MMEVS

MODULAR EVALUATION SYSTEM
for 68HC05 and 68HC08
MICROCONTROLLERS
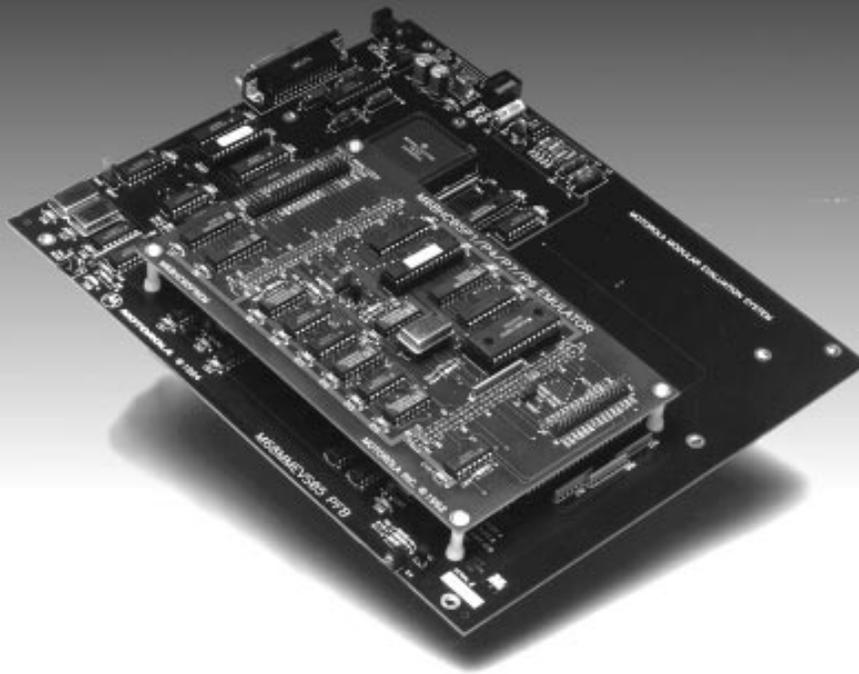
OPERATIONS MANUAL

Ⓜ MOTOROLA

---

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

List of Sections

MOTOROLA

CSIC

MICROCONTROLLERS

Table of Contents

Table of Contents

Contents

Introduction

The M68MMEVS05/08 Motorola Modular Evaluation System (MMEVS) is a tool for developing embedded systems based on an MC68HC05 or MC68HC08 microcontroller unit (MCU). The MMEVS 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 MMEVS 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 MMEVS system contains the platform board (M68MMPFB0508), an emulation module (EM), and a target cable assembly. An EM completes MMEVS functionality for a particular MCU or MCU family. To accommodate emulation of the numerous MCUs available, the MMEVS uses a variety of different EMs. Refer to 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 MMEVS, an IBM (or compatible) host computer and a user-supplied power supply are needed. A 9-to-25 pin RS-232 serial cable also is provided with the MMEVS.

About this Manual

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

- **Installation** on page 13 explains M68MMPFB0508 hardware.

- **Loading and Initialization** on page 21 explains how to load and initialize the MMEVS system software.

- **User Screens** on page 27 explains the window screens, as well as how to use a mouse.

- **Operation Fundamentals** on page 45 describes command usages.

- **Command-Line Commands** on page 61 explains MMEVS command syntax.

- **S-Record Information** on page 137 gives reference information about Motorola S-records.

System Features

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

- Real-time, non-intrusive, in-circuit emulation

- MC68HC11K1 system controller for fast command transfer

- 64 Kbytes of emulation memory to accommodate the largest available ROM size

- 64 hardware instruction breakpoints over the 64-K memory map

- A DOS personality file for each EM. Each personality file provides a foreground memory-map definition.

- Latch-up resistant design (47-Ω series resistor on I/O connections to the target system) to make power-up sequencing unimportant.

