MMDS0508OM/D

MMDS

MODULAR DEVELOPMENT SYSTEM for 68HC05 and 68HC08 MICROCONTROLLERS

OPERATIONS MANUAL

MOTOROLA

MMDS0508

Motorola Modular Development System
Operations Manual

MMDS0508OM/D

Motorola reserves the right to make changes without further notice to any products herein to improve reliability, function, or design. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part.

Motorola and the Motorola logo are registered trademarks of Motorola Inc.

Motorola Inc. is an Equal Opportunity/Affirmative Action Employer

MS-DOS is a registered trademark of Microsoft Corporation. IBM is a registered trademark of IBM Corporation.

MMEVS05 software is © P & E Microcomputer Systems, Inc.*, 1995; All Rights Reserved. Portions of the software are © Borland International, 1987. Portions of the software are © TurboPower Software, 1988.

---

* P & E Microcomputer Systems, Inc.
P.O. Box 2044
Woburn, MA 01888-2044
(617) 353-9206

Revision History

This table summarizes differences between this revision and the previous revision of this emulation module user's manual.

Previous Revision	None
Current Revision	Original release
Date	08/96

Table of Contents

General Description

Contents 11

Introduction 11

About this Manual 12

System Features 13

System Components 15

Host Computer Requirements 16

Acronyms. 17

Installation

Contents 19

Introduction 20

Configuring the Platform Board 22
  Factory Test Header (J1) 23
  Port Voltage Control Headers (J2–J4) 24

Installing the EM 25

Removing the EM 25

Making System Connections 26
  Host Computer Connection 26
  Bus State Analyzer Connection 26
  Target Cable Connection 27
  Power Connection 27

Reset Switch 28

Serial Connector and Cable Information .28
  Serial RS232 Connector 28
  Logic Cables and Connectors 30

Power Supply Fuse Replacement 32

Loading and Initialization

Contents 35
Software Distribution Format 36
Installing MMDS Software 36
Personality Files 37
Using MMDS Software 37
  Running MMDS05 38
  Running MMDS08 39
MMDS Communication 40

User Screens

Contents 41
Introduction 42
Main Window Screens. 42
  Status Area 45
  CPU Window 47
  Source/Code F2 Window 47
    Code F2 Window 47
    Source Window 48
  Variables F8 Window 49
  Memory F3 Window 50
  Debug F10 Window 50
Pop-Up Windows. 51
  Stack Window 52
  Set Memory Window 53
  Baud Window 54
  Emulator Clock Frequency Window 54
  Other Windows 55
Mouse Operation. 56
Changing Screen Colors 58

Operation Fundamentals

Contents 59
Introduction 60
System Initialization. 60
  Setting Communications Baud Rate 61
  Standard Memory Mapping 61

Custom Memory Mapping 62
Initializing the Clock Speed 63
Loading User Software 63
Initializing Memory 63
Initializing Assembly Language 64
Initializing Memory Data 65

System Commands 66
Script Commands 66
Information Commands 67
Log File Commands 68
Debug Screen Control 69
Exit the Environment 70

Debug Commands. 71
Setting CPU Registers 71
Memory Display 71
Reset Control of the Emulation System 72
Using Breakpoints 72
Tracing Instructions 73
Execution Instructions 73

Bus State Analyzer Commands. 74
Configuring the Analyzer 74
Capturing Analyzer Data 74
Viewing Analyzer Data 74

Bus State Analysis

Contents 77

Introduction 78

Operating the Bus Analyzer. 79
Defining Events (Terms) 79
Selecting the Trigger Mode 84
Selecting Options 86
Collecting Bus Data 87
Viewing Data 88
Searching the Trace Buffer 95
Using the Time-Tag Clock 96

Command-Line Commands

Contents 99
Introduction 102
Command Syntax 102
Command Explanations 103

S-Record Information

Contents 199
Introduction 199
S-Record Content 200
S-Record Types. 201
S-Record Creation. 202
S-Record Example 202

Index

Index 205

General Description

Contents

Introduction 11
About this Manual 12
System Features 13
System Components 15
Host Computer Requirements 16
Acronyms. 17

Introduction

The M68MMDS05/08 Motorola Modular Development System (MMDS) is a tool for developing embedded systems based on an MC68HC05 or MC68HC08 microcontroller unit (MCU). The MMDS is a modular emulation system that, when connected to a user's target system, gives the user interactive control of a microcontroller application.

The RAPID software provides an integrated development environment and includes an editor, assembler, and a user interface to the MMDS system. The environment allows for source-level debugging and simplifies writing and debugging code for an embedded MCU system. These features significantly reduce development time.

A complete MMDS system contains the station module (M68MMDS0508), an emulation module (EM), and a target cable assembly. An EM completes MMDS functionality for a particular MCU or MCU family. To accommodate emulation of the numerous MCUs available, the MMDS uses a variety of different EMs. Refer to Motorola's

Development Tool Selector Guide, order number SG173/D, for a complete list of available emulator modules and the appropriate user's manual for EM installation instructions. For connection to a target system, a separately purchased target cable with the appropriate target head also is needed.

To use the MMDS, an IBM (or compatible) host computer is needed. A 9-pin RS-232 serial cable also is provided with the MMDS.

About this Manual

This manual covers MMDS software, hardware, and reference information as follows:

- **Installation** on page 19 explains M68MMDS0508 hardware.
- **Loading and Initialization** on page 35 explains how to load and initialize the MMDS system software.
- **User Screens** on page 41 explains the window screens, as well as how to use a mouse.
- **Operation Fundamentals** on page 59 describes command usages.
- **Bus State Analysis** on page 77 details the built-in state analyzer features.
- **Command-Line Commands** on page 99 explains MMDS command syntax.
- **S-Record Information** on page 199 gives reference information about Motorola S-records.

System Features

The MMDS is a full-featured development system that provides in-circuit emulation. Its features include:

- Real-time, non-intrusive, in-circuit emulation
- Real-time bus state analysis
- MC68HC11K1 system controller for fast command transfer
- Meets ECC92 European electromagnetic compatibility standards
- Four complex data breakpoints; a data breakpoint can be qualified by an address, an address range, data, or externally connected logic clips.
- 32 variables or real-time variables, plus a 32-byte block of real-time memory, mappable anywhere within a 1 Kbyte window over the 64-K HC05/HC08 memory map.
- 64 Kbytes of emulation memory to accommodate the largest available ROM size of current HC05/HC08 MCUs.
- 64 hardware instruction breakpoints over the 64-K memory map
- A DOS personality file for each EM. Each personality file provides a memory-map definition.
- 64K bytes of emulation memory, to accommodate the largest available ROM size of current HC05/HC8 MCUs.
- Latch-up resistant design (47-Ω series resistor on I/O connections to the target system) to make power-up sequencing unimportant.
- Built-in bus state analyzer:
  - 8K x 64 real-time trace buffer
  - Four hardware triggers for controlling real-time bus analysis and to provide breakpoints
  - Nine triggering modes
  - Display of real-time trace data as raw data, disassembled instructions, raw data and disassembled instructions, or assembly-language source code
  - As many as 8190 pre- or post-trigger points

  - Trace buffer can be filled while single-stepping through user software
  - 16-bit time tag, or an optional 24-bit time tag that sacrifices eight logic clips
  - Eight software selections for the time tag clock source, permitting wide time variance between analyzer events
  - 16 general-purpose logic clips, five of which can be used to trigger the bus state analyzer sequencer
- Four software-selectable internally generated oscillator clock sources
- Built-in power supply with 85 to 264 VAC input
- Command and response logging to disk files
- SCRIPT command for automatic execution of a sequence of MMDS commands
- Assembly-language source-level debugging
- RS-232 operation speeds as high as 57600 baud
- On-screen, context-sensitive help via pop-up menus and windows
- CHIPINFO command for memory-map, vectors, register, and pin-out information pertaining to the device being emulated
- Emulation that allows multiple types of reset:
  - RESET command resets target
  - RESETGO command resets target and begins execution
  - WAIT4RESET command resets target via target hardware assertion of the RESET signal
- Mouse or keyboard control of host software
- Status line that displays such information as emulator state, communications port, and communications rate
- Compact size: 15.38 inches (390.6 mm) deep, 10.19 inches (258.83 mm) wide, and 2.75 inches (69.85 mm) high. The station module weighs 6.0 pounds (2.72 kg).

Connections, configuration, and other related information is explained in the installation section of this document. For similar information with regard to EMs, see the corresponding EM user's manual.

System Components

The following items are included with the MMDS:

- **Station module:** the MMDS enclosure, containing the platform board and the internal power supply. The sliding panel in the enclosure top lets you insert an EM easily.
- **9-pin RS-232 serial cable:** The cable that connects the station module to the host computer RS-232 port.
- **Serial adapter:** DB9M to DB25F RS-232 adapter for connecting to a 25-pin serial port on a host system.
- **System software:** RAPID integrated development environment featuring editor, assembler, and assembly source level debugger (3.5-inch diskettes)
- **RAPID documentation:** A *RAPID Integrated Development Environment User's Manual*
- **MMDS documentation**: An *MMDS0508 Operations Manual* (MMDS0508OM/D – this manual).
- **Software Release Guide:** Documentation on the current release of system software.
- **Two logic clip cable assemblies:** twisted-pair cables that connect the station module to your target system, a test fixture, a clock, an oscillator, or any other circuitry useful for evaluation or analysis. One end of each cable assembly has a molded connector, which fits into pod A or pod B of the station module. Leads at the other end of each cable terminate in female probe tips. Ball clips come with the cables.

Separately purchased Motorola personality products include:

- **An emulation module (EM):** One of many printed circuit boards that complete MMDS functionality for one or more particular MCUs. The two DIN connectors on the bottom of the EM fit into connectors on the top of the MMDS0508 platform board for power and signal connections. The EM also has a connector for the target cable. EMs are purchased separately from the platform board and are shipped with a user's manual containing information specific to the module.
- **Optional target cable:** A separately purchased target cable that is part of a cable assembly, used to connect the target system to the MMDS system
- **Optional target head adapter:** A separately purchased target head adapter that is part of a cable assembly, used to connect the target system to the MMDS system

User supplied components include:

- **Host computer:** For further information refer to **Host Computer Requirements**

Host Computer Requirements

The host computer for the MMDS must be hardware and software compatible with IBM AT or PS/2 computers. The host computer must run DOS 5.0 or later versions. The host software requires approximately 512 Kbytes.

An asynchronous communications port, configured as COM1, COM2, COM3, or COM4, is required for communications between the MMDS and the host computer.

For improved product performance, additional system enhancements can be added. These are: 80386- or 80486-based systems, a hard disk drive, and a high-resolution color monitor with either an EGA or VGA graphics adapter card. The MMDS system software also supports a Microsoft, Logitech, or IBM mouse. Other mice may be compatible, but Motorola does not guarantee their satisfactory performance with MMDS software.

Acronyms

Table 1 provides definitions for the acronyms used throughout this manual.

Table 1. Acronym Definitions

Term	Description
M68MMDS0508	The station module where common hardware for all M68HC05 and M68HC08 emulation resides.
EM	An emulation module that connects to the platform board to customize the MMDS for a particular MCU or family of MCUs.
RAPID	An integrated development environment that includes an editor and allows applications such as assemblers and debuggers to be blended into a single environment.
MMDS	Motorola Modular Evaluation System. A generic term that describes a two-board evaluation system consisting of the platform board, one of many emulation modules, and system software (RAPID integrated development environment, CASM assembler and MMDS debugger).
CASM	Cross assembler that assembles M68HC05 (CASM05) or M68HC08 (CASM08) code.

Installation

Contents

Introduction 20

Configuring the Platform Board 22
  Factory Test Header (J1) 23
  Port Voltage Control Headers (J2–J4) 24

Installing the EM 25

Removing the EM 25

Making System Connections. 26
  Host Computer Connection 26
  Bus State Analyzer Connection 26
  Target Cable Connection 27
  Power Connection 27

Reset Switch 28

Serial Connector and Cable Information 28
  Serial RS232 Connector 28
  Logic Cables and Connectors 30

Power Supply Fuse Replacement. 32

Introduction

Complete MMDS installation consists of configuring the platform board, configuring and installing the appropriate emulation module (EM), and making system cable connections. Consult the system components section on page 9 for a list of all the system parts, including a separately purchased EM. Note that EM configuration is specific to a particular EM; follow the guidance of the specific EM user's manual. Follow the guidance given in this section to complete the MMDS installation.

Figure 1 shows the right side of the MMDS station module, with the access panel open. The recessed reset switch and power LED are on the front of the station module. The logic cable A and B connectors (pod A and pod B) are on the right (as you face the station module). When using the logic clip cables, always attach the black clip to ground.

Figure 2 shows the left side of the station module, with the access panel closed. The power cord socket, the power switch, and the 9-pin RS-232 serial connector are on the left as you face the station module. The circular +5 V out connector, also on the left side of the station module, is reserved for future features. A spacer covers the final enclosure cutout, for a future conenctor.)

NOTE: *This manual uses the words left and right as you face the front of the station module.*

Follow the guidance of this chapter to complete the installation of your MMDS.

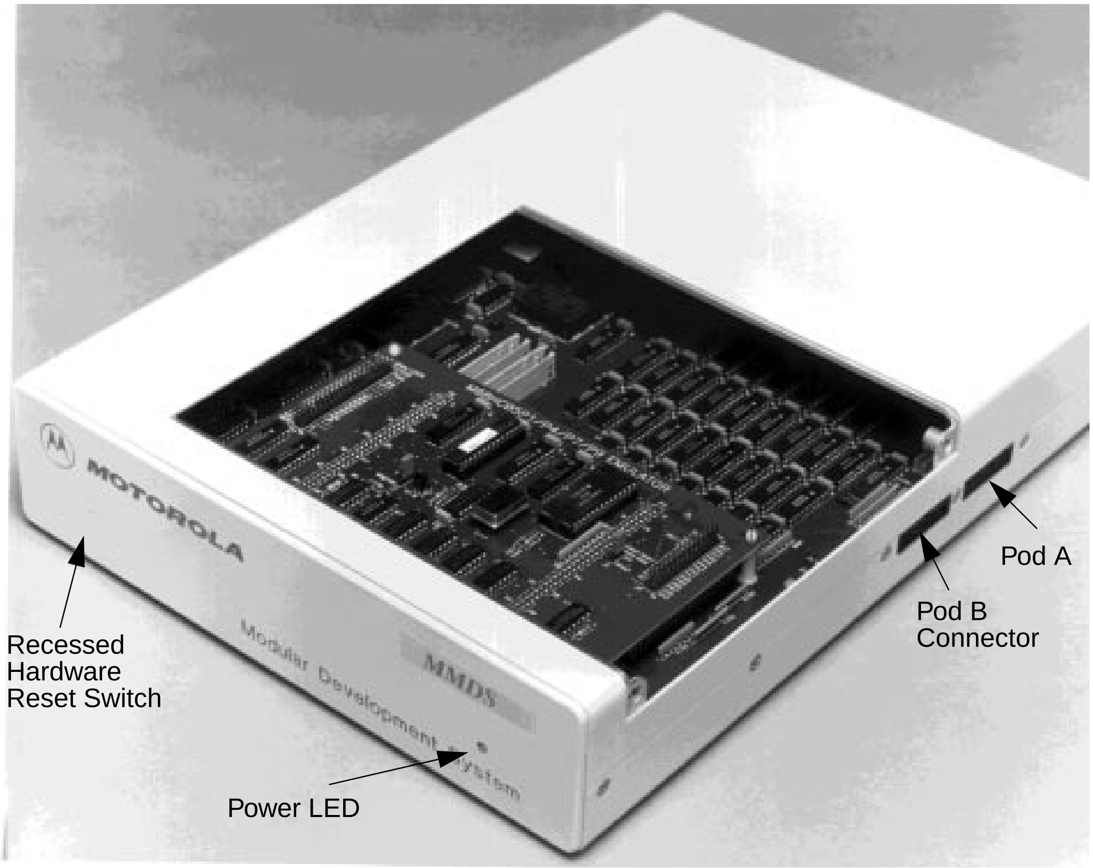

Pod A
Pod B
Connector
Recessed
Hardware
Reset Switch
Power LED
MOTOROLA
Modular Development System
MMDS

Figure 1. M68MMDS0508 Station Module (Right Side)

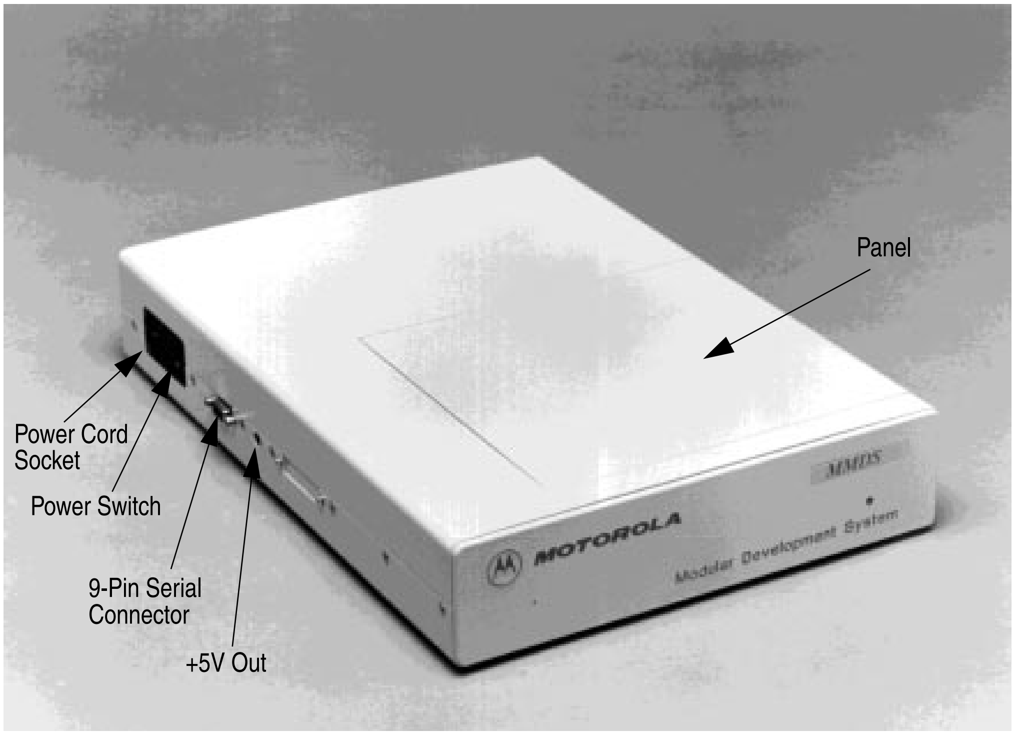

Panel
Power Cord Socket
Power Switch
9-Pin Serial Connector
+5V Out
MMDS
MOTOROLA
Modular Development System

Figure 2. M68MMDS0508 Station Module (Left Side)

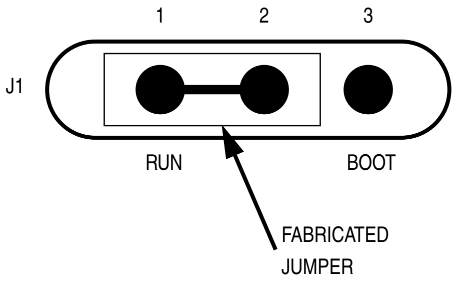
Configuring the Platform Board

The MMDS platform board has four jumper headers, all located near the front of the platform board. Jumper header J1 is for factory test. Depending on the design of each emulation module, jumper headers J2, J3, and J4 may control the voltage levels for ports A through D.

NOTE: *The factory configures the platform board correctly for virtually all users before shipping the MMDS. These platform board jumpers should not be reconfigured unless instructed to do so by an emulation module (EM) user's manual.*

Factory Test Header (J1)

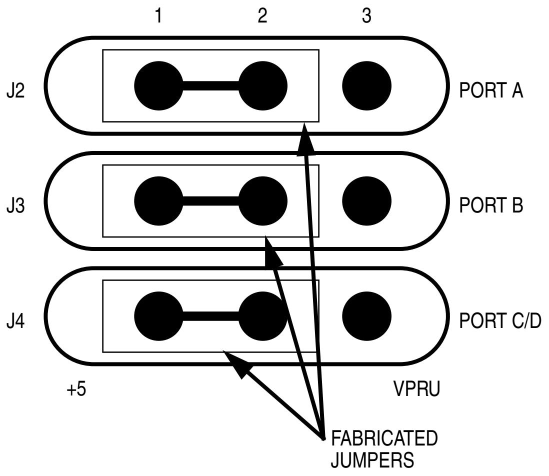
This diagram shows the factory configuration of jumper header J1. The fabricated jumper between pins 1 and 2 is correct for MMDS operation.



The alternate jumper position has significance only for factory personnel.

Port Voltage Control Headers (J2–J4)

Jumper headers J2 through J4, located near the right front corner of the platform board, set the voltage levels for ports A through D. The following diagram shows the factory configuration. The fabricated jumpers between pins 1 and 2 of these headers set the +5-volt level for all ports. This is the correct configuration for MMDS operation, unless the EM user's manual says that the EM is a low-voltage board.



If the EM can operate at low voltage, any of the ports can be operated at the low-voltage level. To do so, reposition the fabricated jumper of the corresponding header to pins 2 and 3. Jumper header J2 controls the voltage level of port A, jumper header J3 controls the voltage level of port B, and jumper header J4 controls the voltage level of port C or D, whichever pertains to the EM.

Installing the EM

Perform these steps to install an EM in an MMDS enclosure:

- Ensure that station module power is off.
- Remove the panel from the station module top by first turning each of the two 1/4-turn speed fasteners to the released position.
- Install the EM on the platform board: Carefully fit the female 96-pin connectors (located on the bottom of the EM) onto the corresponding male DIN connectors on the top of the platform board. Snap the EM onto the plastic standoffs and make sure that the DIN connectors are joined together firmly.
- Connect the target cable, if appropriate.
- Reaffix the panel to the station module top.

NOTE: *Many EM boards may have 64-pin female DIN connectors. If so, these will mate with the male DIN connectors on the platform board. The keyed plastic on the connector will ensure proper alignment.*

Removing the EM

Complete these steps to remove an EM from an MMDS enclosure:

- Ensure station module power is off.
- Remove the panel from the station module top by first turning each of the two 1/4-turn speed fasteners to the released position
- Disconnect the target cable from the EM target connector.
- Unsnap all nylon spacers from the edges of the EM. Then carefully lift the EM straight up, separating it from the platform board.
- Reaffix the panel to the station module top.

Making System Connections

The specific application determines the number of MMDS connections required. At the very least, the platform board and EM must be connected to the host computer and to a power supply. Cable connections are explained in the sections that follow.

Host Computer Connection

Use the 9-pin serial cable to connect a host computer’s 9-pin serial port to the MMDS 9-pin serial cable connector. Note which computer serial port is used. If the COM1 port (the default) is not used, the port number must be specified in the MMDS software start-up command.

If the serial port is a 25-pin connector, use the provided 9- to 25-pin adapter between the host serial port and the cable.

If the development system is operated in the RAPID environment, RAPID must be configured to communicate through the proper serial port.

Bus State Analyzer Connection

If your work session includes bus state analysis, you may need the logic clip cable assemblies. The two logic clip connectors, referred to as pods A and B, allow external events to be captured in the analyzer. External clock signals for the emulator and for the analyzer also can be input through logic clip connections.

The pod A and pod B connectors are located on the right side of the station module. Pod A is nearest the station module power supply. These pod connections correspond to the cable A and cable B selections available in the bus state analyzer configuration window. **Table 2** on page 31 gives pinout information for both logic clip connections.

If you need only one logic cable assembly, connect it to either pod A or pod B. Orient the cable connector so that its pin 1 connects to pin 1 of the pod. The keyed plastic ensures proper orientation. Connect the other end of the logic cable assembly to an external target point. Optionally, connect the probe tips to the ball clips that come with the cable assembly, then connect the ball clips to appropriate external test points.

NOTE: *The white probe of pod A is the external clock input for the emulator and the white probe of pod B is the external clock input for the analyzer. Always connect the black (ground) probe tip to an appropriate ground point of the target system before making logic clip connections to target points.*

If you need the second logic cable assembly, connect it in the same way to the remaining pod connector of the station module. Make target-system connections as for the first cable.

Target Cable Connection

If the MMDS will interface with a user's target system, the target system should be connected to the EM board through a target cable assembly. A cable assembly consists of a target head and a target cable. The target connector will be on the right side of the EM module. Connect one end of the target cable to the EM target connector and the other end of the target cable to the user's target system before power-up. See the specific EM user's manual for specific information on the target head adapter and the appropriate target cable.

Make sure that the target head adapter and target cable mate correctly. Consult the EM manual for proper connection. Connecting the target cable any other way can damage the system.

NOTE: *When connecting a target cable, press only on the rigid plastic connectors at either end of the cable. Pressing on the flexible part of the cable can damage the cable.*

Power Connection

The final MMDS connection is line power. The MMDS power switch is the rocker switch on the left side of the station module. Set the power switch to OFF.

Insert the female end of the power cord into the power cord socket. Then plug the other end of the cord into a line-power outlet and set the power switch to ON. The green LED on the front of the station module lights to confirm system power.

This completes cable connections. Refer to **Loading and Initialization** on page 35 for information on communicating with the MMDS.

Make sure the power to the MMDS is turned off when installing the EM or removing the EM from the station module. Sudden power surges can damage MMDS circuits.

Reset Switch

RS-232 handshake signals control MMDS resets. A reset initializes the control board from its start-up point. If the host serial port does not implement handshaking, reset the MMDS manually. Press gently to trip the switch.

Serial Connector and Cable Information

This section contains pin identification and signal names for connectors common to all MMDS systems. For pinout information for a particular EM connector, refer to the corresponding EM user's manual.

Serial RS232 Connector

This diagram shows pin numbering for the 9-pin female serial connector of the platform board. **Table 1** lists the signals transmitted on the 9-lead serial connection.

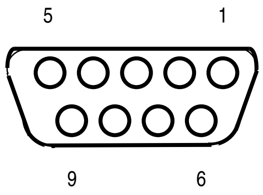


Table 1. Serial Connector and Cable Pin Assignments

Connector Pin	Mnemonic	Signal
1	DCD	DATA CARRIER DETECT — Output signal that indicates detection of an acceptable carrier signal
2	RX	RECEIVE DATA — Serial data output line
3	TX	TRANSMIT DATA — Serial data input line
4	DTR	DATA TERMINAL READY — Input signal that indicates on-line/in-line/active status
5	GND	GROUND
6	DSR	DATA SET READY — Output signal that indicates on-line/in-line service/active status
7	RTS	REQUEST TO SEND — Input signal that requests permission to transfer data
8	CTS	CLEAR TO SEND — Output signal that indicates a ready-to-transfer data status

Logic Cables and Connectors

The diagram below shows the pin numbering for both pod A and pod B logic cable connectors of the station module. **Table 2** details the pinout designations where an LC_X assignment gives each logic clip connection a name. The logic clips are used to capture data in the bus state analyzer. (Pin 9 of both pods provides connection to an external ground.) In addition, the pod connectors are used as external clock inputs for the emulator clock and bus state analyzer timetag. The table also provides color code information for each pod.

