ZGATE™ Technologies

ZGATE™ Embedded Security Development Kit

User Manual

UM024502-1012

Warning: DO NOT USE THIS PRODUCT IN LIFE SUPPORT SYSTEMS.

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

 Information in this publication concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded. ZILOG, INC. DOES NOT ASSUME LIABILITY FOR OR PROVIDE A REPRESENTATION OF ACCURACY OF THE INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED IN THIS DOCUMENT. ZILOG ALSO DOES NOT ASSUME LIABILITY FOR INTELLECTUAL PROPERTY INFRINGEMENT RELATED IN ANY MANNER TO USE OF INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED HEREIN OR OTHERWISE. The information contained within this document has been verified according to the general principles of electrical and mechanical engineering.

ZGATE, eZ80, eZ80Acclaim! and eZ80Acclaim*Plus!* are trademarks or registered trademarks of Zilog, Inc. All other product or service names are the property of their respective owners.

Revision History

Each instance in the Revision History table below reflects a change to this document from its previous version. For more details, click the appropriate links in the table.

Date	Revision Level	Description	Page
Oct 2012	02	Corrected figures for improved image quality.	17, 35, 36, 37, 39, 41, 42
Oct 2012	01	Original issue.	n/a

Safeguards

The following precautions must be observed when working with the devices described in this document.

Caution: Always use a grounding strap to prevent damage resulting from electrostatic discharge (ESD).

Table of Contents

List of Figures

List of Tables

The ZGATE Embedded Security Development Kit

ZGATE technology incorporates the eZ80F91 MCU and Zilog's full-featured TCP/IP stack with a world-class embedded firewall. This highly-configurable firewall protects the ZTP networking layers from attack by discarding suspicious packets before they reach ZTP and your embedded application. The ZGATE firewall includes a static packet filtering engine that filters packets according to user-defined configuration rules and a stateful packet inspection engine that can automatically filter suspicious packets based on unusual activity. Additionally, select ZGATE products include threshold-filtering mechanisms that can minimize the affect of packet floods.

Zilog's ZGATE Embedded Security Development Kit, part number ZGATE000100ZCOG, provides a general-purpose platform for creating a design based on this eZ80F91 microcontroller, which has been preprogrammed with a ZGATE security code. The eZ80F91 MCU is a member of Zilog's eZ80Acclaim*Plus!* product family, which offers an on-chip EMAC and Flash memory.

This document provides instructions for setting up and configuring your ZGATE Embedded Security Development Board and includes schematic diagrams and a discussion of Board features and ZDSII.

The first sections of this document guide you through the following tasks:

- Download and Install the ZDSII Software and Documentation on page 4
- Installing the USB Smart Cable Driver on page 4
- Installing the FTDI USB-to-UART Driver on page 7
- Connecting the ZGATE Embedded Security Development Board to your PC on page 8
- Starting the ZGATE Demo Sample Program on page 11

Further details, including memory configurations, jumper settings and a listing of sample projects can be found in Appendix A. ZGATE Embedded Security Development Board on page 105.

Figure 1 shows an image of the ZGATE Embedded Security Development Kit.

Figure 1. The ZGATE Embedded Security Development Kit

Kit Contents

Table 1 lists the contents of the ZGATE Embedded Security Development Kit.

Table 1. ZGATE000100ZCOG Contents

Item	Description	Quantity
1	ZGATE Embedded Security Development Board	1
2	USB Smart Cable	1
3	6-Circuit Ribbon Cable	1
4	A (male) to Mini-B USB Cable	1
5	ZGATE Embedded Security Development Kit Flyer (FL0145)	1

Kit Features

The key features of the ZGATE Embedded Security Development Kit are:

- ZGATE Embedded Security Development Board, which includes:
  - eZ80F91 MCU operating at 50 MHz, with 256 KB of internal Flash memory and 8 KB of internal SRAM memory
  - On-chip Ethernet Media Access Controller (EMAC)
  - 8 MB of Flash memory
  - Up to 1 MB of off-chip SRAM memory
  - A USB interface that provides:
    - Power to the Board with overcurrent protection
    - Connection to the eZ80F91 MCU's UART0 block
  - DB9 connected to the eZ80F91 MCU's UART1 block
  - Optional external power connection
  - Ethernet port and PHY
  - Real-Time Clock support
  - One 64-pin header with all available GPIO ports connected to it
- USB Smart Cable
- ZDS II Software and Documentation (free download)

Supported Operating Systems

The ZGATE Embedded Security Development Board supports the following operating systems:

- Microsoft Windows 7 (32-bit/64-bit)
- Microsoft Windows Vista (32-bit/64-bit)
- Microsoft Windows XP

Download and Install the ZDSII Software and Documentation

Observe the following steps to install your ZDSII software and documentation.

Note: If you have already installed *ZDSII – eZ80Acclaim! <version>* and have downloaded the ZGATE software and documentation by following the procedure on the paper insert in your kit (FL0145), skip ahead to the Installing the USB Smart Cable Driver section.

1. Prior to connecting the ZGATE Embedded Security Development Board to your development PC, download ZDS II for eZ80Acclaim! v5.2.1 (or later) from the **Downloadable Software** category in the Zilog Store.
2. Run the software installation file and follow the on-screen instructions to install ZDSII.

Download and Install the Source Code and Documentation

ZGATE software and documentation is available as a downloadable file from the Zilog Store. Observe the following brief procedure to download and install your ZGATE software.

1. In a web browser, visit the Zilog Store. At the top left, under **Categories**, click **Downloadable Software** to present a list of the Zilog software available in the Store. In this list, click **ZGATE Software and Documentation**; the *Product ID* for this software is SD00019. On the **Product Details** page, click the **Add to Cart** button to download the ZGATE Software and Documentation files to your hard drive.[1]
2. When the download is complete, unzip the file to your hard drive. Double-click the installation file named `ZGATE000100ZCOG_<version>.exe`, and follow the on-screen instructions.

Installing the USB Smart Cable Driver

The USB Smart Cable can be installed on PCs that run on Windows 7 (32- and 64-bit), Windows Vista (32- and 64-bit) and Windows XP operating systems. The procedures in this section will guide you through the USB Smart Cable installation process.

1. If you're a first-time visitor to the Zilog Store, you will first be required to register as a Zilog Store user before downloading your software. Returning visitors must log in to purchase or download.

Windows 7 32/64 Systems

Observe the following steps to install the USB Smart Cable on a Windows 7 system.

1. Connect the USB Smart Cable to a USB port on your development PC. When the PC detects the new hardware, it will display the Installing device driver software dialog.
2. Windows automatically searches for the driver; this process can take a few moments. Because there is no option to terminate this search process, wait for the search to complete.

   If the driver was previously installed, Windows will automatically install the USB Smart Cable driver. If this is the case, skip ahead to Step 9. If Windows cannot find the driver, close the search dialog and proceed to the next step.
3. From the **Start** menu, navigate via the **Search Programs and files** menu, and enter **Device Manager** in the Search field to cause the Device Manager to appear in a list of search results.
4. From this search list, click **Device Manager** to open the Device Manager dialog, which presents a list of devices that operate on your PC. Find **Other devices**, toggle it to view a sublist of additional devices, and right-click your mouse on **USB Smart Cable**.
5. In the submenu that appears, click **Update Driver Software...**.
6. In the **Update Driver Software – USB Smart Cable** dialog that appears, click the **Browse my computer for driver Software** option.
7. Click the **Browse...** button to browse to one of the following driver directories, depending on the configuration of your PC.

   On 32-bit Windows 7 systems, navigate to:

   ```
   <ZDSII Installation Directory>\device drivers\USB\x32
   <ZDSII Installation CD>\device drivers\USB\x32
   ```

   On 64-bit Windows 7 systems, navigate to:

   ```
   <ZDSII Installation Directory>\device drivers\USB\x64
   <ZDSII Installation CD>\device drivers\USB\x64
   ```
8. Click **Next** to install the driver. On 32-bit: Windows systems, a security dialog will appear; select **Install this driver software anyway**.
9. Click **Close** after the Wizard finishes the installation.

Windows Vista 32/64 Systems

Observe the following steps to install the USB Smart Cable on a Windows Vista system.

1. Connect the USB Smart Cable to a USB port on the development PC.

2. After the PC detects the new hardware, it will display the Found New Hardware Wizard dialog box. Click **Locate and install driver software (recommended)**.
3. Depending on your development PC's User Account Control settings, Windows may ask for permission to continue the installation. Click **Continue**.
4. When the Insert the Disc dialog appears, select **I don't have the disc. Show me other options.** Click the **Next** button to display the **Windows couldn't find driver** dialog.
5. Select **Browse my computer for driver software (advanced)** to display the Browse For Driver dialog, which prompts you to key in or browse for the location of the driver's `.inf` file. Depending on the type of computer you use (32- bit or 64-bit), use the **Browse...** button to navigate to one of the following paths, then click the **Next** button.

   On 32-bit Vista systems, navigate to:

   ```
   <ZDSII Installation>\device drivers\USB\x32
   <ZDSII Installation CD>\device drivers\USB\x32
   ```

   On 64-bit Vista systems, navigate to:

   ```
   <ZDSII Installation>\device drivers\USB\x64
   <ZDSII Installation CD>\device drivers\USB\x64
   ```

6. When the Windows Security dialog prompts you whether to install or not install, click **Install this driver software anyway** and wait until the installation is completed (Windows may prompt you more than once).
7. When the software has been installed successfully, click **Close**.

Windows XP Systems

Observe the following steps to install the USB Smart Cable on a Windows XP system.

1. Connect the USB Smart Cable to a USB port on the development PC. When the PC detects the new hardware, it will display the Found New Hardware Wizard dialog.
2. In the Wizard, select **Install from a list or specific location (Advanced)**, and click **Next**.

Note: If the Windows Hardware Installation dialog appears, click **Continue Anyway**.

3. In the Please choose your search and installations dialog, select **Search for the best driver in these locations and include this location in search**.
4. Use the **Browse...** button to navigate to one of the following paths:.

   ```
   <ZDSII Installation>\device drivers\USB\x32
   ```

   ```
   <ZDSII Installation CD>\Device Drivers\USB\x32
   ```

5. Click **Next** to locate the appropriate driver.
6. Click **Next**, then click **Finish** to complete the installation.

Installing the FTDI USB-to-UART Driver

An FTDI USB-to-UART driver is required to allow your PC to communicate through its USB port to the on-chip UART of the ZGATE Embedded Security MCU. Observe the following procedure to perform these connections.

1. Ensure that the USB cable is not plugged in to the ZGATE Embedded Security Development Board's P3 connector.
2. Navigate to the following filepath and double-click the CDM20802_setup.exe file to begin the driver installation.

   ```
   <ZDS II Installation>\device drivers\FTDI Uart
   <ZDS II Installation CD>\Device Drivers\FTDI Uart
   ```

3. The installation process will begin and you should observe output similar to the following messages on the screen of your PC:

   ```
   32-bit OS detected
   <installation path>\dpinstx86.exe
   Installation driver
   FTDI CDM driver installation process completed...
   ```

4. When the installation is complete, plug in the Mini-B connector of the second USB cable into the Board, and the larger A connector into the USB port of your PC.
5. If the driver installation was successful, the Ports (COM & LPT) section of the Device Manager will display USB Serial Port (COMx) or similar message, as highlighted in Figure 2.

Figure 2. A Successful USB-to-UART Driver Installation

Note: To launch the Device Manager on Windows 7 systems, launch the Start menu, enter device manager in the Search programs and files field, and press the Enter key.

To open the Device manager on earlier Windows systems, navigate via the following path:

Start ⟶ Control Panel ⟶ System ⟶ Hardware ⟶ Device Manager ⟶ Ports (COM& LPT)

Connecting the ZGATE Embedded Security Development Board to your PC

Observe the following procedure to connect the ZGATE Embedded Security Board to your PC.

Caution: Disconnect or turn off the power to the ZGATE Embedded Security Development Board before connecting or disconnecting the USB Smart Cable.

1. Ensure that the following default jumper settings are configured (see Table 4 on page 108 for reference):

J11	OUT
J12	1–2
J26	IN
J25	2–3
J24	1–2
J23	1–2

2. Connect one end of the 6-circuit ribbon cable provided in your Kit to the USB Smart Cable unit, ensuring that the ribbon's male connector is aligned correctly with the female connector on the unit, as indicated by the red stripe in Figure 3.

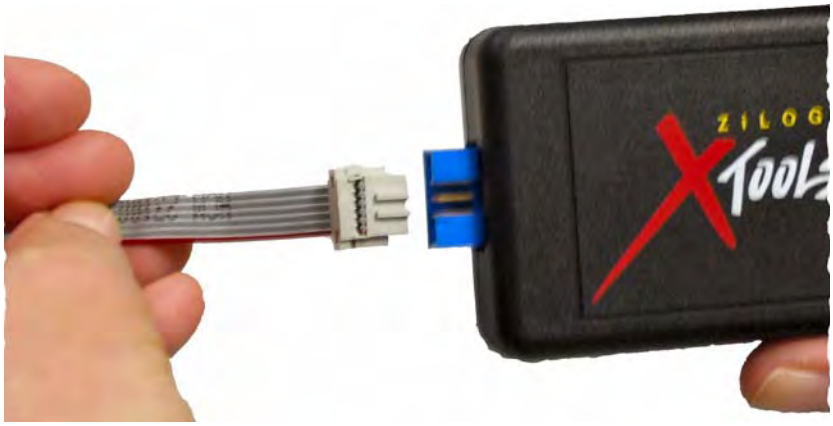


Figure 3. Connecting the Six-Conductor Ribbon Cable to the Serial or USB Smart Cable

3. Connect the other end of the ribbon cable to Debug Connector J1 on the Development Board. Ensure that Pin 1 on the ribbon cable is aligned with Pin 1 on the target connector, as highlighted in Figure 4.

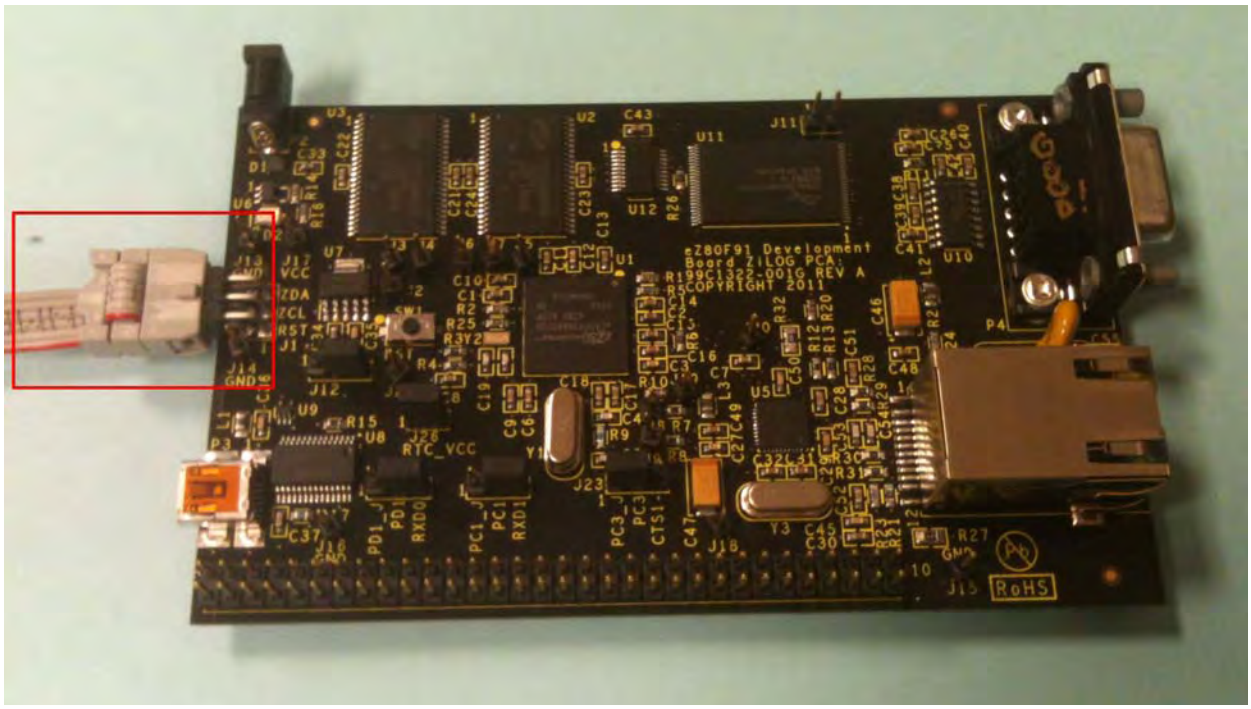


Figure 4. Debug Connector J1

4. Connect an Ethernet CAT5 cable to P1 and to your Ethernet hub.
5. With the USB A (male) to Mini-B cable, connect Port P3 on the ZGATE Embedded Security Development Board to a USB port on the development PC to apply power to the Development Board, as highlighted in Figure 5.

