of B0 to location 0 of B1. All of B0 is moved.

X>

4) X>MOV B0,,B2 <---- Moves all of B0 to B2.

X>

4.7 STATUS COMMAND

There are four forms of the STATUS command.

- Form 1 (Section 4.7.1) is used to display information about the system buffers.
- Form 2 (Section 4.7.2) is used to display information about the serial channels.
- Form 3 (Section 4.7.3) is used to change the serial channel assignments.
- Form 4 (Section 4.7.4) is used to change whether console output is also sent to the printer.

4.7.1 Display Buffer Sizes

The simplest form of the STATUS command displays buffer sizes.

Syntax: Status

Example:

```
X>ST

BUFFER SIZES: B0=0000  B1=0000  B2=2000  B3=0000
BYTES AVAILABLE = 47360  (185 BLOCKS)

X>
```

This shows:

- How many 256-byte blocks are free and available for buffer use, (in this case, 185).
- How many bytes that translates to. In this case, since there are 185 blocks, that amounts to 185 x 256 = 47360 bytes.
- The sizes of the system buffers. The sizes are displayed in hexadecimal. In this case, B2 is the only buffer which is not empty and it contains 2000H (i.e., 8192) bytes.

4.7.2 Assign/Display RS-232 Parameters

This form of the STAT command is used for setting RS-232 parameters.

Syntax: STATUS <ser chnl> [<parameter>=<value>][,<parameter>=<value>..]

where <ser chnl>::= < S1 | S2 | S3 > ,
and the <parameter>'s and associated <value>'s are:

<parameter>	<value>
BAUD	< 110 | 300 | 600 | 1200 | 2400 | 4800 | 9600 | 19200 >
BITS	< 7 | 8 >
STOP	< 1 | 1.5 | 2 >
PARITY	< EVEN | ODD | OFF >
CHDLY	<dec num, 0-255>
LFDLY	<dec num, 0-255>
CRDLY	<dec number, 0-65535>

S1, S2 and S3 are the three serial channels (S1 is the channel physically on connector J1 etc). The <parameter>'s Baud, Bits, Stop and Parity refer to the usual serial communication parameters.

CHDLY is the delay, in milliseconds, to be inserted after each character.

LFDLY is the delay, in milliseconds, to be inserted after each line feed character.

CRDLY is the delay, in milliseconds, to be inserted after each carriage return character.

These delays can be useful when working with slow devices, particularly printers.

Examples:

1. X>S S3 CR=100,LF=50

Causes delays of 100 msec and 50 msec to be inserted after transmission of carriage return and line feed characters on RS-232 channel S3.

2. X>S S2
   BAUD= 9600 PARITY=OFF BITS=8 STOP=1
   CHDLY= 0 MS CRDLY= 0 MS LFDLY= 0 MS

When no parameters are entered after the serial channel number, the current settings are displayed.

4.7.3 Device Assignment

Syntax: STATUS
SET <logical device> = <ser chnl>

where <ser chnl>::= < S1 | S2 | S3 >
<logical device>::= < CN | SD | LP >

This form of the STATUS command allows the user to change which serial channel is used for the system console, printer, and serial data channel. The console can be assigned to any of S1, S2 or S3, but neither SD nor LP can be assigned to the serial device assigned to the console. When the console (CN) is reassigned, the serial data (SD) and printer (LP) channels are also reassigned according to Table 2-2 in Section 2.4. When SD or LP is reassigned, the channel assignments of the two merely swap. For example, assume the following default assignments are in effect

```
CN = S1
SD = S2
LP = S3
```

Then CN could be reassigned to S2 or S3, but neither SD nor LP could be assigned to S1 (directly). If CN were reassigned to S2, then according to Table 2-2, SD would be automatically reassigned to S1.

Example:

```
X>ST SET LP = S2

CN=S1  SD=S3  LP=S2
X>
```

4.7.4 Printer Output Control

Syntax: STATUS
LP[ON | OFF]

If "STATUS LP ON" is issued, all output being listed on the console is also listed on the printer, allowing the user to make a hard copy of a MOLE system session. "STATUS LP OFF" stops console output from being sent to the printer. If the ON/OFF operand is omitted, then the state of the printer output control is displayed.

Example:

```
X>ST LP ON

X>
```

The above example turned on output to the printer channel (LP).

Chapter 5

COMM

5.1 OVERVIEW OF COMM

The communication program, COMM, is a firmware program which allows data file transfers to and from a host system. The host system is connected to the MOLE through one of the MOLE RS-232 ports.

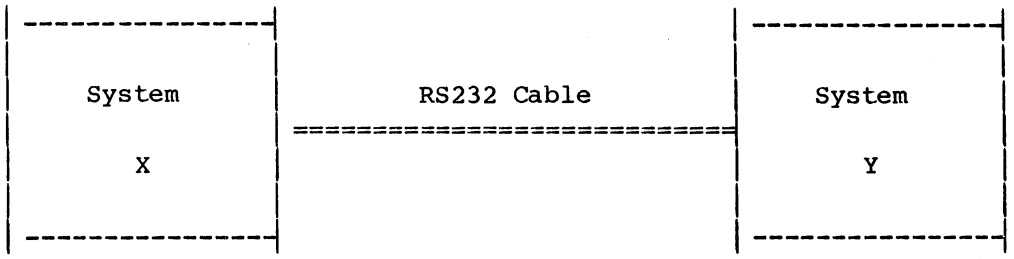
COMM is primarily meant to allow downloading of NSC microcontroller load module files from a host system to a MOLE development station. However, COMM is a flexible program:

- It allows file transfer between the MOLE and any other system running COMM. (COMM is currently available on the COPS PDS and on CP/M-80; so, for example, COMM can be used to transfer files between a COPS PDS and a CP/M system.)
- COMM does its own error checking and transmission retries.
- It permits transfer of both ASCII and binary files.
- It allows the MOLE system to be used as the console for host system. (The MOLE emulates a dumb terminal; see Section 5.1.3, Converse Mode.)

The following sections describe the various configurations in which COMM can operate, with emphasis on connection between a CP/M system and MOLE.

5.1.1 An Introductory Example

Figure 5-1 shows the general setup used to transfer files between two systems X and Y.



FO-03-0

Figure 5-1 General File Transfer Setup

Both systems have asynchronous RS-232 serial ports. (For example, serial channel S2 on the MOLE system, TTY: on an Intel MDS800.) The two ports are connected by a RS-232 cable.

An example of a file transfer is shown below (user input is underlined). Some combinations of systems may require more commands than shown in this example.

On System X (MOLE)		On System Y (CP/M)
X>COMM	(1)	A>COMM (2)
COMM,REV:x,<date>		COMM,REV:x,<date>
C>RE SH XYZ.LM	(3)	#>

Step (1) Invoke COMM on machine X. COMM replies with its signon message and its prompt "C>".

Step (2) Invoke COMM on machine Y, COMM replies with a prompt "#>".

Step (3) Start the actual file transfer. This particular example will cause the MOLE system to Receive file XYZ.LM from the CP/M host and put the data in the shared memory of the MOLE personality board.

NOTES: 1. System Y needs to be executing its COMM program in order for the transfer to take place.

2. The serial port of system Y must be RS-232 compatible:

   a) Both systems must at the same baud rate.

   b) Both systems should have the same settings for parity, number of stop bits and bits-per-character. (In some cases, it is possible to transfer files even if these settings differ.)

   c) The connecting cable and the types of the RS-232 ports must be compatible. See Appendix A.

3) No data transfer or handshaking of any form takes place on invoking COMM. Data exchange starts only when a file transfer command is entered on one of the systems (i.e. in the preceding example, the systems do not communicate with each other until after Step 3 has been entered.

4) In general, the file transfer can be initiated at either system. In the example above, the Step 3 command could equally well have been entered at system Y as "SE XYZ.LM SH" (COMM on system Y sends file XYZ.LM to the MOLE buffer "SH").

Terms describing system configurations are found in Section 5.1.2.

5.1.2 Definitions

"Console" or "CRT" port refers to the system console. This is the physical device (typically S1) which has been assigned as the MOLE console device.

"Serial" port refers to a bidirectional, asynchronous RS-232 port. The MOLE RS-232 channel connected to the host CPU (typically S2).

NOTES: 1. The console and serial ports are different physical devices.

2. COMM always uses the console device to receive commands. It can, however, transfer files over the serial port or the console port. Therefore, it may be possible to transfer files between two systems even if one of them does not have a serial port. This would be a "Type 3" configuration.

As far as using COMM is concerned, systems can be of three different types:

Type 1 This system has two RS-232 ports: the system console, and a peripheral device (serial port). The system is run from an RS-232 CRT terminal connected to the console port.

Examples: MOLE, COPS PDS 2, Intel MDS800

Type 2 A system of this type has a peripheral RS-232 port (i.e. a serial port), but does not have an RS-232 port for the console. Instead, the console is an integral part of the system, or is a video monitor and a keyboard.

Examples: STARPLEX, Intellec Series II

Type 3 This type of system has only one RS-232 port and the port is used as the system console port (i.e. there is no "serial" port).

Examples: COPS PDS I, Dial-up line to mini/mainframe system.

5.1.3 Converse Mode

The converse mode is a "dumb terminal" function in COMM. It allows the MOLE system to act as the console for another system.

Specifically:

1. When the MOLE is put into converse mode, there is a simulated "connection" between the console and serial ports. Characters entered at the console are sent out on the serial port; characters sent in to the serial port are sent out to the console.

2. COMM buffers both character streams. This allows for temporary differences in speed between the two systems.

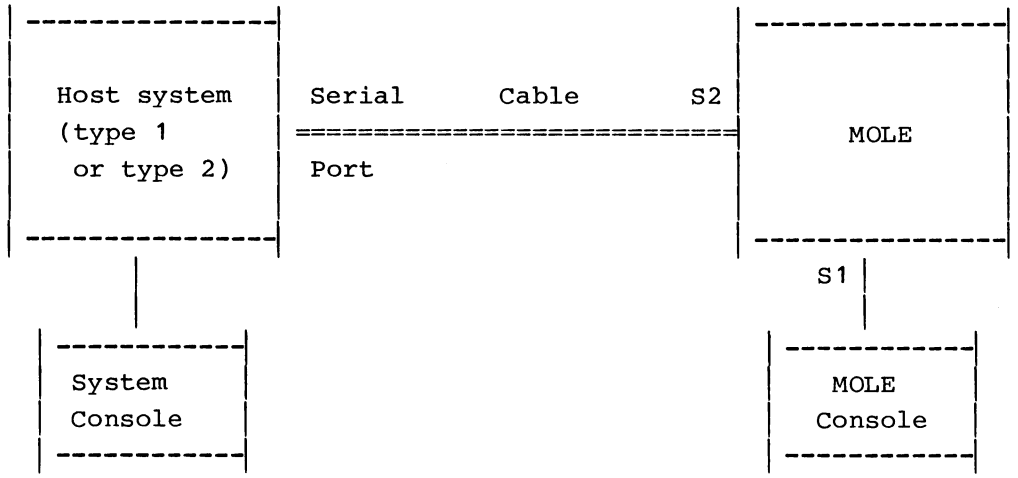
3. While in converse mode, COMM looks for an "escape" character entered at the console. This character causes COMM to exit the converse mode.

