R o m E mÒ

USER'S MANUAL
Version 2.6

MICROSYSTEMS DEVELOPMENT TECHNOLOGIES, INC.
1177 Park Ave
San Jose, CA 95126 USA
Phone (408)280-1226

FAX 408.280.6868
www.RomEm.com
info@msd.com

COPYRIGHT AND TRADEMARK INFORMATION

TABLE OF CONTENTS

QUICK INSTRUCTIONS

To get started right away, follow these directions:

- Plug the RomEm's parallel connector (DB25 on rear panel) to your PC's parallel port using a good quality shielded cable.
- Insert the RomEm DIP Header into your prototype system. **Bottom justify** the header to emulate 24 or 28 pin parts. Pin 1 is on the marked side of the flat cable. Leave the test clips disconnected for now.
- Plug in the AC adapter into the rear panel and switch on the power. The red Power LED next to the power switch will be lit.
- Run the RomEm Plus software on your PC by typing the following (in boldface):

```
C:\>rep
P1U0>p 2              ← to change parallel port to LPT2, for example
2U0>t 27256                ← to emulate a 27256
Checking P2U0 as 27256  RomEm checks the attached unit
P2U0>l test.hex       ← load a file named 'test.hex'
Hex-File load of all bytes from offset 0 to offset 0
Load Completed with No errors (2000 Bytes Loaded)
P2U0>qy               ← quit RomEm and save a configuration file
:\>
```

- You're now emulating!
- To reload your program, you need to type only: **`rep l;q`**

This is because RomEm Plus software will save a configuration file containing the port number, EPROM type, filename, and other configuration data. This file, named rep.cfg will be created in the current working directory on your PC. Note that when RomEm commands are entered directly on the DOS command line, a semi-colon (;) must be used between them.

If you are unsure of what parallel port you have connected RomEm to, type: **p ?** to scan ports for connected RomEm units.

In the interactive mode, type: **help**
for a list of available commands, and type: **h x** to get help on any command x.
Note: The help file (rep.hlp) must be must be in the current directory or the same directory from which the RomEm Plus software (rep.exe) was located.

HOW TO USE MULTIPLE ROMEMS

For connecting two or more units on one parallel port:

- Set the rear panel DIP switches on each unit to a unique number as follows:
  Unit 0: Switch 1 down, Switch 2 down
  Unit 1: Switch 1 up, Switch 2 down
  Unit 2: Switch 1 down, Switch 2 up
  Unit 3: Switch 1 up, Switch 2 up

- Stack the RomEm units on top of one another.

- Connect the 10 x 2 headers on the rear panel with the short ribbon cable.

- Connect a wall transformer to **ONLY ONE** of the units.

- Connect one end of a male to male 25 pin cable to any one of the RomEm units and the other end to a parallel port on your PC.

- Switch all units on.

- **FOR 8 BIT SYSTEMS - EXAMPLE #1**

This is an example of how to emulate two 8 bit EPROMs on a 8 bit bus, loading from a single file:

```
C:\>rep
P1U0>p 2          ← to change parallel port to LPT2, for example
P2U0>t 27010          ← to emulate a 27010
Checking P2U0 as 27010   ← RomEm checks the attached unit
P2U0>l test.bin,,20000   ← load the 1st 20000h bytes from a file named 'test.bin'
Binary-File Load/Verify of 20000 bytes from offset 0 to offset 0
Load/Verify Completed with no errors (20000 Bytes Loaded)
P2U0>u 1          ← to change to RomEm unit #1
P2U1>l test.bin,20000     ← load all bytes after 20000h from a file named 'test.bin'
Binary-File Load/Verify of all bytes from offset 20000 to offset 0
Load/Verify Completed with no errors (xxxx Bytes Loaded)
P2U1>qy           ← quit RomEm and save a configuration file
C:\>
```

- **FOR 16 BIT SYSTEMS - EXAMPLE #1**

This is an example of how to emulate two 8 bit EPROMs on a 16 bit bus, loading from a single file, addressing two physical RomEm units as one logical unit:

```
C:\>rep
P1U0>p 2          ← to change parallel port to LPT2, for example
P2U0>u01          ← to select 0 & 1 interleaved (use 'u10' to swap bytes)
P2U01>t 27010         ← to emulate two 27010s, one for unit 1 and 1 for unit 0
Checking P2U0 as 27010   ← RomEm checks the attached units
Checking P2U1 as 27010   ← RomEm checks the attached units
P2U01>l test.bin     ← load all bytes from a file named 'test.bin'
Binary-File Load/Verify of all bytes from offset 0 to offset 0
Load/Verify Completed with no errors (xxxx Bytes Loaded)
P2U01>qy          ← quit RomEm and save a configuration file
C:\>
```

- **FOR 16 BIT SYSTEMS - EXAMPLE #2**

This is an example of how to emulate two 8 bit EPROMs on a 16 bit bus, loading from a single file, addressing each RomEm unit separately:

```
C:\>rep
P1U0>p 2              ← to change parallel port to LPT2, for example
P2U0>t 27010          ←to emulate a 27010
Checking P2U0 as 27010     ← RomEm checks the attached unit
P2U0>l test.bin,0,,,2      ← load all even bytes from a file named 'test.bin'
Binary-File Load/Verify of all bytes from every 2nd
byte at offset 0 to offset 0
/Verify Completed with no errors (xxxx Bytes Loaded)
P2U0>u 1              ← to change to RomEm unit #1
P2U1>l test.bin,1,,,2← load all odd bytes from a file named 'test.bin'
Binary-File Load/Verify of all bytes from every 2nd
byte at offset 1 to offset 0
Load/Verify Completed with no errors (xxxx Bytes  Loaded)
P2U1>qy               ← quit RomEm and save a configuration file
C:\>
```

TIPS FOR FAST DOWNLOADING

For the fastest downloading (loading a file into the RomEm), use as many of the following tips as possible:

1) Experiment with the port delay using the p (Port) command.
In order to be compatible with the widest variety of systems and parallel ports, RomEm has a built in delay which it uses when writing to the RomEm unit. The default delay is 2. If files load with no errors, try delay values of 1 or 0, for example:
p 2,1 ← use port LPT2, with a delay of 1
A delay of 0 will result in the fastest downloads, if your parallel port can support it. Once the delay has been changed, it will be stored in the rep.cfg configuration file for that port.

