ROMX–2 and ROMX–2XL (256K–1024K) Operating Manual

9 MAY, 1991

CAUTION!

<u>Do not attach the Emulator to your Target System until you read and understand the cautions mentioned in Chapter 2. If you attach the Emulator improperly or apply 120VAC across the Emulator, you may damage the Emulator and possibly your Target system and the Host computer.</u>

Table of Contents

Chapter 1

Introduction

Congratulations. You now have what we believe to be the most cost effective and advanced eprom emulator on the market today. The design philosophy used on the ROMX–2XL allows for simple future expansion of capabilities. All communications with the ROMX–2XL is in printable ASCII characters and it supports Intel hex formats as well as simple block formats. Additionally, the ROMX–2XL supports the MCS–86 extended hex format. Split loads can be done with 2 ROMX–2XLs for use in a 16 bit data path. Resident features include facilities for formatted device listings, menu driven device selection, and more.

The Model ROMX–2XL is intended to be used at 19,200 baud and must use the CTS/DTR handshake lines. XON/XOFF is supported from the computer to the ROMX–2XL so you can use control–S and control–Q to control the ROMX–2XL listing.

A program is supplied with the ROMX–2XL called ROMX. ROMX will allow Upload/Download in either binary or Intel Hex format at up to 19,200 baud on an IBM PC/XT/AT/PS–2 type computer. Other computers may be used for upload and download, taking into account the baud rate and handshakes used.

Other programs are available separately with the Model PTK–1 disk. Provided with that disk are programs to convert between Motorola and Intel Hex and Binary, split between 16 and 32 bit data buses, Binary Edit a file, Uncase source code, Strip source code, and others. Call for price and availability.

Used in conjunction with any terminal or computer with an RS–232 port, the ROMX–2XL is capable of Emulating 1K x 8 through 128K x 8 parts—2758,2716,2516,2532,2732,2564,2764,68764,68766,27128, 27256, 27512, and 27010. These are 24, 28 and 32 pin parts.

In this manual there is no distinction made between the various levels of the ROMX–2 and ROMX–2XL. They both operate the same way with the same commands. The difference is in the part types supported. If there are operational differences, they are so noted.

Underlined text in the following paragraphs indicates new parts. No ROMX will support TMS2716 or 2708 (3 power supply parts) without modifications to eprom socket.

ROMX–2 Original	ROMX–2XL–256K	ROMX–2XL–512K	ROMX–2XL–1024K
—	2508	2508	2508
—	2758	2758	2758
2516	2516	2516	2516
2716	2716	2716	2716
2716B	2716B	2716B	2716B
27C16	27C16	27C16	27C16
—	2532	2532	2532
—	2532A	2532A	2532A
2732	2732	2732	2732
2732A	2732A	2732A	2732A
2732B	2732B	2732B	2732B
27C32	27C32	27C32	27C32
—	2564	2564	2564
2764	2764	2764	2764
2764A	2764A	2764A	2764A
27C64	27C64	27C64	27C64
—	68764	68764	68764
—	68766	68766	68766
27128	27128	27128	27128
27128A	27128A	27128A	27128A
27C128	27C128	27C128	27C128
27256	27256	27256	27256
27256A	27256A	27256A	27256A
27C256	27C256	27C256	27C256
27C256A	27C256A	27C256A	27C256A
—	—	27512	27512
—	—	27C512	27C512
—	—	—	27010
—	—	—	27C010
—	—	—	27C1000
—	—	—	27C1001

Chapter 2

Getting Started Quickly with the ROMX–2XL Eprom Emulator

The ROMX–2XL eprom emulator is very simple to operate. The ROMX interface program will allow you to transfer either Intel Hex or Binary files. ROMX is the program used to control the ROMX–2XL. By attaching the RED or GREEN halt clips to the target you can control the target system with ROMX. When we say target system, we mean the device to which you have the ribbon cable from the ROMX–2XL attached.

This chapter concerns people who will be using the ROMX–2XL with the ROMX software on an IBM PC/XT/AT or compatible type computer. Others should read chapter 6 on using the ROMX–2XL with other computers first, then come back here.

When you unpack your ROMX–2XL, you should be careful not to lose anything. The software may be packed separately between 2 sheets of cardboard. Don't throw anything away until you are sure that you have found everything. A typical package consists of:

1—ROMX–2XL Eprom Emulator

1—ROMX program disk

1—Ribbon cable assembly

1—User Manual with warranty card (please send right away)

1—Wall charger type power supply

1—RS–232 cable (Optional)

CAUTION!

The first thing you should do is make sure that all systems you are connected to have a **common ground!!!** This is generally accomplished if the target system has a 3 wire grounded plug and is plugged into the same AC source as the host computer. If the target system has no power transformer, or has a non–isolated power supply, WATCH OUT! Your warranty is voided if you put 120 Volts across the emulator and you may also damage the host and target systems as well!

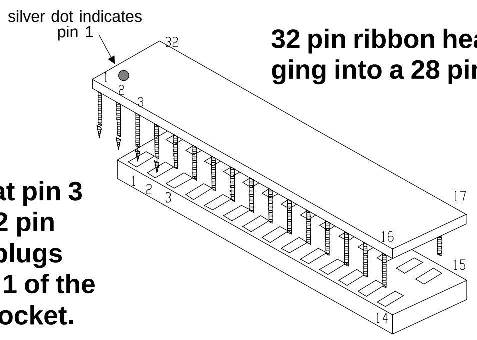
Next, attach your RS–232 cable between the emulator and the COM port that you will be using. If you didn't get a cable from us (purchased as an option), look at chapter 7 on making a cable to be sure that the one you are going to use is compatible. Don't get the cable ends confused. A RS–232 channel on an IBM/compatible is always a MALE DB connector (9 or 25 pin). Make sure that you attach the cable to a SERIAL port and not a printer or other type port.

Once you have the RS–232 cable connected, you can apply power to the ROMX–2XL.

Plug the mini–phone plug into the mini–phone jack. Plug the wall transformer into a 120V AC source. When you have applied power, it is possible that the emulation led on the ROMX–2XL will come on shortly afterwards if the emulator was in the autoemulate mode.

Copy the original disk that came with your emulator to a work disk or your hard disk and store the original in a safe place. On the copy disk, run the RINSTALL program. Type in "ROMX" in response to the question about which program you wish to install, and select 19,200 baud and the COM port that you will be using. Following the directions from RINSTALL, complete the installation and you will be returned to DOS. You only need to run RINSTALL once. Romx is the program that you will normally run to communicate with the emulator. Run ROMX now and the emulator will respond with a prompter indicating an eprom type. If there is a small "e" as the first letter of the prompter, the emulator is in the emulation mode and the only command it will recognize is **TS** for toggle stop. **Issue the TS command now.**

At this point you can connect the ROMX–2XL to your target system. It doesn't matter if the power is applied to the ROMX–2XL or not when you connect it to the target system. But you should NOT have power on the target system when you plug the ribbon cable into it! The 34 pin IDC connector for the ROMX–2XL side of the ribbon cable is keyed so you can't plug it in backwards. The dip header socket, like an eprom, can be plugged in backwards because it is not keyed. Be careful to plug the connector in right side up, because if you get it reversed you may damage either the target system and/or the ROMX–2XL. The silver dot at one end on top of the dip header is pin 1 of that socket. If you are using the 28 pin header cable in a 24 pin site, be sure that the header is justified to the bottom of the eprom socket! A 32 pin dip header plugs into a 28 pin socket in a like manner. See Figure 2.1.



32 pin ribbon header plugging into a 28 pin socket.

Note that pin 3 of the 32 pin header plugs into pin 1 of the 28 pin socket.

