SERIAL EPROM PROGRAMMER USER'S MANUAL

For FIRMWARE VERSION 2.00

by Wayne Perrier
Perrier Technologies

Table of Contents

1. Introduction

The original manual, dated May 1990, was written for the original release v1.0x firmware. It was scanned & OCRd to bring it into a contemporary format. Firmware v2.00 exists, dated April 1993, which has extra commands and supports a few more eproms so this manual was updated with everything known regarding the v2.00 firmware.

The biggest additions and changes from the original manual are:

- Extra v2.00 firmware commands added
- Updated SB (Search Buffer) and SE (Search Eprom) topics with new, simpler, command structure.
- Added command Examples to each help topic
- Extra supported eproms added
- Added section for construction of serial interface cables
- Updated text capture windows from programmer
- Updated firmware version references to v2.00
- Changed field breakdowns for the various programmer download and upload formats
- Added ASCII picture of ZIF socket showing proper chip placement and usage
- Small typos corrected

Peter Schepers
IST, University of Waterloo
December 21, 2010

1.1 Overview

This document is the user's manual for the Serial Eprom Programmer, Firmware Version 2.00. The Serial Eprom Programmer connects to the RS-232C serial port of any computer or terminal and is thus quite versatile. Please read this manual before using the Serial Eprom Programmer.

The Serial Eprom Programmer has an on-board Intel 8031 microcontroller. This microcontroller runs firmware code contained in the supplied 27C256 eprom, and thus all of the intelligence is on-board the eprom programmer itself. All that is required of the host computer is the use of a terminal emulator/communications package to communicate with the Serial Eprom Programmer.

The Serial Eprom Programmer also has either 32Kb or 64Kb of on-board static RAM, which is used as a buffer between the eprom and the host computer. The user will typically download data into this RAM buffer from the host computer or terminal, and then program the data in the RAM buffer into the eprom.

Similarly, the user can copy data from the eprom into the RAM buffer, and then upload this data to the host computer.

1.2 Scope

This document describes the capabilities of the Serial Eprom Programmer, Firmware Version 2.00. A full description is given of the normal operation, and of the commands provided in Firmware Version 2.00. In addition, a description is given of the various download formats which are supported by Firmware Version 2.00.

1.3 Exclusions

This document does not attempt to:

- Describe the sequence of steps necessary to assemble the Serial Eprom Programmer.
- Detail features which may be included in future releases of the Serial Eprom Programmer firmware.
- Detail the hardware or describe the circuitry of the Serial Eprom Programmer.

1.4 Legal Waiver

No responsibility is assumed for damage to eproms or to the Serial Eprom Programmer due to carelessness or improper use by the user.

1.5 Serial Eprom Programmer Capabilities

This section very briefly outlines the capabilities of the Serial Eprom Programmer. A more detailed description of each of the commands may be found in Section 3.

As was stated in Section 1.1, the 32Kb or 64Kb RAM buffer is used rather extensively. Most of the commands involve the RAM buffer. For example, the user can download data into, or upload data from, the RAM buffer using the Download and Upload commands respectively. The user may also modify the RAM buffer using the Edit RAM Buffer, Fill RAM Buffer, or Move commands. The user also has the ability to search for a pattern in the RAM buffer using the Search Buffer command. The user may view the contents of the RAM buffer using the Display RAM Buffer command.

The user may also work with the eprom in the ZIF socket. The user must first select the current eprom type using the Type command. The contents of the eprom may then be viewed using the Display Eprom command. The user may also perform a blank test on the eprom, to ensure that it is blank (contains only FF data bytes), using the Blank Test command. The eprom may also be searched for a byte pattern, using the Search Eprom command. The user may then program bytes from the RAM buffer into the eprom, using the Program Eprom command, or copy bytes from the eprom into the RAM buffer, using the Copy command. As well, the user can compare that the contents of the eprom are the same as the RAM buffer, using the Verify command.

In addition to all of these commands, the user may request help, or perform diagnostics on the Serial Eprom Programmer hardware.

1.6 Firmware Updates

Firmware updates will be supplied as a separate eprom. The current firmware eprom would be removed and replaced with the update. Please complete the form enclosed in the manual and send it in. You will be informed of future firmware updates, which may add additional commands and/ or expand the range of eprom types that the Serial Eprom Programmer can operate on.

2. General Information

2.1 Host Computer / Terminal Configuration

The host computer or terminal RS-232C serial port must be set up correctly for proper operation of the Serial Eprom Programmer.

The serial port should be configured for 8 data bits, no parity, with 1 stop bit. Any of the following baud rates may be selected: 600, 1200, 2400, 4800, 9600, or 19200 baud. In addition, if a terminal emulator communications package is being used, then the backspace character should be set to be non-destructive. This means that when a backspace character is received by the host computer, the terminal emulator simply moves the cursor one character to the left, without erasing that character. This is necessary for proper operation of the Edit command.

Note that any computer or terminal used should have an 80 column display. The Serial Eprom Programmer uses a single carriage return and linefeed to begin new lines, when printing text to the RS-232C serial port.

2.2 Operational Environment

The firmware for the Serial Eprom Programmer is entirely command-driven. This means that there are no "menus" but rather commands are entered on a command-line much like most personal computers. These commands tell the microprocessor on the Serial Eprom Programmer to perform certain functions. The command line is indicated by the Serial Eprom Programmer Prompt, which appears as follows:

```
SEP >
```

Note that the cursor will appear where indicated and the user may then type in a command. Commands are terminated using a carriage return. Note that the command will be terminated automatically if the user attempts to type in more than 32 characters on the command line.

Valid characters which may be entered on the command line include the backspace, carriage return, and all printable ASCII characters. This includes characters with a hex value from 20H through 7EH. If an invalid character is entered then a single “beep” will be produced.

None of the commands are case-sensitive. That is, if one wishes to invoke the Help command, for example, one may type “h” or “H” as both will work.

2.3 Numbers

Many of the Serial Eprom Commands require or allow the user to enter numbers, either as addresses or data. All of these numbers are assumed to be specified in the hexadecimal base. Just the digits of the number should be specified; that is, one should not prefix the digits by a '$' or suffix the digits with an 'H'.

In this manual, however, hexadecimal numbers will be indicated by suffixing the digits of the number with a capital 'H'. For example, to specify 1234 hexadecimal, the string 1234H will be used.

2.4 RAM Addresses

Several of the commands allow the user to enter a RAM address. A RAM address is a hexadecimal number, of maximum length 4 digits. The minimum RAM address is 0000H. Depending on how much RAM is installed in the Serial Eprom Programmer, the maximum RAM address will differ. This table indicates the maximum allowed RAM address for the amount of RAM installed:

Amount of RAM Installed	Maximum RAM Address
32 Kb	7FFFH
64 Kb	FFFFH

When a RAM address is allowed or required for a command, the user may specify any address from 0000H through to the maximum address. If the user attempts to specify a RAM address which is greater than the maximum allowed RAM address, then the following error message appears:

```
Ram address is out of range
```

and the user is returned to the command line (Serial Eprom Programmer Prompt). Obviously, this message will never appear if 64Kb of RAM are installed because every 4-digit hexadecimal address will be in range.