The external clock inputs are through pin 17 of each pod. Pod A pin 17 is the external clock input for the emulator. To use this source, make the desired clock connection to the white probe tip and use the OSC command to select an external source.

Pod B pin 17 is the external timetag input for the bus state analyzer. To use this source, make the desired clock connection to the white probe tip and use the TIMETAG command to select an external time tag source for the analyzer.

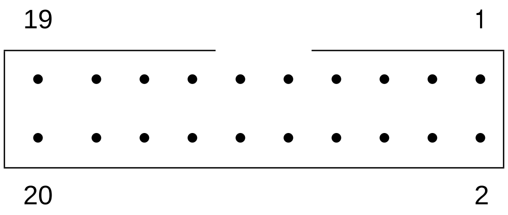


Table 2. Pod and Logic Cable Pin Assignments

Pod Pin	Pod A Signal	Pod B Signal	Probe Color
1	LC0	LC8	Brown (BRN)
2	GND	GND	
3	LC1	LC9	Red (RED)
4	GND	GND	
5	LC2	LC10	Orange (ORG)
6	GND	GND	
7	LC3	LC11	Yellow (YEL)
8	GND	GND	
9	LC4	LC12	Green (GRN)
10	GND	GND	
11	LC5	LC13	Blue (BLU)
12	GND	GND	
13	LC6	LC14	Purple (PUR)
14	GND	GND	
15	LC7	LC15	Gray (GRY)
16	GND	GND	
17	EXT_OSC	TT_OSC	White
18	GND	GND	
19	GND	GND	Black
20	GND	GND	

Power Supply Fuse Replacement

The station module power switch/connector assembly contains a standard 1/4 x 1 1/4 inch, 1.6-ampere, 250-volt ceramic, time-delay fuse. **Figure 3** shows this assembly with its door open (for fuse replacement).

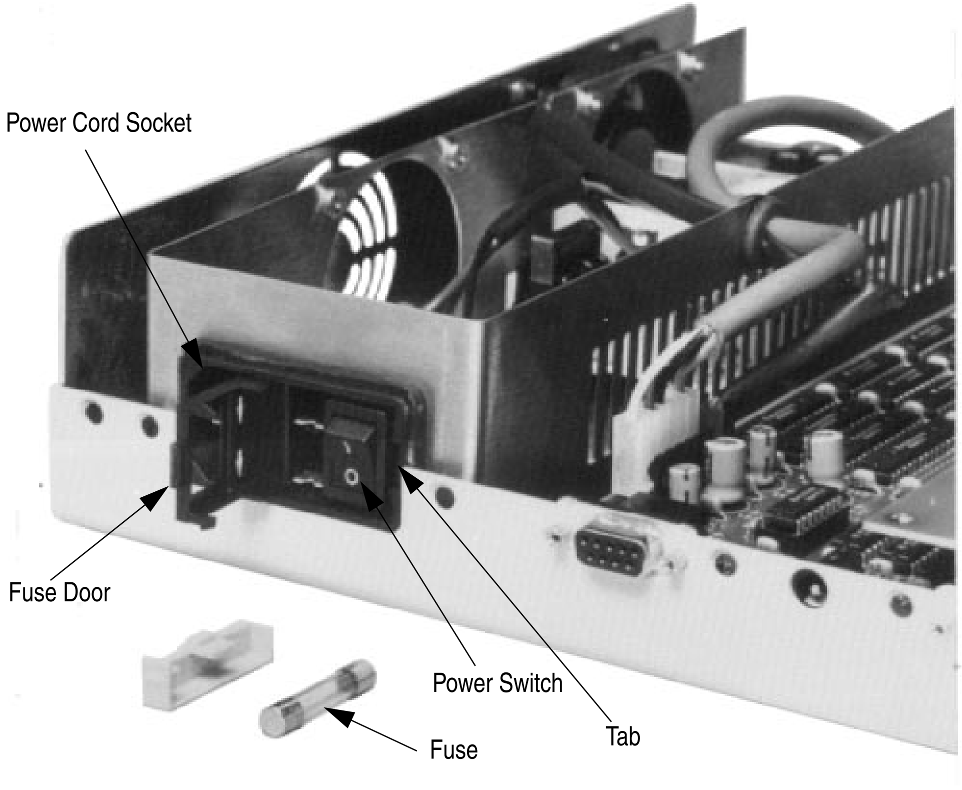

Power Cord Socket
Fuse Door
Power Switch
Tab
Fuse

Figure 3. Power Switch/Connector Assembly

To replace the fuse, perform these steps:

- Press the power switch OFF and disconnect the power cord.
- Insert a small screwdriver at the tab on the right edge of the switch/connector assembly. (**Figure 3** shows where to insert the screwdriver.) Gently pry open the assembly door, which swings open to the left.
- Remove the fuse holder from the switch/connector assembly. Remove the fuse from the holder.
- Insert the replacement fuse into the holder. Then re-install the holder in the switch/connector assembly. Make sure that the fuse holder arrow points down. Close the assembly door.
- Reconnect the power cord. This completes fuse replacement.

Loading and Initialization

Contents

Software Distribution Format 36

Installing MMDS Software 36

Personality Files 37

Using MMDS Software 37
  Running MMDS05 38
  Running MMDS08 39

MMDS Communication 40

Software Distribution Format

MMDS software is distributed on 3.5-inch high-density diskettes. The install process places the RAPID environment files and the MMDS software files in the directory designated during the install process. Table 3 describes the system files required to control the MMDS system, where *x* denotes a version number. Refer to the Software Release Guide for information on other files loaded.

Table 3. MMDS Software Files

File Name	Description
MMDS05.EXE	MMDS05 host software
MMDS05X.EXE	MMDS05 host software for running in a DOS window under Windows
MMDS05Vx.HLP	HELP command windows for the MMDS05 commands
MMDS08.EXE	MMDS08 host software
MMDS08X.EXE	MMDS08 host software for running in a DOS window under Windows
MMDS08Vx.HLP	HELP command windows for the MMDS08 commands

Installing MMDS Software

The install process will place MMDS and all supporting files on a hard drive. To install the MMDS files, insert the distribution diskettes into an active drive. Make the installation drive active by typing the drive letter followed by a colon (`:`) and press `<CR>`. Type `INSTALL` and press `<CR>`. Follow the directions as prompted by the install software.

Refer to the Software Release Guide for further information on the installation process.

Personality Files

Various designs of MCUs require different configurations of the MMDS system. The appropriate setup for each MCU is specified in a unique personality file.

NOTE: *These files are shipped with separately purchased EMs.*

Personality files are usually installed in the directory from which the MMDS software is executed. These personality files have a standard extension of.MEM. If a personality file is not located in the working directory, the software displays a search window used to find the correct file. To determine the personality files used by a particular EM module, refer to the appropriate EM user's manual.

More than one personality file can be installed; the MMDS operating software automatically loads the default personality file that corresponds to the currently connected EM module. As discussed in the following paragraph, other personality files can be loaded via the LOADMEM command or through use of the –M option.

Using MMDS Software

The correct executable file to run is dependent on which type EM is installed on the MMDS platform board. If an HC05 EM is installed, the MMDS05.EXE file should be run. If an HC08 EM is installed, the MMDS08.EXE file should be run.

The following paragraphs discuss the proper syntax for running the HC05 and HC08 software.

Alternatively, the MMDS can be called from within the RAPID environment. Running the MMDS05 or MMDS08 from RAPID may require running RINSTALL, RAPID's configuration set-up program, to set up the directory path, serial port, etc., as described in the RAPID user's manual.

NOTE: *If you plan to use the MMDS software in a DOS window under Windows, use the MMDS05X.EXE or MMDS08X.EXE files.*

Running MMDS05

To call the debugger executable directly from the DOS prompt, type this command:

```
C:\MMDS05>MMDS05
```

Note these five options for the startup command:

1. If the MMDS05 is connected to COM2, COM3, or COM4, add the corresponding integer to the command:
   ```
   C:\MMDS05>MMDS05 2
   ```
2. If the computer has a monochrome monitor, add BW to the command:
   ```
   C:\MMDS05>MMDS05 BW
   ```
3. To specify a personality file to be loaded automatically (instead of the default), add the –M option, followed by the filename (Do not put a space between the M and the filename). If the specified personality file has a .MEM extension, the .MEM extension may be omitted from the filename:
   ```
   C:\MMDS05>MMDS05 –M<filename>
   ```
4. To specify an S-record file (and any map file with the same name) to be loaded automatically, add the filename option. If the specified S-record file has a .S19 extension, the .S19 extension may be omitted from the filename:
   ```
   C:\MMDS05>MMDS05 <filename>
   ```
5. To specify a default baud rate of 9600, add the –B option:
   ```
   C:\MMDS05>MMDS05 –B
   ```

NOTE: *Multiple options in the start-up command should be separated by a space.*

Running MMDS08

To call the debugger executable directly from the DOS prompt, type this command:

```
C:\MMDS08>MMDS08
```

Note these five options for the startup command:

1. If the MMDS08 is connected to COM2, COM3, or COM4, add the corresponding integer to the command:
   ```
   C:\MMDS08>MMDS08 2
   ```
2. If the computer has a monochrome monitor, add BW to the command:
   ```
   C:\MMDS08>MMDS08 BW
   ```
3. To specify a personality file to be loaded automatically (instead of the default), add the –M option, followed by the filename. (Do not put a space between the M and the filename.) If the specified personality file has a .MEM extension, the .MEM extension may be omitted from the filename:
   ```
   C:\MMDS08>MMDS08 –M<filename>
   ```
4. To specify an S-record file (and any map file with the same name) to be loaded automatically, add the filename option. If the specified S-record file has a .S19 extension, the .S19 extension may be omitted from the filename:
   ```
   C:\MMDS08>MMDS08 <filename>
   ```
5. To specify a default baud rate of 9600, add the –B option:
   ```
   C:\MMDS08>MMDS08 –B
   ```

NOTE: *Multiple options in the start-up command should be separated by a space.*

MMDS Communication

The host program establishes communications with the MMDS system and displays the appropriate debug screen as shown in the section on user screens. If the host program fails to establish communications, an error screen appears and the system operation is returned to DOS. The information in the error screen helps determine why the software does not run.

For best performance of the system, communications between the host and the station module should be at the maximum available baud rate. At power-up, the host software automatically sets the maximum baud for the system.

Reduce the baud rate if a communication error message appears. Refer to **Running MMDS05** and **Running MMDS08** on the preceding pages to set the start-up baud rate at 9600. If communication errors persist, turn off the disk cache (SMARTDRV.EXE) on the computer.

Enter commands in response to the MMDS command prompt (**>**). When the emulation and debugging session has been completed, terminate the session by entering the EXIT or QUIT command.

User Screens

Contents

Introduction 42

Main Window Screens. 42
  Status Area 45
  CPU Window 47
  Source/Code F2 Window 47
    Code F2 Window 47
    Source Window 48
  Variables F8 Window 49
  Memory F3 Window 50
  Debug F10 Window 50

Pop-Up Windows. 51
  Stack Window 52
  Set Memory Window 53
  Baud Window 54
  Emulator Clock Frequency Window 54
  Other Windows 55

Mouse Operation. 56

Changing Screen Colors 58

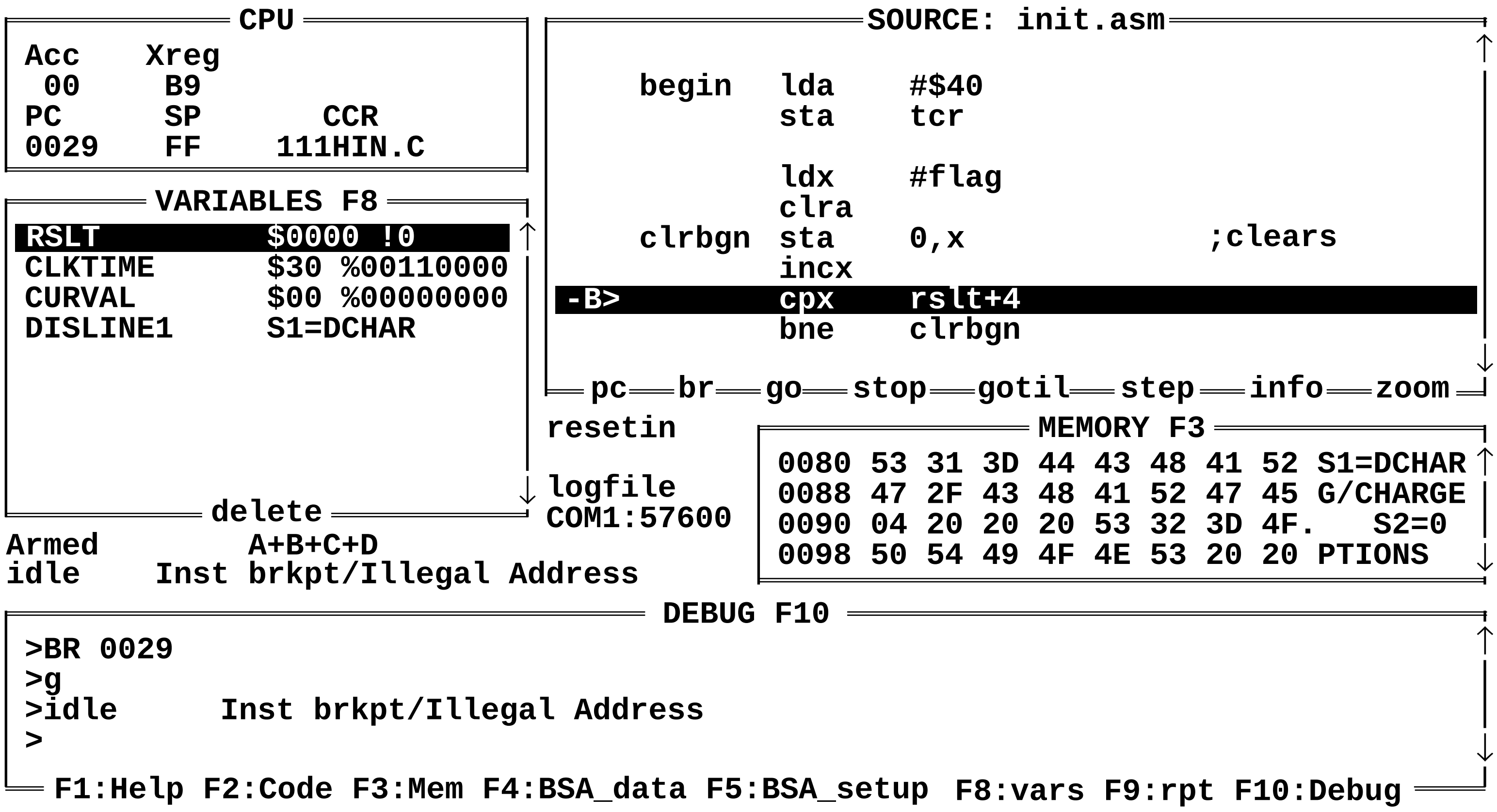
Introduction

The user interface screen to the MMDS development system consists of a status area, five static main windows and several pop-up windows. The screen displays the code, data and status information required for the user to control the emulation environment. This section provides a description of the screen functionality, including mouse operation. The screens associated with the bus state analyzer will be covered in the **Bus State Analysis** section.

Main Window Screens

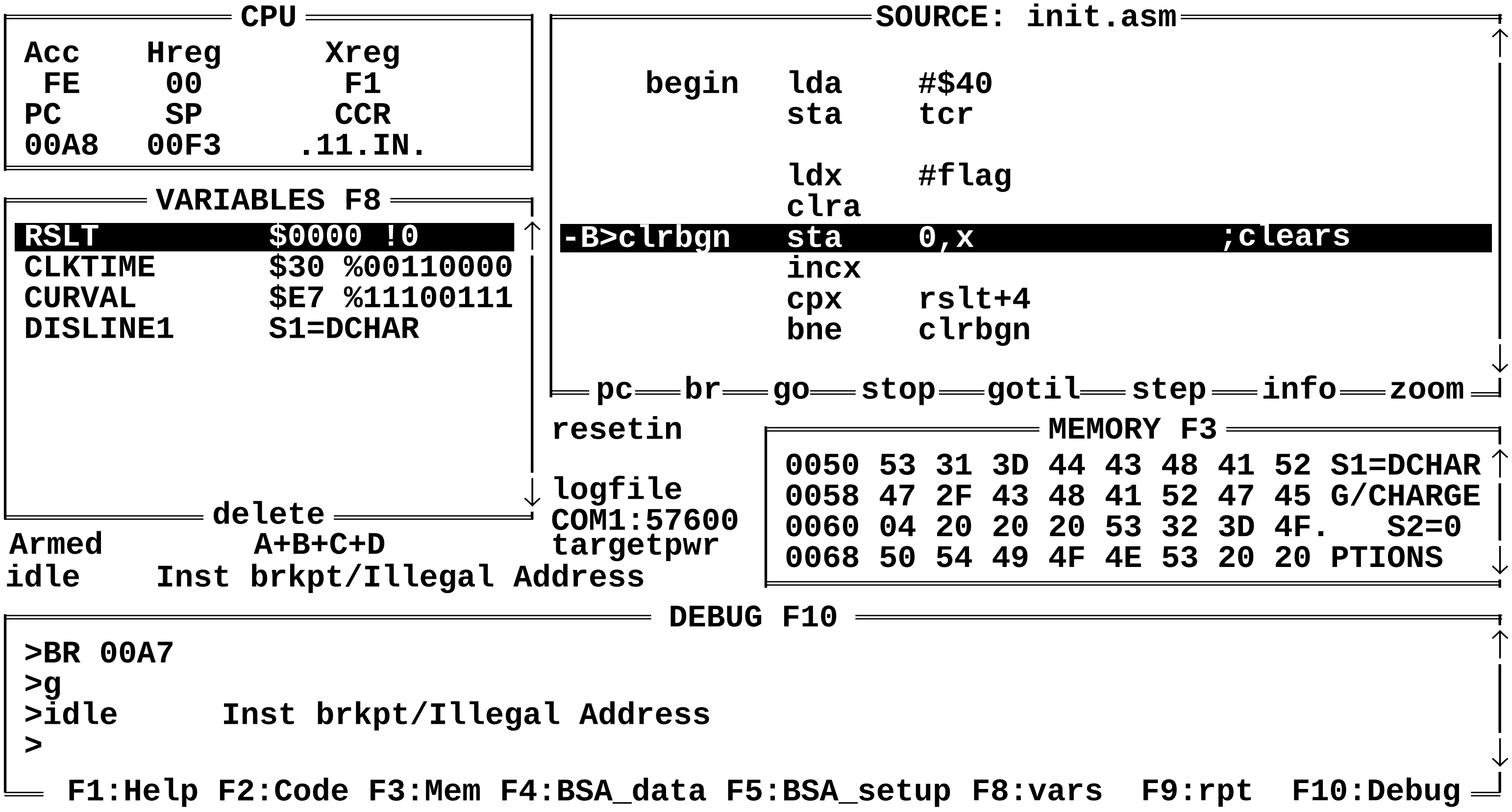
Figure 4 and **Figure 5** show the debug screen for the MMDS05 and the MMDS08 versions of the software, respectively. The screen is identical for both versions of the software except for the CPU window. The screen consists of a status area and five static windows which display source or object code, variables, the command line and the contents of the CPU registers or memory.

To carry out actions associated with a window of the debug screen, select the window. To select a window, press the numbered function key included in the window title. For instance, press the F2 key to select the source/code F2 window, press the F8 key to select the variables F8 window, and so forth. To return to the debug command line, press F10. Note that several operations can also be mouse controlled. Refer to **Mouse Operation** on page 56 for detailed information on mouse usage. **Table 4** lists the key commands available in any of the main windows.



```
CPU
Acc    Xreg
 00     B9
PC      SP       CCR
0029    FF    111HIN.C

VARIABLES F8
RSLT          $0000 !0
CLKTIME       $30 %00110000
CURVAL        $00 %00000000
DISLINE1      S1=DCHAR

delete
Armed         A+B+C+D
idle    Inst brkpt/Illegal Address

SOURCE: init.asm
     begin   lda    #$40
             sta    tcr

             ldx    #flag
             clra
     clrbgn  sta    0,x              ;clears
             incx
-B>          cpx    rslt+4
             bne    clrbgn

pc  br  go  stop  gotil  step  info  zoom
resetin

logfile
COM1:57600

MEMORY F3
0080 53 31 3D 44 43 48 41 52 S1=DCHAR
0088 47 2F 43 48 41 52 47 45 G/CHARGE
0090 04 20 20 20 53 32 3D 4F.   S2=0
0098 50 54 49 4F 4E 53 20 20 PTIONS

DEBUG F10
>BR 0029
>g
>idle     Inst brkpt/Illegal Address
>
F1:Help F2:Code F3:Mem F4:BSA_data F5:BSA_setup F8:vars F9:rpt F10:Debug
```

Figure 4. MMDS05 Debug Screen



```
═══════════ CPU ═══════════   ═══════════════════SOURCE: init.asm═════════════════
 Acc    Hreg      Xreg
  FE     00        F1             begin   lda    #$40
 PC      SP       CCR                     sta    tcr
 00A8   00F3    .11.IN.
                                          ldx    #flag
═══════ VARIABLES F8 ══════               clra
 RSLT            $0000 !0       -B>clrbgn  sta    0,x               ;clears
 CLKTIME         $30 %00110000                incx
 CURVAL          $E7 %11100111                cpx    rslt+4
 DISLINE1        S1=DCHAR                     bne    clrbgn
                                ══pc══br══go══stop══gotil══step══info══zoom══
                                resetin        ═════════ MEMORY F3 ══════════
                                               0050 53 31 3D 44 43 48 41 52 S1=DCHAR
                                logfile        0058 47 2F 43 48 41 52 47 45 G/CHARGE
═══════════ delete ════════    COM1:57600     0060 04 20 20 20 53 32 3D 4F.   S2=0
 Armed         A+B+C+D         targetpwr      0068 50 54 49 4F 4E 53 20 20 PTIONS
 idle    Inst brkpt/Illegal Address
═══════════════════════════════ DEBUG F10 ══════════════════════════════════════
 >BR 00A7
 >g
 >idle      Inst brkpt/Illegal Address
 >
══ F1:Help F2:Code F3:Mem F4:BSA_data F5:BSA_setup F8:vars  F9:rpt  F10:Debug ══
```

Figure 5. MMDS08 Debug Screen

Table 4. Key Functionality of Debug Screen Windows

Key	Description
F1	Activate the HELP pop-up window
F2	Activate the Code F2 window (if object code is displayed)
F3	Activate the Memory F3 window
F4	View the bus state analyzer data screen
F5	View the bus state analyzer setup screen
F8	Activate the Variables F8 window
F9	Repeat the preceding command
F10 or <ESC>	Returns to debug F10 window
↓	Scrolls the window down one line, same as clicking on the ↓ symbol at window edge
↑	Scrolls the window up one line, same as clicking on the ↑ symbol at window edge
Page Down	Scrolls the window down one page
Page Up	Scrolls the window up one page
Alt-X	Terminates host session
Alt-S	Writes screen contents to log file
Home	Scrolls the window to the home line
<Del>	Delete a highlighted variable in the Variables F8 window

Status Area

The status area, located at the left center of the debug screen, displays several items of status information. Table 5 explains the indicators that may appear in this area.

Table 5. Status Area Indicators

Indicator, Position	Status, Meaning
Bus analyzer state, left screen edge, below variables F8 window	**Armed** — bus analyzer is armed and waiting for a trigger **Disarmed** — bus analyzer is disarmed
Bus analyzer sequence mode, below variables F8 window	**Continuous all** — continuous trace, all cycles **Continuous events** — continuous trace, events only **Counted all** — counted trace, all cycles **Counted events** — counted trace, events only **A+B+C+D** — trigger on event A, B, C, or D **A+B>C+D** — trigger on event A or B, then C or D **A>B>C!D** — trigger on events A, B, and C, in order, unless event D occurs **A>B>C>D** — trigger on events A, B, C, and D, in order **Nth A+B+C+D** — trigger on Nth event A, B, C, or D
MCU state, left screen edge above debug F10 window	**Idle, Running, Stopped**, **Wait**, or **In Reset** followed by the reason for a status change
RESETIN signal state, below source/code F2 window	**Resetting** — Target system can reset emulating MCU **(blank)** — Target system cannot reset emulating MCU
RESETOUT signal state, between variables F8 and memory F3 windows	**Resetout** — RESET command resets emulating MCU and the target system **(blank)** — RESET command resets only the emulating MCU
Logging state, between variables F8 and memory F3 windows	**Logfile** — Logging in progress **(blank)** — Logging not in progress
Target system power, between variables F8 and memory F3 windows	**Target pwr** — Target system power is on **(blank)** — Target system power is not detected
Communications port and rate, above debug F10 window	**COMX:BBBBB** — The host software is communicating with the MMDS through port X, at BBBBB baud
Special status message, to the right of the MCU state status area, above debug F10 window	**Inst brkpt/Illegal Address** — A breakpoint or illegal address has been encountered and execution has halted **Write protect** — An attempt was made to write to memory designated as ROM or is unused memory space. Program counter will be at the next instruction that would have been executed had the error not occurred.

CPU Window

The CPU window is located at the upper left of the debug screen. This window displays the contents of the accumulator (A register), the index register (X register), the program counter (PC), the stack pointer (SP), and the condition code register (CCR). When a new value for any of these registers is entered, the new value appears in the window.

NOTE: *For MMDS08 users an additional register, the H register (upper byte of the index register), is shown as well as the V bit in the CCR.*

The CCR bit designators are located at the lower right of the CPU window. The CCR pattern is V11HINZC (V is two’s complement overflow for M68HC08 MCU only and is 1 for M68HC05 MCU, H is half-carry, I is interrupt mask, N is negative, Z is zero, and C is carry). A letter in these designators means that the corresponding bit of the CCR is set; a period means that the corresponding bit is clear.

Note that this window cannot be selected and cannot be used to change values. Instead, this window shows changes made via other windows or changes that occur due to running code.

Source/Code F2 Window

The window located in the upper right corner of the screen has two operating modes. The functionality of the window is different for each of the operating modes. Under certain conditions explained below, you may toggle between the operating modes. One mode (Code F2) displays the object code from a .S19 file loaded into the MMDS system. The other mode (Source) displays the source code from a .MAP file loaded into the MMDS system at the same time as the object code. More detail and a description of the differences in the operating modes are discussed below.

Code F2 Window

On entering MMDS software, the window defaults to object code, the window title is CODE F2, and window contents are a disassembled representation of MCU memory. In this object code display, the disassembled instructions change when corresponding bytes of memory change. To scroll through this window, press the F2 key (to select the window), then use the arrow keys, since the mouse does not function with this window.

Source Window

When a .MAP file exists in the same directory as the .S19 file, the MMDS software loads both files at the same time and the source code generated from the .MAP file is available for use in the debug process. The contents of this window change to source code (and the title changes to SOURCE:filename.asm) if:

1. A .MAP file has been loaded and
2. The program counter (PC) points to a memory area covered by the .MAP file.

Once a .MAP file has been loaded and the PC points to an area of user code, the SOURCE command can be used to toggle between source code and object code. If a mouse is installed, the symbols that appear at the bottom of the source window can be selected. Use the mouse or arrow keys to scroll through the information in the window.