Figure 2.1

There are 2 low–profile sockets provided for you to attach to the dip header plug. These will protect the dip header from bent and broken pins. In the case of the 24 pin low–profile socket, plug it either directly to the 28 pin dip header or to the 28 pin low–profile socket. When you plug the 24 pin socket to the cable, make sure that the socket is justified towards the ground end (the end AWAY from the silver dot). This means that pin 1 and 2 and 27 and 28 of the dip header will be “hanging” in the air. The end of the 24 pin low–profile socket towards the silver dot is still pin 1 of the 24 pin low–profile socket. This is similar to figure 2.1. There are 24 pin dip header ribbon assemblies optionally available.

Once the header is plugged to the target system, you can connect one of the HALT lines. The RED clip is for high true reset lines and will be asserted when the emulator is in the stopped mode (not emulating). The GREEN clip is for low true reset lines. Connect the appropriate clip to the reset circuitry on the target system. If you are using an 8051 processor, use the RED clip. Make sure you attach the clips to the reset line and not Vcc or Vss by accident. When the ROMX–2XL is given the **TE** command or when power is applied with the AUTOEMULATE mode, the HALT and –HALT lines will tri–state to allow the target system to re–boot. You don’t have to use this feature, but we suspect you will want to.

If you are **NOT** using the halt lines, you should load the emulator first and issue the **TE** command so that the ROMX–2XL is emulating before the power is applied the target system. This is so the target system can reset and boot properly.

Now that you have the ROMX–2XL connected and you are in communication with it (with ROMX), you are ready to begin emulation. First you must select the type of part that you want to emulate. This is done by entering a "**m<cr>** " to the emulator from the prompt. Then simply enter the letter corresponding to your choice. Now you are ready to load the program of your choice into the emulator. You load the emulator by issuing a control–F (represented by **^F** later in the manual—this means to hold down the control key and strike an F). This will display a message (from ROMX) to "Enter Command Line —>". You can then type in the name of the file you wish to send to the emulator. You can also enter options after the filename by typing a left bracket ([). Once your file has been loaded into the emulator, You begin emulation by issuing the **TE** command from the emulator command prompter. This will turn on the emulation LED. A **TE** command will cause the HALT and –HALT lines to become tri–state; the RED and GREEN clip become high impedance. If you connected the proper halt lead to your target's reset line, the target should be up and running at this point.

This first example is the simplest way to use the emulator. At the emulator "command prompter" type **^F** and the emulator will respond with "Enter Command Line —>". Type the name of the file and when you strike the "enter" key and the file will be transferred to the emulator. It doesn't matter if you put an extension or not, the ROMX program will assume the extension to be "**.HEX**". Once the file is loaded, you are returned to the emulator command prompter. From there type **TE** to begin emulation.

```
27256> ^F
Enter Command Line —>filename<cr>
(display messages)
27256> TE
e27256>_
```

The following is an example session using two options on the command line. The **%** option tells ROMX that the file that you are using is a binary file. If you had not used this option, ROMX would have assumed an Intel hex file format and a ".HEX" extension to the filename.

The second option specifies a range of bytes within the file to send. If you had left this option off, the whole file would be transferred.

```
e27256> TS
27256> M<cr>

EPROM Emulation Menu
B – 2716
C – 2732
E – 2764
F – 27128
Z – 27256

Enter Selection —>B

2716> <^F>
Enter Command Line —>TEST.BIN   [%,@0-7FF<cr>

(beeps here when done)
2716> TE
e2716>_
```

Now to exit ROMX press **^C**.

Other options are available for use on the command line. Please see the next chapter for a complete list of the commands. Also note that you can accomplish all the same tasks as the above example from the DOS prompt. The line to enter would be:

ROMX TEST.BIN [%, MB, @0-7FF, TS, TE< cr> .

One more option that you will probably want to use is the **TA** option that will cause the emulator to automatically go into the emulation mode after power up.

NOTES

NOTES

Chapter 3

Commands

Hardware lines are provided for a signal to reset or halt the target system, both high true (red clip) called HALT and low true (green clip) called –HALT. Whenever the emulator is in the emulation mode (emulation led lit) then prompter will also indicate this with a small “e” in front of the eprom size being emulated. This also means that the HALT and –HALT lines are tri–state. Whenever the emulator is not emulating, the HALT and –HALT lines are being driven to Vcc and Gnd respectively.

P *Ascii Block Program*

Sending a “P”, followed optionally by an ascii–hex address, and a valid delimiter puts the ROMX–2 into the program mode. Once in the program mode, ascii–hex data to be programmed is sent. The data may be a continuous stream or the bytes (groups of two hex characters or nibbles) may be separated by valid delimiters. The program mode is terminated upon the receipt of an ASCII dollar sign, ’$’ or if an error occurs. Thus, the program command may be used to program one byte or a block of bytes at any given location. Valid delimiters are spaces, commas, carriage returns, line feeds, or dashes. This is the mode that ROMX uses to send “binary” data toROMX–2.

The following example illustrates how 33h and 23h are programmed to locations 444h and 445h in a 2716:

Example:

2716> **P444–33 23$**

2716>_

[ready for next command]

: *Intel Hex Program .*

When in the command state, receipt of a colon is interpreted as the lead character in an Intel hex record. The ROMX–2 automatically enters the program mode and programs the data contained in the hex record at the address specified in the header of the hex record. The check sum is verified at the end of the hex record.

See the section on toggles and hex formats for clarification on how to program two ROMX–2's for use on a true 16 bit data bus. The segment base address register, maintained by the ROMX–2, is automatically cleared if an error occurs, the end record is detected, or if any other command is executed other than the Intel Hex command. ROMX–2 does not echo the characters as they are coming in after the first colon.

R *Block Read.*

The R command, followed optionally by beginning and ending addresses, causes the Model ROMX–2 to output a continuous string of ASCII–HEX characters between the specified addresses. If no addresses are specified, the ROMX–2 will output the entire contents of the selected part size. The R command may be aborted at any time by sending a dollar sign, '$', to the ROMX–2. The following uses the same eprom size programmed in the example of the "P" command. ROMX uses this command to read ram in the "binary" mode.

Example:

2716> **R444,445<cr>**

3323

2716>_

[terminated by cr,lf, followed by prompt]

Note: The R command is primarily for automated reading. If you execute the command line as shown in the above example, you will find that the data output overwrites the command line unless your terminal is in an auto line feed mode.

Example:

3323> **R444,445<cr>**

2716>_

[command line overwritten with data]

OI *Intel Hex File Output .*

The OI command has the same command syntax as the R command. It differs in that the ROMX–2 will output the ram contents as an Intel hex file, including the end record, between the specified addresses or if no addresses are specified, the entire selected eprom size. Again, the command may be aborted if desired with a dollar sign, '$'.

L *List Formatted.*

The L command outputs the data, between optionally specified addresses, in a formatted fashion similar to many dump utilities. If no addresses are specified, the entire contents will be listed and the command may be aborted with the dollar sign, '$'. Each line of the listing includes the beginning address in ascii–hex, sixteen data bytes in ascii–hex and the ascii representation of the data. Non printable bytes are replaced with periods in the ASCII representation field.

Example:

```
2716> L90,AF<cr>
0090   4845 4C4C 4FFF//FFAB 99FF     HELLO.//....
00A0   FFFF FFFF FFDD//FFFF FFFF     ......//....
2716> _    [prompter indicates end of command]
```

Note:The lines are shortened at the “//” to allow printing on this page. Unlike the R and OI commands, the L command will output a carriage return and line feed at the beginning of the listing. This is because the L command is primarily used when the host is functioning as a terminal and it would be be irritating to have the first line of the listing overwrite the command line.

M *Menu.*

The Menu command is used to select the device type you intend to work with. The current device type always becomes part of the command prompter. Sending an M<cr> causes a menu to be output, from which the desired device is then selected. If the code letter for the device is already known, then just send M<code>

and the device will be selected. Selecting a device clears the ram to FF's, as well as selecting the device pinout, and prompter.

