ECROS Technology

www.ecrostech.com

Flash Loader for ZiLOG Z8 Encore!™

User Manual

Manual Revision 1.1

for Flash Loader Version 1.20

November 25, 2004

Table of Contents

Introduction

The ECROS Technology Flash Loader for the ZiLOG Z8 Encore! allows the firmware of a Z8 Encore!-based product to be changed in the field by end-users and / or field service personnel. Existing firmware can be erased and new firmware loaded through one of the serial ports (UARTs) of the microcontroller. The Flash Loader is suitable for use with Terminal Emulator programs widely available on popular computing platforms, such as PCs and PDAs, that are equipped with serial ports. Alternatively, a custom application can be created to automate the process. Careful design allows the Flash Loader to offer the important functions for firmware update using only a small amount of ROM itself.

Features

- Works with both original (Z8F640x series) and enhanced (Z8F642x series) MCUs
- Takes up only 1½ Kbytes of Flash ROM (addresses 0x0200 to 0x7FF)
- Intel Hex16 input format for firmware files; ZDS II compatible
- Tests files for errors, programs Flash and verifies / compares files to Flash
- Insensitive to Terminal Emulator settings regarding line ending characters
- Configurable to skip Flash pages during erasure so as to preserve user settings / data
- Includes a product serial number, which is displayed on Flash Loader activation
- Configurable UART, bit rate, Z8 Encore! clock frequency, etc.

Product Hardware Requirements

A product in which the ECROS Technology Flash Loader is to be used must meet the following requirements:

- It must be possible to force the Z8 Encore! microcontroller to perform a reset, either directly with a reset switch or indirectly as part of the power on sequence.
- One of the Z8 Encore! microcontroller's UARTs must be externally accessible. Only transmit and receive data and signal ground connections are required. Most standard computing equipment, such as a personal computer, requires the data signals to be provided at V.28/RS232 levels. To use such equipment to perform firmware changes an external level converter will be needed if the product does not convert internally.
- The clock frequency of the Z8 Encore! must be such that the UART can be set to operate at a bit rate compatible with the equipment used to perform firmware changes.

The Evaluation Board included in the Z8 Encore!™ Flash Microcontroller Development Kit (ZiLOG part Z8ENCORE000ZCO) meets all these requirements.

Quick Start

This section presents a procedure for quickly putting the Flash Loader through its paces. It assumes you have a Flash Loader distribution unzipped on your computer, the ZiLOG Z8 Encore! Evaluation Board and a computer with a serial port and terminal emulator. ECROS Technology recommends the TeraTerm Pro freeware terminal emulator program. Hyperterminal, which comes with Microsoft Windows, can be used in a pinch. It is also

assumed here that you are familiar with these tools and with developing firmware for the Z8 Encore! in general. If you are not, look for additional details on each step elsewhere in this manual.

1. Plug in the power supply to the evaluation board and connect it to the computer with the serial cable and Target Interface Module (TIM) or the Smart Cable (see the ZiLOG documentation for more information on this if necessary).
2. In the ZDS II IDE, open the project `MakeLod_64n.zdsproj` that comes with the distribution, where *n* is 0 or 2 according to your target MCU. Make sure the release configuration is active and click the Build button. Click the Download Code button to load the Flash Loader into the evaluation board. Click the Stop Debugging button to stop the debugger.
3. Connect the computer's serial (COM) port to P1, the "console" port (UART 0), of the evaluation board. (You can disconnect the TIM or Smart Cable as this will no longer be needed.) If you are using a distribution built for UART1 use P2, the "modem" port instead. Run the terminal emulator program and configure it for the COM port you are using on the computer. Set the bit rate to 9600 bit/s (or the bit rate configured). Select eight data bits, one stop bit and no parity. Turn off all flow control.
4. Hold down the escape key on the computer. After a second or two, press and release the reset button (SW4) on the evaluation board. Continue to hold the escape key down until you see the Flash Loader banner appear in the terminal emulator window.
5. As the Flash Loader has been programmed into Flash via the debug port, the rest of the Flash ROM is blank and there is no need to erase it. Press '`p`' to go to program mode. Activate the terminal emulator file send feature. For example, in TeraTerm Pro, pick File → Send File... on the menu. Browse to the `test` directory of the Flash Loader distribution and select the file `SerializeUart0.hex` or `SerializeUart1.hex`, according to which UART on the evaluation board you are using. Dots will appear as this file is uploaded. When this is finished, you will be in verify mode.
6. Skip verification and just press '`q`' to quit the Flash Loader. The "Serialize" program will run and will ask you to enter a serial number. Enter some string, using no more than 16 characters, which is the default length of the serial number. Press Enter. The program will echo the string back to you for confirmation. Press '`y`'. Your string will be programmed and read back to you again, this time from Flash ROM.
7. Reset the evaluation board by pressing SW4 once more. The Serialize program runs again. A row of dots (Z8F640x MCU) or plus signs (Z8F642x MCU) will appear in the terminal emulator window first. The Flash Loader sends these characters while it is waiting to see if an escape character arrives.
8. One last time, reset the evaluation board, this time holding down the escape key to activate the Flash Loader again. On the line after the banner you will see the serial number that you just programmed. Erase the application program (Serialize) by pressing '`z`'. If you now quit the Flash Loader by pressing '`q`', you will get the error message `no user program!`. The application program area of Flash is empty.