2.5 Eprom Addresses

Several of the commands allow the user to enter an eprom address. An eprom address is a hexadecimal number, of maximum length 4 digits. The minimum eprom address is 0000H. Depending on the type of eprom being used, the maximum eprom address will differ. The following table indicates the maximum eprom address for the allowed types of eproms:

Eprom Type	Maximum Address
2716, 27C16	07FFH
2732, 2732A, 27C32	0FFFH
2764, 2764A, 27C64	1FFFH
27128, 27128A, 27C128	3FFFH
27256, 27C256	7FFFH
27512, 27C512	FFFFH

When an eprom address is allowed or required for a command, the user may specify any address from 0000H through to the maximum address for the current eprom type (the eprom type which was selected with the Type command). If the user attempts to specify an eprom address which is greater than the Maximum Address, then the error message:

```
Eprom address is out of range
```

will appear, and the user will be returned to the command line. Obviously, this message will not appear if the current eprom type is 27512 or 27C512, as all 4-digit hex addresses are valid for those types.

2.6 Auto-baud Feature

Immediately after power up or reset, the Serial Eprom Programmer performs some self-tests, and then waits for the user to enter a carriage return. This carriage return is used to determine what baud rate the host computer or terminal is using. When the baud rate has been recognized, the following power-up logo is displayed:

```
##############################################
#             Serial Eprom Programmer        #
#             (Firmware Version 2.00)        #
#                                            #
#                  Copyright (C)             #
#    Perrier Technologies Apr. 17th, 1993    #
##############################################
```

Note that the baud rate of the computer or terminal must have been one of: 600, 1200, 2400, 4800, 9600, or 19200 baud. If it wasn't, then the Serial Eprom Programmer will keep waiting for a carriage return at one of those baud rates. After the power-up logo appears, the Serial Eprom Programmer prompt will appear, as discussed in Section 2.2.

2.7 Indicator LEDs

There are two LEDs which indicate the state of the Serial Eprom Programmer. The first is the Power-up LED. This LED is illuminated when power is applied to the Serial Eprom Programmer itself.

The second LED is the Socket Active LED. This LED is illuminated whenever power is applied to the ZIF socket of the Serial Eprom Programmer. The user should not attempt to insert or remove an eprom from the ZIF socket when this LED is on, as the eprom may be damaged. However, when the Socket Active LED is off, all pins of the ZIF socket are at 0 volts, and it is safe to remove or insert eproms.

2.8 Orientation of Eproms in the ZIF Socket

The ZIF socket itself is 28 pins in size. Eproms of size 28 pins are placed in the ZIF socket, with pin 1 of the eprom corresponding to pin 1 of the ZIF socket. Some eproms, however, only have 24 pins. These eproms are the 2716, 2732, and 2732A. When these eproms are placed in the ZIF socket, pin 1 of the eprom should be placed in pin 3 of the ZIF socket. Thus, when using 24-pin eproms, pins 1, 2, 27, and 28 of the ZIF socket are unused.

Failure to follow these orientations will result in damage to eproms.

ZIF socket diagram

```
O <--- ZIF handle
|
|  +------------+
|==|1          28|
   |2          27|
   |3          26|
   |4          25|
   |5          24|
   |6          23|
   |7          22|
   |8          21|
   |9          20|
   |10         19|
   |11         18|
   |12         17|
   |13         16|
GND|14         15|
   +------------+
```

The handle on the ZIF socket must be in the UP position (sticking up) before inserting the eprom. Once the eprom is inserted, push the arm down backwards toward the programmer chassis. The chip ground pin must always go in pin 14 in the ZIF socket. Smaller eproms with only 24 pins (2716, 2732) don't use pins 1, 2, 27 and 28.

It is helpful to draw a ground indicator beside pin 14 on the chassis (beside the ZIF socket) so you remember where the ground pin on the eprom should go.

2.9 Programming Eproms

The idea when programming eproms is to first set up the RAM buffer with the data to be programmed into the eprom. This data does not necessarily have to appear in the same place in the RAM as it would appear in the eprom, because one may move data using form (2) of the Program Eprom command.

The RAM is set up by either downloading data from a host computer, editing or filling the RAM buffer using the Edit RAM Buffer or Fill RAM Buffer commands, Download command or a combination of all three of these commands. In addition, the Move command may be used to transfer data around inside of the RAM buffer. An additional command which is sometimes helpful is the Copy command, which copies data from the eprom into the RAM buffer.

Once the user is satisfied with the data in the RAM buffer, it is time to program the eprom. The correct eprom type is selected with the Type command. An eprom of this type is placed into the ZIF socket. The user may elect to perform a blank test on the eprom, using the Blank Test command.

Data is then programmed into the eprom using the Program Eprom command. The Program Eprom command automatically verifies the eprom at the end of the programming operation, to ensure that the data programmed was correct.

Note that on power-up the entire RAM buffer in the Serial Eprom Programmer is filled with FF.

3. Serial Eprom Programmer Commands

This section lists all the commands available with Firmware Version 2.00, and describes the operation of each command.

3.1 Help (H)

Syntax:

1. H <ret> or ? <ret>
2. H <command> <ret> or ? <command> <ret>

Example:

1. h : shows the general HELP window
2. ? sb : gets help on the Show Buffer command
3. h prog : gets help on the Program command

Description:

The Help command provides on-line help services to the user. This help describes the syntax and operation of the Serial Eprom Programmer commands.

If form (1) is used, a panel listing all the commands available is displayed. This panel appears as follows:

```
                        HELP PANEL
                        ----------
H or ? or HELP -- Invoke Serial Eprom Programmer Help
GENERAL -- General Help on using the Serial Eprom Programmer
VER     -- Display Firmware Version
DIAG    -- Enter Diagnostics Submenu to Test Hardware
DB      -- Display 256-byte Buffer Block
DE      -- Display 256-byte Eprom Block
FB      -- Fill Buffer with Byte
E       -- Edit Buffer
DL      -- Download Data to Buffer from Computer
T       -- View Current Eprom Type or Select New Eprom Type
B       -- Blanktest Entire Eprom or Subrange of Eprom
C       -- Copy from Eprom to Buffer
PROG    -- Program Eprom from Ram Buffer
SB      -- Search Buffer for Byte Pattern
SE      -- Search Eprom for Byte Pattern
U       -- Upload Intellec (Intel Hex) File from Buffer to Computer
V       -- Verify Eprom against Buffer
M       -- Move Buffer Block
CHKB    -- Perform 16-bit Buffer Checksum Calculation
CHKE    -- Perform 16-bit Eprom Checksum Calculation
```

```
OV      -- Perform Overlay Test on Eprom

Please enter the command for which help is desired or <ret> to exit:
```

Note that the user may now type in a command to receive help specific to that command. For example, if the user wanted help for the Display Eprom command, he or she would type in:

DE <ret>

If form (2) of the Help command is used, the help panel of the specified command is displayed, circumventing the general help panel listing all commands. For example, to obtain help about the "Type" command, one would enter:

```
H T <ret>
```

directly on the command line.

3.2 Display Firmware Version (VER)

Syntax:

VER <ret>

Description:

The Display Firmware Version command allows the user to view what the current firmware version is for the Serial Eprom Programmer. This enables the user to be sure that the firmware is up to date. When this command is entered, a panel similar to the following is displayed:

```
##############################################
#             Serial Eprom Programmer        #
#             (Firmware Version 2.00)        #
#                                            #
#                  Copyright (C)             #
#    Perrier Technologies Apr. 17th, 1993    #
##############################################
```

Note that this panel is identical to the power-up logo which was displayed immediately after the Serial Eprom Programmer recognized the baud rate being used.