- Four software-selectable internally generated oscillator clock sources

- Command and response logging to disk files

- SCRIPT command for automatic execution of a sequence of MMEVS 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: 8.25 inches deep, 10.5 inches wide

Connections, configuration, specifications, 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 M68MMPFB0508:

- **Platform board:** The M68MMPFB0508 platform board

- **9-to-25 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*

- **MMEVS documentation**: An *MMEVS05/08 Operations Manual* (MMEVS0508OM/D – this manual).

- **Software Release Guide:** Documentation on the current release of system software.

Separately purchased Motorola personality products include:

- **An emulation module (EM):** One of many printed circuit boards that complete MMEVS 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 M68MMPFB0508 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 MMEVS 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 MMEVS system

User supplied components include:

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

- **Power supply:** Required power is +5 volts @ 1 amp

Host Computer Requirements

The host computer for the MMEVS 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 MMEVS 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 MMEVS system software also supports a Microsoft, Logitech, or IBM mouse. Other mice may be compatible, but Motorola does not guarantee their satisfactory performance with MMEVS software.

Acronyms

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

Table 1. Acronym Definitions

Term	Description
M68MMPFB0508	The platform board where common hardware for all M68HC05 and M68HC08 emulation resides.
EM	An emulation module that connects to the platform board to customize the MMEVS 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.
MMEVS	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 MMEVS debugger).
CASM	Cross assembler that assembles M68HC05 (CASM05) or M68HC08 (CASM08) code.
MMEVS05	Motorola Modular Evaluation System for M68HC05 emulation. Requires MMEVS05 debugger software and an emulation module that supports a M68HC05 MCU.
MMEVS08	Motorola Modular Evaluation System for M68HC08 emulation. Requires MMEVS08 debugger software and an emulation module that supports a M68HC08 MCU.

MMEVS0508OM/D

Contents

Introduction

Complete MMEVS installation consists of configuring the platform board, configuring and installing the appropriate emulation module (EM), and making system cable connections. Consult the introductory section of this document for a list of all the system components, 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 MMEVS installation.

Figure 1 shows the M68MMPFB0508 platform board. The hardware reset switch and power LED are on the front of the platform board. The power connector is to the back and the 25-pin RS-232 serial connector is on the left facing the platform board. The circular connector, P4, is also on the left side of the platform board. This connector is the +5-volt out connector, which is reserved for future features.

NOTE: *This manual uses the words left and right relative to left and right hands as the user faces the front of the platform board.*
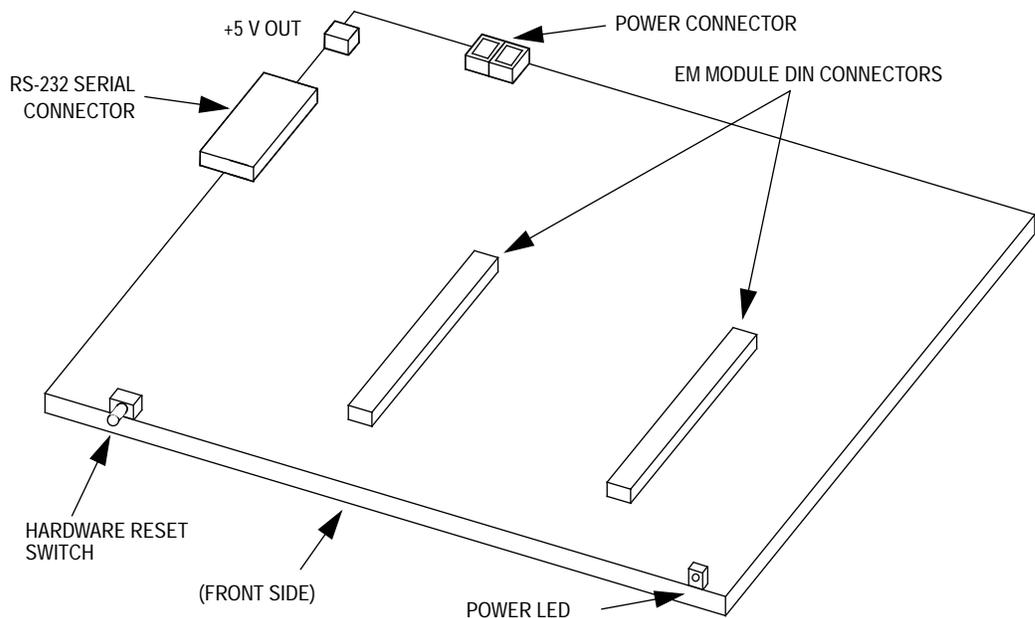
Figure 1. M68MMPFB0508 Platform Board