Example:

```
2716> MC
2732>_
```

Example:

```
2732> M<cr>

2716  – B
2732  – C
2764  – E
27128 – F
27256 – Z

Enter Selection —>Z
27256>_
```

Notice that no carriage return (<cr>) is necessary to make the part selection, only when requesting a MENU.

After a power down, a power up will automatically restore the eprom type and code that was last used. Auto emulation occurs if the 8 bit checksum is 00. See TA command.

T *Toggle Commands.*

The toggle command is used as a prefix to a subset of commands. These commands are as follows:

TAoooo *EnableAutoEmulation.*

This command will change a byte at offset address "oooo" so that the 8 bit checksum of the part is "00". If the 8 bit checksum of a selected eprom is "00" on power up, ROMX–2 goes into "auto–emulation" on next power up as if you had done a "TE" command from the prompter.

If no offset is specified (TA< cr>) the last location in the emulator is adjusted.

Using the TN command after the TA command will show the 16 bit checksum to be “xx00”.

TE *Emulate Eprom*

This command puts the Model ROMX–2 into the emulation mode. The HALT and –HALT lines tri–state allowing the target system to reset and begin operation if you have attached one or both properly.

Most commands issued after a TE command will cause an error message to be issued, (*EM err @ xxxx) except for the TS command (STOP). The prompter will reflect the current state of the emulator. Emulation is indicated by a small “e” in front of the part number. ROMX–2 will automatically begin emulation on power up if the 8 bit checksum of the part is “00”.

TH *Toggle High Byte*

Puts ROMX–2 into the “HIGH split” mode. Any hex code or bytes sent to ROMX–2 will be “split” so that the ODD or HIGH bytes are loaded into ram. This is done by simply putting the bytes with an odd address (byte 1, byte 3, etc...) into sequential locations in ram. For a 16 bit data path you will have to use 2 ROMX–2’s (just as you would have to use 2 eproms). The prompter will indicate HIGH by a small “h” in front of the part number.

TL *Toggle Low Byte.*

Puts ROMX–2 into the “LOW split” mode. Any hex code or bytes sent to ROMX–2 will be “split” so that the EVEN or LOW bytes are loaded into ram. This is done by simply putting even bytes (byte 0, byte 2, etc...) into sequential locations in ram. A 16 bit data path would use 2 ROMX–2s, one in the TH mode and the other in the TL mode. The ROMX–2 that is loaded first should have the control over the HALT or –HALT lines so that it can RESET or HALT the processor of the target system. The prompter will indicate LOW by a small “l” in front of the part number. See the Advanced Examples in the *ROMX*chapter and *Getting Started Quickly*.

TNssss,eeee *Toggle 16 Bit Checksum*

Checksum Command. If no start and end addresses are specified (TN<cr>) the 16 bit checksum is generated for the entire part. This is the sum of all the data bytes added together without carry. A “TNssss,eeee<cr>” generates the checksum from the specified starting (ssss) and ending (eeee) locations. Use this command in conjunction with the TA command to generate compatible checksums for auto– emulation.

TR *Toggle High/low (reset To Normal)*

Exits the “split” mode to normal operation.

TS *Stop Emulation*

Stops ROMX–2 from emulation. It drives the HALT line to +5 volts and the –HALT line to ground to stop the target system and returns to the command state so more commands may be issued. Current limiting is in effect to 50 Ma on either line.

X Logon

An “X” causes the ROMX–2 to return the version number of the firmware installed.

$ Abort

A “$” causes the command in operation to be aborted. This has no effect on the TE command. ROMX–2 returns to the command prompter.

—NOTES—

Chapter 4

Using ROMX

Installation

ROMX is a high speed communication program which runs on IBM PC’s and AT’s. It allows flexible manipulation, transmission and reception of Intel HEX files and binary files.

On the ROMX program disk you will have AT LEAST 2 programs: ROMX.COM and RINSTALL.COM. Other files such as READ.ME files on the disk will contain other documentation. Just running the program may give you help or documentation. ROMX is the program used to communicate with the ROMX–2XL. RINSTALL is the program that you must run to install the serial drivers in ROMX so that you can communicate with the ROMX–2XL.

If you try to run the ROMX program without installing the serial drivers, it will tell you to run the RINSTALL program. Remember that the ROMX license is a single user license.

In the following examples, when you see something within a pair of angle brackets (<>), that indicates a command to strike a control key like enter (carriage return) <cr> control–F <^F> or the space bar <sp>. When you see a letter preceded by a carat, that means to hold down the control key and strike the letter—don’t hold down the letter, just hit it as if you were typing. In other words a ^ would mean to hold down the control key. A <^F> would mean to hold down the control key and type the letter F. DON’T hold down the F key, just the control key. Any menus displayed in these examples may or may not be similar to your version of ROMX–2 or ROMX–2XL.

The following instructions generally follow the format where bold characters are to be typed in by the user. Command lines generally include a prompter generated by the computer or the emulator. We generally assume that you have a hard disk and you have copied your programs to a directory called ROMX and have used the prompt command to indicate drive and path (prompt pg< cr>). This generally works the same way as if you were logged onto a floppy disk.

Insert the ROMX program disk in drive A: and copy the programs to your hard disk with:

```
C:\ROMX> COPY    A:*.*<cr>
```

This will copy all the programs on the ROMX disk over to the subdirectory that you are logged on to on your hard disk. If you don't have a hard disk, use DISKCOPY or COPY to the B: drive. Refer to the DOS manual for specific instructions on using the COPY command. The desired end result is a backing up of the original ROMX copy. Store the original program disk in a safe place.

Now you should insert the backup copy in the drive A: and/or go to the subdirectory where RINSTALL and ROMX are located.

You must first run the RINSTALL program to install the serial drivers for ROMX.

```
C:\ROMX> RINSTALL<cr>

Serial Driver Installation Program Version 9.06b
Copyright 1986, 1987  GTEK, INC.
All Rights Reserved,  worldwide.

Enter name of program to be installed —>romx<cr>

IBM PC/AT or compatible COM1 (IRQ4)
A – 19200 bps
B – 9600 bps
C – 4800 bps
D – 2400 bps
E – 1200 bps

IBM PC/AT or compatible COM2 (IRQ3)
F – 19200 bps
G – 9600 bps
H – 4800 bps
I – 2400 bps
J – 1200 bps

Z - no changes

Enter selection —>A
```

Select LPT# (1,2,3 or return for lpt1:) —>**<cr>**

Do you have a GTEK Super Serial Card on COM2: (Y/N)? —> **N**
Finishing Installation

After the copyright and version number appears, you are asked to select a letter which corresponds to the type of installation you wish to perform.

Most people will probably select the "A" selection which would result in ROMX being set up to communicate at 19,200 BAUD on computer serial port COM1:. The "F" selection would result in the same thing except COM2: would be the serial port selected. You should always select 19,200 baud if you are using ROMX. Press return for lpt1:. If you have a GTEK PCSS–8 Super Serial card in your computer you can then install the software to handle selecting the correct channel when ROMX is run. If you answer N to the SS card prompt then the installation is completed and you are returned to DOS.

IRQ4 is used in conjunction with an interrupt service routine for COM1: when ROMX is invoked if you installed it for COM1:. This is a hardware line on your PC to give the system an interrupt whenever a character is received. If you know that something else in your computer is using this hardware interrupt line, then you should use the other com line, which uses IRQ3 (COM2:).

IRQ3 is also used in the same manner for COM2: when ROMX is invoked if you installed it for COM2:. If you know something in your system uses IRQ3 for interrupts, then you must use the other com port.

