Application Note

Data Communication with the eZ80F91 MCU Using the User Datagram Protocol

AN017903-0208

Abstract

This Application Note describes how to use the User Datagram Protocol (UDP) to exchange data, typically less than 1500 bytes, between a eZ80F91 web server and a PC client. The eZ80F91 web server is Zilog's eZ80F91 microcontrollers unit (MCU) that executes the Zilog TCP/IP stack (ZTP), which supports the UDP protocol. The Application Note describes a client-side implementation that sends a input, and a server-side implementation that receives the client request and sends an appropriate response.

Note: *The source code for this Application Note is contained in the* `AN0179-SC01.zip` *file, available on the www.zilog.com.*

Zilog Product Overview

This section contains brief overviews of the Zilog products used in this Application Note, which includes the award-winning eZ80Acclaim*Plus!*™ MCU and the full-feature ZTP software suite.

eZ80Acclaim*Plus!*™ MCU Family Overview

The eZ80Acclaim*Plus!* family of microcontrollers includes Flash and non-Flash products. The Flash-based eZ80Acclaim*Plus!* MCU device number eZ80F91, and eZ80Acclaim!® device eZ80F92 and eZ80F93, are an exceptional value for designing high performance embedded applications. With speeds up to 50 MHz and an on-chip Ethernet MAC (eZ80F91 only), you 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 eZ80® devices without Flash memory: the eZ80L92 and eZ80190 microprocessors.

ZTP Overview

The ZTP integrates a rich set of networking services with an efficient real-time operating system (RTOS). The operating system is a compact preemptive multitasking, multithreaded 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	IP	PPP
DHCP	DNS	TIMEP	SNMP	TCP	UDP
ICMP	IGMP	ARP	RARP		

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.

Discussion

This section provides a brief overview of the UDP protocol and lists the UDP-related API functions available in ZTP. ZTP is Zilog's TCP/IP software suite[1] that includes UDP among other protocols.

[1]For more details on ZTP, refer to *Zilog TCP/IP Software Suite Programmer's Guide Reference Manual (RM0008)*.

User Datagram Protocol

The User Datagram Protocol is an IETF standard transport layer protocol that runs on the IP layer of the TCP/IP stack. It is defined by RFC 768 as a protocol for sending messages to application programs with a minimum of protocol mechanisms. The RFC also mentions the inherent unreliable nature of the UDP protocol—it is transaction-oriented but datagram delivery is not guaranteed. Therefore, UDP applications transfer data between a client and server without any acknowledgement of the communication. In such a case, user-applications must ensure that messages are sent and received reliably by using an error recovery method.

Thus UDP provides a connection-less method that is fast but inherently less reliable than data transfer over TCP sockets. However data bytes can be easily exchanged between networked nodes. Some of the applications that use UDP by default are Ping, SNMP, and DNS. Because the time required for transferring data via UDP is less, transport of audio and video data streams over UDP in embedded systems is gaining popularity.

A standard UDP datagram consists of a source and a destination IP address, the type of protocol and the data length. Figure 1 displays the UDP datagram and its components.

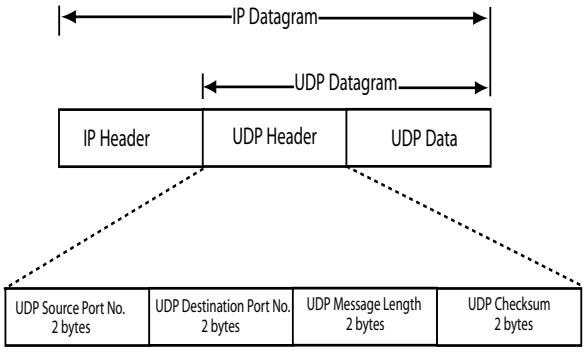

Figure 1. The UDP Datagram

UDP API Functions Available in ZTP

Table 2 lists the APIs available for applications using the UDP layer of ZTP.