3.3 Diagnostics (DIAG)

Syntax:

DIAG <ret>

Description:

The Diagnostics command allows the user to perform low-level tests on the Serial Eprom Programmer hardware. These tests are used primarily during the construction phase of the Serial Eprom Programmer, or when a hardware fault is suspected. When the user enters "DIAG", the Diagnostics Prompt appears:

```
Diag >
```

The user may then enter a command at the cursor position indicated. The user should first enter "H" or "?" to obtain Diagnostics Help:

```
                    Diagnostics Help Panel
                    ----------------------
H or ?                  -- Diagnostics Help
R                       -- Ram test
DATA                    -- Test Eprom Data Latch and Read Buffer
LO <byte>               -- Write byte to Address Bits A0 - A7
HI <byte>               -- Write byte to Address Bits A8, A9, A10 and A12
LED <SET,CLEAR>         -- toggle Socket Active LED
PIN1 <0,5,12.5,21>      -- Set ZIF pin 1 to specified voltage
PIN22 <0,5,12.5,21,25>  -- Set ZIF pin 22 to specified voltage
PIN23 <0,5,25>          -- Set ZIF pin 23 to specified voltage
PIN26 <0,5>             -- Set ZIF pin 26 to specified voltage
PIN28 <0,5,6>           -- Set ZIF pin 28 to specified voltage
PIN20 <SET,CLEAR>       -- Set or clear ZIF pin 20
PIN27 <SET,CLEAR>       -- Set or clear ZIF pin 27
ZERO                    -- Set all ZIF pins to 0 volts
X                       -- Exit Diagnostics

Note that the ZIF socket must not contain an eprom while running Diagnostics.
```

Several commands are possible. Always ensure that there is no eprom in the ZIF socket while running Diagnostics. This is because various pins of the ZIF socket may be set to high voltages for long periods of time during Diagnostics, and this may damage an eprom. The Diagnostics commands are as follows:

HELP - The Help for Diagnostics is available by entering either H or ?. Unlike for the Serial Eprom Programmer itself, help may not be requested for a specific Diagnostics command.

RAM TEST - This command is invoked by simply typing "R" at the Diagnostics prompt. This command performs a non-destructive RAM test using several patterns on the entire RAM buffer, and at the end of the RAM test, prints out a success or failure message.

DATA – This command tests the Eprom Data Latch and Data Read Buffer. This part of the circuitry is responsible for reading and writing data bytes to the eprom. The result of this test is either a pass or fail.

LO - This command writes the specified byte value to the lower 8 address bits of the ZIF socket. These are bits A0 through A7.

HI - This command writes the specified byte value to 4 of the upper 8 address bits of the ZIF socket. These are bits A8, A9, A10, and A12.

LED - This command enables the user to turn the Socket Active LED on (SET) or off (CLEAR). This has no effect on the voltages at the ZIF socket pins.

PIN1 - This command allows the user to set pin 1 of the ZIF socket to a specified voltage. The voltage specified must be one of: 0, 5, 12.5 or 21 volts. Pin 1 will remain at this voltage until explicitly set to 0, either through use of the PIN1 command, the ZERO command, or the X command.

PIN20 - This command allows the user to either set or clear pin 20 of the ZIF socket. Setting pin 20 corresponds to a TTL high at that pin, while clearing the pin corresponds to a TTL low. Either the word “SET” or the word “CLEAR” must follow the “PIN20”.

PIN22 - This command allows the user to set pin 22 of the ZIF socket to a specified voltage. The voltage specified must be one of: 0, 5, 12.5, 21, or 25 volts. Pin 22 will remain at this voltage until explicitly set to 0, either through use of the PIN22 command, the ZERO command, or the X command.

PIN23 - This command allows the user to set pin 23 of the ZIF socket to a specified voltage. The voltage specified must be one of: 0, 5, or 25 volts. Pin 23 will remain at this voltage until explicitly set to 0, either through use of the PIN23 command, the ZERO command, or the X command.

PIN26 - This command allows the user to set pin 26 of the ZIF socket to a specified voltage. The voltage specified must be one of: 0 or 5 volts. Pin 26 will remain at this voltage until explicitly set to 0, either through use of the PIN26 command, the ZERO command, or the X command.

PIN27 - This command allows the user to either set or clear pin 27 of the ZIF socket. Setting pin 27 corresponds to a TTL high at that pin, while clearing the pin corresponds to a TTL low. Either the word "SET' or the word "CLEAR" must follow the “PIN27”.

PIN28 - This command allows the user to set pin 28 of the ZIF socket to a specified voltage. The voltage specified must be one of: 0, 5, or 6 volts. Pin 28 will remain at this voltage until explicitly set to 0, either through use of the PIN28 command, the ZERO command, or the X command.

ZERO - This command clears all pins of the ZIF socket. It also turns the Socket Active LED off. The user is returned to the Diagnostics Prompt.

X - This command allows the user to exit Diagnostics, returning to the Serial Eprom Programmer Prompt.

3.4 Display RAM Buffer (DB)

Syntax:

1. DB <ret>
2. DB <RAM start address> <ret>

Examples:

1. db : Shows contents of the RAM buffer from address 0000H
2. db f0 : Shows contents of the RAM buffer from address 00F0H

Description:

The Display RAM Buffer command provides the user with the capability of displaying bytes of the RAM buffer in 256-byte blocks. The following is typical of the output when form (1) of the Display RAM Buffer command is invoked:

```
0000: 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78  .4Vx.4Vx.4Vx.4Vx
0010: 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78  .4Vx.4Vx.4Vx.4Vx
0020: 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78  .4Vx.4Vx.4Vx.4Vx
0030: 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78  .4Vx.4Vx.4Vx.4Vx
0040: 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78  .4Vx.4VX.4Vx.4Vx
0050: 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78  .4Vx.4Vx.4Vx.4Vx
0060: 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78  .4Vx.4Vx.4Vx.4Vx
0070: 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78  .4Vx.4Vx.4Vx.4Vx
0080: 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78  .4Vx.4Vx.4Vx.4Vx
0090: 12 34 56 78 12 34 56 78 12 34 56 78 FF FF FF FF  .4Vx.4Vx.4Vx....
00A0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
00B0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
00C0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
00D0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
00E0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
00F0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................

Press <.> to exit, <,> for previous block, or any other key for next block
```

Notice that the starting address of each "line" (a line is 16 bytes) is displayed, followed by the hex data for that line, followed by the ASCII representations of the data contained in that line. The user may hit a carriage return to display the next block, a comma to display the previous block, or a period to return to the command line.

If form (2) of this command is used, the block display starts at the RAM address provided by the user. Otherwise, operation is identical to form (1) of this command. The RAM address must be a 1 to 4-digit hexadecimal number.

Note that when displaying the RAM buffer, the display may wraparound through address 0000H. That is, if the RAM address of the next data byte exceeds the maximum RAM address, then the address counter will be reset to RAM address 0000H, and the displaying of data bytes will continue from there.

3.5 Fill RAM Buffer (FB)

Syntax:

1. FB <hex byte> <ret>
2. FB <RAM start address> <RAM end address> <hex byte> <ret>

Examples:

1. fb 30 : fills the entire RAM buffer with 30H
2. fb 0100 4000 44 : fills the RAM buffer from 0100H-4000H with 44H

Description:

The Fill RAM Buffer command allows the user to fill all or part of the RAM buffer with a byte value. For example, prior to programming an eprom, one would typically fill the RAM buffer with FFH, then download a file, and perform the programming operation.

If form (1) is used, the entire RAM buffer is filled with the specified hex byte. The hex byte is either a 1 or 2-digit hex number.

If form (2) is used, the user may specify a sub-range of the RAM buffer to be filled with the hex byte. Note that the <RAM start address> must be less or equal to the <RAM end address>. In other words, a "wrap-around" combination of addresses is not allowed.

Note that on power-up the entire RAM buffer in the Serial Eprom Programmer is filled with FF.