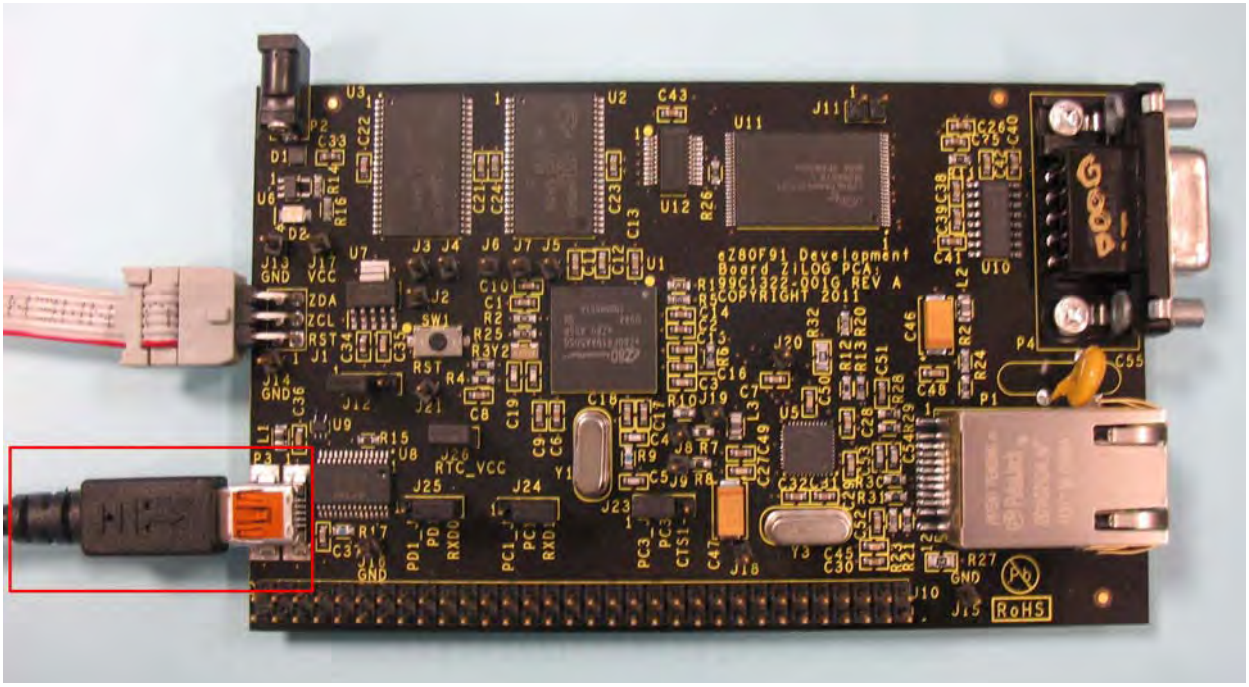


Figure 5. USB-to-UART Port 3 Connector

Note: To use the USB port as a power source, adjust the shunt on J12 to the 1–2 position.

Starting the ZGATE Demo Sample Program

The ZGATE Embedded Security Development Kit includes a sample program that demonstrates how the ZGATE API can be used to enhance the security of a ZTP application. The ZGATE Demo program includes source code to implement several shell commands that modify the filtering behavior of ZGATE at run time. In addition the ZGATE Demo program includes a sample web page with dynamically generated content that can be used to modify ZGATE's configuration using a web browser.

Before starting the ZGATE Demo it is necessary to first complete the installation of ZDS II and the Zilog TCP/IP Software Suite (ZTP). In addition to run the ZGATE Demo shell commands it will be necessary to setup a terminal emulation program (such as HyperTerminal or Tera Term). The terminal program should be configured for 8N1 with no flow control.

To get started with the ZGATE Demo use the following procedure.

1. Launch ZDSII by navigating from the Windows Start menu to **Programs** → **Zilog ZDSII – eZ80Acclaim! <Version>** → **ZDSII – eZ80Acclaim! <Version>**.
2. From the **File** menu in ZDSII, select **Open Project**, and navigate to the following file-path:

   <ZDS Install>\ZTP\ZTP<version>_Lib\ZTP\SamplePrograms\ZGATE_Demo

3. Select the `ZGATE_Demo_ZGATE000100ZCOG.zdsproj` project from within the `ZGATE_Demo` folder and click **Open**. A list of source files will appear in the Workspace panel section.
4. From the **Build** menu, select **Set Active Configuration** to open the Select Configuration dialog box.
5. Select **RAM**, then click **OK** to close the Select Configuration dialog box.
6. From the **Project** menu in ZDSII, select **Settings** to open the Project Settings dialog box. In the Project Settings dialog box, click the **Debugger** tab.
7. On the Debugger page, select **ZGATE000100ZCOG_RAM** from the Target list, then select **USB Smart Cable** from the **Debug Tool** drop-down menu.
8. Click **OK** to close the Project Settings dialog box.
9. If you are prompted to rebuild any affected files, click **Yes**. Otherwise, choose **Build** from the menu bar, then click **Rebuild All**.
10. To run the application, select **Go** from the **Debug** menu.
11. After the application has started, console output should be visible in the terminal emulation program.

For information about how to use the `ZGATE_Demo` sample program, please refer to the Using the ZGATE Demo Sample Program section on page 13.

Recovering the ZGATE Image in eZ80F91 Internal Flash

The ZGATE demo program will not function unless the ZGATE binary image is present in eZ80F91 internal Flash memory. If the ZGATE internal Flash image is accidentally erased or overwritten, please contact the Zilog Technical Support team for assistance in reprogramming the ZGATE binary image into eZ80F91 internal Flash.

After the ZGATE binary image has been restored, it may be necessary to reprogram the ZGATE demo application into the ZGATE Embedded Security Development Board. To learn more, refer to the Starting the ZGATE Demo Sample Program section on page 11.

Using the ZGATE Demo Sample Program

When the ZGATE Demo program starts, the standard ZTP start-up messages are displayed on the console, along with the ZGATE start-up information, as the following example shows.

```
.100 Mbps Full-Duplex Link established

Querying DHCP Server...

DHCP OK

Initializing network stack...

IF  IP addr       Def Gtway      state   type    H/W addr

0   192.168.2.29  192.168.2.1    UP      Ethernet 0 :90:23:0 :1 :1
1   192.168.2.1   192.168.2.2    DOWN    PPP      --


Initializing File System...
Manufacturer code : 0x1
Device Id: S29GL064NTB Failed
Formatting Invalid Volume : EXTFDone
FTPD ready
TELNETD ready
Thu, 4 Sep 2012  9:38:21

ZGATE Firewall v1.00a

HTTPD ready
Login:
A Trap is generated

A Trap is generated
```

When prompted, log in using `anonymous` as both the username and the password.

At this point, the ZGATE Demo functions similarly to the standard ZTP Demo application, with the exception that the ZGATE image stored in eZ80F91 internal Flash memory is protecting the ZTP Demo application from suspicious network activity.

To understand how ZGATE protects ZTP applications, try the following procedure:

1. Start the ZGATE Demo program.
2. On a PC running a terminal emulation program (such as HyperTerminal), open a web browser such as Internet Explorer or Firefox.

3. In the browser's URL field, enter the IP address of the ZGATE Demo program's Ethernet interface. For example, the IP address of the Ethernet interface displayed in the sample code above is `192.168.2.29`.
4. When the ZGATE Demo program home page appears, click the **TCP Port** link on the left side of the page.
5. The TCP Port page displays TCP numbers which ZGATE is either forwarding to ZTP or blocking from ZTP. When the TCP Port list is operating in WHITELIST FILTERING Mode (default setting), port numbers listed on the page are forwarded to ZTP; port numbers not listed are blocked from ZTP. When the TCP Port list is operating in BLACKLIST FILTERING Mode, port numbers listed on the page are blocked from ZTP; port numbers not listed are forwarded to ZTP. Upon initial observation, note that TCP ports 20 and 21 (used for FTP) are forwarded, meaning that ZGATE will pass any incoming packets destined for those ports to ZTP.
6. With the FTP ports set to Forwarding Mode, open a command prompt on your PC and attempt to establish an FTP session to ZTP. As an example, to establish an FTP connection to IP address 192.168.2.29, enter the following command at the command prompt:

   ```
   ftp 192.168.2.29
   ```

   Next, log in with `anonymous` as the username and password. Performing a `dir` command shows that FTP is working. Enter the `quit` command to terminate the FTP connection to ZTP.
7. Return to your browser and delete the numbers 20 and 21 from the list of TCP port numbers. After both boxes are empty, click the **Update** button to send the changes to ZGATE. After a moment, the web page will be redrawn with the deleted port numbers removed from the list.
8. Switch back to the command prompt and attempt to establish another FTP connection with ZTP, as you did in Step 6. This time, however, this connection will fail because ZGATE is blocking TCP ports 20 and 21 from reaching ZTP.
9. After the FTP connection attempt times out, switch to the ZTP console program and enter the `zg_show stats` command to display the program's current statistics. The output will appear similar to the following example:

   ```
   [ZTP EXTF:/]>zg_show stats

   ZGATE filtering enabled

   ZGATE filtering statistics – packets processed by ZGATE

                        Ethernet   IP      UDP     TCP
   Packets received     0          481     174     15
   ```

```
Packets blocked      0          175     174     3
```

Note: Three TCP packets (i.e., attempts to establish an FTP connection) were blocked because of the change made on the ZGATE TCP Port web page.

10. To reenable FTP, return to your browser and click the **Add** button two times to create two empty boxes at the end of the list. In the first box, enter the number 20; in the second box, enter the number 21. Click the **Update** button.
11. Return to the command prompt on the PC and reattempt to establish an FTP connection with ZTP. This time, the connection should succeed.
12. Return to the browser and delete the number 80 from the list of TCP port numbers, then click the **Update** button. This time, the web page will not refresh because TCP port 80 is now blocked; unless the browser receives information from TCP port 80, it will not be able to refresh the display.
13. To verify that TCP port 80 has been blocked, switch to the ZTP console program and enter the `zg_show tcp` command. The output will appear similar to the following example:

```
ZTP EXTF:/]>zg_show tcp

ZGATE filtering enabled

TCP filtering configuration
TCP port whitelist
{1,7,22,23,25,37,42,43,57,88,107,115,162,179,264,443,546,547,99
2,8081,20,21}

[ZTP EXTF:/]>
```

    As you can see, port 80 was removed from the list of TCP port numbers that ZGATE forwarded to ZTP.

14. To reenable browser access to ZTP, enter the following command on the ZTP console:

```
zg_config add tcp_port 80
```

15. Next, issue the `zg_show tcp` command to verify that TCP port 80 is reenabled.
16. Return to your browser one final time to refresh the web page. The browser should again be able to retrieve pages from ZTP.

Persistent ZGATE Configuration Changes

The ZGATE configuration changes made in the previous section only affect the run-time behavior of ZGATE; these run-time changes are lost each time ZGATE is restarted. To validate this scenario, reload the TCP Port configuration page in the web browser and again disable TCP ports 20 and 21 (FTP), which will prevent the PC from establishing an FTP connection with ZGATE. However, if you restart the ZGATE Demo (by entering the `reboot` command on the ZTP console program), you should discover that the PC will again be able to establish an FTP connection to ZTP.

To modify the boot-time (i.e., persistent) configuration of ZGATE, it is necessary to save the run-time changes to a configuration file named `zg_rules.usr` in the file system. Observe the following procedure to create/update the `zg_rules.usr` file to save ZGATE configuration settings across multiple reboots.

Note: The following procedure will not work properly if the Zilog File System (ZFS) is not enabled in the ZTP project.

1. Restart the ZGATE Demo program.
2. On the PC running the terminal emulation program, open a web browser such as Internet Explorer or Firefox.
3. In the browser's URL field, enter the IP address of the ZGATE Demo program's Ethernet interface.
4. When the ZGATE Demo program home page appears, click the TCP Port link on the left side of the page.
5. Delete port numbers 20 and 21 from the list of TCP Ports, if present, then click the **Update** button.
6. In the list of navigation links on the left of the page, click the **Startup Settings** link.
7. In the first section of the output, the current boot time configuration settings are displayed in blue text, as shown in the Figure 6 example. A full listing of these settings follows this figure.

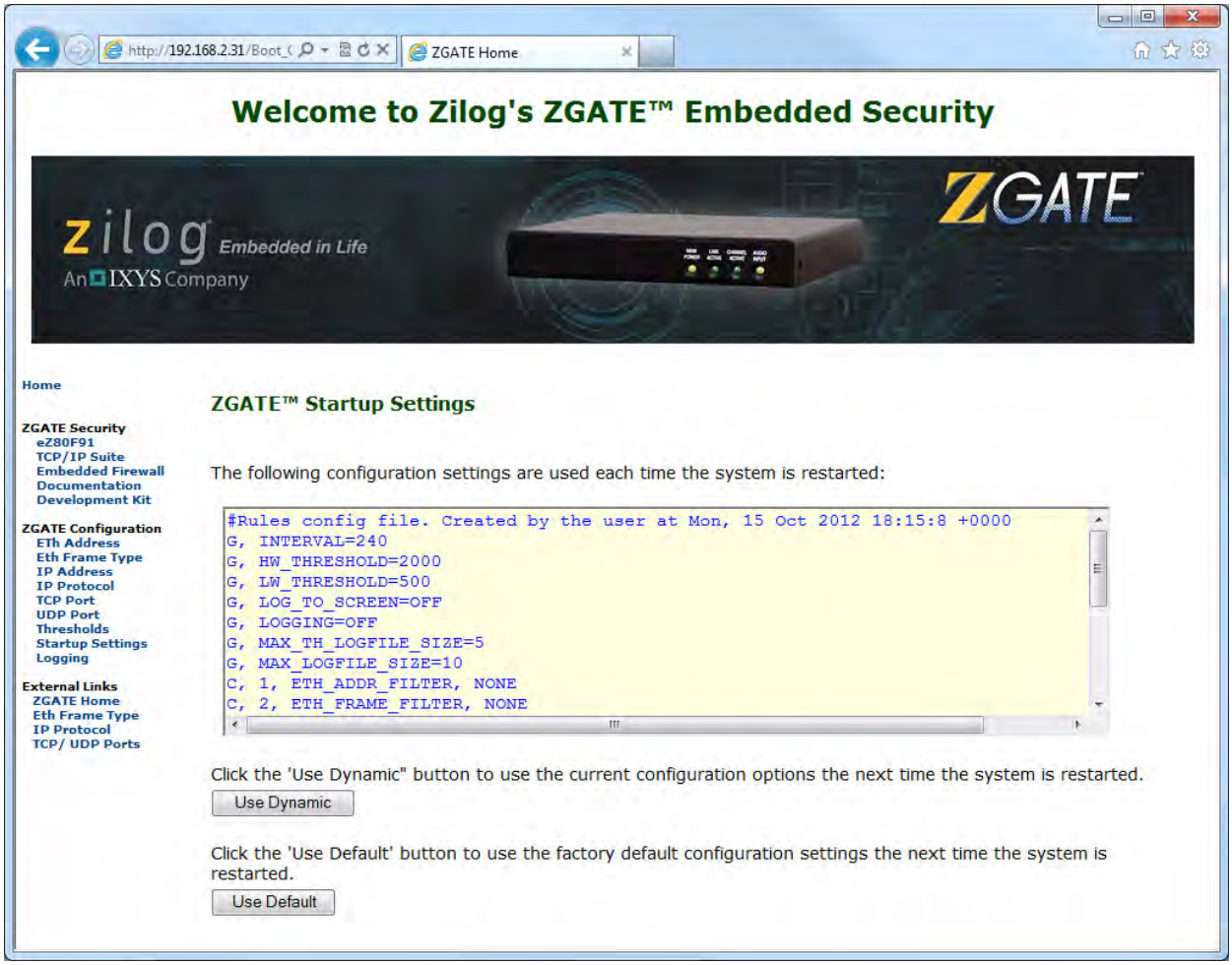


Figure 6. Example Configuration Settings

```
G, INTERVAL=240
G, HW_THRESHOLD=2000
G, LOG_TO_SCREEN=OFF
G, LOGGING=OFF
G, MAX_TH_LOGFILE_SIZE=5
G, MAX_LOGFILE_SIZE=10
C, 1, ETH_ADDR_FILTER, NONE
C, 2, ETH_FRAME_FILTER, NONE
C, 3, IP_SRC_ADDR_FILTER, NONE
C, 4, IP_PROTOCOL_FILTER, WHITELIST
C, 5, TCP_PORT_FILTER, WHITELIST
C, 6, UDP_PORT_FILTER, WHITELIST
```

```
C, 7, ICMP_TYPE_FILTER, NONE
R, 1, WHITELIST, ENABLED, TCP_PORT,
{1,7,20,21,22,23,25,37,42,43,57,80,88,107,115}
R, 2, WHITELIST, ENABLED, TCP_PORT,
{162,179,264,443,546,547,992,8081}
R, 3, WHITELIST, ENABLED, UDP_PORT,
{1,7,22,37,42,53,67,68,69,80,88,123,161,162}
R, 4, WHITELIST, ENABLED, UDP_PORT,
{179,264,514,520,546,547,992}
R, 5, WHITELIST, ENABLED, IP_PROT, {1,2,3,4,6,8,9,17}
```

In the above configuration settings, observe that in the `tcp_port` list, both FTP ports 20 and 21 are listed, even though these ports were removed from the (run-time or dynamic) TCP Port configuration page. This example should explain why FTP access is reenabled each time the system is restarted.

To cause the changes made to the run-time TCP Port configuration to be persistent (i.e., used each time the system is restarted), click the **Use Dynamic** button.

As a result of this procedure, the next time ZGATE restarts, FTP access will not be allowed until it is explicitly added back to the white list, either through the web interface or by using the `zg_config add tcp_port 20 21` shell command.

Altering the ZGATE Static Configuration Settings

Browsing through the Eth Address, Eth Frame Type and IP Address pages of the ZTP Demo program shows that the filtering mode of each of these pages is set to Disabled. As a result, ZGATE will not examine these parameters when determining whether to forward or block packets from ZTP. Furthermore, none of the ZGATE shell commands or web pages can be used to dynamically enable these filtering options at run time. The filtering mode for these parameters is set at the moment the system is started, and can only be modified by either of the following two methods:

- Modify the settings in the `ZGATE_Conf.c` file that is linked to the ZGATE Demo project and rebuild the project. (To learn more, please refer to the Restoring the ZGATE Default Static Configuration section on page 19.)
- Modify the `zg_rules.usr` configuration file resident in the file system using FTP (described below).

Consider a scenario in which it might be necessary to alter ZGATE's persistent configuration settings to prevent untrusted PCs from accessing ZTP. Such a situation could arise if there is a *guest machine* on the local network that should not be allowed to access a ZGATE-protected ZTP device. This situation requires *blacklist filtering*, which causes ZGATE to discard packets that originate from untrusted (blacklisted) sources.

The following procedure provides a sequence for blacklisting specific IP addresses.