Table 2. ZTP-UDP API Services

API Name	Description
Open()	Opens a UDP socket for Data transfer
Control()	Provides UDP-specific device control function
Read()	Receives a UDP data packet
Write()	Sends a UDP data packet
Close()	Closes the UDP socket signalling the end of communication

Developing the UDP Application

This section explains how to interface the eZ80F91 MCU with the PC in order to exchange UDP data packets.

Hardware Interfacing

Figure 2 is a block diagram displaying the setup connecting the eZ80 development platform and the PC. The eZ80F91 MCU contains the ZTP software suite that makes it function as a web server.

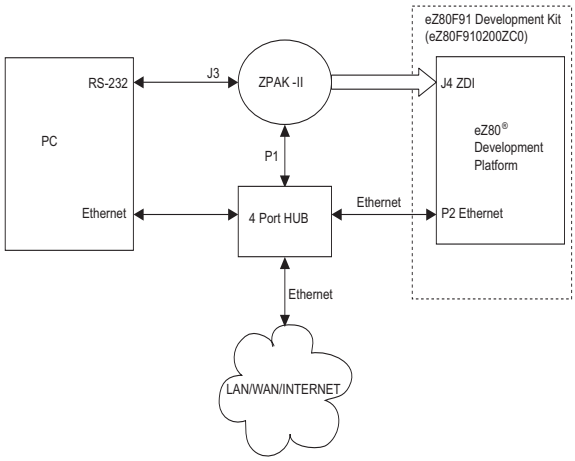

Figure 2. Block Diagram of the Setup Connecting the eZ80F91 MCU and the PC

Software Implementation

The software implementation consists of two parts, briefly described below.

Client-side Implementation—The file, `Clientmain.c` (available in `AN0179-SC01.zip` file), runs on the PC client and is coded in ANSI C. For details of this implementation, see Client-Side Implementation. The flowchart for the client-side software is displayed in Figure 3 on page 9.

Server-side Implementation—The server-side software is implemented as a thread in the ZTP stack, and runs on the eZ80F91 web server. For details of this implementation, see Server-Side Implementation. The flowchart for the server-side software is displayed in Figure 4 on page 10.

The client and server exchange data messages in the form of UDP datagrams.

The following sections describe the details of the client-side and the server-side implementations.

Client-Side Implementation

The following tasks comprise the client-side implementation present in the `Clientmain.c` file:

- The standard APIs available with the Microsoft Windows' TCP/IP stack are included in the `Clientmain.c` file using the statement,

  `#include <winsock2.h>`.
- A buffer size of 100 bytes is allocated for the incoming and outgoing UDP messages in the variables `inBufferSize` and `outBufferSize`, respectively. Initialization of the Windows-side socket is accomplished using the `WSAStartup()` function; an error message is printed on failure to initialize.
- The eZ80F91 server address (in IPv4 format) is stored in the variable `servAddr.sin_addr.s_addr`. This IP address is assigned by the LAN network system administrator for every PC on the network. The `Clientmain.c` file contains a default IP address that should be changed to the IP address specific to you.
- The server-side UDP port number (default value = 5009) is stored in the variable `servAddr.sin_port` and can be changed to any other available port number.
- The client-side port number (default value = 5006) is stored in the variable `local.sin_port`. This default value can be changed according to the availability of port numbers on the PC.

When `Clientmain.c` program is executed, an input menu, is displayed at the DOS prompt. The input is then formatted into a UDP packet/datagram and sent to the server using the ZTP-UDP API `send()`. A typical UDP datagram formed by the `send()` API is displayed in Figure 1 on page 2.

The section, Server-Side Implementation, describes how the server handles the UDP datagram. The `recv()` API is used to collect the response from server; the response is stored in the buffer memory. When the client successfully receives a response from the server, the socket is closed and the received data is printed in the DOS window along with the number of bytes received.

Server-Side Implementation

The following tasks comprise the server-side implementation present in the file `udp_ez80.c`. The file `udp_ez80.c` contains the `test()`, `udpnormal()`, and `normal()` functions.

