E86MON™ Software

User's Manual

Order #21891B

AMD

E86MON™ Software User's Manual, Release 2.0

IF YOU HAVE QUESTIONS, WE'RE HERE TO HELP YOU.

The AMD customer service network includes U.S. offices, international offices, and a customer training center. Expert technical assistance is available from the AMD worldwide staff of field application engineers and factory support staff to answer E86™ and Comm86™ family hardware and software development questions.

Frequently accessed numbers are listed below. Additional contact information is listed on the back of this manual. AMD's WWW site lists the latest phone numbers.

Technical Support

Answers to technical questions are available online, through e-mail, and by telephone.

Go to AMD's home page at **www.amd.com** and follow the **Service** link for the latest AMD technical support phone numbers, software, and Frequently Asked Questions.

For technical support questions on all E86 and Comm86 products, send e-mail to **epd.support@amd.com** (in the US and Canada) or **euro.tech@amd.com** (in Europe and the UK).

You can also call the AMD Corporate Applications Hotline at:

(800) 222-9323	Toll-free for U.S. and Canada
44-(0) 1276-803-299	U.K. and Europe hotline

WWW Support

For specific information on E86 and Comm86 products, access the AMD home page at **www.amd.com** and follow the **Embedded Processors** link. These pages provide information on upcoming product releases, overviews of existing products, information on product support and tools, and a list of technical documentation. Support tools include online benchmarking tools and CodeKit™ software—tested source code example applications. Many of the technical documents are available online in PDF form.

Questions, requests, and input concerning AMD's WWW pages can be sent via e-mail to **webmaster@amd.com**.

Documentation and Literature Support

Data books, user's manuals, data sheets, application notes, and product CDs are free with a simple phone call. Internationally, contact your local AMD sales office for product literature.

To order literature, call:

(800) 222-9323	Toll-free for U.S. and Canada
(512) 602-5651	Direct dial worldwide
(512) 602-7639	Fax

Third-Party Support

AMD FusionE86^{SM} partners provide an array of products designed to meet critical time-to-market needs. Products and solutions available include emulators, hardware and software debuggers, board-level products, and software development tools, among others. The WWW site and the *E86 Family Products Development Tools CD*, order #21058, describe these solutions. In addition, mature development tools and applications for the x86 platform are widely available in the general marketplace.

2.0

Contents

About the E86MON™ Software

Chapter 1

Using the E86MON Software

Chapter 2

Downloading Files

2.0

E86MON Software Commands

2.0

Appendix A

Notes on Terminal Emulators

Appendix B

Utilities Included With the E86MON Software

Appendix C

Error Messages

Appendix D

DOS Emulation Support

Appendix E

Porting E86MON Software to Other Environments

Index

List of Figures

List of Tables

2.0

About the E86MON™ Software

The E86MON™ software is a utility provided with the Am186™ and Am188™ family of boards that enables you to download and run your application code. Currently, this family of boards includes the Net186, SD186ED, SD186EM, SD186ER, SD186ES, SD188EM, SD188ER, and SD188ES demonstration boards, as well as the Am186CC/CH/CU Microcontrollers Customer Development Platform (CDP).

The E86MON software supports rudimentary debugging. Programs can be loaded and run from RAM, or the E86MON software can program the on-board Flash memory with the application code and execute the application from Flash memory.

The E86MON software is designed to demonstrate the capabilities of the Am186™ and Am188™ devices. For a more powerful development environment, AMD *strongly* recommends that developers search our *E86 Family Products Development Tools CD*, order #21058 for compilers, linker/locators, real-time operating systems, complete DOS emulation, source-level debuggers, and other powerful tools provided by our Fusion Partners.

The E86MON software running on an Am186 or Am188 family board enables you to perform quick prototyping and benchmarking of simple algorithms, before investing in x86 development tools. The limited DOS emulator allows you to download and run small .EXE files, which can be developed and tested using standard compilers on a PC running DOS. Full source code for the E86MON software is provided as a teaching and demonstration aid.

2.0

E86MON Software Features

The E86MON software provides features enabling you to perform the following functions:

- Download programs to RAM or Flash memory

- Dump memory

- Enter memory

- Search memory

- Display registers

- Modify CPU registers

- Set breakpoints

- Trace programs

- Peripheral input/output

E86MON Software Documentation

This user's manual provides information on using the E86MON software with the Am186 and Am188 family of boards. These boards already have the E86MON software installed in on-board Flash memory.

If using a version of the E86MON software earlier than Version 3.3.x, AMD strongly recommends upgrading to the most recent version of the E86MON software by using the files supplied on the 3.5-inch disk included in the board kit. For information about upgrading the E86MON software, see "Upgrading the E86MON Software" on page 2-5. The most recent version of the E86MON software is also available via the AMD web site.

About This Manual

Chapter 1, "Using the E86MON Software" provides detailed startup information and a description of the E86MON software architecture.

Chapter 2, "Downloading Files" provides instructions for downloading .HEX and .EXE files from a PC.

Chapter 3, "E86MON Software Commands" contains a description of the E86MON software syntax followed by detailed descriptions of the E86MON software commands in an alphabetical listing.

Appendix A, "Notes on Terminal Emulators" briefly describes how to use the E86MON software with various terminal emulators including DOS PROCOMM PLUS and the HyperTerm program provided with Windows95.

Appendix B, "Utilities Included With the E86MON Software" lists and describes the utilities included with the E86MON software.

Appendix C, "Error Messages" provides an alphabetical listing and detailed description of the error messages generated by the E86MON software.

Appendix D, "DOS Emulation Support" provides a description of the DOS emulation environment supported by the E86MON software.

Appendix E, "Porting E86MON Software to Other Environments" describes how to port the E86MON software to customer hardware.

A standard index is also included.

2.0

Suggested Reference Material

The following AMD documentation may be of interest to the E86MON software user.

- *Am186™ and Am188™ Family Instruction Set Manual*, order #21267
- *Am186™CC Communications Controller Data Sheet*, order #21915
- *Am186™CH HDLC Microcontroller Data Sheet*, order #22024
- *Am186™CU USB Microcontroller Data Sheet*, order #22025
- *Am186™CC/CH/CU Microcontrollers Register Set Manual*, order #21916
- *Am186™CC/CH/CU Microcontrollers User's Manual*, order #21914
- *Am186™ED Microcontrollers Data Sheet*, order #21336
- *Am186™ED Microcontrollers User's Manual*, order #21335
- *Am186™EM and Am188™EM Microcontrollers Data Sheet*, order #19168
- *Am186™EM/EMLV and Am188™EM/EMLV Microcontrollers User's Manual*, order #19713
- *Am186™ER and Am188™ER Microcontrollers Data Sheet*, order #20732
- *Am186™ER and Am188™ER Microcontrollers User's Manual*, order #21684
- *Am186™ES/ESLV and Am188™ES/ESLV Microcontrollers Data Sheet*, order #20002
- *Am186™ES and Am188™ES Microcontrollers User's Manual*, order #21096
- *E86™ Family Products Development Tools CD*, order #21058

For current application notes and technical bulletins, see our World Wide Web page at **www.amd.com**.

2.0

Documentation Conventions

The *E86MON™ Software User's Manual* uses the notational conventions shown in Table 0-1 (unless otherwise noted).

Table 0-1. Notational Conventions

Symbol	Usage
Boldface	Indicates that characters must be entered exactly as shown, except that the alphabetic case is only significant when indicated.
Italic	Indicates a descriptive term to be replaced with a user-specified term.
`Typewriter face`	Indicates computer text input or output in an example or listing.
`.EXE`	Indicates a DOS executable file.
`.HEX`	Indicates an Intel extended hex file.
`<>`	Encloses a required parameter. To include the information described within the angle brackets, type only the parameters, not the angle brackets themselves.
`[]`	Encloses an optional parameter. To include the information described within the brackets, type only the parameter, not the brackets themselves.
`|`	Separates alternate choices in a list. Only one of the choices can be entered.

2.0

2.0

Chapter 1

Using the E86MON Software

This chapter provides information on using the E86MON software, as well as a description of the E86MON software architecture.

Note that the information in this manual is based on using an Am186CC/CH/CU Microcontroller Customer Development Platform with Version 3.3.2 of the E86MON software. Your screens may vary slightly from the ones shown in this manual.

The information in this chapter applies to boards operating at factory default settings and assumes that you have completed the board installation procedures listed in the "Quick Start" chapter of the applicable board user's manual.