3.6 Edit RAM Buffer (E)

Syntax:

1. E <ret>
2. E <RAM address> <ret>
3. E <RAM address> <hex string> <ret>
4. E <RAM address> <"> <ASCII character string> <ret>

Examples:

1. e : Start editing RAM buffer at address 0000H
2. e a000 : Start editing RAM buffer at address A000H
3. e 1234 0c 12 : Place the two bytes 0C, 12 starting at address 1234H
4. e 2345 “insert : Place the ASCII bytes for ***insert*** starting at location 2345H

Description:

The Edit RAM Buffer command enables the user to selectively modify bytes in the RAM buffer.

If form (1) is used, editing starts at RAM address 0000H. A line of 16 bytes is displayed, similar to the following:

```
0000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

where 0000: is the start address of the line. The user may then perform several functions:

A. Type in hex digits over the currently-displayed hex digits. This will not cause memory locations to be modified, as all changes for the current line are temporarily stored in a 16-byte buffer.
B. Hit the space bar to move to the next hex digit to the right on the line.
C. Hit the backspace to move to the next hex digit to the left on the line. Note that the backspace is non-destructive so it simply moves the cursor to the left one digit.
D. Hit an exclamation mark to terminate editing of the current line without saving any of the changes made. The user is then returned to the command line.
E. Hit a period to terminate editing, saving changes to the current line.

F. Hit a comma to save changes to the current line, and edit the previous line.

G. Hit a carriage return to edit the next line, saving changes to the current line.

If form (2) of the Edit RAM Buffer command is used, operation is identical to form (1), except that the user is allowed to specify the start address of the line to be edited. This start address is a 1 to 4-digit hex number.

If form (3) is used, the user may place a series of hex values into the RAM buffer starting at the specified address. The hex string may be any combination of hex characters and space characters. Note that this format does not display a line as in forms (1) and (2), but instead directly places values into the RAM buffer.

Form (4) is similar in operation to form (3), except that instead of specifying hex values to be placed into the RAM buffer, the user specifies an ASCII string, the hex values of which will be placed into the RAM buffer at the starting address specified. For example, the command:

```
E 02 "TEST
```

will place the hex representation of "TEST" (i.e. the hex values 54, 45, 53, 54) into the RAM buffer starting at location 0002H.

3.7 Download (DL)

Syntax:

DL [word size] <format> [byte select] [RAM start address] [RAM end address] <ret>

Example:

1. dl xor : Download a Motorola Exorcisor (XOR) file
2. dl int 1000 1fff : Download an Intel Intellec (INT) file starting at 1000H

Description:

The Download command allows the user to download data from the host computer into the RAM buffer.

Note that the only parameter required is the <format> parameter; the other parameters are optional (as denoted by the square brackets). An example of a minimal Download command is:

```
DL INT
```

which performs an 8-bit Intellec download (accepts every data byte), starting at RAM address 0000H, to a maximum RAM address of FFFFH.

An explanation will now be given for each of the parameters of the Download command.

The [word size] parameter determines how many eproms the downloaded file will be split into. It is one of 8, 16, or 32. If the [word size] parameter is absent, then a default word size of 8 will be assumed. If the word size is 8, then all data bytes received will be placed into the appropriate position in the RAM buffer. If the word size is 16, then only one-half of the data bytes received will actually be accepted (the remainder being discarded). If the word size is 32, then only one-quarter of the data bytes will be accepted. Thus, a 32-bit download is required when the target system for the programmed eprom resides on a 32-bit bus (in other words, 4 eproms are required). Similarly, a 16-bit download is required when the target system has a 16-bit data bus.

The <format> parameter is the only non-optional parameter. It specifies the format that the downloaded data will be in. The format is one of:

- INT (Intellec)

- XOR (Exorcisor)
- HASC (Hex ASCII)
- BIN (Binary)
- DBIN (Dec Binary)

For details on each of these formats, see Appendix A.

The [byte select] parameter is optional. If it is not specified, then it is assumed to be 0. Note that if other parameters are specified (the [RAM start address] or the [RAM end address]), then the [byte select] must be specified. If the previous [word size] parameter was 8 (or not specified), then the byte select must be 0. If [word size] is 16, then [byte select] may be one of 0 or 1. If [word size] is 32, then [byte select] may be one of 0, 1, 2, or 3. For 16 and 32-bit downloads, [byte select] determines which half or quarter of the incoming data to keep. For example, if [word size] is 16, and [byte select] is 0, then data bytes at an even file address would be kept. If [byte select] were 1, then data bytes at an odd file address would be kept. If [word size] is 32, then if [byte select] is 0, data bytes whose file address mod 4 is 0 would be kept. If [byte select] is 1, then data bytes whose file address mod 4 is 1 would be kept. A similar criteria is used for [byte select] of 2 and 3. Thus, one can see that if a 32-bit download were being attempted (4 eproms being programmed), 4 separate downloads would have to be invoked, each one filtering out the data that it needed.

The [RAM start address] parameter is a 1 to 4-digit hex address, which indicates where in the RAM buffer it is desired to place the first accepted data byte. Subsequent accepted data bytes must not be placed in a RAM address below this. If this occurs, then the message:

```
Ram Start Address Underflow occurred
```

will appear. This can only occur with the Intellec and Exorcisor downloads where a given file can contain many record start addresses. Take, for example, an 8-bit Intellec download. Assume that the first accepted data byte should be placed at RAM address 0010H, and that [RAM start address] is 10H. The byte will be placed there. Now, if when processing a subsequent data record with a lower record start address, a data byte is to be accepted and placed at RAM address 0005H. A RAM underflow error will occur, as this address is less than the one specified. Essentially what happens with the Download command is that an offset is calculated from the address of the first received data record to the [RAM start address]. This offset is used in subsequent address calculations, and applied to incoming file addresses. If the result address is less than the [RAM start address], then an error occurs. For this reason, it is important that the file to be downloaded has its data records arranged in an ascending order, with the smallest record address appearing first in the file. If [RAM start address] is absent, then 0000H will be assumed as the default.

The [RAM end address] parameter is a 1 to 4-digit hex address, which indicates the last address where we will allow the placing of received data bytes. If during an address calculation (explained above), the resultant RAM address for a given data byte exceeds the [RAM end address], then an end address overflow error will occur. If the [RAM end address] parameter is absent, then a default equal to the last RAM address will be assumed (either 7FFFH or FFFFH).

Note that if any errors occur during the download, then an error message will be printed and a 3-second timeout will be activated. Incoming bytes will be discarded until a 3-second gap occurs after a byte. It is assumed that the download is over if no data has been transmitted during a 3-second period.

3.8 Type (T)

Syntax:

1. T <ret>
2. T <eprom type> <ret>

Examples:

1. t : View all the available EPROM types to choose one
2. t 27c256 : Set the EPROM type to 27C256

Description:

The Type command allows the user to View or select the current eprom type This will be the type of the eprom to be operated on in the ZIF socket.

If form (1) of this command is used, then a panel listing the valid eprom types is displayed:

```
Eprom type is one of the following:

2716,  27C16
2732,  2732A,  27C32
2764,  2764A,  27C64
27128, 27128A, 27C128
27256, 27C256
27512, 27C512

An eprom type has not yet been entered

Please enter new eprom type or <ret> to exit:
```

The user may then enter a new eprom type at the cursor (following “exit:”) or type a carriage return to go back to the command line. Note that the new eprom type entered must be one from the list displayed above. If not, then the message:

```
Please enter a valid eprom type
```

is displayed. In this circumstance, the previous eprom type is not modified. Similarly, if the user had just entered a carriage return in response to the "Please enter new eprom type or <ret> to exit: " prompt, then the previous eprom type would not have been modified, and the command line would have been returned to.

However, if the user had entered a new eprom type (e.g. 27256), then the next invocation of the "Type" command would produce the following message:

```
Eprom type is one of the following:

2716,  27C16
2732,  2732A,  27C32
2764,  2764A,  27C64
27128, 27128A, 27C128
27256, 27C256
27512, 27C512

Current type is: 27256

Please enter new eprom type or <ret> to exit:
```

Thus, the new eprom type would be displayed. This eprom type is now the current type, and thus, any eprom placed in the ZIF socket should be a 27256.

If form (2) of this command is used, the specified eprom type will be made the current type, thus circumventing the panel displaying all eprom types. An example of this form is:

```
T 27C512
```

This will make 27C512 the current type.

3.9 Display Eprom Block (DE)

Syntax:

1. DE <ret>
2. DE <eprom start address> <ret>

Examples:

1. de : Display EPROM contents from address 0000H
2. de 4000 : Display EPROM contents from address 4000H

Description:

The Display Eprom command allows the user to display bytes, in 256-byte blocks, of the eprom which is placed in the ZIF socket. Prior to using this command, the user must have selected the eprom type using the "Type" command. As well, the eprom must be correctly oriented in the ZIF socket for this command to work properly.

If form (1) is used, a 256-byte "block" of the eprom is displayed, starting at eprom address 0000H. The format of the display is as follows:

```
0000: 72 65 64 20 6F 72 20 3C 72 65 74 3E 20 74 6F 20  red or <ret> to
0010: 65 78 69 74 3A 20 00 0D 0A 49 6E 76 6F 6B 65 20  exit: ...Invoke
0020: 53 65 72 69 61 6C 20 45 70 72 6F 6D 20 50 72 6F  Serial Eprom Pro
0030: 67 72 61 6D 6D 65 72 20 68 65 6C 70 0D 0A 00 0D  grammer help....
0040: 0A 50 52 4F 47 20 3C 72 65 74 3E 20 20 20 6F 72  .PROC <ret>   or
0050: 0D 0A 50 52 4F 47 20 72 61 6D 5F 73 74 61 72 74  ..PROC ram_start
0060: 5F 61 64 64 72 20 72 61 6D 5F 65 6E 64 5F 61 64  _addr ram_end_ad
0070: 64 72 20 65 70 72 6F 6D 5F 73 74 61 72 74 5F 61  dr eprom_start_a
0080: 64 64 72 20 3C 72 65 74 3E 0D 0A 00 0D 0A 44 4C  ddr <ret>.....DL
0090: 20 49 4E 54 20 3C 72 65 74 3E 20 20 28 49 5E 74   INT <ret>  (Int
00A0: 65 6C 6C 65 63 20 64 6F 77 6E 6C 6F 61 64 29 20  e11ec down1oad)
00B0: 20 20 6F 72 0D OA 44 4C 20 58 4F 52 20 3C 72 65    or..DL XOR <re
00C0: 74 3E 20 20 28 45 78 6F 72 63 69 73 6F 72 20 64  t>  (Exorcisor d
00D0: 6F 77 6E 6C 6F 61 64 29 20 20 20 6F 72 0D 0A 44  ownload)   or..D
00E0: 4C 20 54 45 4B 20 3C 72 65 74 3E 20 20 28 54 65  L TEK <ret>  (Te
00F0: 6B 74 72 6F 6E 69 78 20 48 65 78 20 64 6F 77 6E  ktronix Hex down

Press <.> to exit, <,> for previous block, or any other key for next block
```

Notice that the starting address of a "line" (a line is 16 bytes) is displayed, followed by the hex data for that line, followed by the ASCII representations of the data contained in that line. The user may hit a carriage return to display the next block, a comma to display the previous block, or a period to return to the command line.

Note that the eprom address will wrap around; that is, if the address of the next data byte exceeds the maximum eprom address for the current eprom type, then the address counter will be reset to eprom address 0000H and the displaying of data bytes will continue from there.

If form (2) of this command is used, the block display starts at the eprom address provided by the user. Otherwise, operation is identical to form (1) of this command. The eprom address must be a 1 to 4-digit hexadecimal number.

3.10 Blank Test (B)

Syntax:

1. B <ret>
2. B <eprom start address> <ret>

Examples:

1. b : Blank test entire EPROM
2. b 0010 : Blank test EPROM starting at address 0010H

Description:

The Blank Test command allows the user to determine if the eprom in the ZIF socket is blank. This is typically done just before programming an eprom. Before using this command, the user must select the correct eprom type using the "Type" command. As well, the eprom must be correctly oriented in the ZIF socket for this command to work properly.

If form (1) of this command is used, a blank test is performed on the entire eprom, starting at eprom address 0000H. Blank eprom locations always contain FFH. If the eprom is not blank, a message indicating the contents and address of the first non-blank location is displayed:

```
Eprom is not blank at location 0000: 00
```

Of course, the address and data displayed may be different from that shown above. If the eprom is blank, the message:

```
Eprom is blank
```

is displayed since the entire eprom contains all FF's.

If form (2) of this command is used, the blank test starts at the address provided. The eprom is blank tested from that address onward. Messages similar to the ones described above are displayed, depending on the outcome of the blank test.

3.11 Copy From Eprom To RAM Buffer (C)

Syntax:

1. C <ret>
2. C <eprom start address> <eprom end address> <RAM start address> <ret>

Examples:

1. c : Copy entire EPROM to RAM buffer
2. c 1000 3fff 2000 : Copy EPROM from 1000H to 3FFFH to RAM address 2000H

Description:

The Copy command enables the user to copy bytes from the eprom into the RAM buffer.

If form (1) is used, the entire eprom is copied into the RAM buffer, starting at eprom address 0000H and RAM address 0000H. If the eprom is too large to fit into the RAM buffer (for example, suppose that there are only 32Kb of RAM installed in the Serial Eprom Programmer and we are attempting to copy a 27512 eprom into the RAM buffer), then a panel will be displayed indicating the parameters to be used in the copy:

```
Eprom type is:            27512
Eprom size is:            10000H
Ram start address is:     0000H
Eprom start address is:   0000H
Eprom end address is:     FFFFH
Number of bytes to copy:  8000H

Proceed with copy (Y/N)?
```

The user may then decide whether to proceed with the copy or abort.

If form (2) of the Copy command is used, the user is allowed to specify the block of the eprom to be copied into the RAM buffer, starting at the indicated RAM address. Again, if the block specified is too large to fit into the RAM, the panel will be displayed showing the modified parameters to be used in the copy. In addition, the panel will be displayed if the eprom end address is less than the eprom start address, or if the size of the block will not fit in the memory between the RAM start address and the end of RAM.

3.12 Program Eprom (PROG)

Syntax:

1. PROG <ret>
2. PROG <RAM start address> <RAM end address> <eprom start address> <ret>

Examples:

1. prog : Program EPROM with contents from RAM buffer
2. prog 1000 2fff 3000 : Program EPROM with contents of RAM buffer from 1000H-2FFF starting at EPROM address 3000H

Description:

The Program Eprom command allows the user to program bytes from the RAM buffer into the eprom. Note that before using this command, the user must have previously set up the eprom type using the "Type" command. The eprom must be correctly oriented in the ZIF socket to ensure correct operation.

Regardless of the form used to invoke this command, a panel similar to the following appears:

```
Eprom type is:              27256
Eprom size is:              8000H
Ram start address is:       0000H
Ram end address is:         7FFFH
Eprom start address is:     0000H
Number of bytes to program: 8000H