- The main thread of XINU executes the `test()` function upon initialization. This action sets up a command, `udptest`, in ZTP's shell structure.
- The `udptest` command, along with an argument, calls the `udpnormal()` function, which creates a thread—the `normal()` function—in

the XINU environment, with priority number 20.

- The IP address of the client PC and the UDP port numbers for both the client PC and server are supplied as arguments to the ZTP-UDP API `open()` function.
- The server sends a message upon receiving input for the menu displayed at the Windows DOS prompt. The message—*This is message number one/This is message number two*—is sent by the server depending on the number entered by you.
- This message is first copied into the `xgram` data structure and then sent using the ZTP-UDP API `write()` function. The size of the UDP packet is determined from the argument provided with the `udptest` command in HyperTerminal (see Executing the UDP Demo Application on page 7). The run-time errors, including socket open error, memory unavailable, or UDP read and write failure, are printed in the HyperTerminal along with the error codes to simplify debugging.

Adding and Integrating UDP Demo-Specific Files to ZTP

The UDP Demo described in this Application Note requires the eZ80 development platform with an eZ80Acclaim*Plus!* microprocessor/controller and the ZTP stack. For the Demo execution, some of the files specific to the demo must be added and integrated to the ZTP stack before the stack is downloaded onto the eZ80 development platform. This section describes how to add the Demo files to the ZTP stack.

The Demo files that must be added to the ZTP project files are in the `AN0179-SC01.zip` file, available for download at www.zilog.com. The Demo files are of the following types:

- C (`*.c`) files

The ZTP stack is available on the Zilog website and can be downloaded to a PC with a user-registration key. ZTP can be installed in any location as specified by you; its default location is `C:\Program Files\ZiLOG`.

Follow the steps below to add and integrate the UDP Demo files to the ZTP stack:

1. Download ZTP, browse to the location where ZTP is downloaded, and open the `\web-site.Acclaim` folder.
2. Download the `AN0179-SC01.zip` file and extract its contents to a folder on your PC (this folder is referred to as `\UDP_Demo` folder in the rest of the Application Note). Notice the two extracted folders within the `\UDP_Demo` folder:

   `\Server side_Demo`

   `\Client side_Demo`
3. Select and copy all the *.c, and `*.h`, files located in the `\UDP_Demo\Server side_Demo` folder and paste them into the `..\ZTP\Demo` directory.
4. Launch ZDS II, and open the `Acclaim-Demo.pro` file available in the path: `..\ZTP\Demo` directory.
5. Add the `udp_ez80.c` file located in the `..\UDP_Demo\Server side_Demo` folder to the project, using the sequence of steps: **Project → Add Files.**
6. Open the `udp_ez80.c` file from within ZDS II. Change the client IP address and port number in the following line of code:

   ```
   if((dev = open(UDP, "192.1.6.75:5006",(char *) 5009)) == SYSERR)
   ```

   where,

   `192.1.6.75:5006` must be substituted with the IP address and the port number of the client machine being used, and

   `5009` must be substituted with the port number assigned to the eZ80F91 web server.

7. Open the main.c file of the AcclaimDemo project and enter the IP address for the eZ80F91 module in the following BootInfo structure definition:

```
struct BootInfo Bootrecord = {
     "192.168.1.1",/* Default IP address */
     "192.168.1.4",/* Default Gate-way */
     "192.168.1.5",/* Default Timer Server */
     "192.168.1.6",/* Default File Server */
     "",
     "192.168.1.7",/* Default Name Server */
     "",
     0xffffff00UL/* Default Subnet Mask */
     };
```

The Bootrecord variable contains the network parameters and settings (in the four-octet dotted decimal format) that are specific to the local area network at Zilog as default.