Configuring the Platform Board

The M68MMPFB0508 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 M68MMPFB0508. 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 MMEVS operation.
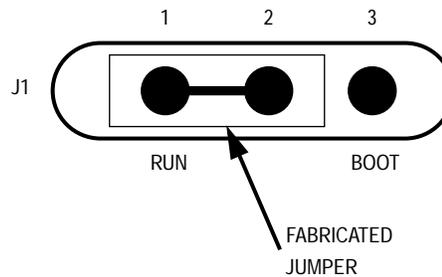
The alternate jumper position has significance only for factory personnel.

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 MMEVS operation, unless the EM user's manual says that the EM is a low-voltage board.
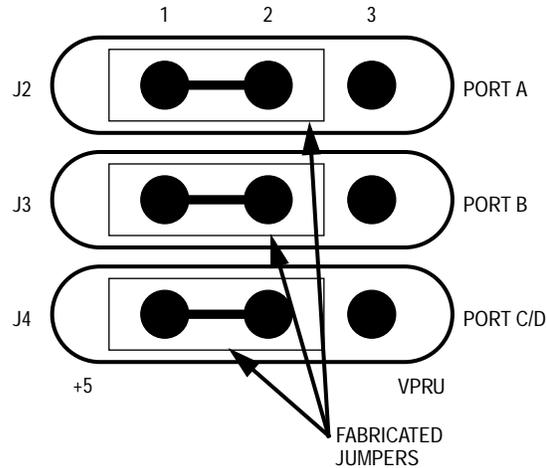
If the EM can operate at low voltage, any of the ports can be operated at the low-voltage level. To do so, reposition the fabricated jumper of the corresponding header to pins 2 and 3. Jumper header J2 controls the voltage level of port A, jumper header J3 controls the voltage level of port B, and jumper header J4 controls the voltage level of port C or D, whichever pertains to the EM.

MMEVS0508OM/D

Installing the EM

Follow steps 1 through 3 to install an EM:

1. Make sure that platform board power is off.

2. Make sure that nylon spacers are in the correct positions for the emulation module (EM).

3. 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 nylon spacers and make sure that the DIN connectors are joined together firmly.

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

Follow steps 1 and 2 to remove an EM:

1. Turn off the power supply to the platform board.

2. 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.

Making System Connections

The specific application determines the number of MMEVS 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 following sections: **Host Computer Connection**, **Target Cable Connection**, and **Power Connection**.

Host Computer
Connection

Use the 9-to-25 pin serial cable to connect a host computer's 9-pin serial port to the MMEVS 25-pin serial cable connector (on the left side of the platform board). Note which computer serial port is used. If the COM1 port (the default) is not used, the port number must be specified in the MMEVS software start-up command.

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

Target Cable
Connection

If the MMEVS 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 and the appropriate target cable.

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

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.

MMEVS0508OM/D

Power Connection

The final MMEVS connection is a user-supplied power supply. The MMEVS requires a +5 Vdc @ 1.0 amp power supply.

Use lever terminal P3 to connect power to the MMEVS. Contact 1 (black lever) is the ground. Contact 2 (red lever) is for V_{DD}, the +5 Vdc power. Use 20 or 22 AWG wire for power connections. For each wire, trim the insulation back a short distance from the end, lift the appropriate P3 lever, insert the bare wire into P3, and close the lever.

Do not use wire larger than 20 AWG in connector P3 because such wire will damage the connector.

Make sure the power to the platform board is turned off when installing the EM or removing the EM from the M68MMPFB0508. Sudden power surges can damage MMEVS circuits.