You have now programmed the ECROS Technology Flash Loader into a system, used it to upload an application program, run the application program and then erased it. This particular program enabled you to add a serial number in an area reserved by the Flash Loader. (Normally, this would be part of a larger system test program intended to make

sure that all the product hardware is working correctly.) Having erased this program, you are ready to upload the product application firmware. The end user of your product can use the Flash Loader to update this firmware to new versions that you supply.

The remainder of this User Manual presents more detail on each of the steps above and adds information about configuring source distributions, testing, diagnosing problems, error messages, etc.

Installing the Flash Loader

The ECROS Technology Flash Loader is a program that runs from the Z8 Encore! Flash ROM when the system is reset. Therefore, it must be installed into the microcontroller during the product manufacturing process. This section tells you what you need to know to prepare the Flash Loader for incorporation in your product.

Configuring and Building the Source Distribution

A source distribution includes C language source files and a ZDS II version 4.9.0 project file to build a Flash Loader object library. Using this library, you complete installation as if you had an object distribution. If you do not have a source distribution, please skip this section.

All configurable parameters and strings, except the serial number, are defined in the C language header file `Configure.h`. You can open the project file `MakeLib.zdsproj` and edit this file in ZDS II or use any text editor.

Banner - Define `BANNER` to be the text that you want sent to the UART when the Flash Loader is activated (by holding down the escape key during reset). The default value is "`ECROS Technology Z8 Encore! Flash Loader, vX.XX` ". The length of this string contributes to the code space used by the Flash Loader and if you make it too long the build may fail. At least 48 characters are available. More may be available in some versions, but only 48 are guaranteed and relying on more may prevent you from moving to newer versions of the Flash Loader.

Clock Frequency - Define `Z8_CLOCK_FREQ` to be the Z8 Encore! clock frequency of your system, in Hz. The default value is 18432000, corresponding to 18.432 MHz. The Flash Loader uses this value to configure the Flash Controller and the UART. It must be set correctly for the Flash Loader to work.

Choice of UART - Define `SELECT_UART` according to which Z8 Encore! UART you want the Flash Loader to use. The default value is 0 for UART0. You can change this to 1 if you want to use UART1. Values other than 0 or 1 will cause a compilation error.

UART Bit Rate - Define `UART_BIT_RATE` to be the serial data rate, in bits per second, at which you want the Flash Loader to operate. The default value is 9600, i.e. 9.6 kbit/s.

Reset Pause Time - Define `RESET_PAUSE_TIME` to be the time, in milliseconds, for which the Flash Loader pauses and looks for escape characters during system reset. The default value is 100 for a pause of 100 ms. This is a reasonable setting when the Flash Loader is used with a PC and Terminal Emulator program and the keyboard repeat rate is set above 10 characters per second. If you select a high UART bit rate, you may find that you are restricted as to the maximum pause time. The flash loader measures this time by sending a counted number of period characters to the UART. The counter is 8 bits. If the

computed count will not fit in 8 bits, you will receive a compilation error.

Flash Pages to Erase - Define `FIRST_ERASE_PAGE` and `LAST_ERASE_PAGE` to be the first and last Flash ROM pages that are to be erased by the Flash Loader when the user selects the erase command. The default values are 8 and 127, so that Flash pages in the range 8 to 127, inclusive, are erased (addresses 0x1000 to 0xFFFF). Specifying a first page below 4, which would erase part of the Flash Loader itself, causes a compilation error. Pages from 4 to the page before the first erased page can be used by the application for storage of settings and data as they will not be erased on a firmware update. For example, the default settings provide four such pages from page 4 to page 7 (addresses 0x0800 to 0x0FFF). Do not specify a last page above the highest page implemented by the microcontroller you are using. Pages above your last page can also be used for the storage of settings and data. The default settings assume a 64Kbyte Flash device.

Serial Number - For information on modifying the serial number string, refer to the next section. You do not need to do this at this time.

To build a Flash Loader object library after completing all configuration, open the project file `MakeLib.zdsproj` with ZDS II version 4.9.0. Make sure the release configuration is active and click the Build button. You now effectively have an object distribution of the Flash Loader configured to your own needs. You can build the final load file as described in Building the Object Distribution, below.