1. Run the ZGATE_Demo project. In your browser, navigate to the IP Address configuration page. Note that Filtering Mode is set to Disabled and that the Add, Update, and Cancel buttons are not present.
2. Click the **Startup Settings** link, and cut and paste the information in the Startup section displayed in your browser to a new text file on the PC. Name this file `zg_rules.usr`, and place it into an appropriate folder on your PC.
3. Next, edit this text file so that the IP_SRC_ADDR_FILTER line is modified to:

   ```
   C, 3, IP_SRC_ADDR_FILTER, BLACKLIST
   ```

   Save the file.
4. Switch to a command prompt on the PC and open an FTP session to ZTP. (If TCP Ports 20 and 21 are not enabled in ZGATE, it might be necessary to browse the TCP Port configuration page and add ports 20 and 21).
5. Issue a `put zg_rules.usr` command in the FTP program to transfer the modified configuration file back to ZGATE.
6. In the browser, refresh the Startup Settings page and note that IP_SRC_ADDR_FILTER has been changed to BLACKLIST Mode.
7. Because ZGATE only reads this file during initialization, it is necessary to reboot ZGATE. ZGATE can be rebooted, for example, by entering `reboot` in the ZGATE Demo console.
8. After the ZGATE demo restarts, browse the IP Address page and note that the filtering mode is now BLACKLIST and that the Add, Update, and Cancel buttons are present.
9. At this point, it is necessary to determine the IP address of a machine on the network to blacklist. On a second PC, browse the ZGATE web page to verify that it is initially able to access ZTP. Open a command prompt on this second PC and issue the `ipconfig` command. Record the resulting IP address.
10. Switch to the browser on the first machine and add the IP address you recorded in Step 9 to the list of IP addresses to be blocked by ZGATE.
11. After entering the IP address of the second PC, click the **Update** button.
12. On the second PC, refresh the web page and note that the page fails to load because this PC has now been blacklisted.

Restoring the ZGATE Default Static Configuration

At startup, ZGATE reads its initial configuration from a file in the Zilog File System (ZFS) or from the rules defined in the `ZGATE_Conf.c` file that is linked to the ZTP application.

ZGATE first checks for the presence of the ZGATE user configuration file, `zg_rules.usr`, in ZFS. If this file is present, ZGATE uses the configuration rules in this file. If this file is not present, ZGATE next looks for the ZGATE default configuration file, `zg_rules.def`, in ZFS. If this default file is present, ZGATE uses the configuration rules in this file.

If neither configuration file is present (or if ZFS support is not included in the ZTP application), ZGATE uses the configuration rules defined in the `ZGATE_Conf.c` file that is compiled and linked to the ZTP application. If the `ZGATE_Conf.c` file is modified, it is necessary to rebuild and download the project.

If ZFS is used and the `zg_rules.usr` file becomes unusable, a next-level static configuration file can be recovered. For example, the `zg_rules.usr` file could become unusable if the IP address filter is configured for WHITELIST filtering, but no IP addresses are added to the filtering list. In this instance, ZGATE will block all IP packets from reaching ZTP, effectively disabling all network communication.

This recovery of the next-level static configuration file can be performed using either of the following two methods:

- Using a browser, navigate to the Startup Settings page of the ZGATE Demo program and click the **Use Default** button.
- Issue the `zg_restore` command on the ZTP console.

Either method will erase the `zg_rules.usr` file and cause ZGATE to boot with the settings defined in the `zg_rules.def` ZFS file (if present) or the settings defined in the `ZGATE_Conf.c` file that is linked to the ZGATE Demo program.

Integrating an Existing ZTP Application with ZGATE

ZGATE support can be added to any ZTP application using the following procedure:

1. From the **Start** menu on a Windows PC, click **Computer** and navigate to the **Sample Programs** folder of the ZTP installation. By default, this folder will be located in the following filepath:

   <ZDS Install>\ZTP\ZTP<version>_Lib \ZTP\SamplePrograms

2. Create a new folder in the Sample Program folder by right-clicking in the display window and selecting **New…**, then **Folder**. Enter an appropriate name for the folder; for example, `MyZgateDemo`.
3. Navigate to the ZGATE_Demo folder and copy the following three files to the folder created in Step 2:
   – `ZGATE_Demo_ZGATE000100ZCOG.zdsproj`

- ZGATE000100ZCOG_Flash.ztgt
- ZGATE000100ZCOG_RAM.ztgt

To copy these files, hold down the **Ctrl** key on your keyboard and click each of the three files. After all three files are highlighted, right-click and select **Copy**. Next, navigate to the folder created in Step 2 and, in the whitespace of the folder window, right-click and select **Paste**.

4. Launch ZDSII by navigating from the Windows **Start** menu to **Programs** → **Zilog ZDSII – eZ80Acclaim! <Version>** → **ZDSII – eZ80Acclaim! <Version>**.
5. From the **File** menu in ZDSII, choose **Open Project**, and navigate to the folder created in Step 2.
6. Select the ZGATE_Demo_ZGATE000100ZCOG.zdsproj project and click **Open**. A list of source files will appear in the Workspace panel.
7. In the Workspace panel, remove all of the files listed under Standard Project Files except for the following 18 files:
   - DataPer_Conf.c
   - emac_conf.c
   - ez80eval.c
   - ez80Wh_Conf_ZDS.c
   - F91PhyInit.c
   - get_heap.asm
   - httpAuth_conf.c
   - ppp_conf.c
   - rtc_conf.c
   - RZK_Conf.c
   - shell_conf.c
   - snmp_conf.c
   - tty_conf.c
   - uart_conf.c
   - ZFS_Conf.c
   - ZTPConfig.c
   - ZTPInit_Conf.c
   - ZTPUsrDetails.c

   To remove these files, click each of the files to be removed (one at a time) and press the **Delete** key.
8. In the Workspace panel, remove all of the files listed under Web Files by clicking each of the files to be removed (one at a time) and pressing the Delete key.

9. Copy the source files from the ZTP project folder for which ZGATE protection is desired into the folder that you created in Step 2.
10. In the ZDSII IDE, right-click the Standard Project Files folder in the Workspace panel and select **Add Files To Project...**.
11. Add the source files copied in Step 9 by holding down the **Ctrl** key while clicking each of the files to be added to the project. After all files are highlighted, click the **Add** button.
12. Double-click the source file listed under Standard Project Files in the Workspace panel that contains your application's `ZTPAppEntry()` function. This file will typically be named `main.c`.
13. At the end of the include directive at the start of this file, add the following directive: `#include "ZGATE.H"`.
14. At the end of the `ZTPAppEntry` routine, add the following function call:

    ```
    ZGATE_initialize();
    ```
15. If your application links a custom library (e.g., a modified website library), choose **Settings** from the **Project** menu, navigate to the **Objects and Libraries** option, and click the **Edit** button.
16. From the **Build** menu, choose **Set Active Configuration** to open the Select Configuration dialog box. Select the **RAM** menu option, then click **OK** to close the Select Configuration dialog box.
17. From the **Project** menu in ZDSII, choose **Settings** to open the Project Settings dialog box, and click the Debugger tab.
18. On the Debugger page, select **ZGATE000100ZCOG_RAM** from the **Target** list, and select **USB Smart Cable** from the **Debug Tool** drop-down menu. Click **OK** to close the Project Settings dialog box.
19. If you are prompted to rebuild any affected files, click **Yes**. Otherwise, choose **Build** from the menu bar, then click **Rebuild All**.
20. To run the application, choose **Go** from the **Debug** menu.

The above procedure ensures that all of the ZDSII project settings required for ZGATE support are automatically used in your application. In particular please be aware that ZTP projects that use ZGATE must use the ZDSII target files included in the ZGATE_Demo folder. This target files defines the ZGATE memory layout which must not be altered by your application. To learn more, refer to the chapter titled ZGATE Memory Layout, on page 44.

ZGATE Packet Filtering

ZGATE is designed to protect networked devices from unwanted and potentially malicious packets by filtering incoming packets before they are processed by ZTP. The ZGATE packet filtering criteria is initially determined by the ZTP application developer when the project is created. At the discretion of the application developer, this ZGATE filtering criteria can also be modified at run time by including the sample set of ZGATE shell commands, the sample ZGATE configuration website, or other utilities created by the application developer using the ZGATE API.

ZGATE supports static filtering and Stateful Packet Inspection (SPI) filtering. Threshold filtering is also provided in select ZGATE builds.

ZGATE augments the ZTP stack with a packet filter to control which packets are processed by ZTP. Static filtering blocks packets based on TCP/UDP port number, IP address, IP protocol, Ethernet MAC address or Ethernet frame type.

SPI maintains information about the state of each connection and uses that information to make filtering decisions. This maintenance allows ZGATE to block packets with improper state information (such as TCP SYN flood attacks) and supports dynamic port allocation protocols.

Threshold-based filtering monitors for surges in traffic from a specific IP address and protects against Denial of Service (DoS) attacks and packet floods.

ZGATE can optionally be configured to log events to a file or to the ZTP console. When logging is enabled, a log entry will be created each time a packet is blocked.

Static Filtering

Static filtering functions by examining each packet and determining if the packet should be blocked based on the information in that packet. Static filtering can be based on a variety of criteria including:

Ethernet Address. Blocks packets based on the sender's Ethernet MAC address.

Ethernet Frame Type. Blocks packets based on the Ethernet frame type.

Port Number. Blocks packets based on the target TCP or UDP port number.

IP Address. Blocks packets based on source IP address.

IP Protocol. Blocks packets based on the IP protocol.

ZGATE provides whitelist or blacklist filtering for each static filtering criterion independently. For example, Ethernet MAC addresses can be configured for blacklist filtering

while TCP Port numbers are configured for whitelist filtering. If static filtering of any particular criterion is not required, the filter can be independently disabled.

Static filtering, also called rules-based filtering, uses a filtering engine to evaluate each packet against configured rules or policies. Rules specify the filtering mode (whitelist, blacklist or disabled), the filtering field (IP address, protocol number, port value, etc.), and the values to be matched.

A whitelist is a list of allowed values. If a packet is received and the value is in the list, it is allowed. If not, it is blocked. A blacklist is the opposite: any values on the list are blocked and all other values are allowed.

For example, consider the following rule set:

Rule 1, WHITELIST, IP source address, {201.87.53.10, 207.87.53.11, 201.87.53.12}

Rule 2, WHITELIST, IP protocol, {1,2,6,17}

Rule 3, BLACKLIST, UDP destination port, {600–799}

Rule 4, BLACKLIST, TCP destination port, {600–799}

When a packet is received, the filtering engine first checks Rule 1. If the source IP address is not in a range of 201.87.53.10–201.87.53.12, the firewall blocks the packet. Otherwise, the filtering engine proceeds to the next rule.

The second rule specifies that IP protocols ICMP, IGMP, TCP and UDP (protocol numbers 1, 2, 6 and 17) are allowed. Packets received with any other protocol value are blocked, even those from a whitelisted IP address. The third and fourth rules specify that UDP and TCP ports 600–799 are blacklisted. Therefore, received UDP or TCP packets that target these ports are blocked.

Packets must pass all criteria or they will be blocked from reaching ZTP.

Stateful Packet Inspection

Stateful Packet Inspection (SPI) maintains information about the state of each connection and uses that information to make filtering decisions. For TCP (a connection-oriented protocol), the protocol connection state is used. In contrast, for connectionless protocols such as UDP, the connection state is inferred as either CLOSED or ESTABLISHED based on how recently a packet was sent or received for a given IP address and UDP port. SPI requires a state table which is updated as connections are established, proceeds through the connection states, and is closed. As packets are received, the firewall validates them based on the current state of the connection, then updates the state table. SPI is protocol-specific; therefore the SPI engine must implement a state transition and state validation routine for each supported protocol.

The ZGATE SPI module only supports the TCP and UDP protocols.

Threshold-Based Filtering

Threshold-based filtering functions by keeping statistics about the packets that are received and by monitoring for threshold crossings. When a threshold crossing is detected, ZGATE begins blocking packets. ZGATE extracts a source IP address from each packet and performs threshold-based filtering using this key.

Threshold-based filtering protects against packet floods such as Denial of Service (DoS) attacks, broadcast packet storms, or any other condition that causes a flood of network traffic that can overwhelm a networked device. The filtering key, the high water and low water thresholds, and the interval length are all configurable. If the number of packets received for a given filter key during an interval exceeds the high water threshold, ZGATE will begin dropping packets.

The threshold-based filtering algorithm is a proprietary burst management algorithm that uses statistical information to determine when to enable and disable filtering. A few characteristics of the burst management algorithm are listed below. For these examples, assume that the interval length is 60 seconds and that the high water threshold is 1000. The source IP is the filtering key.

- The algorithm is not completely deterministic and may enable filtering before the exact number of packets is reached. Filtering could be enabled after a single IP address has sent anywhere from 750 to 1000 packets.
- Filtering will always be enabled at between 75% and 100% of the high water threshold value.
- The algorithm enables filtering based on the threshold crossing (packet count) at any time during the interval. Filtering could be enabled during the first second of the interval or the 60th second.
- Filtering is disabled when the packet count remains below the low water threshold for an entire interval.
- Disabling packet filtering due to threshold crossings only occurs when packets are received at a rate lower than the low water threshold for the entire interval. Therefore, if ZGATE threshold filtering is engaged for a particular IP address and the IP address stops sending packets altogether, the threshold filter will not be disengaged. Only when a packet is received will ZGATE determine if filtering should be disengaged (assuming enough time has passed) and print a message to the log file (if logging is enabled).

If IP address W.X.Y.Z floods the ZTP device with packets such that threshold filtering is enabled, a line will be printed to the log file (if enabled), indicating that filtering is enabled for this IP address. If this IP stops sending packets completely, no message will appear in the log file to indicate that filtering has been disabled. Only when another packet is received (and enough time has passed) will a message appear in the log file (if enabled), indicating that filtering has been disabled.

ZGATE Logging

When ZGATE detects an inbound packet that violates one of its filtering rules (static, SPI or threshold) the packet is discarded and, if logging is enabled, information about the breach is logged to the ZTP console and/or to log files in the Zilog File System (ZFS). Logging to the console and to the file system are controlled independently; either or both options may be enabled or disabled at the same time.

Please be aware that depending on the ZGATE configuration settings and the environment in which the ZGATE device operates, logging can generate a lot of information that must be displayed in a human readable format. Consequently, enabling logging (especially logging to the file system) can impact system performance.

Logging to the File System

When logging to the file system is enabled, ZGATE maintains up to six log files (described below); all log files reside in the root folder of the file system. If the ZTP application does not include ZFS support, ZGATE will not be able to log information to the file system.

`zgate.log`	Records information each time an inbound packet is blocked because it violates one of ZGATE's static or SPI filtering rules.
`zgate_lg.old`	Archive of the last `zgate.log` file. When the size of the `zgate.log` file exceeds the archiving threshold, the `zgate_lg.old` file is erased, the current `zgate.log` file is renamed `zgate_lg.old`, and a new `zgate.log` file is created.
`zg_stats.txt`	Contains a running count of the number of packets received/filtered at each of the Ethernet, IP, UDP and TCP layers. This file is regenerated every 1000 packets.

If the ZGATE device includes support for filtering, the following log files are also used:

`zg_thl.log`	A log entry is created when ZGATE's threshold-filtering logic blocks/reenables packet processing from a particular IP address.
`zg_thl.old`	Archive of the last `zg_thl.log` file. When the size of the `zg_thl.log` file exceeds the archiving threshold (the `zg_thl.old` file is erased, the current `zg_thl.log` file is renamed `zg_thl.old` and a new `zg_thl.log` file is created.
`zg_ts.txt`	Contains a running count of the number of packets processed by ZGATE's threshold-filtering module and the number of those packets that were filtered. This file is regenerated every threshold

interval seconds.

The archival threshold setting for the `zgate.log` and `zg_thl.log` file is specified in the ZGATE configuration settings and can be modified at run time using the `zg_logging` shell command, the logging configuration web page, or the `ZGATE_set_max_logfile_size` API function.

Just before writing a new entry to the appropriate log file, ZGATE determines if the size of the current log file exceeds the archiving threshold. If so, ZGATE erases the previous archive file, if present, then changes the name of the current log file to that of the archive.

Note: These archiving thresholds are not absolute maximums. ZGATE only determines the size of the current log file before adding a new entry and will write the entire new entry to the log file if the initial size is below the threshold.

For example, if the archiving threshold of `zgate.log` is 1000 bytes, the current size of the file is 995 bytes, and ZGATE writes a 30-byte entry to the log, the size of the log file will reach 1025 bytes. The next time ZGATE is about to add an entry to the `zgate.log` file, the file will be archived to `zg_lg.old` because its size now exceeds the archiving threshold. The new log entry will then be the first (and initially only item) in the new `zgate.log` file.

ZGATE Configuration

This chapter describes how ZGATE obtains its initial configuration settings, plus the format of these settings.

ZGATE (Persistent) Start-Up Configuration

During system startup, ZGATE obtains its initial filtering configuration from one of the following three files:

- `zg_rules.usr`
- `zg_rules.def`
- `ZGATE_Conf.c`

ZGATE searches for these files in the above top-down sequence, and stops searching after one of these files is found.

The first two files are optional, user-modifiable text files located in the root folder of the Zilog File System (ZFS). ZGATE will first attempt to read the `zg_rules.usr` file; only if that fails will it attempt to read the contents of the `zg_rules.def` file. A ZGATE shell command, web interface or API can be used to automatically generate the `zg_rules.def` file from ZGATE's current dynamic configuration at the time the file is created. After the operator is satisfied with the ZGATE configuration settings in the `zg_rules.usr` file, the operator can manually copy that file to a back-up file named `zg_rules.def`. Should the `zg_rules.usr` file be accidentally deleted, ZGATE will use the backup (`zg_rules.def`) file on the next system reset. At run time, ZGATE can also be used to forcibly delete the `zg_rules.usr` file to force the use of the `zg_rules.def` or `ZGATE_Conf.c` files on the next system restart.