Operation

ROMX is a "command driven program" as opposed to a "MENU driven program" which means that everything you do is done by entering a "command" on the command line instead of "selecting" the command from a menu. This makes the program very fast when you have learned what the commands are.

There are a 2 ways that commands may be given to ROMX:

1– From the PC or MS DOS command line.

2– From within ROMX.

Commands executed from DOS return to DOS upon completion. Commands executed from within ROMX return to ROMX upon completion. Command lines may be entered from within ROMX by depressing control–F (<^F>).

Examples:

C:\ROMX> **ROMX<cr>**

Enter ROMX and establish communication with the Emulator (assuming everything is connected properly).

C:\ROMX> **ROMX FILENAME< cr>**

Results in communication being established with the Emulator and sending FILENAME.HEX (Intel Hex Format) from the disk to the Emulator. When ROMX is through, you are returned to the DOS system prompt.

C:\ROMX> **ROMX FILENAME< sp> [options]< cr>**

Results in ROMX establishing communication with the Emulator, and then performing according to selected options. See valid options below. The <sp> is optional between the filename and bracket.

Programming the Emulator ram in binary or Intel Hex format or Reading the Emulator ram in the same formats may be accomplished by giving the proper options. Options are always enclosed in square brackets and separated by commas. Invalid commands result in an appropriate and descriptive error message.

Valid Options

R—read file. Read to disk file option. Default is program mode—no R. This will read an Intel Hex file to the disk. This is different from the ROMX–2XL "R" command.

%oooo—binary mode select option. The oooo is the offset within the binary file. If no oooo is specified then 0 (zero) is assumed. The oooo offset has no meaning when R is also specified.

@ssss–eeee—Bounds option. The ssss is the start address and eeee is the ending address. This option always specifies a location within the eprom. When used with an Intel Hex file, this also means to search the hex file for the appropriate load addresses (inclusive) to send to the programmer at the specified addresses. When used with the % option also specified, it means only the addresses within the eprom. If addresses are specified that are outside the eprom absolute addresses, say 0–7FF for a 2716, and you specified

F0800–F0FFF for boundaries, then the address that is used is modulo 7FF.

Mx—Menu option. Allows selection of eprom size by eprom type number.

Tx – toggle command. Allows selection of toggle commands like E, S, R, H, L, etc.

Examples

ROMX<cr> from the DOS command line establishes communication with the Emulator, and after log–on displays the Emulator Command Prompter, which is the currently selected eprom type.

```
C:\ROMX>  ROMX<cr>
Rom Emulator Interface Package Version 9.23
Copyright 1987  GTEK, INC.
All Rights Reserved,  worldwide.
I/O Driver Vers 2.07b - IBM PC/XT/AT
Port  – COM20:, 19200 bps
Printer port – LPT1:

GTEK, Inc.
ModelROMX–2XLV1.04
Copyright 1987
27256> m<cr>

EPROM Emulation Menu
B - 2716
I - 2532
C - 2732
J - 2564
E - 2764
K - 68766
F - 27128
Z - 27256
7 - 27512
= - 27010

Enter Selection —>b
2716>_
```

The Emulator is ready and waiting for a command at this point. Note that you don't have to hit a <cr> (enter) to make the command execute when you type a "b". You don't have to type "M<cr>" every time. You can also make the selection directly, if you know what it is. For instance you know that 27128 is an "F" selection, so you can type MF.

```
2716> MF
27128> TN<cr>
C000
27128>_
```

Results in the Emulator giving you a 16 bit addition of all the 8 bit bytes of the selected eprom, without carry. A blank 27128 gives you C000 for the checksum. Note that you had to hit <cr> to make the TN command execute, because you can specify starting and ending addresses also. (TNsssss,eeeee<cr>)

Valid command letters used with the control key are P, F and C. The ESCape key is also a valid command key, but you do not hold the control key down to press ESC. The ESC key is a valid control character already. The escape control command may also be obtained by pressing Control [(^[) on the IBM keyboard or by holding down the ALT key and entering 027 on the numeric keypad. Pressing and holding the Control key is represented by a carat and the letter that must also be pressed, eg. ^C.

The definitions of the CONTROL commands are:

^P—start sending / stop sending (toggle) data simultaneously to the printer.

^F—enter a command line. (Examples follow.)

^C –Abort most Emulator commands and return to the DOS command prompter or ROMX. This command will work even though you may be in the process of programming, reading, verifying, etc., an eprom.

ESC or ^[—Escape from program. This command is used as an alternative to control–alt–del and is not normally used. This is an EMERGENCY command and the results could be unpredictable.

Using ^F

```
2716> <^F>
```

Enter Command line —>**FILENAME [@0–1FF,TN,TE,TS<cr>**

The options on the command line are automatically sent in the correct order to the emulator. A **TS** command is done first to put ROMX–2XL into the “interactive” mode. **FILENAME.HEX** is opened and any hex data falling between the specified boundaries is sent. The checksum is then calculated between the specified addresses and displayed. ROMX–2XL then goes into the Emulation mode (**TE**).

The options are always set off by an opening square bracket ([) and the ending square bracket (]) is optional. Invalid commands result in an error message and a return to the emulator command prompter.

Definitions

Please note that the listed commands are generally passed on to the Emulator unchanged except for the order in which they appear in the command line. ROMX will send the commands specified to the Emulator in the following order:

1 – TS (stop or interactive mode) toggle command

2 – Menu command (Mx)

3 – Toggle commands (R,H,L)

4 – Program or Read (specified by R or No R)

5 – Checksum (TN)

6 – Emulation (TE)

Some commands, particularly the “R” command, work differently from the Emulator command “R”. The “%” and the “@” command are not valid commands for the Emulator except on the command line. They are used to give ROMX information, not the Emulator.

ssss = starting address, ascii–hex characters

eeee = ending address, ascii–hex characters

oooo = offset amount, ascii–hex characters

A delimiter is a dash (–) or 2Dh, a comma (,) or 2Ch, a space ( ) or 20h, a carriage return or 0Dh or a line feed 0Ah. Carriage return and line feed are represented by a <cr> or <lf> in text since they are not printable.

A filename is a valid DOS filename to be used by ROMX to look for a file on the disk. In case that a percent (%) sign was specified then the filename specified will be taken literally. In other words you must be explicit and give the extension of the filename also. If the percent sign was not specified then ROMX will automatically supply a .HEX extension and look for a .HEX even if you specified an extension of another type, or none. When used in batch files, use 2 percents (%%).

An extension (EXT) is a valid DOS extension for the filename in your directory. You are allowed to use any extension you wish here, (in the binary % option) and the data will be sent to the emulator unchanged. The EXT will only be valid when you have specified a percent sign (%) within the brackets, otherwise it will always be .HEX, regardless of what you really put there.

And Remember!

The effective addressing range of a device is determined by it's size. If a 2K byte device is being used, then it only has 11 significant address lines and only the lowest 11 bits of the address field are significant. Thus, as far as the Emulator is concerned, 000H is equivalent to 800H or F000H in a 2K device.

EXAMPLES

Given TEST.BIN is a binary file with 12093 decimal bytes (2F3DH)

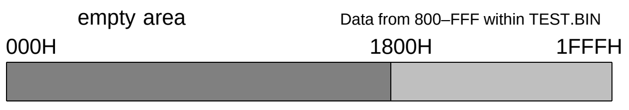


Figure 9.1

in it on the current drive\directory.

e2764> **^F**

Enter command line —>
TEST.BIN[%800,@1800–1FFF,TS,TE<cr>

ROMX–2XL stops emulating first (**TS**) then ROMX looks for a file called **TEST.BIN** on the disk, and when found start sending from relative offset 800H (**%**) from within TEST.BIN to locations 1800 to 1FFFh within the eprom size selected. When upload is complete, emulation begins again (**TE**). See Figure 9.1. A warning is issued if you happen to reach the end of the binary file before the specified number of bytes (@) are read (7FF – 0 + 1 = 800h bytes).