Editing the Product Serial Number

The Flash Loader incorporates a serial number string to allow products that use it to be uniquely identified in the firmware. This string is output to the UART when the Flash Loader is activated. Products that do not have a display device suitable for showing the serial number can thus be queried for this information simply by activating the Flash Loader.

The serial number string is in the assembly file `Startup.asm` and follows the label `_serial_number`. You can open the project file `MakeLod_64n.zdsproj` and edit this file in ZDS II or use any text editor. You will see something like this:

```
 XDEF _serial_number
   DB %55
_serial_number:
   BLKB 16, %FF              ; blank, to be programmed later
;   DB "FL01010310030001"      ; example
   DB 0
```

The statement `DB %55` puts a byte in Flash memory with the value 0x55. This allows the application program to locate the serial number by searching forward from 0x0200 for this "magic" byte. If this capability is not required, this statement can be safely removed without affecting the Flash Loader. The `DB 0` after the string is the string terminator and should not be removed.

The statement `DB "FL01010310030001"` defines the serial number string. If this is commented out, as in the example above, and you want to specify a string, remove the '`;`' comment character in front of this statement and instead put it in front of the statement `BLKB 16, %FF`.

The only use made of the serial number string by the Flash Loader is to display it on activation (that is, when an escape character is detected at the serial port and the Flash

Loader retains control rather than running the application program). You can therefore use any format you wish for the serial number. Typically, serial numbers identify the product, revision level, place and date of manufacture and sequential unit number.

It is safe to change the length of the string to meet your needs. However, if you make the string too long, the Flash Loader will no longer fit in its budgeted space and building it will fail. At least 16 characters are available. More may be available, but only 16 are guaranteed and relying on more may prevent newer versions of the Flash Loader from being used. You can trade off the length of the banner and serial number strings.

Changing the serial number in `Startup.asm` for each unit you build may not fit into your production methodology. An alternative is to replace the string with an equivalent number of unprogrammed bytes (all ones) and to program serial numbers separately from the Flash Loader. For example, a manufacturing test program might be loaded into the product at some point. The serial number can be written to Flash ROM by this program in the reserved space, perhaps with the number itself being sent by some test fixture over the serial port. Note that the Flash Loader itself cannot accept and program a serial number in an Intel Hex file. This string lies within the area of Flash ROM occupied by the Flash Loader and it will therefore refuse to write any data there. In the example above, the statement `BLKB 16, %FF` places 16 all-ones bytes in Flash memory where the serial number should be. In Quick Start, above, you used a simple application program to replace these unprogrammed bytes with an actual serial number.

Building the Object Distribution

An object distribution consists of a library file containing the object code of most of the Flash Loader plus the single assembly source file `Startup.asm`. The Flash Loader has been configured by ECROS Technology according to your specifications at the time of purchase. The only change that you can make is to the serial number string, see Editing the Product Serial Number, above.

If you have a source distribution, you will have built your own library by following the steps in Configuring and Building the Source Distribution, above. You should then continue with the installation at this point.

To build the Flash Loader executable file, open the project file `MakeLod_642.zdsproj` with ZDS II version 4.9.0. If the target MCU is an original Z8 Encore! from the Z8F640x series, use the project file `MakeLod_640.zdsproj` instead [1]. Make sure that the release configuration is active and click the Build button.

Building the Evaluation Distribution

The evaluation distribution has no settings or text that can be changed. All configurable parameters have been given their default values as described in Configuring and Building the Source Distribution, above. The serial number string is fixed to "`Evaluation only!`". Build the Flash Loader in the same way as the object distribution, see Building the Object Distribution, above.

1 The only difference between these project files is the selected CPU Family. However, modifying the CPU Family in ZDS II version 4.9.0 wipes out other essential project settings. The two project files have been supplied to make it unnecessary for you to do this.

Loading the Flash Loader into the Z8 Encore!

The ZDS II project file `MakeLod_64n.zdsproj` builds the Flash Loader in IEEE 695 format ready to load via the debug port. Connect the serial cable and Target Interface Module (TIM) or the Smart Cable to the host computer and the target system. After building, click the Download Code button to load the Flash Loader into the target. When the load has completed, click the Stop Debugging button to stop the debugger.

To check the installation of the Flash Loader, connect the serial port of the product to a computing device and run a terminal emulator application. Set the terminal emulator's serial port settings to eight data bits, one stop bit, no parity and no flow control. Set the bit rate to match to your selection during configuration (or 9,600 bit/s for evaluation distributions). Hold down the computer's escape key for a second or so and then, still holding down the escape key, reset or turn on your product. In the terminal emulator window you will see a line of dots or plus signs[2] followed by the Flash Loader banner, the serial number of the product and a prompt. The Flash Loader is active and ready to upload a firmware file.