This completes the cable connections. The MMEVS software is ready to be installed in the host computer.

Reset Switch

RS-232 handshake signals control MMEVS resets. A reset initializes the control board from its start-up point. If the computer serial port does not implement handshaking, reset the MMEVS 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 MMEVS systems. For pinout information for a particular EM connector, refer to the corresponding EM user's manual.

This diagram shows pin numbering for the 25-pin serial connector of the control board. **Table 1** lists the signal available at each pin, as well as the signals transmitted on the 9-lead serial cable.

13   1

○○○○○○○○○○○○○
○○○○○○○○○○○○

25   14
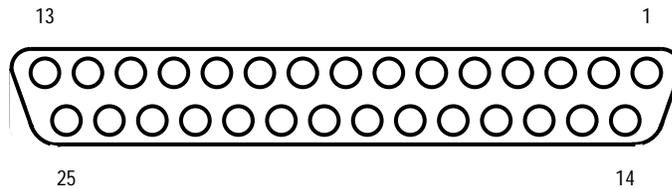
Table 1. Serial Connector and Cable Pin Assignments

Connector Pin	Mnemonic	Signal
1	GND	GROUND
2	TX	TRANSMIT DATA — Serial data input line
3	RX	RECEIVE DATA — Serial data output line
4	RTS	REQUEST TO SEND — Input signal that requests permission to transfer data
5	CTS	CLEAR TO SEND — Output signal that indicates a ready-to-transfer data status
6	DSR	DATA SET READY — Output signal that indicates on-line/in-line service/active status
7	SIG-GND	SIGNAL GROUND — Signal ground or common return connection between the MMEVS and host computer
8	DCD	DATA CARRIER DETECT — Output signal that indicates detection of an acceptable carrier signal
9-19, 21-25	—	No connection
20	DTR	DATA TERMINAL READY — Input signal that indicates on-line/in-line/active status

Loading and Initialization

Contents

Software Distribution Format

MMEVS software is distributed on 3.5-inch high-density diskettes. The install process places the RAPID environment files and the MMEVS software files in the directory designated during the install process. **Table 2** describes the system files required to control the MMEVS system, where *x* denotes a version number. Refer to the Software Release Guide for information on other files loaded.

Table 2. MMEVS Software Files

File Name	Description
MMEVS05.EXE	MMEVS05 host software
MMEVS05X.EXE	MMEVS05 host software for running in a DOS window under Windows
MEVS05Vx.HLP	HELP command windows for the MMEVS05 commands
MMEVS08.EXE	MMEVS08 host software
MMEVS08X.EXE	MMEVS08 host software for running in a DOS window under Windows
MEVS08Vx.HLP	HELP command windows for the MMEVS08 commands

Installing MMEVS Software

The install process will place MMEVS and all supporting files on a hard drive. To install the MMEVS 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.

MMEVS0508OM/D

Personality Files

Various designs of MCUs require different configurations of the MMEVS 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 MMEVS 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 MMEVS 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 MMEVS Software

The correct executable file to run is dependent on which type EM is installed on the MMEVS platform board. If an HC05 EM is installed, the MMEVS05.EXE file should be run. If an HC08 EM is installed, the MMEVS08.EXE file should be run.

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

Alternatively, the MMEVS can be called from within the RAPID environment. Running the MMEVS05 or MMEVS08 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 MMEVS software in a DOS window under Windows, use the MMEVS05X.EXE or MMEVS08X.EXE files.*

Running MMEVS05

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

```
C:\MMEVS05>MMEVS05
```

Note these five options for the startup command:

1. If the MMEVS05 is connected to COM2, COM3, or COM4, add the corresponding integer to the command:
    ```
    C:\MMEVS05>MMEVS05 2
    ```

2. If the computer has a monochrome monitor, add BW to the command:
    ```
    C:\MMEVS05>MMEVS05 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:\MMEVS05>MMEVS05 –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:\MMEVS05>MMEVS05 <filename>
    ```

5. To specify a default baud rate of 9600, add the –B option:
    ```
    C:\MMEVS05>MMEVS05 –B
    ```

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

**Running
MMEVS08**

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

```
C:\MMEVS08>MMEVS08
```