2) If you are using Intel Hex or Motorola S files, check your assembler or compiler to see if it can generate binary files.
Binary files are the quickest to load since there are no extra checksums, headers, addresses, etc. to be stripped out each time the file is loaded.

3) If the EPROM you are emulating is smaller than the memory in RomEm, try the nf (No Folding) option on the t (Type) command.
By default, RomEm software may be loading more than one copy of your file into the RomEm memory. This is done so that you can get RomEm up and running quickly without having to worry about exact pinouts and what to do with unused pins on your target system. This can make download time excessive however, if you are emulating small device types, for example if you are emulating a 2764 but your RomEm can support 27010 or 27040 device types. If this is the case, the fastest downloads will occur if you can either:

a) define the largest possible device type, e.g. 27010 if you have a RomEm with 1MBit of memory. This requires that your target socket be configured for this device.
b) be sure that your target socket conforms to strict JEDEC standards, e.g. Vpp=High, /PGM=High, N.C.=unconnected, etc. for the device type you are emulating, and use the nf (No Folding) option on the t (Type) command as follows:

```
t 2764,nf
```
← emulate a 2764 and load files into only one place in RomEm

4) Try loading with the nv (No Verify) option.
By default, RomEm first loads, then reads back and verifies data every time the l (Load) command is executed. This is the safest method because it insures that data is loaded correctly into RomEm. To speed up downloading however, you can eliminate the verify process using the nv (No Verify) option as follows:

```
l test.hex,,,,,nv
```
← load the file test.hex but do no verify
Once the no verify option is selected, it will be stored in the rep.cfg configuration file for that file. You can also use the vo (Verify Only) option to verify the contents of RomEm without loading.

WARNING: We do not recommend changing to the no verify option, since there is no checking done on the data loaded into RomEm.

1.0 INTRODUCTION

RomEm substitutes RAM for ROM so that code can be developed and debugged without burning ROMs. Code is simply downloaded into RomEm from your PC's parallel printer port. It can then be modified instantly to easily test changes. In addition, RomEm-Plus's intelligent features aid in the debug process.

1.1 RomEm features

- **Emulation of 2716 through 27080** (2KByte - 64KByte or 16KBits - 8MBits) with a single RomEm. RomEm is available in 1MBit, 2MBit, 4MBit or 8MBit versions.
- **Cascadable** up to 16 RomEm units to emulate 16 separate devices, or to emulate larger single devices.
- **Support for 16 bit to 32 bit systems.** More than one physical RomEm unit can be addressed as one logical unit with data interleaved, to make handling 16 and 32 bit bus sizes easy.
- **Portability.** After downloading, RomEm can be disconnected from your computer for true stand alone operation. RomEm stays close to your prototype -- where it should be, not inside your PC!
- **Battery back up and operation.** RomEm can retain its memory. RomEm can be moved and operated where it is needed: to your prototype, to an EPROM programmer, or back to your PC.
- **Reset Feature.** RomEm provides an external reset line as a convenient method of resetting your prototype system, or for providing a pulsed output for other reasons. This output can be configured as normally high, normally low, or normally High-Z.
- **Flexible Power Options.** RomEm can be powered from its own AC adapter so that the target system can be turned on and off without affecting RomEm's memory. When this option is used, RomEm can be used to power the target system also. RomEm can also be powered from the target system by using one of the test clips. RomEm can also be powered by an internal 9V rechargeable battery.
- **Powerful Software.** RomEm Plus interactive software is complete and easy to use. Functions include: Load, Write, Display, Edit, Monitor, Help, Calculator, and more...

- **Demo Mode.** By selecting demo mode in the software (selecting Port 0) your PC's RAM will be used in place of RomEm's RAM. Use this mode to browse and learn RomEm's commands. You can also use it to convert, display, edit, and otherwise manipulate a file.
- **File conversions.** RomEm can load and write binary, Intel Hex, and Motorola S format files. By using demo mode (port 0), RomEm software can be used even without a RomEm unit, to convert files from one format to another.

1.2 System Requirements

IBM PC, XT, AT, PS/2 or compatible
Parallel printer port
DOS 2.1 or higher
125K available conventional RAM

2.0 RomEm HARDWARE

2.1 Front Panel and Cable

2.1.1 Power Switch and LED

The Front Panel power switch will power RomEm via the AC adapter, internal battery, or via the Red test clip. The red power LED will be on steady when the switch is in the on position and power is being supplied. If power is being supplied by the AC adapter or by an internal battery, the RED test clip can be used to supply power to the target system.

The red power LED is off when the power switch is in the off position.

2.1.2 Target Interface - DIP Header

Interface to your target prototype system is provided via the 34 pin header on the front panel. The supplied cable converts this header to a 32 pin DIP plug suitable for plugging into a socket on your prototype system. If you are emulating 24 or 28 pin devices, **bottom justify** RomEm's DIP plug in your prototype socket. Refer to the Appendix for the exact pinout of this cable.

All interface components to the target system are standard CMOS (HCT) parts. These components are socketed on the main RomEm board should they need to be replaced to suit your needs.

2.1.3 Vcc In/Out - Red Test Clip

The Red Test Clip on the flat cable can be used for either input or output as follows:

Vcc Out If RomEm is being powered by the AC adapter or the internal 9V battery, the Red Test Clip is a 5V output and can be used to power the target system. Note that RomEm's power switch will control this output as well. If left unconnected, the target system's power will be separate from RomEm, and the target system can be turned on and off without RomEm losing its memory.

Vcc In The Red Test Clip can also be used to power RomEm from your target system if you are not using an AC adapter or internal battery.

2.1.4 Reset Output - White Test Clip

The RESET signal is an output and can be used for resetting your prototype, or for generating a pulse for other purposes. This is controlled via RomEm Plus software. This output can be configured as normally high, low, or high-Z. See the '**r**' command for more details.

2.2 Rear Panel

2.2.1 Power Input

Power can be supplied to the RomEm using the supplied AC adapter, plugged into the connector on the rear panel. The supplied AC adapter is 9VDC, 500ma.

Power can also be supplied using a 9 Volt battery. This battery must be connected internally to J5 by unscrewing an end plate from the RomEm case and sliding the board out. A standard alkaline battery or a rechargeable ni-cad can be used.

2.2.2 Parallel Interface

The DB25 connector on the rear of the RomEm unit is designed to be connected to a parallel printer port of an IBM compatible PC with a 25 pin Male-to-Male cable. By using a parallel connection, code can be loaded in and out of RomEm's memory much faster than using serial methods.

