Application Note

ZTP-Based Flash Loader Utility for the eZ80F91 MCU

AN022404-0608

Abstract

This application note describes a Flash Loader utility that can be used to upload a Zilog TCP/IP (ZTP) stack based user application (firmware) to Zilog's eZ80F91 MCU's external Flash memory via the Internet. The application note describes the client- and server-side applications and provides instructions to integrate the server-side application (the Flash Loader driver) with ZTP. The ZTP-based user application (firmware) is enhanced so that program control can be transferred readily between the user application and the Flash Loader driver; this application note contains instructions to provide such functionality to the ZTP-based user application.

The Flash Loader uploads the user application to the eZ80F91 target MCU via the internet. The Flash Loader drivers integrated with the ZTP stack resides in the MCU's internal Flash memory. The Flash Loader uploads the user applications only to external Flash memory.

The Flash Loader discussed in this application supports AMD Flash device **AM29LV160BB**. This AMD Flash device is required to be populated on the eZ80® development platform of the Modular Development Kit (**eZ80F910100KITG**). The client GUI application is available for installation and can be downloaded and installed to the PC.

Notes:

1. *The source code file associated with this application note,* `AN0224-SC01.zip` *is available for download at* www.zilog.com.
2. *The source code files in this document are intended for use with the ZTP Software Suite v2.1.0 and the Zilog Developer Studio II—IDE for eZ80Acclaim!® v4.11.0 (ZDS II v 4.11.0).*

Zilog® Product Overview

This section provides brief overview of the Zilog products used in this application note, which includes the award-winning eZ80Acclaim*Plus!*™ microcontrollers and the full-feature ZTP software suite.

eZ80Acclaim*Plus!* MCU Family Overview

The Flash-based eZ80Acclaim*Plus!* MCUs, device number eZ80F91, are an exceptional value for customers designing high performance embedded applications. With speeds up to 50 MHz and an on-chip Ethernet MAC, designers have the performance necessary to execute complex applications supporting networking functions quickly and efficiently. Combining on-chip Flash and SRAM, eZ80Acclaim*Plus!* devices provide the memory required to implement communication protocol stacks and achieve flexibility when performing in-system updates of application firmware.

Zilog also offers two eZ80Acclaim!® devices with Flash memory (eZ80F92 and eZ90F93) and two eZ80® devices without Flash memory (the eZ80L92 and eZ80190 microprocessors).

Zilog TCP/IP Software Suite Overview

The Zilog TCP/IP (ZTP) Software Suite integrates a rich-set of networking services with an efficient real-time operating system (RTOS). The operating system is a compact preemptive multitasking, multi-threaded kernel with inter-process communications (IPC) support and soft real-time attributes. Table 1 lists the standard network protocols implemented as part of the embedded TCP/IP protocol stack in ZTP.

Table 1. Standard Network Protocols in ZTP

HTTP	TFTP	SMTP	Telnet
DHCP	DNS	TIMEP	SNMP
ICMP	IGMP	ARP	RARP
IP	PPP	TCP	UDP
SNTP	FTP	SSL	

Many TCP/IP application protocols are designed using the client-server model. The final stack size is link-time configurable and determined by the protocols included in the build.

TCP/IP Overview

For two computers to exchange information over a network, several components must be in place before data can actually be sent and received. A crucial component is the protocol that defines the parameters of the communication between them. The most popular protocol used is the Transmission Control Protocol/Internet Protocol (TCP/IP). This section discusses in brief about the TCP/IP protocols and sockets and socket connections.

By convention, TCP/IP refers to a suite of protocols, all based on the Internet Protocol (IP). Using the Internet Protocol any system on any network can communicate with any other system as easily as if both were on the same physical network.

The Transmission Control protocol was developed as a connection-oriented protocol that exists in the Transport layer of the OSI model, which defines the standard for networking protocols. The TCP is built over the IP and it offers a reliable, full-duplex byte stream such that you can focus on writing the application rather than handling the dropped data packets.

TCP is known as a connection-oriented protocol because the sender of a message must contact and maintain a dialog with the recipient. That is, before two programs can begin to exchange data they must establish a connection with each other. This connection is established with a three-message handshake in which both the programs (client and server) exchange packets and establish the initial packet sequence numbers.

Sockets

ZTP suite provides API functions that allow an application program to send and receive individual TCP data packets in a networked communication setup. An important step in the process of sending or receiving data packet via TCP is to create sockets, which are communication end-points. However, creating a socket in itself does not assure the exchange of information; a connection must be established with a socket on the corresponding program at the other end of the communication setup. To establish this connection, the socket address of the corresponding application (to be connected) must be known. After establishing a connection to the socket in the corresponding application the application program can consider/process the data to be read or written.

When a program attempts a Read or Write operation via a socket, it may not be able to complete the operation immediately for a variety of reasons. For example, a Read operation on a socket cannot be completed because the remote host is yet to send some data. In such a situation, when there is no data waiting to be read, one of two things occur: the Read function waits until some data is written to the socket (blocking socket), or the Read func-

tion returns immediately (non-blocking socket) with an error that indicates that there is no data to be read.