Note these five options for the startup command:

1. If the MMEVS08 is connected to COM2, COM3, or COM4, add the corresponding integer to the command:
```
C:\MMEVS08>MMEVS08 2
```

2. If the computer has a monochrome monitor, add BW to the command:
```
C:\MMEVS08>MMEVS08 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:\MMEVS08>MMEVS08 –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:\MMEVS08>MMEVS08 <filename>
```

5. To specify a default baud rate of 9600, add the –B option:
```
C:\MMEVS08>MMEVS08 –B
```

NOTE: *Multiple options in the startup command should be separated by a space.*

MMEVS Communication

The host program establishes communications with the MMEVS 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 MMEVS05** and **Running MMEVS08** on the preceding pages to set the startup baud rate at 9600. If communication errors persist, turn off the disk cache (SMARTDRV.EXE) on the computer.

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

MMEVS0508OM/D

User Screens

Contents

Introduction

The user interface screen to the MMEVS 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.

Main Window Screens

Figure 2 and **Figure 3** show the debug screen for the MMEVS05 and the MMEVS08 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 42 for detailed information on mouse usage. **Table 3** lists the key commands available in any of the main windows.

MMEVS0508OM/D

```
┌──────── CPU ────────┐        ┌──────────────── SOURCE: init.asm ────────────────┐
 Acc   Xreg
 00    B9                        begin  lda   #$40
 PC    SP    CCR                        sta   tcr
 0029  FF    111HIN.C
                                        ldx   #flag
┌─── VARIABLES F8 ───┐                  clra
│RSLT     $0000 !0   │           clrbgn sta   0,x              ;clears
 CLKTIME  $30 %00110000                 incx
 CURVAL   $00 %00000000          -B>    cpx   rslt+4
 DISLINE1 S1=DCHAR                       bne  clrbgn
└─────── delete ─────┘          └─pc──br──go──stop──gotil──step──info──zoom────────┘

                                 resetin         ┌──────────── MEMORY F3 ───────────┐
                                 logfile          0080 53 31 3D 3D 44 43 48 41 52 S1=DCHAR
                                 COM1:57600        0088 47 2F 43 48 41 52 47 45 45 G/CHARGE   S2=0
                                                   0090 04 20 20 53 32 3D 4F .   20 20 PTIONS
                                                   0098 50 54 49 4F 4E 53 20 20
                                                  └──────────────────────────────────┘

                                ┌──────────── DEBUG F10 ────────────┐
                                 idle    Inst brkpt/Illegal Address

                                 >BR 0029
                                 >g
                                 >idle   Inst brkpt/Illegal Address
                                 >
                                 F1:Help F2:Code F3:Mem   F8:vars F9:rpt F10:Debug
                                └────────────────────────────────────┘
```

Figure 2. MMEVS05 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 ═══
              delete                        logfile        0050 53 31 3D 3D 44 43 48 41 52 S1=DCHAR
                                            COM1:57600     0058 47 2F 43 48 41 52 47 45 G/CHARGE
                                            targetpwr      0060 04 20 20 53 32 3D 4F .  S2=0
                                                           0068 50 54 49 4F 4E 53 20 20 PTIONS