Use a good quality shielded printer cable no longer than 8 feet. If you experience problems interfacing to RomEm through the parallel port, it could be due to an inferior cable.

If you have the need for multiple RomEm units, you can use a separate parallel port for each unit, or you can use the 20 pin bus header to stack RomEm's (see below).

2.2.3 20 Pin Bus Header

Use the 20 pin bus header on the rear of the RomEm if you want to connect more than one unit to a single parallel port. In this case, stack the RomEms on top of each other and connect them with a short interface cable between the 20 pin headers on the rear panel. Contact MicroSystems Development to purchase the appropriate cable. Up to four RomEms can be connected in this manner. **Each RomEm must have a unique Unit Number.** The RomEm software addresses each RomEm unit based on this jumper setting.

2.2.4 Unit Number Switches

The Unit Number Switches are located next to the 20 pin header on the rear panel. This is the Unit Number (or address) of the RomEm as seen by the RomEm Plus software. It has no effect on emulation. Up to four RomEms can be connected to a single parallel port on a PC. Each RomEm connected to the same parallel port must have a unique Unit Number. **If only one RomEm is connected to your PC, switch both Unit Number switches DOWN (ON). This is the factory default setting.** RomEm Plus software will refer to this RomEm unit as Unit 0. If there is more than one unit connected to a parallel port, each must have a unique Unit Number from 0 to 3, and will be referenced by this number with the RomEm Plus software. Tell the software which units are connected by using the 'u' (unit) command.

Unit Number Switches: Leave at 0 (both down) for a single RomEm unit.Set to 1, 2, or 3 for multiple units connected together on the same parallel port.

UNIT NUMBER 0: BOTH SWITCHES DOWN **** FACTORY DEFAULT**

UNIT NUMBER 1: SWITCH 1 UP, SWITCH 2 DOWN

UNIT NUMBER 2: SWITCH 1 DOWN, SWITCH 2 UP

UNIT NUMBER 3: BOTH SWITCHES UP

NOTE: If two or more RomEm units are connected to the same parallel port, and their Unit Number switches are set to the same number, neither RomEm unit will work properly!

2.3 Internal Battery Jumper (J7)

Jumper J7 selects the type of battery installed, if any. Install the jumper if you are using a rechargeable nickel cadmium battery and would like the AC Adapter to continuously charge it. Remove the jumper to disable charging or if you are not using a rechargeable battery.

Jumper J7: **Install** to recharge a rechargeable battery with the AC Adapter
Remove to disable charging, or for a non-rechargeable battery

WARNING: Installation of this jumper with a non ni-cad battery may cause the battery to leak, melt, or explode!

3.0 RomEm Plus SOFTWARE

3.1 General Operation

RomEm Plus software is used to operate the RomEm unit(s). It is invoked by typing: **rep** <enter>. Up to four emulator RomEm units may be connected to any one parallel port. The program accepts commands from the user. These commands can be used to perform actions on the RomEm units connected to the PC.

A configuration file named "rep.cfg" is maintained in the current working directory to allow termination and re-invocation of the program without losing most of the working environment. In addition, a DOS environment variable, "ROMEM" allows the user to specify a fixed list of commands to be executed whenever the program is started.

Upon invocation of the program, the following steps are performed:

(1) If the file REP.CFG exists in the current directory:
   (a) The file is opened and the default configuration is set from its contents.
   (b) The emulator units defined in that configuration are initialized.
   (c) The current port and unit are set to those defined in the configuration.
   (d) Other default command arguments are initialized from information in the configuration file.
   (e) If the symbol "/N" is present in either the environment variable or on the DOS command line, this entire step is bypassed and the configuration defaults to: no defined emulator units, current port = 1, and current unit = 0.
   This step is referred to as "configuration initialization" throughout this document.

(2) Any commands in the ROMEM environment variable are executed.

(3) Any commands on the DOS command line are executed.

(4) Commands are accepted from the keyboard.

In interactive mode, commands are entered at the keyboard in response to a greater-than (">") prompt character. The prompt character is always preceded by the current port and unit number in the form "PxxxxUn", where xxxx is the 1-4 digit (hex) port value and N is the one-digit unit number. For example: P1U0 for Port 1 and Unit 0 (the default). The maximum length of any command line is 71 characters.

HELP is available by pressing h in the interactive mode. The file rep.hlp must reside in the same drive and directory as the rep.exe file, in the current directory, or in the DOS path.

Note that when more than one command is entered on a line, a semi-colon (;) must be used between them. For example:

rep r h ; l test.hex ; r l ; qy

The above line can be inserted into a batch file, to start RomEm, set reset high, load a file called test.hex, and set reset low.

3.2 Syntax and Usage

A command consists of a single command character followed by zero or more arguments. **There must be no space between the command character and the first argument.** Commands can be given in upper or lower case. Where multiple arguments are required, arguments are separated from each other by commas.

More than one command may be placed on a line. Commands are separated from each other by one or more spaces or tabs.

Required arguments are always first, and optional arguments are always last. An omitted optional argument takes on a default value that is specific for each optional argument of each command. Some take on fixed values while others take on values dependent on the current state and environment. Arguments are of four types: keywords, file names, values, and strings:

keywords These arguments must exactly match one of a list of strings allowable for that argument (case is not significant).

file names May be any valid DOS drive/path/filename string.

values **A hexadecimal number** is written with no special syntax (**example: 142c**).
A decimal value must be preceded by a number sign (**example: #25**).
An ASCII character must be preceded by a single apostrophe (**example: 'f or 'F**).
Case is significant for ASCII only.

operators Values may be combined using the following set of operators:
+ add - subtract * multiply / divide
& bitwise and | bitwise or ^ bitwise exclusive or
{ shift left } shift right ~ not (1's complement)
There is no precedence. Evaluation is left to right, but parentheses can be used to control the order of operation.
For example: 124C+#10-'A is equal to 1215 or #4629.

$ registers There are ten registers, $0 to $9, which can be assigned a value with the = command.
For example,
=11,22,33 will assign he value of 11 to $0, 22 to $1,
and 33 to $2. The $ registers may be used in

subsequent expression, enabling quite sophisticated functions.

strings A string argument is any set of ASCII characters enclosed in quotation marks, including spaces, tabs, and commas. If a string argument is the last thing on a command line, then the closing quotation mark is optional.