2716> **<^F>**

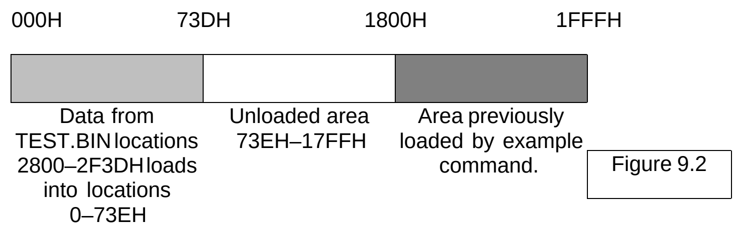

000H
73DH
1800H
1FFFH
Data from TEST.BIN locations 2800–2F3DH loads into locations 0–73EH
Unloaded area 73EH–17FFH
Area previously loaded by example command.

Figure 9.2

Enter command line —>**TEST.BIN[%2800,TS,TE<cr>**

ROMX–2XL stops emulating first then ROMX looks for a file called TEST.BIN on the disk and when found start sending from relative offset 2800H from within the TEST.BIN to locations starting at 0 through the amount of data that is left in the file. However, the program will terminate when it encounters the end of the file you are sending from since there are only 2F3D–2800+1=73Eh bytes left in the file TEST.BIN left to send. If you had specified @0–7FF for boundaries, you would have gotten an error message warning you that ROMX did not send 800H bytes to the emulator. See Figure 9.2.

Reading the ram to a disk file is accomplished with the 'R' option. You might want to do this if you have done any "patching" on the code in the ram. The "R" option always uses the Intel Hex format.

C:\GTEK> **ROMX filename[r,ts,te<cr>**

Results in reading the ram to the Intel hex disk file, FILENAME.HEX. You should probably get into the habit of specifying "ts" whenever you issue a command from DOS in case ROMX–2XL is in the Emulation mode. A "te" will cause ROMX–2XL to begin emulation again when done with the specified commands.

C:\GTEK> **ROMX filename[r,%,ts<cr>**

Results in reading the ram to a binary disk file whose name is FILENAME. (no extension was specified.). Note that an offset value included with the % has no meaning during a read operation. This case does not continue emulation after the read is done.

C> **ROMX [tn,mz,ts<cr>** or **< ^ F>** from within ROMX

Enter Command –>**[tn,ts,mz<cr>**

Results in selecting 27256 (note menu selection has side effect of resetting all toggles and clearing the ram) and calculating the checksum. You are returned to DOS afterward if the command was issued from the DOS command line. You remain in ROMX if the command was issued using <^F>.

Advanced Examples

C:\GTEK> **ROMX filename[mz,te,@4000–7fff,ts<cr>**

Select 27256, load ram with hex data from filename between 4000 and 7FFF inclusive into the same ram locations and emulate.

C> **ROMX<cr>**

e27256> **TS**

27256> **<^F>**

Enter Command Line —>**filename [tn<cr>**

27256>TN

2343

27256>TE

e27256>_

Line 1 enters the ROMX program. When initialized (power on boot), ROMX–2XL enters emulation automatically if the checksum of the part is 00. Stop it by typing TS. Code is then loaded to ram with <^F> and specifying the filename and the checksum option. When the checksum is displayed you are returned to the command prompter. From there type TE to begin emulation. If you don't have the HALT or –HALT lines hooked, you will have to reset the target system or apply power.

Another Example:

2716> **^F**

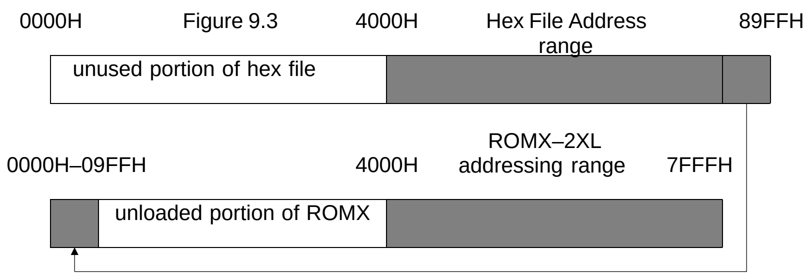
Enter Command –>**filename [MZ,TE,@4000–89FF<cr>**

2716>MZ

(selects 27256, loads ram)

27256>TE

e27256>_



The above has the same effect as the previous example except that the specified end address is greater than the size of the Eprom. When that happens the "extra code" above 7FFF "wraps around" to the beginning of the Eprom. See Figure 9.3.

This can happen with any hex file that is larger than the eprom size and you don't specify any boundaries with the "@" option.

Using ROMX with 16 bit data paths and 2 com ports.

We assume 2 ROMX–2XL's; one hooked to com1: and one hooked to com2:. The unit hooked to com1: has its HALT or –HALT clips hooked to reset on the target system. The first time, install 2 copies of ROMX:

C> **copy romx.com romcom1.com<cr>**

C> **ren romx.com romcom2.com<cr>**

C> **rinstall<cr>** —— use romcom1 filename, com1:

C> **rinstall<cr>** —— use romcom2 filename, com2:

C> **romcom1 filename.hex [ts,mc,tl<cr>**

C> **romcom2 filename.hex [ts,te,mc,th<cr>**

C> **romcom1 [te<cr>**—— allows target system to boot.

To repeat, start at the first use of romcom1 again (5th line), since you have installed for com1: and com2: already. Line 5 causes ROMX–2XL to halt and set the LOW byte mode, while loading code. This unit must have the HALT or –HALT lines attached to the target system. Line 6 causes the second ROMX–2XL to halt emulating, set the HIGH byte mode, load the file into ram and then begin emulation. The HALT and –HALT lines are not hooked into the target system since the first ROMX–2XL is controlling the target system reset. Line 7 then causes the first ROMX–2XL to tristate it's HALT and –HALT lines so the target system can begin operation.

—NOTES—

Chapter 5

Diagnostics

ROMX will handle all error codes coming from the emulator and display the appropriate message. ROMX operates the way that the diagnostics below describe.

1. All error codes to be issued by the Model ROMX–2 are preceded by an asterisk, '*'. This makes error trapping very easy.

2. All errors cause the ROMX–2 to be returned to the command state.

3. Errors are output on a real time basis, i.e., they are output as soon as they are detected.

4. Error codes include the address, (nnnn), where the error occurred.

ROMX–2 Error Codes

*DT ERR @ nnnn – DATA error.
Not valid hex data. Character is not a 0 through 9 or A through F.

*CS ERR @ nnnn – CHECK SUM error.
Data check sum does not add up to what was sent in the HEX record. Only applies to Intel hex format transfer.

*SN ERR @ nnnn – SYNTAX error.
Not a valid Emulator command. See commands.

*SL ERR @ nnnn – SELECT error.
No such menu code.

*EM ERR @ 0000 – EMULATION error.
Tried to perform a command while ROMX–2 was emulating. This can happen if you try to do a Menu command while the little “e” is indicated on the prompter, or other commands. You should do a “TS” first.

See chapter on ROMX for Warning and error messages from ROMX.

—NOTES—

Chapter 6

ROMX–2XL Interfacing Notes

You will find this chapter useful if you are not using the ROMX software. The Model ROMX–2XL is surprisingly easy to interface. If you have a PC, XT or AT, use the ROMX program. See chapter 7 for instructions on how to make a cable if you did not purchase one from GTEK. To use the ROMX–2 or ROMX–2XL without using the supplied ROMX program for a PC type computer, follow these guidelines below:

1. CTS/DTR hardware handshaking. The Model ROMX–2XL is configured as data communications equipment, which means that the DTR (Data Terminal Ready) line is an input to the ROMX–2XL which when pulled low forces the ROMX–2XL to stop sending. On the other hand, the CTS (Clear To Send) line is an output from the ROMX–2XL. When the CTS line on your computer goes low, you should stop sending to ROMX–2XL. See Specifications for the RS–232 Cable.