There are several Alt-commands associated with the source window. The list of Alt-commands appears at the bottom of the debug screen when the <Alt> key is pushed. **Table 6** lists the key commands available in this window when a source code is displayed.

NOTE: *The F2 function key does not activate this window in the source code operating mode. The Source window and the Debug F10 window will both be active at the same time. Use the mouse or arrow keys to select/scroll the information in the Source window.*

Table 6. Source Window Key Commands

Name	Key	Description
Breakpoint	Alt-B	Sets or removes a breakpoint at highlighted line
Exit	Alt-X	Terminates debug session
Find	Alt-F	Finds the first occurrence of the specified string in the source file
Find Next	Alt-L	Finds the next occurrence of the specified string in the source file
GoTil	Alt-G	Executes code from the current PC address to the highlighted line
List Modules	Alt-M	Lists available source code modules
PC	Alt-P	Sets the program counter (PC) to the address on the highlighted line
Zoom	Alt-Z	Toggles the size of the source window between normal and enlarged

Variables F8 Window

The variables F8 window, located at the left side of the debug screen, is initially blank. The window shows as many as 11 variables, specified via the RTVAR or VAR command. Press the F8 function key to select this window so that the arrow keys can highlight existing variables. If more than 11 variables have been declared, use the arrow keys and the page up/down keys to display all variables. To delete a previously set variable, highlight the variable and press the delete key.

NOTE: *The delete operation can be performed via the mouse by selecting the variable and clicking on the word DELETE at the bottom of the window.*

It is possible to specify as many as 32 variables via the VAR and RTVAR commands. The variables appear with their current values in hexadecimal, binary, decimal, or ASCII format. Refer to the section entitled **Command-Line Commands** for more information about the RTVAR and VAR commands.

Memory F3 Window

The memory F3 window, located at the right side of the debug screen, displays the contents of 32 memory locations. The value stored at a specific location is displayed in both hexadecimal and ASCII format. In the ASCII area to the right in the window, control and other non-printing characters appear as periods (.). As the contents of these locations are modified, the new values appear in this window. Use the scroll bar to the right of the window to display other areas of memory.

To select this window, press the F3 function key. The scroll bar disappears; use the arrow keys and the page up/down keys to display lower or higher address ranges.

If real-time memory has been declared, the 32-byte block beginning at the declared location will be displayed while code is executing.

If real-time memory has not been declared, dashes replace the values when code is executing. Updated values reappear when execution stops.

Refer to the section entitled Command-Line Commands for more information about the RTMEM command.

Debug F10 Window

The debug F10 window, located at the bottom of the debug screen, is the default active window. This window contains the command line, identified by the command prompt (**>**). Enter (type) a command at the prompt. To process the command, press <CR> (that is, press the ENTER, RETURN, or carriage-return key). The software displays any additional prompts, messages, or data that pertain to the command. If the command is not entered correctly or is not valid, the software displays an appropriate error message. Refer to the section entitled **Command-Line Commands**f or a list and explanation of the of the available commands.

After command execution, the software again displays the command prompt. As a new line appears in the debug F10 window, preceding lines scroll upward. The window displays as many as four lines. When any other window is selected, the cursor disappears from the debug F10 window. To return to the debug F10 window, press the F10 function key.

The MMDS maintains a command buffer of commands entered on the command line. The mouse can be used to select the ↓ and ↑ arrow symbols to sequence forwards or backwards through the command buffer. Press the <CR> key to then repeat the command. To repeat the last command executed at any time, press the F9 key.

Pop-Up Windows

In addition to the five main windows, several temporary pop-up windows such as the bus state analyzer windows, the stack window, the set memory window, the baud window, and the emulator clock frequency window may appear during MMDS operation. **Table 7** lists the key commands for these subordinate windows.

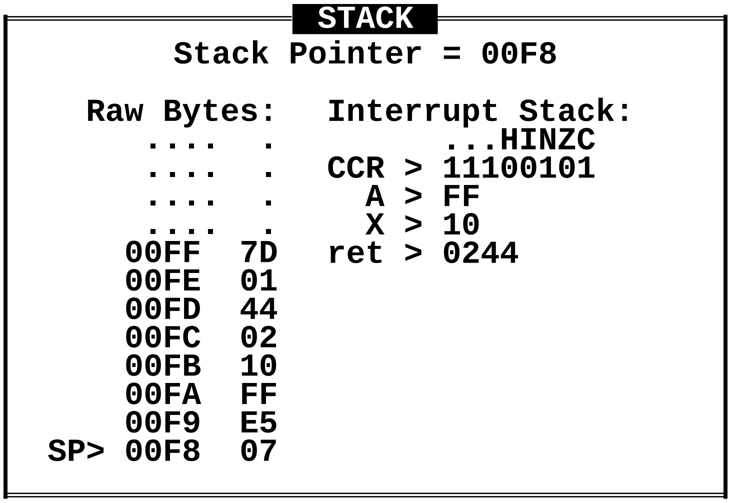
Table 7. Window Key Commands

Key	Description
↓	Moves cursor down one line
↑	Moves cursor up one line
←	Moves cursor left
→	Moves cursor right
Home	Moves cursor to top line of window
End	Moves cursor to bottom line of window
Page Down	Scrolls down one page (HELP only)
Page Up	Scrolls up one page (HELP only)
F6	Saves memory map to file (SETMEM only) and applies memory map to the MMDS
F7	Applies memory map to emulator and returns to debug screen (SETMEM only)
<CR>	Applies selection to emulator and returns to debug screen, except SETMEM For HELP, displays window for selected item For COLORS, accepts the existing color selection For STACK, returns to the debug screen
<ESC>	Returns to debug screen without applying selection to emulator For COLORS, returns to the debug screen without accepting any more colors For STACK, returns to the debug screen

Stack Window

The temporary stack window appears near the center of the debug screen when the STACK command is entered. As **Figure 6** shows, this window displays the contents of the SP register at the top of the window. The 12 bytes at the top of the stack are displayed to the left. The interpretation of the stack shown to right in the window is valid only if the last push to the stack was caused by an interrupt. Press the <ESC> key to remove the stack window and return to the debug window.

NOTE: *The interrupt stack data to the right side of the window is an interpretation of the top five bytes on the stack. If the last push to the stack was due to a BSR or JSR instruction, five bytes were not pushed to the stack and the interrupt stack information is invalid.*

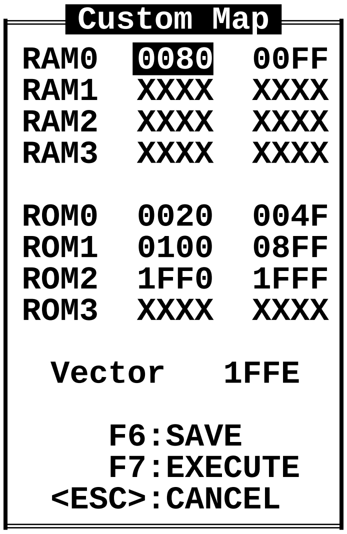


```
                     STACK
         Stack Pointer = 00F8

   Raw Bytes:   Interrupt Stack:
      ....  .         ...HINZC
      ....  .   CCR > 11100101
      ....  .     A > FF
      ....  .     X > 10
     00FF  7D   ret > 0244
     00FE  01
     00FD  44
     00FC  02
     00FB  10
     00FA  FF
     00F9  E5
 SP> 00F8  07
```

Figure 6. Stack Window

Set Memory Window

The temporary set memory window (Figure 7) appears near the center of the debug screen. Enter the set memory (SETMEM) command to customize the memory map. The SETMEM command allows mapping over memory defined as RAM, ROM, or undefined. However, mapping over internal resources such as option RAM, I/O, or EEPROM is not allowed.



```
         Custom Map
RAM0   0080   00FF
RAM1   XXXX   XXXX
RAM2   XXXX   XXXX
RAM3   XXXX   XXXX

ROM0   0020   004F
ROM1   0100   08FF
ROM2   1FF0   1FFF
ROM3   XXXX   XXXX

  Vector    1FFE

     F6:SAVE
     F7:EXECUTE
  <ESC>:CANCEL
```

Figure 7. Set Memory Window

Baud Window

The temporary baud window (**Figure 8**) appears near the center of the debug screen when the baud (BAUD) command is entered. The BAUD command sets the baud rate for communications between the system controller and the host computer. This window shows the available baud rates.

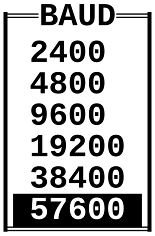


Figure 8. Baud Window

Emulator Clock Frequency Window

Depending on the EM board used, the MMDS platform board can supply the oscillator clock for the MCU's OSC1 input. This clock source can be an internally generated source or an external source input through the white logic clip of pod A. Note that the EM being used will require a specific jumper configuration in order to use the platform board clock source. Refer to the EM user's manual for the availability of this feature.

For the MMDS05, five clock frequencies are available. The four internally generated clock frequencies are: 8 MHz, 4 MHz, 2 MHz, and 1 MHz, and an external clock source. Entering emulator clock (OSC) command without the designated frequency brings up the temporary MMDS emulator clock frequency window near the center of the debug screen. Use the up/down arrow keys to select the emulator MCU's clock frequency and press <CR> to complete the selection. The default emulator clock rate is 2 MHz as shown in **Figure 9**.

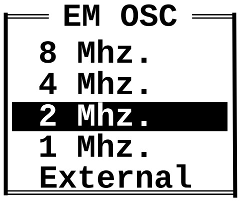


Figure 9. MMDS05 Emulator Clock Frequency Window

For the MMDS08, six clock frequencies are available. The five internally generated clock frequencies are available: 16 MHz, 8 MHz, 4 MHz, 2 MHz, and 1 MHz and an external clock source. The default emulator clock rate is 4 MHz as shown in **Figure 10**.

Before changnig the clock rate, make sure that the emulation MCU supports the desired frequency and the appropriate jumpers are set correctly.

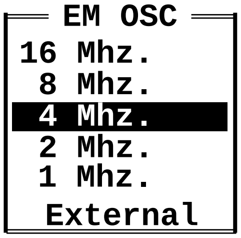


Figure 10. MMDS08 Emulator Clock Frequency Window

Other Windows

In addition to the screen windows described in this section, several other transient dialog windows will be encountered. Those windows encountered while using the bus state analyzer will be discussed in the **Bus State Analysis** section. Many are for file search selections in which the directory paths can be scanned for a desired file. Other windows will appear when using help commands. To select an item from a menu or file list, move the highlight cursor using the ↑ and ↓ keys to the desired item and press <CR>. To close a window without selecting an item, press the <ESC> key.

Mouse Operation

MMDS software supports a Microsoft, Logitech, or IBM mouse. Install the mouse according to the manufacturer's instructions, using the accompanying mouse driver software. Mice made by other manufacturers may be compatible, but Motorola does not guarantee their performance with the MMDS system.

Some MMDS operations can be accomplished by using an installed mouse to select a desired function. Note that the select symbols are only visible if a mouse is installed.

The mouse can be used to scroll through the source window, variables F8, memory F3, the bus state analyzer data F4, and debug F10 windows.

“Clicking on” an item means positioning the mouse cursor on the item, then quickly pressing and releasing the left mouse button. The mouse operations are:

- General
  - Scroll through a window — Click on the ↑ and ↓ symbols to right edge of the selected window.
- Variable F8 Window
  - Highlight items in the source and variables F8 windows — Move mouse over desired item and click.
  - Delete the highlighted variable in the variables F8 window — Click on the word DELETE at the bottom of the window.
  - Pressing the right button of the mouse to duplicate the functionality of the <ESC> key.

- Source Window
  - Set the PC to the address of the instruction on a highlighted line — Click on PC at the bottom of the source window.
  - Set or clear a breakpoint at the highlighted instruction in the source window — Click on BR at the bottom of the source window.
  - Begin executing instructions starting at the PC address — Click on GO at the bottom of the source window.
  - Stop executing instructions — Click on STOP at the bottom of the source window.
  - Execute instructions beginning with the instruction at the address in the PC and stopping at the highlighted instruction in the source F2 window — Click on GOTIL at the bottom of the source window.
  - Execute the instruction at the address in the PC — Click on STEP at the bottom of the source window.
  - Display the source file line number of the highlighted line of the source F2 window, along with its address, disassembled contents, and the name of the file — Click on INFO at the bottom of the source window.
  - Toggle the size of the source window between normal and enlarged — click on ZOOM at the bottom of the source window.

Changing Screen Colors

To change screen colors, enter the COLORS command from the debug screen; the colors window appears. This window includes a list of screen elements and a matrix of foreground/background color combinations. Each color combination has a 2-digit hexadecimal number.

A prompt asks for the color of the first screen element. To accept the current color, press <CR>. To change the color, enter the number of the choice, then press <CR>. A new prompt asks for the color of the next element. Select the color for each element in the same way. The command ends when a color has been selected for the last screen element or when ESC is pressed.

In the color matrix, rows correspond to background colors and columns correspond to foreground colors. This means that color choices from the same row result in differently colored letters and numbers against the same background color. Making the background of highlights and help screens a different color sets these elements off from the main screen.

The software stores color selections in file COLORS.05 or COLORS.08. When MMDS is executed again, the software applies the newly selected colors. Use the color selection file with another system to retain the selected colors.

NOTE: *Delete the COLORS file from the MMDS subdirectory to return to the default colors.*

Operation Fundamentals

Contents

Introduction 60

System Initialization. 60
  Setting Communications Baud Rate 61
  Standard Memory Mapping 61
  Custom Memory Mapping 62
  Initializing the Clock Speed 63
  Loading User Software 63
  Initializing Memory 63
    Initializing Assembly Language 64
    Initializing Memory Data 65

System Commands 66
  Script Commands 66
  Information Commands 67
  Log File Commands 68
  Debug Screen Control 69
  Exit the Environment 70

Debug Commands. 71
  Setting CPU Registers 71
  Memory Display 71
  Reset Control of the Emulation System 72
  Using Breakpoints 72
  Tracing Instructions 73
  Execution Instructions 73

Bus State Analyzer Commands. 74
  Configuring the Analyzer 74
  Capturing Analyzer Data 74
  Viewing Analyzer Data 74

Introduction

An emulation system gives the user the tools needed to develop an embedded MCU application in the most efficient way possible.

This section describes the basic operation of the MMDS. Detail of specific commands is available in **Command-Line Commands**.

Operation of the MMDS may be divided into four main areas:

- System Initialization
- System Information
- Debug Operation
- Bus State Analyzer Operation

A start-up script file, described in **Script Commands** on page 66**,** can be set up to perform a set of commands automatically each time the MMDS software is run. This start-up file must have the name STARTUP.05 or STARTUP.08.

System Initialization

Initializing the MMDS system includes:

- Initializing the communications baud rate
- Setting the memory map
- Initializing the clock speed
- Loading user software and a symbol table
- Initializing memory

Each part of initialization and use of the appropriate commands is discussed here.

Setting Communications Baud Rate

For best system performance, communications between the host and the station module should be at the maximum available baud rate. At power-up, the MMDS system automatically negotiates the maximum baud for the system. All data transfers between the host computer and the station module are at the specified baud rate; maximum performance is at the highest rate the computer supports. Use the BAUDCHK command to determine and set that rate. However, if the software displays communications error messages, reduce the baud rate.

Use the BAUD command to change the baud rate. The possible rates are 2400, 4800, 19200, 38400, and 57600 baud. If the BAUD command is entered with no rate value, the baud window appears over the debug screen. To select a rate from this window, use the arrow keys to highlight the rate, then press <CR> or double click the mouse when the cursor is on the desired baud rate.

Standard Memory Mapping

Various MCU designs require different memory map configurations of the MMDS system. The appropriate memory map is specified in a personality file for each MCU that the EM supports. These files are shipped with the separately purchased EMs. Refer to the appropriate EM user’s manual to determine the personality files used by a particular EM module. Personality files are usually installed in the directory from which the MMDS software is executed. If a personality file is not located in that directory, the software displays a search window used to find the correct file.

The MMDS operating software automatically loads the default personality file that corresponds to the EM module currently connected.

Custom Memory Mapping

For creating custom memory configurations, use the customize memory map (SETMEM) command. When this command is entered, the set memory window appears over the debug screen. Via this window, as many as four blocks of RAM and four blocks of ROM can be defined. (ROM is write-protected; attempting to write to ROM stops program execution.)

NOTE: *The SETMEM command can be used to expand the normal RAM and ROM ranges temporarily during debugging. Be sure to restore the original size and configuration of the MCU memory before final debugging. Otherwise, the code could fail to fit or run in an MCU's memory space.*

For each memory block, specify the address range and memory type. To write the map to a file, press the F6 function key, then enter a filename in response to the prompt. To prevent loss of system files, a custom filename should not duplicate any files shipped with an EM module. Press the F7 key to apply the map to memory without saving the map to a file for future use. To cancel the command, press <ESC>.

Use the load personality file (LOADMEM) command to load a stored custom map during future emulation sessions. Note that the LOADMEM command can be part of the start-up script file, so that loading the custom map becomes an automatic part of MMDS start-up (.MEM files also can be loaded at start-up by using the –M option with the MMDS executable.)

The LOADMEM command also can be used to restore the standard memory mapping. Refer to the appropriate EM user's manual to determine the default personality files used by a particular EM module.

Initializing the Clock Speed

The MMDS platform board can supply an oscillator clock source for the MCU’s OSC1 input. Clock control is available via the OSC command. Note that many EMs require a specific jumper configuration so that this clock source can be used. Refer to the specific EM user's manual for EM clock source information.

For the MMDS05, five clock frequencies are available: four internally generated clock frequencies, 8 MHz, 4 MHz, 2 MHz, and 1 MHz and an external clock source. The external clock source should be supplied via the white logic clip of pod A. Entering the emulator clock (OSC) command without the designated frequency brings up the temporary MMDS emulator clock frequency window near the center of the debug screen.

For the MMDS08, six clock frequencies are available: five internally generated clock frequencies, 16 MHz, 8 MHz, 4 MHz, 2 MHz, and 1 MHz, and an external clock source.

Loading User Software

Software for the target system must be available on the host computer in S-record format. Use the LOAD command to load an S-record file into the emulator and the accompanying map (symbol) file into the host computer. The assembler shipped with the MMDS (CASM) has the ability to generate a current map each time an assembly program is assembled.

The specific S-record to load can be specified on the command line by typing the S-record filename after the LOAD command. If the file has the extension.S19, then the extension can be omitted. Alternatively, if no filename is specified, then a search window appears, displaying the files with a.S19 extension in the current directory.

Initializing Memory

During a debugging session, specific memory locations should contain known values. The required values are stored in memory as numeric values or as instructions assembled individually. The commands that store and manipulate the contents of memory are described in the following paragraphs.

Initializing Assembly Language

The assemble instructions (ASM) command is important for making minor alterations to object code. This command displays the specified address and its contents followed by a prompt. Enter a valid instruction and press <CR>. The command assembles the code, stores the code in memory at the indicated address, and displays the instruction. The command then updates its location counter and displays the updated address and a prompt for the next instruction. The ASM command continues to assemble code one line at a time until a period (.) is entered.

NOTE: *Changes made to code via the ASM command cannot be saved to an S-record file or to a source code file. This command should be used only to create and modify code to be run during the current debug session.*

NOTE: *If the source/code F2 window shows source code and the ASM command is used to modify the code, the source/code F2 window continues to show unmodified source code. Enter the CLEARMAP command to delete the source code display. To incorporate modifications into source code, reassemble the code and download again.*

The disassemble instructions (DASM) command complements the ASM command. The DASM command allows memory contents to be disassembled, displaying the assembly instructions that correspond to the values in the specified memory address range. Each DASM command disassembles three instructions and displays the addresses, the opcodes, and the operands where appropriate. When the DASM command is entered with two addresses, it disassembles instructions beginning at the first address and ending with the instruction at the second address. If the range includes three or more instructions, only the last three disassembled instructions are displayed in the debug F10 window. The entire block is written to a log file when one is open.

Initializing Memory Data

The block fill (BF) command allows placement of numeric values in a block of memory addresses. This command defines a block of memory, then places a byte or word pattern throughout the range.

The memory modify (MM) command lets the user interactively examine and modify contents of memory locations. If any data arguments are entered with this command, the system stores the values beginning at the specified address.

When only an address is supplied, the command displays the contents of the address followed by a prompt. Enter the value and press <CR>. The command displays the next address and its contents. The command continues to store the values entered until a period (.) is entered.

Both the BF and MM writes to memory are verified; a "write did not verify" message is displayed if the write could not be verified. Note that this message may be acceptable in some situations, such as writing to registers that have write-only bits.

System Commands

System commands for the MMDS perform these functions:

- Executing commands contained in script files
- Obtaining information about the emulator and the host system
- Capturing and saving data displayed on the screen in a log file
- Controlling the format of the debug screen
- Leaving the MMDS environment temporarily or permanently

The following paragraphs cover usage of the system commands.

Script Commands

The execute script file (SCRIPT) command reads commands from a script file and passes them to the command interpreter for execution. Entering the one SCRIPT command has the same effect as entering the sequence of commands contained in the script file individually. Using script files saves time and promotes accuracy.

A script file is a text file of MMDS commands and is appropriate for any sequence of commands that is used often. A special script file, given the filename STARTUP.05 or STARTUP.08, executes automatically each time the MMDS software is loaded.

Sometimes a script file must contain a pause between commands. The pause (WAIT) command causes the command interpreter to wait before processing subsequent commands. As part of the WAIT command, the wait time can be entered in seconds. If a time value for the WAIT command is not entered, the command interpreter pauses for five seconds.

NOTE: *All values entered on the MMDS command line are hexadecimal. The input value 10, for example, is the decimal value 16.*

The BELL command will sound the computer bell the specified number of times. This is useful to let the user know script command execution has reached a certain point.

The REM command adds a display comment to a script file. When the script file is executed, the system displays this comment.

NOTE: *All other MMDS commands can be contained in the script file.*

Information Commands

The EVAL command performs mathematical operations on two numerical arguments. It displays the value of the result in hexadecimal, decimal, octal, and binary formats denoted by the suffixes H, T, O, and Q. If the value is equivalent to an ASCII character, the ASCII character also is displayed. This command supports addition (+), subtraction (–), multiplication (*) and division (/).

The REG command displays the contents of the CPU registers in the debug F10 window. The command also will display the instruction pointed to by the current program counter value.

The temporary stack window appears near the center of the debug screen when the STACK command is entered. This window displays the contents of the SP register at the top of the window. The 12 bytes at the top of the stack are displayed to the left. The interpretation of the stack shown to right in the window is valid only if the last push to the stack was caused by an interrupt.

NOTE: *The interrupt stack data to the right side of the window is an interpretation of the top five bytes on the stack. If the last push to the stack was due to a BSR or JSR instruction, five bytes were not pushed to the stack and the interrupt stack information is invalid.*

For information about a highlighted line in the source/code F2 window (filename, line number, address, and so forth) use the INFO command.

To display the value of a symbol defined in a map (symbol) file, use the WHEREIS command.

The display version (VER or VERSION) command displays the version number of the host software and the personality file.

The system information (SYSINFO) command shows the amount of host computer memory remaining.

The display memory map (SHOWMEM) command displays the RAM and ROM range of the current map.

The display help information (HELP) command displays a dialog window from which to access the MMDS help system. Note that the help system is context sensitive: Highlight an element of a screen, then press the F1 help key for corresponding help information.

The chip help information (CHIPINFO) command gives access to register and user vector locations, MCU memory-map, and pin-out information specific to the part being emulated. Note that this help information is only available on certain parts. A message is displayed if no help is available. Note that the help system is context sensitive: Highlight an element of a screen, then press the <CR> for corresponding help information.

The view file (VF) command allows for text file viewing within a pop-up screen.

Log File Commands

The MMDS can maintain a log file that will capture events displayed on the debug screen. Entries in the log include:

- Commands entered on the command line
- Commands read from a script file
- Responses to commands
- Error messages
- Notifications of changes in status, such as breakpoints and WAIT or STOP instructions.
- Pictures of the main screen
- Pictures of the bus state analyzer data screen

With the log file (LF) command, a file can be opened to receive information being logged. If the specified file already exists, the system allows appendage of the current log information to that file, or replacement of file contents with the current log information. While the log file remains open, the log information is written to the file. Enter another LF command to terminate logging to the file.

NOTE: *The LF command does not automatically append a filename extension to log files. Motorola recommends that the extension .log for log files be used.*

The save screen (SNAPSHOT) command will save the debug screen to an opened log file.

The first and last 20 records of analyzer data can be logged by first using the upload trace buffer (GETBSA) command to upload the analyzer data. The go to trace buffer start (HOMEBSA) command followed by a log bus state analyzer (SCREENBSA) command will save the opening records in the bus state analyzer data to an open log file. The go to trace buffer end (ENDBSA) command followed by a log bus state analyzer (SCREENBSA) command will save the ending records in the bus state analyzer data to an open log file.

Debug Screen Control

The source window display (SOURCE) command toggles between source code and disassembled code in the source/code F2 window located at the upper right of the debug screen. On entering MMDS software, the window defaults to disassembled code, the window title is CODE F2, and window contents are a disassembled representation of MCU memory. In this object code display, the disassembled instructions change when corresponding bytes of memory change. Source code will be displayed if an S-record file and its corresponding map file are loaded and the PC points to a memory area covered by the map file.

Once a .MAP file has been loaded and the PC points to an area of user code, the SOURCE command can be used to toggle between source code and object code. If a mouse is installed, the symbols that appear at the bottom of the window can be selected. Use the mouse or arrow keys to scroll through the information in the window. Note that the F2 key does not pertain to this window if it shows source code.

NOTE: *When memory data that was generated from a source file is altered, the modified code appears in the code window but not the source file window. Use the CLEARMAP command to delete the source file from the source code display.*

The resize source window (ZOOM) command toggles the size of the source window between normal and enlarged. The enlarged window can be helpful by allowing improved visibility of comments in the source file. The enlarged window will remove the CPU registers window from the debug screen. The registers will be re-displayed by typing the ZOOM command again.

The set screen colors (COLORS) command can be used to alter the default screen colors displayed in the various windows of the emulation environment. To return to the default colors, delete the filename COLORS.05 or COLORS.08 in the MMDS working directory.

Exit the Environment

The debug environment can be exited either temporarily or permanently. To shell to DOS temporarily, use the SHELL command. Type EXIT at the DOS prompt to return to the MMDS environment.

To exit the current debug session permanently, execute the EXIT or QUIT command. The emulation system also can be exited by pressing the Alt-X keys.

Debug Commands

The MMDS commands that apply to the debugging phase of system development are described in this section.

Setting CPU Registers

The contents of the CPU registers and the condition code register are displayed in the CPU window. These registers – A, H (MMDS08 only), X, PC, SP, and CCR – contain the environment for execution of an instruction and, after the instruction has been executed, the results. Any of these registers, except SP, can be modified by entering the corresponding register designator command and an appropriate value. The commands that affect the CPU registers are A, ACC, X, XREG, PC, CCR, H, I, N, Z, and C. Additional commands to support the M68HC08 MCU are HX, HREG, and V. When <CR> is pressed, the register display shows the new value. Refer to **Command-Line Commands** for examples on how to modify CPU register values.