Ideas for a procedure for the full evaluation of the Flash Loader are given in Evaluation Guidelines, below.

If you wish to program the Flash Loader into the Z8 Encore! microcontroller other than through the debug port, refer to the application notes by ZiLOG on this subject. You may need to modify the ZDS II project files to obtain the necessary executable format, but *do not change the CPU Family* or essential project settings will be lost.

Building Application Firmware

In order to be compatible with the ECROS Technology Flash Loader, the product's application firmware must be built so as to conform to the rules described in this section.

It is not necessary to use any specific version of ZDS II to build application firmware.

Program Start Address

The application program must start at (execute from) address 0x0038. In some versions of the ZiLOG ZDS II tools for the Z8 Encore!, for example version 4.2.1, if you use the standard startup module you will have no other choice of start address. More recent versions, for example version 4.5.1 and later, no longer hard-code this and you must enter a linker directive to specify the program start address. In the ZiLOG ZDS II IDE, select Project → Settings... on the menu to bring up the **Project Settings** dialog. On the Linker tab, select Input in the Category: pick list. Then, click the Add Directives button in the Link Control File area to bring up the **Additional Linker Directives** dialog. In the edit pane, type locate startup at $38. You must repeat this for each configuration of your project. Use the unnamed pick list at the top left of the **Project Settings** dialog to change the configuration to which your settings are being applied. Refer to ZiLOG documentation for more information on establishing the program start address, particularly if you are not using the standard startup module.

2 The Flash Loader automatically detects which CPU family it is running on and adjusts its operation accordingly. If it determines that it is running on an original (Z8F640x series) MCU, dots are printed after reset. If it is running on an enhanced (Z8F642x series) MCU, plus signs are printed.

Code Memory

Because the Flash Loader resides in ROM at the same time as the application program, the application must be restricted to use only other Flash ROM areas. The simplest way to avoid using the locations taken up by the Flash Loader is to specify two ranges of addresses, 0x0000 to 0x01F7 and 0x0800 to the top of ROM. In the ZiLOG ZDS II IDE, select Project → Settings... on the menu to bring up the **Project Settings** dialog. For ZDS II versions prior to 4.9.0, click the Target tab. In the Code edit box of the Memory area, enter 0-1F7,0800-FFFF (for a 64 Kbyte Flash device). For ZDS II version 4.9.0, click the Linker tab and select the Address Spaces category. Fill in the Code edit box as above. You must repeat this for each configuration of your project with which you want to use the Flash Loader.

In order to completely erase prior versions when updating firmware, the values you chose for `FIRST_ERASE_PAGE` and `LAST_ERASE_PAGE` during configuration (refer to Flash Pages to Erase , above) must match the second range of code memory. The first range is on page 0, which is automatically erased. With the code memory setting described above, the corresponding first and last erase pages would be 4 and 127. For Z8 Encore! devices with less than 64 Kbytes of Flash, both the upper limit of code memory and the last erase page must be reduced. For example, with a 48 Kbyte device, they should be BFFF and 95 respectively.

You can exclude any part of Flash from the code memory and erase pages for any reason with the exception of page 0. You must have code on page 0 because this is where the application program must start (the Flash Loader runs the application by branching to address 0x0038). The erase command always erases page 0 and restores the Flash Loader's start address of 0x0200 to the reset vector. Locations 0x01F8 to 0x01FF on this page are also reserved for Flash Loader status information. You could, however, exclude pages 4 to 7 by having the second code memory range begin at 0x1000 and configuring the first erase page to be 8 (which is in fact the default). Flash memory in the address range 0x0800 to 0x0FFF is now neither used for application program code and data nor is it erased by the Flash Loader. This space might, for example, be used by the application program for the storage of user options and data which persist over firmware updates.

Executable Format

The linker must be instructed to output an Intel Hex16 format file. In the ZiLOG ZDS II IDE, select Project → Settings... on the menu to bring up the **Project Settings** dialog. On the Linker tab, select Output in the Category: pick list. Then, select Intel Hex16 - Records in the Executable Format: pick list or, in ZDS II 4.9.0, select the Intel Hex16 - Records check box. You must repeat this for each configuration for which you want this output format. The Intel Hex16 files produced will end in `.hex`.

To use the ZiLOG debugger, you will want to have a project configuration set to output the IEEE 695 format that can be uploaded with the Target Interface Module (TIM) or the Smart Cable. You could, for example, set the “Debug” configuration this way and have the “Release” configuration set to produce Intel Hex16. It is also possible to create new configurations, for example duplicating “Release” for each output format. Refer to the ZiLOG documentation and on-line help for information on managing configurations. In ZDS II 4.9.0 it is possible to generate multiple output files from the same configuration.