5.1.4 System Configurations

Various system configurations are possible, depending on the "type" of host system being used. This section describes the possible hookups with a MOLE development station.

Configuration 1 - Type 1 or Type 2 Systems

The serial ports of the two systems are connected by an RS-232 cable.



Here, MOLE port S1 functions as the MOLE console port and S2 functions as the serial port.

Operation:

1) Each system is run from its own console, independently of the other system. The serial link is used only for file transfer.

2) This is the setup used in the example of Section 5.1.1 - use the file transfer procedure shown there. Converse mode is not used in this configuration.

Advantages:

1) The host system is free for other use once a load module has been downloaded to the MOLE.

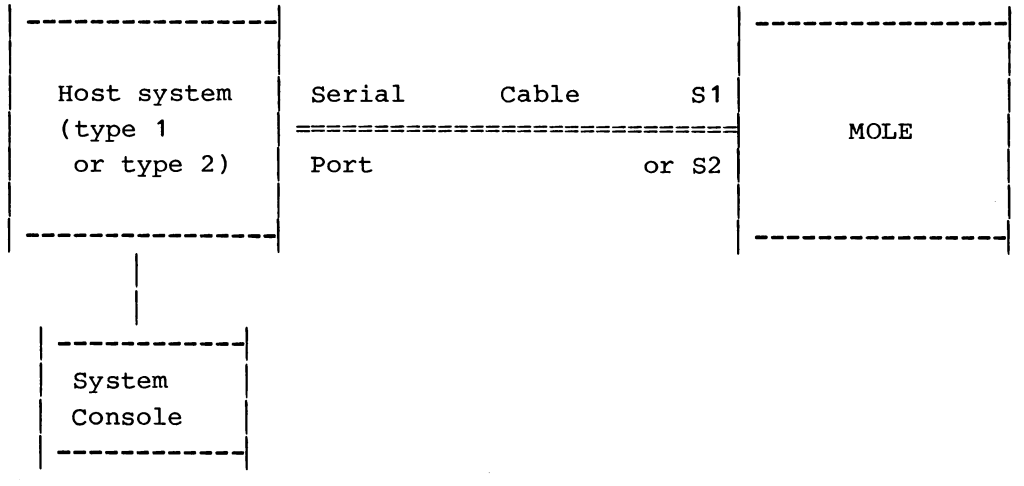
Problems:

1) Need to move between the MOLE and host system consoles for emulation and for editing or assembling source files.

2) A CRT terminal is needed for MOLE.

Configuration 2 - Type 1 or Type 2 Systems

In this configuration, the system serial port is connected to the MOLE console port.



NOTE: On the MOLE, either S1 or S2 can function as the console port. The choice should be made on the basis of port compatibility (see Appendix A).

Here, only the MOLE console port is used.

Operation:

1) The host system is operated as normal.

2) To operate MOLE, invoke COMM on the host system and enter converse mode. The host system console acts as the console for MOLE.

3) To transfer a file:

   a) while in converse mode, invoke COMM on the MOLE

   b) exit converse mode, thus returning to COMM on the host system (COMM on MOLE remains "active".)

   c) enter the file transfer command

Advantages:

1) A separate CRT terminal for MOLE is not needed.

2) No need to move between consoles - all work is done from the host system console.

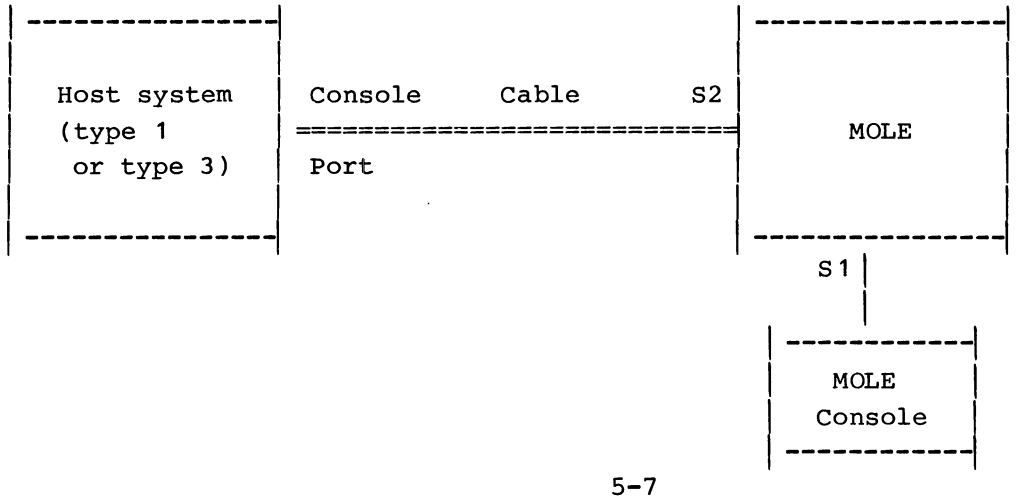
Problems:

1) Host system is tied up when the MOLE is being used for emulation.

2) It is necessary to get a COMM that allows the host system to be put in converse mode for operating MOLE. (On some CP/M systems, it may be necessary to modify the CP/M COMM before this mode of operation is possible.)

Configuration 3 - Type 1 or Type 3 Systems

This is the only possible setup for using a Type 3 system. The host console port connected to the MOLE serial port.



Operation:

1) This setup is the same as configuration 2 shown above, with the MOLE and host systems interchanged. Operation is analogous.

Advantages:

1) No need to customize COMM for the host system hardware.

2) All the work is done at one terminal. The MOLE is put into converse mode for operating the host system.

Problems:

1) The host system is idle while the MOLE is being used for emulation.

2) An escape character is reserved for exiting converse mode on the MOLE. Therefore, this character will not be available for host system operation. (The chances of this being a real problem are remote, since the user can define what the escape character is).

5.2 COMM COMMANDS

COMM commands are entered from the console. The prompt "C>" indicates that the program is ready to accept a new command. Most MOLE COMM commands may be broken down into two main categories: converse-related, and data transfer-related commands. Of the converse-related, only the converse command itself is normally used (Section 5.3). Of the data transfer-related commands, only the SEND and RECEIVE commands are used routinely (Sections 5.6.1 and 5.6.2).

The command formats are listed alphabetically in Table 5-1. The definitions used in command formats are listed in Table 5-2.

TABLE 5-1 COMM COMMANDS

COMMAND	OPERANDS	DESCRIPTION
BREAK		Redefine break key for converse mode.
CTRL	[Y | N]	Enable/disable display of control characters to the console in converse mode.
CMD	<cmd-string>	Send command to other machine.
CONVERSE		Enter converse mode.
ECHO	[Y | N]	Enable/disable console character echo in converse mode.
END		Goto MOLE EXEC.
ESCAPE		Redefine escape character for converse mode.
FRAMING	[Y | N]	Enable/disable UART framing error checking.
HELP		List these commands to console.
MSG	<msg-string>	Send message to other machine.
OVERRUN	[Y | N]	Enable/disable UART overrun error checking.
PARITY	[Y | N]	Enable/disable UART parity error checking.
RECEIVE	<Mole buf>[<opt>[<opt>...]] [<host file>][<host opt>[<host opt>...]]	Receive the file on the other COMM machine into the MOLE buffer.
SEND	<Mole buf>[<opt>[<opt>...]] [<host file>][<host opt>[<host opt>...]]	Send MOLE buffer to the file on other COMM machine.
TIMEOUT	[<short>][,<long>]	Set transfer timeouts.
TRY	[<trys>]	Set error retries/transmission block.

The minimum abbreviations of the commands are shown underlined. When all operands are optionally left off, command responds with present values.

TABLE 5-2 COMMAND OPERAND DEFINITIONS

OPERAND	DESCRIPTION
Y	Yes
N	No
<cmd-string>	Valid command on other machine.
<msg-string>	Message string to send to the other machine's console.
<Mole buf>	Valid MOLE buffer name, B0-B3 or any personality board buffer ("SH" etc.).
<opt>	Option for <Mole buf> Option | Operand | Definition $ASCII | | ASCII file $BINARY | | Binary file $LINES | first line[/last line] | Send indicated lines of ASCII file $OFFSET | [-]<offset> | Address offset Where: <offset> := one to four hex digits. $OVRL | | Don't erase buffer $RANGE | <addr>/<addr> | Limit data to given range Where: <addr> := one to four hex digits.
<host file>	Valid filename on host machine. If file name contains blanks, or starts with $ or ' (single quote) or " (double quote), then the file name must be enclosed in quotation marks (enclosed in double quote marks if name contains single quote marks and visa versa).

TABLE 5-2 (Cont.)

OPERAND	DESCRIPTION
<host opt>	Valid option for host file on other COMM machine.
<short>	Short communication timeout in secs, range 1-10.
<long>	Long communication timeout in secs, range 1-1000.
<trys>	Number of trys for one transmission block, range 1-100.

5.3 CONVERSE COMMAND

In the converse mode, the MOLE CRT acts as the CRT for the other machine. The serial port of the MOLE is tied to the CRT port of the other machine.

Syntax: CONVERSE

The MOLE is put into converse mode, where the MOLE console acts as a terminal to the other machine. This command is used only with System Configuration 3 (Section 5.1.4).

Example (conversing with a CP/M host):

```
C> C

<CR>

-A> COMM

-#> <escapekey>

C>
```

NOTE: Assume character "-" is the converse prompt.

In the example, entering "C" puts the MOLE in the converse mode. Assume a CP/M host computer, with the user logged on drive "A". When the <escape key> is pressed, the MOLE exits the converse mode, returning to the MOLE COMM prompt - (leaving the MOLE in its COMM program ready for communication). The host machine must be running the COMM program in the command mode in order to execute a file transfer from MOLE COMM.

5.4 CONVERSE PARAMETER SPECIFICATION COMMANDS

5.4.1 BREAK

Syntax: BREAK

This command defines the break character. When this console character is entered in the converse mode, a 1-second break is sent over the serial port. This interrupts the output of the other machine if the other machine responds to a break (the MOLE system does not). The MOLE system prompts for the new break character with "BREAK CHARACTER?". If the user enters <cr> in response, the current break character is displayed.

Example:

```
C>BR

BREAK CHARACTER? <cr>

RUBOUT
C>BR

BREAK CHARACTER? <ctrl-A>

CTRL A
C>
```

In this example, the user finds that the current break character is the RUBOUT character. The break character is then changed to <ctrl-A>. The RUBOUT character is the default break character.