The special character, backslash ("\") causes the next character or characters to have special meaning as follows:

\r	represents a single carriage return character, hex 0d.
n	represents a single line feed character, hex 0a.
"	represents a quotation mark.
xDD	represents a single character whose hexadecimal equivalent is DD (example: \x0d is equivalent to \r).
\\	represents a single backslash character.

3.3 Editing Keys

A rich set of editing keys is supported during command line entry. This has been done for those users who wish to run inside of rep as their operating environment, but who miss the features of DOSKEY. The last 200 commands are kept in a history buffer.

Backspace	back up over and delete previous character.
Escape	go to beginning of line and erase line.
Left Arrow	move cursor left without erasing.
Right Arrow	move cursor right without erasing.
Home	go to beginning of line without erasing.
End	go to end of line.
Ctrl-Left Arrow	move to first char of word.
Ctrl-Right Arrow	move to first char of next word.
Ctrl-Home	same as Escape.
Ctrl-End	erase from cursor to end of line.
Ins	toggle insert/overstrike mode, cursor shape changes.
Del	delete character at cursor.
Up Arrow	recall previous command for editing. May be used to scroll backwards through the last fifty commands. Position always starts at the bottom of the list for each new command line. Recalled commands that are executed with no editing are copied to the bottom of the list.
Down Arrow	scroll down in recall list. .
Tab	Move cursor right 10 characters.
Shift-Tab	Move cursor left 10 characters.
PgUp	Recall search backward. Look for command that starts with what is on the line up to the current cursor position.
Ctrl-PgUp	Recall very first command in the buffer
PgDn	Recall search forward
Ctrl-PgDn	Clear command line and move recall pointer all the way to the end of the buffer
Enter	Execute current line

3.4 Demo Mode

Normally, RomEm Plus software operates on the RomEm memory connected to parallel port 1, 2, 3, or 4. In order to test and use the RomEm Plus software without having a RomEm attached, define a unit on **'Port 0' by doing a 'p0' command**. RomEm software will them use your PC's RAM in place of a RomEm. You can then load, display, edit, and write files just as you would if a RomEm were connected.

RomEm can use extended memory to simulate RomEms with large amounts of memory. The memory must be supported via an XMS 3.0 (or greater) driver such as HIMEM.SYS or QEMM. Detection of the driver is automatic when the 't' command is executed to allocate a certain size of memory. See the 't' command for more details on XMS and conventional memory usage.

p (port, select parallel port)

Purpose: Select parallel port for subsequent operations.

Format: p <value>, [<type> or <delay>]

Arguments: The <value> argument is required any must be 0, 1, 2, 3, 4 or any value greater than FF. Values of 1, 2, 3, and 4 indicate LPT1, LPT2 LPT3 and LPT4 as defined in the ROM BIOS port address table and must have a non-zero port address in that table to be valid.

A <value> of '?' causes all parallel ports to be scanned for attached RomEm units. A table of attached units will be displayed, and the prompt will change to the first unit found, if the prompt was not already at a currently defined unit. Warning - using this command may have undesired effects on printers or other devices attached to other parallel ports.

A <value> greater than ff hex represents an absolute parallel port address (I/O address of the data output register) and may or may not be a duplicate of a port address in the ROM BIOS table.

A 0 value directs the RomEm software to use your PC's memory in place of RomEm. This is useful for demonstration, when converting files from one format to another, or when editing files.

<type> is *only* valid for Port 0 and can be:

x Allocate only out of extended RAM. Must have an extended memory driver such as HIMEM.SYS or QEMM loaded.

c Allocate only out of conventional memory. Limits the size of EPROM type that can be defined with the t command.

xc DEFAULT Allocate first out of extended then out of conventional

cx Allocate first out of conventional, then out of extended

<delay> is *not* valid for Port 0, and specifies the delay, from 0 to 99 used when writing to RomEm through the parallel port. Some parallel ports cannot handle very high data rates, and an additional delay is necessary. The default delay is 2. If files load with no errors, try delay values of 1 or 0 for faster loading. If you occasionally experience errors when downloading to RomEm, try a higher delay.

Comments: No error message is given for a bad port selection if the value given is within range. An undefined or inaccessible port will not be flagged until the first attempt to use that port ('t' command).

All arguments are saved in the configuration file and may be displayed using the ? command. If a configuration file was loaded at program startup, then the current port specified in that file is selected until the first 'p' command is executed. If no configuration was loaded, then LPT1 is the default.

The RomEm prompt will change to reflect the new port number or address.

Examples:

p 1 Selects LPT1 (default) with the default delay of 2.

p 2,0 Selects LPT2 with no delays (for fastest subsequent downloads)

p ? Scans all parallel ports for attached RomEm units at the default delay of 2

p ?,1 Scans all parallel ports for attached RomEm units at delay of 1

p 278 Selects parallel port at hex address 278.

p 0 Selects PC memory instead of a connected RomEm.

p 0,c Selects PC memory and uses conventional memory only.

t (type, define EPROM type)

Purpose: Define type of EPROM / ROM to be emulated on the current port and unit.

Format: t <keyword>,<fold-type>

Arguments: <keywords> are EPROM part numbers with the '27' prefix being optional, or ROM byte sizes in hex or decimal. The acceptable keywords are:

2716	16	2048	800	2K
2732	32	4096	1000	4K
2764	64	8192	2000	8K
27128	128	16384	4000	16K
27256	256	32768	8000	32K
27512	512	65536	10000	64K
27010	010	131072	20000	128K
27020	020	262144	40000	256K
27040	040	524288	80000	512K
27080	080	1048576	100000	1024K

<fold-type> refers to way in which this RomEm unit is connected and the location in which RomEm will load files into it. There are three type of 'folding' types as follows:

nf NO FOLDING This is the fastest loading/verifying mode but imposes the most restrictions on the target system.

The emulated ROM image is loaded in one and only one place in the emulator RAM space. All pins labeled Vcc, /PGM, Vpp, etc must be in their proper states on the target system socket. All pins labeled N/C for the EPROM device type must be unconnected. In addition, when the 32 pin RomEm target cable is plugged into a 24 or 28 pin socket., the unused pins must be left unconnected.

pf PARTIAL FOLDING (default) This is not quite as fast as the 'nf' mode but eliminates very long load times for small target devices into large emulator RAMs while not being quite so insistent about the states of certain pins.