2. Xon/Xoff software handshaking. ROMX–2XL looks for XON/XOFF coming in from the computer to start/stop sending characters. You will find it useful when using some of the commands, to stop and start the data flow to your screen. XON and XOFF are the keyboard equivalents of control–Q and control–S respectively.

3. Please note that ROMX–2XL communicates at rates up to 19,200 baud. Set the baud rate by sending a break of 100 ms to ROMX–2XL and then restore the break. Wait more than 5 ms and then send an 80H character to ROMX–2XL at the baud rate you wish to send. ROMX–2XL supports 1200 to 19,200.

Automation Hints

When you automate the transfer of data from your computer to the ROMX–2XL, you should examine the echoed characters to see if an asterisk, “*” has been sent. If you receive one, it means that an error message will follow and that the Emulator will return to the command state. Any automation software should take this into account. The host need only trap the error message.

The effective addressing range of a device is determined by it's size. If a 2K byte device is being used, then it only has 11 significant address lines and only the lowest 11 bits of the address field are significant. Thus, as far as the ROMX–2XL is concerned, 000H is equivalent to 800H or F000H for a 2K device.

ROMX–2XL is in the command state after the prompter is sent. The prompter always ends with a '>'. You can use this character to let your program know that an R, OI, or L command has finished.

You should probably have one mode of operation where you communicate directly with the ROMX–2XL (turn your computer into a terminal). This will give you easy use of the L, P, and M commands.

—NOTES—

Chapter 7

Specifications

Dimensions (H x W x D):

1.00" x 2.7" x 5.6" (25.4mm x 68.58mm x 142.24mm)

Power Requirements:

9Vdc @ 500 Ma via wall charger type adapter.

Battery:

Ram permanently backed up by a lithium battery, part number BR1225–T2V

Interface Connector:

DB25S Configured as Data Communication Equipment. (DCE)

Data word size:

1 Start bit, 8 data, 1 stop bit, no parity

Baud Rate:1,200 to 19,200 baud

Weight:

.45 pounds (202g)

Operating Environment:

45 – 95 deg F. (7 – 35 deg C.)

5% to 95% non–condensing relative humidity.

EmulatorINTERFACE

The model ROMX–2XL has a DB25S connector configured as Data Communication Equipment (DCE).

ROMX–2XL DB25S connector:

Pin#	Direction	Function
1 (EG)	< —>	Equipment Ground.
2 (TXD)	< —	Transmit Data. INPUT
3 (RXD)	—>	Receive Data. OUTPUT
4 (RTS)		nc
5 (CTS)	—>	Clear To Send. OUTPUT +12V – can receive data
6 (DSR)	—>	Always + 10 v power on OUTPUT.
7 (SG)	< —>	Signal Ground. (logical ground)
20 (DTR)	< —	Data Terminal Ready. INPUT +12V – allowed to send data

Since the ROMX–2XL does not support RTS, you must jumper the computer end of the cable or use the CTS line from the ROMX–2XL to support those signals if you need to.

Making A Cable.

GTEK's part number for the cable for ROMX–2XL and an IBM is RSMDCE for a 25 pin cable and RSATDCE for a 9 pin cable.

IBM PC/XT and compatibles (25 pins)

Straight through: 1—1 (EG), 2—2 (TXD), 3—3 (RXD), 5—5 (CTS), 7—7 (SG), 20—20 (DTR).

IBM AT and compatibles (9 pins)

As before the "same name" connects to the "same name". The pin connections, however, are as follows:
DB9 DB25 (pin numbers)
1——8 (CD—Not supported by ROMX–2, 2XL or ROMX)

2——3(RXD)

3——2(TXD)

4——20(DTR)

5——7 (SG)

6——6 (DSR—Is +10V when ROMX–2, 2XL powered on)

7——4 (RTS—Not supported by ROMX–2, 2XL or ROMX)

8——5 (CTS)

9——nc

Pin 9 of the DB9 (RI) is not supported on ROMX–2 or 2XL.
If you are not using ROMX to communicate with the ROMX–2XL, you probably should attach pin 7 to pins 6 and 1 on the computer (9 pin) side of the cable, or pin 6 on the ROMX–2XL to pin 6 and 1 on the computer side. This should keep CD and DSR enabled.

—NOTES—

—NOTES—

Chapter 8

Intel Hex Format

Data Record

Byte #	
1	Colon (:)
2—3	Number of binary data bytes
4—5	Load address, high byte
6—7	Load address, low byte
8—9	Record type
10—x	Data bytes, pairs of ascii–hex characters
x+1—x+2	Checksum, pair of ascii–hex characters
x+3—x+4	Carriage Return, Line Feed

End Record

Byte #	
1	Colon (:)
2—3	Record length, must be “00”
4—7	Execution address
8—9	Record type (01)
10—11	Check sum
12—13	CR,LF

Extended Address Record (MCS–86 Hex Format)

Byte #	
1	Colon (:)
2—3	Record length, should be “02”
4—7	Load address field, should be “0000”
8—9	Record type, must be “02”
10—13	USBA (segment address)
14—15	Check sum
16—17	CR,LF

Start Address Record (MCS–86 Format)

Byte #	
1	Colon (:)
2—3	Record length, “04”
4—7	“0000”
8—9	Record type, “03”
10—13	8086 Code segment value
14—17	8086 Instruction pointer value
18—19	Check sum
20—21	CR,LF

The Start Address Record should not be allowed to be sent to the ROMX–2XL, because it can’t make any use of it.

The checksum is the two’s compliment of the 8–bit sum, without carry, of all the data bytes, the two bytes in the load address, and the byte count.

—NOTES—

Chapter 9

GHEX.EXE and STOHEX.COM

GHEX.EXE is a program provided for you to be able to convert a binary file into an INTEL.HEX file. The capability of transferring "binary" files is built–in to the ROMX.COM program, but you may want to use it for convenience to re–generate an Intel Hex file after using Debug to modify an Intel Hex file.

General usage is:

C> **GHEX filename.ext<cr>**

OR

C> **GHEX filename.ext offset<cr>**

Offset is a 16 bit ascii–hex number that specifies where you want your code to begin in the HEX file.

C> **GHEX filetest.bin<cr>**

Will result in an INTEL.HEX file being created on your disk by the name filetest.hex. The load addresses begin at 0000H since no offset was specified. GHEX does not destroy the input file.

C> **GHEX filetest.bin AA55<cr>**

Will result in an INTEL.HEX file being created on your disk by the name filetest.hex, just like before except the load addresses start at the offset address of AA55H in the file.

If your binary file is not an exact multiple of 128, GHEX will "pad" it with nulls until it reaches an even division of 128. This will result in your hex file containing up to 127 nulls at the end. If you don't want this to occur, pad your binary file in advance (using DEBUG) with "FF's" or whatever data you want there.

The version of GHEX that is supplied with ROMX–2XL will handle binary files up to 65536 bytes long. Any higher than that and GHEX will "wrap around" to 0000H for a load address without inserting a segment record type 02. You can do this yourself with a word processor. Refer to chapter 6 for the format. An example is shown below:

```
:10FFF000FFFF....FF11 (load addr=FFF0 type=00 cksum=11)
```

:020000021000EC (seg addr=1000 type=02 cksum=EC)

:10000000FFFF....FF00 (load addr=0000 type=00 cksum=00)

GHEX cannot use a path specification in the filename. GHEX does not return an errorlevel code for batch file processing.

STOHEX

STOHEX.HEX is a program that will convert Motorola S–records into Intel hex records. To use it, use I/O redirection.