                                            ═══════════════ DEBUG F10 ═══════════════
idle   Inst brkpt/Illegal Address

>BR 00A7
>g
>idle   Inst brkpt/Illegal Address
>
══ F1:Help  F2:Code  F3:Mem        F8:vars  F9:rpt  F10:Debug ══
```

Figure 3. MMEVS08 Debug Screen

Table 3. 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
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 4** explains the indicators that may appear in this area.

Table 4. Status Area Indicators

Indicator, Position	Status, Meaning
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	**Resetin** — 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 MMEVS 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 MMEVS08 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 MMEVS system. The other mode (Source) displays the source code from a .MAP file loaded into the MMEVS 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 MMEVS 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 MMEVS 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 5** 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 5. Source Window Key Commands

Name	Key	Description
Breakpoint	Alt-B	Sets or removes a breakpoint at highlighted line
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

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 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 command. 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 VAR command.

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.

Dashes replace the values when code is executing. Updated values reappear when execution stops.

**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** on page 61 for 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 MMEVS 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 stack window, the set memory window, the baud window, and the emulator clock frequency window may appear during MMEVS operation. **Table 6** lists the key commands for these subordinate windows.

Table 6. 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 MMEVS
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

MMEVS0508OM/D

Stack Window   The temporary stack window appears near the center of the debug screen
when the STACK command is entered. As **Figure 4** 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 4. Stack Window

Set Memory Window

The temporary set memory window (**Figure 5**) 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 5. Set Memory Window

Baud Window   The temporary baud window (**Figure 6**) 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.
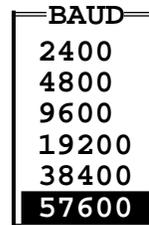
```
┌═BAUD═┐
│ 2400 │
│ 4800 │
│ 9600 │
│ 19200│
│ 38400│
│ 57600│
└──────┘
```

Figure 6. Baud Window

Emulator Clock Frequency Window

Depending on the EM board used, the M68MMPFB0508 platform board can supply the oscillator clock for the MCUs OSC1 input. Note that the EM being used will require a specific jumper configuration in order to use this clock source. Refer to the EM user's manual for the availability of this feature.

For the MMEVS05, four internally generated clock frequencies are available: 8 MHz, 4 MHz, 2 MHz, and 1 MHz. Entering emulator clock (OSC) command without the designated frequency brings up the temporary MMEVS 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 7**.

For the MMEVS08, five internally generated clock frequencies are available: 16 MHz, 8 MHz, 4 MHz, 2 MHz, and 1 MHz. The default emulator clock rate is 4 MHz as shown in **Figure 8**.

Before changing the clock rate, make sure that the emulation MCU supports the desired frequency and the appropriate jumpers are set correctly.
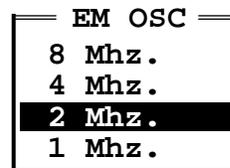
```
═══ EM OSC ═══
 8 Mhz.
 4 Mhz.
 2 Mhz.
 1 Mhz.
```

Figure 7. MMEVS05 Emulator Clock Frequency Window