Note: *Modify the above structure definition with appropriate IP addresses within your local area network.*

8. In the main.c file, add the following function prototypes and global variables:

```
//prototype functions
void test (void);
```

9. In the same main.c file, after the line shell_init(fd);, add the function, test();.

10. Open the eZ80_HW_Config.c file and change the default MAC address (provided by ZTP) such that each eZ80 development platform on the LAN contains a unique MAC address. The following line of code is present in the eZ80_HW_Config.c file:

```
const BYTE f91_mac_addr [EP_ALEN] = {0x00, 0x90, 0x23, 0x00, 0x0F, 0x91};
```

In the 6-byte MAC address displayed above, the first three bytes must not be modified; the last three bytes can be used to assign a unique MAC address to the eZ80 development platform.

11. Open the ipw_ez80.c file. For this application, DHCP is disabled; therefore, ensure the following:

```
b_use_dhcp = FALSE
```

12. Save and close the AcclaimDemo project.

Compiling and Building the Client-Side Project

Follow the steps below to compile and build the client-side project:

1. Launch the Microsoft Visual C++ IDE and open the file Clientmain.c (located in the\UDP_Demo\Client side_Demo folder).
2. Set the IP address and the port number of the eZ80F91 web server in this file, as explained in Client-Side Implementation on page 3.
3. From within the Microsoft Visual C++ IDE navigate to **File → New → Projects.**
4. Select **Win32 Console Application** from the list in the left panel and provide a project name and path for the client-side project.
5. Click **OK**. The **Win32 Console Application** dialog box is displayed.
6. Select **An Empty Project** radio button in answer to the question on the kind of project to be created; click **FINISH** to close the **Win32 Console Application** dialog box.
7. Navigate to **Project → Add to Project → Files.** Browse to the Clientmain.c file and add it to the client-side project.

8. Navigate to **Project** → **Settings** → **Link** and add the library file, `ws2_32.lib` in the **Object/library module** text field.
9. Compile and build the client-side project.
10. Save and close the client-side project.

To execute the client-side program, see Executing the UDP Demo Application on page 7

Demonstrating the UDP Application

This section contains the requirements and instructions required to setup and run the UDP Demo. The Demo demonstrates the exchange of datagrams between the eZ80F91 MCU running ZTP (with UDP services) and a PC client running the UDP services of Microsoft Windows-based TCP/IP stack.

Setup

The basic setup to assemble the Demo is displayed in Figure 2 on page 2. This setup displays the connections between the PC, LAN/WAN/Internet, and the eZ80 Development Platform with the eZ80F91 Module.

The requirements are classified under hardware and software.

Hardware Requirements

The hardware requirements to execute the UDP Demo are as follows:

- eZ80F91 Development Kit, which includes the following:
  - eZ80 Development Platform
  - eZ80F91 Module
  - 9 V DC Power Supply
  - ZPAKII Debug Interface Module, with power supply
  - 4-port 10 BaseT Ethernet Hub with power supply
- PC with HyperTerminal and Microsoft Visual C++ IDE

Software Requirements

The software requirements to execute the UDP Demo are as follows:

- Zilog Developer Studio II—IDE for eZ80Acclaim! (ZDS II)
- Zilog's TCP/IP Software Suite (ZTP)
- Microsoft Visual C++ IDE to build the client-side program

Settings

HyperTerminal Settings

The HyperTerminal settings include:

- Set HyperTerminal to 57.6 kbps Baud and 8-N-1, with no flow control

Jumper Settings

For the eZ80 Development Platform are as follows:

- J2 is ON
- J3, J7, J11, J20, J21, J22 are OFF
- For J14, connect 2 and 3
- For J19, CS_EX_IN is ON, MEM_CEN1 and, MEM_CEN2, and MEM_CEN3 are OFF

For the eZ80F91 Module on the eZ80 development platform are as follows:

- JP3 is OFF

Procedure

Follow the procedure below to setup the UDP Demo application prior to execution:

1. Ensure that the required Demo files are added and integrated to ZTP before proceeding. For details, see Adding and Integrating UDP Demo-Specific Files to ZTP on page 4.