If neither the `zg_rules.usr` or `zg_rules.def` files can be located in the root folder of ZFS (or if ZFS support is not included in the ZTP application), then ZGATE obtains its initial configuration settings from the contents of the `ZGATE_Conf.c` file that is linked to the ZTP application. The `ZGATE.lib` library file contains a default `ZGATE_Conf.c` file which will be linked to the ZTP application regardless of whether the application uses a custom `ZGATE_Conf.c` file. Application developers can copy the `ZGATE_Conf.c` file from its default location (in the `..\ZTP\Conf` installation folder) into the current project folder to modify the settings, as appropriate.

Sample ZGATE Configuration Files

A sample listing of the ZGATE_Conf.c file that is linked to the ZGATE Demo project is shown in the following code example:

```
const char *ZGATE_ConfigStrings[] =
{
   "G, INTERVAL=240",
   "G, HW_THRESHOLD=2000",
   "G, LW_THRESHOLD=499",
   "G, LOG_TO_SCREEN=OFF",
   "G, LOGGING=OFF",
   "G, MAX_TH_LOGFILE_SIZE=5",
   "G, MAX_LOGFILE_SIZE=10",
   "C, 1, ETH_ADDR_FILTER, NONE",
   "C, 2, ETH_FRAME_FILTER, NONE",
   "C, 3, IP_SRC_ADDR_FILTER, NONE",
   "C, 4, IP_PROTOCOL_FILTER, WHITELIST",
   "C, 5, TCP_PORT_FILTER, WHITELIST",
   "C, 6, UDP_PORT_FILTER, WHITELIST",
   "C, 7, ICMP_TYPE_FILTER, NONE",
   "R, 1, WHITELIST, ENABLED, TCP_PORT,
{1,7,20,21,22,23,25,37,42,43,57,80,88,107,115}",
   "R, 2, WHITELIST, ENABLED, TCP_PORT,
{162,179,264,443,546,547,992,8081}",
   "R, 3, WHITELIST, ENABLED, UDP_PORT,
{1,7,22,37,42,53,67,68,69,80,88,123,161,162}",
   "R, 4, WHITELIST, ENABLED, UDP_PORT,
{179,264,514,520,546,547,992}",
   "R, 5, WHITELIST, ENABLED, IP_PROT, {1,2,3,4,6,8,9,17}",
   ""
};
```

When this configuration information is written to a file in ZFS, only the strings enclosed within quotation marks should appear in the file, as shown in the following code example:

```
G, INTERVAL=240
G, HW_THRESHOLD=2000
G, LOG_TO_SCREEN=OFF
G, LOGGING=OFF
G, MAX_TH_LOGFILE_SIZE=5
G, MAX_LOGFILE_SIZE=10
C, 1, ETH_ADDR_FILTER, NONE
C, 2, ETH_FRAME_FILTER, NONE
C, 3, IP_SRC_ADDR_FILTER, NONE
C, 4, IP_PROTOCOL_FILTER, WHITELIST
```

```
C, 5, TCP_PORT_FILTER, WHITELIST
C, 6, UDP_PORT_FILTER, WHITELIST
C, 7, ICMP_TYPE_FILTER, NONE
R, 1, WHITELIST, ENABLED, TCP_PORT,
{1,7,20,21,22,23,25,37,42,43,57,80,88,107,115}
R, 2, WHITELIST, ENABLED, TCP_PORT,
{162,179,264,443,546,547,992,8081}
R, 3, WHITELIST, ENABLED, UDP_PORT,
{1,7,22,37,42,53,67,68,69,80,88,123,161,162}
R, 4, WHITELIST, ENABLED, UDP_PORT, {179,264,514,520,546,547,992}
R, 5, WHITELIST, ENABLED, IP_PROT, {1,2,3,4,6,8,9,17}
```

Format of a ZGATE Configuration File

ZGATE configuration files contain three types of records: global settings, configuration settings, and rules, as described below.

Global Settings

The format of a global setting record is:

`"G" <parameter> "=" <value>`

In the above string, `<parameter>` indicates one of the following parameters:

LOG_TO_SCREEN

LOGGING

INTERVAL

HW_THRESHOLD

LW_THRESHOLD

MAX_TH_LOGFILE_SIZE

MAX_LOGFILE_SIZE

The `<value>` assigned to a global setting depends on the `<parameter>` used. Table 2 indicates the range of values allowed for each of the global settings parameters.

Table 2. Global Settings Allowable Values

Parameter	Permissible Range of Values
LOG_TO_SCREEN	"ON" or "OFF"
LOGGING	"ON" or "OFF"
INTERVAL	Number between 10 and 1800 inclusive.

Table 2. Global Settings Allowable Values

Parameter	Permissible Range of Values
HW_THRESHOLD	Value ≥ 100; must be at least twice the value of LW_THRESHOLD.
LW_THRESHOLD	Value ≥ 20; must be less than half the value of HW_THRESHOLD.
MAX_TH_LOGFILE_SIZE	Number between 10 and 100 inclusive.
MAX_LOGFILE_SIZE	Number between 10 and 100 inclusive.

Note: The global settings for the INTERVAL, HW_THRESHOLD, LW_THRESHOLD and MAX_TH_LOGFILE_SIZE parameters will be ignored by ZGATE devices that do not support threshold filtering.

Static Filter Configuration

The format of a static filtering configuration record is:

`"C" <ConfigNumber> "," <Filter_Type> ","<Filter_Mode>`

In the above string, `<ConfigNumber>` is a monotonically increasing number assigned to the configuration settings by the creator of the file. ZGATE does not use, examine or validate this value.

`<Filter_Type>` must represent one of the following parameters:

ETH_ADDR_FILTER

ETH_FRAME_FILTER

IP_SRC_ADDR_FILTER

IP_PROTOCOL_FILTER

TCP_PORT_FILTER

UDP_PORT_FILTER

Because static filtering is performed on inbound packets, the Ethernet address filter examines the source address of the Ethernet frame. Similarly, the TCP and UDP port number static filters examine the destination port numbers of inbound TCP and UDP packets.

`<Filter_Mode>` must represent one of the following parameters:

NONE

WHITELIST

BLACKLIST

If the filtering mode is set to NONE, ZGATE does not use the corresponding `Filter_Type` parameter when deciding if inbound packets should be filtered. If a particular `Filter_Type` is set to NONE, the ZGATE API, shell and web commands to add/remove entries from the `Filter_Type` static filtering list do nothing.

If `Filter_Mode` is either WHITELIST or BLACKLIST then for each inbound packet ZGATE will extract the `Filter_Type` field from the inbound packet (if applicable) and scan the corresponding `Filter_Type` static filtering list for a matching entry. The packet is then forwarded to ZTP for processing or discarded based on the filtering mode and whether a matching entry was found:

If `Filter_Mode` is BLACKLIST, then ZGATE discards the packet if a matching entry was found; otherwise the packet is routed to ZTP for processing.

If `Filter_Mode` is WHITELIST, then ZGATE only forwards the packet to ZTP if a matching entry was found; otherwise the packet is discarded.

Note: The filtering mode (`Filter_Mode`) of all ZGATE `Filter_Type` filters cannot be changed at run time. There is no ZGATE API, shell command or web interface that will allow the operator to change a filter's filtering mode.

Static Filtering Rules

Static filtering rules are used to populate one of the static filtering lists. The format of a static filtering rules record is:

```
"R" <ConfigNumber> "," <Filter_Mode> "," <State> "," < List_Name> ","<List_Values>
```

In the above string, `<ConfigNumber>` is a monotonically increasing number assigned to the rule by the creator of the file. ZGATE does not use, examine or validate this value.

`<Filter_Mode>` must match the filtering mode specified in the corresponding static filter configuration record.

`<State>` is either ENABLED or DISABLED. ZGATE will only process the filtering rule if `<State>` is set to ENABLED; otherwise ZGATE ignores the rule.

`<List_Name>` must represent one of the following parameters:

ETH_ADDR. Defines static filtering list entries for the ETH_ADDR_FILTER.

ETH_FRAME. Defines static filtering list entries for the ETH_FRAME_FILTER.

IP_ADDR. Defines static filtering list entries for the IP_SRC_ADDR_FILTER.

IP_PROT. Defines static filtering list entries for the IP_PROTOCOL _FILTER.

TCP_PORT. Defines static filtering list entries for the TCP_PORT_FILTER.

UDP_PORT. Defines static filtering list entries for the UDP_PORT _FILTER.

<Value> is a list of one or more comma-separated items within braces. The format and meaning of list items depends on the filtering list to which entries are being added. TCP (and UDP) port number list items must be between 1 and 65535. Ethernet MAC address list items must be six hexadecimal values separated by colons; e.g., `ab:cd:ef:01:23:45`. IP address list items must be entered in dotted decimal format using four numbers between 0 and 255 separated by periods; e.g., `192.168.2.30`. Ethernet frame type list values must be between 1 and 65535; IP protocol number list values must be between 1 and 254.

ZGATE Run-Time Configuration

After system startup, ZGATE's configuration can be viewed/modified using the ZGATE shell commands, the web interface, or programmatically through the ZGATE API.

ZGATE's `zg_save` shell command , the Startup Settings web page, and the `ZGATE_save` API cause ZGATE to save the current ZGATE configuration to the `zg_rules.usr` file in the Zilog File System (only if the ZTP application includes ZFS support). These saved settings will then be used as the ZGATE start-up configuration the next time the system restarts.

The ZGATE `zg_restore` shell command, the Startup Settings web page, and the `ZGATE_restore` API cause ZGATE to delete the `zg_rules.usr` file in the Zilog File System (only if the ZTP application includes ZFS support). As a result, ZGATE will use the start-up settings in the `zg_rules.def` file in ZFS (only if the ZTP application include ZFS support) or the configuration settings in the `ZGATE_Conf.c` file that is linked to the ZTP application as the ZGATE start-up configuration the next time the system restarts.

ZGATE Processing Overview

ZGATE performs filtering at two different layers: the Ethernet packet layer and the IP packet layer. When Ethernet frames are received, ZGATE filters them against the Ethernet frame type and the Ethernet MAC address filtering rules.

At the IP packet layer, ZGATE first checks all packets using the Stateful Packet Inspection (SPI) filtering engine. If the SPI engine determines that a packet is associated with an already-established connection, then no further filtering is performed, and the packet is not blocked by ZGATE. As a result, unnecessary rechecking of the TCP/UDP port number, IP protocol, and IP address, etc., is prevented because these packets were all acceptable at the time the connection was established.

If the SPI layer does not accept or block a packet, ZGATE static filtering is performed. The final step is to perform threshold-based filtering (if it is enabled for the ZGATE product).

Statistics

ZGATE maintains statistics about the number of packets processed and blocked by each filter. These statistics do not necessarily reflect the number of packets received by the ZGATE device, or even by the ZGATE firewall. For example, the number of packets processed by the ZGATE TCP and UDP filters is typically much fewer than the total number of TCP and UDP packets received. Many of the UDP & TCP packets will be associated with an established connection, will therefore pass SPI filtering, and will not require being filtered by the static filtering engine.

Using the ZGATE Web Interface

The ZGATE Demo program includes a sample website that can be used to modify the static filtering lists and the threshold-filtering parameters (only available in the `ZGATE_TIER3` image). This web interface, shown in Figure 7, can also be used to view ZGATE's persistent configuration file and control logging activity.

Every page in the ZGATE sample website includes navigation links along the left side of the page. Click one of the links in the ZGATE Configuration section to view/alter the behavior of ZGATE.

Figure 7. The ZGATE Web Interface

The first six links in the ZGATE Configuration section (highlighted in Figure 7) are used to view/modify one of ZGATE's six static filtering lists: Ethernet MAC address, Ethernet Frame Type, IP Address, IP Protocol, TCP Port Number and UDP Port Number. All of these pages have a similar layout and identical user controls. Figure 8 presents an example of the TCP Port Number Configuration page.

Figure 8. TCP Port Number Configuration Page

At the top of the page, Filtering Mode and Maximum List Size are displayed. The filtering mode will be set to one of the following values: Disabled, Whitelist or Blacklist. If the filtering mode is Disabled, ZGATE does not use the corresponding filtering list when deciding if an inbound packet should be forwarded or discarded. If the filtering mode is Whitelist, ZGATE will route the packet to ZTP if a specific field in that packet matches one of the entries in the corresponding filtering list. For example, in the context of TCP Port numbers, ZGATE will only forward received a TCP packet to ZTP if the destination TCP Port number matches one of the values displayed in the list; otherwise the packet is

discarded. If the filtering mode is Blacklist, ZGATE will route the packet to ZTP if a specific field in that packet does not match any of the entries in the corresponding filtering list. For example, in the context of TCP Port numbers, ZGATE will only forward the received TCP packets to ZTP if the destination TCP Port number does not match any of the values displayed in the list.

It is not possible to change the filtering mode of any of the ZGATE static filtering lists. This setting is determined by the persistent configuration information ZGATE reads during system startup; these settings can be viewed using the Startup Settings navigation link. To learn more, refer to the Altering the ZGATE Static Configuration Settings section on page 18.

Maximum List Size indicates the maximum number of entries that the corresponding filtering list can hold. After the size of a list reaches its maximum, it is no longer possible to add new entries to the list unless one of the existing entries is first removed. If the filtering mode is disabled, the list size will be shown as 0.

The middle of the page displays text that describes how ZGATE uses the filtering list. If the filtering mode is Disabled, nothing else is displayed on the page. Otherwise, another paragraph will be displayed, containing instructions about how to use the Add/Update and Cancel buttons. These buttons are not displayed when the filtering mode is set to Disabled.

Following the text is the filtering list. Each entry in the filtering list (displayed within the input field) is numbered, starting at 1 and incrementing to Maximum List Size. When ZGATE searches the filtering list, it will first look at Item 1 for a match, then Item 2 and so forth, until a match is found or the end of the list is reached.

To add an entry to the list, click the **Add** button. As a result, a new (empty) input field will be displayed at the bottom of the page, as indicated in Figure 9.

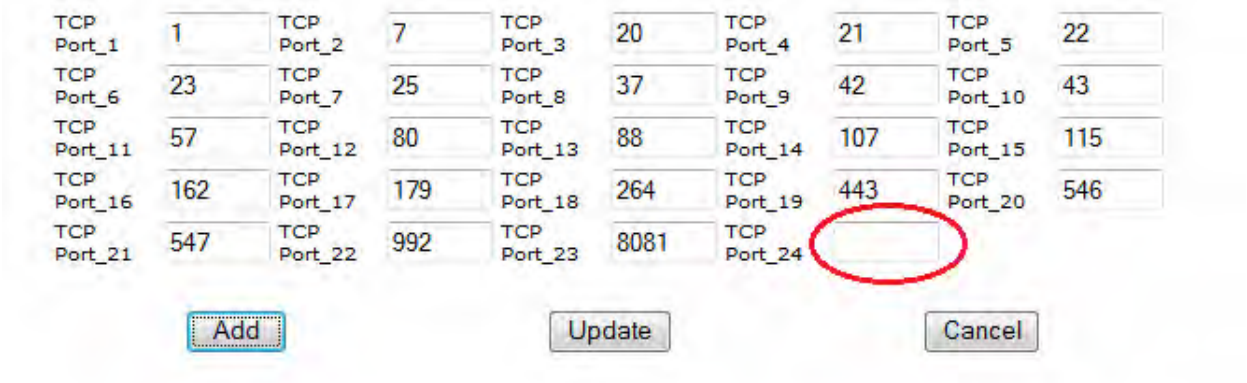


Figure 9. The ZGATE Filtering List

To enter a value in an empty input field, position the cursor anywhere inside the input field and left-click. You may then enter a value for the new filtering list entry. After entering the value to add to the list, click the **Update** button.

To add multiple values, click the **Add** button multiple times. Enter all of the new values to be added, and click the **Update** button. Pressing the Tab key will move between input fields in ascending order, and Shift+Tab will move between input fields in descending order.

Clicking the **Cancel** button causes all changes to the list since the last time the Update button was clicked (or the page was refreshed) to be discarded.

To remove an entry from the filtering list, position the cursor inside the corresponding input field, then press the Backspace key (if the cursor appears behind the displayed value) or the Delete key (if the cursor appears in front of the displayed value). After the input field is empty, click the **Update** button. Multiple entries can be removed from the filtering list by deleting the displayed value in multiple input fields, then clicking the **Update** button.

To change an entry, position the cursor within the input field of the item to be modified, enter the desired value, then click the **Update** button.

When entering TCP (and UDP) port numbers, the value entered must be between 1 and 65535. When entering an Ethernet MAC address, six hexadecimal values must be entered, separated by colons; for example, `ab:cd:ef:01:23:45`. When entering IP addresses, values must be entered in dotted decimal format using four numbers between 0 and 255 and separated by periods; for example, `192.168.2.30`. When entering Ethernet frame types, entered values must be between 1 and 65535. When entering IP Protocol numbers, entered values must be between 1 and 254.

ZGATE Threshold Filtering Configuration Page

At the top of the Threshold Filtering Configuration page, the filtering mode is displayed as either Enabled or Disabled. Threshold filtering is only available in select ZGATE devices. If threshold filtering is not included on your ZGATE device, the filtering mode displayed on this page will appear as Disabled, and nothing else is shown. Otherwise, the Threshold Filtering parameters are displayed.

An example of the ZGATE threshold filtering page is shown in Figure 10.