Memory Display

Memory contents are displayed in the memory F3 window. When code execution is stopped, 32 consecutive bytes of memory are displayed in both hexadecimal and ASCII format. The memory display (MD) command specifies the beginning location of the 32 bytes displayed. The window can be scrolled via the mouse or by selecting the window (F3) and using page up/down keys to view other memory ranges.

A special use of the memory F3 window is for displaying real-time memory during code execution. The display real-time memory (RTMEM) command specifies the beginning location of a 32-byte clock to display and periodically update during code execution.

The display real-time variable (RTVAR) command and display variable (VAR) command display the specified address and its contents in the variables F8 window. Real-time variables are periodically updated during code execution while normal variables are only displayed when code execution is stopped.

As many as 32 variables can be declared in the variables F8 window. The window shows 11 variables at a time. If a map file has been loaded, symbols (labels) from the source code can be used as arguments.

The variables can be displayed in byte, word, or string format. A byte display is hexadecimal and binary, while a word display is hexadecimal and decimal, and a string display is ASCII.

For an ASCII string, the number of characters displayed can be specified. Control and other non-printing characters appear as periods (.).

Reset Control of the Emulation System

The RESET command resets the emulation MCU and sets the PC to the contents of the reset vector. User code is not executed during this command. The RESETGO command carries out the same actions as the RESET command, then starts code execution from the PC-value address. The RESETIN command allows the reset signal to enter into the emulation system through the target cable; this signal must be enabled for correct operation of the WAIT4RESET command. The RESETOUT command allows the RESET command to send a reset signal out the target cable.

Using Breakpoints

The set instruction breakpoint (BR) command sets a breakpoint at a specific address or at each address of a range. Breakpoint addresses must be instruction fetch (opcode) addresses. A maximum of 64 breakpoints can be set. If the BR command is entered without any address, the command displays all active breakpoints. To clear breakpoints, use the clear breakpoints (NOBR) command.

An instruction breakpoint occurs when the MCU accesses an instruction at a specified address or an address within a specified address range. When execution arrives at a breakpoint address, emulation stops just before execution of the instruction at that address and the software displays this message:

```
idle      Inst brkpt/Illegal Address
```

A properly defined breakpoint permits analysis of the contents of registers and memory locations and the states of various signals at designated addresses in the program.

NOTE: *The idle status also occurs if the system attempts to execute code at an address not defined as a valid memory address.*

Tracing Instructions

The step (ST, STEP, and T are identical) commands will execute a specified number of instructions beginning at the current PC value. The STEPFOR command begins instruction execution at the current PC value, continuing until a key is pressed or until execution arrives at a breakpoint. The STEPTIL command executes instructions from the current PC value to a specified address.

NOTE: *Do not use any step command (ST, STEP, STEPFOR, STEPTIL, or T) if the PC points to internal RAM (such as option RAM) or EEPROM or if the code branches into internal RAM or EEPROM.*

The step commands are not real-time. They execute one instruction at a time, then return control to the monitor. Do not rely on timer values.

Execution Instructions

The go (G or GO) command starts emulation at the address in the PC or at an address entered with the command. Execution continues until it encounters a breakpoint, until the bus analyzer (optionally) stops it, or until the STOP command is entered. If a second address is entered with the G or GO command, execution stops at the second address. The GOTIL command starts emulation at the location in the PC and stops at the address entered with the command. The STOP command stops the emulator.

The RESETGO command resets the MCU, fetches the reset vector address, and begins code execution at that address.

Bus State Analyzer Commands

The bus state analysis (BSA) feature of the MMDS allows for analyzing the operation of the MCU being emulated. The address, data, free-running timer value, and select test points can be captured and analyzed using this powerful feature. This section introduces the analyzer commands, with a more detailed view of analyzer usage given in **Bus State Analysis**.

Configuring the Analyzer

The BSA is configured initially by pressing F5 to enter the BSA setup screen. Various trigger points and trigger modes can be configured in the setup screen. The trigger configurations can be saved and used during future debug sessions using the load bus state analyzer setup (LOADTRIGGERS) command.

The timetag clock source (TIMETAG) command configures the resolution of the free-running timer value captured in the analyzer.

The set multiplexer (SXB) command controls the option to either increase the bandwidth of the free-running timer counter or enable capturing additional target test points through pod B.

Capturing Analyzer Data

Once the analyzer is set up to a desired trigger configuration, the arm bus state analyzer (ARM) command enables the analyzer to begin capturing bus cycles in the trace buffers.

The analyzer can be configured to disarm after the analyzer has found the trigger and subsequently filled its trace buffer. Alternatively, the disarm bus state analyzer (DARM) command deactivates the analyzer bus cycle captured from the command line.

Viewing Analyzer Data

The captured BSA data is stored initially in trace buffer memory on the MMDS platform board. The data is uploaded to the host computer and viewed on screen by pressing the F4 function key. Alternatively, the upload trace buffer (GETBSA) command will upload the analyzer data without displaying it to the screen. The set BSA timetag mode (BSATT) command will specify the format for displaying timetag data in the analyzer data screen. The set BSA display mode (BSAMODE) command will specify the format for displaying records in the analyzer data screen. Subsequently, pressing the F4 key will display the data to the screen without requiring another data upload.

The analyzer can be set up to detect up to four trigger events, referred to as event A, event B, event C and event D. If a captured record is one of the four trigger events, the event will be displayed in the BSA data screen. To quickly see where an event A first occurred in a set of captured analyzer data, use the go to next A event (NEXTA) command to display the bus analyzer data of the first A event. Executing the command again will find and display the next event occurrence. If no more events exist, the last captured cycle is displayed. Identical commands for events B, C, and D are also available.

The go to next event (NEXTE) command searches for and displays the next occurrence of any of the four possible events.

The print trigger (SHOWTRIGGER) command finds the first occurrence of a trigger event in the bus state analyzer data.

The first and last 20 records of analyzer data can be saved to an open log file by first using the upload trace buffer (GETBSA) command to upload the analyzer data. The go to trace buffer start (HOMEBSA) command followed by a log bus state analyzer (SCREENBSA) command will save the opening records in the bus state analyzer data to an open log file. The go to trace buffer end (ENDBSA) command followed by a log bus state analyzer (SCREENBSA) command will save the ending records in the bus state analyzer data to an open log file.

The display trace buffer (SHOWBSA) command will dump ranges of analyzer data to the debug window.

Bus State Analysis

Contents

Introduction 78

Operating the Bus Analyzer. 79
  Defining Events 79
  Selecting the Trigger Mode 84
  Selecting Options 86
  Collecting Bus Data 87
  Viewing Data 88
  Searching the Trace Buffer 95
  Using the Time-Tag Clock 96

Introduction

The MMDS bus state analyzer (BSA) shows the logical state of the target MCU bus. Next to emulation of a target-system MCU, this is the most important capability of a development tool: it enables you to determine what is occurring in a system without actually disturbing the system.

At the end of each MCU clock cycle, the BSA takes a snapshot of the logical states of the target MCU bus. Then the analyzer stores the snapshots in the trace buffer, according to its mode. (This action is known as storing cycles.) The trace buffer can hold as many as 8191 cycles. (Note that the analyzer is a *bus state* analyzer: It does not show signal hold or setup times.)

As part of analyzer initialization, you define certain patterns of logical states as events (or terms). Then you select the analyzer mode: continuous, counted, or any of five sequential modes. This determines which cycles the analyzer stores.

Data collection (cycle storage) begins when you arm the analyzer and start program execution. Data collection continues until execution stops, through a specified number of events, or through a defined sequence of events.

The bus state analyzer provides several ways to view collected data: raw data, disassembled instructions, mixed raw data and disassembled instructions, or source code.

Operating the Bus Analyzer

To operate the bus state analyzer, you must define events (or terms), select the bus state analyzer mode, specify any options, collect data, then view the data. The following paragraphs explain these actions.

Defining Events

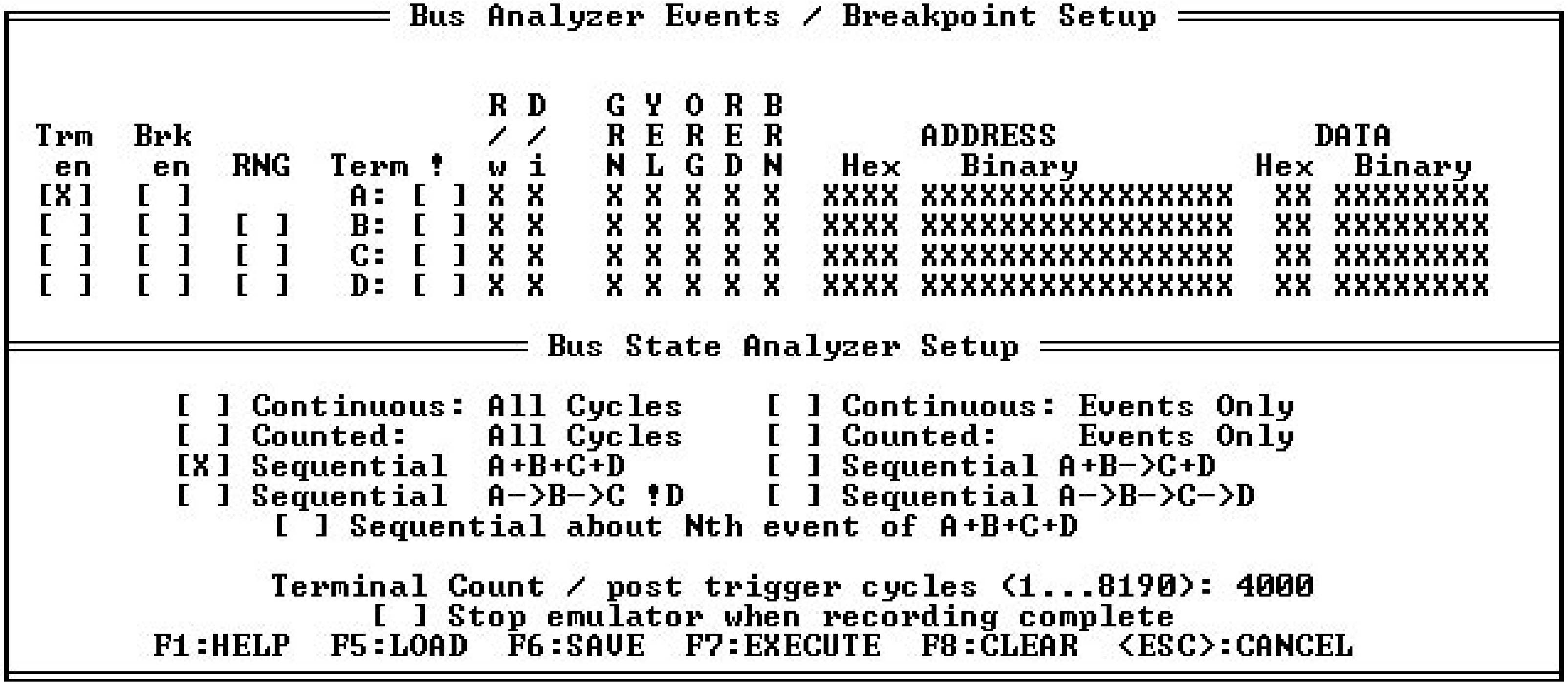
An event, is a 32-bit value, named A, B, C, or D. You define an event by entering values in one of the term lines of the bus state analyzer setup screen (see **Figure 11**). To bring up this screen, press the F5 function key from the debug screen. **Table 8** lists event-definition values and their meanings; **Table 9** lists key commands for the setup screen.

When the setup screen first appears, the cursor is at the **Trm en** (term enable) field of the event A line. Using the arrow keys or mouse, move the cursor to the space between the **Trm en** brackets for the desired term. Press the space bar (or point to the space with the cursor and click the left mouse button) to put an X in this space. Then move the cursor to other fields to enter values that define the rest of the event. For control signal and logic clip fields, type 0, 1, or X (don't care). For the address and data, use either the hexadecimal field or the binary field, Type a hexadecimal digit or X in the hexadecimal field spaces; type a 0, 1, or X in the binary field spaces.

NOTE: *Only five of the pod A logic clips are used in event definition. Other logic clips can be captured only in the logic analyzer data.*

When you have defined your terms, press the F7 key to apply the definitions. If you want to save the definitions to a file, press F6 (then enter a filename in response to the prompt) before you press F7.

NOTE: *If you use the backspace or delete key while in a field, you must completely refill the field with 0, 1, or X or the software will not allow you to leave the field.*



```
══════════════ Bus Analyzer Events / Breakpoint Setup ══════════════

                           R D  G Y O R B
Trm  Brk                   / /  R E R E R      ADDRESS            DATA
 en   en   RNG   Term !    w i  N L G D N  Hex   Binary          Hex  Binary
[X]  [ ]          A: [ ]   X X  X X X X X  XXXX XXXXXXXXXXXXXXXX  XX XXXXXXXX
[ ]  [ ]   [ ]    B: [ ]   X X  X X X X X  XXXX XXXXXXXXXXXXXXXX  XX XXXXXXXX
[ ]  [ ]   [ ]    C: [ ]   X X  X X X X X  XXXX XXXXXXXXXXXXXXXX  XX XXXXXXXX
[ ]  [ ]   [ ]    D: [ ]   X X  X X X X X  XXXX XXXXXXXXXXXXXXXX  XX XXXXXXXX
══════════════════ Bus State Analyzer Setup ════════════════════════

      [ ] Continuous: All Cycles    [ ] Continuous: Events Only
      [ ] Counted:    All Cycles    [ ] Counted:    Events Only
      [X] Sequential  A+B+C+D       [ ] Sequential A+B->C+D
      [ ] Sequential  A->B->C !D    [ ] Sequential A->B->C->D
           [ ] Sequential about Nth event of A+B+C+D

           Terminal Count / post trigger cycles (1...8190): 4000
                [ ] Stop emulator when recording complete
    F1:HELP  F5:LOAD  F6:SAVE  F7:EXECUTE  F8:CLEAR  <ESC>:CANCEL
```

Figure 11. Bus State Analyzer Setup Screen

Table 8. Event Definition Values

Field	Values	Meaning
Trm en – Term enable	X Blank	Enable the term Disable the term
Brk en – Breakpoint enable	X Blank	Make the term a data breakpoint (hardware breakpoint) (1) Do not make the term a breakpoint
RNG – Range	X Blank	Makes the event the end of a range Range does not apply to the event
! – Negation	X Blank	Complements the term Does not complement the term
R/W – Read/Write	0 1 X	MCU write cycle MCU read cycle Either read or write cycle
D/I – Data/Instruction	0 1 X	Instruction fetch Data Don't care
Pod A Clips: GRN (Green), **YEL** (Yellow), **ORG** (Orange), **RED** (Red), **BRN** (Brown)	0 1 X	Logic level 0 Logic level 1 Don't care
Address Hex	0–F, X	Hexadecimal address value (X is don't care)
Address Binary	0, 1, X	Binary address value (X is don't care)
Data Hex	0–F, X	Hexadecimal data value (X is don't care)
Data Binary	0, 1, X	Binary data value (X is don't care)
(1) You may have to disarm the BSA before you can see data in the BSA data screen.		

Table 9. Setup Screen Key Commands

Name	Key	Description
Move Down	↓	Moves cursor down to lower line.
Move Up	↑	Moves cursor up to higher line.
Move Left	←	Moves cursor to selection to the left.
Move Right	→	Moves cursor to selection to the right.
Next Item	Tab	Moves cursor to next item
Preceding Item	Shift-Tab	Moves cursor to preceding item
HELP	F1	Displays Help window.
LOAD	F5	Loads a trigger file (.SET).
SAVE	F6	Writes definitions to a trigger file (.SET).
EXECUTE	F7	Applies definitions to bus analyzer and returns to Debug screen.
CLEAR	F8	Clears definitions.
CANCEL	ESC	Cancels definitions and returns to Debug screen.

Part of event definition can be defining ranges from one term to another. To establish a range, you put an X in the range field of a term-definition line of the setup screen. But note that the event A line does not have a range field. This is because event A can only start a range.

To configure any of the four range patterns:

A to B:	Put an X in the event B range field.
B to C:	Put an X in the event C range field.
C to D:	Put an X in the event D range field.
A to B and C to D:	Put Xs in the range fields of events B and D.

Also note that you need not define all four terms. When you have defined all appropriate terms, you are ready to select the bus state analyzer trigger mode per **Selecting the Trigger Mode**.

Remember that the MMDS stores event definitions as 32-bit values. The R/W bit is the most significant bit (MSB); the D0 bit is the least significant bit (LSB). A range is between two such 32-bit values, *not* between values of address fields. In range mode, the BSA triggers every time the input falls between the range starting term (the first 32-bit value) and the range ending term (the second 32-bit value).

Selecting the Trigger Mode

To select a mode, put an X in one of the nine mode fields in the bottom half of the bus state analyzer setup screen. Table 10 explains the modes.

Table 10. Analyzer Modes

Mode	Description
Continuous: all cycles	When you enter the ARM and GO commands, the trace buffer begins storing data from first cycle. This continues until execution arrives at a breakpoint, or until you enter the DARM or STOP command.
Continuous: events only	When you enter the ARM and GO commands, the trace buffer begins storing data when data matches an event definition. This continues until execution arrives at a breakpoint, or until you enter the DARM or STOP command.
Counted: all cycles	When you enter the ARM and GO commands, the trace buffer begins storing data from the specified number from first cycle. (A breakpoint can stop storage before the analyzer stores the specified number of cycles, as can the DARM or STOP command.)
Counted: events only	When you enter the ARM and GO commands, the trace buffer begins storing data that match an event definition for the specified number of cycles. (A breakpoint can stop storage before the analyzer stores the specified number of cycles, as can the DARM or STOP command.)
A+B+C+D	When you enter the ARM and GO commands, the trace buffer begins storing data from the first cycle run. This continues through the occurrence of event A, B, C, or D (whichever is enabled); data storage ends after the specified number of post-trigger cycles.
A+B→C+D	When you enter the ARM and GO commands, the trace buffer begins storing data from the first cycle. This continues through the occurrence of two events: A or B, followed by C or D. Data storage ends after the specified number of post-trigger cycles. If you select this mode, you must enable event A, event B, or both. You must enable event C, event D, or both. Otherwise, the bus state analyzer never can be triggered.

Table 10. Analyzer Modes (Continued)

Mode	Description
A→B→C!D	When you enter the ARM and GO commands, the trace buffer begins storing data from all cycles. This continues through the occurrence of three events, A, B, and C, in order, if event D does not occur. (If D occurs, the sequencer starts again looking for event A.) Data storage ends after the specified number of post-triggert cycles. If you select this mode, you must enable events A, B, and C. Otherwise, the bus state analyzer never can be triggered. If you disable event D, you convert this mode to a simple, three-event sequence.
A→B→C→D	When you enter the ARM and GO commands, the trace buffer begins storing data from all cycles. This continues through the occurrence of four events, A, B, C, and D, in order. Data storage ends after the specified number of post trigger cycles. If you select this mode, youmust enable all four events A, B, C, then D. Otherwiser, the bus state analyzer never can be triggered.
Nth event: A+B+C+D	When you enter the ARM and GO commands, the trace buffer begins storing data from N occurrences of cycles that match the definitions of events A, B, C, or D (whichever are enabled). Then the bus state analyzer captures the next 4096 cycles.

By selecting the terminal post trigger count, the user can control the number of cycles that is stored. This can be used to speed uploading of the BSA data if only a small portion of data is needed.

Note that the terminal count or post trigger cycles are valid only for counted or sequential modes. For a counted mode, this field specifies the number of cycles to be stored. For a sequential mode, this field specifies the number of cycles to be stored *after* the trigger sequence occurs.

An X in the stop-emulator field stops program execution when bus state analyzer recording is done.

After selecting the mode, you can begin collecting data by pressing the F7 (execute) key. This returns you to the debug screen. **Selecting Options** explains how to continue from this point.

Other function keys in the setup screen allow for clearing of the setup screen as well as storing and using setup files. To cancel the entire bus state analyzer setup, press <ESC>. To clear the setup screen, making it ready to redefine events and reselect a mode, press the F8 key.

To save this bus state analyzer setup to a file, press the F6 key: a subordinate window prompts for a filename. To load a bus state analyzer setup already saved to a file, press the F5 key: a subordinate window prompts for the filename. Entering the filename fills in the setup-screen values; press the F7 key to return to the debug screen. (An alternative way to load a saved setup is to enter the LOADTRIGGERS command from the debug screen. This method bypasses the setup screen.)

Selecting Options

An optional part of analyzer setup is specifying the frequency and source of the time tag clock. This clock provides a time reference value in each frame of the trace buffer. (**Using the Time-Tag Clock** gives more information about the time tag clock.) Enter the time tag clock source (TIMETAG) command; this command brings up the small time-tag window in the center of the debug screen. This window gives you these choices (16 Mhz is the default):

16 Mhz	Selects the 16 MHz oscillator.
8 Mhz	Selects the 8 MHz oscillator.
4 Mhz	Selects the 4 MHz oscillator.
2 Mhz	Selects the 2 MHz oscillator.
1 Mhz	Selects the 1 MHz oscillator.
External	Selects the external clock
Programmable	Selects the programmable clock.
Emulator	Selects the emulator clock, the bus clock of the emulating MCU.

If you select *External*, connect the TT_OSC clip (white) of the pod B cable to the external clock source. (The pod B connector is the closest to the front of the station module.

If you select *Programmable*, you must enter a frequency in the range of 50 Hz to 50 kHz, as the pop-up window requests. The MMDS will provide a frequency close to the desired frequency and display it to the screen. For example, entering 1000 will result in a timetag frequency of 1024 Hz.

If you select *Emulator*, the system uses the MCU's bus clock. In effect, this stores the number of bus cycles.

Another setup option is specifying whether to store high-order time tag bits (increasing the time tag from 16 to 24 bits) or data from the pod B logic clips. To do so, enter the set multiplexer (SXB) command with the appropriate tags or clips parameter value. (The default is clips.)

Collecting Bus Data

To begin data collection, enter the ARM command, which arms the bus state analyzer. The BSA status changes to `Armed`. The bus state analyzer mode appears on the status line.

Next, enter the GO command, which starts program execution. The MCU status changes to `Running`. If you are in a sequential mode, you may be able to follow the occurrence of events from the highlighting changes. (Such highlighting changes may be too fast to be helpful.) Data collection continues through the specified number of counted events or post-trigger cycles, or until code execution stops.

NOTE: *The GO command is not the only program-execution command that works with the bus state analyzer. Alternative commands are: G, GOTIL, STEP, STEPFOR, STEPTIL, and T.*

If you enter either trace command (STEP or T) without a parameter value when the bus state analyzer is armed, the analyzer trace window appears over the debug screen. This temporary window shows the cycles of the instruction just traced.

To manually halt data collection, enter the DARM or STOP command. Entering the DARM command disarms the analyzer; the analyzer state changes to `Disarmed`. (The DARM command does not stop emulation.) Entering the STOP command stops data collection and emulation.

When data collection stops, you are ready to look at the data, per **Viewing Data**.

Viewing Data

To view bus analyzer data, press the F4 function key from the debug screen; this key brings up the bus state analyzer data screen (**Figure 12**). The word **loading** flashes in the upper right corner of the screen as the software loads trace-buffer contents into the host computer. The data is not fully visible until loading is done (**loading** stops flashing), although cycles immediately preceding and following the trigger cycle become valid early during loading.

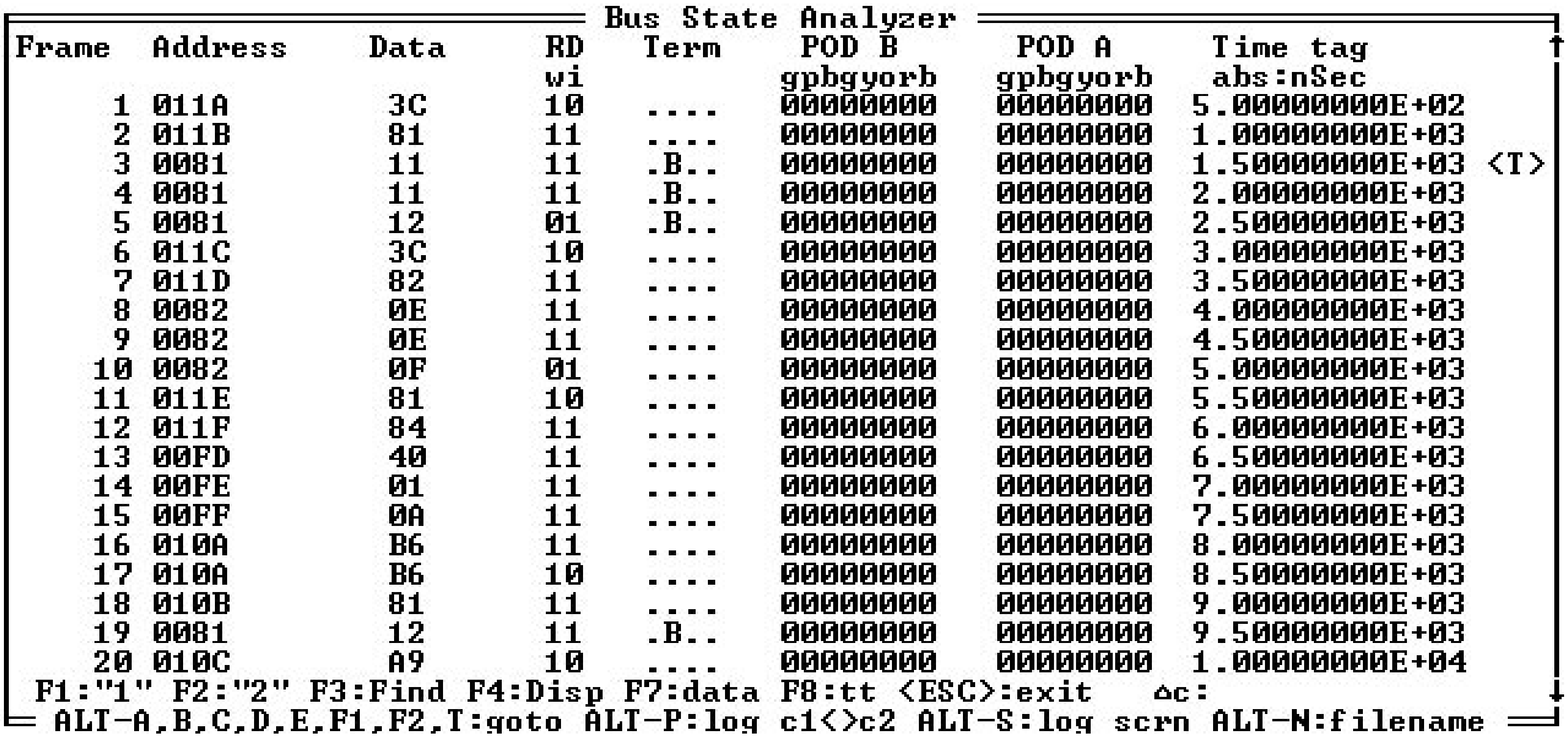
NOTE: *Be sure to use the highest possible baud rate. If you use a lower baud rate, it can take several minutes to load trace-buffer data into the host computer.*

The data screen displays trace buffer contents as raw bus cycles, as disassembled instructions, as mixed instructions and raw bus cycles, or as source code. In the mixed display, the associated raw bus cycles follow each disassembled instruction. Press the F4 key repeatedly to change the display from one form to another. **Table 11** explains other key commands for the data screen.