For a blocking socket, the Read operation completes and execution of the program resumes only after the remote system writes some data on the socket. For a non-blocking socket, the application must recognize the error condition and handle the situation appropriately.

In this application note, both the blocking (for the Read operation) and nonblocking (for the Write operation) sockets are implemented.

Developing a ZTP-Based Flash Loader for the eZ80F91 MCU

The eZ80® CPU-based eZ80F91 MCU can write to its own program memory space. The eZ80F91 MCU features an on-chip Flash Controller that erases and programs the on-chip Flash memory.

The ZTP-based Flash Loading functionality is implemented in three parts:

- Server-side TCP socket programming and integrating it with the ZTP.
- Client-side TCP socket programming (providing a GUI).
- Programming the execution sequence.

Server-Side TCP Socket Programming

The ZTP server-side application contains the project `ZTPDemo_F91_Mini.zdsproj` to which the Flash Loader driver is integrated.

The main function in the `ZTPDemo_F91_Mini.zdsproj` project creates a TCP Read/Write thread, `tcp_rw`, with a stack size of 2048 bytes and a priority of 20. When this thread executes, it opens a TCP socket on port 5000 to listen for requests from any foreign port.

The `tcp_rw` thread also creates a TCP listen queue to handle multiple connection requests. The thread waits for a connection request from the client. After receiving a connection request, the TCP Read/Write thread is created, which invokes the `read()` API to receive data packets from the client and uses the `write()` API to send an acknowledgement to the client.

The received data from the client-side is stored in the receive buffer. The data packets sent by the client follow a certain format, which is discussed in the Client-Side TCP Socket Programming on page 6.

If the first element of the receive buffer is `0xFD`, it means that the client requests to establish a connection with the server-side program; `0x04` is transmitted to the client to indicate that connection is established. The server-side program waits to receive more data packets.

If the first element of the receive buffer is `0xFE`, it means that the client has requested to reset the target board; the server sends the character `0x06` to the client to indicate that the processor reset operation was successful. The socket is then closed and program execution starts from location `0x000000`.

If the first element of the receive buffer is `0xFF`, it means that the client requires the server-side program to get into the Flash loading mode; the character `0x05` is sent to the client to indicate that the Flash Loader utility is now ready to update the firmware. The value of the `operating_mode` flag is changed to `0x00` and the socket is closed. The server-side program waits for data from the client to begin programming into the external Flash memory.

If the first element of the receive buffer is a character other than `0xFF`, `0xFE`, or `0xFD`, it represents the actual data to be loaded to external Flash memory. The data to be programmed is transferred from the client, one line at a time, with a maximum of 128 bytes per line.

The first element of the receive buffer indicates the length of the data. The next three bytes indicate the starting address of the external Flash memory location to which the data is to be programmed. Accordingly, the data is programmed to external Flash memory. When programming of the single data packet is completed, the Flash Loader driver sends an acknowledgement character, 0x01, to the client to indicate that Flash loading is in progress. After this acknowledgement, the client sends the next data packet to be programmed.

When the final data packet is sent to the Flash Loader driver, the client transmits 0xFC to indicate that data transfer is complete. Upon receiving this data complete indication, the Flash Loader driver stops further programming of Flash memory and sends the 0x02 character to the client to indicate that Flash loading is complete.

The operating_mode flag value is changed to 0x01 and the program resets to begin execution from location 0x000000. Because the operating_mode flag is set to 0x01, the program disables internal Flash memory and program execution begins from external Flash.

Table 2 lists the one-byte messages that are sent by the server-side program to the client as application-level acknowledgements.

Table 2. One-Byte Messages Sent by Server-Side Program to the Client

Byte	Message
0x01	Indicates Flash Loading is under progress
0x02	Indicates Flash Loading is completed
0x03	Indicates an Address Error
0x04	Indicates Connection Established
0x05	Indicates Ready to update the firmware
0x06	Indicates Processor Reset Successful

Integrating the Flash Loader Driver with ZTP

The Flash Loader is interfaced with ZTP and is initially programmed to the internal Flash memory of the eZ80F91 MCU using ZPAK II and ZDS II. This Flash Loader driver is saved permanently in the internal Flash and is protected from power failures during the Flash programming or communication failures between the client and the server.