2. Make the connections as per Figure 2 on page 2. Follow the jumper settings provided in Jumper Settings.
3. Connect the 9 V power supply to the eZ80F91 Development Kit.
4. Connect the 5 V power supply to ZPAKII and the 7.5 V power supply to the Ethernet HUB.
5. Launch the HyperTerminal and follow the settings provided in the HyperTerminal Settings section.
6. From within the HyperTerminal, press z repeatedly, and then press the reset button on ZPAKII to view the menu to set the ZPAKII IP address.
7. Enter H to display help menu, and follow the menu instructions to obtain the IP address for ZPAKII in order to download the Demo file. This ZPAKII IP address must be entered in the ZDS II.
8. Launch ZDS II—eZ80Acclaim! and open the Demo project file (`AcclaimDemo.pro`) located in the path: `..\ZTP\Demo.`
9. Open the `main.c` file. Ensure that the `BootInfo` structure contains information that is relevant to your network configuration.
10. Build the AcclaimDemo project and download the resulting file to the eZ80F91 Module on the eZ80 development platform, using ZDS II.
11. Open the file `Clientmain.c` in Microsoft Visual C++ IDE. Ensure that the IP address and port number of the eZ80F91 web server in this file are set as explained in Client-Side Implementation on page 3.
12. Compile and build the `Clientmain.c` file into a new project as explained in Compiling and Building the Client-Side Project on page 5.
13. Run both the client- and server-side projects. For more information, see Executing the UDP Demo Application.

Executing the UDP Demo Application

Follow the below procedure to execute and test the UDP Demo application.

1. Execute the AcclaimDemo by clicking the **GO** icon in the ZDS II tool bar.
2. Execute `Clientmain.c` from Microsoft Visual C++ IDE by pressing **CTRL+F5.** A DOS window opens up as the user-interface with the following menu:

```
Menu: :
   1. Read Flash message1
   2. Read Flash message2
Enter 1 or 2
```

3. In the HyperTerminal console, at the prompt, enter

   `udptest 100`

   to execute the `normal()` thread on the eZ80F91 web server. The argument `100` specifies the size of the UDP packet.
4. In the DOS window, enter either number 1 or 2 after the menu display and press **Enter**. The number is sent as a UDP message to the eZ80F91 web server.

   The response (a UDP packet) is parsed by the PC client-side program and is printed in the DOS window. Observe that the message,

   `This is message number one`

   is displayed at the DOS prompt when 1 is entered.

   And,

   `This is message number two`

   is displayed at the DOS prompt when 2 is entered.
5. To exit from the program, make the HyperTerminal window active, press **Enter**, and close all running programs.

Summary

This Application Note demonstrates the exchange of datagrams between a PC client and the eZ80F91 web server using the UDP layer of ZTP stack. Compared to TCP, UDP functions are easy to use and provide a faster way of exchanging messages. UDP is beneficial for eZ80F91-based products, provided that all the errors arising in this kind of transaction are taken care of.

The source code for programs provided in this Application Note for both the client-side and the server-side, can be used as templates, to build complex applications. UDP is the underlying protocol for SNMP, TFTP, BOOTP, TIMEP and DNS protocol layers, all of which are supported in ZTP.

References

The documents associated with eZ80Acclaim!®, and eZ80Acclaim*Plus!*™ family of products are listed below:

- The full text of the RFC 768 is available at the URL http://www.ietf.org/rfc/rfc768.txt
- eZ80F91 MCU Product Specification (PS0192)
- Zilog Developer Studio II—eZ80Acclaim! User Manual (UM0144)
- Zilog TCP/IP Software Suite Programmer's Guide Reference Manual (RM0008)

Appendix A—Flowcharts

This Appendix displays the flowcharts for the UDP application implementation described in this Application Note.

Figure 3 displays the flowchart for the client-side program, `Clientmain.c`.