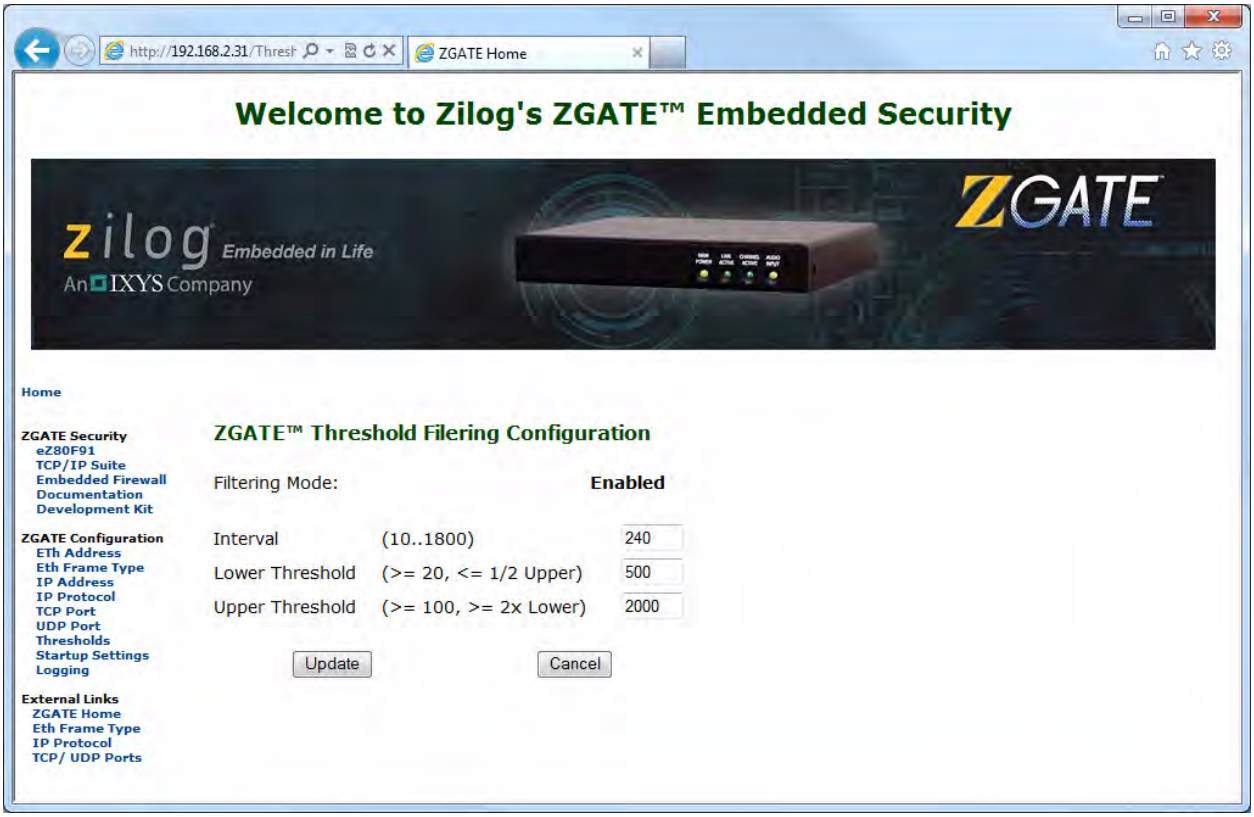


Figure 10. The Threshold Filtering Configuration Page

The Threshold Filtering Configuration page lists the *Interval* parameter, which represents the number of seconds during which ZGATE tracks the number of packets originating from a particular source IP address. The minimum interval is 10 seconds and the maximum interval is 1800 seconds (30 minutes).

Upper Threshold marks the point at which ZGATE will start discarding packets from a particular node. In Figure 10, this Upper Threshold is set to 2000 and the interval is 240 seconds. Therefore, if a node attempts to send ZTP more than 2000 packets within 240 seconds, ZGATE will start filtering packets from that source.

When the upper threshold is crossed, ZGATE will continue to filter packets from the source IP address until the source lowers its transmission rate to less than the lower threshold. In Figure 10, this lower threshold is 500, meaning that after threshold filtering for a particular source IP address has been engaged, it will remain engaged until the source sends ZTP fewer than 500 packets in a 240-second interval.

The Lower and Upper thresholds are related, and values entered must follow the restrictions displayed on the page; i.e., the Lower threshold must contain a value of 20 or greater,

but it must also be less than or equal to half the Upper threshold. The Upper threshold must be a value greater than 100 and also at least twice the size of the lower threshold.

To change one or more of these threshold values, position the cursor within the appropriate input field and enter the desired value. If you prefer, press the Tab key to move between input fields, or reposition the cursor to another input field to alter more than one parameter. After all values have been entered, click the **Update** button.

Clicking the **Cancel** button causes all changes to the threshold filtering parameters since the last time the Update button was clicked (or the page was refreshed) to be discarded.

ZGATE Startup Settings Page

The first section of the ZGATE Startup Settings page displays the contents of the configuration file ZGATE uses to set its initial configuration. The next time ZGATE is powered on, it will again use the displayed configuration settings, effectively ignoring any run-time changes that might have been made on any of the other ZGATE configuration pages or by issuing ZGATE shell commands through the ZTP console. However, the ZGATE Startup Settings page can be used to write the current ZGATE configuration (which reflects changes made on other ZGATE configuration pages) to a file in the Zilog File System (but only if the ZTP project includes ZFS support).

An example of the ZGATE Threshold filtering page is shown in Figure 11.

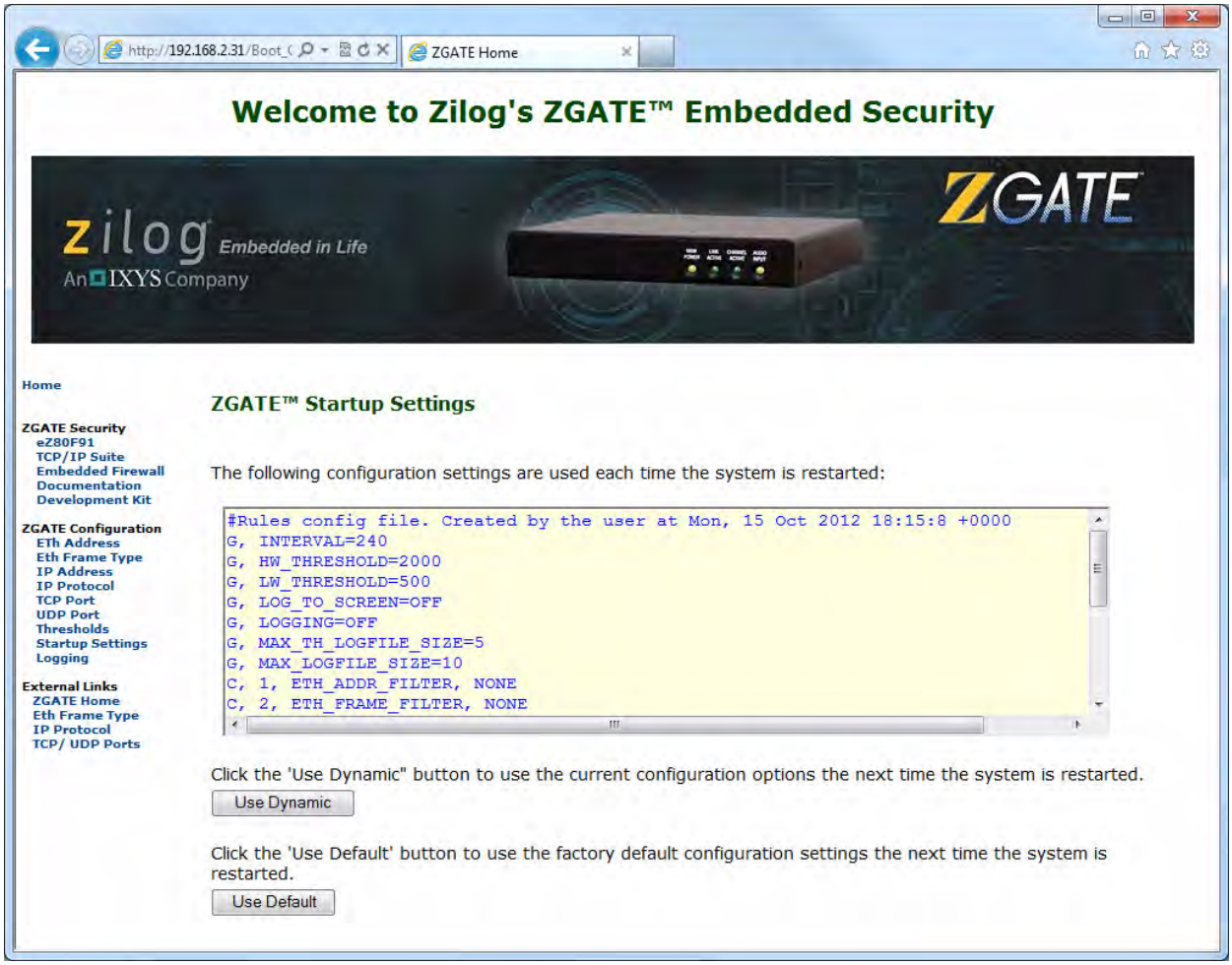


Figure 11. The Startup Settings Page

To make ZGATE use its current settings the next time it is powered on, click the **Use Dynamic** button. As a result, ZGATE will write its current configuration to a file named `zg_rules.usr` in the root of the file system. If this file already exists, it will be overwritten with the current ZGATE configuration settings. If ZFS support is not included in the ZTP application, clicking the Use Dynamic button does nothing.

Clicking the **Use Default** button causes ZGATE to delete the `zg_rules.usr` file from the root of the file system. If ZFS support is not included in the ZTP application, clicking the Use Default button does nothing. Upon deletion of the `zg_rules.usr` file, the initial configuration settings that ZGATE uses the next time it is powered on will depend on whether the operator has manually created a `zg_rules.def` file in the root folder of the file system.

If the root folder of the file system does not contain a file named `zg_rules.usr`, then during system startup, ZGATE will search for a file named `zg_rules.def` in the root folder of the file system. If that file is not found, or if ZFS support is not included in the ZTP application, ZGATE will read its initial configuration settings from the `ZGATE_Conf.c` file that was linked to the ZTP application.

ZGATE Logging Page

The ZGATE Logging Configuration page is used to turn on/off logging to the ZTP console and/or to the Zilog File System (ZFS).

An example of the ZGATE Threshold filtering page is shown in Figure 12.

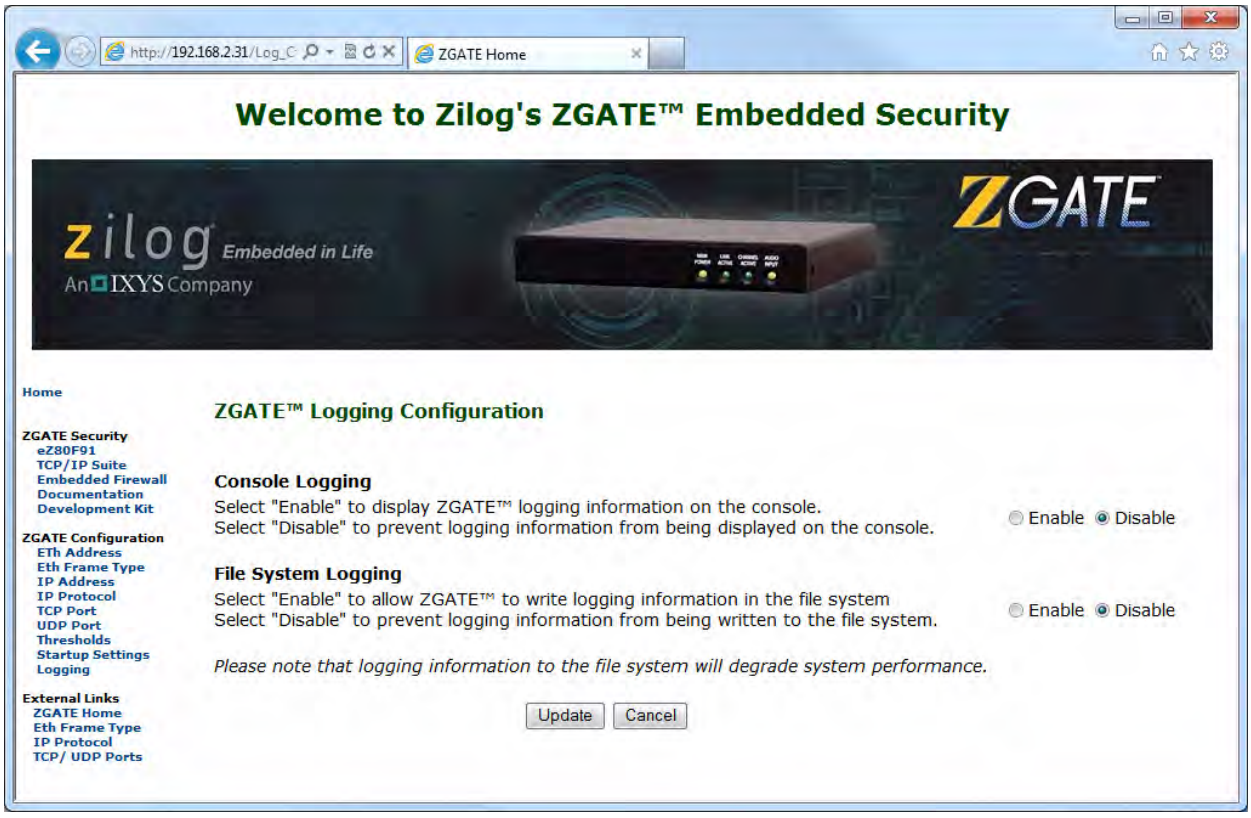


Figure 12. The ZGATE Logging Configuration Page

To enable (or disable) *console logging*, click the **Enable** (or **Disable**) radio button to the right of the Console Logging information, then click the **Update** button. To enable (or dis-

able) *file system logging*, click the **Enable** (or **Disable**) radio button to the right of the File System Logging information, then click the **Update** button.

Upon clicking the **Cancel** button, any changes made to the Console and File System logging configuration since the last time the Update button was clicked – or the Logging Configuration page was regenerated – will be discarded; the current logging settings will remain in effect.

ZGATE Memory Layout

ZGATE imposes the following restrictions on the memory layout of the system:

- ZGATE requires exclusive access to the first 64KB of eZ80F91 internal Flash
  - Customer applications that use internal Flash must use a starting address of `0x010000`
  - eZ80F91 Internal Flash must be enabled and mapped to address `0x000000`
- ZGATE requires exclusive access to all 8KB of eZ80F91 general-purpose RAM
  - eZ80F91 EMAC RAM must be mapped to `0xFFC000`
- External Flash must be connected to CS0
  - A ZTP application targeting a *Flash* or *Copy to RAM* build configuration must use a start address of `0x100000`
- External RAM must be connected to CS1 (if the ZTP application targets a RAM build)
  - A ZTP application using a RAM build configuration must use a starting address of `0x040000`

During system startup, ZGATE will look for ZTP applications to launch at specific addresses in the memory map, and will proceed in the following sequence:

- Look for an application at address `0x040000`. If an application is found at this address, ZGATE will start the ZTP application. If an application is not found, then:
- Look for a ZTP application at address `0x010000`. If an application is found at this address, ZGATE will start the ZTP application. If an application is not found, then:
- Look for a ZTP application at address `0x100000`. If an application is found at this address, ZGATE will start the ZTP application. If an application is not found, then:
- Place the eZ80F91 into Sleep Mode using an eZ80 SLP instruction.

ZGATE applications should use the ZTGT files included in the ZGATE_Demo folder. These target files have been created with the appropriate settings for the ZGATE Embedded Security Development Kit. Similarly, the ZDSII `ZGATE_Demo_ZGATE000100ZCOG.zdsproj` project file contains the appropriate memory settings for the ZGATE Embedded Security Development Kit.

Table 3 defines the range of addresses used in eZ80F91 MCU memory for ZGATE.

Table 3. ZGATE Memory Layout

Address Range	Usage
0x000000–0x00FFFF	Preprogrammed ZGATE image in internal Flash.
0x010000	Start address of ZTP user application targeting eZ80F91 MCU internal Flash.
0x040000	Start address of ZTP user application targeting external RAM (CS1).
0x100000	Start address of ZTP user application targeting external Flash (CS0).
0xFC0000–0xFDFFFF	eZ80F91 EMAC RAM.
0xFFE000–0xFFFFFF	ZGATE run-time RAM (8KB).

ZGATE Shell Command Reference

The ZGATE Demo program includes source code to a set of shell commands that can be used to modify the behavior of ZGATE at run time. These shell commands are available on the ZTP console, which is accessed through a PC terminal emulation program such as Tera Term or HyperTerminal via a serial cable connected to the ZGATE Embedded Security Development Kit. These shell commands are also available through a PC-based Telnet client (provided that ZTP's Telnet server has been enabled). The ZGATE shell commands provide a superset of the functionality of the web interface.

Some of the commands described in this section will not operate if the corresponding ZGATE feature is either disabled or not included in the ZGATE device. For example, if Ethernet address filtering is disabled, the shell command to add an entry to the Ethernet Address filtering list will not function. As another example, because threshold filtering is only available on select ZGATE devices, the shell command to alter the threshold filtering parameters will not function on all ZGATE devices.

The ZGATE shell commands cannot be used to enable a filtering feature that has been disabled in the start-up configuration settings. Similarly, the shell commands cannot be used to change the filtering mode of a filtering list from/to whitelist filtering to/from blacklist filtering.

From within the ZTP console, entering `help` followed by the name of the ZGATE shell command will display a description of the command's function. Entering the name of a command followed by `?` will display the syntax rules for the command.

The ZGATE shell commands implemented in the ZGATE demo program make use of the ZGATE configuration API (see the chapter titled ZGATE API Reference, on page 54 for more information).

zg_show

Syntax

```
zg_show <stats|eth|ip|tcp|udp|th|all>
```

Subcommands

`stats`	Displays statistical information.
`eth`	Displays firewall Ethernet filtering configuration.
`ip`	Displays firewall IP filtering configuration.
`tcp`	Displays firewall TCP filtering configuration.
`udp`	Displays firewall UDP filtering configuration.
`th`	Displays firewall Threshold filtering configuration.
`all`	Displays all firewall filtering configuration information.

Description

The `zg_show` command displays statistical and filtering information.