If the data capture mode was sequential, the data screen includes a trigger indicator (**<T>**). This screen indicator separates the pre-trigger and post-trigger cycles.

The F1 and F2 keys mark cycles <1> and <2>, respectively. The bus state analyzer uses these marked cycles in time-tag difference calculations and logging. The software displays the time tag difference, Δ**c**, in the lower right corner of the screen. (An **R**, by the Δc value, indicates a rollover of the time tag value between the occurrence of cycles <1> and <2>.)

If a log file is open, you can save bus state analyzer data to the log file. The system logs the information in the selected view mode. While logging is under way, the SHOWTRIGGER, NEXTA, NEXTB, NEXTC, NEXTD, and NEXTE commands log trace buffer cycles. To copy the current data screen to the log file, use the Alt-S key command. Use the Alt-P key command to log from the <1> cycle to the <2> cycle.



```
                        Bus State Analyzer
Frame  Address   Data    RD   Term    POD B     POD A     Time tag
                         wi          gpbgyorb  gpbgyorb   abs:nSec
    1 011A        3C     10   ....   00000000  00000000  5.00000000E+02
    2 011B        81     11   ....   00000000  00000000  1.00000000E+03
    3 0081        11     11   .B..   00000000  00000000  1.50000000E+03 <T>
    4 0081        11     11   .B..   00000000  00000000  2.00000000E+03
    5 0081        12     01   .B..   00000000  00000000  2.50000000E+03
    6 011C        3C     10   ....   00000000  00000000  3.00000000E+03
    7 011D        82     11   ....   00000000  00000000  3.50000000E+03
    8 0082        0E     11   ....   00000000  00000000  4.00000000E+03
    9 0082        0E     11   ....   00000000  00000000  4.50000000E+03
   10 0082        0F     01   ....   00000000  00000000  5.00000000E+03
   11 011E        81     10   ....   00000000  00000000  5.50000000E+03
   12 011F        84     11   ....   00000000  00000000  6.00000000E+03
   13 00FD        40     11   ....   00000000  00000000  6.50000000E+03
   14 00FE        01     11   ....   00000000  00000000  7.00000000E+03
   15 00FF        0A     11   ....   00000000  00000000  7.50000000E+03
   16 010A        B6     11   ....   00000000  00000000  8.00000000E+03
   17 010A        B6     10   ....   00000000  00000000  8.50000000E+03
   18 010B        81     11   ....   00000000  00000000  9.00000000E+03
   19 0081        12     11   .B..   00000000  00000000  9.50000000E+03
   20 010C        A9     10   ....   00000000  00000000  1.00000000E+04
 F1:"1" F2:"2" F3:Find F4:Disp F7:data F8:tt <ESC>:exit   ∆c:
  ALT-A,B,C,D,E,F1,F2,T:goto ALT-P:log c1<>c2 ALT-S:log scrn ALT-N:filename
```

Figure 12. Bus State Analyzer Data Screen

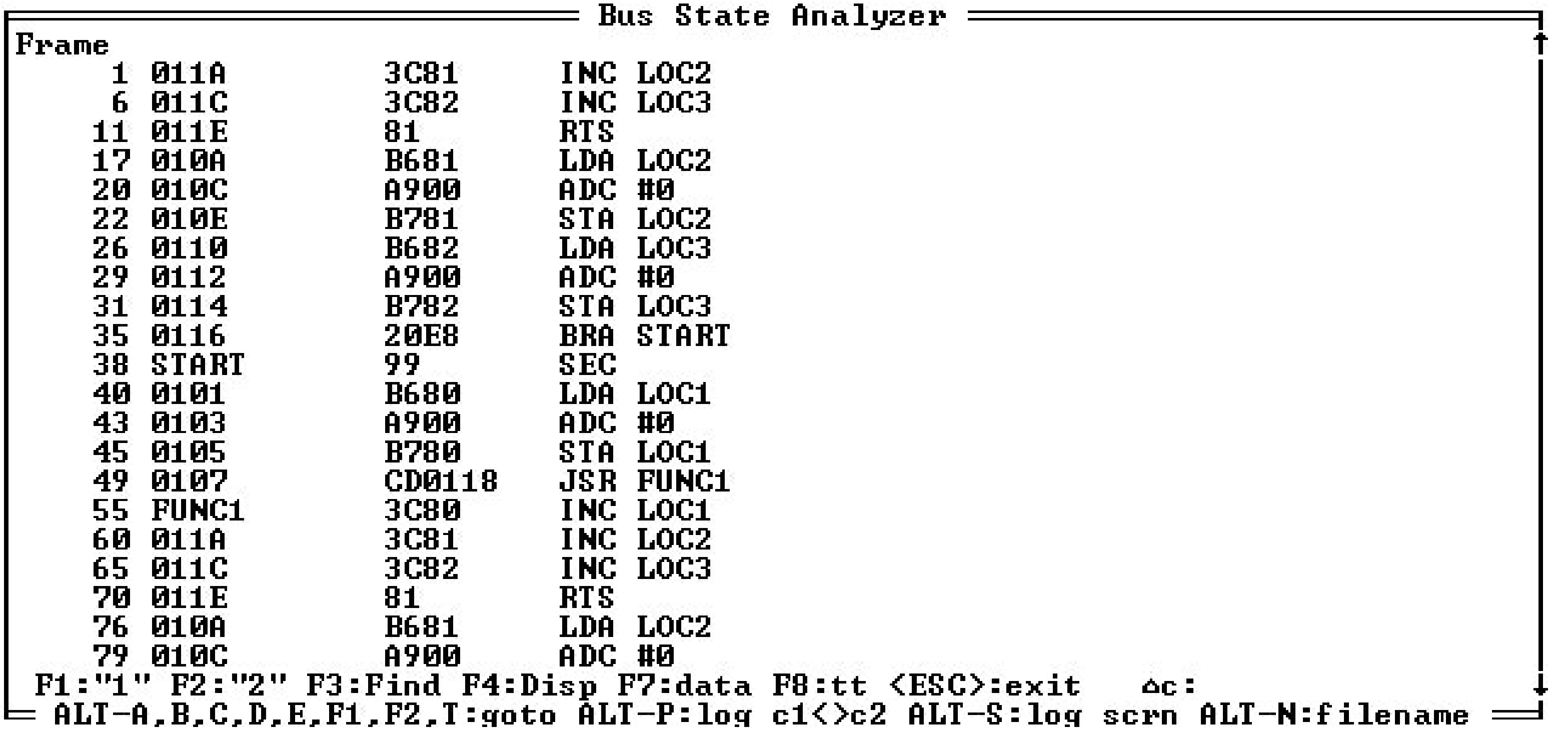
Table 11. Data Screen Key Commands

Name	Key	Description
Scroll Down	↓	Scrolls cursor down to next line.
Scroll Up	↑	Scrolls cursor up to preceding line.
Page down	Page Down	Scrolls down to next page.
Page up	Page Up	Scrolls up to preceding page.
Home	Home	Scrolls to first frame.
End	End	Scrolls to highest-numbered frame.
Next A	Alt-A	Scrolls to next term A frame.
Next B	Alt-B	Scrolls to next term B frame.
Next C	Alt-C	Scrolls to next term C frame.
Next D	Alt-D	Scrolls to next term D frame.
Next E	Alt-E	Scrolls to next frame that contains any term.
"1"	F1	Marks highlighted frame as cursor 1.
"2"	F2	Marks highlighted frame as cursor 2.
Go to cursor 1	Alt-F1	Scrolls to cursor 1.
Go to cursor 2	Alt-F2	Scrolls to cursor 2.
Go to trigger	Alt T	Scrolls to trigger frame.
Find	F3	Defines a search pattern and scrolls to frame that matches pattern.
Disp	F4	Changes display mode to next in sequence: Raw, Instructions, Mixed, Source.
data	F7	Toggles display in Data column between hexadecimal and binary.
tt	F8	Changes time tag mode to next in sequence: absolute, relative, none, cycles.
Log cursor 1 - cursor 2	Alt-P	Writes frames from cursor 1 through cursor 2 to log file.
Log screen	Alt-S	Writes the frames displayed on the screen to log file.
Return	ESC	Return to Debug screen.
Display source name	Alt-N	Display the source name: line number

Figure 12 shows the data screen as it displays raw bus cycles. **Figure 13** shows this screen's display of instructions, **Figure 14** shows a mixed instructions and raw bus cycle display, and **Figure 15** shows this screen's display of source code. (Repeatedly press the F4 key to cycle through display modes.)

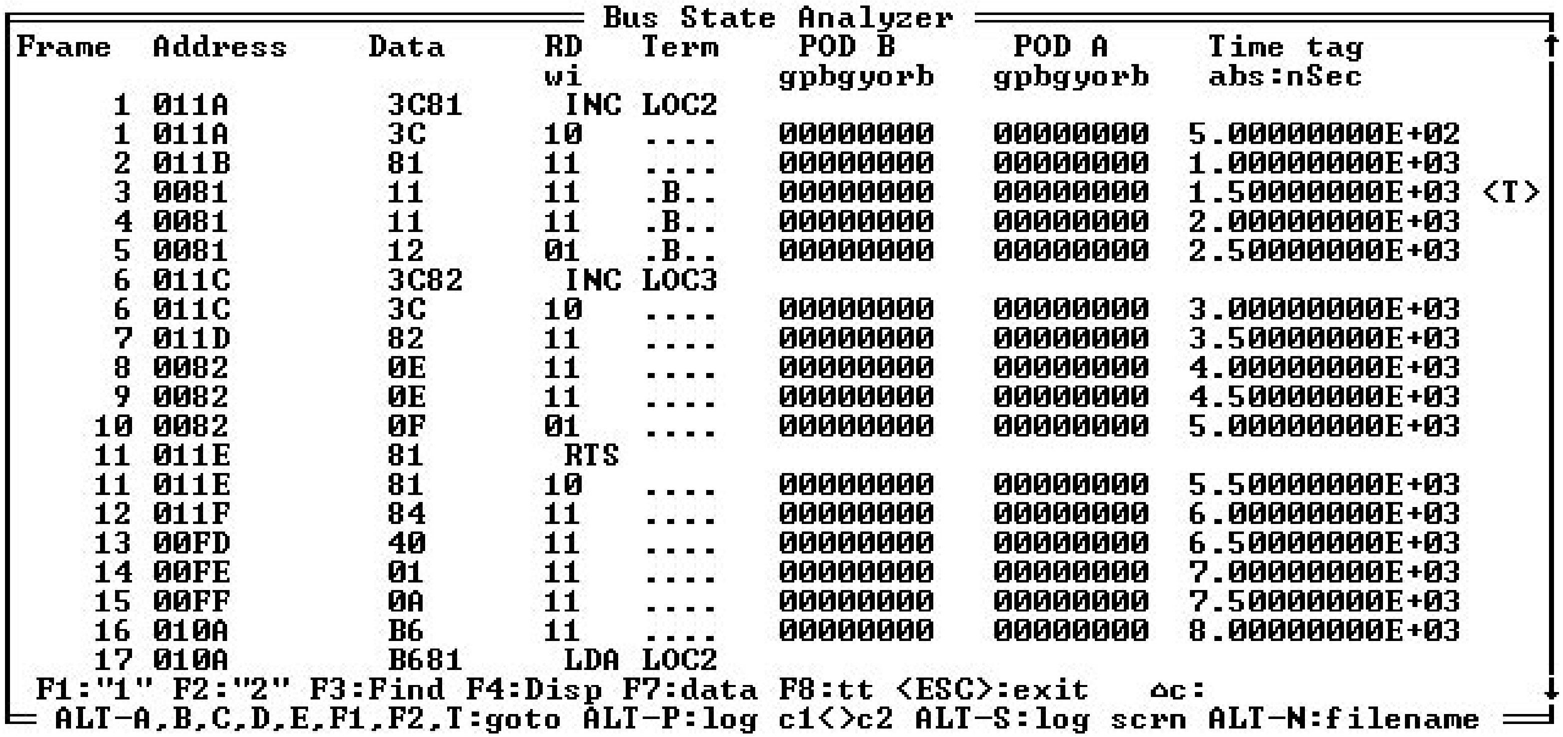
NOTE: *The instruction display includes only the frames that contain instruction fetch cycles; the instructions are displayed in disassembled form. A frame is one line of BSA data, valid at the end of a bus cycle. Frames are numbered sequentially from the first bus cycle.*

For a source code display (**Figure 15**), the source file must be in the directory with the object file. The source code display shows information similar to the instructions display, but it also displays the comments from the source code file.



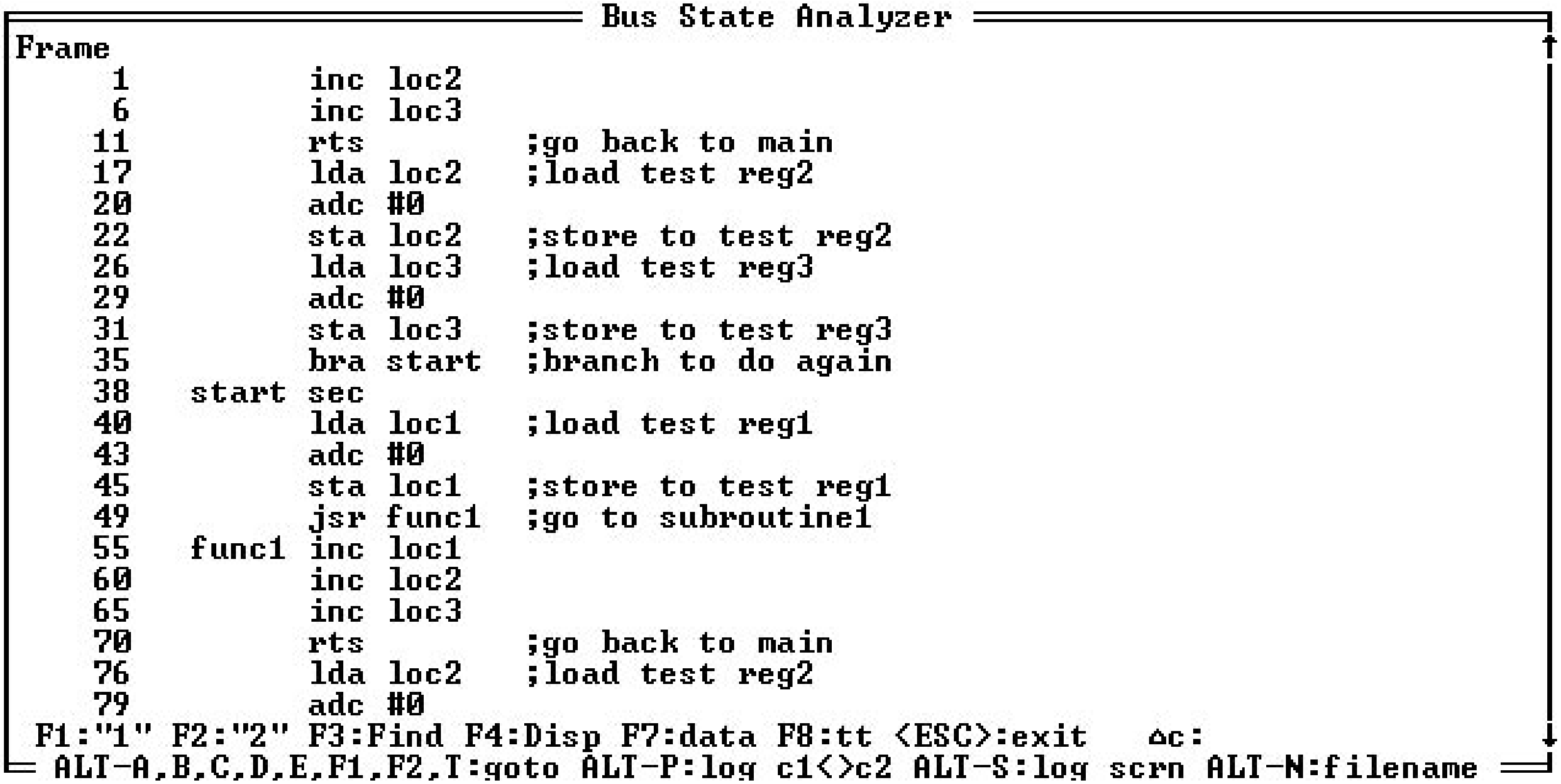
```
═════════════════════════ Bus State Analyzer ═════════════════════════
Frame
     1 011A         3C81      INC LOC2
     6 011C         3C82      INC LOC3
    11 011E         81        RTS
    17 010A         B681      LDA LOC2
    20 010C         A900      ADC #0
    22 010E         B781      STA LOC2
    26 0110         B682      LDA LOC3
    29 0112         A900      ADC #0
    31 0114         B782      STA LOC3
    35 0116         20E8      BRA START
    38 START        99        SEC
    40 0101         B680      LDA LOC1
    43 0103         A900      ADC #0
    45 0105         B780      STA LOC1
    49 0107         CD0118    JSR FUNC1
    55 FUNC1        3C80      INC LOC1
    60 011A         3C81      INC LOC2
    65 011C         3C82      INC LOC3
    70 011E         81        RTS
    76 010A         B681      LDA LOC2
    79 010C         A900      ADC #0
 F1:"1" F2:"2" F3:Find F4:Disp F7:data F8:tt <ESC>:exit   △c:
═ ALT-A,B,C,D,E,F1,F2,T:goto ALT-P:log c1<>c2 ALT-S:log scrn ALT-N:filename ═
```

Figure 13. Instructions Display



```
═══════════════════════════ Bus State Analyzer ═══════════════════════════
Frame  Address    Data     RD   Term    POD B      POD A      Time tag
                           wi           gpbgyorb   gpbgyorb   abs:nSec
     1 011A        3C81     INC LOC2
     1 011A        3C      10   ....    00000000   00000000   5.00000000E+02
     2 011B        81      11   ....    00000000   00000000   1.00000000E+03
     3 0081        11      11   .B..    00000000   00000000   1.50000000E+03 <T>
     4 0081        11      11   .B..    00000000   00000000   2.00000000E+03
     5 0081        12      01   .B..    00000000   00000000   2.50000000E+03
     6 011C        3C82     INC LOC3
     6 011C        3C      10   ....    00000000   00000000   3.00000000E+03
     7 011D        82      11   ....    00000000   00000000   3.50000000E+03
     8 0082        0E      11   ....    00000000   00000000   4.00000000E+03
     9 0082        0E      11   ....    00000000   00000000   4.50000000E+03
    10 0082        0F      01   ....    00000000   00000000   5.00000000E+03
    11 011E        81       RTS
    11 011E        81      10   ....    00000000   00000000   5.50000000E+03
    12 011F        84      11   ....    00000000   00000000   6.00000000E+03
    13 00FD        40      11   ....    00000000   00000000   6.50000000E+03
    14 00FE        01      11   ....    00000000   00000000   7.00000000E+03
    15 00FF        0A      11   ....    00000000   00000000   7.50000000E+03
    16 010A        B6      11   ....    00000000   00000000   8.00000000E+03
    17 010A        B681     LDA LOC2
 F1:"1" F2:"2" F3:Find F4:Disp F7:data F8:tt <ESC>:exit   ∆c:
═ ALT-A,B,C,D,E,F1,F2,T:goto ALT-P:log c1<>c2 ALT-S:log scrn ALT-N:filename ═
```

Figure 14. Mixed Raw Cycles and Instructions Delay



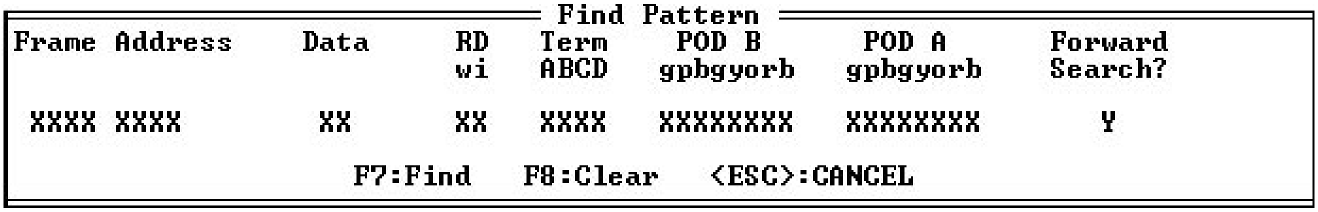
```
═══════════════════════ Bus State Analyzer ═══════════════════════
Frame
     1           inc loc2
     6           inc loc3
    11           rts          ;go back to main
    17           lda loc2     ;load test reg2
    20           adc #0
    22           sta loc2     ;store to test reg2
    26           lda loc3     ;load test reg3
    29           adc #0
    31           sta loc3     ;store to test reg3
    35           bra start    ;branch to do again
    38     start sec
    40           lda loc1     ;load test reg1
    43           adc #0
    45           sta loc1     ;store to test reg1
    49           jsr func1    ;go to subroutine1
    55     func1 inc loc1
    60           inc loc2
    65           inc loc3
    70           rts          ;go back to main
    76           lda loc2     ;load test reg2
    79           adc #0
 F1:"1" F2:"2" F3:Find F4:Disp F7:data F8:tt <ESC>:exit   ∆c:
═ ALT-A,B,C,D,E,F1,F2,T:goto ALT-P:log c1<>c2 ALT-S:log scrn ALT-N:filename ═
```

Figure 15. Source Code Display

Searching the Trace Buffer

The bus state analyzer includes a search utility, enabling you to search the trace buffer for a frame that contains a specific bit configuration. To start this utility, press the F3 key from the data screen. This brings up the find pattern window (**Figure 16**). Define a search pattern by filling in fields of this window; initially, all fields have X (don't care) values. Find searches from the point of the cursor to the end of the buffer. Use the arrow keys to move between fields. **Table 12** lists the key commands for the find pattern window.

The frame field is decimal. To scroll directly to a specific frame, enter the frame number in this field. If this is a don't care entry, put a string of four Xs in this field.



```
                         Find Pattern
Frame Address      Data      RD   Term    POD B       POD A       Forward
                             wi   ABCD   gpbgyorb    gpbgyorb     Search?

 XXXX XXXX          XX       XX   XXXX   XXXXXXXX    XXXXXXXX        Y

                  F7:Find   F8:Clear   <ESC>:CANCEL
```

Figure 16. Find Pattern Window

Table 12. Find Pattern Window Key Commands

Name	Key	Description
Move Left	←	Moves cursor one character to the left.
Move Right	→	Moves cursor one character to the right.
Next Field	Tab	Moves cursor to next field.
Preceding Field	Shift-Tab	Moves cursor to preceding field.
Find	F7	Scrolls to next frame that matches the selected pattern and returns to BSA data window.
Clear	F8	Clears the selected pattern.
Cancel	ESC	Returns to BSA data window without scrolling.

The address and data fields are hexadecimal. In these fields, you can specify a range by using the Xs for the less-significant digits. For example, 03XX in the Address field searches for addresses in the range of 0300--03FF. You can specify an X in any digit position to cause that digit to be ignored in the search.

The R/W and D/I fields and the two pod fields are binary. Enter 0 or 1 for each bit to be used in the search, and Xs for the bits to be ignored. Setting the R/W bit searches for read bus cycles; clearing this bit searches for write bus cycles. Setting the D/I bit searches for data bus cycles; clearing this bit searches for instruction fetch bus cycles.

You can search for bus cycles in which one or more terms are true by entering A, B, C, or D in the term field. When you have defined the search pattern completely, press F7 to start the search. The search begins at the current (highlighted) frame, and proceeds toward the highest-numbered frame in the trace buffer. (To clear the fields of the screen, press the F8 key.)

Using the Time-Tag Clock

There are four time-tag display modes: absolute, relative, cycles, and none. An absolute display mode shows the time reference from the first bus cycle. A relative display mode shows the time between bus cycles. A cycle display mode shows the cycle reference from the first bus cycle. None blanks the timetag display. You can cycle through these display modes using the F8 key while the BSA data screen is open.

The data screen displays the time tag as a number of seconds when you use the 1 MHz, 2 MHz, 4 MHz, 8 MHz, or 16 MHz clock. To time the execution of a portion of the code, use either the raw bus cycle mode or the mixed mode of the display, with the absolute time-tag format. Select the beginning cycle, and press the F1 key to mark it <1>. Select the ending cycle, and press the F2 key to mark it <2>. The software calculates the time between the two frames, then displays the difference (Δc) in the lower right corner of the data screen. (An **R**, by the Δc value, indicates a rollover of the time tag value between the occurrence of cycles <1> and <2>.)

For example, if the beginning time tag is 3.26778887E03 and the ending time tag is 3.2677928E03, the difference is 0.00000400 seconds, or 4 µs.

If the time tag is represented in clock periods, the procedure is the same, but the ∆c value is the number of time-tag clock cycles. Multiply the result by the time-tag clock period to obtain the elapsed time between the beginning and ending cycles.

For example, if the beginning time-tag value is 219, and the ending time-tag value is 234, the difference is 15 time-tag cycles. At a time-tag clock frequency of 4 MHz, the time-tag clock period is 0.25 µs, and the elapsed time is 3.75 µs. Had the same time-tag values been obtained with a time-tag clock frequency of 500 kHz (a clock period of 2 µs), the elapsed time would be 30 µs.