```
═══ EM OSC ═══
16 Mhz.
 8 Mhz.
 4 Mhz.
 2 Mhz.
 1 Mhz.
```
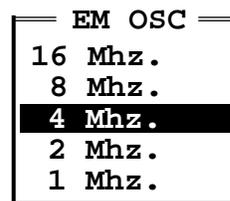
Figure 8. MMEVS08 Emulator Clock Frequency Window

Other Windows   In addition to the screen windows described in this section, several other transient dialog windows will be encountered. 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

MMEVS 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 MMEVS system.

Some MMEVS 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, 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.

MMEVS0508OM/D

- 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 MMEVS 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 MMEVS subdirectory to return to the default colors.*

Operation Fundamentals

Contents

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 MMEVS. Detail of specific commands is available in **Command-Line Commands** on page 61.

Operation of the MMEVS may be divided into three main areas:

- System Initialization
- System Information
- Debug Operation

A start-up script file, described in **Script Commands** on page 52**,** can be set up to perform a set of commands automatically each time the MMEVS software is run. This start-up file must have the name STARTUP.05 or STARTUP.08.

System Initialization

Initializing the MMEVS 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 MMEVS 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 MMEVS 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 MMEVS 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 MMEVS 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 MMEVS start-up (.MEM files also can be loaded at start-up by using the –M option with the MMEVS 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 M68MMPFB0508 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 MMEVS05, four internally generated clock frequencies are available: 8 MHz, 4 MHz, 2 MHz, and 1 MHz. Entering the emulator clock (OSC) command without the designated frequency brings up the temporary MMEVS emulator clock frequency window near the center of the debug screen.

For the MMEVS08, five internally generated clock frequencies are available: 16 MHz, 8 MHz, 4 MHz, 2 MHz, and 1 MHz.

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 MMEVS (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.

MMEVS0508OM/D

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.

MMEVS0508OM/D

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 MMEVS 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 MMEVS 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 MMEVS 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 MMEVS 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 MMEVS command line are hexadecimal. The input value 10, for example, is the decimal value 16.*

MMEVS0508OM/D

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 MMEVS 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 MMEVS 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.

Log File Commands

The MMEVS 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

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.

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 MMEVS 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 MMEVS 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 MMEVS 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 MMEVS 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 (MMEVS08 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** on page 61 for examples on how to modify CPU register values.

Memory Display

Memory contents are displayed in the memory F3 window. Thirty-two 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.

The VAR command displays the specified address and its contents in the variables F8 window. 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.

Command-Line Commands

Contents

Introduction

Keyboard entry is the primary means of MMEVS 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.

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 8** for command choices.

<argument> One or more arguments. **Table 7** 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 7. Argument Types

Argument Type	Syntax Indicators	Explanation
Numeric	<n>, <rate>, <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.
Commands	<commands>	Items from the command set may be used as an argument for the HELP command.
Operator	<op>	+ (add); − (subtract); * (multiply); or / (divide)
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 8 lists the command-line commands. Individual explanations of each of these commands follow the table. Note that the command parser of the MMEVS host software is not case sensitive.

Table 8. Command Summary

Mnemonic	Description	Mnemonic	Description
A	Set accumulator	OSC	Select emulator clock frequency
ACC	Set accumulator	PC	Set program counter
ASM	Assemble instructions	QUIT	Terminate host session
BAUD	Set communications baud rate	REG	Display registers
BAUDCHK	Baud rate check	REM	Add comment to script file
BELL	Sound bell	RESET	Reset emulation MCU
BF	Block fill	RESETGO	Reset and restart MCU
BR	Set instruction breakpoint	RESETIN	Reset input enable
C	Set/Clear C bit	RESETOUT	Reset output enable
CCR	Set condition code register	SCRIPT	Execute script file
CHIPINFO	Chip help information	SETMEM	Customize memory map
CLEARMAP	Remove symbols	SHELL	Access DOS
COLORS	Set screen colors	SHOWMEM	Display memory map
DASM	Disassemble instructions	SNAPSHOT	Save screen to a log file
EVAL	Evaluate argument	SOURCE	Source window display
EXIT	Terminate host session	ST	Single step (Trace)
G	Begin program execution	STACK	Display stack
GO	Begin program execution	STEP	Single step (Trace)
GOTIL	Execute program until address	STEPFOR	Step forever
H	Set/Clear H bit	STEPTIL	Single step to address
HELP	Display help information	SYSINFO	System information
HREG*	Set H register	T	Single step (Trace)
HX*	Set HX index register	V*	Set overflow V bit
I	Set/Clear I bit	VAR	Display variable
INFO	Display line information	VER	Display version
LF	Log file	VERSION	Display version
LOAD	Load S19 file	WAIT	Pause between commands
LOADMAP	Load symbols	WAIT4RESET	Wait for target reset
LOADMEM	Load personality file	STOP	Stop program execution
MD	Memory display	WHEREIS	Display symbol value
MM	Memory modify	X	Set X index register
N	Set/Clear N bit	XREG	Set X index register
NOBR	Clear breakpoints	Z	Set/Clear Z bit
* MMEVS08 only		ZOOM	Resize source window

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.

MMEVS0508OM/D

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.

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
0100    9D     NOP       >CLRA
0100    4F     CLRA
0101    9D     NOP       >.
```

The second example shows the ASM command with no argument:
```
>ASM
0101    9D     NOP       >STA      0A
0101    B70A   STA       0A
0103    9D     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, MMEVS 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 MMEVS command.*

BAUDCHK

Baud Rate Check

BAUDCHK

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

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.

MMEVS0508OM/D

CCR CCR

Set Condition Code Register

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 9**. 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 9. 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 MMEVS is executed again, the software applies the newly selected colors.

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

Syntax:

    COLORS

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.*

MMEVS0508OM/D

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 EXIT

Terminate Host Session

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.

MMEVS0508OM/D

GO GO

Begin Program Execution

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.
```

MMEVS0508OM/D

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.