Follow the steps below to integrate the Flash Loader driver with ZTP:

1. Download ZTP to an appropriate file location on your PC. When the download is complete, browse to this location. Open the ZTPDemo folder, which is located in the following filepath:

   <installed directory>\ZTP\SamplePrograms

2. Install eZ80F91—ZTP Projects Library available under Application Sample Libraries or download the source code AN0224-SC01.zip available at www.zilog.com. Go to the following installation path

   ...\ZiLOG\Applications_Library\eZ80F91_ZTP_Projects_Library 1.0.0\ZTP based Flash Loader Utility\source

   and observe the following folders:

\FL_Client	Contains the Setup.exe file (Install shield Utility for client-side application)
\FL_Server	Contains the Flash Loader driver files and the ZTP configuration files
\FL_UserApp	Contains the ZTP configuration files and files to be added to the user application to integrate these files to ZTP

3. Rename the following original files that are located in the corresponding filepaths:

Main.Mini.c <installed directory>\ZTP\SamplePrograms\ZTP-Demo

4. Rename the vectors24.asm file, which is located in the following filepath:

   <installed directory>\ZDSII_eZ80Acclaim!_4.11.0\src\boot\common

Note: *Step 3 and Step 4 are required to retain the ZTP/ZDS II original files as these files are modified and supplied for use with ZTP based Flash Loader utility.*

5. Copy all the files located in the following folders to the ...\ZTPDemo folder:
   - ...\ZiLOG\Applications_Library\eZ80F91_ZTP_Projects_Library 1.0.0\ZTP based Flash Loader Utility\source\FL_Server folder
   - ...\ZiLOG\Applications_Library\eZ80F91_ZTP_Projects_Library 1.0.0\ZTP based Flash Loader Utility\include
6. Launch ZDS II eZ80Acclaim!® v4.11.0 and open the project file, ZTPDemo_F91_Mini.zdsproj, which is located in the following filepath: ...\ZTP\SamplePrograms\ZTPDemo. Select the target option eZ80F91ModDevKit_FLASH from **Project → Settings → Debugger**
7. Navigate in ZDS II eZ80Acclaim! v4.11.0 via the **Project → Add Files** menus to add the following Flash Loader driver files to the project. These driver files are located in the ..\ZTP-Demo folder:
   - update_flag.asm
   - server_socket.c
   - process.c
   - flashtypes.c
   - flashldr.c
   - main.mini.c
   - vectors24.asm
8. The default IP address can be modified in the ZTPConfig_Mini.c file and the EMAC address can be modified in the emac_conf.c file. As the vectors24.asm file is a ZDS II start-up file, Zilog® recommends you to create a backup of the original file.
9. Navigate to **Project → Settings → C → Category** within ZDS II. Select Deprecated as the Category and disable the option **Disable ANSI promotions**.
10. Navigate to **Project → Settings → Debugger → Setup** within ZDS II. The **Configure Target** dialog box appears. Set the Chip Select registers as shown below:

Chip Select	Lower Bound	Upper Bound	Control Register	Bus Mode
CS0	C0	CF	00	02
CS1	B8	B9	28	02
CS2	80	BF	00	81
CS3*	60	7F	A8	02

*CS3 is the chip select for the External Flash AM29LV160BB.

11. Check and select the Enable Flash option in the Internal Memory field of the **Configure Target** dialog box. Ensure that the address upper byte is at 00.
12. Navigate to **Project → Settings → Linker → Category**. Select **Address Space** as the **Category**, and ensure the following settings:

ROM	000000–03FFFF
RAM	B7E000–B9FFFF
ExtIO	0000–FFFF
Int I/O	0–FF
FlashInfo	0–1FF

13. Save and build the project. The ZTPDemo_F91_Mini_FLASH.linkcmd file is generated in the ..\ZTPDemo folder.
14. Warnings about multiple declarations in the vectors24.obj file are generated. Open the ZTPDemo_F91_Mini_FLASH.linkcmd file generated in the ..\ZTPDemo folder.
15. In the ZTPDemo_F91_Mini_FLASH.linkcmd file, replace

    ```
    ORDER .RESET,.IVECTS,.STARTUP,CODE,DATA
    ```

    with

    ```
    ORDER .mystartup,.RESET,
    .IVECTS,.STARTUP,CODE,DATA
    ```

    This operation ensures that code execution will start from the custom start-up file (mystartup in the vectors24.asm file).
16. In the ZTPDemo_F91_Mini_Flash.linkcmd file, delete the first occurrence of the following target linker command.

    ```
    <installed
    directory>\ZDSII_~2.1\lib\ZiLOG
    \vectors24.obj, \
    ```
17. Save the ZTPDemo_F91_Mini_Flash.linkcmd file.
18. Navigate to **Project → Settings → Linker → Category** within ZDS II eZ80Acclaim!® v4.11.0 and select **Commands** as the **Category**. Select the **Use Existing** radio button. Browse and select the saved ZTPDemo_F91_Mini_Flash.linkcmd file, which is located in the following filepath:

    ```
    <installed directory>\SampleProgramsZTPDemo
    ```

    Note: *The settings in the* ZTPDemo_F91_Mini_FLASH.linkcmd *file ensure that the external Flash memory is mapped to the CS3 Lower Bound of* 0x60h. *As a result, external Flash memory is accessible beyond* 0x600000h. *ROM* 000000-03FFFF *RAM* B7E000-B9FFFF *ExtI/O* 0000-FFFF *Int I/O* 0-FF *Flash Info* 0-1FF. *For detailed information on chip selects, refer to eZ80F91 Modular Development Kit User Manual (UM0170).*
19. Save the ZTPDemo_F91_Mini.zdsproj project.