5.4.2 ESCAPE

Syntax: ESCAPE

The command defines the escape character. When this escape character is entered in the converse mode, the converse mode is exited (see CONVERSE example). The MOLE system prompts for the new escape character with "ESCAPE CHARACTER?". If the user enters <cr> in response, the system shows the current escape character.

Example:

```
C>BR

ESCAPE CHARACTER? <cr>

CTRL [
C>ES

ESCAPE CHARACTER? <ctrl-A>

CTRL A
C>
```

In the example, the user finds that the current escape character is the CTRL [character. The escape character is then changed to <ctrl-A>. "CTRL [" is the default escape character; on most terminals this key is labelled "ESC".

5.4.3 CTRL

Syntax: CTRL [Y | N]

This switch controls filtering of control characters from serial port in converse mode. Control characters are defined as ASCII characters in the range 0H to 1FH and 7FH; CR and LF characters are excluded. This switch should be turned off only if the CRT displays undesirable control characters. If "Y" or "N" is not entered with the command, the MOLE responds with the current state of the switch. Initially the switch is set to "Y".

Example:

```
C>CT

CTRL Y
C>CT N

CTRL N
C>
```

In the example, the current state of the switch is found to be "Y". The switch is then changed to "N".

5.4.4 ECHO

Syntax: ECHO [Y | N]

This switch controls the echoing of console characters in converse mode. This switch should be set on if other machine does not echo (the MOLE system always echoes). Setting the echo switch from "N" to "Y" is equivalent to changing a CRT from full-duplex to half-duplex. If "Y" or "N" is not entered with the command, the MOLE system responds with the current state of the switch. The switch is initially set to "N".

Example:

```
C>EC

ECHO N
C>EC Y

ECHO Y
C>
```

In this example, the current value of the switch is found to be "N". The switch is changed to "Y".

5.4.5 PROMPT

Syntax: PROMPT

This command controls the prompt character appearing at the start of the CRT line during converse mode. The purpose of the prompt character is to remind users that they are in the converse mode. Setting the character to a " " (blank) disables the prompt. If the user only enters a <CR> with the command, the MOLE responds with the current prompt character. The prompt character is initially a "-".

Example:

```
C>PR

PROMPT CHARACTER? <cr>

C>PR

PROMPT CHARACTER? +

+
C>
```

In this example, the current prompt character is found to be "-". The character is then changed to "+".

5.5 HELP COMMAND

Syntax: HELP

This command lists syntax of all commands to the console.

5.6 DATA TRANSFER COMMANDS

The principal communication commands are RECEIVE and SEND, which are used to transfer files to and from the MOLE system from other COMM machines. MOLE COMM contains translation routines for loading and saving MOLE emulation data. When receiving a file, the received data may be translated from a load module format or ASCII data.

When data is being transferred a "." is put out to the console for each good data block transmitted; this is to show that the system is active. At the end of a successful binary or load module transfer the actual number of file bytes is displayed on the console. At the end of a successful ASCII transmission, the actual number of lines transmitted (number of <cr><lf> sequences) is displayed on the console.

Several categories of errors are possible when receiving or sending a file: the command specification could be incorrect; the transmission itself may have errors and the transfer may be aborted by the user. The error messages and their likely causes are discussed with the receive and send examples. There are additional communication commands which control the error checking/recovery of transmissions (i.e. FRAMING, OVERRUN, PARITY, TIMEOUT, TRY).

5.6.1 RECEIVE

Syntax: RECEIVE <mole buf>[<opt>[<opt>...]]
<host file>[<host opt>[<host opt>...]]

This command is used to receive data into a MOLE buffer from a host file on another COMM machine. When receiving a host file, if the user does not specify otherwise, the received data is unpacked from a load module format, and the unpacked raw data is then placed in the specified MOLE buffer. To receive a raw binary file with no translation the user would specify the "$BINARY" option.

When receiving load module or binary data the user may specify limits on the received data range, this is done with the "$RANGE' option. Only data within the address range following the "$RANGE" will be loaded into the buffer. The "$RANGE" option in conjunction with the "$OFFSET" option will allow users to load portions of very large host files into MOLE buffers for PROM programming. The "$OFFSET" option adds the hexadecimal offset following the "$OFFSET" to the placement address of incoming file data. The offset is accepted as a positive offset unless a "-" is prefixed to the offset, creating a negative offset.

Unless the overlay option is used, a Brain board buffer is "erased" when the first data is received. Whether a personality board buffer is "erased" when it is the destination buffer depends both upon the overlay option and on the particular personality board. See the personality board's user's manual to determine what a particular board does. When a personality board buffer is erased, the memory is set to all zeros.

Although there are no text manipulation firmware utilities in the MOLE, it is possible to load ASCII host files into MOLE buffers. To load an ASCII file the user would specify the "$ASCII" option.

Example:

```
C>R  SH  A.LM

FILE ON THIS MACHINE
MOLE   SH
FILE ON OTHER MACHINE
CP/M  A:A.LM
TRANSMISSION COMPLETE
MOLE   SH
SAVED 0400 HEX BYTES

C>
```

The above example received a load module file into personality board shared memory buffer "SH".

Example:

```
C>R  B0 $R 8000/A000 $O -8000 BIGFILE.LM

FILE ON THIS MACHINE
MOLE   B0
FILE ON OTHER MACHINE
CP/M  A:BIGFILE.LM
TRANSMISSION COMPLETE
MOLE   B0
SAVED 2000 HEX BYTES

C>
```

This loads the buffer B0 with the data contained within the address range 8000 to A000 and relocates the data to begin at address zero within the buffer.

5.6.2 SEND

Syntax: SEND <mole buf>[<opt>[<opt>...]]
<host file>[<host opt>[<host opt>...]]

This command is used to send MOLE buffer data to a host file on another COMM machine. If the user does not specify otherwise, the send-data is packed into a load module format, and the load module data is sent to the host file. The user would specify the "$BINARY" option to send raw binary buffer data with no translation.

When sending load module or binary data the user may specify limits on the sending data range. This is done with the "$RANGE" option. Only data within the address range following the "$RANGE" will be sent to the host file. To complement the "$RANGE" option is the "$OFFSET" option. The "$OFFSET" option adds the hexadecimal offset following the "$OFFSET" to all data placed into the load module or hex file of the host. The offset is accepted as positive unless a "-" is prefixed to the offset, creating a negative offset.

To save ASCII data from a MOLE buffer the "$ASCII" option must be specified. If a specific line range of the ASCII data is to be saved, the "$LINES" option may be used. Only the range of lines specified following the "$LINES" will be saved to the host file.

Example:

```
C>S  B0 $B A.COM

FILE ON THIS MACHINE
MOLE   B0
FILE ON OTHER MACHINE
CP/M  A.COM
TRANSMISSION COMPLETE
MOLE   B0
SENT 0200 HEX BYTES

C>
```

The above example sent the binary image of MOLE buffer B0 to the host file A.COM.

5.6.3 CMD

Syntax: CMD <command-string>

This command sends the command string over the serial port to the other machine to be executed. The other machine will respond to the command as if it were received from its console(the other machine must have its serial port enabled). The command transmission includes error checking.

Example:

1) Sending machine

```
C>CM SPORT N

.....
C>
```

2) Receiving Machine

```
#>.....
CMD: SPORT N
SPORT N
#>
```

In the above example, the MOLE system user disconnects the the CP/M host from the COMM link by sending the CP/M COMM commmand "SPORT N".

5.6.4 MSG

Syntax: MSG <message-string>

This command sends a message string over the serial port to the other machine. The other machine displays the message on its console (the other machine must have its serial port enabled). The message transmission includes error checking.

Example:

1) Sending Machine

```
C>MS I AM DONE

.....
C>
```

2) Receiving Machine

```
C>.....
MSG: I AM DONE
C>
```

5.6.5 Transfer Errors

The transfer errors and transfer abort error messages are given in Tables 5-3 and 5-4.

TABLE 5-3 SEND/RECEIVE OPERAND ERRORS

MESSAGE	DESCRIPTION
BUFFER NAME ERROR	Illegal buffer name
OPTION ERROR	Illegal option
ASCII/BINARY CONFLICT	ASCII/binary conflict between options or between file types on host and the MOLE system. See Note.
BUFFER EMPTY	No data in buffer to send

NOTE: If this message occurs after the "FILE ON THIS MACHINE" message, the error is due to conflicting options on the MOLE. If the message occurs after the "FILE ON OTHER MACHINE" message, the error is due to either a conflict between an option and the host file type, or between file types between the host and MOLE, e.g. the file type on the MOLE is ASCII while the file type on the host is binary.

TABLE 5-4 ABORT ERROR MESSAGES

MESSAGE	DESCRIPTION
USER ABORT	User aborted transfer
BAD LINK UP	No response from host
TRANSMISSION TIMEOUT	No response from host
BAD TRANSMISSION	Unknown host error
PARITY ERROR	UART parity errors
FRAMING ERRORS	UART framing errors
OVERRUN ERROR	UART overrun errors
OTHER MACHINE ABORT	Abort from host
ILLEGAL TRANSMISSION	Bad Transmission data format
FILE FORMAT ERROR	File translation error
INCOMPLETE FILE	Unexpected end of formatted file

5.7 DATA TRANSFER PARAMETER SPECIFICATION COMMANDS

These commands are used only if default transfer parameters are not acceptable.

5.7.1 FRAMING

Syntax: FRAMING [Y | N]

This command controls a switch which enables or disables UART framing error checking during transfer. If "Y" or "N" is not entered with the command, the system responds with the current state of the switch. The switch is initially set to "Y", enabled.

Example:

```
C>FR

FRAMING Y
C>FR N

FRAMING N
C>
```

In the example, the current value of the switch is found to be "Y". The switch is then changed to "N".

5.7.2 OVERRUN

Syntax: OVERRUN [Y | N]

This command controls a switch which enables or disables UART overrun error checking during transfer. If "Y" or "N" is not entered with the command, the system responds with the current state of the switch. The switch is initially set to "Y", enabled.

Example:

```
C>OV

OVERRUN Y
C>OV N

OVERRUN N
C>
```

In the example, the current value of the switch is found to be "Y". The switch is then changed to "N".

5.7.3 PARITY

Syntax: PARITY [Y | N]

This command controls a switch which enables or disables UART parity error checking during transfer. If "Y" or "N" is not entered with the command, the system responds with the current state of the switch. The switch is initially set to "Y", enabled.

Example:

```
C>PA

PARITY Y
C>PA N

PARITY N
C>
```

In the example, the current value of the switch is found to be "Y". The switch is then changed to "N".