The emulated ROM image is loaded in all possible positions in the emulator RAM space assuming that only those for the physical device size are connected. Pins such as Vcc, /PGM, and Vpp can be in any state. When emulating a 28 pin device however, pins 1, 2, 30, and 31 on the 32 pin target cable must be left unconnected, and when emulating a 24 pin device, pins 1, 2, 3, 4, 29, 30, 31, and 32 on the target cable must be left unconnected.

ff FULL FOLDING This mode results in the slowest download, but is the safest mode because no assumptions about special or unconnected pins are made.

The emulated ROM image is loaded into all possible locations in the emulator RAM. For example, when emulating a 27256 with a 1Megabit RomEm, the ROM image will be 'folded' (loaded) 4 times,

at addresses 0h, 8000h, 10000h, and at 18000h. Because load times may be excessive, we recommend that you do not regularly use this mode, rather check/alter your target socket so that one of the above two modes can be used.

Comments: Execution of the 't' command causes a "quick test" to be run on the selected parallel port and RomEm unit. This verifies that the unit can be accessed, and that its memory can be read and written (not an exhaustive RAM test).

The <keyword> argument is optional if the type for the current port and unit has already been defined in a previously successful 't' command or in the default configuration.

Upon successful completion of a 't' command the port, unit and type become part of the current configuration, and can be displayed using the ? command. If the configuration is saved upon program exit, then a quick test" will be run on the unit during the next configuration initialization.

If the RomEm unit fails the quick test (e.g. not connected to the PC), the current unit information is taken out of the configuration.

If the current unit number consists of more than one physical unit, i.e. u01, each physical unit will be tested as the selected type.

Note that the unit is taken out of emulation mode during this test.

Examples:

t 2764	Set EPROM type to 2764 and do a quick test of the current RomEm unit on the current port.
t 010	Set EPROM type to 27010 and do a quick test of the current RomEm unit on the current port.
t 2764,nf	Set EPROM type to 2764, do a quick test of the current RomEm unit on the current port, and when loading, load the file into only one location in RomEm (make sure your target socket has any special pins in the proper state).
t	Perform a quick device check on the current unit.

u (unit, select RomEm Unit Number)

Purpose: Select emulator unit (on current parallel port) to be used for subsequent operations.

Format: u <value>

Arguments: The <value> parameter is required and must be 0, 1, 2, 3, or a combination of up to four different numbers. This number **must** correspond to the Unit Number switches on the rear panels of the attached RomEm units.

Comments: No error message is given for a bad unit selection if the value given is within range. An inaccessible or non-functional unit will not be flagged until the first attempt to use that port ('t' command).

A <value> which is a combination of numbers such as 10, 01, 0123, etc. is used to indicate that data should be interleaved between the chosen unit numbers. This is for 16 or 32 bit wide systems.

The <value> is stored in the configuration and can be displayed using the ? command.

If a configuration file was loaded at program startup then the current unit specified in that file is selected until the first 'u' command is executed. If no configuration was loaded then unit 0 is selected until the first 'u' command.

The RomEm prompt will change to reflect the new unit number.

Examples:

u 1 Select Unit 1 on the current parallel port.

u01 Selects Units 0 and 1 on the current port. Data will subsequently be interleaved when loaded. Even bytes will be loaded into unit 0 and odd bytes will be loaded into unit 1.

u10 Selects Units 1 and 0, but bytes will be loaded opposite of the above.

u0123 Selects Units 0, 1, 2 and 3. Subsequent loads will cause every 4th byte to be loaded into each RomEm unit.

3.6 File Commands

l (load file into RomEm)

Purpose: Load current emulator unit's RAM from a file.

Format: l <filename>,<start-value>,<length-value>,<target-value>,<width-value>,<verify-opt>

Arguments: All arguments are optional. If ALL arguments are omitted, then the load command arguments saved in the configuration for the current port and unit are used.

<filename> gives the drive/path/name of a binary file, an Extended Intel Hex file, or a Motorola S-format file to be loaded. The type of the file is determined automatically by inspection of the beginning of the file. If no filename is given, the default is taken from the configuration file, if one exists.

The remaining arguments are used to select exactly what from the file is loaded where. They may be omitted, and default values will be used. Interpretation of the optional arguments differs depending on the file type. This is described below.

If any other arguments are specified, they must **all** be given with the command. None will be used from the configuration.

Comments: Load arguments are stored in the configuration and may be displayed using the ? command.

This command must be preceded by a successful 't' command (or configuration initialization) with the same port and unit selected. The unit is taken out of emulation mode during this operation.

Binary File Load
<start-value> is the file offset of the first byte to be loaded. The initial default is 0.

<length-value> is the number of bytes to be loaded. The resulting value may not cause a read past the end of the file or a load address past the end of the defined type.

<target-value> is the byte offset into the defined emulated ROM where the first byte is to be loaded. The default is 0. The value may not cause a byte to be loaded past the end of the defined emulated ROM.

<width-value> is used to cause the loading of every Nth byte of the input data into the emulated ROM. The default is 1.

For example, to load all of the even bytes of a 4K file into a 2K emulator "l imagefile,0,#2048,0,2" could be used. To load the odd bytes "l imagefile,1,#2048,0,2" could be used.

<verify-opt> is the option to verify the download or not, and can be:

v to verify the file (default). This causes the file to be read back out of RomEm memory and compared with what was loaded. This is the safest option but does cause the download to take a little longer because of the read back.
nv for No Verify. The downloaded file is not read back and verified. This will result in fast downloads, but is not recommended until you have had some experience with using RomEm on your system, and have not experienced any download errors.
vo for Verify Only. This options does not load, but only verifies the contents of RomEm with the specified file.

Extended Intel Hex File
Complete support for Extended Intel Hex Files is provided. The reader should be somewhat familiar with this format before attempting to understand what follows.

In order to understand how the arguments apply to a hex file it is useful to construct the following mental model: Consider a 16 megabyte address space. An Extended Hex File contains instructions for placement of data bytes into various locations in this address space. Think of the hex file being converted into a 16 Megabyte binary file with file addresses from 0 to FFFFFF. The arguments are then applied to this binary file in the same way as described for binary files above.

The only other thing to remember is that in an Extended Intel Hex File, data records before any segment record are applied to segment 0000. Also note that bytes of the 16 Meg binary file that have no data specified for them in the hex file are left unchanged in the emulated ROM and will therefore retain their previous random data or filled data.

Motorola S-format File
The same scenario applies for mapping of arguments into a large memory image. A Motorola S format file maps onto a 16 Megabyte binary file.