Other Miscellaneous Rules

Fairly obviously, an application program should not contain code that erases the Flash ROM pages containing parts of the Flash Loader. These are pages 0 to 3, that is the address range 0x0000 to 0x07FF. This rule might be violated in some circumstances. The purpose of the program might be to replace the Flash Loader with a newer version, in which case it must first erase the old one. Flash pages can be erased if their contents are saved elsewhere and restored. If an application wishes to modify Flash page 0, most of which is available, it must save and restore the contents of addresses 0x01F8 to 0x1FF and restore the reset vector to 0x0200, *not its own start address*.

Using the Flash Loader

This section describes the process of using the ECROS Technology Flash Loader to update the firmware in a product.

Configuring a Terminal Emulator Program

One method of sending commands and the firmware file to the Flash Loader is using a computing device, such as a personal computer or PDA, that is equipped with a serial port and is running a terminal emulator program. The terminal emulator must be set up to use the designated serial port and to configure the communication parameters correctly. The serial port must be set for the correct data rate (e.g. 9,600 bit/s), eight data bits, no parity and one stop bit. Flow control should be disabled.

Most terminal emulators allow the user to send a file to the serial report either directly or using some sort of file transfer protocol. The Intel Hex16 file must be sent to the Flash Loader byte-for-byte as it is stored on disk. A file transfer protocol must *not* be used. It is sometimes difficult to tell which command of the terminal emulator achieves this goal. Probably the best way to tell is that if you are asked to make a choice between "kermit", "xmodem", etc., then you are using the wrong command.

The Flash Loader is not sensitive to how the terminal emulator transmits line endings. It will accept either carriage return or line feed or both in either order.

Step-by-Step Firmware Update

The first step in performing a firmware update is to connect the product and computing device with a suitable serial data cable, power on the computer and run and (if necessary) configure the terminal emulator. The details of this step will depend on the nature of the product, how the serial port is accessed, etc.

The second step is to power on the product and activate the Flash Loader. The escape key of the computer must be pressed and held down while the product is powered on and goes through a reset. You can also power on the product and then cause it to reset if there is a means to do this. Hold down the escape key for a second or two so that the auto-repeat function of the computer keyboard begins sending escape characters rapidly. Do not release the escape key until the reset is complete and you see the Flash Loader banner message. The banner message will be as you configured it, see Banner, above. Below this, you will see the serial number, again as configured, see Editing the Product Serial Number, above. If the serial number string is configured to be unprogrammed locations so that the serial number can be defined in a separate step, but this has not been done, it will appear as garbage characters. Finally, you will see the Flash Loader prompt, which is

the mode indicator, “test”, followed on the next line by a right angle bracket, “>”.

An optional step at this point is to test the firmware file to be uploaded. This allows you to make sure that the file contains no errors that would prevent it from being programmed into Flash. If the file is corrupt, the firmware improperly built or you've just got hold of the wrong file, you can find out now before you erase the existing firmware. Since the Flash Loader is already in test mode, just send the firmware file using the terminal emulator. You should see a row of dots (the period character), one for each record in the Intel Hex16 file. If instead of a dot you see a number, this indicates an error. Refer to Firmware Update Errors, below, to find out what the error numbers mean. At the end of the upload, the Flash Loader will display a count of the errors. You should not proceed with the firmware update using this file if you encounter errors. End users who find themselves in this situation should attempt to obtain a replacement firmware file. If this also does not work, they will have to refer the problem to the product manufacturer or other supplier of the firmware.

The third step in updating firmware is to erase the existing contents of Flash ROM. To execute the erase command, press the 'z' key. You will see the message “erasing” and after a few seconds a new prompt will appear. The mode indicator is now “prgm”. The Flash Loader has automatically changed from test mode to program mode. If no firmware is currently programmed into the product, the erase operation is not necessary. To switch directly to the program mode without erasing, press the 'p' key.

Now that the Flash is empty, you can proceed to the fourth step. Send the new firmware file to the product using the method appropriate to the terminal emulator application. Be sure not to use a file transfer protocol, but to just send the file as it is stored on disk. As each record in the file is sent, you will see a dot appear. It may take a minute or two to complete the upload of a large file. Do not interrupt the update by resetting or powering off the product, unplugging the cable or quitting the terminal emulator application. When the upload is finished, a new prompt will appear. Now the mode indicator is “vrfy”. The Flash Loader has automatically changed from the program mode to the compare / verify mode. If a number appears in place of any of the dots, an error has occurred. You will see a report of the total number of errors at the end of the transfer. Refer to Firmware Update Errors, below, to find out what the error numbers mean. If the firmware file was tested or comes from a trusted source, the only errors that should occur at this stage are hardware related. You will also get “Flash Location Not Empty” errors if you did not erase the old Flash contents.