5.7.4 TIMEOUT

Syntax: TIMEOUT [[<short>][,<long>]]

This command allows the user to change the transmission timeouts. The short timeout (in seconds) is the maximum time between characters in one transmission block. Normally, it is set to one. The long timeout (in seconds) is the maximum time between transmission blocks, and allows for disk accesses and other delays. The long timeout must be at least twice the short timeout. The timeout is initially set to "TI 1,10". If the command is entered without operands, the current setting will be displayed.

Increasing either short or long timeout will increase transmission retry time. Normally, short timeout should be left unchanged. Long timeout should be be increased if the disk access time or other delays cause retries.

Example:

```
C>TI

TIMEOUT 1,10
C>TI 1,15

TIMEOUT 1,15
C>
```

In the example, the current setting were found to be one second and ten seconds. The long setting is changed to 15 seconds.

5.7.5 TRY

Syntax: TRY [<tries>]

This command sets the number of tries of one transmission block before abort. If <tries> is set to a large value, it will take a very long time for both machines to abort if the communication line is lost. However, if the number of tries is set to a small number (for example, two) the transfer will be aborted when two consecutive errors occur which may result in an unnecessary abort. If no tries are entered with the command, the system responds with the current number of tries.

Example:

```
C>TR

TRYS 16
C>TR 10

C>
```

In the example, the current number of tries is found to be 16. The number of tries is then changed to 10.

5.8 ABORT ERROR MESSAGES

Table 5-4 lists the transmission abort error messages. There are two phases to the transmission. The probable cause of an error message depends upon whether the error occurs during phase 1, in the transition to phase 2, or in phase 2.

First, during the setup transmission, the two machines exchange information about each other. During this phase of the transmission the short timeout is fixed at one second, the long timeout at five seconds; the number of tries is fixed at eight; UART errors are disabled. This part of the transmission occurs just after the transfer is invoked. Aborts during this phase are generally due to a bad communications line or to incompatible UART settings on the two machines.

At the start of phase 2, the user-defined timeouts, tries and UART errors take effect and the message FILE ON OTHER MACHINE is displayed. Aborts during the start-up of phase 2 are usually due to incorrect UART parameters.

During the last part of the transmission, the actual file transfer occurs. Aborts during this phase of the transmission are mostly due to a phone line with marginal quality, or due to actual loss of the line.

In the following examples, note that a "." is shown for each good transmission and an "R" for each retry. The first of the following examples shows an error occuring during phase 1, the second example shows an error occuring during the transition to phase 2, while examples 3 and 4 show transmission errors occurring in phase 2.

1) The following example shows an abort due to a bad telephone line or bad UART setup just after invoking the transfer

```
C>R SH A.LM

FILE ON THIS MACHINE
MOLE  SH
R

BAD LINK UP
TRANSMISSION ABORTED !!!

C>
```

2) The following example shows an abort due to incorrect UART setup between the two machines. Note the 16 retries (assume user set tries=16) show that the user defined values have been exhanged between machines; therefore the communication line is good, but UART parity is wrong.

```
C>R SH A.LM $A

FILE ON THIS MACHINE
MOLE  SH
UART PARITY ERROR
TRANSMISSION ABORTED !!!

C>
```

After the FILE ON THE OTHER MACHINE message, the actual file transfer starts. Aborts then are usually due to bad phone line, loss of line, or a fatal error on the other machine, such as a disk full. After the abort message, a line will state how many lines or bytes (if any) were saved in the receive file, or approximately how many lines or bytes were sent from the send file.

3) This example shows an abort while sending a file, and also shows some retries which did not lead to an abort. Number of lines sent is approximate.

```
C>S SH $LI 10/19 A.SRC

FILE ON THIS MACHINE
MOLE  SH
FILE ON OTHER MACHINE
PDS VOL2:A.SRC
RR...R.RRRRRRRRRRRRR

TRANSMISSION TIMEOUT
TRANSMISSION ABORTED !!!
MOLE  SH
SENT 15 AS LAST LINE

C>
```

4) The following example shows an abort while receiving a file. Number of bytes saved is exact, unlike the Send example above, where it is approximate.

```
C>R SH A.LM

FILE ON THIS MACHINE
MOLE  SH
FILE ON OTHER MACHINE
PDS VOL1:A.LM
BAD TRANSMISSION
TRANSMISSION ABORTED !!!
MOLE  SH
SAVED 0200 HEX BYTES

C>
```

5.9 USER ABORT

During a file transfer the user who invoked the transfer may abort it by hitting the carriage return, and responding to the continue query with "N". After a user abort, allow time for the other machine to timeout before any further communication is attempted.

Example:

```
CONTINUE(Y/N,CR=Y)? N
```

Chapter 6

PROG

6.1 OVERVIEW OF PROG

PROG is a MOLE system program for operating the PROM programmer on the MOLE Brain Board. PROG is used to program PROMS from a MOLE buffer; if a microcontroller program has been developed using the personality board monitor, and is in shared memory, it can be saved on a PROM by using PROG.

The PROM socket (XU1) is a 28-pin zero-insertion-force socket located on the lower left of the MOLE Brain board. The PROM programming voltage (PROM dependent, typically 25 volts or 21 volts at 50mA) must be supplied by the user to connector J8-3. See Chapter 2 for the description of the power supply connector. The associated ground connection is J8-4. The user should only apply programming voltage while programming a PROM. Disconnect the voltage when done. Under no circumstances should the programming voltage be applied when the MOLE system is being powered up or down; this may result in damage to the PROM programmer. Jumper W14 is associated with the PROM programmer: W14-1 and W14-2 should always be connected, and W14-3 left open. When programming 24-pin devices (2716, 2732), the PROM must be offset so that pin 1 of the PROM is inserted in pin 3 of the 28-pin PROM socket XU1.

6.2 GENERAL RULES

1) All operands (addresses, data values and checksums) should be entered, in hexadecimal.

2) PROG always considers one of the data buffers to be in "use". This buffer contains data to be programmed into, or read from, a PROM. The buffer in use can be set to one of the system buffers, B0-B3, or a personality board buffer such as SH (shared memory).

3) The buffer in use must be at least as large as the PROM being programmed. When PROG is first entered, it assumes the buffer in use is B0, and attempts to expand this to a size of 8K bytes. (This size is required for a 2764 EPROM.) There are commands which can be used to expand or reduce the size of the buffer as desired.

4) Under default conditions, PROG will program location 0 of the buffer into location 0 of the PROM and so on. This mapping can

be shifted by using the <base addr> operand of the USE command. For example, defining a <base addr> of 0200 means that location 0 of the PROM will be programmed using physical location 0200 of the buffer, location 1 of the PROM using 0201 of the buffer, etc. Also, a command to list locations 10 through 1F of the buffer will list physical locations 0210 through 021F of the buffer.

The automatic address offset introduced by the <base addr> affects the following commands:

- ALTER
- DEPOSIT
- DUMP
- LIST
- PROGRAM
- VERIFY

The <base addr> does not affect these commands:

- CMPARE
- ERASE
- EXPAND
- MOVE

To minimize chances of error, <base addr> should be specified only when needed, and set back to 0 when no longer needed.

5) Care must be exercised to see that the PROM programming voltage applied is that required by the PROM being programmed. (The circuitry on the MOLE Brain board will cause a drop of 0.2 volts or less between J8-5 and the Vpp pin of the PROM.)

6) After each location is programmed, PROG will verify that it was programmed correctly. Any discrepancy is reported. PROG does not check if the PROM is erased before attempting to program it.

6.3 PROG COMMANDS

The following sections describe the commands and syntax of the commands available in PROG.

Table 6-1 lists the definitions for operands used by the commands.

TABLE 6-1 PROG OPERAND SYNTAX

OPERAND	DEFINITION
<addr>	One to four hex digits
<addr range>	<addr>[/<addr>]
<buf>	B0 | B1 | B2 | B3 or any personality board memory buffer.
<value>	Hex number in the range 0/FF

6.3.1 ALTER

Syntax: ALTER <addr>[[,[<value>][,[<value>]]]...],<value>

Alter the contents of consecutive data buffer locations to the specified hexadecimal values, beginning at the specified address. Consecutive commas will increment the current address pointer, leaving the data at these locations unaltered.

Example:

```
P>A 10,60,,44

P>
```

The example places 60 in location 10, leaves location 11 unchanged, and places 44 in location 12.

6.3.2 CHIP

Syntax: CHIP [2716 | 2732 | 2764 | 2816]

This command allows the user to display and change the PROM chip type that system is configured for dumping and programming. If no chip number is specified the current chip number is displayed.

Example:

```
P>CH 2732

PROM 2732
P>
```

This sets the programmer for 2732's.

6.3.3 CMPARE

Syntax: CMPARE <source-buf>,[<addr range>],
<target-buf>[,<start-addr>]

This compares the source buffer with the the target buffer. Only the address range is compared; if no address range is given, the entire source buffer is compared. If no start address is given, the comparision begins at zero of the target buffer, otherwise the comparision begins at the address given.

NOTE: This command is identical to the CMPARE command of the EXEC. See Section 4.4 for more details and more examples.

Example:

```
P>CMP B0,100/2FF,B2,200

A:0157  B0:0C  B2:0D
A:0158  B0:1C  B2:1D
A:0159  B0:3F  B2:FF
COMPARE DONE
P>
```

In the example, the comparision starts with buffer B0 location 100 compared with buffer B1 location 200. Therefore, locations 157 through 159 of B0 differ from locations 257 through 259 of B1.

6.3.4 DEPOSIT

Syntax: DEPOSIT <value>,<addr range>

Copies the value to each location in the specified address range of the use-buffer.

Example:

```
P>DE FF,0/200
P>
```

The above example copies 0FF into use-buffer locations 0 through 0200.

6.3.5 DUMP

Syntax: DUMP

Dumps the PROM into the use-buffer. The buffer's size must be equal to, or greater than, the size of the PROM being dumped. (See EXPAND Command, Section 6.3.8). The checksum displayed is the 16-bit sum of all of the bytes of the PROM.

Example:

```
P>DU

CHECKSUM = 3857
P>
```

6.3.6 END

Syntax: END

Exit to MOLE EXEC.

Example:

```
P>E

X>
```

6.3.7 ERASE

Syntax: ERASE <buf>[,<start addr>]

If buffer <buf> is a system buffer B0 through B3 then the command deallocates the specified buffer's memory. If buffer <buf> is SH (shared memory), then the command zeros all of shared memory.

NOTE: This command is identical to the ERASE command of the EXEC. See Section 4.5 for more details and more examples.

Example:

```
P>ER B0

P>
```

The above example erased buffer B0.

6.3.8 EXPAND

Syntax: EXPAND [<addr>]

This command is used to open the size of the current use-buffer to the address specified. This command will only expand the system buffers B0 through B3. If no <addr> is specified then the name, base and size of the current use-buffer is printed.

Example:

```
P>EX 3FFF

BUFFER IN USE: B0 0800 . MAX ADDR = 37FF
P>
```

This example opened the size of the use-buffer B0 to 3FFF and showed the buffer had a non-zero base of 0800.