MMEVS0508OM/D

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.

HREG Set H Register HREG

NOTE: *This command is for the MMEVS08 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 HX

Set H:X Index Register

NOTE: *This command is for the MMEVS08 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.

Set/Clear I Bit

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 logfile   Start logging. Write to file logfile (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 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 .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.

MMEVS0508OM/D

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.

MMEVS0508OM/D

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>*]

MMEVS0508OM/D

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.

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 M68MMPFB0508 platform board can supply an oscillator clock source for the MCU's OSC1 input.

For the MMEVS05, four internally generated clock frequencies are available: 8 MHz, 4 MHz, 2 MHz, and 1 MHz. Entering emulator clock (OSC) command without the designated frequency brings up the temporary MMEVS 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 MMEVS08, five internally generated clock frequencies are available: 16 MHz, 8 MHz, 4 MHz, 2 MHz, and 1 MHz. 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>*]

Where:

   <rate>   8, 4, 2, or 1.

Examples:

   >OSC 4   Use the 4-MHz internal emulator clock.

   >OSC   Bring up the emulator clock window. The current
             oscillator setting will be highlighted.

PC PC

Set Program Counter

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 REG

Display Registers

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 MMEVS 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 MMEVS RESETIN state.

RESETOUT   Reset Output Enable   RESETOUT

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

Entering this command toggles the MMEVS 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 MMEVS RESETOUT state.

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 MMEVS 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   Display all drive B files that have the extension .xyz then execute the selected file.

MMEVS0508OM/D

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 10**. 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 10. Custom Map Window

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 MMEVS 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 MMEVS from DOS, enter EXIT at the DOS prompt.

MMEVS 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.

MMEVS0508OM/D

SHOWMEM   Display Memory Map   SHOWMEM

The SHOWMEM command displays only the MMEVS 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.

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 MMEVS 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 5** 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 ST

Single Step (Trace)

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 11** 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 11. 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.

MMEVS0508OM/D

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.

MMEVS0508OM/D

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.

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.

V

Set/Clear V Bit

V

> ***NOTE:*** *This command is for the MMEVS08 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.

MMEVS0508OM/D

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 <*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

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

MMEVS0508OM/D

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.

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 MMEVS 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.

MMEVS0508OM/D

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 Z

Set/Clear Z Bit

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.

Syntax:

ZOOM

Example:

>ZOOM   Resize the source window.

S-Record Information

Contents

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 A-2**.

Table A-2. 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 MMEVS 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 MMEVS.

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
S113002041E900084#42234300182342000824A952
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 A-3**.

Table A-3. 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 A-4**.

Table A-4. 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	
	28	5F		BHCC	$0161
	24	5F		BCC	$0163
	22	12		BHI	$0118
	22	6A		BHI	$0172
	00	04	24	BRSET	0, $04, $012F
	29	00		BHCS	$010D
	08	23	7	BRSET	4, $23, $018C

Checksum	2A	Checksum of the first S1 record.

The 16 character pairs shown in the code/data field of **Table A-4** 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 A-5**.

Table A-5. 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 A-6** 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 A-6. 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

MMEVS0508OM/D

MOTOROLA

CSIC

MICROCONTROLLERS

How to reach us:

MFAX: RMFAX0@email.sps.mot.com – TOUCHTONE (602) 244-6609

INTERNET: http://Design-NET.com

USA/EUROPE: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036. 1-800-441-2447

JAPAN: Nippon Motorola Ltd.; Tatsumi-SPD-JLDC, Toshikatsu Otsuki, 6F Seibu-Butsuryu-Center, 3-14-2 Tatsumi Koto-Ku, Tokyo 135, Japan. 03-3521-8315

HONG KONG: Motorola Semiconductors H.K. Ltd.; 8B Tai Ping Industrial Park, 51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852-26629298

Ⓜ **MOTOROLA**

MMEVS0508OM/D