Examples:

l	Load the last file used (as defined in the configuration file) for the current unit into RomEm.
l test.hex	Loads the file named "test.hex" from the current drive and directory into the current RomEm.
l test.bin,,,8000	Loads the file named "test.bin" into RomEm with an offset of 8000 hex. This will load the file starting at address 8000h in RomEm.
l test.bin,,100	Loads the first 100 hex bytes of the file "test.bin".
l test.bin,,10,,,nv	Loads the first 10 hex bytes of the file "test.bin", and does not verify the data.
l test.hex,,,,,nv	Loads the file "test.hex" and does not verify the data.

Note: the $0 register is set to the address of the highest byte loaded, plus 1.

q (quit RomEm software)

Purpose: Quit RomEm Plus software, leaving all units in emulation mode.

Format: q <keyword>

Arguments: <keyword> can be: y or n. 'y' will automatically update the configuration file. 'n' will not update the configuration file.

Comments: If the configuration has not been changed during the session being concluded, then <keyword> is ignored and no further action with regard to the configuration file is taken.

If the configuration has changed during the session, and no <keyword> is given, the user is asked if he wants to update the configuration and must respond with 'y' or 'n'.

Note: RomEm Plus software cannot be re-entered without taking the unit out of emulation mode. Use the ! command to exit and return to RomEm software without disturbing the RomEm units.

Examples:

q Quit RomEm Plus software. If the configuration has changed, the user will be asked whether or not he wishes to update the file.

q n Quit RomEm Plus software without updating the configuration file.

q y Quit RomEm Plus software and update the configuration file if it has changed.

w (write RomEm memory to a file)

Purpose: Write an image of the current emulator's memory to a file

Format: w <filename>,<start-value>,<length-value>,<mode>,<type>,<rec-length>,<fill>,
<target>,<entry>,<signon>

Arguments: The <filename> argument is required and specifies the file to be written to.

<start-value> and <length-value> define the portion of the emulator memory that is to be written. <start-value> defaults to 0 and <length-value> defaults to the rest of the defined size of the memory. Combined, they must not reference a byte outside of the defined type.

<mode> is optional and may be:

r	replace an existing file
a	append onto existing file
c	cancel if the file exists
?	ask operator what to do if the file already exists (default)

<type> is optional and may be:

b	binary file (default)
h	Intel extended HEX format
h-	Intel extended HEX format without an EOF record
h32	Intel linear extended HEX-32 format
h32-	Intel linear extended HEX-32 format without an EOF record
s	Motorola S format
s-	Motorola S format without an EOF record
s28	Motorola S28 format
s28-	Motorola S28 format without an EOF record
s37	Motorola S37 format
s37-	Motorola S37 format without an EOF record

Notes: H type automatically selects conventional extended HEX or linear extended HEX-32 based on the size of the required output addresses. S type automatically selects S19 (16 bit), S28 (24 bit) or S37 (32 bit) based on the largest address that must be output.

<rec-length> can be from #1 to #255 (default #32) and specifies the maximum number of data bytes that are to go into each formatted record.

<fill> is a byte value from #0 to #255 (ff) which causes all these values present at the beginning and the end of any record to be eliminated from the formatted file. If the record consists entirely of this value, it will not be written to the file.

<target> is the offset address that the first output data byte is to be given in the output file. If omitted, <start-value> is assumed.

<entry> is the entry point address that is to be stored into the formatted file. For Intel HEX, the address is output in a 'start address record' or a 'linear start address record'. For Motorola S, the address is output in the appropriate place in the EOF record.

<signon> may be present for Motorola S format files only and causes a signon record to be generated with the given string text.

Comments: This command must be preceded by a successful 't' command (or configuration initialization) with the same port and unit selected. The unit is taken out of emulation mode during this operation.

Examples:

w new.bin	Write the entire contents of RomEm memory to a binary file named "new.bin" in the current drive and directory.
w junk,0,ff	Write the first 256 bytes of the RomEm memory to a binary file named "junk" in the current drive and directory.
w test.bin,,,r	Write RomEm memory to a binary file named "test.bin" and replace (overwrite) the file if it already exists.
w xyz.hex,,,,h,,ff	Write the contents of RomEm to an Intel HEX file named "xyz.hex", but do not include any values of ff (hex) that may be present at the beginning and the end.
w xyz.s,,,,s,,ff	Same as above, but write a Motorola S format file to "xyz.s".

@ (use command file)

Purpose: Proceed by taking commands from a given file.

Format: @ <filename>,<keyword>

Arguments: The <filename> argument is required, and must be an ASCII text file.

<keyword> is an optional argument that allows control over what is displayed on the screen as the commands in a command file are executed. If omitted, all command lines and progress reports are displayed just as if the commands are entered at the keyboard. Allowed keywords are:
V - display all normal progress reports,
Q - suppress progress reports,
E - echo command lines in the normal way, and
N - suppress command line echo.

Comments: Each line of the file is treated like a command line entered from the keyboard. Blank lines and lines beginning with two slashes (//) are ignored. When the end of the file is reached or an error occurs in a command, control returns to the keyboard. Lines are displayed as they are executed. If an error occurs in the execution of any command line in the file, all file reading is terminated, and control is returned immediately to the keyboard.

Examples:

@ run.re Proceed by taking commands from the file "run.re" in the current drive and directory.

3.7 Edit Commands

d (display RomEm memory)

Purpose: Display (or Dump) data from the current RomEm's memory.

Format: d <start-value>,<length-value>,<width-value>

Arguments: The optional <start-value> is the offset in the emulator memory of the first byte to be displayed and must be within range of the defined type. The default is the next byte after the last byte displayed by the most recent 'd' command for the same emulator unit or 0 if no 'd' command precedes.

The optional <length-value> is the number of bytes to be displayed. It defaults to the <length-value> of the most recent 'd' command (for any port/unit) or to #128 if no 'd' command precedes. If the <start-value> plus <length-value> exceeds the size of the defined ROM, the display address will automatically wrap to 0 when the end of the defined ROM is reached.

The <width-value> is also optional and tells how displayed bytes are to be grouped. A new display line is started every <width-value> bytes. It defaults to the <width-value> of the most recent 'd' command (for any port/unit) or to #16 if no 'd' command precedes.

Comments: All of these arguments are saved in the configuration file and re-instated during configuration initialization.
During the display, the user may press Control-S to temporarily suspend the display, or Control-C to cancel the display operation.

If the current unit number consists of more than one physical unit, i.e. u01, data will be displayed in an interleaved fashion, taking one byte from each physical unit on a round robin basis.