6.3.9 HELP

Syntax: HELP

Prints out the command summary.

6.3.10 LIST

Syntax: LIST [<addr range>]

List use-buffer data, in hex, in the range specified. The default range is the current address to current address +256.

Example:

```
P>L 0/5

0000  00 44 60 33  51 0F
P>
```

6.3.11 MOVE

Syntax: MOVE <source buf>,[<addr range>]
,<destination buf>[,<store addr>]

Move the source buffer data in the address range specified to the destination buffer and start storing the data at the start address. If an <addr range> is not specified, then all of the source buffer is assumed. If the <store addr> is not specified, then zero is assumed.

NOTE: This command is identical to the MOVE command of the EXEC. See Section 4.6 for more details and more examples.

Example:

```
P>MO B0,10/30,B1,5

P>
```

In this example the contents of buffer B0 locations 10 through 30 were placed in locations 5 through 25 of buffer B1.

6.3.12 PROGRAM

Syntax: PROGRAM [<addr range>]

Programs the PROM from use-buffer. The range option allows the user to program single bytes or a range of bytes within the PROM. If no range is specified, the entire PROM is programmed.

The checksum displayed is the 16-bit sum of the bytes actually programmed.

Example:

```
P>P

INSERT PROM IN SOCKET AND APPLY Vpr. READY (Y/N,CR=YES)? <CR>
PROGRAMMING OVER. CHECKSUM = 0 1AC
** REMOVE Vpr AFTER PROGRAMMING
P>
```

6.3.13 USE

Syntax: USE <buf>[<,base addr>]

Set or display the buffer that will be used for PROG data operations. The command will automatically expand the buffer to 1FFF. The base address is an offset that will be added to all address calculations in the following commands:

- o LIST
- o ALTER
- o DUMP
- o PROGRAM
- o VERIFY

If no base address is specified then zero is assumed.

Example:

```
P>US B0,1000

BUFFER IN USE: B0 1000 . MAX ADDR = 1FFF
P>
```

This example set the use-buffer to B0 with an offset into the buffer of 01000 and opened the buffer to an address size of 01FFF.

6.3.14 VERIFY

Syntax: VERIFY

Compare the PROM to the use-buffer. Display all addresses which the data differ.

Example:

```
P>V

VERIFY DONE
P>
```

This verified that the current use-buffer completely matched the PROM.

Chapter 7

DIAGNOSTICS

7.1 INTRODUCTION

DIAG is a MOLE system firmware program which tests the on-board ROM, RAM, and RS-232 channels J2 and J3.

There are three main tests:

- ROM test
- RAM test
- RS232 test

The RAM test consists of four subtests:

- Address test
- Pattern test
- Retention test
- Word test

To invoke DIAG from the EXEC, type:

```
X>DIAG

MOLE BRAIN BOARD DIAGNOSTICS, REV:x,<date>
D>
```

The command syntax is summarized in Section 7.2. Details of the tests are described in Section 7.3.

7.2 DIAG COMMAND SUMMARY

The minimum recognized abbrevation for a command or parameter is underlined.

ALL	[/C] [/<num>]
END	
HELP	
RAM	[W] [A] [P] [R] [/C] [/<num>]
ROM	[/C] [/<num>]
RS232	[/C] [/<num>]

Where:

/C	Causes the test to continue after errors; the default is to stop after the first error.
/<num>	Number of times to repeat the test; the default is once.
A	Address test
P	Pattern test
R	Retention test
W	Word test

7.3 DIAG COMMANDS

The options "C" and <num> must be preceded by "/". A space or a comma can be used to separate options.

NOTES:
1. Performing a RAM test (using the RAM or ALL commands) will destroy the system buffers B0..B3 contained in Brain board memory. After the diagnostics have been run, the system buffers must be initialized by resetting the MOLE (press switch SW1). Data in personality board shared memory is not affected.

2. The RS-232 test can be performed only when serial channel S1 (RS-232 connector J1) is the system console.

7.3.1 ALL

Will invoke the three main tests: ROM, RS-232 and RAM. All four RAM subtests are performed. If "/C" is given along with the command, testing will continue even when errors occur. The default is to stop after the first error. The /<num> option can be used to specify the number of times to repeat the test sequence.

7.3.2 END

Will return to EXEC.

7.3.3 HELP

This command will display a summary of DIAG commands.

7.3.4 RAM

If the "RAM" command is given without any options, all of the subtests will be executed. If there are any RAM errors, the message "RAM TEST ERROR" will be sent to the console. The RAM test is composed of four subtests, which are explained below.

NOTE: Each subtest tests all 64K of Brain board memory.

The continuous test option ("/C") does not affect the operation of each subtest. When a subtest detects an error, it will print an error message and stop. The /C option determines whether or not to move on to the next test/subtest.

1) RAM subtest - Word

This test fills all of memory with a certain value, e.g. 00H, and then verifies each location. The test is repeated with different patterns, for total of 20 values.

The values are:

00H	55H	01H	02H	04H	08H	10H	20H	40H	80H
FFH	AAH	FEH	FDH	FBH	F7H	EFH	DFH	BFH	7FH

If there are no errors, the message "WORD TEST PASSED" is displayed. If an error is detected, the address of the error, expected data and actual data are displayed.

Example:

```
WORD TEST AT 0100H, EXPECTED/RECEIVED: 00/80
```

2) RAM subtest - Address

This fills memory with a non-zero incrementing pattern (01H-0FFH), and then verifies it. If there is no error, the message "ADDRESS TEST PASSED" is displayed. If a mismatch is found, the address of the error, the expected data and actual data are displayed.

Example:

```
ADDRESS TEST AT 0100H, EXPECTED/RECEIVED: 01/FE
```

3) RAM subtest - Pattern

This test fills memory with 00H, verifies the volume, writes the data's complement (0FFH), and verifies it then the following steps are performed:

- Read data complement in maximum location (0FFFFH).
- Write data to the minimum location (initially 0000H), and verify it.
- Repeat steps 1 and 2 for the area under test, with minimum locations 0,1,2... and maximum locations 0FFFFH, 0FFFEH....

The entire process is then repeated with an initial data pattern of 0FFH. If there is no error, the message "PATTERN TEST PASSED" is displayed.

If any mismatch occurs at any point, the program will display the address where the error occurred, the expected data and the actual data.

Example:

```
PATTERN TEST AT 0100H, EXPECTED/RECEIVED: 00/FF
```

4) RAM subtest - Retention

This test checks the refresh circuitry on the Brain board. Memory is filled with 55H, and the data is verified after a delay of a few seconds. The test is repeated with a pattern of 0AAH.

The message "RETENTION TEST PASSED" indicates that there was no error, otherwise a message of the form

```
RETENTION TEST AT 0100H, EXPECTED/RECEIVED: 55/00
```

is displayed.

7.3.5 RS-232

In this test J3 is configured as an RS-232 data set, and J2 and J3 are connected through an RS-232 cable. A set of characters is sent to J3, the data is complemented and sent back to J2. If there is an error, the appropriate error message is sent to the console. (Refer to Appendix B for the jumper settings required to configure J3 as a data set.)

Example:

```
D>RS

CONFIGURE J3 AS A DATA SET AND CONNECT J2 TO J3
ENTER <CR> WHEN READY
<cr>

SUBTEST 1:  J2=TX  J3=RX
SUBTEST 2:  J2=RX  J3=TX
D>
```

(J2=TX means that the subtest transmitted through J2 and received through J3.)

The configure message is given to ensure that the user connected the J2 and J3 ports. If errors occur, one of the following messages will be sent to the console:

```
*** ERROR: RECEIVER NOT READY
```

and/or

```
*** ERROR: TRANSMITTER NOT READY
```

and/or

```
*** ERROR:  TX DATA:<data>   RX DATA:<data>
```

7.3.6 ROM

This test will calculate the checksum of each firmware PROM on the Brain board and compare it with the checksum stored in the PROM. If the values disagree, a message giving the PROM-number of the defective PROM is sent to the console. No message is displayed if there are no errors.

Chapter 8

MULTIPLE MOLE OPERATION

8.1 INITIAL SETUP AND RULES

There are two configurations in which more than one MOLE system can be hooked to a development system. These rules apply to both configurations:

1) The MOLE system nearest the development system is called M1, the next M2 and so on. Each MOLE system must have Brain board SW2 (Switch 2) set to indicate its number. A maximum of four MOLE systems can be connected. Positions 1, 2 and 3 of SW2 are used to indicate the MOLE system number. On the board, position 1 is labelled "A2", position 2 is labelled "A1", and position 3 is labelled "A0". Also, each position is marked "1" and "0" to indicate the setting.

   The required settings of SW2 are:

MOLE	A2 (Pos 1)	A1 (Pos 2)	A0 (Pos 3)
M1	0	0	1
M2	0	1	0
M3	0	1	1
M4	1	0	0

   Here, 0 and 1 refer to the markings on the board and not to markings on the switch package itself. Leave positions 4 through 8 of SW2 unchanged.

2) S3 on all of the MOLEs should be configured as a data set. (This is the shipping configuration, see Appendix B.)

3) Serial channel S2 of M1 must be connected to the host system serial port. S3 of M1 is connected to S2 of M2, S3 of M2 is connected to S2 of M3 and so on. Use 25-wire cables to make these connections.

4) Setting up SW3 (Switch 3)

   - On M1, switch positions 1, 2, 3, 4 must be OFF, positions 5, 6, 7, 8 must be ON.
   - On the last MOLE system in the chain, switch positions 1, 2, 3, 4 must be ON; positions 5, 6, 7, 8 must be OFF.

- On all MOLEs in-between the first and the last, all eight positions of switch 3 must be ON.

Here, "ON" and "OFF" refer to the markings on the DIP switch package.

Switch 3 is to enable signals onto the RS-232 connectors J2 and J3. These signals are meant exclusively for multiple MOLE operation and are not standard RS-232 signals.

CAUTION

Double check the switch settings described in 1 and 4. Also, set all positions of SW2 and SW3 to their default settings when the MOLE system is no longer being used in a multiple MOLE configuration. In particular, make sure that all eight positions of SW3 are set to OFF, since there is a possibility of damage if some positions of SW3 are ON and the MOLE system is connected to a development system.

8.2 COMMANDS

There are three commands associated with multiple MOLE operation: CONNECT, DISCONNECT and IDENT. The operation of the commands varies according to the user's hardware configuration. For detailed understanding of the commands the user is directed to the configuration examples.

8.2.1 CONNECT Command

Syntax: CONNECT < SYS | M1 | M2 | M3 | M4 >

This command will connect the user to the requested system if it is available.

8.2.2 DISCONNECT Command

Syntax: DISCONNECT

This disconnects the MOLE system from a connected system.