Example

```
[ZTP EXTF:/]>zg_show ip
ZGATE filtering enabled

IP filtering configuration
IP protocol whitelist
{1,2,3,4,6,8,9,17}
IP address blacklist
{192.168.2.25}

[ZTP EXTF:/]>
```

zg_config

Syntax

```
zg_config < add|remove|set> <parameter> <value(s)>
```

Subcommands

`add`	Adds one or more new values to the static filtering list corresponding to `parameter`. The `add` subcommand will print an error message if the user attempts to add a value that is already on the list.
`remove`	Removes one or more new values to the static filtering list corresponding to `parameter`. The `remove` subcommand will print an error message if the user attempts to remove a value that is not on the list.
`set`	Alters the specified ZGATE threshold filtering parameter.

Description

The `zg_config` command modifies ZGATE filtering parameters.

When the `add` or `remove` subcommands are specified, `parameter` must be one of the following values: `eth_frame`, `eth_addr`, `IP_Addr`, `IP_Prot`, `tcp_port` or `udp_port`.

Value(s) is a list of one or more items separated by spaces; the format and meaning of the value(s) depends on the static filtering list targeted by `parameter`.

`eth_frame:`	Value(s) is an Ethernet frame type. Each item in the value(s) list must be between 1 and 65535.
`eth_addr:`	Value(s) is the Ethernet MAC address of a device that sends packets to ZTP. Each item in the value(s) list must contain six hexadecimal numbers (each between 0 and `0xFF`) separated by colons; for example, "ab:cd:ef:01:23:45".
`IP_Addr:`	Value(s) is the IPv4 address of a device that sends packets to ZTP. Each item in the value(s) list must be in dotted decimal format using 4 numbers (each between 0 and 255) separated by periods; for example, `192.168.2.30`.
`IP_Prot:`	Value(s) is an IP protocol number. Each item in the value(s) list must be between 1 and 254.
`tcp_port:`	Value(s) is a ZTP TCP port number. Each item in the value(s) list must be between 1 and 65535.

`udp_port:` Value(s) is a ZTP TCP port number. Each item in the value(s) list must be between 1 and 65535.

The `set` subcommand is used to adjust one of the threshold filtering parameters, and will not function on ZGATE devices which do not support threshold filtering. When the `set` subcommand is specified, `parameter` must be one of the following values: interval, HW or LW.

Because the `set` subcommand modifies a single threshold-filtering parameter, the value(s) list may only contain a single number. The format and meaning of the value depends on the threshold filtering parameter targeted by the `set` subcommand:

`interval:` Value is the threshold filtering interval expressed in seconds. This value must be between 10 and 1800.

`HW:` Value is the threshold filtering High Water (upper) threshold. This value must be at least 100 and must be at least double the LW threshold.

`LW:` Value is the threshold filtering Low Water (lower) threshold. This value must be at least 20 and not more than half the HW threshold.

Examples

```
[ZTP EXTF:/]>zg_config add eth_frame 2048
[ZTP EXTF:/]>zg_config del eth_addr 00:90:23:0c:f5:e2
[ZTP EXTF:/]>zg_config add IP_Addr 192.168.2.97
[ZTP EXTF:/]>zg_config add IP_Prot 1 2 3 4
[ZTP EXTF:/]>zg_config del tcp_port 20 21
[ZTP EXTF:/]>zg_config add udp_port 137 138

[ZTP EXTF:/]>zg_config set interval 300
[ZTP EXTF:/]>zg_config set lw 500
[ZTP EXTF:/]>zg_config set uw 2000
```

zg_restore

Syntax

```
zg_restore
```

Description

The `zg_restore` command deletes the `zg_rules.usr` file (if present) from the root folder of the Zilog File System (if used). As a result, the default configuration settings in `zg_rules.def` (if present) are restored in the root folder of ZFS (if used) or the compile time configuration settings in the `ZGATE_Conf.c` file that are linked to the ZTP application. ZGATE will use the default configuration the next time it is restarted. The device must be restarted for this change to take effect.

If ZFS is not enabled, this command does nothing.

Example

```
[ZTP EXTF:/]>zg_restore

ZGATE restored to factory default configuration. Reboot now to
begin using the default config.

[ZTP EXTF:/]>
```

zg_save

Syntax

```
zg_save
```

Description

The `zg_save` command saves any configuration changes made using ZGATE shell commands or the web interface to a file named `zg_rules.usr` located in the root folder of the file system (if used). The next time the system boots, it will use the saved setting instead of the default configuration settings in the `zg_rules.def` file (if present) or the default settings in the `ZGATE_Conf.c` file that is linked to the ZTP application.

If ZFS is not enabled, this command does nothing.

Example

```
[ZTP EXTF:/]>zg_save

Configuration changes saved to persistent storage.

[ZTP EXTF:/]>
```

zg_logging

Syntax

```
zg_logging <size 1...1000|th_size 1..1000|screen on/off|file on/
off|show>
```

Subcommands

`size`	Sets the maximum log file size. The value specified must be between 1 and 1000 to set the maximum log file size between 1KB and 1MB.
`th_size`	Sets the maximum threshold log file size. The value specified must be between 1 and 1000 to set the maximum threshold log file size between 1KB and 1MB.
`screen`	Used to turn on/off logging to the ZTP console.
`file`	Used to turn on/off logging to the Zilog File System (ZFS).
`show`	Displays the ZGATE threshold logging configuration.

Description

The `zg_logging` command modifies ZGATE's logging behavior. ZGATE can be configured to log information to the ZTP console (viewable on a PC running a terminal emulation program) and/or to the Zilog File System (only if the ZTP project includes ZFS support).

Note: When logging is enabled (especially if logging to ZFS), system performance will be impacted.

To learn more, refer to the chapter titled ZGATE Logging, on page 26.

Examples

```
[ZTP EXTF:/]>zg_logging size 10
[ZTP EXTF:/]>
[ZTP EXTF:/]>zg_logging th_size 5
[ZTP EXTF:/]>
[ZTP EXTF:/]>zg_logging screen off
[ZTP EXTF:/]>
[ZTP EXTF:/]>zg_logging file on
[ZTP EXTF:/]>
[ZTP EXTF:/]>zg_logging show
```

```
ZGATE Logging Configuration

ZGATE logging config.
Max Logfile size = 10KB  Max Threshold Logfile size = 5KB  Logging
to screen disabled  Logging to file enabled
[ZTP EXTF:/]>
```

ZGATE API Reference

This section describes the ZGATE Application Programming Interface (API). The function prototypes described in this chapter are available in the `ZGATE.h` header file located in the `..\ZTP\Inc` folder.

ZGATE_st_filter_eth

Function Prototype

```
INT8 ZGATE_st_filter_eth( ether * pEthFrameHdr );
```

Parameters

`pEthFrameHdr`	Pointer to the Ethernet frame header.

Return Value

`ZGATE_FORWARD_PACKET`	The frame should be processed.
`ZGATE_DO_NOT_FORWARD_PACKET`	The frame should be dropped.

Description

`ZGATE_st_filter_eth` performs static filtering of Ethernet frames based on the source Ethernet MAC address and Ethernet frame type, if these filters are enabled. This function is the ZGATE entry point for filtering Ethernet frames. It should only ever be called from within the Ethernet MAC driver when an Ethernet frame is received from the network. To learn more, see the `RxFunc` routine in the `RZK\Conf\emac_conf.c` file.

Note: For performance reasons, this function should not be called directly from the Ethernet driver's interrupt service routine. It should be called from the Ethernet driver's RZK Interrupt thread.

ZGATE_initialize

Function Prototype

```
void ZGATE_initialize( void );
```

Parameters

None.

Return Value

None.

Description

This function initializes the ZGATE memory pools and data structures and reads the ZGATE filtering rules. These filtering rules are either read from a file in the file system or placed in a C file that is linked to the ZTP application. ZGATE searches for a configuration file containing the filtering rules in the following sequence:

1. Read the filtering rules from a file named `zg_rules.usr` in the root folder of the file system. If this file is not found – or if ZFS is not enabled – then:
2. Read the filtering rules from a file named `zg_rules.def` in the root folder of the file system. If this file is not found – or if ZFS is not enabled – then:
3. Read the filtering rules from a file named `ZGATE_Conf.c` that is linked to the ZTP application.

ZGATE will not perform any packet filtering until the `ZGATE_initialize` function is called. This API must be called before calling any other ZGATE API.

ZGATE_AddShellCmds

Function Prototype

```
void ZGATE_AddShellCmds( void );
```

Parameters

None.

Return Value

None.

Description

The ZGATE demo program includes an optional set of shell commands described in the chapter titled ZGATE Shell Command Reference, on page 46. The `ZGATE_AddShellCommands` API is called during system initialization to include the ZGATE shell commands in the ZTP application. If this API is not called, none of the shell commands described in the ZGATE Shell Command Reference will be available.

ZGATE_WebInit

Function Prototype

```
void ZGATE_WebInit( void );
```

Parameters

None.

Return Value

None.

Description

The ZGATE demo program includes an optional website to configure ZGATE's filtering behavior through a browser (see the chapter titled Using the ZGATE Web Interface, on page 35). The `ZGATE_WebInit` API is called during system initialization to include the ZGATE website in the ZTP application. Applications that use the ZGATE web interface must not call `http_init` to another website to ZTP.

ZGATE_get_received_stats

Function Prototype

```
void ZGATE_get_received_stats( char * pBuff );
```

Parameters

pBuff	A pointer to a buffer into which received packet statistics will be placed. This buffer should be at least 60 bytes in length.

Return Value

None.

Description

This function generates a NULL-terminated ASCII string containing a count of the packets processed by each of the ZGATE static filtering layers. The string contains the number of packets processed by the following four filters, and in the following sequence: Ethernet filter, IP filter, UDP filter and TCP filter.

Note: These counts are for the static filters only. These counts do not reflect the number of each type of packet received by the device – only those processed by each filter. For example, most TCP packets will be recognized by the SPI filter as part of an existing connection. Because these packets belong to a connection that has already been validated, these packets will not be passed through the TCP static filter, and do not show up in the TCP filter packet count.

ZGATE_get_blocked_stats

Function Prototype

```
void ZGATE_get_blocked_stats( char * pBuff );
```

Parameters

pBuff	A pointer to a buffer into which the received packet statistics will be placed. This buffer should be at least 60 bytes in length.

Return Value

None.

Description

This function generates a NULL-terminated ASCII string containing a count of the packets blocked by each of the ZGATE static filtering layers. The string contains the number of packets blocked by the following four filters, and in the following sequence: Ethernet filter, IP filter, UDP filter and TCP filter.

get_th_config_string

Function Prototype

```
void ZGATE_get_th_config_string(char *buff);
```

Parameters

`pBuff`	A pointer to a buffer into which the threshold configuration information will be placed. This buffer should be at least 110 bytes in length.

Return Value

None.

Description

This function generates a NULL-terminated ASCII string containing the threshold configuration information. The string contains the following parameters, in sequence: threshold interval length, high water threshold and low water threshold.

ZGATE_eth_frame_filtering_type

Function Prototype

```
INT16 ZGATE_eth_frame_filtering_type( void );
```

Parameters

None.

Return Value

`ZGATE_FILTER_DISABLED`	ZGATE does not examine the Type field when deciding whether to forward or discard received Ethernet frames.
`ZGATE_BLACKLIST_FILTERING`	ZGATE discards received Ethernet frames if the Type field matches an entry in the `eth_frame` filtering list.
`ZGATE_WHITELIST_FILTERING`	ZGATE forwards received Ethernet frames if the Type field matches an entry in the `eth_frame` filtering list.

Description

This function returns the filtering mode of the Ethernet frame type filter.

ZGATE_eth_addr_filtering_type

Function Prototype

```
INT16 ZGATE_eth_addr_filtering_type( void );
```

Parameters

None.

Return Value

`ZGATE_FILTER_DISABLED`	ZGATE does not examine the source address field when deciding whether to forward or discard received Ethernet frames.
`ZGATE_BLACKLIST_FILTERING`	ZGATE discards received Ethernet frames if the source address field matches an entry in the `eth_addr` filtering list.
`ZGATE_WHITELIST_FILTERING`	ZGATE forwards received Ethernet frames if the source address field matches an entry in the `eth_addr` filtering list.

Description

This function returns the filtering mode of the Ethernet source address filter.

ZGATE_ip_prot_filtering_type

Function Prototype

```
INT16 ZGATE_ip_prot_filtering_type( void );
```

Parameters

None.

Return Value

`ZGATE_FILTER_DISABLED`	ZGATE does not examine the protocol field when deciding whether to forward or discard received IP packets.
`ZGATE_BLACKLIST_FILTERING`	ZGATE discards received IP packets if the protocol field matches an entry in the `IP_Prot` filtering list.
`ZGATE_WHITELIST_FILTERING`	ZGATE forwards received IP packets if the protocol field matches an entry in the `IP_Prot` filtering list.

Description

This function returns the filtering mode of the IP protocol filter.

ZGATE_ip_addr_filtering_type

Function Prototype

```
INT16 ZGATE_ip_addr_filtering_type( void );
```

Parameters

None.

Return Value

ZGATE_FILTER_DISABLED	ZGATE does not examine the source address field when deciding whether to forward or discard received IP packets.
ZGATE_BLACKLIST_FILTERING	ZGATE discards received IP packets if the source address field matches an entry in the IP_Addr filtering list.
ZGATE_WHITELIST_FILTERING	ZGATE forwards received IP packets if the source address field matches an entry in the IP_Addr filtering list.

Description

This function returns the filtering mode of the IP source address filter.

ZGATE_tcp_port_filtering_type

Function Prototype

```
INT16 ZGATE_tcp_port_filtering_type( void );
```

Parameters

None.

Return Value

`ZGATE_FILTER_DISABLED`	ZGATE does not examine the destination TCP port number when deciding whether to forward or discard received TCP packets.
`ZGATE_BLACKLIST_FILTERING`	ZGATE discards received TCP packets if the destination TCP port number field matches an entry in the `tcp_port` filtering list.
`ZGATE_WHITELIST_FILTERING`	ZGATE forwards received TCP packets if the destination TCP port number field matches an entry in the `tcp_port` filtering list.

Description

This function returns the filtering mode of the TCP destination port number filter.

ZGATE_udp_port_filtering_type

Function Prototype

```
INT16 ZGATE_udp_port_filtering_type( void );
```

Parameters

None.

Return Value

`ZGATE_FILTER_DISABLED`	ZGATE does not examine the destination UDP port number when deciding whether to forward or discard received UDP datagrams.
`ZGATE_BLACKLIST_FILTERING`	ZGATE discards received UDP datagrams if the destination UDP port number field matches an entry in the `udp_port` filtering list.
`ZGATE_WHITELIST_FILTERING`	ZGATE forwards received UDP datagrams if the destination UDP port number field matches an entry in the `udp_port` filtering list.

Description

This function returns the filtering mode of the UDP destination port number filter.

ZGATE_th_filtering_on

Function Prototype

```
INT16 ZGATE_th_filtering_on( void );
```

Parameters

None.

Return Value

TRUE	Threshold filtering is enabled
FALSE	Threshold filtering is disabled.

Description

This function is used to determine if threshold filtering is supported on the ZGATE device.

Threshold filtering is only available on select ZGATE devices. If this API is called on a ZGATE device that does not include threshold filtering support, FALSE will be returned. If the ZGATE device supports threshold filtering, threshold filtering cannot be disabled.

ZGATE_filtering_on

Function Prototype

```
INT16 ZGATE_filtering_on(void);
```

Parameters

None.

Return Value

TRUE	Static and SPI filtering are enabled.
FALSE	Static and SPI filtering is disabled.

Description

This function is used to determine if static and SPI filtering are currently active. If ZGATE successfully initializes, static and SPI filtering will be enabled. It is not possible to disable these filtering mechanisms after ZGATE successfully initializes. The `zg_rules.usr`, `zg_rules.def` and `ZGATE_Conf.c` files can be used to selectively disable static filtering based on individual filtering criteria.

ZGATE_set_th_interval

Function Prototype

```
INT16 ZGATE_set_th_interval( INT32 Interval );
```

Parameters

`Interval`	The threshold filtering interval duration, in seconds. The interval must be at least 10 and no more than 1800 seconds.

Return Value

`TRUE`	The threshold filtering interval was successfully changed.
`FALSE`	The specified interval is invalid.

Description

ZGATE's threshold filtering module constantly measures packet flow over a specific interval of time to determine if the packet flow from a particular source has exceeded filtering thresholds. This API is used to set the ZGATE threshold filtering interval.

ZGATE_set_th_HW

Function Prototype

```
INT16 ZGATE_set_th_HW( INT32 HW_Thresh );
```

Parameters

HW_Thresh	The new threshold filtering high-water (upper) threshold. This high-water threshold must be greater than 100 and must be at least double the low-water threshold.

Return Value

TRUE	The threshold filtering high-water threshold was successfully changed.
FALSE	The specified high-water threshold value is invalid.

Description

After ZGATE detects that a node on the local network is sending packets to ZTP at a rate approaching the high-water threshold, subsequent packets from that source IP address will be blocked until the node's transmission rate falls below the low-water threshold. This API is used to alter the high-water threshold.

ZGATE_set_th_LW

Function Prototype

```
INT16 ZGATE_set_th_LW( INT32 LW_Thresh );
```

Parameters

LW_Thresh	The new threshold filtering low-water (lower) threshold. This low-water threshold must be greater than 20 and must be at no more than half the high-water threshold.

Return Value

TRUE	The threshold filtering low-water threshold was successfully changed.
FALSE	The specified low-water threshold value is invalid.

Description

After ZGATE detects that a node on the local network is sending packets to ZTP at a rate approaching the high-water threshold, subsequent packets from that source IP address will be blocked until the node's transmission rate falls below the low-water threshold. This API is used to alter the low-water threshold.

ZGATE_add_tcp_port

Function Prototype

```
INT16 ZGATE_add_tcp_port( INT32 PortNumber );
```

Parameters

`PortNumber`	TCP port number to be added to the TCP port static filtering list. The value of `PortNumber` must be between 0 and 65535.