If you have not completed the procedures in the board user's manual, you should perform the procedures described in "Getting Started" on page 1-2 in this manual. If you have completed the "Quick Start" procedures, turn to page 1-3.

2.0

Getting Started

NOTE: Versions of the E86MON software prior to 3.0 require you to press the Break key, rather than type an **a** as described in step 2.

1. Invoke the terminal emulation program at 19200 baud, no parity, 8 data bits, and 1 stop bit; enable the software flow control (Xon/Xoff), if supported. For more information about terminal emulators, see Appendix A, "Notes on Terminal Emulators".

2. Reset the board by depressing and releasing the RESET switch on the board. If the board has LEDs or a TIP attached with LEDs, the LEDs will light up as they did on power-up.

   During the first three seconds, type an **a** in the terminal window to ensure that the E86MON software uses the correct baud rate. When the E86MON software receives an **a**, it adjusts its baud rate (if necessary) and displays the welcome message and prompt.

   If you type a character other than an **a**, or type no character at all, the E86MON software still displays the welcome message and prompt, but may be using an incorrect baud rate. Depressing and releasing the RESET switch gives you another opportunity to type an **a**.

Before continuing in this chapter, you should make sure you see the E86MON welcome screen shown in Figure 1-1 on your PC monitor.

```
Welcome to AMD's E86MON!      (? <Enter> for help)
cc86mon:
```

Figure 1-1. E86MON Software Welcome Screen

When the E86MON software displays the welcome screen and prompt, it also changes the LED pattern. Previous versions of the E86MON software simply stop updating the LED display; Versions 3.11 and above change the display to one of eight alternating patterns the E86MON software uses when it is waiting for keyboard input.

To display the version number and available commands, type **?** and press Enter. To load the monitor extensions, use the **LL** command.

If the version number shown at the top of the help screen is less than 3.3.x, AMD strongly recommends you upgrade to the most recent version of the E86MON software by using the files supplied on the 3.5-inch disk included in the board kit. For detailed information on upgrading your E86MON software, see "Upgrading the E86MON Software" on page 2-5.

The most recent version of the E86MON software is also available via the AMD web site.

E86MON Software

After the Am186 or Am188 family board is powered on or reset, the E86MON software enters a three-second delay loop waiting for you to type an **a**. If an **a** is not detected on an asynchronous serial port within the three-second delay, the E86MON software does one of three things, as described below:

- If you have installed a startup program in the Flash memory, *and* the startup vector at F7FF:0000 has been filled in with a far jump, the E86MON software uses the startup vector to jump to your program.

- Otherwise, if you have used the **W** command to store a DOS .EXE program in the Flash memory and used the **P AutoRun** command to mark it for running at startup, then the E86MON software executes that DOS program.

- If the startup vector at F7FF:0000 has not been filled in and the **P AutoRun** command has not been used, then after the three-second delay loop, the E86MON software displays the welcome screen and prompt, but must assume the baud rate. On SD186ED, SD186ES, and SD188ES demonstration boards, and on the Am186CC/CH/CU Microcontrollers Customer Development Platform, the E86MON software must also assume to which serial port the terminal is connected. If the baud rate or serial port does not match that of the terminal, you see nothing, or you see garbled characters. The monitor also loads any extensions stored in Flash memory.

  You can change the default baud rate and serial port by using the **P** command. See Chapter 3, "E86MON Software Commands" for a description of the E86MON commands.

The automatic baud rate detection feature is useful in the following circumstances:

• When a user program is installed, but you want to invoke the monitor instead

• When the programmed baud rate does not match the terminal baud rate

• When the serial port does not match the serial port the terminal is plugged into

• When the programmed CPU speed does not match the actual CPU speed (the bit clock is divided down from the CPU clock)

• When you do not want to wait three seconds for the monitor to boot

The automatic baud rate detection is designed to detect baud rates from 2400 to 115200, but its success depends on the CPU type and speed. At the default CPU frequency of 40 MHz, Am186CC/CH/CU Microcontroller CDP, SD186, and Net186 boards can reliably detect baud rates up to 115200; SD188 demonstration boards can reliably detect baud rates up to 57600.

Note that when the autobaud feature of the monitor is used, monitor extensions must be loaded manually with the **LL** command.

You can use the **P** command to set the default baud rate. This is especially useful when the baud rate is so high relative to the CPU type and frequency that the automatic baud rate detection is unreliable.

2.0

Architecture

The E86MON software remains resident in the upper sectors of the Flash memory. The user application resides in the sectors below the E86MON software. The E86MON software provides debug and download functionality while creating a virtual reset vector 32K below 1 Mbyte. (Note that some flash devices require the reset vector to be lower in memory due to their sector size. For example, the Am29F040 has a fixed sector size of 64K, so the reset vector must be located 64K below 1 Mbyte.)

The processor begins fetching instructions after reset at FFFF0h, or 16 bytes below the top of memory (see Figure 1-2). A user application can be positioned to start from the virtual reset (e.g., located so that the program's boot code resides in the 16 bytes immediately below the monitor, at F7FF0h).

Monitor Reset Vector	← FFFFFh / ← FFFF0h
Permanent Variable Storage	
Monitor Program	
User Application Start Vector	← F8000h / ← F7FF0h
User Application/Monitor Upgrade	
	← F0000h
Monitor Extensions	← Top of Protect-Flash range* / ← E0000h
User Application	← Bottom of Flash memory

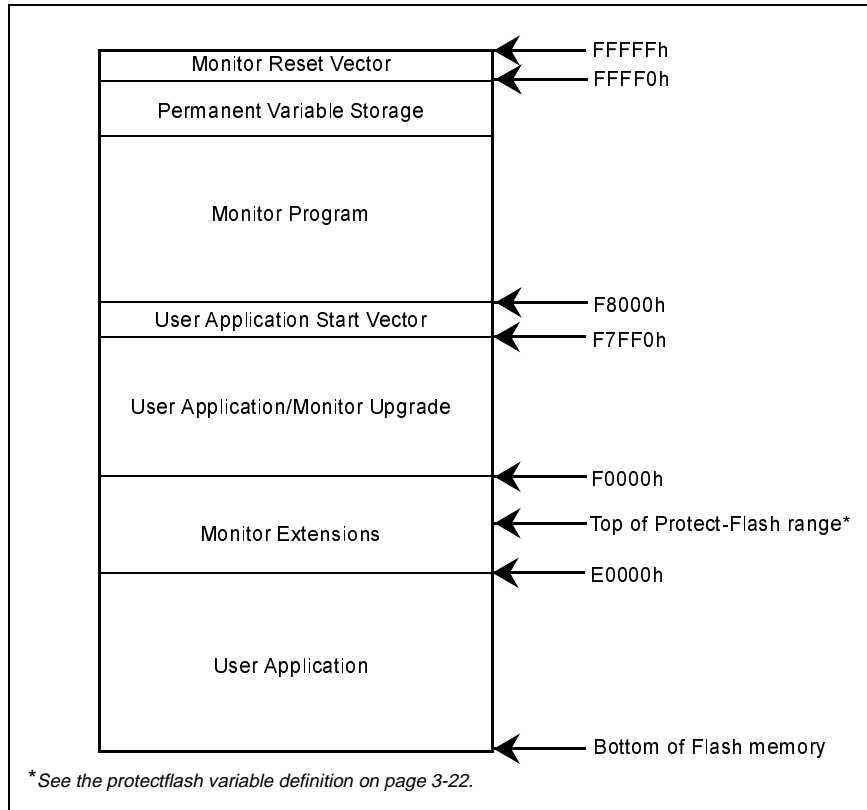*See the protectflash variable definition on page 3-22.

Figure 1-2. Flash Memory Format

The vector table occupies RAM from 0h to 3FFh. The E86MON software uses the vectors shown in Table 1-1. All other interrupt vectors may be utilized by the user application.