Optionally at this point, you can verify that the firmware was successfully programmed. Since the Flash Loader is already in compare / verify mode, simply send the firmware file again. For each record, a dot means no mismatches and a digit indicates an error. Since the Flash Loader verifies each byte as it is written in program mode, a “Compare / Verify Failure” will only occur if the Flash memory retained the data for a short time and then lost it. This would indicate a hardware problem.

The final step is to terminate the Flash Loader and run the new firmware. You can reset or power cycle the product to achieve this. Alternatively, just press 'q' to quit the Flash Loader. It will announce that it is running the user program. Finish up by disconnect the serial cable (if desired).

Other Commands

If you get partway through a file upload and want to terminate it prematurely, first stop the upload using the terminal emulator. The Flash Loader is now hung waiting for the rest of the data and an end-of-file record that will never come. To abort the file read, press the escape key. This is counted as a error to make sure you know the firmware upload did not complete. You will get the error number in place of a dot and an error count before the prompt returns.

The escape key can also be used to get back to the Flash Loader prompt if the firmware file has been truncated and the end-of-file record is missing.

You can manually change between test, verify and program modes by pressing the '`t`', '`v`' and '`p`' keys, respectively. The mode indicator in the prompt will change to "`test`", "`vrfy`" or "`prgm`" and lets you know what mode you are currently in at all times.

If you want to find out whether the firmware in the product matches a particular firmware file, activate the Flash Loader, switch to compare / verify mode by pressing the '`v`' key and upload the file. Any occurrences of the "Compare / Verify Failure" error indicate that the file is not the same as the current firmware in Flash.

It is possible to upload firmware in multiple Intel Hex files. During programming, the Flash Loader does not check that the whole of the application program space is empty, just that the locations being programmed are empty (or already contain the desired data). Therefore, you can "overlay" several files as long as the locations they program do not overlap. Strictly speaking, they can overlap but then the data in the overlapping areas must be identical in all files. In fact, you can program the exact same file twice. The second time around no actual Flash programming will be done as the Flash Loader does not program locations that already contain the desired data.

Command Summary

escape – The escape command activates the Flash Loader on system reset. It also aborts a file upload that ended before the end-of-file record was received.

`z` – Erase the configured application program area of Flash memory. After the erasure, the Flash Loader automatically goes to program mode.

`t` – Go to test mode. In this mode, records from an uploaded Intel Hex16 file are tested for proper construction and compatibility with the Flash Loader.

`v` – Go to compare / verify mode. In this mode, records from an uploaded file are tested and also compared to the contents of Flash ROM. Any record containing data that is not the same as what is currently stored is flagged with a "Compare / Verify Failure" error.

`p` – Go to program mode. In this mode, records from an uploaded file are tested and if no error is found the data in the record is programmed into Flash ROM. Note that each memory location is processed in sequence so that errors late in the file do not prevent earlier data from being programmed.

`:` – Read an Intel Hex16 input record. A user will not normally enter this command. Its purpose is to trigger the reading of a record from an Intel Hex file when one is uploaded.

`*` – Ignore input until after the next line end. There is no need for a user to enter this command. Its purpose is to skip over comments in Intel Hex files.

q – Quit the Flash Loader and run the application program. If no application program has been programmed into Flash, the message "`no user program!`" will appear.

newline – Repeat the prompt (can be used to make sure that the Flash Loader is alive).

A Flash Loader Driver

Although the ECROS Technology Flash Loader has been designed so that an end user of moderate technical sophistication can use it with a computer and terminal emulator, you may want to create a custom driver program to run on the computer. This removes the possibility of misconfiguring the terminal emulator and hides the rather terse commands behind a user interface of your design.

Diagnosing Problems

Banner does not Appear, only a Row of Dots or Plus Signs

The Flash Loader pauses only briefly after reset to look for an escape character at the serial port. You can configure the length of the pause, within certain limits, see Reset Pause Time, above. If when you try to activate the Flash Loader you see only dots or plus signs in the terminal emulator, escape characters are not being detected.

End users experiencing this problem should look for the following causes:

The escape key must be held down long enough to start the auto-repeat feature of the keyboard before resetting or powering on the product. Check the repeat delay setting of the computer or hold the escape key longer before resetting the product.

The keyboard repeat rate must be sufficiently high that at least one escape arrives at the UART after the system leaves reset and before the Flash Loader passes control to the application program. The repeat rate may be set too low. Check and increase the repeat rate if necessary.

If sometimes you get into the Flash Loader and sometimes you don't, there is nothing wrong with trying repeatedly until you get in.