Return Value

`TRUE`	`PortNumber` was successfully added to the TCP port static filtering list.
`FALSE`	`PortNumber` is invalid, already exists in the TCP port static filtering list, the list is full, or the TCP Port filter is disabled.

Description

When the TCP Port filter is operating in WHITELIST (forwarding) or BLACKLIST (blocking) Mode, it extracts the destination TCP port number from inbound TCP data segments and compares the target port number to entries in the TCP port list. If a match is found, the packet is forwarded to ZTP if the filter operates in WHITELIST Mode, and blocked from TCP if the filter operates in BLACKLIST Mode.

This API is used to add an entry to the TCP Port static filtering list. Entries cannot be added to the TCP Port list if the TCP Port filter is disabled.

ZGATE_remove_tcp_port

Function Prototype

```
INT16 ZGATE_remove_tcp_port( INT32 value );
```

Parameters

`PortNumber`	The TCP port number to be removed from the TCP port static filtering list. The value of `PortNumber` must be between 0 and 65535.

Return Value

`TRUE`	`PortNumber` was successfully removed from the TCP port static filtering list.
`FALSE`	`PortNumber` is invalid, does not exist in the TCP port static filtering list, or the TCP Port filter is disabled.

Description

When the TCP Port filter is operating in WHITELIST (forwarding) or BLACKLIST (blocking) Mode, it extracts the destination TCP port number from inbound TCP data segments and compares the target port number to entries in the TCP port list. If a match is found, the packet is forwarded to ZTP if the filter operates in WHITELIST Mode and blocked from TCP if the filter operates in BLACKLIST Mode.

This API is used to remove an entry from the TCP Port static filtering list. Entries cannot be removed from the TCP Port list if the TCP Port filter is disabled.

ZGATE_add_udp_port

Function Prototype

```
INT16 ZGATE_add_udp_port( INT32 PortNumber );
```

Parameters

`PortNumber`	UDP port number to be added to the UDP port static filtering list. The value of `PortNumber` must be between 0 and 65535.

Return Value

`TRUE`	`PortNumber` was successfully added to the UDP port static filtering list.
`FALSE`	`PortNumber` is invalid, already exists in the UDP port static filtering list, the list is full, or the UDP Port filter is disabled.

Description

When the UDP Port filter is operating in WHITELIST (forwarding) or BLACKLIST (blocking) Mode, it extracts the destination UDP port number from inbound UDP datagrams and compares the target port number to entries in the UDP port list. If a match is found, the datagram is forwarded to ZTP if the filter operates in WHITELIST Mode and blocked from TCP if the filter operates in BLACKLIST Mode.

This API is used to add an entry to the UP Port static filtering list. Entries cannot be added to the UDP Port list if the UDP Port filter is disabled.

ZGATE_remove_udp_port

Function Prototype

```
INT16 ZGATE_remove_udp_port( INT32 value );
```

Parameters

`PortNumber`	UDP port number to be removed from the UDP port static filtering list. The value of `PortNumber` must be between 0 and 65535.

Return Value

`TRUE`	`PortNumber` was successfully removed from the UDP port static filtering list.
`FALSE`	`PortNumber` is invalid, does not exist in the UDP port static filtering list, or the UDP Port filter is disabled.

Description

When the UDP Port filter is operating in WHITELIST (forwarding) or BLACKLIST (blocking) Mode, it extracts the destination UDP port number from inbound UDP datagrams and compares the target port number to entries in the UDP port list. If a match is found, the datagram is forwarded to ZTP if the filter operates in WHITELIST Mode and blocked from TCP if the filter operates in BLACKLIST Mode.

This API is used to remove an entry from the UP Port static filtering list. Entries cannot be removed from the UDP Port list if the UDP Port filter is disabled.

ZGATE_add_eth_addr

Function Prototype

```
INT16 ZGATE_add_eth_addr( ether_addr_t EthAddr );
```

Parameters

`EthAddr`	MAC Address to be added to the Ethernet address static filtering list. `EthAddr` must be an array of 6 hexadecimal bytes (canonical format) representing the 48-bit IEE Ethernet MAC address of the device to be added to the list.

Return Value

`TRUE`	`EthAddr` was successfully added to the Ethernet address static filtering list.
`FALSE`	`EthAddr` already exists in the Ethernet address static filtering list, the list is full or Ethernet address filtering is disabled.

Description

When the Ethernet Address filter is operating in WHITELIST (forwarding) or BLACK-LIST (blocking) Mode, it extracts the source MAC address from inbound Ethernet frames and compares the address to entries in the Ethernet Address filtering list. If a match is found, the frame is forwarded to ZTP if the filter operates in WHITELIST Mode and blocked from TCP if the filter operates in BLACKLIST Mode.

This API is used to add an entry to the Ethernet Address static filtering list. Entries cannot be added to the list if the Ethernet Address filter is disabled.

ZGATE_remove_eth_addr

Function Prototype

```
INT16 ZGATE_remove_eth_addr( ether_addr_t value );
```

Parameters

`EthAddr`	MAC Address to be added to the Ethernet address static filtering list. `EthAddr` must be an array of 6 hexadecimal bytes (canonical format) representing the 48-bit IEE Ethernet MAC address of the device to be removed from the list.

Return Value

`TRUE`	`EthAddr` was successfully removed from the Ethernet address static filtering list.
`FALSE`	`EthAddr` does not exist in the Ethernet address static filtering list, or Ethernet address filtering is disabled.

Description

When the Ethernet Address filter is operating in WHITELIST (forwarding) or BLACK-LIST (blocking) Mode, it extracts the source MAC address from inbound Ethernet frames and compares the address to entries in the Ethernet Address filtering list. If a match is found, the frame is forwarded to ZTP if the filter operates in WHITELIST Mode and blocked from TCP if the filter operates in BLACKLIST Mode.

This API is used to remove an entry from the Ethernet Address static filtering list. Entries cannot be removed from the list if the Ethernet Address filter is disabled.

ZGATE_add_eth_frame

Function Prototype

```
INT16 ZGATE_add_eth_frame( INT32 FrameType );
```

Parameters

FrameType	Ethernet frame type to be added to the Ethernet frame type static filtering list. The value of FrameType must be between 0 and 65535.

Return Value

TRUE	FrameType was successfully added to the Ethernet frame type static filtering list.
FALSE	FrameType is invalid, already exists in the Ethernet frame type static filtering list, the list is full or the Ethernet frame type filter is disabled.

Description

When the Ethernet Frame Type filter is operating in WHITELIST (forwarding) or BLACKLIST (blocking) Mode, it extracts the Type field from inbound Ethernet frames and compares the frame type to entries in the Ethernet Frame Type filtering list. If a match is found, the frame is forwarded to ZTP if the filter operates in WHITELIST Mode and blocked from TCP if the filter operates in BLACKLIST Mode.

This API is used to add an entry to the Ethernet Frame Type static filtering list. Entries cannot be added to the list if the Ethernet Frame Type filter is disabled.

ZGATE_remove_eth_frame

Function Prototype

```
INT16 ZGATE_remove_eth_frame( INT32 FrameType );
```

Parameters

FrameType	Ethernet frame type to be removed from the Ethernet frame type static filtering list. The value of FrameType must be between 0 and 65535.

Return Value

TRUE	FrameType was successfully removed from the Ethernet frame type static filtering list.
FALSE	FrameType is invalid, does not exist in the Ethernet frame type static filtering list or Ethernet frame type filtering is disabled.

Description

When the Ethernet Frame Type filter is operating in WHITELIST (forwarding) or BLACKLIST (blocking) Mode, it extracts the Type field from inbound Ethernet frames and compares the frame type to entries in the Ethernet Frame Type filtering list. If a match is found, the frame is forwarded to ZTP if the filter operates in WHITELIST Mode and blocked from TCP if the filter operates in BLACKLIST Mode.

This API is used to remove an entry from the Ethernet Frame Type static filtering list. Entries cannot be removed from the list if the Ethernet Frame Type filter is disabled.

ZGATE_add_ip_addr

Function Prototype

```
INT16 ZGATE_add_ip_addr( INT32 IP_Addr );
```

Parameters

`IP_Addr`	The IP address to be added to the IP source address static filtering list. `IP_Addr` must be a 32-bit IPv4 address in little-endian (presentation mode) format (e.g., 192.68.1.15 would be represented by the value `0xC0A8010F` which, on the eZ80F91 MCU, is stored in memory as `0x0F`, `0x01`, `0xA8`, `0xC0`).

Return Value

`TRUE`	`IP_Addr` was successfully added to the IP Address static filtering list.
`FALSE`	`IP_Addr` already exists in the IP Address static filtering list, the list is full or the IP Address filter is disabled.

Description

When the IP Address filter is operating in WHITELIST (forwarding) or BLACKLIST (blocking) Mode, it extracts the source IP address from inbound IP packets and compares the IP address to entries in the IP Address filtering list. If a match is found, the frame is forwarded to ZTP if the filter operates in WHITELIST Mode and blocked from TCP if the filter operates in BLACKLIST Mode.

This API is used to add an entry to the IP Address static filtering list. Entries cannot be added to the list if the IP Address filter is disabled.

ZGATE_remove_ip_addr

Function Prototype

```
INT16 ZGATE_remove_ip_addr ( INT32 IP_Addr );
```

Parameters

`IP_Addr`	IP address to be removed from the IP source address static filtering list. `IP_Addr` must be a 32-bit IPv4 address in little-endian (presentation mode) format (e.g., 192.68.1.15 would be represented by the value `0xC0A8010F` which, on the eZ80F91 MCU, would be stored in memory as `0x0F`, `0x01`, `0xA8`, `0xC0`).

Return Value

`TRUE`	`IP_Addr` was successfully removed from the IP Address static filtering list.
`FALSE`	`IP_Addr` does not exist in the IP Address static filtering list, or the IP Address filter is disabled.

Description

When the IP Address filter is operating in WHITELIST (forwarding) or BLACKLIST (blocking) Mode, it extracts the source IP address from inbound IP packets and compares the IP address to entries in the IP Address filtering list. If a match is found, the frame is forwarded to ZTP if the filter operates in WHITELIST Mode and blocked from TCP if the filter operates in BLACKLIST Mode.

This API is used to remove an entry from the IP Address static filtering list. Entries cannot be removed from the list if the IP Address filter is disabled.

ZGATE_add_ip_prot

Function Prototype

```
INT16 ZGATE_add_ip_prot( INT32 IP_Prot );
```

Parameters

`IP_Prot`	IP protocol number to be added to the IP Protocol number static filtering list. The value of `IP_Prot` must be between 1 and 254.

Return Value

`TRUE`	`IP_Prot` was successfully added to the IP Protocol number static filtering list.
`FALSE`	`IP_Prot` is invalid, already exists in the IP Protocol static filtering list, the list is full or the IP Protocol filter is disabled.

Description

When the IP Protocol filter is operating in WHITELIST (forwarding) or BLACKLIST (blocking) Mode, it extracts the IP protocol field from inbound IP packets and compares it to entries in the IP Protocol number static filtering list. If a match is found, the frame is forwarded to ZTP if the IP Protocol filter operates in WHITELIST Mode and blocked from TCP if the filter operates in BLACKLIST Mode.

This API is used to add an entry to the IP protocol static filtering list. Entries cannot be added to the list if the IP protocol filter is disabled.

ZGATE_remove_ip_prot

Function Prototype

```
INT16 ZGATE_remove_ip_prot( INT32 IP_Prot );
```

Parameters

`IP_Prot`	IP protocol number to be removed from the IP Protocol number static filtering list. The value of `IP_Prot` must be between 1 and 254.

Return Value

`TRUE`	`IP_Prot` was successfully added to the IP Protocol number static filtering list.
`FALSE`	`IP_Prot` is invalid, does not exist in the IP Protocol static filtering list or the IP Protocol filter is disabled.

Description

When the IP Protocol filter is operating in WHITELIST (forwarding) or BLACKLIST (blocking) Mode, it extracts the IP protocol field from inbound IP packets and compares it to entries in the IP Protocol number static filtering list. If a match is found, the frame is forwarded to ZTP if the IP Protocol filter operates in WHITELIST Mode, and blocked from TCP if the filter operates in BLACKLIST Mode.

This API is used to remove an entry from the IP protocol static filtering list. Entries cannot be removed from the list if the IP protocol filter is disabled.

ZGATE_get_list_size

Function Prototype

```
INT16 ZGATE_get_list_size( UINT8 ListId );
```

Parameters

`ListId`	Identifies the target ZGATE static filtering list. Permissible values are: • ZGATE_ETH_FRAME_LIST • ZGATE_ETH_ADDR_LIST • ZGATE_IP_PROT_LIST • ZGATE_IP_ADDR_LIST • ZGATE_TCP_PORT_LIST • ZGATE_UDP_PORT_LIST

Return Value

`-1`	Invalid `ListId`.
`otherwise`	The maximum number of entries the list can hold.

Description

This function returns the maximum size of the specified list.

ZGATE_use_default_config

Function Prototype

```
void ZGATE_use_default_config( void );
```

Parameters

None.

Return Value

None.

Description

This function deletes the user configuration file named `zg_rules.usr` (if present) from the root folder of the file system. If the ZTP application does not include ZFS support, this function does nothing.

The next time the system is powered on, ZGATE will read its initial configuration settings from the `zg_rules.def` file located in the root folder of the file system. If `zg_rules.usr` does not exist or ZFS support is not included in the ZTP application, ZGATE will read its start-up configuration settings from the `ZGATE_Conf.c` file that is linked to the ZTP application.

ZGATE_save_config_changes_to_persistent

Function Prototype

```
void ZGATE_save_config_changes_to_persistent( void );
```

Parameters

None.

Return Value

None.

Description

This function saves the current ZGATE configuration to a file named `zg_rules.usr` in the root folder of the file system. The next time the system is powered on, it will use the configuration settings in the `zg_rules.usr` file instead of the configuration settings in the `zg_rules.def` file (located in the root folder of the files system) or the `ZGATE_Conf.c` file that is linked to the ZTP application.

If this file already exists, the previous contents of the file are overwritten with the new configuration settings. If the ZTP application does not include ZFS support, this function does nothing.

ZGATE_enable_logging_to_screen

Function Prototype

```
void ZGATE_enable_logging_to_screen( void );
```

Parameters

None.

Return Value

None.

Description

This function enables ZGATE logging information to be displayed on the ZTP console. To learn more, refer to the chapter titled ZGATE Logging, on page 26.

ZGATE_disable_logging_to_screen

Function Prototype

```
void ZGATE_disable_logging_to_screen( void );
```

Parameters

None.

Return Value

None.

Description

This function prevents ZGATE logging information from being displayed on the ZTP console.

ZGATE_enable_logging_to_file

Function Prototype

```
void ZGATE_enable_logging_to_file( void );
```

Parameters

None.

Return Value

None.

Description

This function enables ZGATE logging information to be written to log files in the file system. To learn more, refer to the chapter titled ZGATE Logging, on page 26.

ZGATE_disable_logging_to_file

Function Prototype

```
void ZGATE_disable_logging_to_file( void );
```

Parameters

None.

Return Value

None.

Description

This function prevents ZGATE logging information from being written to the file system.

ZGATE_set_max_logfile_size

Function Prototype

```
INT16 ZGATE_set_max_logfile_size( INT32 logfile_size, INT32
th_logfile_size );
```

Parameters

`logfile_size`	The archiving threshold of the `zgate.log` file in units of 1000 bytes.
`th_logfile_size`	The archiving threshold of the `zg_thl.log` file in units of 1000 bytes.

Permissible values for either parameter are:

`-1`	The parameter is ignored. The current archiving threshold is not altered.
`1..1000`	Sets the archiving threshold between 1,000 and 1,000,000 characters.

Return Value

`FALSE`	One or both of the input parameters is invalid.
`TRUE`	Both parameters were accepted.

Description

When ZGATE detects an inbound packet that violates one of its filtering rules (static, SPI or threshold), the packet is discarded and, if file logging is enabled, a new entry is added to the corresponding log file. Static filtering and SPI filtering violations are logged to a file named `zgate.log`. Threshold filtering exceptions are logged to a file named `zg_thl.log`.

All log files (and archives) are maintained in the root folder of the Zilog File System (ZFS). If the ZTP application does not include ZFS support, ZGATE logging information will only be displayed on the ZTP console. In this instance, the `ZGATE_set_max_logfile_size` API serves no useful purpose.

This function sets the maximum size the log files are allowed to reach before being archived. The `zgate.log` file will be archived to `zg_log.old`, and `zg_thl.log` will be archived to `zg_thl.old`. The threshold point for archiving is determined by the value of the `logfile_size` and `th_logfile_size` parameters.

When this API is used to change the archiving threshold of only one of the log files, the value of the opposite argument should be –1. For example, to set the archiving threshold for a threshold filtering log file (`zg_thl.log`) to 2,000 characters, execute the following call:

```
ZGATE_set_max_logfile_size( -1, 2 );
```

To learn more, please refer to the chapter titled ZGATE Logging, on page 26.

ZGATE_get_logging_config

Function Prototype

```
void ZGATE_get_logging_config( char * pBuff );
```

Parameters

`pBuff`	A pointer to a buffer into which the logging information is written. This buffer should be at least 160 bytes in length.

Return Value

None.

Description

This function generates a NULL-terminated ASCII string containing logging configuration information. The string contains, in sequence, the maximum logfile size, the maximum threshold logfile size, logging to screen on/off, and logging to file on/off.

ZGATE_build_UDP_port_list

Function Prototype

```
void ZGATE_build_UDP_port_list( char * pBuff );
```

Parameters

`pBuff`	A pointer to a buffer into which the UDP Port filtering list is written. This buffer should be at least 602 bytes in length.

Return Value

None.

Description

This function generates a NULL-terminated, comma-separated, ASCII string of the UDP Port filtering list. The list is enclosed within braces; for example, `{1,2,3}\0`.