Client-Side TCP Socket Programming

The client program is provided as Install Shield utility and is located in the ...\FL_Client folder of the Application library installation path. The folder is created after installing application library as mentioned in the previous section. Follow the steps below to download and install the client-side program on the PC:

1. Go to the ..\FL_Client folder and double-click the Setup.exe file.
2. Follow the instructions of the **InstallShield Wizard** to install the **Flash Loading Facilitator** program on the desktop of your PC.
3. Double-click the Flash Loading Facilitator icon on your desktop to open the GUI.

Figure 1 on page 7 and Figure 2 on page 10 display the flow of Flash_Loader_Client.exe client-side program (see Appendix A—Flowcharts on page 13).

The function of the client-side executable file is to establish a connection to the eZ80F91 server and begin communicating. The TCP socket program initializes the Windows TCP socket library using the WSAstartup() function. The TCP socket is opened using the socket (AF_INET, SOCK_STREAM, 0) standard API.

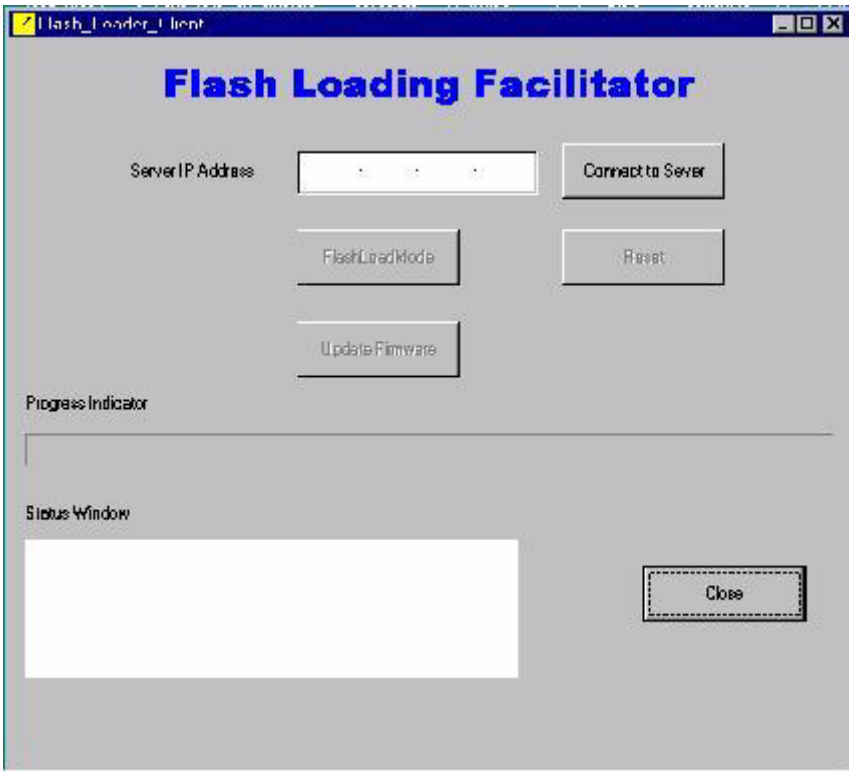


Figure 1. The GUI for Flash Loader Utility

To connect the client to the server, enter the server IP address in the **Server IP Address** text field and click the **Connect to Server** button; the server receives the request. When a connection is established, the `Connection Established` message is displayed in the GUI's status window. The eZ80F91 server address, entered via the GUI, is stored in the `servAddr.sin_addr.s_addr` variable.

When a TCP socket is opened on the client side, click the **RESET** button in the Flash Loading Facilitator GUI to transmit the `0xFE` character to request that the server restart the eZ80F91 target processor. An acknowledgement from the server is accordingly displayed in the status window.

Clicking the **Flash Load Mode** button to causes the client to transmit the `0xFF` character to the server. Upon receiving this character, the server places itself into Flash Load mode and an appropriate acknowledgement is sent to the client; this acknowledgement is displayed in the status window of the GUI.

Click the **Update Firmware** button, this allows you to select a HEX file to be downloaded. Each record (line) in the HEX file is read and stored in the following format:

Data Length (N byte)	Address (3 byte)	Data (N byte)	Check Sum (1 byte)

From this format, Data Length indicates the total number of data bytes to be programmed, excluding the 3 byte address and checksum bytes.

The 3 byte address byte indicates the starting address of the memory location in which the data is to be stored. The checksum byte is normally used for error checking. However, because the TCP protocol manages error detection and correction, transmission of the checksum byte is not required. Therefore, the checksum byte is not sent. As a result, the actual data packet transmitted through TCP adheres instead to the following format:

Data Length (N byte)	Address (3 byte)	Data (N byte)

This packet is transmitted from the client via TCP to the server. After this packet's data is successfully handled by the server, the server sends the `0x01` acknowledgement character to indicate that data is being loaded into Flash memory. The progress indicator on the Flash Loader GUI is updated as Flash loading continues.