As a last resort, look for breaks or miswiring from the computer's transmit data output to the product's receive data input. Perhaps you can "hear" the product but the product can't "hear" you and therefore is missing your escape.

Nothing Happens At All

If nothing at all appears in the terminal emulator, not even dots or plus signs, suspect a bad serial port connection or misconfiguration of the terminal emulator. Also make sure that you are using the UART for which the Flash Loader was configured.

If you are using a terminal emulator in a windowed operating system, make sure that when you hold the escape key the terminal emulator window has the keyboard focus.

Note that if you have replaced the firmware in the product using the Z8 Encore! debug port, you will have erased the Flash Loader and will need to put it back before you can use it.

Firmware Update Errors

The ECROS Technology Flash Loader performs extensive error checking with the goal that the end-user should be clearly informed about whether the firmware update is or will be successful. However, because of the small code budget, the reporting of errors is necessarily terse and diagnosis of the problem must be performed by inspecting the file. The Flash Loader is not intended for locating problems with the format of the input file.

For each record in the input file, a single character is output at the UART. If no errors are encountered, this will be the period character and serves to assure the end user that the upload is making progress. Errors in the record are reported by replacing the period with a number from 1 to 9, according to the error. Only one error can be reported for a record. When the upload is complete, the total number of records containing errors is shown. In end user situations, errors should be rare and indicate gross problems such as corrupt or incorrect firmware files, failure to erase the existing firmware, etc.

The following sections describe each of the detected and reported errors. Where an error is associated with a particular Flash Loader mode, this is indicated in the description.

Error 1 – Compare / Verify Failure

The Flash Loader reports error 1 in compare / verify mode to indicate that data in the file uploaded did not match the contents of Flash ROM at the same address. An end user encountering this error should interpret it according to context as a verification failure or an indication that the currently stored firmware is not the firmware in the file uploaded.

Error 2 – Bad Checksum

The last two characters of an Intel Hex16 record comprise the checksum. The Flash Loader verifies the checksum and if it is not correct reports error 2. Assuming that properly constructed and tested firmware files are supplied, if an end user sees this error the likely causes are corruption of the file or upload of the wrong file. Note that an Intel Hex32 file is likely to cause this error. The ECROS Technology Flash Loader only reads Intel Hex16 files[3].

Error 3 – Bad Record Type

The seventh and eighth characters of an Intel Hex16 record comprise the record type. The only valid types are 00 (normal record) and 01 (end of file). Any other value will cause the Flash Loader to report error 3. Assuming that properly constructed and tested firmware files are supplied, if an end user sees this error the likely causes are corruption of the file or upload of the wrong file. Note that an Intel Hex32 file is likely to cause this error.

Error 4 – Non-Hexadecimal Digit

With the exception of comments and the leading colon character, lines in Intel Hex files should consist entirely of hexadecimal digits (characters 0 to 9 and A to F or a to f). If a character is encountered in a record (after the colon) that is not a hexadecimal digit, the Flash Loader reports error 4. Assuming that properly constructed and tested firmware files are supplied, if an end user sees this error the likely causes are corruption of the file

3 As of version 4.6.1, ZiLOG's ZDS II for the Z8 Encore! produces only Intel Hex16 output even if Intel Hex32 is specified in the project settings.

or upload of the wrong file. A line break added in the middle of a record of an otherwise valid firmware file will cause this error.

Error 5 – Flash Location Not Empty

Before writing data to an address in Flash ROM, the Flash Loader checks to see what is currently stored at that address. If the desired data is already in place, the programming is skipped and normal operation continues. Otherwise, the Flash Loader requires that the location be in the erased state of all ones. If this is not so, error 5 is reported. The policy of only programming locations in the erased state ensures compliance with ZiLOG's requirement that a Flash location be written no more than twice between erasures. Error 5 can only occur in program mode or during the execution of an erase command (when the Flash Loader restores its start address to the reset vector).

If an end user encounters this error, it is likely to be because the Flash ROM already contains application firmware or some other data. An erase operation must be performed before uploading new firmware. This error will also occur if the Z8 Encore! device fails in such a way that an erase operation does not return Flash ROM contents to the erased state. The recurrence of error 5 after an erase operation indicates a hardware problem.

Error 6 – Flash Controller Error

Error 6 indicates a hardware problem. The Flash Loader reports this error if it is unable to unlock the Z8 Encore! Flash Controller. This error can only occur in program mode or during the execution of an erase command.

Error 7 – Programming Failure

After each byte is programmed into the Z8 Encore! Flash ROM, the Flash Loader reads back the ROM contents and compares it to the data written. If the data read does not match what was written, error 7 is reported. This could occur if the microcontroller is faulty or the Flash has reached the end of its erase / write lifetime. This error can only occur in program mode or during the execution of an erase command (when the Flash Loader restores its start address to the reset vector).