Table 1-1. E86MON Software Vector Description

Vector	Purpose
01h	Single-step interrupt
03h	Breakpoint interrupt
08h	Reserved for future use by E86MON
11h	Serial port 1 interrupt
12h	Timer 1 interrupt
14h	Serial port 0 interrupt
20h	DOS terminate interrupt
21h	DOS service interrupt
22h-2Fh	Reserved for future use by E86MON

2.0

E86MON reserves RAM from 400h to 40Fh for internal use, and the top 4096 bytes of RAM for data and stack. See the corresponding board user's manual to determine this location. The RAM user area (410h to the bottom of the stack) is used to store the downloaded program. The format of the RAM memory is illustrated in Figure 1-3.
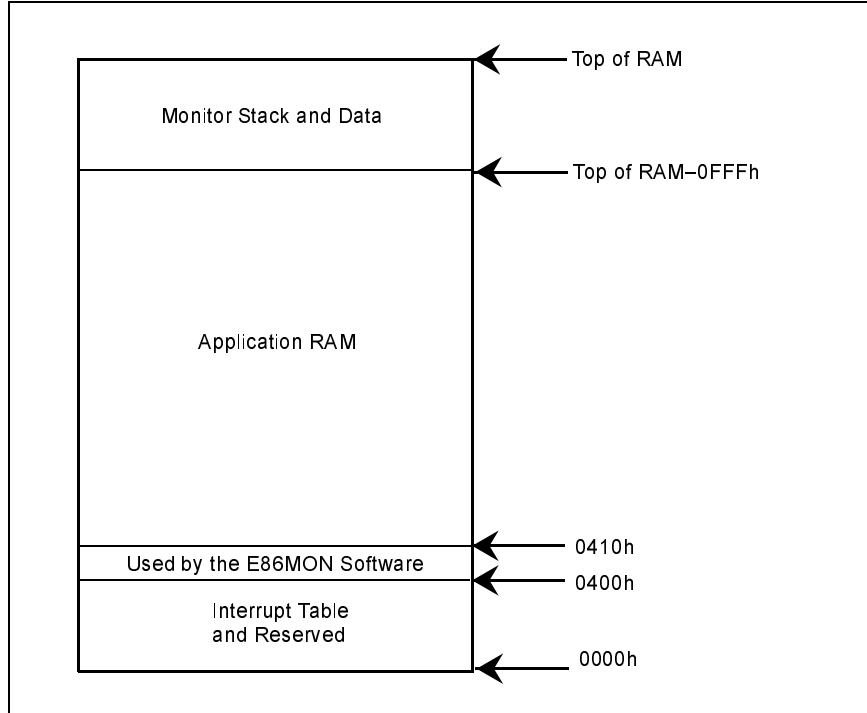
Figure 1-3. RAM Memory Format

The Flash memory is accessed using the Upper Chip Select ($\overline{\text{UCS}}$) signal, and the RAM is accessed using the Lower Chip Select ($\overline{\text{LCS}}$) signal. Both the $\overline{\text{UCS}}$ and $\overline{\text{LCS}}$ signals are programmed by the monitor to operate with zero wait states by default. The **P WaitStates** command can be used to set LCS DRAM wait states on boards populated with an Am186CC/CH/CU or Am186ED microcontroller device. The format of the RAM memory is illustrated in Figure 1-3.

The Peripheral Control Block (PCB) is left in the default I/O space (FF00h). You should add the peripheral register offset to FF00h to access the peripherals.

For More Information...

If you need more information about:

- Board hardware
  See the Functional Description chapter in the corresponding board user's manual.

- Problems with the board or the E86MON software
  See Appendix C, "Error Messages".

- The Am186 or Am188 family of microcontrollers
  See the appropriate Am186 and Am188 microcontroller user's manual or data sheet.

2.0

Chapter 2

Downloading Files

This chapter explains how to load files from a host PC to the evaluation board, and how to upgrade your software.

- If you want to download .HEX files, see page 2-2.

- If you want to download .EXE files, see page 2-4.

- If you want to upgrade your E86MON software, see page 2-5.

Downloading .HEX Files

There are four example .HEX files you can download. The files are located in the \OUT subdirectory on the floppy disk that was provided in the board kit.

`AMDDHRY.HEX`	A RAM-based DHRYSTONE benchmark
`SECONDS.HEX`	A RAM-based seconds counter
`TESTMON.HEX`	A RAM-based minimal DOS emulator test
`LEDS.HEX`	A Flash-memory-based demonstration

The source code for all these files is under the \SAMPLES subdirectory. The E86MON software supports the downloading of Intel extended .HEX files into RAM or Flash memory. The .HEX file should contain type 2 extended address records, which specify the load address in the 1 Mbyte address range. The last record in the file should be a type 1 EOF record.

A file being downloaded to RAM for execution should be located between 410h and the start of the monitor data at the end of the RAM (see Figure 1-3 on page 1-7). A file being downloaded to Flash memory for execution should be located between the start of the Flash memory and E0000h (see Figure 1-2 on page 1-5). The **I** command shows the size and location of the free RAM, as well as information about the size and location of the Flash memory.

It is not permissible for the file to have some sections download to RAM and other sections download to Flash memory because the E86MON software relocates itself to RAM (destroying any loaded RAM program) to burn the Flash when it is downloading to Flash memory. The E86MON software reports a range error on the download of such a file.

If downloading into Flash memory, you should first make sure the target download area is empty by using the **X** command to erase the Flash sectors. Unless you are storing multiple programs into Flash, the easiest way to do this is to use **XA** to erase all the application sectors. Note that if sectors are protected with the protectflash variable, the **XA** command does not erase them (see page 3-22). However, they can be erased with the **X** *number* command, where *number* is the sector number.

If your application requires DOS int21 support, make sure the monitor extensions are loaded before downloading the application (see the **LL** command).

2.0

There is no specific command to download .HEX files. Simply start transferring with the terminal program in ASCII or raw ASCII mode. The E86MON software echoes the first record as it receives it. When the E86MON software parses the record and determines that it is a .HEX file record, the E86MON software switches into a file transfer mode. The type 1 EOF record at the end of the file switches the E86MON software back to command mode.

While downloading, the E86MON software displays one dot per eight lines of the .HEX file being transferred.

If an error is encountered during the download, an error message is printed, and the E86MON software stays in download mode until it receives an Escape character (1Bh). After receiving an Escape character, the E86MON software prints a more detailed error message, then returns to command mode. For more information on the E86MON error messages, see Appendix C, "Error Messages".

If you are downloading to Flash, and the .HEX file contains a type 3 start address record, and the application code vector at F7FF0 has not been programmed, the E86MON software automatically programs a far jump to the start address into this vector. On subsequent reset sequences, if you do not type an **a** in the first three seconds (forcing entry to the E86MON software), the E86MON software notices there is a valid far jump at F7FF0, and vectors there to automatically start up the application program.

The MAKEHEX utility with Version 3.3.x of the monitor has changed from previous versions. Hex files built with the older (3.2.x) MAKEHEX utility do not download with the Version 3.3.x monitor. Additionally, the 3.3.x MAKEHEX files do not execute on a 3.2.x monitor. See Appendix C, "Error Messages".

2.0

Downloading .EXE Files

The E86MON software can download and run DOS executable files, enabling you to use affordable, readily available, and familiar PC-based compilers and assemblers to develop initial test and benchmarking code. The E86MON software provides a minimal subset of DOS interrupt 21h functionality, which is fully described in Appendix D, "DOS Emulation Support".

Most compilers are capable of generating .EXE files that work within this DOS environment, as long as you do not use library functions requiring file-based I/O. Unlike some prior versions of the E86MON software, Versions 3 and later do not support direct downloading of .EXE files. Instead, the E86MON software supports AMD extensions to the Intel extended .HEX file format, and includes a conversion program that converts .EXE files into this extended .HEX file format.

To convert the .EXE file into a .HEX file, use the MAKEHEX utility in the \OUT subdirectory supplied on the floppy disk included in the board kit. For example, to convert SAMPLE.EXE into SAMPLE.HEX, simply type **MAKEHEX SAMPLE** (assuming MAKEHEX.EXE is in the path).

When you have converted the .EXE file, it can be downloaded to the E86MON software as described in "Downloading .HEX Files" on page 2-2. After downloading the file, you can set parameters for the program (if it expects a command line) with the **N** command, and then start execution with the **G** command.

Alternatively, you can use the **W** command before downloading the file, to program it into Flash. Because Flash is nonvolatile, the program can then be run multiple times (even after power has been cycled) with the **L** command.

2.0

Upgrading the E86MON Software

Files used to upgrade the E86MON software are provided in the \OUT subdirectory on the floppy disk included in the board kit. The latest version of the E86MON software is also available on the web.

The files listed below are in the \OUT subdirectory on the floppy disk for all boards but the SD186ER:

EMON332U.HEX	Main upgrade file
EMON332R.HEX	RAM-based version of the E86MON software
EMON332B.HEX	Bootable (suitable for programming with a PROM programmer) version of the E86MON software
EXTEM332.HEX	Extensions to provide support for a DOS emulator (Int21), command-line editor (additional LED patterns), and help

NOTE: The 332 portion of the filename represents Version 3.3.2. The version included on your floppy disk may vary.