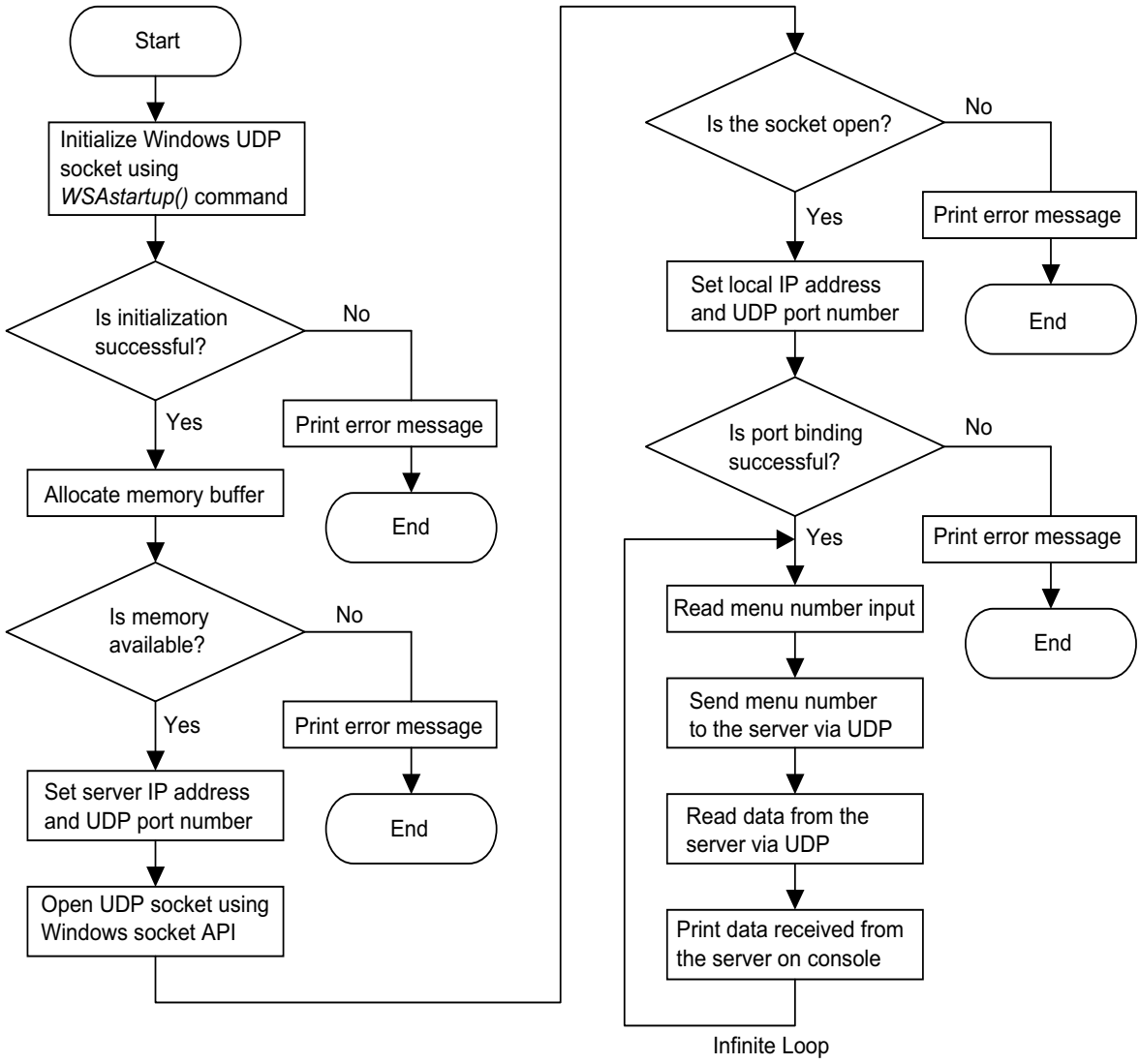

Figure 3. Flowchart for the Client-Side Program

Figure 4 displays the flowchart for the execution of the UDP-related thread, the `normal()` function.