Upon successful programming, the server sends another acknowledgement character, `0x02`, to indicate that Flash loading is complete. As a result, the status window displays the following message: `FLASH LOADING COMPLETED SUCCESSFULLY.`, and the eZ80F91 webserver is automatically reset.

The Sequence of Program Execution

By default, the status of the server, indicated by the `operating_mode` flag, determines whether program execution must occur within internal Flash or external Flash memory. During every reset of the webserver, the `operating_mode` flag is checked. When the value of the flag is `0x00`, the program residing in the internal Flash, which is the ZTP-based Flash Loader driver, starts executing. The Flash Loader driver gets ready to update the firmware (user application). When the value of the operating_mode flag is `0x01`, the firmware residing in the external Flash memory starts executing.

However, while developing the firmware, you must include ZTP-based code to listen for specific commands on the TCP socket to update the existing firmware. In addition, the code must also transfer program control back to the Flash Loader driver after the firmware is uploaded. Developing this code is discussed in the Additional Instructions for the ZTP-based Firmware section.

It must be noted that at any given time either the ZTP with the Flash Loader driver (on the Internal Flash) or the ZTP-based firmware (user application) on the external Flash memory executes and not both.

Additional Instructions for the ZTP-Based User Application

The additional instructions performed on the ZTP-based user application are detailed below:

1. Launch ZDS II—eZ80Acclaim!® v4.11.0 and open the project file `ZTPDemo_F91_Mini.zdsproj`, which is located in the following filepath: `...\ZTP\SamplePrograms\ZTPDemo`. Ensure this is a new ZTP based project; do not use the already integrated Flash loader project for this purpose.
2. Remove the `main.Mini.c` file from the `ZTPDemo_F91_Mini.zdsproj` Project directory as shown in the ZDS II project window.
3. Copy all the files located in the `...\FL_UserApp` folder to the `..\ZTPDemo` folder.
4. Add the following files located in the `...\ZTPDemo` folder to the project, using the sequence of steps **Project → Add Files:**
   - `main_Mini.c`
   - `flash_loader_interface.c`
   - `jumpto_internal_flash.asm`
   - `custom_startup_usr_app.asm`
5. Navigate to **Project → Settings → C → Category** within ZDS II. Select Deprecated as the

Category and disable the option **Disable ANSI promotions**.

6. Select the target option **eZ80F91ModDevKit_FLASH** from **Project → Settings → Debugger**. Navigate to **Project → Settings → Debugger → Setup** within ZDS II. The **Configure Target** dialog box appears. Set the following Chip Select registers as shown below:

Chip Select Register	Lower Bound	Upper Bound	Control Register	Bus Mode
CS0	C0	CF	00	02
CS1	B8	B9	28	02
CS2	80	BF	00	81
CS3	00	23	A8	02

7. Disable the option **Enable Flash** under the **Internal Memory** field of the **Configure Target** dialog box. This will disable the internal Flash of eZ80F91.
8. Navigate to **Project → Settings → Linker → Category**. Select **Address Space** as the **Category**, and ensure the following settings:

ROM	000000–23FFFF
RAM	B7E000–B9FFFF
ExtIO	0000–FFFF
Int	0–FF
FlashInfo	0–1FF

9. Save and build the project. The `ZTPDemo_F91_FLASH.linkcmd` file is generated in the `..\ZTPDemo` folder. Open the `ZTPDemo_F91_FLASH.linkcmd` file generated in the `..\ZTPDemo` folder.
10. In the `ZTPDemo_F91_FLASH.linkcmd` file, replace

    ```
    ORDER .RESET,.IVECTS,.STARTUP,CODE,DATA
    ```

    with

    ```
    ORDER .userstartup,.RESET,.IVECTS,.STARTUP,CODE,DATA
    ```

    This operation ensures that code execution will start from the custom startup file (userstartup in the `custom_startup_usr_app.asm` file)
11. Save the `ZTPDemo_F91_FLASH.linkcmd` file.
12. Navigate to **Project → Settings → Linker → Category** within ZDS II. Select Command as the Category and select the **Use Existing** radio button. Browse and select the saved `ZTPDemo_F91_Mini_Flash.linkcmd` file, which is located in the following filepath:

    ```
    <installed directory>\SampleProgams\ZTPDemo
    ```
13. Save the `ZTPDemo_F91_Mini.zdsproj` project.

Demonstrating the Flash Loader Utility

This section contains all the requirements and instructions to demonstrate the Flash Loader utility developed for the ZTP-based embedded eZ80Acclaim!® webserver. To demonstrate the Flash Loader utility, the eZ80F91 Modular Development Kit (eZ80F910100KITG), the ZPAK II unit, and the PC must be set up as shown in Figure 2. The setup shown below specifically refers to ZPAK II. For setup with other debug tools, for example, USB Smart Cable, Ethernet Smart Cable, and so on, refer to their respective user manual.