For the SD186ER board, the files are:

EMNR332U.HEX	Main upgrade file
EMNR332R.HEX	RAM-based version of the E86MON software
EMNR332B.HEX	Bootable (suitable for programming with a PROM programmer) version of the E86MON software
EXTEM332.HEX	Extensions to provide support for a DOS emulator (Int21), command-line editor (additional LED patterns), and help

To upgrade versions 3 or later of E86MON (prior versions must be upgraded twice—first to Version 3, then to Version 3.3.x):

1. Use the **XA** command to erase the application Flash sectors.

2. Download EMON332U.HEX, the upgrade file, to the board. It is not necessary to type any command to do this; the new E86MON software automatically recognizes a file download when it detects the colon that starts the file.

3. Use the **G** command to go to the new E86MON software, which is running from user Flash memory space.

2.0

4. Type an **a** to establish communication with the new monitor. You are now running the E86MON software from the copy of the E86MON software in the application area of the Flash memory. This can be verified with the **I** command.

5. Type a **Z** and press <Enter> to initiate the upgrade. The software asks if this is really what you want to do. Answer **Y** to perform the upgrade, but do not do this if your power is not stable. If the upgrade is aborted before it finishes, you may need to send the board back to AMD to have the Flash reprogrammed.

6. The E86MON software is now upgraded and automatically runs out of the new boot copy of the software. Type an **a** within three seconds to establish communication with the boot copy of the E86MON software.

7. You can now use the **XA** command to remove the application copy of the E86MON software, and then download any desired .HEX file to the application area of the Flash memory.

2.0

Chapter 3

E86MON Software Commands

This chapter provides a description of the E86MON software syntax followed by detailed descriptions of the E86MON software commands in an alphabetical listing. The prompt displayed in the examples assumes a board populated with an Am186™CC/CH/CU microcontroller. The prompt you see may vary slightly.

Note that although uppercase letters are used for all of the commands, you can use either lower case or upper case.

2.0

E86MON Software Syntax

Table 3-1 shows the command parameters most commonly used by the E86MON software.

Table 3-1.   E86MON Software Syntax

Parameter	Definition
Byte	1 or 2 hexadecimal digits
Word	1–4 hexadecimal digits
Decimal	1–9 hexadecimal digits
Address	Can be entered in typical x86 segment:offset format (e.g., F800:0) to refer to the base of the monitor, or a linear address can be entered in 5 hexadecimal digits (e.g., F8000). If the linear address is used, the E86MON software interprets the first 4 digits as the segment, and the last digit as the offset. Most commands that do not alter memory also support short addresses. A short address is an address that specifies only the offset (1–4 hexadecimal digits). The current value of the DS register is implicitly used for the segment. Commands that alter memory require you to specify the full address. Addresses can contain simple math, for example, DS:BX*2+10. Math operators supported include *, +, and –. If multiplication is used, it must be the first operator.

2.0

Table 3-1.   E86MON Software Syntax (Continued)

Parameter	Definition
Range	An address range can be specified in two ways: as *<address>* *<space>* *<address>*, specifying the address of the start of the range and the address immediately after the end of the range, or as *<address>* **L** *<length>*, which explicitly specifies the address of the start of the range and the length of the range. The commands listed below are interpreted the same way and dump 1024 bytes beginning at 16K: D 400:0 400:400 D0:4000 400:400 D 04000 L 400 d040000l400 As the last command shows, spaces only matter where the E86MON software would have trouble distinguishing the end of one number from the beginning of the next one, and all commands can be entered in upper or lower case.
List	A list is a collection of bytes. Each byte can be specified with 1 or 2 hexadecimal digits, with the bytes separated by spaces, and ASCII data can be specified in single or double quotes. The following command places an ASCII string, complete with carriage return and two line feeds, at 16K: E 04000 "This is a quoted string" 0D A,0A Note that other than the mandatory 5 digits for a linear address, numbers do not require leading zeroes. Also note that commas are optional. They can be used instead of, or with, spaces.

2.0

E86MON Software Commands

: – Begin Download

Begin download (:) is not actually a command. The colon character (**:**) is the first character transmitted as part of an Intel extended .HEX file.

For more information about downloading files, see Chapter 2, "Downloading Files".

; – Comment

Comment does not perform any action. It can be used for commenting scripts used with the E86MON software.

2.0

<Break> – Break Key

When the E86MON software receives an RS232 break (usually invoked by pressing Alt-B or Ctrl-Break on the terminal emulator), it will break into the debugger. This is useful in some cases when an application appears "hung" because you can find out where it is executing. However, note that <Break> can also be used to debug the E86MON software itself, so you should be careful how many times you press <Break> without pressing **G** to continue program execution. Too many breaks cause a stack overflow within the E86MON software.

2.0

B – Set Breakpoint

Syntax

B   *<address>*

Description

The **Set Breakpoint (B)** command sets a breakpoint at *address* by overwriting the opcode with the breakpoint instruction (INT 3).

```
cc86mon: B 0410:0000
```

After you enter the Go (**G**) command and the breakpoint instruction is reached, the message, *Stopped at breakpoint*, and the current register values are displayed:

```
cc86mon: g 410:0000
Stopped at breakpoint
AX=0000  BX=0000  CX=0000  DX=0000  SP=0400  BP=0000  SI=0000  DI=0000
DS=0041  ES=0041  SS=7F51  CS=0410  IP=0000   NV UP EI PL ZR NA PE NC
cc86mon:
```

The opcode is restored when the processor encounters the INT 3 and re-enters the E86MON software. At this point, you can set another breakpoint, trace execution using the Trace (**T**) command, or resume execution using the Go (**G**) command.

NOTE: The **B** command is not supported in Flash memory space.

Note that only one breakpoint is active at a time — setting one removes any previous breakpoint.

C – Compare Memory

Syntax

```
C <range> <address>
```

Description

The **Compare Memory (C)** command compares the contents of memory located in an address *range* to the contents of memory beginning at an *address.* Only bytes which are different are shown.

```
cc86mon: c 410:0 410:4 410:4
0410:0000  01  00  0410:0004
0410:0002  01  00  0410:0006
0410:0003  97  94  0410:0007
cc86mon:
```

2.0

D – Dump Memory

Syntax

```
D [range|address]
```

Description

The **Dump Memory (D)** command displays the Am186CC/CH/CU microcontroller customer development platform's memory contents at *address*:

```
cc86mon: d 0 11f
0041:0000   61 62 63 22 64 65 66 88  ff 7f e5 2c f7 ff d7 9c   abc"def....,....
0041:0010   dd bb c5 3b f7 fb d7 7b  dd bb c5 3b f7 fb d7       ...;...{...;...
cc86mon:
```

If a range is specified, that range is displayed. If only an address is specified, the dump begins at that address and continues for 128 bytes. If no address is specified, the **D** command dumps 128 bytes beginning where the most recent dump command finished.

```
cc86mon: d
0041:0010                                              7b                    {
0041:0020   ff 7f e5 2c f7 ff c7 8d  ff 7f e7 2c 44 00 c7 bc   ...,.......,D...
0041:0030   2f bf 2f 1e f7 ed f6 24  bf bf 2f 9f f7 ed f7 e4   /./....$../.....
0041:0040   f7 d7 f6 d7 ff bf 7b b7  ff f7 f6 d7 ff bf 7b bf   ......{.......{.
0041:0050   af bf 2f 9f f7 ff f7 a4  2f bf 2f 9f f7 ed f7 a4   ../....././.....
0041:0060   ff f7 f6 d7 ff bf 7b bf  f7 f7 f6 d7 ff bf 7b bf   ......{.......{.
0041:0070   dd bb c5 3b f7 fb d7 7b  dd bb c5 3b f7 fb c7 7b   ...;...{...;...{
0041:0080   33 20 55 73 65 20 74 68  65 20 27 4e 27 20 63 6f   3 Use the 'N' co
0041:0090   6d 6d 61 6e 64 20 74 6f  20 73 65 74 20 63 6f       mmand to set co
cc86mon:
```

E – Enter Memory

Syntax

```
E  <address> [list]
```

Description

The **Enter Memory (E)** command alters a memory location or locations beginning at *address*.

```
cc86mon: d 41:0 L20
0041:0000   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
0041:0010   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
cc86mon: e 41:0 "Four score and seven years ago" 0d
cc86mon: d 41:0 L20
0041:0000   46 6f 75 72 20 73 63 6f  72 65 20 61 6e 64 20 73  Four score and s
0041:0010   65 76 65 6e 20 79 65 61  72 73 20 61 67 6f 0d 00  even years ago..
cc86mon:
```