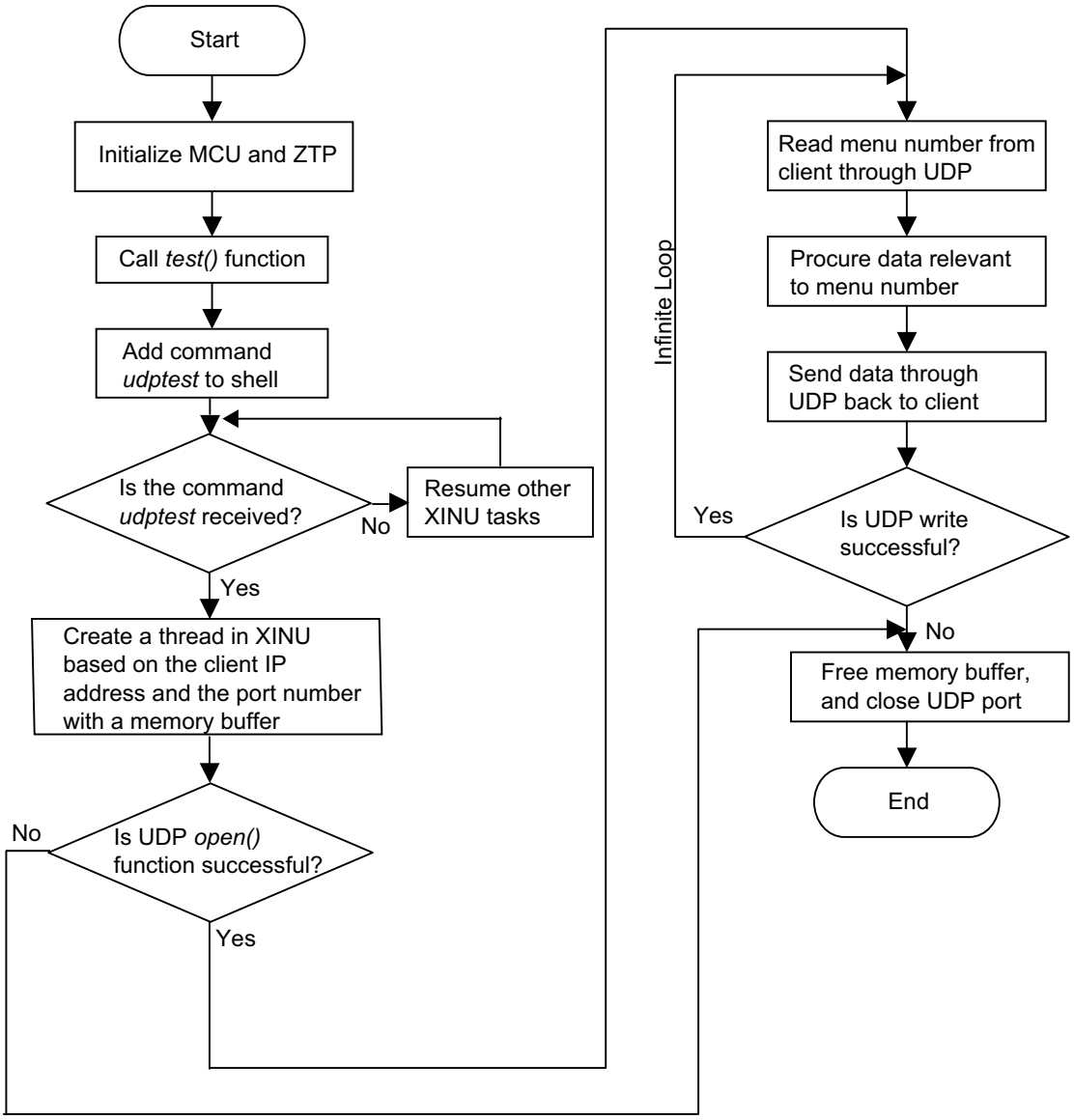

Figure 4. Flowchart for the UDP Thread, the `normal()` Function

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