Command-Line Commands

Contents

Introduction 102

Command Syntax 102

Command Explanations 103
  A — Set Accumulator 106
  ACC — Set Accumulator. 107
  ARM — Arm Bus State Analyzer 108
  ASM — Assemble Instructions 109
  BAUD — Set Communications Baud Rate 110
  BAUDCHK — Baud Rate Check. 111
  BELL — Sound Bell 112
  BF — Block Fill 113
  BR — Set Instruction Breakpoint 114
  BSAMODE — BSA Display Mode. 115
  BSATT — BSA Timetag Mode 116
  C — Set/Clear C Bit 117
  CCR — Set Condition Code Register 118
  CHIPINFO — Chip Help Information. 119
  CLEARMAP — Remove Symbols. 120
  COLORS — Set Screen Colors 121
  DARM — Disarm Bus State Analyzer 122
  DASM — Disassemble Instructions 123
  ENDBSA — Go to Trace Buffer End. 124
  EVAL — Evaluate Argument. 125
  EXIT — Terminate Host Session 126
  G — Begin Program Execution. 127
  GETBSA — Upload Trace Buffer 128
  GO — Begin Program Execution 129
  GOTIL — Execute Program until Address 130
  H — Set/Clear H Bit 131

HELP — Display Help Information 132
HOMEBSA — Go to Trace Buffer Start 133
HREG — Set H Register. 134
HX — Set H:X Index Register 135
I — Set/Clear I Bit 136
INFO — Display Line Information 137
LF — Log File 138
LOAD — Load S19 File. 139
LOADMAP — Load Symbols 140
LOADMEM — Load Personality File. 141
LOADTRIGGERS — Load Bus State Analyzer Setup 142
MD — Memory Display 143
MM — Memory Modify 144
MM — Memory Modify 145
N — Set/Clear N Bit 146
NEXTA — Go to Next A Event 147
NEXTB — Go to Next B Event 148
NEXTC — Go to Next C Event 149
NEXTD — Go to Next D Event 150
NEXTE — Go to Next Event 151
NOBR — Clear Breakpoints 152
OSC — Select Emulator Clock Frequency 153
PC — Set Program Counter 154
QUIT — Terminate Host Session 155
REG — Display Registers. 156
REM — Add Comment to Script File. 157
RESET — Reset Emulation MCU 158
RESETGO — Reset and Restart MCU. 159
RESETIN — Reset Input Enable 160
RESETOUT — Reset Output Enable 161
RTMEM — Set Real-Time Memory Block. 162
RTVAR — Display Real-Time Variable. 163
RTVAR — Display Real-Time Variable. 164
SCREENBSA — Log Bus State Analyzer Screen 165
SCRIPT — Execute Script File 166
SETMEM — Customize Memory Map 167
SETMEM — Customize Memory Map 168
SHELL — Access DOS. 169

SHOWBSA — Display Trace Buffer 170
SHOWMEM — Display Memory Map 171
SHOWTRIGGER — Print Trigger 172
SNAPSHOT — Save Screen 173
SOURCE — Source Window Display 174
ST — Single Step (Trace) 175
STACK — Display Stack. 176
STEP — Single Step (Trace) 177
STEPFOR — Step Forever. 178
STEPTIL — Single Step to Address 179
STOP — Stop Program Execution 180
SXB — Set Multiplexer 181
SYSINFO — System Information 182
T — Single Step (Trace) 183
TIMETAG — Time Tag Clock Source. 184
V — Set/Clear V Bit. 186
VAR — Display Variable 187
VAR — Display Variable 188
VER — Display Version 189
VERSION — Display Version 190
VF — View File 191
WAIT — Pause between Commands 192
WAIT4RESET — Wait for Target Reset 193
WHEREIS — Display Symbol Value. 194
X — Set X Index Register 195
XREG — Set X Index Register 196
Z — Set/Clear Z Bit 197
ZOOM — Resize Source Window. 198

Introduction

Keyboard entry is the primary means of MMDS control. Individual commands are entered at the command-line prompt in the debug F10 window. The commands are used to initialize emulation memory, display and store data, debug user code, and control flow of code execution.

This section explains the rules for command syntax and arguments, then gives individual explanations for each command. Some of these commands can be executed via mouse control. For detail on using the mouse, see the section titled **Mouse Operation** on page 56.

Command Syntax

A command-line command is a line of ASCII text that is entered via the computer keyboard. Press <CR> to terminate each line, activating the command. The typical command syntax is:

```
> <command> [<argument>]. . .
```

Where:

>	The command prompt. The system displays this prompt in the debug F10 window when ready for another command.
<command>	A command name in upper- or lower-case letters. Refer to **Table 14** for command choices.
<argument>	One or more arguments. **Table 13** explains the many kinds of possible arguments.

In command syntax descriptions, brackets (**[]**) enclose optional items, a vertical line (**|**) means *or*, and an ellipsis (**. . .**) means the preceding item can be repeated. Except where otherwise noted, numerical values in examples are hexadecimal.

Table 13. Argument Types

Argument Type	Syntax Indicators	Explanation
Numeric	<n>, <count>, <data>	Hexadecimal values, unless otherwise noted. Leading zeros can be omitted. For decimal values, use the prefix **!** or the suffix **T**. For binary values, use the prefix **%** or the suffix **Q**. Example: 54 = !100 = 100T = %1100100 = 1100100Q
Address	<address>	Four or fewer hexadecimal digits (leading zeros can be omitted). If an address is decimal or binary, use a prefix or suffix per the explanation of numeric arguments.
Range	<range>	A range of addresses or numbers. Specify the low value, then the high value, separated by a space. Leading zeros can be omitted.
Symbol	<symbol>, <label>	Symbols of ASCII characters, usually symbols from source code
Filename	<filename>	The name of a file in DOS format; eight or fewer ASCII characters. An optional extension (three or fewer characters) can be included after a period. If the file is not in the current directory, precede the file name with a complete path.
Frequency	<rate>	Decimal values that specify clock speeds
Commands	<commands>	Items from the command set may be used as an argument for the HELP command.
Operator	<op>	+ (add); – (subtract); * (multiply); or / (divide)
Source	<source>	The source for possible clock sources used in the MMDS
Type	<type>	Specifies byte, word, or string data operations
Text	<text>	The text entered will be displayed when command is executed.
Termination	<terminator>	The terminator controls command flow in the memory modify command.

Command Explanations

Table 14 lists the command-line commands. Individual explanations of each of these commands follow the table. Note that the command parser of the MMDS host software is not case sensitive.

Table 14. Command Sumary

Mnemonic	Description
A	Set accumulator
ACC	Set accumulator
ARM	Arm Bus State Analyzer
ASM	Assemble Instruction
BAUD	Set Communications Baud Rate
BAUDCHK	Baud Rate Check
BELL	Sound Bell
BF	Block Fill
BR	Set Instruction Breakpoint
BSAMODE	Sets BSA Display Mode
BSATT	Sets BSA Timetag Mode
C	Set/Clear C Bit
CCR	Set Condition Code Register
CHIPINFO	Chip Help Information
CLEARMAP	Remove Symbols
COLORS	Set Screen Colors
DARM	Disarm Bus State Analyzer
DASM	Disassemble Instructions
ENDBSA	Go to Trace Buffer End
EVAL	Evaluate Argument
EXIT	Terminate Host Session
G	Begin Program Execution
GETBSA	Upload Trace Buffer
GO	Begin Program Execution
GOTIL	Execute Program until Address
H	Set/Clear H Bit
HELP	Display Help Information
HOMEBSA	Go to Trace Buffer Start
HREG*	Set H Register

Table 14. Command Sumary (Continued)

Mnemonic	Description
HX*	Set HX index register
I	Set/Clear I Bit
INFO	Display Line Information
LF	Log File
LOAD	Load S19 File
LOADMAP	Load Symbols
LOADMEM	Load Personality File
LOADTRIGGERS	Load Bus State Analyzer Setup
MD	Memory Display
MM	Memory Modify
N	Set/Clear N Bit
NEXTA	Go to Next A Event
NEXTB	Go to Next B Event
NEXTC	Go to Next C Event
NEXTD	Go to Next D Event
NEXTE	Go to Next Event
NOBR	Clear Breakpoints
OSC	Select Emulator Clock Frequency
PC	Set Program Counter
QUIT	Terminate Host Session
REG	Display Registers
REM	Add Comment to Script File
RESET	Reset Emulation MCU
RESETGO	Reset and Restart MCU
RESETIN	Reset Input Enable
RESETOUT	Reset Output Enable
RTMEM	Set Real-Time Memory Block
RTVAR	Display Real-Time Variable

* MMDS08 Only

Table 14. Command Sumary (Continued)

Mnemonic	Description
SCREENBSA	Log Bus State Analyzer Screen
SCRIPT	Execute Script File
SETMEM	Customize Memory Map
SHELL	Access DOS
SHOWBSA	Logs and Displays Bus State Analyzer
SHOWMEM	Display Memory Map
SHOWTRIGGER	Print Trigger
SNAPSHOT	Save Screen
SOURCE	Source Window Display
ST	Single Step (Trace)
STACK	Display Stack
STEP	Single Step (Trace)
STEPFOR	Step Forever
STEPTIL	Single Step to Address
STOP	Stop Program Execution
SXB	Set Multiplexer

Table 14. Command Sumary (Continued)

Mnemonic	Description
SYSINFO	System Information
T	Single Step (Trace)
TIMETAG	Timetag Clock Source
V*	Set/Clear V Bit
VAR	Display Variable
VER	Display Version
VERSION	Display Version
VF	View a File
WAIT	Pause between Commands
WAIT4RESET	Wait for Target Reset
WHEREIS	Display Symbol Value
X	Set X Index Register
XREG	Set X Index Register
Z	Set/Clear Z Bit
ZOOM	Resize Source Window

* MMDS08 Only

A Set Accumulator A

The A command sets the accumulator to a specified value. (The A and ACC commands are identical.)

Syntax:

A <*n*>

Where:

<*n*>	The value to be loaded into the accumulator.

Example:

>A 10	Set the accumulator to 10.

ACC Set Accumulator ACC

The ACC command sets the accumulator to a specified value. The ACC and A commands are identical.

Syntax:

ACC <*n*>

Where:

<*n*>	The value to be loaded into the accumulator

Example:

>ACC 20	Set the accumulator to 20.

ARM Arm Bus State Analyzer ARM

The ARM command arms the bus state analyzer. When armed, the analyzer records bus cycles while the emulator is executing user code. Arming the analyzer clears the current contents of the analyzer trace buffer. The word `armed` appears in the status area of the debug screen.

Syntax:

```
ARM
```

Example:

```
>ARM        Arm the bus state analyzer for user code bus
cycles.
```

ASM Assemble Instructions ASM

The ASM command assembles M68HC05 Family or M68HC08 Family instruction mnemonics and places the resulting machine code into memory at a specified address.

The command displays the specified address, its contents, and a prompt for an instruction. As each instruction is entered, the command assembles the instruction, stores and displays the resulting machine code, and displays the contents of the next memory location with a prompt for another instruction. To terminate the command, enter a period (.).

Syntax:

ASM [<*address*>]

Where:

<*address* >	An address at which the assembler places the first machine code generated. If <*address*> is not specified, the system checks the address used by the previous ASM command, then uses the following address for this ASM command.

Examples:

The first example shows the ASM command with an address argument:

```
>asm 100
01009D  NOP>CLRA
01004F  CLRA
01019D  NOP>.
```

The second example shows the ASM command with no argument:

```
>ASM
01019D  NOP>STA  0A
0101B70ASTA0A
01039D  NOP>.
```

NOTE: *Changes made to code via this command cannot be saved to an S-record file or to a source code file. This command should be used only to create and modify code to be run during the current debug session.*

BAUD Set Communications Baud Rate BAUD

The BAUD command changes the baud rate for communications between the system controller and the host computer. For best performance of the system, communications should be at the maximum available baud rate. Reduce this rate if the software displays communications error messages. Entering this command without a rate argument calls up the baud rate window. A baud rate can be selected via this window.

NOTE: *At power-up, MMDS software automatically sets the maximum baud rate for the system. If the baud rate is reduced but communication errors persist, turn off the disk cache (for instance, SMARTDRV.EXE).*

Syntax:

BAUD [<*rate*>]

Where:

<*rate*> One of these decimal baud-rate values:
2400
4800
9600
19,200
38,400
57,600

Example:

>BAUD 9600 Change the communications baud rate to 9600.

NOTE: *To specify a default baud rate of 9600, add the –B option when first running the MMDS command.*

BAUDCHK Baud Rate Check BAUDCHK

The BAUDCHK command sets the communication rate between the host software and the MMDS system. The command first checks communication at the maximum possible rate of 57600 baud and successively lowers the rate until communication with the MMDS is established.

Syntax:

```
BAUDCHK
```

Example:

```
>BAUDCHK

57600 baud communicates well
```

The command displays a message indicating the maximum available baud rate.

BELL Sound Bell BELL

The BELL command sounds the computer bell the specified *hexadecimal* number of times. The bell sounds once if an argument is not entered. To turn off the bell as it is sounding, press any key.

Syntax:

```
BELL [<n>]
```

Where:

<*n*>	The *hexadecimal* number of times to sound the bell

Examples:

>BELL	Sound the bell once.
>BELL C	Sound the bell 12 (decimal) times.
>BELL 12	Sound the bell 18 (decimal) times.

BF Block Fill BF

The BF command fills a block of memory with a specified byte or word. If the system cannot verify a write to one of the designated memory locations, it will stop command execution and report an error condition.

Syntax:

```
BF[.<type>] <range> <n>
```

Where:

<*type*>	Size of <*n*>:	
	B	<*n*> is an 8-bit value (the default)
	W	<*n*> is a 16-bit value
<*range*>	A block (range) of memory defined by beginning and ending addresses.	
<*n*>	A value to be stored in a byte or word of the specified block. If <*type*> is specified to be a byte value, then <*n*> is an 8-bit value. If <*type*> is specified to be a word value, then <*n*> is a 16-bit value and is stored in each word of the block.	

Examples:

>BF 200 20F FF	Store FF hexadecimal in bytes at addresses 200 to 20F.
>BF.W 100 11F 4143	Store 4143 in words at addresses 100 to 11F.

BR Set Instruction Breakpoint BR

The BR command sets an instruction breakpoint at a specified address or range of addresses. If a map file has been loaded, symbols (or labels) from the source code can be used as arguments. The maximum number of all instruction breakpoints is 64. For a list of all active breakpoints, enter this command without any parameter value.

A breakpoint occurs only on an address that contains an opcode (that is, an instruction fetch address). Although this command sets breakpoints at each address of a range, breakpoints occur only at the opcode addresses within the range. The system displays an error message if the address is within the range defined by a previous BR command or if the range of a new BR command overlaps the range of an existing BR command. An error message also appears if setting a 65th breakpoint is attempted.

Syntax:

BR [<*address*>|<*range*>|<*symbol*>]

Where:

<*address*>	The address for a breakpoint
<*range*>	The range of addresses for breakpoints; a beginning address and an ending address, separated by a space.
<*symbol*>	The label of an instruction in source code.

Examples:

>BR 100	Set a breakpoint at address 100.
>BR 130 13F	Set 16 breakpoints at addresses 130 through 13F.
>BR START	Set a breakpoint at address label START in code.
>BR 1000 103F	Set 64 breakpoints at addresses 1000 through 103F. Note that trying to set additional breakpoints, without clearing some of these breakpoints, would bring up the error message: Too many breakpoints

BSAMODE BSA Display Mode BSAMODE

The BSAMODE command sets the bus state analyzer display mode to raw, instruction, mixed, or source. This command is equivalent to the BSA data screen F4 key.

Syntax:

Display BSA <*n*>

Where:

<*n*>	One of the following decimal values:
1	Raw
2	Instruction
3	Mixed Raw and Instruction
4	Source

Example:

>C 0	Clear the C bit of the CCR.

BSATT BSA Timetag Mode BSATT

This command sets the bus state analyzer timetag mode to absolute, relative, none, or cycles. This command is equivalent to the BSA data screen F8 key.

Syntax:

```
BSA TT <n>
```

Where:

<*n*>	One of the following decimal values:
1	Absolute
2	Relative
3	None
4	Cycles

Example:

>C 0	Clear the C bit of the CCR.

C Set/Clear C Bit C

The C command sets the C bit of the condition code register (CCR) to the specified value.

NOTE: *The CCR bit designators are located at the lower right of the CPU window. The CCR pattern is V11HINZC (V is two's complement overflow for M68HC08 MCU only and is 1 for M68HC05 MCU, H is half-carry, I is IRQ interrupt mask, N is negative, Z is zero, and C is carry). A letter in these designators means that the corresponding bit of the CCR is set; a period means that the corresponding bit is clear.*

Syntax:

```
C 0|1
```

Where:

0	Clears the C bit
1	Sets the C bit

Example:

>C 0	Clear the C bit of the CCR.

CCR Set Condition Code Register CCR

The CCR command sets the condition code register (CCR) to the specified *hexadecimal* value.

NOTE: *The CCR bit designators are located at the lower right of the CPU window. The CCR pattern is V11HINZC (V is two's complement overflow for M68HC08 MCU only and is 1 for M68HC05 MCU, H is half-carry, I is IRQ interrupt mask, N is negative, Z is zero, and C is carry). A letter in these designators means that the corresponding bit of the CCR is set; a period means that the corresponding bit is clear.*

Syntax:

CCR <*n*>

Where:

<*n*>	The new *hexadecimal* value for the CCR

Example:

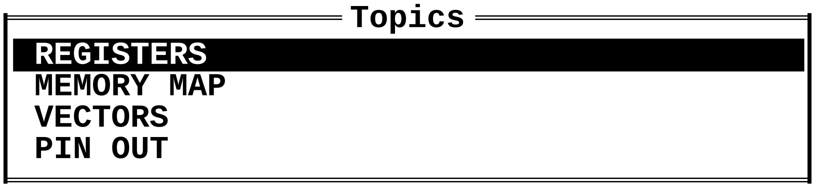
>CCR E4	Set the CCR to E4 (N bit set, others clear).

CHIPINFO Chip Help Information CHIPINFO

The CHIPINFO command accesses register, memory-map, vector, and pin-out information about the emulation MCU. Entering this command brings up the topics window as shown in Figure 17. Select a topic to bring up a subordinate window. (To select a topic, click on it; alternatively, highlight the topic, then press <CR>.)

The subordinate windows and their contents are:

- **REGISTERS** — Register addresses of the MCU being emulated. Selecting an address opens another subordinate window that displays each bit of the register.
- **MEMORY MAP** — The memory map for the MCU being emulated.
- **VECTORS** — The vectors for the MCU being emulated.
- **PIN OUT** — The pin outs for the MCU being emulated.



```
Topics
REGISTERS
MEMORY MAP
VECTORS
PIN OUT
```

Figure 17. Topics Window

Syntax:

CHIPINFO

Example:

>CHIPINFO Access emulation MCU information.

CLEARMAP Remove Symbols CLEARMAP

The CLEARMAP command removes the symbol definitions in the host computer. If a map file is loaded, symbols (or labels) from the source code can be used as arguments for many other commands.

Syntax:

```
CLEARMAP
```

Example:

```
>CLEARMAP    Clear symbols and their address definitions.
```

COLORS Set Screen Colors COLORS

The COLORS command sets the screen colors. Entering this command brings up the colors window. This window includes a list of screen elements and a matrix of foreground/background color combinations; each color combination has a 2-digit hexadecimal number.

A prompt asks for the color of the first screen element. To accept the current color, press <CR>. To change the color, enter the number of the choice, then press <CR>. A new prompt asks for the color of the next element. Select the color for each element in the same way. The command ends when a color for the last screen element is selected or when ESC is pressed.

In the color matrix, rows correspond to background colors and columns correspond to foreground colors. This means that color choices from the same row result in differently colored letters and numbers against the same background color. Making the background of highlights and help screens a different color sets these elements off from the main screen.

The software stores color selections in file COLORS.05 or COLORS.08; when MMDS is executed again, the software applies the newly selected colors.

NOTE: *Delete the COLORS.05 or COLORS.08 file from the MMDS subdirectory to return to the default colors.*

Syntax:

```
COLORS
```

DARM Disarm Bus State Analyzer DARM

The DARM command disarms the bus state analyzer. When disarmed, the analyzer does not record bus cycles. The word `Disarmed` appears in the status area of the debug screen. (If the bus state analyzer is already disarmed, this command does nothing.)

Syntax:

```
DARM
```

Example:

```
>DARM      Disarm the bus state analyzer.
```

DASM Disassemble Instructions DASM

The DASM command disassembles three or more machine instructions, displaying the addresses and the contents as disassembled instructions. Disassembly begins at the specified address. The valid address range is $0000 to $FFFF.

Syntax:

DASM *<address1> [<address2>]*

Where:

<*address1*>	The starting address for disassembly. *<Address1>* must be an instruction opcode. If only an *<address1>* value is entered, the system disassembles three instructions.
<*address2*>	The ending address for disassembly. If an *<address2>* value is entered, disassembly begins at *<address1>* and continues through *<address2>*. The screen scrolls upward as addresses and their contents are displayed, leaving the last instructions in the range displayed in the window.

Example: Disassemble and display three instructions beginning at address 100:

```
>DASM 100
0100        A6E8        LDA #0E8
0102        B702        STA 0002
0104        4F          CLRA
```

NOTE: *For a range larger than three commands, log files can be used to store DASM responses to a file.*

ENDBSA Go to Trace Buffer End ENDBSA

The ENDBSA command sets the trace buffer pointer to the last record of trace data. This command requires an open log file and loaded uploaded trace buffer data. Subsequent execution of the SCREENBSA command logs the last 20 records of analyzer data to the open log file.

Syntax:

```
ENDBSA
```

Example:

>LF <*filename*>	Open a log file.
>GETBSA	Upload captured analyzer.
>ENDBSA	Position trace buffer pointer to last record.
>SCREENBSA	Log first 20 records to open log file.

EVAL Evaluate Argument EVAL

The EVAL command performs mathematical operations on two numerical arguments. It displays the value of the result in hexadecimal, decimal, octal, and binary formats denoted by the suffixes H, T, O, and Q. (Note that octal numbers are not valid as operand values. Operand values are 15 bits or less.) If the value is equivalent to an ASCII character, the ASCII character is also displayed. This command supports addition (+), subtraction (–), multiplication (*) and division (/).

Syntax:

EVAL <*n1*> <*op*> <*n2*>

Where:

<*n1*>	A number to be evaluated or the first operand of a simple expression to be evaluated
<*op*>	The arithmetic operator (+, –, *, or /) of a simple expression to be evaluated
<*n2*>	The second operand of a simple expression to be evaluated

Example: Evaluate the sum of hexadecimal numbers 45 and 32 then display the result in four bases and as an ASCII character:

```
>EVAL 45 + 32

0077H 119T 000157O 0000000001110111Q “w”
```

NOTE: *The host will not inform of an operation that resulted in an overflow. Also, the result of a division operation will be the quotient.*

EXIT Terminate Host Session EXIT

The EXIT command terminates the host session and returns to DOS. The EXIT and QUIT commands are identical. Another way to end a host session is to enter the ALT-X keyboard combination.

Syntax:

```
EXIT
```

Example:

```
>EXIT        Return to DOS.
```

G Begin Program Execution G

The G command starts execution of code in the emulator at the current address or at a specified address. If one address is entered, it is the starting address. If two addresses are entered, execution begins at the first and stops at the second. If a map file has been loaded, symbols (or labels) from the source code can be used as arguments. The G and GO commands are identical.

If no address or only one address is specified, execution continues until a STOP command is entered, a breakpoint occurs, or an error occurs.

Syntax:

G *[<address1>|<symbol>] [<address2>|<symbol>]*

Where:

<*address1*>	Execution starting address. If an <*address1*> value is entered, the system loads the value into the program counter (PC), then starts execution at the address in the PC. If an <*address1*> value is not entered, execution begins at the address already in the PC.
<*address2*>	Execution stop address. The <*address2*> value must be an instruction fetch address; if it is not, code execution continues as if the command had no <*address2*> value.
<*symbol*>	The label of an instruction in source code.

NOTE: *Be careful about using the G, GO, or GOTIL commands if the code branches into internal RAM (for instance, option RAM) or EEPROM. An execution stop address is invalid for internal locations.*

Examples:

>G	Begin code execution at the current PC value.
>G 145	Begin code execution at address 145.
>G START	Begin code execution at label START in source code.
>G 200 271	Begin code execution at address 200. End code execution just before the instruction at address 271.

GETBSA Upload Trace Buffer GETBSA

The GETBSA command uploads the contents of the bus state analyzer trace buffer to the host computer. This is convenient when using a script file in conjunction with the bus state analyzer. Alternatively, the trace buffer data can be retrieved and displayed using the F4 function key.

Syntax:

```
GETBSA
```

Example:

```
>GETBSA     Upload trace buffer contents to the host
computer.
```

GO Begin Program Execution GO

The GO command starts execution of code in the emulator at the current address or at a specified address. If one address is entered, it is the starting address. If two addresses are entered, execution begins at the first and stops at the second. If a map file has been loaded, symbols (or labels) from the source code can be used as arguments. The GO and G commands are identical.

If no address or only one address is specified, execution continues until a STOP command is entered, a breakpoint occurs, or an error occurs.

Syntax:

GO *[<address1>|<symbol>] [<address2>|<symbol>]*

Where:

<*address1*>	Execution starting address. If an <*address1*> value is entered, the system loads the value into the program counter (PC), then starts execution at the address in the PC. If an <*address1*> value is not entered, execution begins at the address already in the PC.
<*address2*>	Execution stop address. The <*address2*> value must be an instruction fetch address; if it is not, code execution continues as if the command had no <*address2*> value.
<symbol>	The label of an instruction in source code.

NOTE: *Be careful about using the G, GO, or GOTIL commands if the code branches into internal RAM (for instance, option RAM) or EEPROM. An execution stop address is invalid for internal locations.*

Examples:

>GO	Begin code execution at the current PC value.
>GO 145	Begin code execution at address 145.
>GO START	Begin code execution at label START in source code.
>GO 200 271	Begin code execution at address 200. End code execution just before the instruction at address 271.

GOTIL Execute Program until Address GOTIL

The GOTIL command executes the program in the emulator, beginning at the address in the program counter (PC). Execution continues until the program counter contains the specified address during an opcode fetch cycle. If a map file has been loaded, symbols (or labels) from the source code can be used as arguments.

Syntax:

GOTIL <*address*> | <*symbol*>

Where:

<*address*>	Execution stop address. The <*address*> value must be an instruction fetch address; if it is not, code execution continues as if the command had no <*address*> value.
<*symbol*>	The label of an instruction in source code.

NOTE: *Be careful about using the G, GO, or GOTIL commands if the code branches into internal RAM (for instance, option RAM) or EEPROM. An execution stop address is invalid for internal locations.*

Example:

>GOTIL 0FF0	Execute the program in the emulator up to address 0FF0.

H Set/Clear H Bit H

The H command sets the H bit of the condition code register (CCR) to the specified value.

NOTE: *The CCR bit designators are at the lower right of the CPU window. The CCR pattern is V11HINZC (V is two's complement overflow for M68HC08 MCU only and is 1 for M68HC05 MCU, H is half-carry, I is IRQ interrupt mask, N is negative, Z is zero, and C is carry). A letter in these designators means that the corresponding bit of the CCR is set; a period means that the corresponding bit is clear.*

Syntax:

```
H 0|1
```

Where:

0	Clears the H bit
1	Sets the H bit

Example:

>H 1	Set the H bit of the CCR.

HELP Display Help Information HELP

The HELP command displays a list of help topics such as commands and function keys.

If commands are selected, the software displays an alphabetic index of the command set from which a command can be selected. To select help for a command, highlight the command using the page up/down keys and the arrow keys, then press <CR>. The command description screen shows the command name and its syntax and describes the command. When appropriate, the description includes examples and clarifying notes.

Selecting key commands brings up a list of screens in which function-key assignments differ. Select a screen to see its function-key assignments.

Use the arrow keys to scroll within the page; use the page up and page down keys to see other pages.

To exit the HELP data base and return to the previous screen, press the ESC key.

Syntax:

HELP [<*command*>]

Where:

<*command*>	Name of a command for which a description is needed

Examples:

>HELP	Display the HELP screens.
>HELP ASM	Display the description of the ASM command.

Related key command:

Pressing <F1> pulls up the main help window.

HOMEBSA Go to Trace Buffer Start HOMEBSA

The HOMEBSA command sets the trace buffer pointer to the first record of trace data. This command requires an open log file and loaded uploaded trace buffer data. Subsequent execution of the SCREENBSA command logs the first 20 records of analyzer data to the open log file.