Error 8 – Bad Address

To be usable with the ECROS Technology Flash Loader, application firmware must not make use of Flash ROM locations reserved by the Flash Loader for its own use and must execute from a specified address (see Building Application Firmware, above). Error 8 is reported if *either* an address specified in the firmware file falls into the reserved range *or* the address is the reset vector and the data is not the correct application start address. An end user should not see this error if properly constructed firmware files are supplied.

Error 9 – User Abort

If the user presses the escape key to break out of an aborted file upload, the Flash Loader reports error 9. This is done to make it clear that the upload was not successful.

Evaluation Guidelines

Before you purchase the ECROS Technology Z8 Encore! Flash Loader, you will want to evaluate it to make sure that it meets your needs. To do this, you will need the following:

- The evaluation distribution of the Flash Loader. Full object and source distributions can also be put through the evaluation procedure described here.
- A Z8 Encore!-based hardware platform meeting the requirements described in Product Hardware Requirements, above.
- An application program of some sort to upload into the hardware.
- A computer with a serial port and terminal emulator program. Any personal computer with a COM port is suitable.

Basic Testing

Basic testing can be accomplished by following the procedure described in Quick Start, above. If you have a source distribution, you should also try out the configuration options described in Configuring and Building the Source Distribution. If you do this, and also if you have an object distribution configured for you by ECROS Technology, be sure to modify the procedure according to the configuration changes you made or requested. For example, if you requested that your Flash Loader use a bit rate of 19200 bit/s, make sure you set the terminal emulator bit rate to this value during evaluation (rather than to 9600 bit/s).

Additional Testing

You will want to make sure that the Flash Loader accepts valid Intel Hex files as input, detects any errors in these files, correctly indicates mismatches in compare / verify mode, etc. To do this, you should generate your own application firmware file and derive from it a set of Intel Hex files with various changes, some valid and some not. These can be sent to the Flash Loader in test, compare / verify and program modes and its behavior observed.

The set of files that ECROS Technology uses to test the Flash Loader is included in the `test` folder of all paid distributions. The reference file, from which the others are derived, is `SerializeUart1.hex`. This is the Serialize application used in Quick Start, above, with UART1 of the target system used to enter the serial number.

The following files in this folder have been changed in allowable ways and should be treated by the Flash Loader in an identical manner to the original:

- `AllLowerCase.hex` – all characters are lower case (a to f instead of A to F)
- `BlankLines.hex` – blank lines have been added
- `OnlyNewlines.hex` – carriage returns have been removed, leaving only newlines
- `OnlyReturns.hex` – newlines have been removed, leaving only carriage returns
- `WithComments.hex` – comment lines have been added

Other files have had various errors introduced into them to show how the Flash Loader handles these situations.

- `BadChecksumLine6.hex` – the checksum on line six has been made incorrect
- `BadCodeAddress.hex` – the program uses locations in Flash ROM reserved by the Flash Loader
- `BadRectypeLine15.hex` – the record type on line 15 is invalid (02)

- `DeletedCharLine20.hex` – a character has been deleted in line 20
- `ExtraLineAtLine10.hex` – line 10 has been broken up into two lines
- `FileEndsInLine13.hex` – the file has been truncated in the middle of line 13
- `NonHexDigitLine9.hex` – there is a non-hexadecimal digit (I) in line 9
- `TruncatedLine4.hex` – line 4 has been truncated (chopped short)
- `TwoSmallChanges.hex` – two very small changes in the program source code result in isolated compare / verify errors
- `WrongRunAddress.hex` – the program runs from the wrong address (0x0080)

Objective of Evaluation

The objective of an evaluation of the Flash Loader should be to make sure that it will meet your requirements for firmware update in the field. These may be different if you have only your own field service people perform the update as opposed to end users and on the likely technical sophistication of your users.

ECROS Technology has put the Flash Loader into the field among users of varied technical knowledge. The first version fielded was sensitive to line endings in the uploaded file. Some users failed to configure their terminal emulators correctly and this was the only cause of support calls. This shortcoming has been corrected in the current version, which is insensitive to line endings.

By using the ECROS Technology Flash Loader in your product, you are taking advantage of field experience. ECROS Technology has made every effort to ensure that the Flash Loader is reliable and easy to use, consistent with its very low ROM footprint. Just as with your own firmware, thorough testing is necessary to ensure that problems do not arise in the field. Other users of the ECROS Technology Flash Loader contribute to this testing. However, you are ultimately responsible for determining its suitability to your particular purpose. The liability of ECROS Technology is limited to the refund of the purchase price.