Proceed with programming (Y/N)?
```

Of course, the type of the eprom will not necessarily be 27256, but will be the type previously selected using the "Type" command. The user may then enter ’Y’ to program the eprom, or ’N’ to return to the Serial Eprom Programmer prompt.

If form (1) of this command is used, the entire eprom is programmed with the contents of the RAM buffer, starting at RAM address 0000H. Note that only the number of bytes required to fill the eprom are actually programmed. For example, if the eprom being programmed is a 2716, of size 800H bytes, then only RAM addresses 0000H through 07FFH are used to program the eprom. Addresses above 07FFH are not involved. If the size of the eprom is greater than the

size of the RAM buffer, then only a number of bytes equal to the size of the RAM buffer is programmed into the eprom.

If form (2) of this command is used, then the user can program a portion of the eprom with a specified block of the RAM buffer. Whenever this form is used, a panel similar to the one previously shown is displayed. Note that the parameters displayed may be different than those specified, as the Serial Eprom Programmer checks the incoming parameters to ensure that they are valid and make sense. If the user types 'N' to the prompt, then the programming operation does not occur, and the user is returned to the command line. If the user types 'Y', then programming occurs. The message:

```
Programming...
```

is displayed, and the Socket Active LED is lit to indicate that there is power applied to the eprom. Note that a blank test is not automatically performed by the Serial Eprom Programmer prior to programming. When the programming operation is complete, a Verify is performed on the eprom to ensure that the programming operation was successful. During this stage, the message:

```
Verifying...
```

is displayed. If the verify fails, the message:

```
The Verify operation failed at:
Ram Address:   xxxxH Contents: zzH
Eprom Address: yyyyH Contents: WWH
```

is displayed. Note that ww, xxxx, yyyy, and zz will not appear as shown, but instead will contain the actual locations and values where the Verify operation failed. If the Verify passed, then the message:

```
The programming operation passed
```

is displayed. Note that if the programming operation itself failed (before the Verify was reached), then the following message is displayed:

```
The programming operation failed at location xxxxH
```

where xxxx will be replaced with the actual address where programming failed.

Typical programming times are:

Eprom Type	Seconds to program
27C64	2.5
27C128	5

27C256	10
27C512	20
2764A	35
2764	45
27128A	70
27128	90
2716	103
27256	140
2732, 2732A	205
27512	280

3.13 Search RAM Buffer For Pattern (SB)

Syntax:

1. SB <search start address> <hex bytes> <ret>
2. SB <search start address> "<ASCII char> <ret>

Examples:

1. sb 0100 4c 42 : Search RAM buffer starting at address 0100H for bytes 4C, 42
2. sb 1000 “test : Search RAM buffer starting at address 1000H for string ***test***

Description:

The Search Buffer command allows the user to search for a single byte, a byte string or a text string in the RAM buffer.

Form (1) searches for HEX bytes starting from <search start address>. The hex bytes can be 1 or 2-digit hexadecimal numbers.

Form (2) searches for ASCII text bytes. This frees up the user from having to know the code for an ASCII character. The ASCII character must be prefixed by the double quote character. If the search pattern is found, then the following message is displayed:

```
The target string was found at address yyyyH
```

where yyyy is the address at which the first occurrence of the search pattern was located.

If the search pattern is not found, then the following message appears:

```
The target string was not found
```

3.14 Search Eprom For Pattern (SE)

Syntax:

1. SE <search start address> <hex bytes> <ret>
2. SE <search start address> "<ASCII char> <ret>

Examples:

1. se 1234 33 44 : Search EPROM from address 1234H for bytes 33, 44
2. se 2000 “test : Search EPROM from address 2000H for string ***test***

Description:

The Search Eprom command allows the user to search for a single byte, a byte string or a text string in the eprom placed in the ZIF socket. Note that the user must have previously selected the eprom type using the **TYPE** command, and that the eprom must be correctly installed in the ZIF socket for the command to work properly.

Form (1) searches for HEX bytes starting from <search start address>. The hex bytes can be 1 or 2-digit hexadecimal numbers.

Form (2) searches for ASCII text bytes. This frees up the user from having to know the code for an ASCII character. The ASCII character must be prefixed by the double quote character. If the search pattern is found, then the following message is displayed:

```
The target string was found at address yyyyH
```

where yyyy is the address at which the first occurrence of the search pattern was located.

If the search pattern is not found, then the following message appears:

```
The target string was not found
```

3.15 Upload (U)

Syntax:

U <RAM start address> <RAM end address> <file start address> <pause> <ret>

Examples:

1. u 0 7fff 0 1 : Upload an Intel Intellec file from RAM buffer address 0000-7FFFH starting at Intel record address 0000H, with an upload delay value of 1

Description:

The Upload command gives the user the capability of transferring data from the RAM buffer of the Serial Eprom Programmer to the host computer. This may be useful for more complex editing or examination of the data. The format of the data transferred is that of an Intellec file. See Appendix A.1 for details of this format.

The first and second parameters of this command specify the block of RAM buffer data that is to be transferred. All data bytes, from <RAM start address> through <RAM end address> inclusive, will be transferred. Note that the <RAM end address> must not be less than <RAM start address>, and that both of these parameters must be in the range of valid RAM addresses for the amount of RAM installed. These numbers are 1 to 4-digit hexadecimal numbers.

The third parameter specifies the start address of the first record in the Intellec "file" which is to be transferred. This parameter is a 1 to 4-digit hexadecimal number.

The fourth parameter, <pause>, provides a delay equal to <pause> milliseconds between Intellec records. This allows slower computers time to store the data to disk or time for whatever processing must be done. The <pause> parameter is a 1 or 2-digit hex number, which ranges from 00H through FFH.

Note that all parameters must be included for the Upload command. The following is sample output when the Upload command:

```
U 0 ff 0 0
```

is invoked:

```
:10000000123456781234567812345678123456 78A0
:10001000123456781234567812345678123456 7890
:10002000123456781234567812345678123456 7880
:10003000123456781234567812345678123456 7870
:10004000123456781234567812345678123456 7860
:10005000123456781234567812345678123456 7850
:10006000123456781234567812345678123456 7840
:10007000123456781234567812345678123456 7830
:10008000123456781234567812345678123456 7820
:10009000123456781234567812345678123456 0025
:1000A00000D700FF00FF000000F500FF00FF000088
:1000B00000FF00FF00FF000000FF00FF00FF000046
:1000C00000CD00FF00FF000000FD00FF00FF00006A
:1000D00000EF00FF00FF000000D500FF00FF000060
:1000E00000C600FF00FF000000CD00FF00FF000081
:1000F00000FD00FF00FF000000FZ00FF00FF000615
:100000001FF
```

3.16 Verify Eprom (V)

Syntax:

1. V <ret>
2. V <RAM start address> <RAM end address> <eprom start address> <ret>

Examples:

1. v : Verify entire EPROM against RAM contents
2. v 1000 2fff 0000 : Verify EPROM from address 0000H against RAM buffer address range 1000-2FFFH

Description:

The Verify Eprom command allows the user to verify that the contents of the eprom are the same as the RAM buffer. This command essentially compares the eprom with the RAM buffer and indicates where they differ. The parameters are structured in a manner identical to the "Program Eprom" command. form (1) of this command is used, then the entire eprom is verified against the RAM buffer, starting at eprom address 0000H and RAM address 0000H. A number of bytes equal to the size of the eprom are compared.

If form (2) is used, then the user may specify a sub-range of the RAM buffer to be compared to the eprom, starting at the user-specified address.

If either form is selected, a panel similar to the following is displayed:

```
Eprom type is:              27256
Eprom size is:              8000H
Ram start address is:       0000H
Ram end address is:         7FFFH
Eprom start address is:     0000H
Number of bytes to verify:  8000H