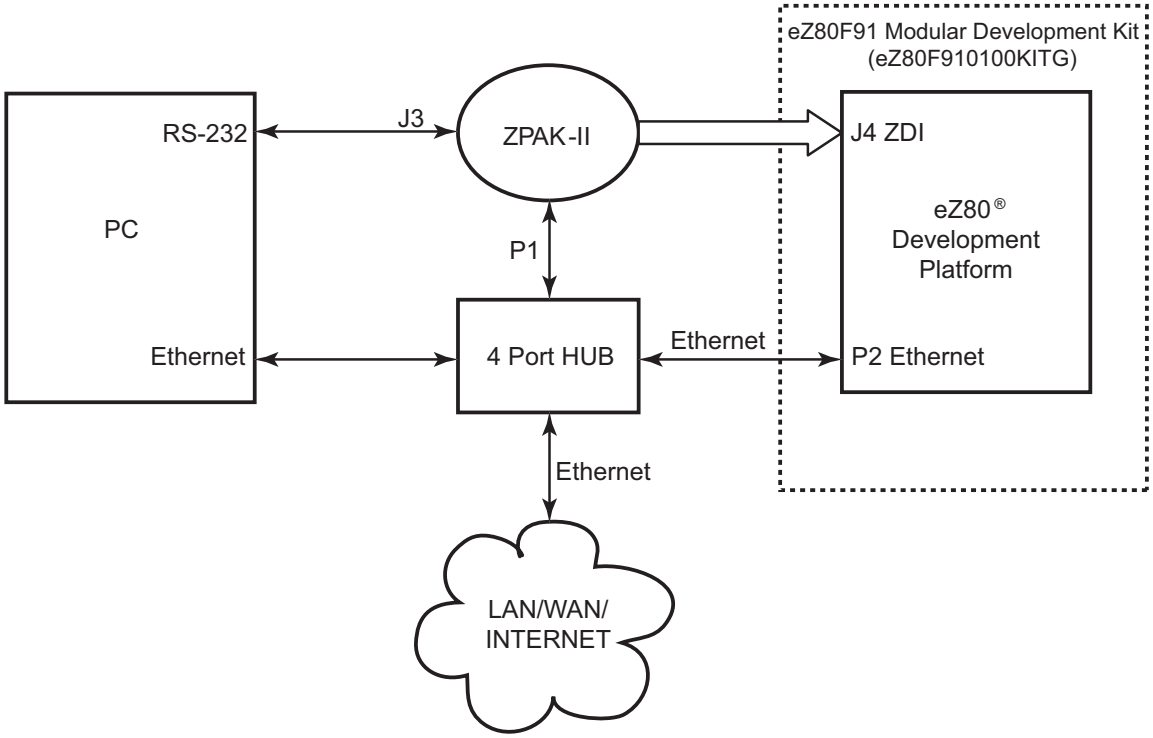


Figure 2. Setup to Demonstrate the Flash Loader Utility

The hardware and software requirements for the Demo are provided below.

Hardware Requirements

The hardware required for the Demo include:

- eZ80F91 Development Kit (eZ80F910100KITG)
- PC with an Internet Browser

Software Requirements

The software requirements to execute the Flash Loader utility include:

- Zilog Developer Studio II—IDE for eZ80Acclaim!® v4.11.0 (ZDS II eZ80Acclaim! v4.11.0).
- Zilog TCP/IP (ZTP) Software Suite v2.1.0.
- The `Flash_Loader_Client.exe` client-side program to be downloaded onto the PC contained in the folder `..\ZTP based Flash Loader Utility\source\FL_CLIENT`.
- The ZTP-based Flash Loader utility contained in the `..\ZTP based Flash Loader Utility\source\FL_SERVER` folder.
- User application (firmware) project file integrated with ZTP (files contained in `..\ZTP based Flash Loader Utility\source\FL_UserAPP` folder).

Settings

HyperTerminal Settings

The HyperTerminal settings include:

- Set HyperTerminal to 57600-8-N-1, with no flow control

Jumper Settings

For the eZ80® Modular Development Kit Platform:

- J4, J9 are ON
- J6, J8 are OFF

For the eZ80F91 Mini Module:

- JP1 is ON

External Flash

Footprint (U2) provided in the development platform is required to be populated with AMD Flash device AM29LV160BB.