If a list is not provided, you are prompted with the current value at that location:

```
cc86mon: e 41:0
0041:0000   46   "S"
```

You can enter a new value for that location, hit return to accept the current value, or type a period (**.**) to exit back to the main menu. Subsequent memory locations are displayed for possible alteration until you exit by typing a period (**.**) or by pressing the Escape key:

```
cc86mon: e 41:0
0041:0000   53  "Six "
0041:0004   20  3f
0041:0005   73  .
cc86mon:
cc86mon: d 41:0 41:20
0041:0000   53 69 78 20 3f 73 63 6f  72 65 20 61 6e 64 20 73  Six ?score and s
0041:0010   65 76 65 6e 20 79 65 61  72 73 20 61 67 6f 0d 00  even years ago..
cc86mon:
```

2.0

F – Fill Memory

Syntax

F *<range>* *<list>*

Description

The **Fill Memory (F)** command fills the contents of memory located in an address *range* with the *list* of hex bytes or quoted characters.

```
cc86mon: f 41:0 L80 "*-Empty.Memory-*"
cc86mon: d 41:0 41:80
0041:0000   2a 2d 45 6d 70 74 79 2e  4d 65 6d 6f 72 79 2d 2a   *-Empty.Memory-*
0041:0010   2a 2d 45 6d 70 74 79 2e  4d 65 6d 6f 72 79 2d 2a   *-Empty.Memory-*
0041:0020   2a 2d 45 6d 70 74 79 2e  4d 65 6d 6f 72 79 2d 2a   *-Empty.Memory-*
0041:0030   2a 2d 45 6d 70 74 79 2e  4d 65 6d 6f 72 79 2d 2a   *-Empty.Memory-*
0041:0040   2a 2d 45 6d 70 74 79 2e  4d 65 6d 6f 72 79 2d 2a   *-Empty.Memory-*
0041:0050   2a 2d 45 6d 70 74 79 2e  4d 65 6d 6f 72 79 2d 2a   *-Empty.Memory-*
0041:0060   2a 2d 45 6d 70 74 79 2e  4d 65 6d 6f 72 79 2d 2a   *-Empty.Memory-*
0041:0070   2a 2d 45 6d 70 74 79 2e  4d 65 6d 6f 72 79 2d 2a   *-Empty.Memory-*
cc86mon:
```

The list is replicated as many times as it takes to fill the range. The size of the list does not need to fit evenly in the range; the last copy of the list is truncated to fit.

2.0

G – Go to Address

Syntax
```
G [[=]address]
```

Description
The **Go to Address (G)** command transfers execution control to a user program, which runs until it encounters a breakpoint or trace interrupt, you press the Break key, or the program exits via a DOS call.

If an address is given on the command line, that address is loaded into the CS:IP register pair before transferring control. If a .HEX file containing a start address record is downloaded, CS:IP is set from that record, so you do not need to explicitly specify the address on the command line.

H – Hex Math or Help

Syntax

H [*word word*]

Description

The **Hex Math (H** *word word***)** command performs five hexadecimal math operations on the two hex *words* given:

```
cc86mon: h 3f7 03a
3F7+3A=431, 3F7-3A=3BD, 3F7*3A=E5F6
3F7/3A=11, 3F7%3A=1D
cc86mon:
```

The **Help (H)** command (without two hex words) displays the help menu:

```
E86 Boot Monitor -- Version 3.32  1998/07/22
                     Copyright (C) 1994-1998 AMD, Austin, Texas, USA
Commands:
  B addr        Set breakpoint    M range address     Move memory
  C range addr  Compare memory    N [arglist]         Name .exe arguments
  D [range]     Dump memory       O[W] word byte|word Output[word]
  E addr [list] Enter memory      P [varname decimal] Boot Parameters
  F range list  Fill memory       R [reg name or '?'] Display/alter regs
  G [[=]addr]   Go to address     S range list        Search for string
  H [word word] Hex Math or help  T [=address] [word] Trace 'n' steps
  I             Info about system W [mnemonic name]   Write .EXE to flash
  I[W] word     Input[word]       X sectnum|A         Exterminate flash
  J             Autobaud again    Z                   Upgrade monitor
  L[G][decimal] Load RAM from ROM :                   Begins download
  LL            Load Library      ;                   Comment

byte == 1-2 hex digits
word == 1-4 hex digits
addr == word:word or 5 hex digits for absolute address
range == addr addr  OR  addr L length
list  == list of hex bytes or quoted characters
decimal == 1-9 decimal digits

cc86mon:
```

NOTE: Before you can use the **H** command, the monitor extensions must be loaded (see the **LL** command on page 3-17).

I – Information About System

Syntax

```
I
```

Description

The **Information About System (I)** command displays information specific to the evaluation board running the E86MON software:

```
cc86mon: i

  Module  Code Segment/Length   Data Segment/Length
   EXTEMON         E000    21C0         37C0    03F0
   E86MON          37FF    773C         3F75    08B0

Free data paragraphs: 377e
Current system time:  1442.051

Flash device:  29F800T -- 512K bytes organized as 256K X 16
App sectors:   0 at 8000, 1 at 9000, 2 at A000, 3 at B000,
               4 at C000, 5 at D000, 6 at E000, 7 at F000

Boot sectors:  8 at F800, 9 at FA00, 10 at FC00

Memory block at 0041:0000, size 377E, owner 0041
cc86mon:
```

2.0

I – Input

Syntax

```
I [W] <word>
```

Description

The **Input (I)** command followed by a word inputs from a byte-wide port and displays the results. **IW** followed by a word inputs from a word-wide port and displays the results. This command also allows read word access to the processor's Peripheral Control Block, which is mapped to the upper portion of I/O space.

```
cc86mon: j
Change your terminal to desired baudrate, then press 'a'.

cc86mon:
```

2.0

J – Autobaud Again

Syntax
```
J
```

Description
The **Autobaud Again (J)** command causes the automatic baud rate detection to be invoked. When you have entered this command, you can change the terminal's baud rate. On the Am186CC/CH/CU, Am186ED, Am186ES, and Am188ES microcontrollers, you can also connect to the alternate serial port. When the terminal program is set up properly, type an **a** to re-establish connection with the monitor. Note that automatic baud rate detection may not be reliable at baud rates that are high relative to the CPU frequency and bus width. At a CPU frequency of 40 MHz, Am186CC/CH/CU, Net186, and SD186 boards can reliably detect baud rates up to 115200. SD188 demonstration boards can reliably detect baud rates up to 57600.

```
ed86mon: J
Change your terminal to desired baudrate, then press 'a'.
ed86mon:
```

2.0

L/LL – Load / Load Library

Syntax
```
L  [G] [decimal]
LL
```

Description
The **Load (L)** command loads a previously stored .EXE file from Flash memory to RAM. If no parameters are given, a list of currently stored programs is displayed. If a decimal number is given, the corresponding program is copied from Flash memory to RAM. Programs are stored to Flash memory using the **W** command, and can be made bootable with the **P AutoRun** command. The **LG** command is equivalent to the **L** command immediately followed by a **G** command (e.g., to load and run the program).

```
cc86mon: L
     1   8000:0000    seconds
cc86mon:L G 1
```

The **Load Library (LL)** command loads the extensions (EXTEM332.HEX) into memory. Loading extensions provides support for a DOS emulator (Int21), and command-line editor (additional LED patterns).

If the extensions are not loaded in flash, they can be downloaded using the EXTEM332.HEX file supplied on the E86MON software disk. See Chapter 2, "Downloading Files" for more information.

2.0

M – Move Memory

Syntax

M *<range>* *<address>*

Description

The **Move Memory (M)** command moves the contents of memory located in an address *range* to memory beginning at *address*.

```
cc86mon: d 41:0 L20
0041:0000   4d 6f 76 65 20 74 68 69  73 21 00 00 00 00 00 00   Move this!......
0041:0010   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
cc86mon: m 41:0 41:a 41:10
cc86mon: d 41:0 L20
0041:0000   4d 6f 76 65 20 74 68 69  73 21 00 00 00 00 00 00   Move this!......
0041:0010   4d 6f 76 65 20 74 68 69  73 21 00 00 00 00 00 00   Move this!......
cc86mon:
```

N – Name Arguments

Syntax

```
N   <ascii>
```

Description

The **Name Arguments (N)** command allows command-line arguments to be passed to loaded programs. The syntax is slightly different than DOS debug in that the actual program name should not be entered, only the arguments. AMD recommends that test programs do not rely on command-line arguments as it may be easy to forget the **N** command.