Examples:

d 100 Dump 128 bytes starting at address location 100 hex.

d Dump the *next* 128 bytes.

d 100,#32,#8 Dump 32 bytes, 8 bytes per line, starting at 100 hex.

Note: The $0 register is always set to the last byte value displayed.

e (edit RomEm memory)

Purpose: Edit (or Enter) data in the current RomEm's memory.

Format: e <start-value>,<data-value OR data-string>...

Arguments: The <start-value> argument gives the offset into the memory of the first byte to be entered. <start-value> may be omitted if and only if the immediately previous command was also an 'e' command. In this case it will default to the next byte address after the last byte entered by the previous command. The first 'e' command of a session must always have a <start-value> argument.

Any number of <data-value> and/or <data-string> arguments may follow. A <data-value> argument must be in the range 0 - ff hex and causes one byte to be written to the memory. A <data-string> argument is enclosed in quotes and causes all bytes of the string to be written to successive memory locations. The <start-value> plus the length of the arguments must not exceed the size defined for the current emulated ROM ('t' command).

Comments: If no data arguments are given, a byte-by-byte interactive editing mode is begun. The current value of the next byte address is displayed in hex, and the user may enter a replacement value for it in hex, decimal or ASCII. If no value is entered, the contents of the address are not changed. A space proceeds to the next address. Backspace will back up, but only to the beginning of the current line. A return terminates the command.

If the current unit number consists of more than one physical unit, i.e. u01, data will be displayed in an interleaved fashion, taking one byte from each physical unit on a round robin basis.

Examples:

e 0	Begin the interactive editing process at address 0.
e 2a7,0d	Enters the hex value 0d into hex address 2a7.
e 200,#76	Enter the decimal value 76 into hex address 200.
e 100,"Hello	Enters the ASCII values for the string "Hello" into memory starting at hex address 100.

f (fill RomEm memory)

Purpose: Fill all or a portion of the current emulator unit's RAM with a certain value.

Format: f <fill-value>,<start-value>,<length-value>

Arguments: <fill-value> is required and must be within the range 0 - ff hex.

<start-value> is optional and specifies the offset within the emulated ROM where filling is to start. It may not specify a byte that is beyond the end of the defined emulated ROM. The default is 0.

<length-value> is optional and specifies the number of bytes to be filled. When combined with <start-value> it may not cause filling to go beyond the end of the defined type. The default is to fill the rest of the defined ROM.

Comments: The unit is taken out of emulation mode during this operation.

Examples:

f ff — Fill the entire memory of the current unit with hex ff.

f 'A — Fill the entire memory of the current unit with ASCII "A" (hex 41).

f #255,0,ff — Fill the first 256 bytes (0 to ff hex) of the current unit with decimal 255 (hex ff).

s (sum, display 32 bit checksum)

Purpose: Display the 32 bit checksum of the current emulator's memory.

Format: s <start-value>,<length-value>

Arguments: The optional <start-value> and <length-value> arguments specify the starting address and the number of bytes to be included in the checksum. The <start- value> default is 0 and the <length-value> default is the rest of the defined size of the memory.

Comments: The unit is taken out of emulation mode during this operation. This command must be preceded by a successful 't' command (or configuration initialization) with the same port and unit selected.

The output is of the form XXXX-XXXX, which are the eight hex digits of the 32 bit checksum. the first 4 digits can be ignored for a 16 bit checksum, the first 6 can be ignored for an 8 bit checksum.

Examples:

s Display the checksum for the entire defined memory.

s 0,2000 Display the checksum for the first 2000h bytes of memory. This is equivalent to the s command with no arguments if the defined type is 2764.

s 0,#4094 Display the checksum for the first 4094 bytes of memory.

Note: The $0 register is always set to the 32 bit result value.

r (reset, controls the Reset clip)

Purpose: Pulse the reset line, or set it to a steady state.

Format: r <keyword>

Arguments: <keyword> is two letters to specify the normal (i.e. inactive) and asserted (active) states of the reset output. Using a keyword causes this command the reset signal to be set to the normal state based on the given keyword.

The allowable letters in the keyword are:

h - High (+5V)
l - Low (0V)
z - High impedance

The six allowable two letter keywords are:

hz hl lh lz zh zl

If <keyword> is omitted, the reset signal is pulsed to its active state for 100 to 150 milliseconds. An additional delay of 100 to 150 milliseconds is done before the next command is executed.

Comments: The <keyword> is saved in the configuration file and can be displayed using the ? command. During subsequent configuration initializations, the unit's reset line will be set to the inactive state.

The default state, if no 'r' command with keyword has been given, is zl, inactive high impedance, and active low.

The unit is not taken out of emulation mode during this operation. This command must be preceded by a successful 't' command (or configuration initialization) with the same port and unit selected.

Examples:

r — Pulses the reset signal from its current inactive state to the active state.

r lh — Set the reset signal low and define this as the inactive state. Subsequent r command with no arguments will pulse the reset signal high.

r zh — Set the reset signal to high impedance and define this as the inactive state. Subsequent r command with no arguments will pulse the reset signal high.

r hl — Set the reset signal high and define this as the inactive state. Subsequent r command with no arguments will pulse the reset signal low.

3.8 Misc Commands

h (help)

Purpose: Get on-line help.

Format: h <keyword>

Arguments: <keyword> is any command letter. Details about that command will be displayed.

Comments: If no arguments are given, a list of available commands and arguments is displayed. A file named "rep.hlp" is used, and must reside in the same drive and directory as the rep.exe file, in the current directory, or in your DOS path.

Examples:

h	Display a list of available commands.
help	Display a list of available commands. Same as typing: h.
h t	Display help for the 't' command.
h m	Display help for the 'm' command.

k (key, define function keys)

Purpose: Display or define a function key macro.

Format: k <key>,<command-string>

Arguments: The <key> can be:

1 through 10 to specify F1 through F10
a1 through a10 to specify Alt-F1 thorugh Alt-F10
s1 through s10 to specify Shift-F1 thorugh Shift-F10
c1 through c10 to specify Ctrl-F1 thorugh Ctrl-F10

The <command-string> is a list of RomEm commands (enclosed in quotes) to be executed with the selected function key.

Comments: An automatic return is signified by a "\r" at the end of the string. Otherwise the user must press return to execute the command**.**

If the first item in the string is "\r", then replacement strings typed on the command line before pressing the function key will be used in function key expansion, in place of the replacement flags "%0" through "%9". Replacement strings on the command line are separated by spaces. The text of one macro may be inserted into another by using %fn, %an, %sn, or %cn.