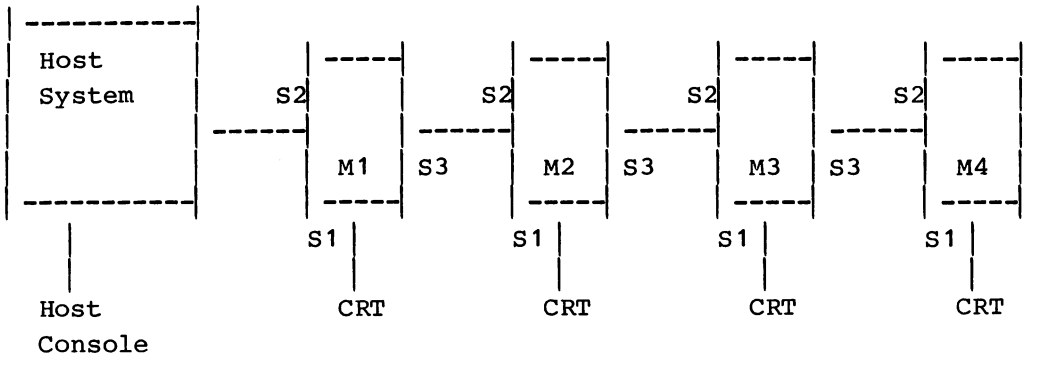
8.2.3 IDENT Command

Syntax: IDENT

This command is used to identify a MOLE system. The MOLE system identifies itself by displaying M1 or M2, etc.

8.3 MULTIPLE MOLE - CONFIGURATION 1

In this scheme, each MOLE is operated from its own CRT. The host system is accessed only to download object code files.



In this configuration each MOLE is operated from its own CRT terminal.

To transfer a file to, or from, the host system:

1) Make sure COMM is running on the host system

2) Enter the following command on the MOLE:

   X>CONNECT SYS

3) If the bus is free (i.e. not being used by another MOLE system), the MOLE system responds:

   SYS CONNECTED

   Now a file can be transferred normally using COMM's receive or send commands.

   Once the file transfer is over, the following command should be entered:

   X>DISCONNECT

   DISCONNECT DONE

   This will free the bus for other users to gain access to the host system.

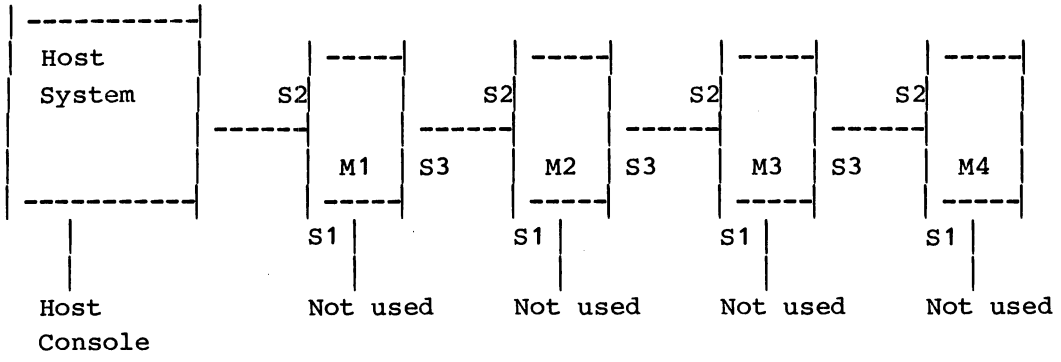
4) If the bus was not free when the CONNECT command was entered (step 2), the response will be:

SYS NOT AVAILABLE (Mi)

Where Mi indicates which MOLE system is currently using the system.

8.4 MULTIPLE MOLE - CONFIGURATION 2

In this scheme, the MOLE stations are operated by a single person from the host system console.



The host system is used in the normal fashion, from its console.

To operate M1:

1) Invoke COMM on the host system,

2) Enter the converse mode on the host. Now M1 can be operated from the host console. S2 acts as the M1 console. Object code files can be downloaded to M1 to perform emulation.

3) To operate any of the other MOLE stations (e.g. M3) type

-X>CONNECT M3	<---- Command to MOLE M1
-<CR>	<---- Enter a carriage return
-X>	<---- This is the prompt from M3 (if it was in EXEC when disconnected)

To confirm that M3 is indeed being accessed, enter

```
-X>IDENT

-M3                 <---- MOLE M3 identifies itself.
-X>
```

M3 can now be operated as usual. Any other MOLE station can be similarly accessed by using the CONNECT command.

4) The DISCONNECT command is not used in this configuration.

Appendix A

RS-232 INTERFACING

This appendix describes the RS-232 cable required to connect systems for file transfer using COMM. The type of cable needed depends on the two RS-232 ports being hooked together.

RS-232 ports can be of two types - Data Terminal Equipment (DTE) or Data Communication Equipment (DCE). Signals are sent or received on different pins depending upon whether the port is a DCE or DTE; for example a DTE transmits its data on pin 2, while a DCE receives its data on pin 2.

Table A-1 lists the relevant RS-232 signals, the conventional pin number to which the signal is assigned, and the direction of data flow. ("To DCE" means that the signal is an input to a DCE type port, and therefore an output of a DTE port.)

In general, to determine how a particular RS-232 port has been wired, refer to the manufacturer's schematics or documentation. However, Table A-2 lists the RS-232 ports for some systems.

The standard assumes that a DCE port will connect to a DTE port; the cable for this application is an ordinary (one-to-one) cable. However a cable can be made to connect like terminal ports if it transposes some of the signals. Such a cable is shown in Figure A-1. This cable is adequate for connecting two similar ports on any combination of MOLE, STARPLEX or COPS PDS.

Not all "RS-232" ports use all of the RS-232 signals. If one port uses the "request to send" and "clear to send" signals while the other does not, an inverting cable like that shown in Figure A-2 must be used. In this case, the CTS and RTS pins on each device are shorted together. (The cable in Figure A-2 is also an example of an inverting cable.)

TABLE A-1 RS-232 SIGNALS (25-PIN CONNECTOR)

RS-232 SIGNALS (25-PIN CONNECTOR)		
PIN	SIGNAL NAME	DIRECTION
1	Chassis Ground	-
2	Transmitted Data	To DCE
3	Received Data	From DCE
4	Request To Send (RTS)	To DCE
5	Clear To Send (CTS)	From DCE
6	Data Set Ready (DSR)	From DCE
7	Signal Ground	-
20	Data Terminal Ready (DTR)	To DCE

TABLE A-2 RS-232 PORTS ON SEVERAL MACHINES

RS-232 CONNECTOR	IS WIRED AS
PDS/PS2 CRT port	Data Set (DCE)
PDS2 Serial port	- See Note -
MOLE port S1	Data Set (DCE)
MOLE port S	Data Terminal (DTE)
MOLE port S3	- See Note -
STARPLEX Serial port	- See Note -
Intel MDS800 TTY:	Data Set (DCE)
Intel Series II TTY:	Data Set (DCE)

NOTE: This port can be reconfigured using jumpers. (Refer to the appropriate hardware manual.) The default wiring (i.e. as shipped) is for the port to resemble a Data Set (DCE).

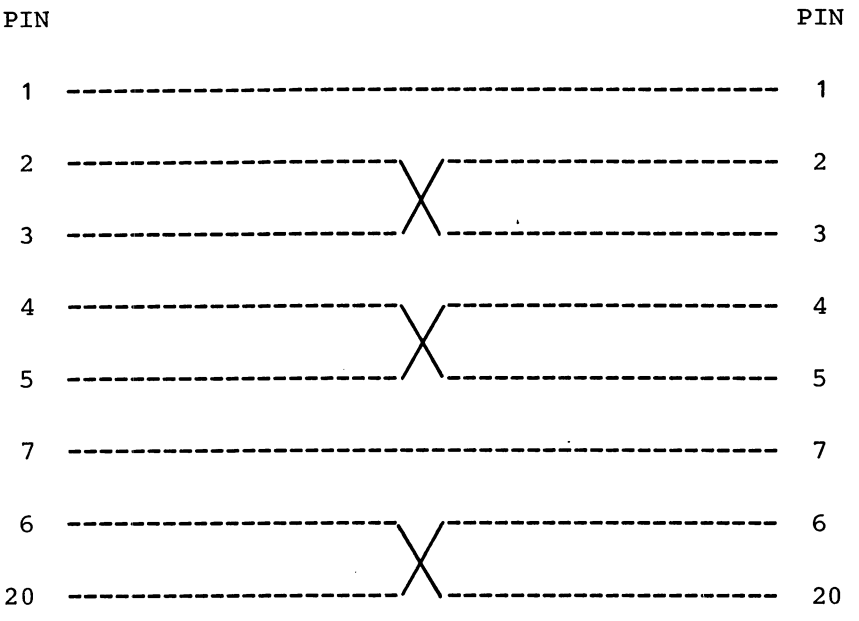


Figure A-1 Inverting Cable

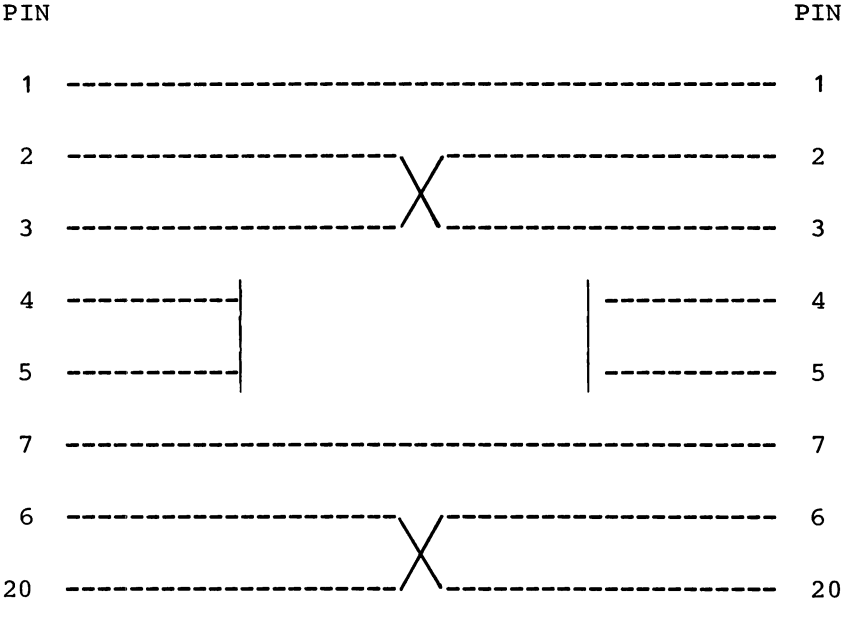


Figure A-2 Inverting Cable - RTS and CTS Shorted

Appendix B

JUMPERS AND SWITCHES

B.1 JUMPERS

Table B-1 lists the MOLE jumpers (except jumpers W6, W7 and W8, see Section B.2). The locations of these jumpers are shown in Figure B-1. Default jumper settings are shown underlined. (The board is shipped with all jumpers in their default settings.)