Example:

C> **STOHEX < infilename.ext > outfilename.ext< cr>**

The left and right angle bracket (around infilename.ext) in the example above are explicit. The angle brackets around “cr” mean hit the “enter” key. The above example will read the infilename.ext and convert each record to Intel hex record format and then output them to outfilename.ext. Be sure to specify “.HEX” as the extension for the outfilename.ext in the above example.

If you have S2 and/or S3 records present (which means you have end records of S8 and S7), you may have to edit infilename.ext to make your end record be S9 instead of S7 or S8. STOHEX 1.0 does not recognize S7 or S8 as a valid “S” format record.

Optional Software Package PTK–1’s GHEX2

There is a new version of GHEX (GHEX2) supplied in an optional Programmers ToolKit utility program package available from GTEK. The new version is different in 4 ways. 1) GHEX2 can generate hex files larger than 64K. 2) GHEX2 can use path specifications in the filename. 3) GHEX2 returns an errorlevel code for batch processing. If the error return code is not 0, then some kind of fatal error occurred while GHEX2 was running and the batch file should take appropriate action. 4) GHEX2 can handle binary files up to 1 Megabyte in size.

—NOTES—

Chapter 10

Using DEBUG.COM

You may use DEBUG.COM (supplied with PC or MS–DOS) in conjunction with our GHEX.EXE to modify an INTEL.HEX file without worrying about the checksums in the INTEL.HEX file.

The following is a short tutorial to modify a 4K byte INTEL.HEX file with DEBUG. The procedure is to run DEBUG first.

```
C> DEBUG<cr>
–_
```

From the – prompter within DEBUG use the N command to specify the name of your INTEL.HEX file.

```
–Nfilename.HEX<cr>
–_
```

Use the L command to load the hex file with an offset (if it begins at 0000H). You must do this since if it starts loading at 0000H within the segment, it will overwrite your file control block at 5Ch.

```
–L   100<cr>
–_
```

The CX register now contains the number of bytes read into memory with an offset of 100. You may have to modify the CX register to properly reflect the correct number bytes you must write back to the disk. Remember that this is going to write from CS:CX when you issue the command.

```
–RCX<cr>
CX: 1000<cr>
–_
```

Your data is now loaded into the memory of the computer at offset 100H. Use the E command to modify the bytes you need to modify. An example of modifying locations starting at 0A55H with data is shown. Locations A55H through A57H contain FFH.

```
–EA55  01  02  03<cr>
```

```
–_
```

Now specify a new file name to write to the disk with since you can't use an extension of HEX with the file you are writing. You want to call it a BIN or IMG file instead since that is what the data really is anyway.

```
–NNEWFILE.BIN<cr>
–_
```

Now you can use the Write command to write the new data to the disk. DEBUG will write an exact image of CS:CX bytes to the disk starting at an offset of 0100H bytes.

```
–W<cr>
Writing 1000H bytes
–_
```

Now you can use GHEX to make it an INTEL.HEX file or use the BINARY transfer feature of ROMX to write the data to the blank ROMX–2XLram.

Optional Utility Package PTK–1

There are other utilities available from GTEK as an option to take the place of DEBUG and GHEX called the Programmer's ToolKit. In the toolkit you get a program called BED.COM that will take the place of DEBUG and is much easier to use. Other programs supplied will convert the Intel Hex file to binary to be edited. There is even a program called STOHEX.COM that will convert a Motorola hex file to an Intel Hex file. On the disk you get 7 programs:

1– BED.COM—A binary file editor.

2 – HEX2BIN.COM—An Intel Hex file to Binary file converter.

3 – STOHEX.COM—A Motorola file to Intel Hex file converter.

4 – SPLIT.COM—A 16 or 32 bit to 8 bit file converter.

5 – GHEX2.COM—A binary file to Intel Hex file converter.

6 – STRIP.COM—An ascii file bit 7 stripper program.

7 – UNCASE.COM—An ascii file case converter program.

All of these programs give you the capability of specifying file paths and getting errorlevel codes returned for batch file processing.

Appendix A

Warranties

Limited Warranty

GTEK, INC., warrants to the original purchaser of this GTEK, INC., product that it is to be in good working order for a period of 1 Year from the date of purchase from GTEK, INC., or an authorized GTEK, INC., dealer. Should this product, in GTEK, INC.'s opinion, malfunction during the warranty period, GTEK will, at its option, repair or replace it at no charge, provided that the product has not been subjected to misuse, abuse, or non–GTEK authorized alterations, modifications, and / or repairs.

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

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 1 Year Period. Some states do not allow limitations on how long an implied warranty lasts, so the above limitations may not apply to you.

Under No Circumstances Will GTEK, INC. 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. Some states do not allow the exclusion or limitation of incidental or consequential damages for consumer products, so the above limitations or exclusion may not apply to you.

This Warranty Gives You Specific Legal Rights, And You May Also Have Other Rights Which May Vary From State to State.

The limited warranty applies to hardware products only.

ROMX Software license Agreement

“This software is a proprietary product of GTEK, Inc. It is protected by copyright and trade secret laws. It is licensed (not sold) for use on a single micro–computer system, and is licensed only on the condition that you agree to this LICENSE AGREEMENT.” GTEK, INC. provides this program and licenses its use worldwide. You assume responsibility for the use of this software to achieve your intended results, and for the installation, use and results obtained from the software.

License

The Licensee may:

a. use the program on a single machine;

b. copy the program into any machine readable or printed form for backup or modification purposes in support of your use of the program on the single machine;

c. modify the program and/or merge it into another program for your use on the single machine (Any portion of this program merged into another program will continue to be subject to the terms and conditions of this Agreement.): and,

d. transfer the program and license to another party if the other party agrees to accept the terms and conditions of this Agreement. If you transfer the program, you must at the same time either transfer all copies whether in printed or machine–readable form to the same party or destroy any copies not transferred; this includes all modifications and portions of the program contained or merged into other programs.

You must reproduce and include the copyright notice on any copy, modification or portion merged into another program.

You May Not Use, Copy, Modify, Or Transfer The Program, Or Any Copy, Modification or Merged Portion, In Whole Or In Part, Except as Expressly Provided For In This License. If You Transfer Possession Of Any Copy, Modification Or Merged Portion Of The Program To Another Party, Your License Is Automatically Terminated.

Term

The license is effective until terminated. You may terminate it at any other time by destroying the program together with all copies, modifications and merged portions in any form. It will also terminate upon conditions set forth elsewhere in this Agreement or if you fail to comply with any term or condition of this Agreement. You agree upon such termination to destroy the program together with all copies, modifications and merged portions in any form.

ROMXLimitedWarranty

This Product Is Not A Consumer Product Within The Meaning Of The Uniform Commercial Code And Applicable State Law. The Program 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 The Program Is With You. Should The Program Prove Defective, You (Not GTEK, INC.) Assume The Entire Cost Of All Necessary Servicing, Repair Or Correction. Some States Do Not Allow The Exclusion Of Implied Warranties, So The Above Exclusion May Not Apply To You. This Warranty Gives You Specific Legal Rights And You May Also Have Other Rights Which Vary From State To State.

GTEK, Inc. does not warrant that the functions contained in the program will meet your requirements or that the operation of the program will be uninterrupted or error free. However, GTEK, Inc. warrants the diskette(s) on which the program is furnished, to be free from defects in materials and workmanship under normal use for a period of ninety (90) days from date of delivery to you as evidenced by a copy of your receipt.

Licensee herein acknowledges that the software licensed hereunder is of the class which inherently cannot be tested against all contingencies by Licensor. Licensee acknowledges Licensee's obligation to test all programs produced by the licensed software to determine suitability and correctness prior to use.