O – Output Word

Syntax

```
O <word> <byte>
O [W] <word> <byte>|<word>
```

Description

The **Output Word (O)** command outputs the second parameter (byte or word) to the port given in the first parameter. Use **OW** for word-wide outputs, **O** for byte-wide outputs.

```
cc86mon: OW ff80 abcd
cc86mon:
```

P – Boot Parameters

Syntax
```
P  [VariableName DecimalValue]
```

Description
The **Boot Parameters (P)** command sets or displays information about the boot
parameters of the system. The monitor stores these values in the Flash memory
device between the end of the monitor and absolute address FFF00. Variables can
be altered around five hundred times before this area becomes full.

```
cc86mon: p
    baudrate      = 19200
    cpuspeed      = 40000000
    led           = 1
    refresh_hz    = 64000
    waitstates    = 0
    autorun       = 0
    monitorport   = 1
    protectflash  = 000e0000
cc86mon:
```

Note that because you cannot execute from the Flash memory device while
modifying it, this command moves the monitor to RAM and then back, destroying
any user program already loaded.

E86MON supports the following permanent variables:

autorun   When this is non-zero, it selects which .EXE program to load from
             the Flash memory and run at boot time. The **W** command is used to
             store .EXE programs into Flash memory. The .EXE programs can be
             listed using the **L** command.

baudrate   This defines the default baudrate used if an **a** is not detected during
             the 3-second autobaud period at boot. (Note that the baud rate is not
             correct unless **cpuspeed** is also correct.)

cpuspeed   This defines the speed of the CPU to the monitor. This is required for correct default baud rate set up and correct DRAM refresh rate selection (Am186CC/CH/CU and Am186ED devices only). It is also used to set the internal timer tick correctly. The internal timer tick is used by benchmark programs and also governs the speed of the LED patterns.

hhkey   This should only be nonzero when the monitor is running on a Hamilton-Hallmark "keychain" system. It sets up various GPIOs correctly for this system.

led   When this is nonzero, the monitor uses the LEDs to show current status. When this is zero, the monitor does not change the LEDs. Set this to zero if the LED GPIOs are connected to other hardware and you do not want the E86MON software disturbing that hardware.

refresh_hz   This value is useful only for Am186CC/CH/CU and Am186ED devices. It defines the rate at which refresh cycles occur. The default rate is 64 KHz, which provides 512 refresh cycles every 8 ms. Note that the actual refresh rate is not correct unless the **cpuspeed** permanent variable matches the actual CPU clock speed.

waitstates   This value is used only on Am186CC/CH/CU and Am186ED devices to select wait states for the LCS DRAM.

protectflash   Locations between this variable and the monitor upgrade sector (Figure 1-2 on page 1-5) are not erased by the **XA** command. A specific sector can still be erased with the **X** *<sector number>* command.

2.0

R – Display/Alter Register

Syntax

```
R [RegisterName]
R F [FlagName]
```

Description

The **R** command with no parameters displays the current state of all registers and flags. **R** can also be used to set the value of any register or flag bit.

To examine a register, use **R** *RegisterName*. For example, **R AX** prints the current value of the AX register and prompts you for a new value.

To change a register without examining it, use **R** *RegisterName Word*. For example, **R AX 5000** changes the value of AX to 5000h.

To examine the flags, use **R F.** This prints the current flag values and prompts you for a two-letter code to change them. Flag names are the same as those used by DOS debug. If you use an incorrect flag name, the E86MON software prompts you with a list of valid flag names.

To change a flag without examining it, use **R F** *FlagName*. For example, **R F DN** sets the direction flag so that the direction flag is "down".

Note that in most situations, spaces are optional. The commands described in the previous few paragraphs could be entered by you as **RAX**, **RAX5000**, **RF**, and **RFDN**.

S – Search for List

Syntax

```
S <range> <list>
```

Description

The **Search for List (S)** command searches a given *range* for a *list* of bytes. The starting address of each occurrence of the list within the range is displayed. There is no display if the list is not found within the range.

```
cc86mon: s f8000 l200 "AMD"
0/00f8002
cc86mon:
```

2.0

T – Trace

Syntax

```
T [=address] [word]
```

Description

The **Trace (T)** command traces or steps through the execution of a program. Either single or multiple instructions can be executed by using the optional *word* parameter.

The starting CS and IP values can be set with the optional *address* parameter.

When the **T** command is entered, the Single-Step Flag (TF) of the Processor Status Flags (FLAGS) register is set, the stack is restored, and execution is resumed for one instruction. At this point a trace interrupt occurs, the registers are pushed back onto the stack, and the E86MON software prompt is displayed.

After the completion of the **T** command, the address at the stopped location is displayed on the screen. If the *word* parameter was entered, the trace command displays each address traced.

```
cc86mon: t =410:0 2
0410:0002
AX=1234  BX=0000  CX=0000  DX=0000  SP=03FA  BP=0000  SI=0000  DI=0000
DS=0041  ES=0041  SS=7F51  CS=0410  IP=0004   NV UP EI PL NZ NA PO CY
cc86mon: t
AX=1234  BX=0000  CX=0000  DX=0000  SP=03FA  BP=0000  SI=0000  DI=0000
DS=0041  ES=0041  SS=7F51  CS=0410  IP=0006   NV UP EI PL NZ NA PE NC
cc86mon:
```

W – Write to Flash

Syntax

```
W [mnemonic name]
```

Description

The **Write to Flash (W)** command initiates a download of a relocatable .HEX file (generated by running the host program MAKEHEX on a DOS executable) to the Flash memory device. The mnemonic name can be given so that the program can be identified later if multiple programs are stored in the Flash memory device. Programs are stored starting at the lowest address of the Flash memory device. Use the **L** command later to move a program into RAM for execution, or use the **P AutoRun** command to cause the monitor to load and run a program at boot time.

```
cc86mon: w seconds
Begin hex file transfer (using XON/XOFF) now...
Transferring hex file (Press Esc to abort)...........................
Device programmed successfully
Note -- the flash operation used (overwrote) the RAM.
cc86mon: l
    1  8000:0000    seconds
cc86mon:
```

2.0

X – Exterminate Flash

Syntax

```
X <sector number> | A
```

Description

The **Exterminate Flash (X)** command erases one of the sectors in the application area of the Flash memory, or, if **A** is given, erases all of them (unless the protectflash variable is used; see page 3-22). The **I** command can be used to retrieve information about the sectoring of the Flash memory device. Use 0 to refer to the first sector, 1 to the next one, etc.

```
cc86mon: xa
Erasing flash sector at 80000....
Erasing flash sector at 90000...
Erasing flash sector at A0000...
Erasing flash sector at B0000...
Erasing flash sector at C0000...
Erasing flash sector at D0000...
Erasing flash sector at E0000...  Protected! (not erased by XA)
Erasing flash sector at F0000...

Note -- the flash operation used (overwrote) the RAM.

cc86mon:
```

Note that because you cannot execute from the Flash memory device while modifying it, this command moves the E86MON software to RAM and then back, destroying any user program already loaded into RAM.

2.0

Z – Upgrade Monitor / Restore Flash / Lock Monitor in RAM

Syntax

Z

Description

The **Upgrade Monitor (Z)** command upgrades the boot monitor. It can be used for three purposes:

1. To upgrade the monitor:

   From a monitor that is running at the upgrade location (normally F0000h, but depends on the Flash memory type), type **Z** to upgrade the boot monitor in the same Flash memory device.

2. To restore Flash:

   From a monitor that is running at the boot monitor location (F8000h), type **Z** to replace a dead monitor in a different Flash memory device (on boards that support a CS switch from one Flash memory device to another, such as the Am186CC/CH/CU, Net186, and SD186ED boards).

3. To lock the monitor in RAM:

   Type **Z**, then enter **N** at the prompt. The monitor copies itself to RAM and remains executing from RAM. Extensions are not relocated to RAM.

See "Upgrading the E86MON Software" on page 2-5 for more information.

Appendix A

Notes on Terminal Emulators

This appendix provides information on using the E86MON software with various terminal emulators.

Most Windows programs send a break by using Ctrl-Break.

DOS PROCOMM PLUS sends a break by using Alt-B.