B.2 J3 CONFIGURATION (W6, W7, W8)

Jumpers W6, W7 and W8 determine if J3 is configured as a data set or data terminal device. The default setup is as a data set. Table B-2 lists the jumper settings for W6, W7 and W8. Figure B-1 shows the locations of jumpers W6, W7, W8.

To configure J3 as a data terminal device,the required settings are:

W6: 1-3, 2-4
W7: 1-3, 2-4
W8: 1-3

B.3 SWITCHES

SW1 This pushbutton switch is used to reset the MOLE system.

SW2 Default setting: All eight positions OFF. Positions 1, 2 and 3 are used to define the MOLE number for multiple MOLE operation as decribed in Section 8.1. Positions 4-8 are not used.

SW3 Default setting: All eight positions OFF. These switches are used exclusively for multiple MOLE operation and should be left off unless needed. See Section 8.1. Positions 1-4: When ON, enable signals onto J2 pins 18, 9, 10 and 11. Positions 5-8: When ON, enable signals onto J3 pins 18, 9, 10 and 11.

TABLE B-1 JUMPER FUNCTIONS

JUMPER	SETTING	FUNCTION
W1	ON	This connects an on-board generated -7 volts supply to the devices requiring them. This jumper is for test purposes only & should not be removed.
W2	1-3	Controls CTS/ input of UART for S1. In this position, the CTS/ input is grounded.
	1-2	CTS/ input of the UART is connected to the RTS signal from J1.
W3	ON	Connects Chassis Gnd to Signal Gnd on J1. (i.e. J1-1 to J1-7).
W4	ON	Connects Chassis Gnd to Signal Gnd on J2.
W5	ON	Connects Chassis Gnd to Signal Gnd on J3.
W6,W7,W8		These jumpers are used together. See Section B.2, "J3 Configuration," below.
W9	1-3	Controls CTS/ input of UART for S2. In this position,the CTS/ input is grounded.
	1-2	CTS/ input of the UART is connected to the CTS signal from J2.
W10	1-3	Controls CTS/ input of UART for S3. In this position,the CTS/ input is grounded.
	1-2	CTS/ input of the UART is connected to the RTS signal from J3.
W11	ON	This is for test purposes only and should not be removed.

TABLE B-1 (Cont.)

JUMPER	SETTING	FUNCTION
W12	ON	This is for test purposes only and should not be removed.
W13	ON	This is not used at present, and should not be removed.
W14	1-2	Feeds +5 volts to XU1-pin 26.
	1-3	Feeds a logic signal to XU1-pin 26. This jumper setting is not used at present.

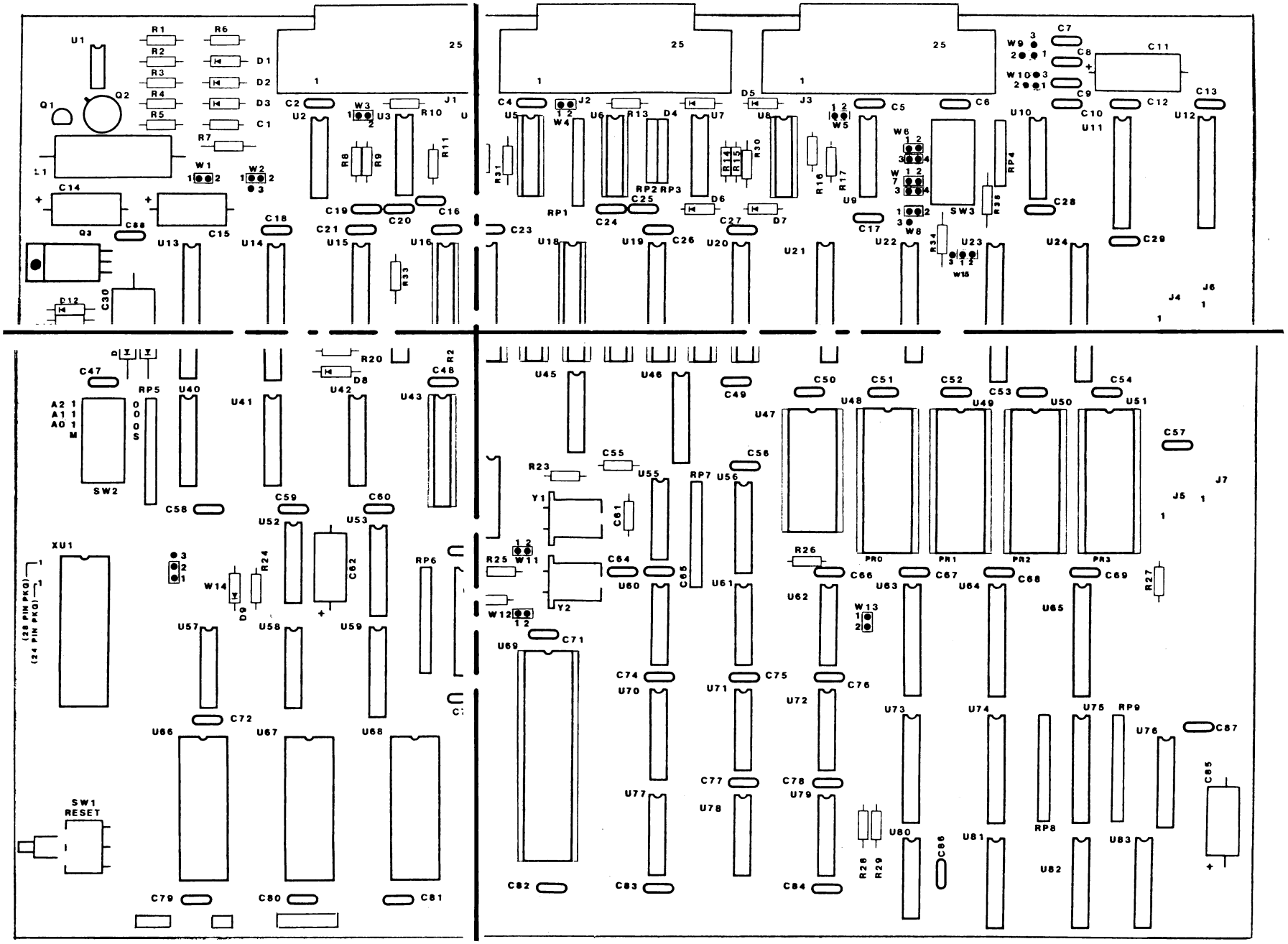


Figure B-1 MOLE Brain Board Jumper Locations FO-04-0

Appendix C

A SAMPLE MOLE SESSION

This is a brief example of a development session with the COPS CMOS MOLE System and a CP/M host. In the session a program is developed for a COPS chip, the program is assembled, transfered to the MOLE system, tested and burned into a PROM.

First enter the program. This program will clear all the RAM digits to zero of a COP420C. The CP/M editor ED.COM is used to enter the program into the source file SAMPLE.MAC.

```
A>ED SAMPLE.MAC
NEW FILE
:*I
1:      .CHIP   424C
2:              CLRA            ; CLEAR B REG
3: NEXTREG:
4:              XABR            ; STORE UPDATED B
5: CLRDIG:
6:              CLRA
7:              XIS             ; STORE 0 IN DIGIT
8:              JP    CLRDIG    ; SKIP IF END OF REG
9:              XABR            ; INC B REG
10:             AISC  13        ; ADD 1 AND SKIP REG = 3
11:             JP    NEXTREG
12: STOP:
13:             JP    STOP
14:↑Z           .END
15:

:*E
```

The COP400 cross assembler ASM400.COM is used to assemble the source and create the load module SAMPLE.LM.

```
A>ASM400 SAMPLE/L

COP400 CROSS ASSEMBLER,REV:A, 22 AUG 83

END PASS 1
END PASS 2

NO ERROR LINES
```

To transfer the load module to the MOLE System the COMM.COM program is invoked on the CP/M host and the user moves to the MOLE console. At the MOLE station, the user enters the MOLE COMM program and enters the command to receive the load module into emulation memory.

```
A>COMM

COMM REV:A, JULY 22,1983

#>SP Y

#>
```

At this point the user now moves to the MOLE system to complete the session.

```
EXEC,REV:x,<date>
X>COMM

COMM,REV:x,<date>
C>RE SH SAMPLE.LM

FILE ON THIS MACHINE
MOLE  SH
FILE ON OTHER MACHINE
CP/M SAMPLE.LM
TRANSMISSION COMPLETE
MOLE  SH
SAVED 0009 HEX BYTES
```

With the program in shared memory the user moves to the COPS CMOS MONITOR and performs a trace and breakpoint.

```
C>@MON


COPS CMOS MONITOR, REV:x, <date>
CHIP NUMBER (DEFAULT=424C) ? <CR>


CHIP BEING EMULATED: COP 424C


M>R


CHIP IS RESET
M>TR 001
```

```
TRACE ENABLED
ADDR: 001  OCCUR: 1  PRIOR: 0  GO: N
M>G

TRACED AT A:001


M>TY 0/10

  0    0 A:001      E:11111111
  1    1 A:002      E:11111111
  2    2 A:003      E:11111111
  3    3 A:004      E:11111111
  4    4 A:002      E:11111111
  5    5 A:003      E:11111111
  6    6 A:004      E:11111111
  7    7 A:002      E:11111111
  8    8 A:003      E:11111111
  9    9 A:004      E:11111111

M>BR 8

BREAKPOINT ENABLED
ADDR: 008  OCCUR: 1    GO: N
M>R

M>G

BREAKPOINTED AT A:008
A:0 B:00 C:0 TI:34 G:0 Q:00 L:00 I:4  P:008   C8    JP 008
M0:0000000000000000 M1:0000000000000000 M2:0000000000000000
                                                M3:0000000000000000

M>S

STEP: 1
A:0 B:00 C:0 TI:3F G:0 Q:00 L:00 I:4  P:008   C8    JP  008
M0:0000000000000000 M1:0000000000000000 M2:0000000000000000
                                                M3:0000000000000000
```

The program is working; user enters the MOLE PROG program to burn a PROM.

```
M>@PROG

PROG,REV:x,<date>

P>CH 2716

PROM 2716



P>US SH

BUFFER IN USE: SH . MAX ADDR = 07FF

P>P

INSERT PROM IN SOCKET AND APPLY Vpr. READY (Y/N,CR=YES)? <CR>
PROGRAMMING OVER. CHECKSUM = 0 1AC
** REMOVE Vpr AFTER PROGRAMMING
P>
```

Appendix D

INPUT LINE EDITING SUMMARY

<ctrl-A> Insert characters at current cursor position. The insertion mode is exited when a control key or carriage return is entered.

<ctrl-B> Backspace the cursor to the beginning of the previous word.