Proceed with Verifying (Y/N)?
```

The values listed in each line may differ, depending on the parameters the user typed in and the type of eprom which has previously been selected using the "Type" command. The user may now decide to either abort or proceed with the verify. Only the responses ’Y’ or ’N’ are accepted. If the user aborts, then the Serial Eprom Programmer prompt appears and the user is

returned to the command line. If the user proceeds with the verify, then it will either pass or fail.

If the verify passes, then the message:

```
The Verify operation passed
```

will appear. If the verify fails, then a message similar to the following is displayed:

```
The Verify operation failed at:
Ram Address:   0006H Contents: 45H
Eprom Address: 0006H Contents: 46H
```

Of course, the addresses and data displayed may be different than those shown above. The addresses indicated are the addresses of the first byte which differed. Form (2) of the Verify Eprom command may be used when the block of the eprom to be compared to the RAM buffer is not at the same address as the RAM buffer block.

3.17 Move (M)

Syntax:

M <source start address> <source end address> <destination start address> <ret>

Examples:

1. m 1000 2fff 4000 : Move RAM contents from 1000-2FFFH to address 4000H+

Description:

The Move command gives the user the ability to copy blocks of the RAM buffer into other areas of the RAM buffer. The <source start address> and <source end address> delimit the source block (the block to be copied). The source block is copied into the area indicated by the <destination start address>. The first byte of the source block (at address <source start address>) is copied into the first byte of the destination block (at address <destination start address>). Bytes are copied until the entire block has been copied. Note that the destination block can overlap the source block; this is allowable.

The source and destination blocks specified must fit into the RAM buffer. That is, there must be enough room between the <destination start address> to the end of the RAM for the block being copied to fit. If there is not enough room, then the following message will be displayed:

```
Destination Address was too high for block size
```

3.18 Checksum RAM Buffer (CHKB)

Syntax:

1. CHKB <ret>
2. CHKB <RAM start address> <RAM end address> <ret>

Examples:

1. chkb : Checksum entire RAM buffer
2. chkb 1500 1800 : Checksum RAM buffer from address 1500-1800H

Description:

The CHKB command performs a 16-bit (2-byte) checksum on all or a part of the RAM buffer. If form (1) is used, a checksum of the entire buffer (32K or 64K) is calculated. If form (2) of the command is used, a checksum is performed on the RAM buffer from the starting address up to and including the ending address. The <RAM start address> must be less than the <RAM end address>.

3.19 Checksum Eprom (CHKE)

Syntax:

1. CHKE <ret>
2. CHKE <eprom start address> <eprom end address> <ret>

Examples:

1. chke : Checksum entire EPROM
2. chke 2000 3fff : Checksum EPROM from address 2000-3FFFH

Description:

The CHKE command performs a 16-bit (2-byte) checksum on all or a part of the eprom. If form (1) is used, a checksum of the entire eprom is calculated. If form (2) of the command is used, a checksum is performed on the eprom from the starting address up to and including the ending address. The <eprom start address> must be less than the <eprom end address>.

3.20 Overlay Test (OV)

Syntax:

1. OV <ret>
2. OV <RAM start address> <RAM end address> <eprom start address> <ret>

Examples:

1. ov : Overlay test entire RAM contents against EPROM
2. ov 0400 0fff 1000 : Overlay test RAM contents from address 0400-0FFFH against EPROM address 1000H+

Description:

This command enables the user to determine if the data in the RAM buffer can be successfully programmed into the non-blank eprom in the ZIF socket. If successful, this saves the user from erasing the eprom first. If it fails, the eprom will have to be erased first before programming can continue.

If form (1) of the command is used, the entire RAM buffer is checked against the eprom contents, starting from address 0000H. A number of bytes equal to the size of the eprom is checked.

If form (2) of the command is used then only a portion of the RAM buffer is checked against the eprom. The check will start at <RAM start address> and <eprom start address> and continue up to and including <RAM ending address>. The value for <RAM start address> must be less than <RAM ending address>.

```
Eprom type is:              27256
Eprom size is:              8000H
Ram start address is:       0000H
Ram end address is:         7FFFH
Eprom start address is:     0000H
Number of bytes to verify:  8000H

Proceed with overlay check (Y/N)?
```

A. Download & Upload File Formats

This appendix details each of the five file formats available on the Serial Eprom Programmer Firmware Version 2.00 for upload and/or download. Any one of these formats can be used during invocation of the Download command (see Section 3.7). Only Intellec is supported when uploading from the programmer.

A.1 Intellec Format (download & upload)

The Intellec format is also generally known as "Intel Hex". This format, when displayed, consists of:

1. A start code (a colon character) ":"
2. The number of data bytes in an individual record, e.g. 1AH = 26
3. The address of the first byte of data in an individual record, e.g. 0000H
4. The record type. This is one of:
    a. 00H - Data Record (has data)
    b. 01H - End Record (last line, no data usually)
5. The data in bytes, e.g. 12 34 56 78
6. The checksum of an individual record, e.g. 28H

Consider the following set of Intellec records, which have a start address of 0000H, and an end address of 009BH. See how the description above matches up to the columns:

```
1 2   3   4          5                                                    6
- -- ---- -- ------------------------------------------------------- --
: 1A 0000 00 12345678123456781234567812345678123456781234567812345678 28
: 1A 001A 00 56781234567812345678123456781234567812345678123456781234 86
: 1A 0034 00 12345678123456781234567812345678123456781234567812345678 F4
: 1A 004E 00 56781234567812345678123456781234567812345678123456781234 52
: 1A 0068 00 12345678123456781234567812345678123456781234567812345678 C0
: 1A 0082 00 56781234567812345678123456781234567812345678123456781234 1E
: 00 0000 01                                                          FF
```

Calculation of the Intellec Checksum

Consider the following data record:

```
:01001A00568F
```

1. The start code and the checksum are removed: ":" and 8F
2. Five bytes remain: 01 00 1A 00 56
3. These are added together: 01 + 00 + 1A + 00 + 56 = 71
4. The total (71) is converted into binary: 0111 0001
5. The binary number is complemented: 1000 1110 (8E)
6. A one is added to this: 1000 1111 (8F). This is two's complement.
7. 8F is the checksum, as shown above.

Note that for longer data records, the calculation of step 3) may produce a result of greater than one byte. In this circumstance, only the least significant byte is actually selected to undergo the subsequent steps of the checksum calculation.

A.2 Exorcisor Format (download only)

The Exorcisor Format is sometimes referred to as "Motorola S-Records". This format, when displayed, consists of:

1. The start code “S”
2. The record type. This is one of:
    a. 1 - Data Record (line has data)
    b. 9 - End Record (last line, usually no data)
3. The number of bytes in an individual record. Note that this sum includes the two address bytes, all data bytes, and the checksum byte. E.g. 1DH = 29
4. The two-byte address of the first data byte in this record, if this is a Data Record. If this is the End Record, then the address is typically that of the first Data Record transmitted. E.g. 0000H
5. The data in bytes, e.g. 12 34 56 78
6. The checksum of an individual record, e.g. A4H

Consider the following set of Exorcisor records, which have a start address of 0000H and an end address of 008FH. See how the description above matches the columns below:

```
1 2 3   4             5                                                  6
- - -- ---- ---------------------------------------------------- --
S 1 1D 0000 12345678123&5678123456781234567812345678123h5678123456781234 24
S 1 1D 001A 5678123456781234&5678123456781234567812345678123456781234 82
S 1 1D 003A 123h5678123&5678123456781234567812345678123456781234 F0
S 1 1D 004E 5678123h567812345678123h567812345678123&567812345678 4E
S 1 1D 0068 12345678123456781234567812345678123456781234567812345678 BC
S 1 11 0082 567812345678123h567812345678                          62
S 9 03 0000                                                       FC
```

Calculation of the Exorcisor Checksum

Consider the following data record:

```
S104001A568B
```

1. The Start Code, the Record Type and the Checksum are removed: “S”, “1”, “8B”

2. Four Bytes remain: 04 00 1A 56
3. These are added together: 04 + 00 + 1A + 56 = 74H
4. This total is converted into binary: 0111 0100 (74H)
5. The binary representation is reversed. This is the ones complement: 1000 1011 (8B)
6. 8B is the checksum, as shown above.

Note that for longer data records, the calculation of step 3) may produce a result of greater than one byte. In this circumstance, only the least significant byte is actually selected to undergo the subsequent steps of the checksum calculation.

A.3 Binary Format (download only)

The Binary format is the most fundamental format. It consists of the data alone, and as a result, is not normally displayable. This data is in the form of 8-bit bytes. The Binary format lacks any start code, stop code, address field, checksum, or record type.

Binary formats are typically used for speed of transmission, and for testing of the serial communications channel. To download binary data, the Serial Eprom Programmer uses a timeout scheme. The Serial Eprom programmer will wait for an indefinite period of time for the first binary byte to be received, but after that, a 3-second timeout is employed. That is, if the first data byte is received, then a 3-second timeout is started. If a byte is received before that timeout expires, then the timeout is started over. Eventually, the 3-second timeout will expire (this should happen at the end of the transmission). The binary download is then complete.

A.4 Dec Binary Format (download only)

The Dec Binary format is a slight improvement over Binary. It consists of a start code (01), a null (00) prior to transmission, a byte count, an address, and a single checksum of all data transmitted. The Dec Binary format is arranged as follows:

1. Start code (01H)
2. NULL code (00H) prior to transmission
3. Byte count (max FFFFH)
4. Address of first byte
5. Data bytes, total is BYTE COUNT (from 4)
6. Checksum

Files encoded in the Dec Binary format are not directly displayable, because they may contain non-printable characters. Dec Binary is used primarily for speed of transmission of data. Note that unlike the Binary format, no timeout is used for Dec Binary, as a Byte Count is provided, which indicates to the Serial Eprom Programmer when the transmission will end.

A.5 Hex ASCII Format (download only)

The Hex ASCII format, when displayed, consists of data alone. This data, unlike Binary, however, is in an ASCII form, and is thus displayable. Several invisible characters are required for the proper operation of this download format. These characters are:

Start code: ASCII character 02H (STX). This character precedes all data being transmitted.

Stop code: ASCII character 03H (ETX). This character follows the last byte of data being transmitted, and thus is the last character transmitted.

Space character: ASCII character 20H (SP). Space characters may be used to separate the bytes being transmitted. Note that space characters are optional, and any number of space characters may be present between pairs of hex digits (data bytes). Space characters are ignored by the Serial Eprom Programmer during Hex ASCII downloading.

Carriage return: ASCII character 0DH (CR). Carriage returns may be used to split the data into "lines", for ease of editing and readability on the host computer. A carriage return simply causes a period character to be printed to the screen during Hex ASCII downloading.

Hex ASCII lacks any facility for a checksum, address, or byte count. Each data byte consists of two hexadecimal digits, with no spaces between the digits of a single data byte. The following is a sample Hex ASCII file:

```
02 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
   12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
   12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
   12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
   12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
   12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
   12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
   12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78 03
```

B. Serial Eprom Programmer Specifications

This appendix gives the electrical and physical specifications of the Serial Eprom Programmer:

Processor:	**8031 or 80C31**
Clock Frequency:	**11.059200 MHz**
Baud Rates:	**600, 1200, 2400, 4800, 9600, 19200**
Printed Circuit Board Size:	**6 x 6 inches (15 cm x 15 cm)**
Firmware Eprom Type:	**27C256, 250 ns access time, 32 Kb**
RAM Buffer:	**32 Kb or 64 Kb Static RAM**
On-Board Voltage Converter:	**12 VDC to 28 VDC, 85% efficiency**
Eproms Supported (Firmware Version 2.00):	**2716, 27C16,** **2732, 2732A, 27C32,** **2764, 2764A, 27C64,** **27128, 27128A, 27C128,** **27256, 27C256,** **27512, 25C512**

Programming Performance:

Eprom Type	Time to Program
27C64	2.5 seconds
27C128	5 seconds
27C256	10 seconds
27C512	20 seconds
2764A	35 seconds
2764	45 seconds
27128A	70 seconds
27128	90 seconds
2716	103 seconds
27256	140 seconds
2732, 2732A	205 seconds
27512	280 seconds

C. Serial Interface Cable Specifications

The serial eprom programmer uses a DB25 female serial connector and is connected to a PC with a straight-through serial cable.

The following cable pinout will allow direct connection to a 25-pin serial port on a PC.

DB25 (Female)		DB25 (Male)
2 (Tx)	->	2 (Tx)
3 (Rx)	->	3 (Rx)
4 (RTS)	->	4 (RTS)
5 (CTS)	->	5 (CTS)
6 (DSR)	->	6 (DSR)
7 (GND)	->	7 (GND)
8 (CD)	->	8 (CD)
20(DTR)	->	20 (DTR)
CHASSIS	->	CHASSIS

The following cable pinout will allow direct connection to a 9-pin serial port on a PC.

DB9 (Female)		DB25 (Male)
1 (CD)	->	8 (CD)
2 (Rx)	->	3 (Rx)
3 (Tx)	->	2 (Tx)
4 (DTR)	->	20 (DTR)
5 (GND)	->	7 (GND)
6 (DSR)	->	6 (DSR)
7 (RTS)	->	4 (RTS)
8 (CTS)	->	5 (CTS)
CHASSIS	->	CHASSIS

D. ASCII Character Chart

This appendix provides a chart of the ASCII character set, which will aid the user in understanding download formats. All values given for the characters are in hexadecimal.

Char	Value	Char	Value	Char	Value	Char	Value
NUL	00	SP	20	@	40	`	60
SOH	01	!	21	A	41	a	61
STX	02	"	22	B	42	b	62
ETX	03	#	23	C	43	c	63
EOT	04	$	24	D	44	d	64
ENQ	05	%	25	E	45	e	65
ACK	06	&	26	F	46	f	66
BEL	07	`	27	G	47	g	67
BS	08	(	28	H	48	h	68
HT	09	)	29	I	49	i	69
NL	0A	*	2A	J	4A	j	6A
VT	0B	+	2B	K	4B	k	6B
NP	0C	'	2C	L	4C	l	6C
CR	0D	-	2D	M	4D	m	6D
SO	0E	.	2E	N	4E	n	6E
SI	0F	/	2F	O	4F	o	6F
DLE	10	0	30	P	50	p	70
DC1	11	1	31	Q	51	q	71
DC2	12	2	32	R	52	r	72
DC3	13	3	33	S	53	s	73
DC4	14	4	34	T	54	t	74
NAK	15	5	35	U	55	u	75
SYN	16	6	36	V	56	v	76
ETB	17	7	37	W	57	w	77
CAN	18	8	38	X	58	x	78
EM	19	9	39	Y	59	y	79
SUB	1A	:	3A	Z	5A	z	7A
ESC	1B	;	3B	[	5B	{	7B
FS	1C	<	3C	\	5C	|	7C
GS	1D	=	3D	]	5D	}	7D
RS	1E	>	3E	^	5E	~	7E
US	1F	?	3F	_	5F	DEL	7F

Questions:

1. Should I include simple step-by-step guides

    a. Basic downloading and programming
    b. Basic copying with uploading

2. Add real pictures and screen shots