Procedure to Execute the Flash Loader Utility Demo

Follow the steps below to execute the Flash Loader demo:

1. Ensure that the Flash Loader driver files are added and integrated to ZTP before proceeding. For more details, see the Integrating the Flash Loader Driver with ZTP on page 4.
2. Make the connections according to Figure 2 on page 10. Use the jumper settings provided above.
3. Connect a 5 V power supply to the eZ80 development platform of the eZ80F91 Modular Development Kit (eZ80F910100KITG).
4. Connect the 5 V power supply to the ZPAK II unit and the 7.5 V power supply to the Ethernet HUB.
5. Launch HyperTerminal and follow the settings provided in the HyperTerminal Settings on page 10.
6. From within HyperTerminal, press the `z` key repeatedly, then press the **RESET** button on the ZPAK II unit to view the menu, with which you will set the IP address of the ZPAK II unit.
7. Enter `H` to display the **Help** menu, and follow the instructions in this menu to obtain the IP address for ZPAK II for purposes of downloading the Flash Loader driver. This ZPAK II IP address must be entered using ZDS II.
8. Launch ZDS II eZ80Acclaim!® and open the `ZTPDemo_F91_Mini.zdsproj` project file, which is located in the following path:

   `..\ZTPDemo`
9. Open the `ZTPConfig_Mini.c` file. Ensure that the information in this file is relevant to your network configuration. Use the IP address in the Ethernet configuration to send the client request from the Flash Loading Facilitator GUI.
10. Build the `ZTPDemo_F91_Mini.zdsproj` project and use ZDS II to download the resulting HEX file to the eZ80F91 Mini Module on the eZ80® development platform. Before downloading from the Flash Loader window, ensure that Flash base is set at `600000h` and AM29LV160BB is selected from the available Flash devices. The Flash Loader driver now resides within internal Flash memory on the eZ80F91 MCU.
11. Using ZDS II, open the user application project file.
12. Ensure that these files are integrated with ZTP (see the Additional Instructions for the ZTP-Based User Application on page 8).
13. Compile and build the user project. The resulting HEX file must be uploaded to external Flash memory on the eZ80F91 Modular development Kit.

Uploading the User HEX File

Follow the steps below to upload the user HEX file to the external Flash of eZ80F91 Modular development Kit:

1. Execute the `Flash_Loader_Client.exe` file. The **Flash Loading Facilitator** GUI screen appears.

2. Observe that the **Connect to Server** button is the only activated button on the screen.
3. Enter the IP address for the eZ80 development platform, then click the **Connect to Server** button. The client establishes a connection with the server, and a `Connection Established` message is displayed in the status window on the GUI. Observe that the **Connect to Server** button is now deactivated and the other buttons are activated.
4. Click the **Flash Load Mode** button. The server sends a message to the client, indicating that the server is in Flash Load mode.
5. Browse and select the user application HEX file to be loaded by clicking the **Update Firmware** button. Programming of the user HEX file to the external Flash memory starts. The **Progress Indicator** is updated as the firmware update occurs.
6. When the programming of Flash memory is completed, the `Flash loading completed` message is displayed in the status window**.** The newly programmed HEX file begins to execute.
7. Repeat the procedure from Step 3 to start uploading an additional firmware application.

Summary

A ZTP-based Flash Loader utility is described in this application note. The Flash Loader utility allows you to update firmware to the external Flash memory space of the eZ80F91 MCU from a remote location via the Internet.