<ctrl-D> Truncate line at current cursor position. Characters after the cursor are ignored and the line gets entered into the system as if the return key had been hit.

<ctrl-E> Move the cursor to the end of the line. The characters in the line are not affected. If a <ctrl-E> is the very first character entered on a line, it will recall the previous command line. This line can then be edited and entered.

<ctrl-F> Forward space the cursor to the beginning of the next word.

<ctrl-H> Backspace one position. The character backspaced over is not removed from the input line buffer.

<ctrl-L> Forward space the cursor by one position. The character moved over is not affected.

<ctrl-Q> Abort the line.

<ctrl-S> Move the cursor to the start of the line. The characters in the line are not affected.

<ctrl-W> Forward space one character beyond the next character typed.

<ctrl-X> Delete the character at the current cursor position.

<ctrl-Z> Forward space to the next occurrence of the next character typed.

Appendix E

MOLE POWER-ON DIAGNOSTICS

The following tests are performed when the MOLE is powered-on:

1) A read/write test of the upper 4K bytes of Brain board RAM, addresses 0F000H through 0FFFFH.

   The only error message from this test is:

   RAM ERROR AT ADDRESS aaaa. DATA WRITTEN/READ= ww/rr

2) A test of the four Brain board system firmware PROMs. The test checks that the correct PROM is in the correct socket and that each PROM has the correct checksum.

   The possible error messages are the same as those for the DIAG ROM command, which are messages giving the PROM number(s) for the PROM(s) which failed the test.

National Semiconductor

MICROCOMPUTER SYSTEMS DIVISION

READER'S COMMENT FORM

In the interest of improving our documentation, National Semiconductor invites your comments on this manual.

Please restrict your comments to the documentation. Technical Support may be contacted at:

(800) 538-1866 — U.S. non CA
(800) 672-1811 — CA only
(408) 733-2600

Please rate this document according to the following categories. Include your comments below.

	EXCELLENT	GOOD	ADEQUATE	FAIR	POOR
Readability (style)	☐	☐	☐	☐	☐
Technical Accuracy	☐	☐	☐	☐	☐
Fulfills Needs	☐	☐	☐	☐	☐
Organization	☐	☐	☐	☐	☐
Presentation (format)	☐	☐	☐	☐	☐
Depth of Coverage	☐	☐	☐	☐	☐
Overall Quality	☐	☐	☐	☐	☐

NAME _____ DATE _____

TITLE _____

COMPANY NAME/DEPARTMENT _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

Do you require a written reply? ☐ Yes ☐ No

Comments:

_____

_____

_____

_____

_____

_____

FOLD, STAPLE, AND MAIL.

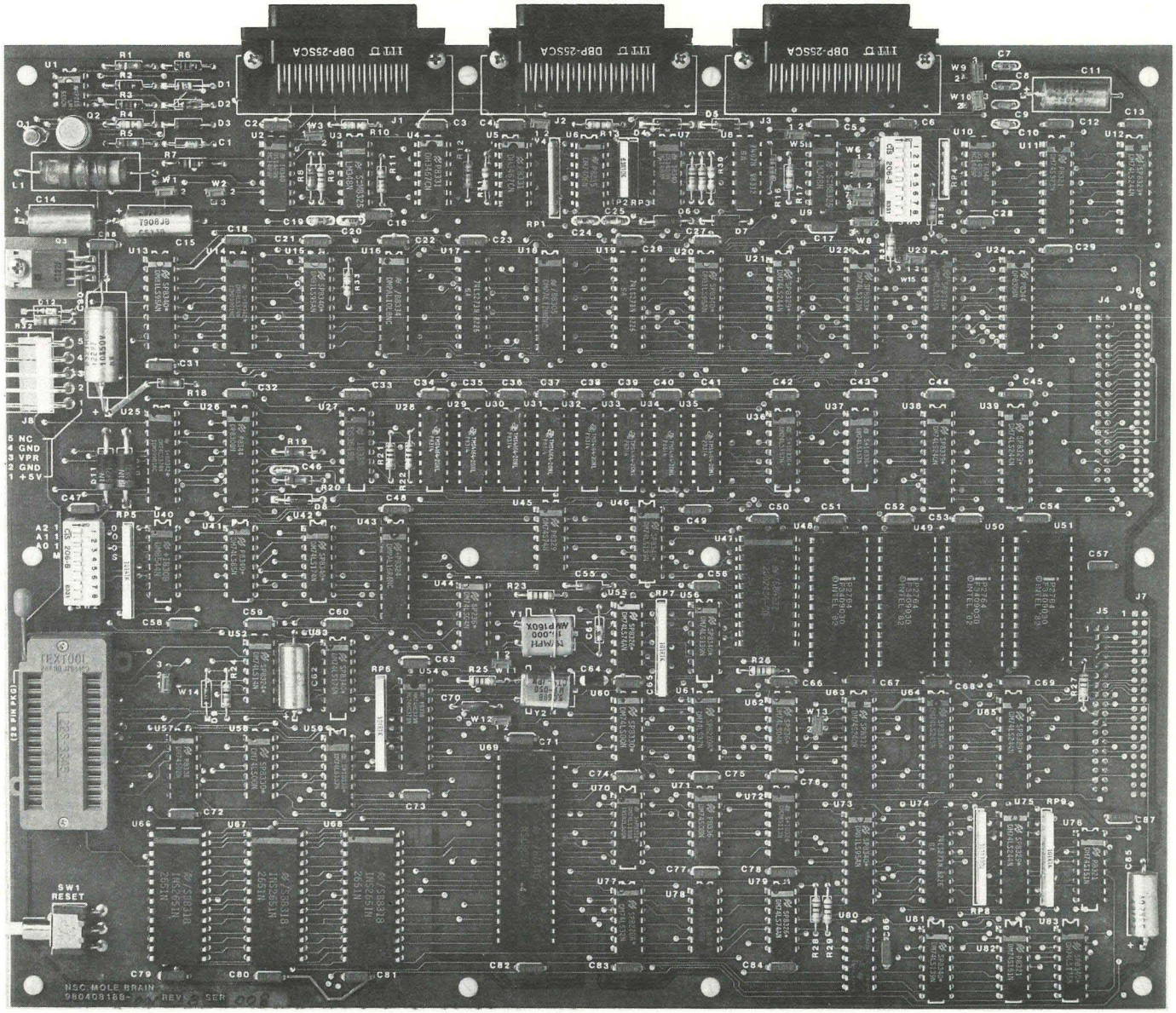
NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 409 SANTA CLARA, CA

POSTAGE WILL BE PAID BY ADDRESSEE

National Semiconductor Corporation
Microcomputer Systems Division
Technical Publications Dept. 8278, M/S 7C261
2900 Semiconductor Drive
Santa Clara, CA 95051



F0-00-0

Figure 2-1 MOLE Brain Board

National Semiconductor Corporation

Microcomputer Systems Division

National Semiconductor Corporation
2900 Semiconductor Drive
Santa Clara, California 95051
Tel: (408) 721-5000
TWX: (910) 339-9240

National Semiconductor
5955 Airport Road
Suite 206
Mississauga, Ontario
L4V1R9 Canada
Tel: (416) 678-2920
TWX: 610-492-8863

Electronica NSC de Mexico SA
Hegel No. 153-204
Mexico 5 D.F. Mexico
Tel: (905) 531-1689, 531-0569,
531-8204
Telex: 017-73550

NS Electronics Do Brasil
Avda Brigadeiro Faria Lima 830
8 Andar
01452 Sao Paulo, Brasil
Telex: 1121008 CABINE SAO PAULO
113193 INSBR BR

National Semiconductor GmbH
Furstenriederstraße Nr. 5
D-8000 München 21
West Germany
Tel.: (089) 5 60 12-0
Telex: 522772

National Semiconductor (UK), Ltd.
301 Harpur Centre
Horne Lane
Bedford MK40 1TR
United Kingdom
Tel: 0234-47147
Telex: 826 209

National Semiconductor Benelux
Ave. Charles Quint 545
B-1080 Bruxelles
Belgium
Tel: (02) 4661807
Telex: 61007

National Semiconductor (UK), Ltd.
1, Bianco Lunos Allè
DK-1868 Copenhagen V
Denmark
Tel: (01) 213211
Telex: 15179

National Semiconductor
Expansion 10000
28, Rue de la Redoute
F-92 260 Fontenay-aux-Roses
France
Tel: (01) 660-8140
Telex: 250956

National Semiconductor S.p.A.
Via Solferino 19
20121 Milano
Italy
Tel: (02) 345-2046/7/8/9
Telex: 332835

National Semiconductor AB
Box 2016
Stensätravägen 4/11 TR
S-12702 Skärholmen
Sweden
Tel: (08) 970190
Telex: 10731

National Semiconductor
Calle Nunez Morgado 9
(Esc. Dcha. 1-A)
E-Madrid 16
Spain
Tel: (01) 733-2954/733-2958
Telex: 46133

National Semiconductor Switzerland
Alte Winterthurerstrasse 53
Postfach 567
CH-8304 Wallisellen-Zürich
Tel: (01) 830-2727
Telex: 59000

National Semiconductor
Pasilanraitio 6C
SF-00240 Helsinki 24
Finland
Tel: (90) 14 03 44
Telex: 124854

NS Japan K.K.
POB 4152 Shinjuku Center Building
1-25-1 Nishishinjuku, Shinjuku-ku
Tokyo 160, Japan
Tel: (03) 349-0811
TWX: 232-2015 NSCJ-J

National Semiconductor Hong Kong, Ltd.
1st Floor,
Cheung Kong Electronic Bldg.
4 Hing Yip Street
Kwun Tong
Kowloon, Hong Kong
Tel: 3-899235
Telex: 43866 NSEHK HX
Cable: NATSEMI HX

NS Electronics Pty. Ltd.
Cnr. Stud Rd. & Mtn. Highway
Bayswater, Victoria 3153
Australia
Tel: 03-729-6333
Telex: AA32096

National Semiconductor PTE, Ltd.
10th Floor
Pub Building, Devonshire Wing
Somerset Road
Singapore 0923
Tel: 652 700047
Telex: NATSEMI RS 21402

National Semiconductor Far East, Ltd.
Taiwan Branch
P.O. Box 68-332 Taipei
3rd Floor, Apollo Bldg.
No. 218-7 Chung Hsiao E. Rd.
Sec. 4 Taipei Taiwan R.O.C.
Tel: 7310393-4, 7310465-6
Telex: 22837 NSTW
Cable: NSTW TAIPEI

National Semiconductor (HK) Ltd.
Korea Liaison Office
6th Floor, Kunwon Bldg.
No. 2, 1-GA Mookjung-Dong
Choong-Ku, Seoul, Korea
C.P.O. Box 7941 Seoul
Tel: 267-9473
Telex: K24942

Order No. 420408188-001

Publication No. 420408188-001A