The HyperTerm program that comes with Windows 95 and Windows NT works with the E86MON software, but other commercially available programs work better.

HyperTerm has the following problems:

- Especially if the UART FIFO is enabled, priority is given to receive, making it difficult to abort a long dump done in error. Shutting off the UART FIFO helps this dramatically.

- The only way to abort a file transmission seems to be to Disconnect and Reconnect. Unfortunately, when you do this, HyperTerm forgets it has the file open, and you have to exit and restart HyperTerm to close the file.

- For transmitting .HEX files, the baud rate seems to be immaterial to HyperTerm. Unless your computer is exceptionally fast, transfers at 115200 bps operate no faster than at 19200.

- HyperTerm cannot send a raw ASCII file without either stripping or adding linefeeds, but this limitation does not affect version 3.2.1 or later of E86MON.

2.0

Appendix B

Utilities Included With the E86MON Software

This appendix provides a description of the MAKEHEX and EDITMON utilities that are included with the E86MON software.

The MAKEHEX Utility

MAKEHEX is a DOS utility program included with the E86MON software both in C source form and as an executable (.EXE) file.

The MAKEHEX utility takes a DOS .EXE, .COM, or .BIN file, and creates a .HEX file suitable for downloading to the E86MON software. There are two ways to use MAKEHEX:

```
MAKEHEX SAMPLE
```

This takes SAMPLE.COM, SAMPLE.BIN, or SAMPLE.EXE, and creates a relocatable .HEX file named SAMPLE.HEX. The E86MON software loads this file straight to RAM, relocates it, and prepares it for running. Alternatively, the **W** command accepts this file and stores it to Flash for subsequent copying to RAM.

The second way to use MAKEHEX is:

```
MAKEHEX SAMPLE <segment>
```

If a segment is given, the file should not contain any relocation items. MAKEHEX creates a .HEX file which is located at the given segment. This option is used to build the E86MON software itself.

NOTE: The MAKEHEX utility with Version 3.3.x of the monitor has changed from previous versions. Hex files built with the older (3.2.x) MAKEHEX utility do not download with the Version 3.3.x monitor. Additionally, the 3.3.x MAKEHEX files do not execute on a 3.2.x monitor. See Appendix C, "Error Messages".

The EDITMON Utility

EDITMON is a DOS utility program included with the E86MON software both in C source form and as an executable (.EXE) file.

The EDITMON utility allows editing of the E86MON software permanent variables inside the .EXE file before the E86MON software is converted using MAKEHEX.

EDITMON is useful for generating a file that is to be burned using a PROM programmer (e.g., EMON332B.HEX). The makefile of the E86MON software for the Hamilton-Hallmark Amber reference design uses EDITMON to set the CPU frequency, and to set the HHKEY permanent variable.

For more information on the EDITMON utility, see Appendix E, "Porting E86MON Software to Other Environments".

2.0

Appendix C

Error Messages

This appendix provides an alphabetical listing of error messages generated by the E86MON software.

Error Messages

`'permvar' is not a valid permanent variable`

The *permvar* name entered in the **P** command is not a valid permanent variable name. Check the spelling and try again. A list of valid permanent variables can be displayed by typing the **P** command with no parameters.

`Byte already programmed (erase first)`

Use the **X** command to clear the Flash before programming.

`Cannot erase the area you are running from!`

You usually encounter this error if you have loaded a copy of the E86MON software into the application area and are running from it. Press Reset or type **G FFFF0**, and type an **a** within three seconds to execute from the monitor in the boot location.

`Cannot identify device`

The E86MON software could not identify the type of Flash on the board.

`Cannot modify flash while executing from it`

This message should not be seen during normal operation.

Error Messages (continued)

```
Cannot modify flash within second monitor invocation (G
FFFF0 to restart monitor)
```

```
Cannot update monitor — app boot vector must point to
new monitor
```

The application boot pointer does not point to the new monitor. Use the **XA**
command to clear the application boot vector, and download the new monitor again
to cause the application boot vector to be filled in properly.

```
Cannot update monitor — must update from app flash area
```

The new monitor is not executing from the correct location (F0000). Use
MAKEHEX to reload at the correct location.

```
Cannot update monitor — new hardware init code too large
```

The size of the new hardware initialization area (in EMSTART.ASM) is larger than
224 bytes.

```
Cannot update monitor — new monitor too large
```

The new monitor does not fit into the space provided.

```
Erase operation failed
```

Your Flash device may require replacement.

```
.EXE file does not fit in memory
```

This message is sometimes seen on the SD186ER demonstration boards. Try to
make your program smaller.

```
File error: Invalid segment or start record at file line
number x.
```

The MAKEHEX utility with Version 3.3.x of the monitor has changed from
previous versions. Hex files built with the older (3.2.x) MAKEHEX utility do not
download with the Version 3.3.x monitor. Additionally, the 3.3.x MAKEHEX files
do not execute on a 3.2.x monitor. Rebuild your hex file using the new MAKEHEX
utility.

2.0

Error Messages (continued)

`Flash not formatted correctly (use XA to erase)`

There is not enough unused flash available to hold the relocatable program. Free up flash by using the **XA** command.

`Help not available — use LL command to load extension with help`

The sector containing the monitor extensions was erased (E0000), or the extensions were not installed. Download the extensions into Flash.

`Insufficient RAM for program's requirements`

A DOS program has requested more RAM memory than is available on the target hardware.

`Invalid checksum at file line number x`

This message indicates a problem with your .HEX file or that line noise was encountered during the download. Some processor variants, especially on early SD186EM and SD188EM systems, are susceptible to minute baud rate variations between the board and the terminal. If you suspect this is the case, set your terminal program for 2 stop bits.

`Invalid EOF record at file line number x`

This message indicates a problem with your .HEX file or that line noise was encountered during the download. Some processor variants, especially on early SD186EM and SD188EM systems, are susceptible to minute baud rate variations between the board and the terminal. If you suspect this is the case, set your terminal program for 2 stop bits.

`Invalid sector number`

Use the **I** command to determine the number of sectors in the device. The **XA** command only erases application sectors.

Error Messages (continued)

`Invalid Segment or start record at file line number x`

This message indicates a problem with your .HEX file or that line noise was encountered during the download. Some processor variants, especially on early SD186EM and SD188EM systems, are susceptible to minute baud rate variations between the board and the terminal. If you suspect this is the case, set your terminal program for 2 stop bits.

`Invalid start location record at file line number x`

This message indicates a problem with your .HEX file or that line noise was encountered during the download. Some processor variants, especially on early SD186EM and SD188EM systems, are susceptible to minute baud rate variations between the board and the terminal. If you suspect this is the case, set your terminal program for 2 stop bits.

`Memory region out of range: 1/00f0000 20 at file line number x. Note — the flash operation used (overwrote) the RAM.`

The hex file tried to load Flash to a reserved Flash location.

`No program loaded`

The default CS:IP register pair, which is set up when the E86MON software is entered, and whenever a program terminates, points to code that produces this message when executed.

`Non-hex character in record at file line number x`

This message indicates a problem with your .HEX file or that line noise was encountered during the download. Some processor variants, especially on early SD186EM and SD188EM systems, are susceptible to minute baud rate variations between the board and the terminal. If you suspect this is the case, set your terminal program for 2 stop bits.

Error Messages (continued)

```
Note — the flash operation used (overwrote) the RAM
```

This is simply an informational message indicating that the contents of the RAM may no longer be what you expect.

Flash programming code must be executed from RAM because code cannot be fetched from FLASH while flash is being programmed. Because of this, E86MON is copied to RAM prior to any flash write operation. This will overwrite any prior RAM contents.

```
Program not found in flash
```

A program corresponding to the parameter given in the **L** command was not found. Use the **L** command with no parameter to display a list of loaded programs.

```
Program operation failed
```

Your Flash device may require replacement.

```
Record does not start with colon at file line number x
```

This message indicates a problem with your .HEX file or that line noise was encountered during the download. Some processor variants, especially on early SD186EM and SD188EM systems, are susceptible to minute baud rate variations between the board and the terminal. If you suspect this is the case, set your terminal program for 2 stop bits.

```
Record Length Incorrect at file line number x
```

This message indicates a problem with your .HEX file or that line noise was encountered during the download. Some processor variants, especially on early SD186EM and SD188EM systems, are susceptible to minute baud rate variations between the board and the terminal. If you suspect this is the case, set your terminal program for 2 stop bits.

```
The current value of 'permvar' does not match its boot
value.
```

You tested a permanent variable change, and then tried to update the boot area of the Flash memory. Set all permanent variables the way you want them (from the boot monitor) before running and updating the E86MON software from the application area of the Flash memory.