Syntax:

```
HOMEBSA
```

Example:

>LF <*filename*>	Open a log file.
>GETBSA	Upload captured analyzer.
>HOMEBSA	Position trace buffer pointer to first record.
>SCREENBSA	Log first 20 records to open log file.

HREG Set H Register HREG

NOTE: *This command is for the MMDS08 only.*

The HREG command sets the upper byte of the index register to the specified value.

Syntax:

```
HREG <n>
```

Where:

<*n*>	The new value for the H register

Example:

```
>HREG F0    Set the H register value to F0.
```

HX Set H:X Index Register HX

NOTE: *This command is for the MMDS08 only.*

The HX command sets both bytes of the concatenated index register (H:X) to the specified value.

Syntax:

HX <*n*>

Where:

<*n*>	The new value for the H:X register

Example:

>HX 0400	Set the H:X index register to $400.

I Set/Clear I Bit I

The I command sets the I bit in the condition code register (CCR) to the specified value.

NOTE: *The CCR bit designators are at the lower right of the CPU window. The CCR pattern is V11HINZC (V is two's complement overflow for M68HC08 MCU only and is 1 for M68HC05 MCU, H is half-carry, I is IRQ interrupt mask, N is negative, Z is zero, and C is carry). A letter in these designators means that the corresponding bit of the CCR is set; a period means that the corresponding bit is clear.*

Syntax:

```
I 0|1
```

Where:

0	Clears the I bit
1	Sets the I bit

Example:

>I 1	Set the I bit of the CCR.

INFO Display Line Information INFO

The INFO command displays information about the highlighted line in the source window. This information includes the name of the file being displayed in the window, the line number, address, corresponding object code, and the disassembled instruction.

Syntax:

```
INFO
```

Example:

```
>INFO
Filename     :     05TESTCO.ASM     Line number  :  !117
Address      :     $0100
Disassembly  :     0100        99   SEC
```

If a map file is loaded and the highlighted instruction has a label, the label will be displayed in the disassembly line in place of the address.

Example:

```
>INFO
Filename     :     05TESTCO.ASM     Line number :   !117
Address      :     $0100
Disassembly  :     START          99   SEC
```

LF — Log File — LF

The LF command starts or stops logging of commands and responses to an external file. If logging is not enabled, enter this command to start logging. While logging remains in effect, any line that is appended to the command log window also is written to the log file. Logging continues until another LF command is entered; this second command disables logging and closes the log file.

If the specified file does not already exist, this command creates the file. If the specified file does exist already, the command prompts for overwrite or append:

```
File exists, Rewrite or Append? [R]:
```

If <CR> (accept the default) is pressed, or R and <CR>, the log entries overwrite the data in the existing file. If A and <CR> are pressed, the system appends log entries to the file.

Syntax:

```
LF <filename>
```

Where:

<*filename*>	The DOS filename of the log file; the command interpreter does not assume a filename extension.

Examples:

>LF log file	Start logging. Write to file log file (in the current directory) all lines added to the command log window.
>LF	If logging is enabled: Disable logging and close the log file.

LOAD Load S19 File LOAD

The LOAD command loads a file in .S19 format (and any map file with the same name) into the emulator. If no argument is supplied, the command pops up a file select window.

Syntax:

LOAD [<*filename*>]

Where:

<*filename*>	The name of the S-record file to be loaded. An extension of .S19 is the default and can be omitted. The extension must be specified for files with other extensions. A path name followed by the asterisk (*) wildcard character can be entered. In that case, the command displays a window that lists the files in the specified directory that have the .S19 extension.

Examples:

>LOAD PROG1.S19	Load file PROG1.S19 and its map file into the emulator at the load addresses in the file.
>LOAD PROG2	Load file PROG2.S19 and its map file into the emulator at the load addresses in the file.
>LOAD A:*	Display the names of the .S19 files on the diskette in drive A: for user selection of a file.

LOADMAP Load Symbols LOADMAP

The LOADMAP command loads a map file that contains symbol information from source code. If no argument is supplied, the command pops up a file select window.

Syntax:

```
LOADMAP [<filename>]
```

Where:

<*filename*>	The name of the map file to be loaded. An extension of .MAP is the default and can be omitted. The extension must be specified for files with other extensions. A pathname followed by the asterisk (*) wildcard character can be entered. In that case, the command displays a window that lists the files in the specified directory that have the .MAP extension.

Examples:

>LOADMAP PROG1.MAP	Load map file PROG1.MAP into the host computer.
>LOADMAP PROG2	Load map file PROG2.MAP into the host computer.
>LOADMAP A:*	Display the names of the .MAP files on the diskette in drive A:, for user selection of a file.

LOADMEM Load Personality File LOADMEM

Personality files are used to customize the emulation memory map for a specific microcontroller device. A personality file to be loaded could have been shipped with an emulation module (EM) or could have been created by pressing the F6 key in the SETMEM window.

The LOADMEM command loads the memory map for the emulator with the map information from the specified file.

Syntax:

LOADMEM [<*filename*>]

Where:

<*filename*>	The name of the memory-mapping file to be loaded. An extension of .MEM is the default and can be omitted. If a pathname followed by the asterisk (*) wildcard character is entered, the command displays a window that lists the files in the specified directory that have the .MEM extension. If a .MEM file is selected that is not appropriate for the current EM installed, an error will be generated.

Examples:

>LOADMEM 000P4V01.MEM	Make 000P4V01.MEM the current memory-mapping file.
>LOADMEM 003FEV01.MEM	Make 003FEV01.MEM the current memory-mapping file.
>LOADMEM A:*	Display the names of the .MEM files on the diskette in drive A:, for user selection of a file.

LOADTRIGGERS Load Bus State Analyzer Setup LOADTRIGGERS

The LOADTRIGGERS command loads the bus state analyzer setup information from the specified file. To write such a file, use the bus state analyzer setup screen to define the triggers, then press the F6 key.

Syntax:

LOADTRIGGERS [<*filename*>]

where:

<*filename*>	The name of the setup file to be loaded. An extension of .SET is the default and can be omitted. The extension must be specified for files with other extensions. You can enter a pathname followed by the asterisk (*) wildcard character. In that case, the command displays a window that lists the files in the specified directory that have the .SET extension.

Examples:

>LOADTRIGGERS BSA.SET	Make BSA.SET the current BSA setup file.
>LOADTRIGGERS BSA8	Make BSA8.SET the current BSA setup file.
>LOADTRIGGERS A:*	Display the names of the .SET files on the diskette in drive A:, for user selection of a file.

MD Memory Display MD

The MD command displays (in the memory F3 window) the contents of 32 emulation memory locations. The specified address is the first of the 32 locations. If a log file is open, this command also writes the first 16 values to the log file.

Syntax:

MD <*address*>

Where:

<*address*>	The starting memory address for display in the memory window

Example:

>MD 1000	Display the contents of 32 bytes of memory beginning at address 1000.

MM Memory Modify MM

The MM command lets the user interactively examine and modify contents of memory locations. Writes to memory are verified and a "write did not verify" is displayed if the write could not be verified. Note that this message may be acceptable in some situations, such as writing to registers that have write-only bits.

If any data arguments are entered with this command, the system stores the values, beginning at the specified address. This command does not alter the contents of CPU registers such as the program counter (PC).

Syntax:

MM <*address*> [<*data*>] [<*data*>]

Where:

<*address*>	The address of a memory location to be modified
<*data*>	The value(s) to be stored at the <*address*> location. If more than one data byte is supplied, the two data bytes are stored in consecutive memory locations starting at the address argument.

If <*data*> is not supplied, the command flow will display the current contents of the specified address and an entry prompt for data. The syntax for entry at the MM data prompt is:

[<*data*>][<*terminator*>]

MM Memory Modify MM

Where:

<*data*>	The value to be stored at the <*address*> argument.
<*terminator*>	The command terminator character controls the next step in the command flow. The four choices are:

a. If no <*terminator*> is supplied, address flow will sequence forward.

b. If the equal (=) character is entered, flow will stay at the current address.

c. If the carat (^) character is entered, flow will sequence backward to the previous address.

d. If the period (.) character is entered, flow will terminate and return to the command line prompt.

Examples:

The first example does not have a <*data*> value in the command line, permitting entry of new values for consecutive addresses. Entering a period instead of a new value stops the command:

```
>MM 1000

1000 = 0F >05

1001 = 10 >.
```

The second example includes a <*data*> value, so the command modifies only one memory location:

```
>MM 100 00
```

N Set/Clear N Bit N

The N command sets the N bit of the condition code register (CCR) to the specified value.

NOTE: *The CCR bit designators are at the lower right of the CPU window. The CCR pattern is V11HINZC (V is two's complement overflow for M68HC08 MCU only and is 1 for M68HC05 MCU, H is half-carry, I is IRQ interrupt mask, N is negative, Z is zero, and C is carry). A letter in these designators means that the corresponding bit of the CCR is set; a period means that the corresponding bit is clear.*

Syntax:

```
N 0|1
```

Where:

0	Clears the N bit
1	Sets the N bit

Example:

>N 1	Set the N bit of the CCR.

NEXTA Go to Next A Event NEXTA

The NEXTA command displays the next occurrence of an A event in the debug window. If a log file is open, this command also writes that frame to the log file. If the NEXTA event cannot be found, the last captured event in the trace buffer is displayed.

Syntax:

```
NEXTA
```

Example:

>NEXTA	Scroll the bus state analyzer display to the next A event.

Related Key Command:

Alt-A while in the bus state analyzer data window.

NEXTB Go to Next B Event NEXTB

The NEXTB command displays the next occurrence of a B event in the debug window. If a log file is open, this command also writes that frame to the log file. If the NEXTB event cannot be found, the last captured event in the trace buffer is displayed.

Syntax:

```
NEXTB
```

Example:

>NEXTB	Scroll the bus state analyzer display to the next B event.

Related Key Command:

Alt-B while in the bus state analyzer data window.

NEXTC Go to Next C Event NEXTC

The NEXTC command displays the next occurrence of a C event in the debug window. If a log file is open, this command also writes that frame to the log file. If the NEXTC event cannot be found, the last captured event in the trace buffer is displayed.

Syntax:

```
NEXTC
```

Example:

>NEXTC	Scroll the bus state analyzer display to the next C event.

Related Key Command:

Alt-C while in the bus state analyzer data window.

NEXTD Go to Next D Event NEXTD

The NEXTD command displays the next occurrence of a D event in the debug window. If a log file is open, this command also writes that frame to the log file. If the NEXTD event cannot be found, the last captured event in the trace buffer is displayed.

Syntax:

```
NEXTD
```

Example:

>NEXTD	Scroll the bus state analyzer display to the next D event.

Related Key Command:

Alt-D while in the bus state analyzer data window.

NEXTE Go to Next Event NEXTE

The NEXTE command positions the bus state analyzer display at the next occurrence of any event. The data record is displayed in the debug window. If a log file is open, this command also writes that frame to the log file. If another event is not found, the last captured event in the trace buffer is displayed.

Syntax:

```
NEXTE
```

Example:

```
>NEXTE    Scroll the bus state analyzer display to the next
event.
```

Related Key Command:

Alt-E while in the bus state analyzer data window.

NOBR Clear Breakpoints NOBR

The NOBR command clears one instruction breakpoint, all instruction breakpoints, or all instruction breakpoints within an address range. If a map file has been loaded, symbols (or labels) from the source code can be used as arguments. If this command has only one argument, it clears the breakpoint at that address. If this command has no argument, it clears all current breakpoints. If this command has two address values, it clears all instruction breakpoints in the range the addresses define.

Syntax:

NOBR [<*address*>**|**<*range*>**|**<*symbol*>]

Where:

<*address*>	The address of a single breakpoint to be removed
<*range*>	The range of addresses from which all breakpoints should be removed
<*symbol*>	The label of an instruction in source code.

Examples:

>NOBR	Clear all current instruction breakpoints.
>NOBR 120	Clear the instruction breakpoint at address 120.
>NOBR 120 140	Clear all instruction breakpoints in the address range 120 to 140.
>NOBR START	Clear a previously set breakpoint at address label START in source code.

OSC Select Emulator Clock Frequency OSC

The MMDS platform board can supply an oscillator clock source for the MCU’s OSC1 input.

For the MMDS05, five clock frequencies are available. The four internally generated clock frequencies are available: 8 MHz, 4 MHz, 2 MHz, and 1 MHz and an external clock source. Entering emulator clock (OSC) command without the designated frequency brings up the temporary MMDS emulator clock frequency window near the center of the debug screen. Use the up/down arrow keys to select the emulator MCU's clock frequency and press <CR> to complete the selection. The default emulator clock rate is 2 MHz.

For the MMDS08, six clock frequencies are available. The five internally generated clock frequencies are available: 16 MHz, 8 MHz, 4 MHz, 2 MHz, and 1 MHz and an external clock source. The default emulator clock rate is 4 MHz.

Entering this command without the <*rate*> argument brings up the EM oscillator window. An oscillator frequency can be selected via this window.

NOTE: *Many EMs require a specific jumper configuration in order to use this clock source. Refer to the EM user's manual for EM clock source information.*

Syntax:

OSC [<*rate*> | <*source*>]

Where:

<*rate*>	8, 4, 2, or 1
<*source*>	External

Examples:

>OSC 4	Use the 4-MHz internal emulator clock.
>OSC EX	Use the external emulator clock.
>OSC	Bring up the emulator clock window. The current oscillator setting will be highlighted.

PC Set Program Counter PC

The PC command sets the program counter (PC) to the specified address.

Syntax:

PC <*address*>

Where:

<*address*>	The new address value for the PC

Example:

PC 0500	Set the PC to 0500.

QUIT Terminate Host Session QUIT

The QUIT command terminates the host session and returns to DOS. The QUIT and EXIT commands are identical. Another way to end a host session is to enter the ALT-X keyboard combination.

Syntax:

```
QUIT
```

Example:

```
>QUIT       Return to DOS.
```

REG Display Registers REG

The REG command displays the contents of the CPU registers in the debug F10 window. The command also will display the instruction pointed to by the current program counter value.

Syntax:

```
REG
```

Example:

```
>REG
PC:1196 A:00 X:90 SP:FF CCR:FA [BRCLR 1,0003,119C]
```

REM Add Comment to Script File REM

The REM command adds a display comment to a script file. When the script file is executed, the system displays this comment.

Syntax:

```
REM <text>
```

Where:

<text>	The display comment.

Example:

>REM Program executing	Display program executing only during script file execution.

RESET Reset Emulation MCU RESET

The RESET command resets the emulation MCU and sets the program counter to the contents of the reset vector. This command does *not* start execution of user code. To reset and execute user code, use the RESETGO or WAIT4RESET command.

Syntax:

```
RESET
```

Example:

```
>RESET    Reset the MCU.
```

RESETGO Reset and Restart MCU RESETGO

The RESETGO command resets the emulation MCU, sets the program counter (PC) to the contents of the reset vector, then starts execution from that address.

Syntax:

```
RESETGO
```

Example:

```
>RESETGO      Reset the MCU and go.
```

RESETIN Reset Input Enable RESETIN

The RESETIN command makes it possible for the target system to reset the emulating MCU.

Entering this command toggles the MMDS state with regard to a reset signal from the target system. If this state is enabled, a reset signal from the target system resets the emulating MCU. If this state is disabled, a reset signal from the target system cannot reset the emulating MCU. The word Resetin appears in the debug screen status area to show the enabled state.

The state must be enabled for proper operation of the WAIT4RESET command.

NOTE: *Certain EMs include a hardware jumper that governs target resets. Such a jumper must be configured correctly to use the RESETIN command. Consult the EM user's manual for additional information.*

Syntax:

```
RESETIN
```

Example:

```
>RESETIN        Toggle the MMDS RESETIN state.
```

RESETOUT **Reset Output Enable** RESETOUT

The RESETOUT command makes it possible for the MMDS RESET command to reset the target system.

Entering this command toggles the MMDS state with regard to resetting the target system. If this state is enabled, entering the RESET command resets both the emulating MCU and the target system. The word Resetout appears in the debug screen status area to show the enabled state. If this state is disabled, entering the RESET command resets only the emulating MCU.

The RESETOUT command also pertains to resets done via the RESETGO command.

NOTE: *Certain EMs include a hardware jumper that governs target resets. Such a jumper must be configured correctly to use the RESETOUT command. Consult your EM user's manual for additional information.*

Syntax:

```
RESETOUT
```

Example:

```
>RESETOUT     Toggle the MMDS RESETOUT state.
```

RTMEM Set Real-Time Memory Block RTMEM

The RTMEM command enables real-time-memory, starting at a specified address. The real-time memory consists of 32 bytes of dual-ported memory that is assigned to any valid memory address by this command. While the emulator is running, the system displays enabled real-time memory in the real-time memory window (this window replaces the memory F3 window). The display updates as the memory contents change. Entering the RTMEM command without an argument disables real-time memory, restoring previous memory map attributes.

Real-time memory consists of memory enabled by the RTMEM command, plus real-time variables created via the RTVAR command. All this real-time memory must fit within a 1-Kbyte block. If an RTMEM command would result in real-time memory that would not fit within the 1-Kbyte block (due to established real-time variables), the system will not accept the RTMEM command.

If any of the real-time memory overlays MCU I/O or EEPROM addresses, it is available only for monitoring emulation MCU writes. (You should not try to modify such locations.) You can monitor and modify real-time memory locations that do not overlay MCU I/O or EEPROM addresses.

Syntax:

RTMEM [<*address*> **|** <*symbol*>]]

where:

<*address*>	The beginning address of the real-time-memory.
<*symbol*>	A symbol loaded from a .MAP file

Example:

>RTMEM 0200	Set the address of the real-time-memory to 0200 through 021F.

RTVAR Display Real-Time Variable RTVAR

The RTVAR command displays the specified address and its contents in the variables F8 window as a real-time variable. If a MAP file is loaded, symbols from the source code can be used as arguments.

As many as 32 variables can be declared in the variables F8 window. The window shows 11 at a time. Using the RTVAR command establishes a real-time variable. The variable value will periodically be updated in the variables F8 window during emulation. You can also enter a new value for a real-time variable during emulation.

The <*type*> argument enables display of variables in byte, word, or string format. A byte display is hexadecimal and binary, a word display is hexadecimal and decimal, and a string display is ASCII.

For an ASCII string, the optional <*n*> argument specifies the number of characters; the default is the maximum 11 characters. Control and other non-printing characters appear as periods (.).

Syntax:

RTVAR[.<*type*>] <*address*>|<*symbol*> [<*n*>]

Where:

<*type*>	The variable type to display: B (byte, the default), W (word), or S (string)
<*address*>	The address of the memory variable
<*n*>	The number of characters to be displayed. Used only with the string type. If <*n*> is omitted, 11 ASCII characters will be visible in the window, beginning at the <*address*> argument location.
<*symbol*>	A symbol loaded from a .MAP file

RTVAR Display Real-Time Variable RTVAR

Examples:

>RTVAR 100	Display (in hexadecimal and binary) the byte at address 100
>RTVAR.B 110	Display (in hexadecimal and binary) the byte at address 110
>RTVAR.W 102	Display (in hexadecimal and decimal) the word at address 102
>RTVAR.S 200 5	Display the 5-character ASCII string starting at address 200

(The RTVAR command also establishes variables for display in the variables F8 window, but such variables are not real-time. The 32-variable maximum applies to variables established by both the RTVAR and VAR commands.)

Real-time memory consists of memory enabled by the RTMEM command, plus real-time variables created via the RTVAR command. All this real-time memory must fit within a 1-Kbyte block. If an RTVAR command would create a real-time variable that would not fit within the 1-Kbyte block (due to established real-time variables or memory enabled by the RTMEM command), the system will not accept the RTVAR command.

If any of the real-time memory overlays MCU I/O or EEPROM addresses, it is available only for monitoring emulation MCU writes. (You should not try to modify such locations.) You can monitor and modify real-time memory locations that do not overlay MCU I/O or EEPROM addresses.

SCREENBSA Log Bus State Analyzer Screen SCREENBSA

The SCREENBSA command copies the current bus state analyzer display to an open log file.

Syntax:

```
SCREENBSA
```

Example:

>SCREENBSA	Copy the bus state analyzer display to the log file.

SCRIPT Execute Script File SCRIPT

The SCRIPT command executes a script file, which contains a sequence of emulator commands. Executing the script file has the same effect as executing the individual commands one after another. This makes a script file convenient for any sequence of commands that is needed often, such as unit test or initialization command sequences.

The REM and WAIT commands are useful primarily within script files. The REM command allows a comment to be displayed while the script file executes. The WAIT command establishes a delay between the execution of commands of the script file.

NOTE: *A script file can contain the SCRIPT command. Script files can be nested as many as 15 levels deep.*

If the script file has the filename STARTUP.05, the script file will be executed each time the MMDS is started.

Syntax:

SCRIPT <*filename*>

Where:

<*filename*>	The name of the script file to be executed. An extension of .SCR is the default and can be omitted. The extension must be specified for files with other extensions. A path name followed by the asterisk (*) wildcard character can be entered. In that case, the command displays a window that lists the script files in the specified directory that have the .SCR extension. A file can be selected from the list.

Examples:

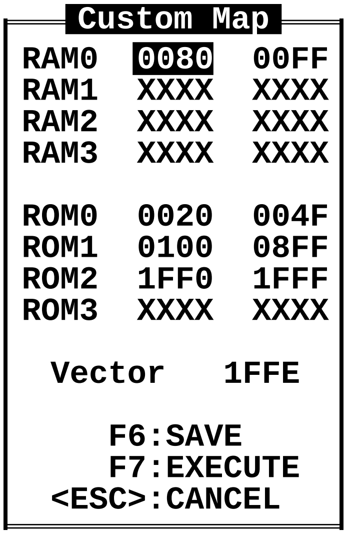
>SCRIPT INIT.SCR	Execute commands in file INIT.SCR.
>SCRIPT *	Display all .SCR files, then execute the selected file.
>SCRIPT A:*	Display all .SCR files in drive A, then execute the selected file.
>SCRIPT B:*.xyz extension	Display all drive B files that have the .xyz then execute the selected file.

SETMEM Customize Memory Map SETMEM

The SETMEM command allows customizing of the memory map. Entering this command brings up the custom map window as shown in Figure 18. The current RAM and ROM configuration will be shown in the window. To modify the map, enter the desired address ranges.

To write the modified map to a file for future use, press Save (F6), then enter the filename at the prompt. The system saves the new .MEM file under the specified name. If a file by the specified name already exists, a notice is made with the option to overwrite. The emulator can load this file using the LOADMEM instruction at a future time.

Pressing Execute (F7) will use the newly defined memory map for the current debug session only.



```
Custom Map
RAM0  0080  00FF
RAM1  XXXX  XXXX
RAM2  XXXX  XXXX
RAM3  XXXX  XXXX

ROM0  0020  004F
ROM1  0100  08FF
ROM2  1FF0  1FFF
ROM3  XXXX  XXXX

 Vector   1FFE

    F6:SAVE
    F7:EXECUTE
 <ESC>:CANCEL
```

Figure 18. Custom Map Window

SETMEM Customize Memory Map SETMEM

The SETMEM command allows mapping over undefined memory or memory defined as RAM or ROM. Do not map over such internal resources as option RAM, I/O, or EEPROM. The SETMEM command automatically maps around internal resources.

NOTE: *The SETMEM command can be used to expand the normal RAM and ROM ranges temporarily during debugging. Be sure to restore the original size and configuration of the MCU memory before final debugging. Otherwise, the code could fail to fit or run in an MCU's memory space.*

The SETMEM and SHOWMEM commands only show MMDS resources. That is memory that is resident on the control board during emulation. Use the CHIPINFO command memory map feature to view internal I/O, option RAM, and EEPROM locations.

Syntax:

```
SETMEM
```

SHELL Access DOS SHELL

The SHELL command allows access to DOS in the host computer. To return to MMDS from DOS, enter EXIT at the DOS prompt.

MMDS continues to run during the shell to DOS. This could mean that the memory for other software is insufficient.

Syntax:

```
SHELL
```

Example:

>SHELL	Access the DOS shell. To return to the emulator session, type EXIT at the DOS prompt.

SHOWBSA Display Trace Buffer SHOWBSA

This command copies a specified range of the current BSA display to the debug window and an open log file.

Syntax:

SHOWBSA <*range*>

Where:

<*range*>	The range of BSA frames a beginning frame and an ending frame separated by a space.

SHOWMEM Display Memory Map SHOWMEM

The SHOWMEM command displays only the MMDS resources. That is memory that is resident on the control board during emulation. Use the CHIPINFO command memory map feature to view internal I/O, option RAM, and EEPROM locations.

Syntax:

```
SHOWMEM
```

Example:

```
>SHOWMEM     Display current memory map blocks.
```

SHOWTRIGGER Print Trigger SHOWTRIGGER

The SHOWTRIGGER command displays the trigger frame of the bus state analyzer buffer. If a log file is open, this command also writes the trigger frame to the log file.

Syntax:

```
SHOWTRIGGER
```

Example:

`>SHOWTRIGGER`	Display the bus state analyzer trigger frame.

SNAPSHOT Save Screen SNAPSHOT

The SNAPSHOT command saves a copy of the main screen to the open log file. A log file must be open or this command has no effect.

NOTE: *The main screen includes certain extended ASCII characters. When subsequently viewing a screen snapshot, a standard ASCII editor will display a few characters that do not match the original screen.*

Syntax:

```
SNAPSHOT
```

Example:

```
>SNAPSHOT    Capture screen, save to a log file.
```

SOURCE Source Window Display SOURCE

The SOURCE command toggles between source code and disassembled code in the source/code F2 window, located at the upper right of the debug screen. On entering MMDS software, the window defaults to disassembled code, the window title is CODE F2, and window contents are a disassembled representation of MCU memory. In this object code display, the disassembled instructions change when corresponding bytes of memory change. To scroll through this window, press the F2 key (to select the window), then use the arrow keys.

The contents of the source/code F2 window change to source code when the SOURCE command is executed if:

1. A map file has been loaded (a map file is loaded with the S-record LOAD command) and
2. The program counter (PC) points to a memory area covered by the map file.

Once a .MAP file has been loaded and the PC points to an area of user code, the SOURCE command can be used to toggle between source code and object code. If a mouse is installed, the symbols that appear at the bottom of the window can be selected. Use the mouse or arrow keys to scroll through the information in the window. Note that the F2 key does not pertain to this window if it shows source code. **Table 6** lists the key commands available in this window when a source code is displayed.

NOTE: *When memory data that was generated from a source file is altered, the modified code appears in the code window but not the source file window. Use the CLEARMAP command to clear the source file from the host system.*

Syntax:

```
SOURCE
```

Example:

```
>SOURCE       Toggle the display in the source/code F2
window.
```

ST Single Step (Trace) ST

The ST command executes a specified *hexadecimal* number of instructions, beginning at the current program counter (PC) address value. If a number is not specified, this command executes one instruction. The ST, STEP, and T commands are identical.

Syntax:

```
ST [<count>]
```

Where:

<*count*>	The *hexadecimal* number of instructions to be executed. Hexadecimal $7FF is the maximum value.