If the k command is invoked with no arguments, a list of the currently defined function keys is displayed. If only a <key> argument is supplied, the macro for that function is deleted (factory supplied macros can be changed but not deleted).

NOTES: A single backslash character has special meaning in string arguments, so a double backslash "\\" must be used to represent a single backslash.

The factory supplied function key macros are:

F1 "\rc%0 ; m\r"

To use this macro, type an address and then press F1. The COMPARE address will be set and monitor mode will be entered.

F2 "; as ; m\r"

Activate the SNAP feature and enter monitor mode.

F3 "; at m\r"

Activate the TRIGGER feature and enter monitor mode.

F10 "; qn\r"

This macro will QUIT the RomEm interactive software without updating the configuration file. If any configuration parameters have changed, they will be lost.

Examples:

k Display a list of the currently defined function keys.

k 4 Delete the definition for function key F4.

k a5,"t2764 ;fff ;l a:test.bin\r"

Sets function key Alt-F5 to define RomEm as a 2764, fill all locations with ff hex, then load a file named 'test.bin' from drive a:.

k 7,"\rl d:\\assembly\\%0.hex\r"

This will set function key F7 to load a hex file from the 'assembly' directory on drive d:. Not the use of the double backslash to represent a single backslash.

To use this macro, type the name of the file (without .hex extension), then press F7.

! (run DOS or a DOS command)

Purpose: Execute a single DOS command, or exit to DOS and remain resident.

Format: [command]

Arguments: none

Comments: If nothing follows the '!' on the input line, DOS is nvoked in interactive mode. Upon entering an "EXIT" command to DOS, command processing continues with the next input line with everything still intact.

If anything follows the '!' on the same input line, the remainder of the line is executed as a single DOS command and command processing continues immediately with the next command input line.

Examples:

!	Exit to DOS but leave the RomEm Plus software resident in RAM. Type exit to return to RomEm Plus software.
!dir	Run the DOS dir command, and return to RomEm Plus software.

= (calculator)

Purpose: Provide a means to compute values given in hexadecimal, decimal or ASCII.

Format: =<value>,<value>...

Arguments: <value> is any valid value argument as defined previously. The next line will shown the resulting value from that expression.

Comments: Allowed operators are:

+	add
-	subtract
*	multiply
/	divide
&	bitwise and
|	bitwise or
^	bitwise exslusive or
{	shift left
}	shift right
~	not (1's complement)

Examples:

=45+23 Computes the value of hex 45 plus hex 23, and displays the result in hex, decimal, and ASCII.

='A-#10 Computes the value of ASCII A minus decimal 10.

Note: Values can be assigned to $ registers.

? (display configuration)

Purpose: Displays current configuration and environment information in tabular form.

Format: ?

Arguments: none

Comments: This commands causes a table of current information to be displayed. The table contains one line of information for each current RomEm unit. Each line shows such pertinent information as port, unit, type, active features, etc.

The configuration file actually contains more information than is displayed using this command. For example, the arguments for the dump command are saved, but are not shown in this display.

Examples:

? Display current configuration information.

TROUBLESHOOTING

If you are having difficulty loading a file into RomEm, i.e. the t (Type) command reports that the emulator is not operational, or the l (Load) command is reporting errors:

1) Try a different parallel port if possible.

2) Use a good quality shielded cable no longer than 6 feet to connect RomEm to your parallel port. RomEm usually comes with such a cable. Longer cables or extensions may cause problems.

3) Try increasing the port delay with the p (Port) command.
In order to be compatible with the widest variety of systems and parallel ports, RomEm has a built in delay which it uses when writing to the RomEm unit. The default delay is 2. Occasionally, this may not be enough and you may need to try delay values of 3 or more, for example:

```
p 2,4                ← use port LPT2, with a delay of 4
```

The greater the delay, the more time it will take to download, but it is important to get it working first, then make it more efficient if you can. Once the delay has been changed, it will be stored in the rep.cfg configuration file for that port.

If RomEm loads with no errors, but your target system is not operating correctly:

1) It is always a good idea to start with a known good system by burning your code into an EPROM and trying it. This must work first before you attempt to make RomEm work in your system.

2) Try the ff (full folding) option on the t (type) command, for example:

```
t 27512,ff
```

Then reload your file. This will cause RomEm to load your data into all possible locations within the addressable range. If this works, there is probably a pin or two on your target system which is being held to a 'non-standard' logic level. This does not necessarily need to be corrected, but it will take longer to download files into RomEm.

3) You may need faster or different drivers to interface with your target. HCT interface devices are standard. Replacing these with LS, F or ACT logic may be required in some systems.

4) Faster RAMs in RomEm may be needed in your application. We have faster versions available. Contact us for more information.

5) In some cases a shorter target cable (IDC to DIP Plug) may be necessary. Contact us if you think you may need one of these and we will be happy to supply one.

LIMITED WARRANTY

MicroSystems Development Technologies, Inc. (MSD) warrants to the original purchaser of this product that it is to be in good working order for a period of twelve (12) months from the date of purchase from MSD or its authorized dealer. Should this product, in the opinion of MSD, malfunction during the warranty period, MSD will, at its option, repair or replace it at no charge, provided that the product has not been subjected to misuse, abuse, or non authorized alternations, modifications, and/or repairs.

Products requiring Limited Warranty service during the warranty period should be delivered to MSD with proof of purchase. If the delivery is by mail, you agree to insure the product or assume all risk of loss or damage in transit. You also agree to prepay shipping charges to MSD.

ALL EXPRESS AND IMPLIED WARRANTIES FOR THIS PRODUCT INCLUDING, BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE LIMITED IN DURATION TO THE ABOVE TWELVE (12) MONTH PERIOD.

If software is included, it is provided "as is" without warranty of any kind, either expressed or implied, including, but not limited to the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of these programs is with you. Should the programs prove defective, you (and not MSD or its authorized dealer) assume the entire cost of all necessary servicing, repair or correction. MSD does not warrant that the functions contained in the programs will meet your requirements or that the operation of the programs will be uninterrupted or error free.

UNDER NO CIRCUMSTANCES WILL MSD BE LIABLE IN ANY WAY TO THE USER FOR DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF, OR INABILITY TO USE, SUCH PRODUCT.
This warranty gives you specific legal rights and you may have other rights which vary from state to state.