Error Messages (continued)

```
The default value of 'permvar' cannot be updated.
```

You updated the boot monitor, ran it, changed a permanent variable, and are now trying to update it again. Use the **XA** command and then download the new monitor again before you update the boot monitor.

```
Unexpected Sentinel
```

This message indicates an error in processing Flash vector information.

```
Unknown record type at file line number x
```

This message indicates a problem with your .HEX file or that line noise was encountered during the download. Some processor variants, especially on early SD186EM and SD188EM systems, are susceptible to minute baud rate variations between the board and the terminal. If you suspect this is the case, set your terminal program for 2 stop bits.

```
User aborted download at file line number x
```

An ASCII control character (e.g., escape) was received during the download.

```
You must be running from the boot monitor to update
permanent variables.
```

You are running the E86MON software from a RAM copy or from the application area of the Flash memory. Press reset, or type **G FFFF0**, and type an **a** within three seconds to run from the boot area of the Flash memory.

```
W command requires relocatable file
```

After entering the **W** command, you downloaded a .HEX file that was not created from a .EXE file using the supplied MAKEHEX utility.

2.0

Appendix D

DOS Emulation Support

This appendix provides a description of the DOS emulation environment supported by the E86MON software.

The E86MON software supplies a minimal DOS emulation environment for running converted .EXE files. When an AMD-extended .HEX file is downloaded, a Program Segment Prefix (PSP) is generated for the file. Most of the PSP's fields that are examined by programs are filled in, and the program is relocated into RAM. One difference from normal DOS environments is that the default near heap is very small, especially on SD186ER demonstration boards. Use the far heap if this is a problem (e.g., _fmalloc).

The E86MON software also supports DOS emulation for ROM images. When a ROM image makes its first int 21h call, the E86MON software enables interrupts for the serial port and timer (which would not have been enabled if the E86MON software jumped straight to the application vector at reset). Then, the E86MON software is ready to provide services.

The following DOS interrupts are supported:

Int 20h	Terminate program execution
Int 21h, service 00h	Terminate program execution
Int 21h, service 01h	Character input with echo
Int 21h, service 02h	Character output
Int 21h, service 03h	Aux input (reads LED code)
Int 21h, service 04h	Aux output (writes LED code)
Int 21h, service 06h	Direct console I/O
Int 21h, service 07h	Raw input
Int 21h, service 08h	Raw input
Int 21h, service 09h	Display string terminated with '$'
Int 21h, service 0Ah	Buffered console input
Int 21h, service 0Bh	Check Keyboard Status
Int 21h, service 0Ch	Flush and execute keyboard function
Int 21h, service 25h	Set interrupt vector

Int 21h, service 2Ah	Get date (returns dummy date)
Int 21h, service 2Ch	Get time
Int 21h, service 2Dh	Set time
Int 21h, service 30h	Get version (returns 2.10)
Int 21h, service 35h	Get interrupt vector
Int 21h, service 3Fh	File read (only supported on console)
Int 21h, service 40h	File write (only supported on console)
Int 21h, service 44h	Get device information (only supported on console)
	Subfunction 0; Get device data
Int 21h, service 48h	Allocate memory
Int 21h, service 49h	Free memory
Int 21h, service 4Ah	Resize memory
Int 21h, service 4Ch	Terminate process with return code

If a downloaded program attempts to invoke an unsupported int 21h function, execution of the program is halted, and the E86MON software displays its command prompt. At this point, you can examine the registers in the downloaded program to determine which unsupported int 21h function was invoked.

In many cases, execution of the int 21h function is not critical to the operation of the downloaded program, and you can simply use the **G** command to resume program execution. However, if the execution of the unsupported int 21h function *is* critical to operation of the downloaded program, you must either modify the program to avoid execution of the unsupported function, or modify the E86MON software to add support for this function.

2.0

Appendix E

Porting E86MON Software to Other Environments

Although AMD only supports the use of the E86MON software on the AMD Am186 and Am188 family of boards, and assumes no liability for improper operation of the software, many customers find it advantageous to use the E86MON software for initially bringing up the hardware on their own designs.

Historically, customers wishing to use the E86MON software have modified the source code to adapt the software to their designs. However, versions 3.2.1 and later of the E86MON software can be used to bring up and test most hardware designs *without* modifying the source code, by using EDITMON.EXE to set the E86MON software's boot parameters.

If you are using Version 3.2.1 and later of the E86MON software, your \OUT subdirectory contains the files EDITMON.EXE and EMON332.EXE. The following command sequence modifies EMON332.EXE to change the CPU speed to 20 MHz, and the baud rate to 9600:

```
EDITMON EMON332 CPUSPEED 20000000
EDITMON EMON332 BAUDRATE 9600
```

When you have done this, you can use MAKEHEX.EXE to convert EMON332.EXE into EMON332.HEX. You want to use the *segment* parameter of MAKEHEX to locate the E86MON software in the top 32K of the device image:

```
MAKEHEX EMON332 7800    — For 512-KByte devices
MAKEHEX EMON332 3800    — For 256-KByte devices
MAKEHEX EMON332 1800    — For 128-KByte devices
MAKEHEX EMON332  800    — For 64-KByte devices
```

Before downloading EMON332.HEX to the device programmer, you should ensure that all memory in the programmer is erased to FFs. If this is not done, the E86MON software may assume that there is an application program loaded and may try to jump to it after booting.

At boot, the E86MON software automatically determines the Flash memory type and size (for most AMD Flash devices) and automatically determines the RAM size. If the E86MON software is programmed into an EPROM or a non-AMD Flash memory device, it does not perform operations that require programming the Flash memory, but can still perform RAM operations.

The E86MON software sets $\overline{UCS}$ to cover the upper 512K of the address space, and sets $\overline{LCS}$ to cover the size of the RAM in the lower address space.

If the Flash memory device is smaller than 512K, the device is aliased multiple times throughout the upper 512K of the address space.

If your application requires use of $\overline{MCS}$ in the upper 512K of the address space, you should first reprogram the size of $\overline{UCS}$ to match the actual Flash size, and then program $\overline{MCS}$ to the desired address. Otherwise, there is no need to change the programming of $\overline{UCS}$.

If source code modification of the E86MON software is required, see the file list in the README.TXT file to help determine which files require modification. Also, if you feel the modifications would be beneficial to others, please contact AMD EPD to submit them for possible inclusion in subsequent E86MON software versions. For information on how to contact AMD, see the numbers on the back of your manual.

2.0

Index

E

F

G

H

I

L

2.0

M

makehex program
    description, B-1
    using, 2-4
    version, 2-3
math, hex math command, 3-13
memory
    Flash organization, 1-5, 1-7
    RAM organization, 1-6, 1-7
messages, C-1
mnemonic name, 3-26
monitor location, 3-14
monitor size, 3-14
move memory (m) command, 3-18

N

name arguments (n) command, 3-19
name program in Flash memory, 3-26

O

output word (o) command, 3-20

P

parameters
    boot, 3-21
    command, 3-2
parity setting, 1-2
peripheral control block (PCB), 1-7
port settings, 3-21
porting E86MON software, E-1
power-on processing, 1-3
program
    autorun, 3-21
    load, 3-17
protectflash variable, 3-22

R

RAM location, 3-14
RAM size, 3-14
refresh_hz variable, 3-22
register (r) command, display/alter, 3-23
reset processing, 1-3
run program, 3-12

S

sample download files, 2-2
search for list (s) command, 3-24
serial connection, 3-16
set break point, 3-7
si register, 3-23
single step, 3-25
sp register, 3-23
ss register, 3-23
startup program, 1-3
status
    register, 3-23
    system, 3-14
stop bit setting, 1-2
support, iii
syntax, 3-2
system information, 3-14

T

technical support, iii
terminal emulators, A-1
third-party support, iii
time, 3-14
trace (t) command, 3-7, 3-25

2.0

U

$\overline{\text{UCS}}$, 1-7
upgrade E86MON software, 2-5
upgrade monitor (z) command, 3-28
utilities
    editmon, B-2
    makehex, B-1

V

variables, boot, 3-21
version number, 1-3

W

waitstates variable, 3-22
welcome screen, 1-2
write to flash (w) command, 3-26
WWW support, iii

X

xon/xoff flow control, 1-2

2.0