Limitations of Remedies

GTEK, Inc.'s entire liability and your exclusive remedy shall be:

1. the replacement of any diskette(s) not meeting GTEK's "Limited Warranty" and which is returned to GTEK, Inc. with a copy of your receipt, or

2. if GTEK, Inc. or the dealer is unable to deliver a replacement diskette(s) which is free of defects in materials or workmanship, you may terminate this Agreement by returning the program and your money will be refunded.

In No Event Will GTEK, INC. Be Liable To You For Any Damages, Including Any Lost Profits, Lost Savings Or Other Incidental Or Consequential Damages Arising Out Of The Use Or Inability To Use Such Program Even If GTEK, Inc. Has Been Advised Of The Possibility Of Such Damages, Or For Any Claim By Any Other Party.

Some States Do Not Allow The Limitation Or Exclusion Of Liability For Incidental Or Consequential Damages So The Above Limitation Or Exclusion May Not Apply to You.

General

You may not substitute, assign or transfer the license or the program except as expressly provided in this Agreement. Any attempt otherwise to sublicense, assign or transfer any of the rights, duties or obligations hereunder is void.

This Agreement will be governed by the laws of the State of Mississippi.

Should you have any questions concerning this Agreement, you may contact GTEK, Inc. by writing to:

GTEK, Inc.

Sales and Service

P. O. Box 2310

Bay St. Louis, MS 39521–2310

Service

For warranty service or non warranty service, contact GTEK, INC. at (601) 467–8048 to obtain an RMA (Return of Material Authorization number). We will need the serial number and date of purchase. Send the Emulator, freight prepaid to:

GTEK,INC.

RMA ####

399 Highway 90

Bay St. Louis, MS 39520

Be sure to include the RMA on and in the package so we will know what to do with it. Out of warranty service charges are determined on an hourly labor plus materials basis.

—NOTES—

Appendix B

Typical HALT and —HALT Hookup

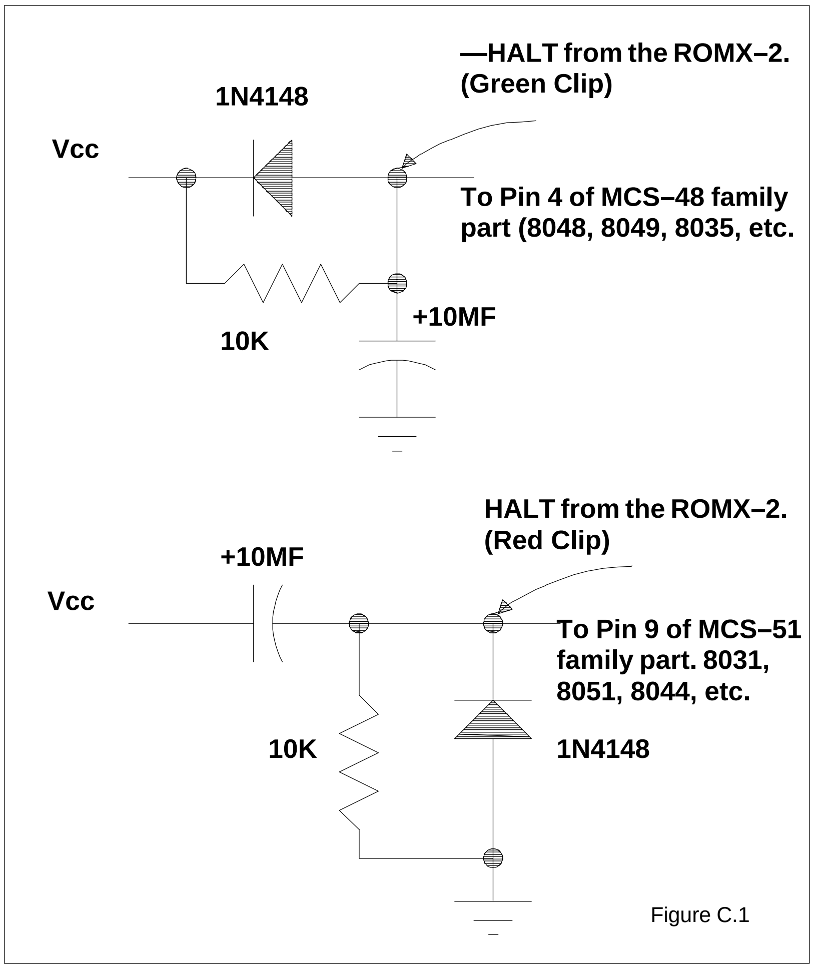


Figure C.1

Another typical —HALT hookup

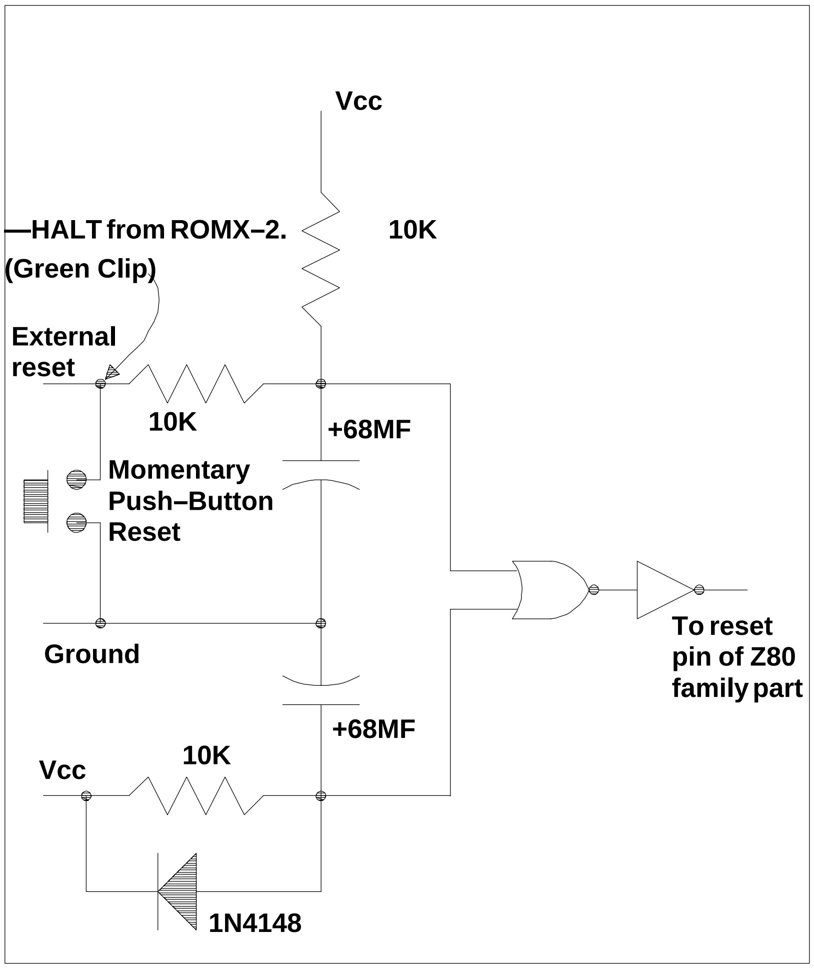


Figure C.2

Appendix C

GTEK is a registered trademark and PTK–1, Programmer’s ToolKit, PGMX, PGX, Model 7228, Model 9000, ROMX and ROMX–2 are trademarks of GTEK, Inc.

IBM is a registered trademark, and PC, XT, AT, PS/2 are trademarks of International Business Machines Corporation.

Intel is a registered trademark and Intelligent, MCS–86, QuickPulse are trademarks of the Intel Corporation.

MS–DOS is a registered trademark and DOS and QuickBasic are trademarks of Microsoft Corporation.

Motorola is a registered trademark of Motorola Inc.

Sidekick is a trademark of Borland, International.

—NOTES—

—NOTES—

—NOTES—

—NOTES—

—NOTES—