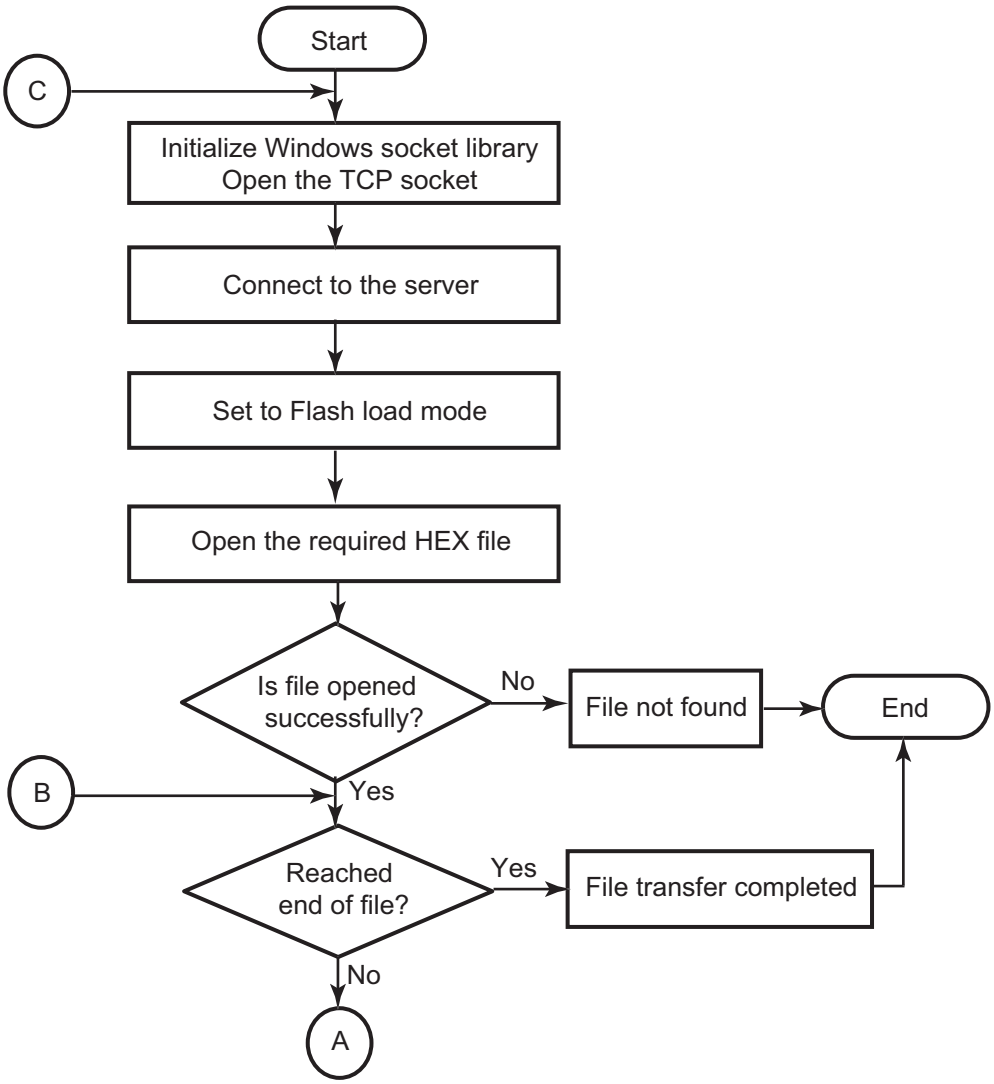
References

The documents associated with eZ80F91 MCU, the eZ80® CPU, and ZTP available on www.zilog.com are provided below:

- eZ80® CPU User Manual (UM0077)
- eZ80F91 Modular Development Kit User Manual (UM0170)
- eZ80F91 MCU Product Specification (PS0192)
- eZ80F91 Mini Enet Module Product Specification (PS0236)
- http://www.8052.com/tutintel.phtml
- Zilog TCP/IP Stack API Reference Manual (RM0040)
- Zilog TCP/IP Software Suite Programmer's Guide (RM0041)
- Zilog TCP/IP Software Suite Quick Start Guide (QS0049)
- Zilog Developer Studio II—eZ80Acclaim!® User Manual (UM0144)

Appendix A—Flowcharts

Figure 3 displays the client-side TCP socket programming flow.



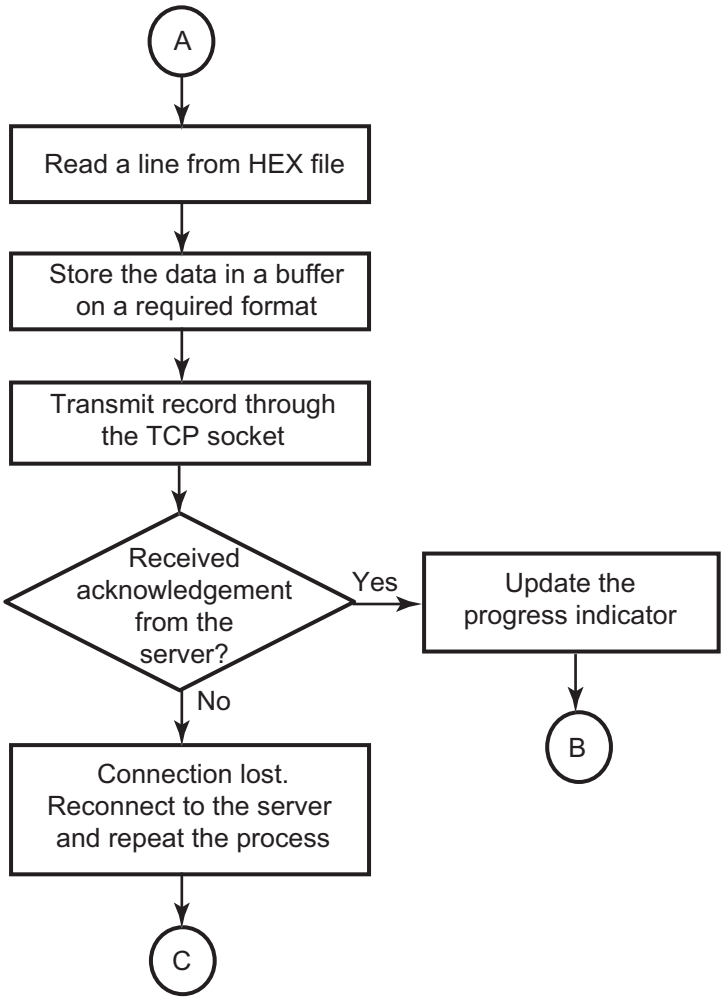


Figure 3. Client Side TCP Socket Programming Flow

Warning: DO NOT USE IN LIFE SUPPORT

LIFE SUPPORT POLICY

ZILOG'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS PRIOR WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF ZILOG CORPORATION.

As used herein

Life support devices or systems are devices which (a) are intended for surgical implant into the body, or (b) support or sustain life and whose failure to perform when properly used in accordance with instructions for use provided in the labeling can be reasonably expected to result in a significant injury to the user. A critical component is any component in a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system or to affect its safety or effectiveness.

Document Disclaimer