NOTE: *Do not use any step command (ST, STEP, STEPFOR, STEPTIL, or T) if the PC points to internal RAM or EEPROM or if the code branches into internal RAM or EEPROM.*

The step commands are not real-time. They execute one instruction at a time, then return control to the monitor. Do not rely on timer values.

Examples:

>ST	Execute the instruction at the current PC address value.
>ST 2	Execute two instructions, starting at the current PC address value.

STACK Display Stack STACK

The temporary stack window appears near the center of the debug screen when the STACK command is entered. As Figure 19 shows, this window displays the contents of the SP register at the top of the window. The 12 bytes at the top of the stack are displayed to the left. The interpretation of the stack shown to the right in the window is valid only if the last push to the stack was caused by an interrupt. Press the ESC key to remove the stack window and return to the debug window.

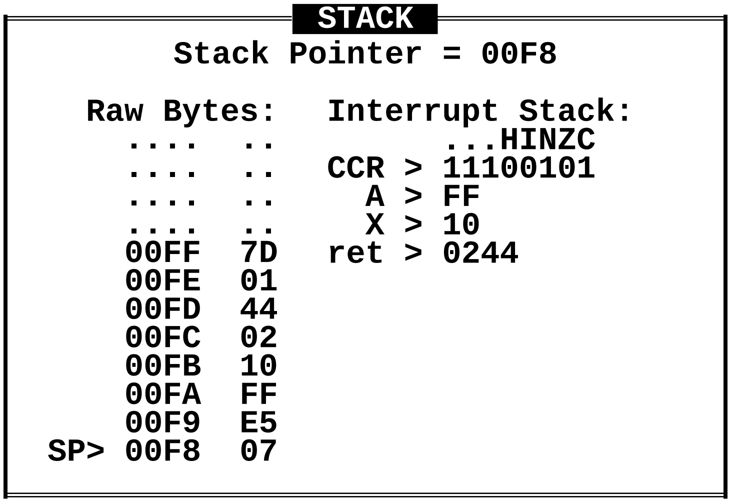
Syntax:

```
STACK
```

Example:

```
>STACK    Display the current configuration of the stack.
```

NOTE: *The interrupt stack data to the right side of the window is an interpretation of the top five bytes on the stack. If the last push to the stack was due to a BSR or JSR instruction, five bytes were not pushed to the stack and the interrupt stack information is invalid.*



```
                     STACK
        Stack Pointer = 00F8

   Raw Bytes:   Interrupt Stack:
     ....  ..         ...HINZC
     ....  ..   CCR > 11100101
     ....  ..     A > FF
     ....  ..     X > 10
     00FF  7D   ret > 0244
     00FE  01
     00FD  44
     00FC  02
     00FB  10
     00FA  FF
     00F9  E5
 SP> 00F8  07
```

Figure 19. Stack Window

STEP Single Step (Trace) STEP

The STEP command executes a specified *hexadecimal* number of instructions, beginning at the current program counter (PC) address value. If a number is not specified, this command executes one instruction. The STEP, ST, and T commands are identical.

Syntax:

STEP [<*count*>]

Where:

<*count*>	The *hexadecimal* number of instructions to be executed. Hexadecimal $7FF is the maximum value.

NOTE: *Do not use any step command (ST, STEP, STEPFOR, STEPTIL, or T) if the PC points to internal RAM or EEPROM or if the code branches into internal RAM or EEPROM.*

The step commands are not real-time. They execute one instruction at a time, then return control to the monitor. Do not rely on timer values.

Examples:

>STEP	Execute the instruction at the current PC address value.
>STEP 2	Execute two instructions, starting at the current PC address value.

STEPFOR Step Forever STEPFOR

The STEPFOR command begins continuous instruction execution, beginning at the current program counter (PC) address value. Execution stops when a key is pressed.

Syntax:

```
STEPFOR
```

NOTE: *Do not use any step command (ST, STEP, STEPFOR, STEPTIL, or T) if the PC points to internal RAM (for instance, option RAM) or EEPROM or if the code branches into internal RAM or EEPROM.*

The step commands are not real-time. They execute one instruction at a time, then return control to the monitor. Do not rely on timer values.

Example:

```
>STEPFOR       Execute instructions continuously until the
               user presses a key.
```

STEPTIL Single Step to Address STEPTIL

The STEPTIL command continuously executes instructions from the current program counter (PC) address value until the PC reaches the specified address.

Syntax:

STEPTIL <*address*>

Where:

<*address*>	The address at which instruction execution stops; this location must be an instruction address.

NOTE: *Do not use any step command (ST, STEP, STEPFOR, STEPTIL, or T) if the PC points to internal RAM or EEPROM or if the code branches into internal RAM or EEPROM.*

The step commands are not real-time. They execute one instruction at a time, then return control to the monitor. Do not rely on timer values.

Example:

>STEPTIL 0400	Execute instructions continuously until the PC value is 0400 during an opcode fetch cycle.

STOP Stop Program Execution STOP

The STOP command stops user program execution and updates the debug screens with current data.

Syntax:

```
STOP
```

Example:

```
>STOP      Stop program execution and update the debug
screen.
```

SXB Set Multiplexer SXB

The SXB command sets the analyzer to capture either a 16-bit timetag counter and the eight pod B logic clips or a 24-bit timetag counter. The default is to use the 16-bit timetag and pod B.

NOTE: *When using the logic clip cables, always attach the black clip to ground.*

Syntax:

```
SXB CLIPS|TAGS
```

Where:

CLIPS	Directs the analyzer to capture the pod B logic clips with 16-bit timetag counter.
TAGS	Directs the analyzer to capture a 24-bit timetag counter.

Example:

```
>SXB TAGS  Select extended time tag bits.
```

SYSINFO System Information SYSINFO

The SYSINFO command calls to DOS for the amount of memory available, then displays this information in the debug F10 window.

Syntax:

```
SYSINFO
```

Example:

```
>SYSINFO        Show system information.
```

```
Total memory available: 187488

Largest free block: 187488
```

T Single Step (Trace) T

The T command executes a specified *hexadecimal* number of instructions beginning at the current program counter (PC) address value. If a number is not specified, this command executes one instruction. The T, ST, and STEP commands are identical.

Syntax:

T [<*count*>]

Where:

<*count*>	The *hexadecimal* number of instructions to be executed. Hexadecimal $7FF is the maximum value.

NOTE: *Do not use any step command (ST, STEP, STEPFOR, STEPTIL, or T) if the PC points to internal RAM (for instance, option RAM) or EEPROM or if the code branches into internal RAM or EEPROM.*

The step commands are not real-time. They execute one instruction at a time, then return control to the monitor. Do not rely on timer values.

Examples:

>T	Execute the instruction at the current PC address value.
>T 4	Execute four instructions beginning at the current PC address value.

TIMETAG Time Tag Clock Source TIMETAG

The TIMETAG command selects the source and frequency for the analyzer time tag clock. Entering this command with no parameter values brings up the time tag window, from which you can select a frequency or a source. If you select a frequency, the system uses an internal source of the frequency chosen. If you select a source, the system prompts for an appropriate frequency value.

If you select the external source, connect logic clip TT_OSC (white) of the pod B cable to the external clock source. (The pod B connector is the closest to the front of the station module.)

NOTE: *When using the logic clip cables, always attach the black clip to ground.*

Syntax:

TIMETAG [[<*source*>] [<*rate*>]]

Where:

<*source*> The source for the analyzer timetag clock. The clock can be provided from one of four sources:

1. Internal (MMDS platform board), programmable, external or emulation module. With no source specified, the internal source is understood.

To specify other possible clock sources, use the following:

2. EM Use the bus clock of the MCU being emulated
3. PR Use a programmable clock in the range of 50 to 50,000 Hz
4. EX Use an external clock provided through the TT_OSC logic clip (white) of pod B

TIMETAG Time Tag Clock Source TIMETAG

<rate>	The frequency for the timetag clock. When specifying the internal source, the value is in MHz with 1, 2, 4, 8, and 16 as the possible options. The programmable timetag source requires an integer value between 50 and 50,000 Hz. When an external source or the MCU bus clock are used for the timetag clock, the <rate> value entered is only used for timing calculations displayed in the analyzer data screen.

Examples:

>TIMETAG	Display the time tag window (for user selection of a frequency or source).
>TIMETAG 8	Select the 8-MHz time tag frequency.
>TIMETAG PR 120	Select a 120-Hz programmable source.
>TIMETAG EX 32768	Select the external source and notify host that frequency is 32.768 kHz.

V Set/Clear V Bit V

NOTE: *This command is for the MMDS08 only.*

The V command sets the V bit in the condition code register (CCR) to the specified value.

NOTE: *The CCR bit designators are at the lower right of the CPU window. The CCR pattern is V11HINZC (V is overflow, H is half-carry, I is IRQ interrupt mask, N is negative, Z is zero and C is carry). A letter in these designators means that the corresponding bit of the CCR is set; a period means that the corresponding bit is clear.*

Syntax:

```
V 0|1
```

Where:

0	Clears the V bit
1	Sets the V bit

Example:

>V 0	Clears the V bit in the CCR.

VAR Display Variable VAR

The VAR command displays the specified address and its contents in the variables F8 window. If a map file has been loaded, symbols and labels from the source code can be used as arguments.

As many as 32 variables can be declared in the variables F8 window. The window shows 11 at a time. Using the VAR command establishes such a variable.

(The RTVAR command also establishes variables for display in the variables F8 window, but such variables are not real-time. The 32-variable maximum applies to variables established by both the RTVAR and VAR commands.)

The <*type*> argument enables display of variables in byte, word, or string format. A byte display is hexadecimal and binary, a word display is hexadecimal and decimal, and a string display is ASCII.

For an ASCII string, the optional <*n*> argument specifies the number of characters; the default is the maximum 11 characters. Control and other non-printing characters appear as periods (.).

Syntax:

```
VAR[.<type>] <address>|<symbol> [<n>]
```

Where:

<*type*>	The variable type to display: B (byte, the default), W (word), or S (string)
<*address*>	The address of the memory variable
<*n*>	The number of characters to be displayed. Used only with the string type. If <*n*> is omitted, 11 ASCII characters will be visible in the window, beginning at the <*address*> argument location.
<*symbol*>	A symbol loaded from a .MAP file

VAR Display Variable VAR

Examples:

>VAR 100	Display (in hexadecimal and binary) the byte at address 100
>VAR.B 110	Display (in hexadecimal and binary) the byte at address 110
>VAR.W 102	Display (in hexadecimal and decimal) the word at address 102
>VAR.S 200 5	Display the 5-character ASCII string starting at address 200

VER Display Version VER

The VER command displays the version of the host software and of the current personality (.MEM) file. The abbreviated VER is equivalent to the VERSION command.

Syntax:

```
VER
```

Example:

>VER	Display the version numbers of the host software and the currently loaded personality file.

VERSION Display Version VERSION

The VERSION command displays the version of the host software and of the current personality (.MEM) file. The abbreviated VER form of this command also can be used.

Syntax:

```
VERSION
```

Example:

>VERSION	Display the version numbers of the host software and the currently loaded personality file.

VF View File VF

The VF command allows the user to view any file. If no argument is supplied, the command pops up a file select window.

Syntax:

VF [<*filename*>]

Where:

<*filename*>	Any text file

Examples:

>VF test, log	Displays test.log in view file screen
>VF *.asm	Displays the names of the .asm files

NOTE: *The view file buffer size is determined by the total memory available (see SYSINFO command), if the selected file exceeds memory resources. The entire file will not be viewed.*

WAIT Pause between Commands WAIT

The WAIT command causes the command interpreter to pause for a specified *hexadecimal* number of seconds. (The default is five.) This command is useful primarily in script files.

Syntax:

```
WAIT [<n>]
```

Where:

<*n*>	The *hexadecimal* number of seconds to pause.

Example:

>WAIT A	Pause the command interpreter for 10 seconds.

WAIT4RESET Wait for Target Reset WAIT4RESET

The WAIT4RESET command puts the emulation MCU into the reset state until the target system provides a reset signal.

For this command to function properly, enable the state of the MMDS with a reset signal from the target system. (See the explanation of the RESETIN command.) To restore the emulator to the IDLE state, enter the RESET command.

Syntax:

```
WAIT4RESET
```

Example:

```
>WAIT4RESET     Wait for reset.
```

WHEREIS Display Symbol Value WHEREIS

The WHEREIS command displays a symbol or address. If the argument is a symbol, this command displays the symbol's address. If the argument is an address, this command displays the corresponding symbol, if one is assigned. If the symbol is the same as a hexadecimal address, the command shows the hexadecimal address, not the address of the symbol.

Syntax:

WHEREIS <*symbol*> | <*address*>

Where:

<*symbol*>	A symbol listed in the symbol table
<*address*>	An address for which a symbol is desired

Examples:

>WHEREIS START	Display the symbol START and its value.
>WHEREIS 0100	Display the value 0100 and its symbol, if any.

X Set X Index Register X

The X command sets the index register (X) to the specified value. The X command is identical to the XREG command.

Syntax:

```
X <n>
```

Where:

<*n*>	The new value for the X register

Example:

>X 05	Set the index register value to 05.

XREG Set X Index Register XREG

The XREG command sets the index register (X) to the specified value. The XREG command is identical to the X command.

Syntax:

XREG <*n*>

Where:

<*n*>	The new value for the X register

Example:

>XREG F0	Set the index register value to F0.

Z Set/Clear Z Bit Z

The Z command sets the Z bit in the condition code register (CCR) to the specified value.

NOTE: *The CCR bit designators are at the lower right of the CPU window. The CCR pattern is V11HINZC (V is two's complement overflow for M68HC08 MCU only and is 1 for M68HC05 MCU, H is half-carry, I is IRQ interrupt mask, N is negative, Z is zero, V is overflow, and C is carry). A letter in these designators means that the corresponding bit of the CCR is set; a period means that the corresponding bit is clear.*

Syntax:

```
Z 0|1
```

Where:

0	Clears the Z bit
1	Sets the Z bit

Example:

>Z 0	Clears the Z bit in the CCR.

ZOOM Resize Source Window ZOOM

The ZOOM command toggles the size of the source window between normal and enlarged. Another way to resize the source window is to enter the ALT-Z keyboard combination.

Syntax:

```
ZOOM
```

Example:

```
>ZOOM     Resize the source window.
```

S-Record Information

Contents

Introduction 199

S-Record Content 200

S-Record Types. 201

S-Record Creation. 202

S-Record Example 202

Introduction

The Motorola S-record format was devised for the purpose of encoding programs or data files in a printable format for transportation between computer systems. This transportation process can therefore be monitored and the S-records can be easily edited.

S-Record Content

When observed, S-records are essentially character strings made of several fields which identify the record type, record length, memory address, code/data, and checksum. Each byte of binary data is encoded as a two-character hexadecimal number: the first character representing the high-order four bits and the second the low-order four bits of the byte.

Five field which comprise an S-record are shown below:

TYPE	RECORD LENGTH	ADDRESS	CODE/DATA	CHECKSUM

where the fields are composed as shown in **Table 15**.

Table 15. S-Record Field Description

Field	**Printable Characters**	**Contents**
Type	2	S-record type — S0, S1, etc.
Record Length	2	Character pair count in the record, excluding the type and record length.
Address	4, 6, or 8	2-, 3-, or 4-byte address at which the data field is to be loaded into memory.
Code/Data	0–2n	From 0 to n bytes of executable code, memory loadable data, or descriptive information. For compatibility with teletypewriter, some programs may limit the number of bytes to as few as 28 (56 printable characters in the S-record).
Checksum	2	Least significant byte of the one's complement of the sum of the values represented by the pairs of characters making up the record length, address, and the code/data fields.

Each record may be terminated with a CR/LF/NULL. Additionally, an S-record may have an initial field to accommodate other data such as line numbers generated by some time-sharing system.

Accuracy of transmission is ensured by the record length (byte count) and checksum fields.

S-Record Types

Eight types of S-records have been defined to accommodate the several needs of the encoding, transportation, and decoding functions. The various Motorola upload, download, and other record transportation control programs, as well as cross assemblers, linkers, and other file-creating or debugging programs, utilize only those S-records which serve the purpose of the program. For specific information on which S-records are supported by a particular program, the user manual for that program must be consulted.

NOTE: *The MMDS supports only the S0, S1, and S9 record types. All data before the first S1 record is ignored. Thereafter, all records must be S1 type until the S9 record, which terminates data transfer.*

An S-record format may contain the following record types:

S0	Header record for each block of S-records. The code/data field may contain any descriptive information identifying the following block of S-records. The address field is normally zeroes.
S1	Code/data record and the two-byte address at which the code/data is to reside.
S2–S8	Not applicable to MMDS.
S9	Termination record for a block of S1 records. Address field may optionally contain the two-byte address of the instruction to which control is to be passed. If not specified, the first interplant specification encountered in the input will be used. There is no code/data field.

Only one termination record is used for each block of s-records. Normally, only one header record is used, although it is possible for multiple header records to occur.

S-Record Creation

S-record format programs may be produced by dump utilities, debuggers, cross assemblers, or cross linkers. Several programs are available for downloading a file in S-record format from a host system to an 8-bit or 16-bit microprocessor-based system.

S-Record Example

Shown here is a typical S-record format, as printed or displayed:

```
S00600004844521B
S1130000285F245F2212226A000424290008237C2A
S11300100002000800082529001853812341001813
S113002041E900084E42234300182342000824A952
S107003000144ED492
S9030000FC
```

The above format consists of an S0 header record, four S1 code/data records, and an S9 termination record.

The S0 header record is described in **Table 16**.

Table 16. S0 Record Description

Field	S-Record Entry	Description
Type	S0	S-record type S0, indicating a header record.
Record Length	06	Hexadecimal 06 (decimal 6), indicating six character pairs (or ASCII bytes) follow.
Address	0000	Four-character two-byte address field, zeroes.
Code/Data	48 44 52	Descriptive information identifies the following S1 records: ASCII H, D, and R — "HDR"
Checksum	18	Checksum of S0 record.

The first S1 record is explained in **Table 17**.

Table 17. S1 Record Description

Field	S-Record Entry	Description
Type	S1	S-record type S1, indicating a code/data record to be loaded/verified at a two-byte address.
Record Length	13	Hexadecimal 13 (decimal 19), indicating 19 character pairs, representing 19 bytes of binary data, follow.
Address	0000	Four-character two-byte address field; hexadecimal address 0000, indicates location where the following data is to be loaded.
Code/ Data	**Opcode**	**Instruction**
	285F	BHCC$0161
	245F	BCC$0163
	2212	BHI$0118
	226A	BHI$0172
	000424	BRSET0, $04, $012F
	2900	BHCS$010D
	08237	BRSET4, $23, $018C
Checksum	2A	Checksum of the first S1 record.

The 16 character pairs shown in the code/data field of **Table 17** are the ASCII bytes of the actual program.

The second and third S1 code/data records each also contain $13 (19) character pairs and are ended with checksum 13 and 52, respectively. The fourth S1 code/data record contains 07 character pairs and has a checksum of 92.

The S9 termination record is explained in **Table 18**.

Table 18. S9 Record Description

Field	S-Record Entry	Description
Type	S9	S-record type S9, indicating a termination record.
Record Length	03	Hexadecimal 03, indicating three character pairs (three bytes) follow.
Address	0000	Four-character two-byte address field, zeroes.
Code/Data		There is no code/data in a S9 record.
Checksum	FC	Checksum of S9 record.

Each printable ASCII character in an S-record is encoded in binary. **Table 19** gives an example of encoding for the S1 record. The binary data is transmitted during a download of an S-record from a host system to a 8- or 16-bit microprocessor-based system.

Table 19. Example of S-Record Encoding

TYPE				LENGTH				ADDRESS								CODE/DATA									CHECKSUM			
S		1		1		3		0		0		0		0		2		8		5		F		...	2		A	
5	3	3	1	3	1	3	3	3	0	3	0	3	0	3	0	3	2	3	8	3	5	4	6	...	3	2	4	1
0101	0011	0011	0001	0011	0001	0011	0011	0011	0000	0011	0000	0011	0000	0011	0000	0011	0010	0011	1000	0011	0101	0100	0110	...	0011	0010	0100	0001

Index

A

A, ... 106
ACC 107
Arguments, Command-Line
    Commands 103
ARM, ... 108
ASM 109

B

BAUD 61, 110
Baud Rate 61
BAUDCHK 61, 111
BELL 112
BF 113
BR 114
BSAMODE 115
BSATT 116
Bus State Analyzer 77
  Collecting Bus Data 87
  Defining Events 79
  Searching the Trace Buffer 95
  Selecting Options 86
  Selecting the Trigger Mode 84
  Using the Time-Tag Clock 96
  Viewing Data 88

C

C 117
Cables, Connecting
  Host Computer 26
  Power 27
  Target 27
CCR 118
CHIPINFO 119
CLEARMAP 120
COLORS 121
Colors, Changing Screen 58, 70, 121
Command Lines
  A 106
  ACC 107
  ARM 108
  ASM 64, 109
  BAUD 54, 61, 110
  BAUDCHK 61, 111
  BELL 66, 112
  BF 65, 113
  BR 57, 72, 114
  BSAMODE 115
  BSATT 116
  C 117
  CCR 118
  CHIPINFO 68, 119
  CLEARMAP 120
  COLORS 58, 70, 121
  Commands 102
  DARM 122
  DASM 64, 123
  ENDBSA 124
  EVAL 67, 125
  EXIT 70, 125, 126
  G 73, 127
  GETBSA 128

GO 57, 73, 129
GOTIL 57, 73, 130
H 131
HELP 68, 132
HOMEBSA 133
HREG 134
HX 135
I 136
INFO 57, 67, 137
LF 68, 138
LOAD 139
LOADMAP 140
LOADMEM 62, 141
LOADTRIGGERS 142
MD 71, 143
MM 65, 144, 145
N 146
NEXTA 147
NEXTB 148
NEXTC 149
NEXTD 150
NEXTE 151
NOBR 72, 152
OSC 63, 153
PC 154
QUIT 70, 155
REG 156
REM 67, 157
RESET 72, 158
RESETGO 72, 73, 159
RESETIN 72, 160
RESETOUT 72, 161
RTMEM 162
RTVAR 163
SCREENBSA 165
SCRIPT 66, 166
SETMEM 53, 62, 167, 168
SHELL 70, 169
SHOWBSA 170
SHOWMEM 68, 171
SHOWTRIGGER 172
SNAPSHOT 69, 173
SOURCE 48, 69, 174
ST 73, 175
STACK 52, 176
STEP 57, 73, 177
STEPFOR 73, 178
STEPTIL 73, 179
STOP 57, 180
Summary (Table) 103
SXB 181
Syntax 102
SYSINFO 67, 182
T 73, 183
TIMETAG 184
V 186
VAR 49, 188
VER 67, 189
VERSION 67, 190
VF 191
WAIT 66, 192
WAIT4RESET 72, 193
WHEREIS 67, 194
X 195
XREG 196
Z 197
ZOOM 57, 70, 198
Command Types
  BSA 74
  Debug 71
  Initialization 60
  System 66
Command Types, Bus State Analyzer 77
Connector, Cable
  Pin Assignments 28
  Signal Descriptions 28

CPU Registers
  Setting 71

D

DARM 122
DASM 123

E

EM
  Installing 25
  Removing 25
ENDBSA 124
EVAL 125
EXIT 126

F

Fuse Replacement 32

G

G 127
GETBSA 128
GO 129
GOTIL 130

H

H 131
Hardware Installation
  Configuring the Platform Board 22
  Connecting Cables 26
  Fuse Replacement 32
  Installing the EM 25
  Introduction 20
  Pin Assignments, Connector 28
  Removing the EM 25
  Reset Switch 28
  Signal Descriptions, Connector 28
HELP 132
HOMEBSA 133
Host Computer
  Requirements 16
HREG 134
HX 135

I

I 136
INFO 137
Initialization and Loading 35
  Clock Speed 63
  Memory 63
Initialization Commands 60
Introduction 11

J

Jumper Headers
  Factory Test (J1) 23
  Port Voltage Control (J2–J4) 24

K

Key Commands
  Debug Screen Windows 45
  Main Windows 42
  Pop-Up Windows 51
  Source/Code F2 Window 49

L

LF 138
LOAD 139
Loading and Initialization 35
LOADMAP 140

LOADMEM 141
LOADTRIGGERS 142

M

MMDS0508 22
Manual
    Organization 12
MD 143
Memory
    Mapping 61, 62
MM 144, 145
MMDS
    Introduction 11
MMDS05
    Debug Screen 42, 43
    Running 38
MMDS08
    Debug Screen 42, 44
    HREG 134
    HX 135
    Introduction 11
    Running 39
Mouse Operation 56

N

N 146
NEXTA 147
NEXTB 148
NEXTC 149
NEXTD 150
NEXTE 151
NOBR 152

O

OSC 153

P

PC 154
Pin Assignments, Connector 28
Platform Board, Configuration 22
Pod 31
Port Voltage Control
        Jumper Headers (J2–J4) 24

Q

QUIT 155

R

REG 156
REM 157
RESET 158
RESETGO 159
RESETIN 160
RESETOUT 161
RTMEM 162
RTVAR 163

S

SCREENBSA 165
Screens
    Baud Window 54
    CPU Window 47
    Debug F10 Window 50
    Emulator Clock Frequency Window . . . 54
    Emulator Clock Frequency
        Window MMDS05 . . . . 54, 153
    Emulator Clock Frequency
        Window MMDS08 . . . . 55, 153
    Introduction 42
    Memory F3 Window 50
    MMDS05 Debug Screen 43

- MMDS08 Debug Screen 44
- Set Memory Window 53
- Source/Code F2 Window 47
- Stack Window 52
- Status Area 45
- User 41
- Variables F8 Window 49

SCRIPT 166

Script Files
- STARTUP.05 . . . . 60, 66
- STARTUP.08 . . . . 60, 66

Serial 29

SETMEM 167, 168

SHELL 169

SHOWBSA 170

SHOWMEM 171

SHOWTRIGGER 172

SNAPSHOT 173

Software
- Distribution Format 36
- Installation 36
- Personality Files 37
- Target, Loading 63
- Using 37

SOURCE 174

ST 175

STACK 176

STARTUP.05 (Script File) 60, 66

STARTUP.08 (Script File) 60, 66

STEP 177

STEPFOR 178

STEPTIL 179

STOP 180

SXB 181

SYSINFO 182

System
- Commands 66
- Components 15
- Connections 26

T

T 183

TIMETAG. 184

V

V 186

VAR 187, 188

VER 189

VERSION 190

VF 191

W

WAIT 192

WAIT4RESET 193

WHEREIS 194

Windows Operation 37

X

X 195

XREG 134, 135, 196

Z

Z 197

ZOOM 198

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Motorola data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and Ⓜ are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

How to reach us:
USA/EUROPE/Locations Not Listed: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036.1-800-441-2447 or 602-303-5454
MFAX: RMFAX0@email.sps.mot.com – TOUCHTONE 602-244-6609
INTERNET: http://Design-NET.com
JAPAN: Nippon Motorola Ltd.; Tatsumi-SPD-JLDC, 6F Seibu-Butsuryu-Center, 3-14-2 Tatsumi Koto-Ku, Tokyo 135, Japan. 03-81-3521-8315
ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd.; 8B Tai Ping Industrial Park, 51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852-26629298

MOTOROLA