ZGATE_build_TCP_port_list

Function Prototype

```
void ZGATE_build_TCP_port_list( char * pBuff );
```

Parameters

`pBuff`	A pointer to a buffer into which the TCP Port filtering list is written. This buffer should be at least 602 bytes in length.

Return Value

None.

Description

This function generates a NULL-terminated, comma-separated, ASCII string of the TCP Port filtering list. The list is enclosed within braces; for example, `{1,2,3}\0`.

ZGATE_build_ip_addr_list

Function Prototype

```
void ZGATE_build_ip_addr_list( char * pBuff );
```

Parameters

`pBuff`	A pointer to a buffer into which the IP Address filtering list is written. This buffer should be at least 1602 bytes in length.

Return Value

None.

Description

This function generates a NULL-terminated, comma-separated, ASCII string of the IP Address filtering list. The list is enclosed within braces; for example, "{192.168.1.10,192.168.1.11}\0".

ZGATE_build_ip_prot_list

Function Prototype

```
void ZGATE_build_ip_prot_list( char * pBuff );
```

Parameters

`pBuff`	A pointer to a buffer into which the IP Protocol filtering list is written. This buffer should be at least 402 bytes in length.

Return Value

None.

Description

This function generates a NULL-terminated, comma-separated, ASCII string of the IP Protocol filtering list. The list is enclosed within braces; for example, "{1,2,3}\0".

ZGATE_build_eth_addr_list

Function Prototype

```
void ZGATE_build_eth_addr_list( char * pBuff );
```

Parameters

`pBuff`	A pointer to a buffer into which the Ethernet MAC Address filtering list is written. This buffer should be at least 1802 bytes in length.

Return Value

None.

Description

This function generates a NULL-terminated, comma-separated, ASCII string of the Ethernet MAC Address filtering list. The list is enclosed within braces; for example, "{11:22:33:44:55:66,FF:EE:DD:CC:BB:AA:99}\0".

ZGATE_build_eth_frame_list

Function Prototype

```
void ZGATE_build_eth_frame_list( char * pBuff );
```

Parameters

`pBuff`	A pointer to a buffer into which the Ethernet Frame Type filtering list is written. This buffer should be at least 602 bytes in length.

Return Value

None.

Description

This function generates a NULL-terminated, comma-separated, ASCII string of the Ethernet Frame Type filtering list. The list is enclosed within braces; for example, "{2048}\0".

inet_pton

Function Prototype

```
INT8 inet_pton( INT8 af, const char *src, unsigned char *dst );
```

Parameters

`af`	Address family; only AF_INET (1) is supported.
`src`	A pointer to the IP address string in dotted decimal format.
`dst`	A pointer to an INT32 where the network format of the IP address will be stored.

Return Value

`TRUE`	`src` was successfully converted to network format.
`FALSE`	`src` contains an invalid address.

Description

This function converts an IP address in presentation (printable string) format into network format.

ZTP applications can also use the `UINT32 name2ip( char * pAddrString )` API to obtain the 32-bit IPv4 address corresponding to the dotted decimal `(pAddrString` parameter (or domain name).

eth_string_to_num

Function Prototype

```
UINT8 eth_string_to_num( char *buff, ether_addr_t eth_addr );
```

Parameters

`buff`	A pointer to the Ethernet address string.
`ether_addr`	The six-byte array into which the numeric representation of the Ethernet address will be stored.

Return Value

`TRUE`	Ethernet address was successfully converted.
`FALSE`	`buff` contains an invalid Ethernet address string.

Description

This function converts an Ethernet address in presentation (printable string) format into numeric format.

Appendix A. ZGATE Embedded Security Development Board

The purpose of the ZGATE Embedded Security Development Kit is to provide a set of hardware and software tools for applications based on the eZ80F91 microcontroller. An image of the ZGATE Embedded Security Development Board is shown in Figure 13; see Figure 14 for a block diagram.

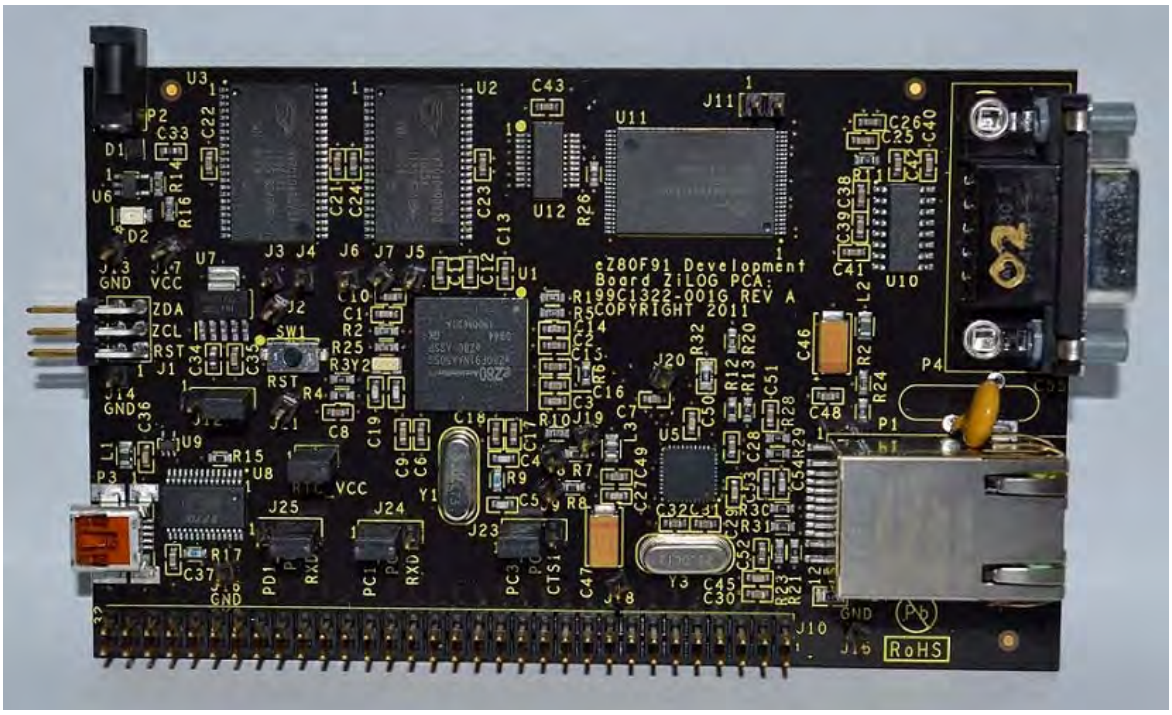


Figure 13. The ZGATE Embedded Security Development Board

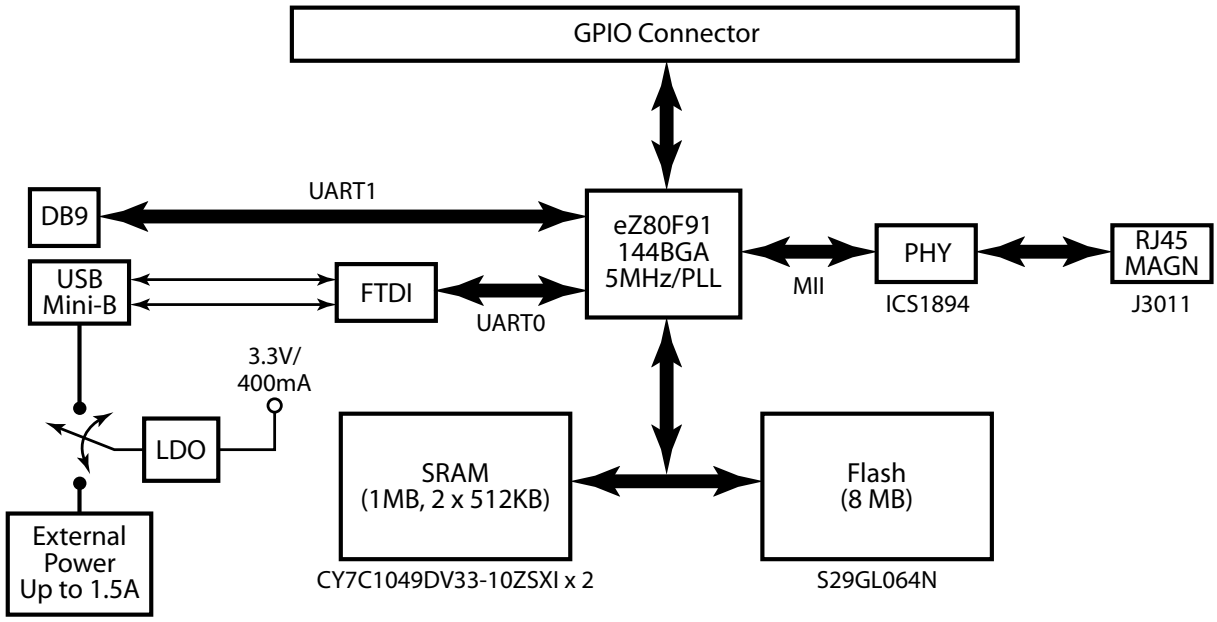


Figure 14. ZGATE Embedded Security Development Kit Block Diagram

Note: Schematic diagrams for the ZGATE Embedded Security Development Board are provided in Appendix B. Schematic Diagrams on page 111.

The ZGATE Embedded Security Development Board is driven by the eZ80F91 MCU with an external 5MHz (Y1) crystal and an on-chip PLL programmed to run on the eZ80F91 MCU at its internal frequency of 50MHz. The chip employs an Ethernet Media Access Controller (EMAC) with a Media-Independent Interface (MII) that allows the eZ80F91 MCU to it to interface to all available industry-standard PHYs.

For this ZGATE Embedded Security implementation, the ICS1894-40 PHY (U5) was selected to provide a suitable price/performance solution.

Debug connector (J1) provides access to the eZ80F91 MCU so that the user can download and control the execution of the sample projects provided with the Kit or his/her own project(s) using the USB Smart Cable included in the Kit.

Memory

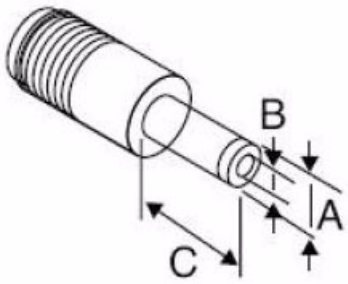
On the ZGATE Embedded Security Development Board, the eZ80F91 MCU features the following memory configurations:

- 8KB on-chip SRAM
- 256KB on-chip Flash
- 1MB off-chip SRAM; U2 and U3 are accessible by chip selects CS1 and CS2
- 8MB off-chip Flash: U11 is accessible by chip select CS0

To learn more about the operation of the eZ80F91 MCU's internal memory, refer to the eZ80F91 ASSP Product Specification (PS0270).

Power Sources

The Board features two power source options. It can be powered by connecting Port P3 (USB Mini-B) to the USB port of the development PC using the USB A to USB mini-B cable included in the Kit. The other option is to connect a female plug to an external 5VDC source with at least 300–400mA of current to Port P2. A drawing of an optional external power supply plug is shown in Figure 15.



Legend
A = 3.5mm
B = 1.3mm
C = 6mm or longer

Figure 15. Female Plug

When the Board is powered via a USB connection, communication with the development PC can be established through the FT232RL chip (U8) to provide a USB-to-serial interface. The UART0 block of the eZ80F91 MCU is connected to this chip.

An on-board USB port (U6) provides overcurrent protection in the event of a short or if a device is connected to the Board that requires more than 500mA @3.3V. If either condition occurs, LED2 will illuminate.

If the Board is powered via a wall outlet and P3 is not used, communication with the PC can be established through a DB9 female connector (P4) which is connected to the UART1 block of the eZ80F91 MCU.

All signals that control access to on-board memory are routed to the Board's test points; please refer to Appendix B. Schematic Diagrams on page 111.

Jumper Settings

All available GPIO ports that exist on the eZ80F91 MCU are routed to connector J10, which is a standard 0.1"-pitch header. These signals are only routed to the odd-numbered pins; all even-numbered pins are connected to GND. Please refer to Appendix B. Schematic Diagrams on page 111 for the exact signal connections to J10. The remaining jumpers are described in Table 4.

Table 4. ZGATE Embedded Security Development Board Jumper Settings

Jumper Name	Description	State	Function	Factory Setting
J11	Flash WP	In	On-board Flash is disabled for writing.	
		Out	On-board Flash is enabled for writing.	Out
J12	Power Source	1–2	USB-powered.	In
		2–3	Wall-powered.	
J26	RTC_V_{CC}	In	RTC is powered by on-board V_{CC}.	In
		Out	External 3.3V source should be connected to J26.2	
J25	RXD0	1–2	Pin L12 of U1 (PD1_RXD0) is connected to J10.61	
		2–3	Pin L12 of U1 (PD1_RXD0) is connected to U8.1 (TXD).	In
J24	RXD1	1–2	Pin G10 of U1 (PC1_RXD1) is connected to J10.45.	In
		2–3	Pin G10 of U1 (PC1_RXD1) is connected to U10.12 (R1OUT).	
J23	CTS1	1–2	Pin F12 of U1 (PC3_CTS1) is connected to J10.41.	
		2–3	Pin F12 of U1 (PC3_CTS1) is connected to U10.9 (R2OUT).	In

Zilog Developer Studio

The Zilog Developer Studio II Integrated Development Environment (ZDSII IDE) is a complete stand-alone system that provides a state-of-the-art development environment. Designed to run on the Windows Vista, Windows 7 and Windows XP Professional operating systems, ZDSII integrates a language-sensitive editor, project manager, C Compiler, assembler, linker, librarian and source-level symbolic debugger that supports code development for the eZ80F91 device. For more information about ZDSII, refer to the Zilog Developer Studio II – eZ80Acclaim! User Manual (UM0144).

ZDSII Flash Loader Utility

The Flash Loader utility integrated within ZDSII allows a convenient way to program on-chip Flash memory. Refer to the Zilog Developer Studio II – eZ80Acclaim! User Manual (UM0144) for more details.

ZDSII Sample Projects

A number of sample projects are included with the ZGATE installation software (for reference, see the Download and Install the Source Code and Documentation section on page 4). These projects, listed in Figures 16 and 17, will be accessible upon installation.

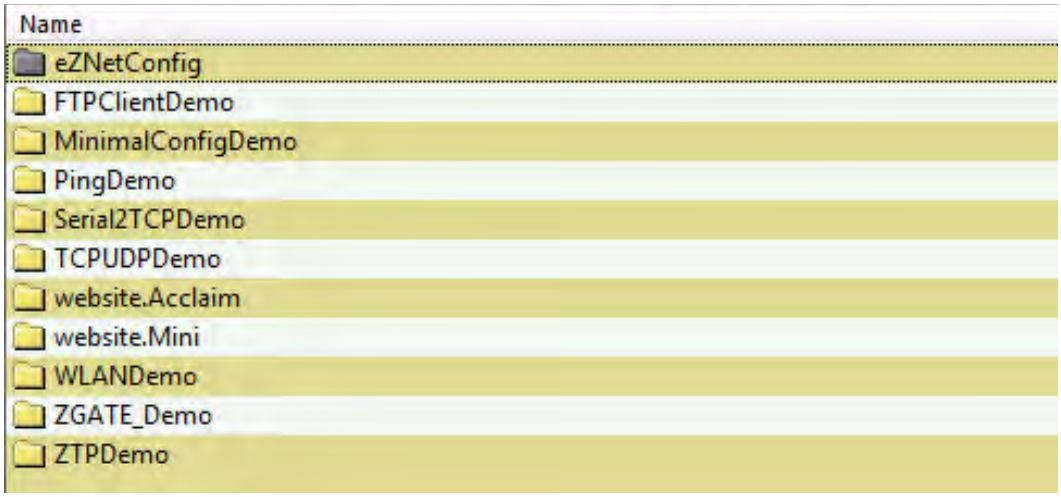


Figure 16. ZTP Sample Projejcts

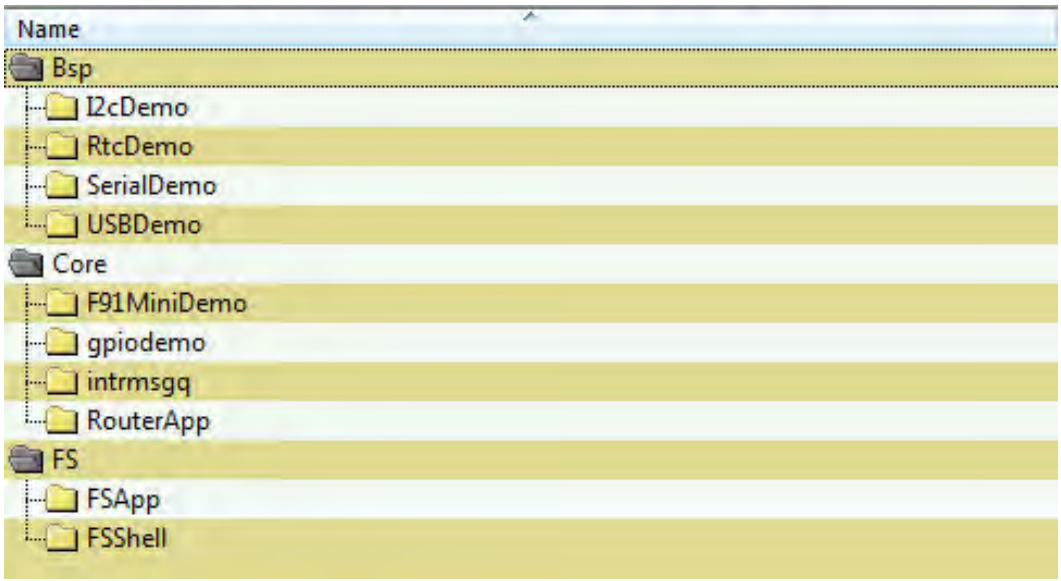


Figure 17. RZK Sample Projejcts

Appendix B. Schematic Diagrams

Figures 18 through 21 display schematic diagrams of the ZGATE Embedded Security Development Board and its interfaces.

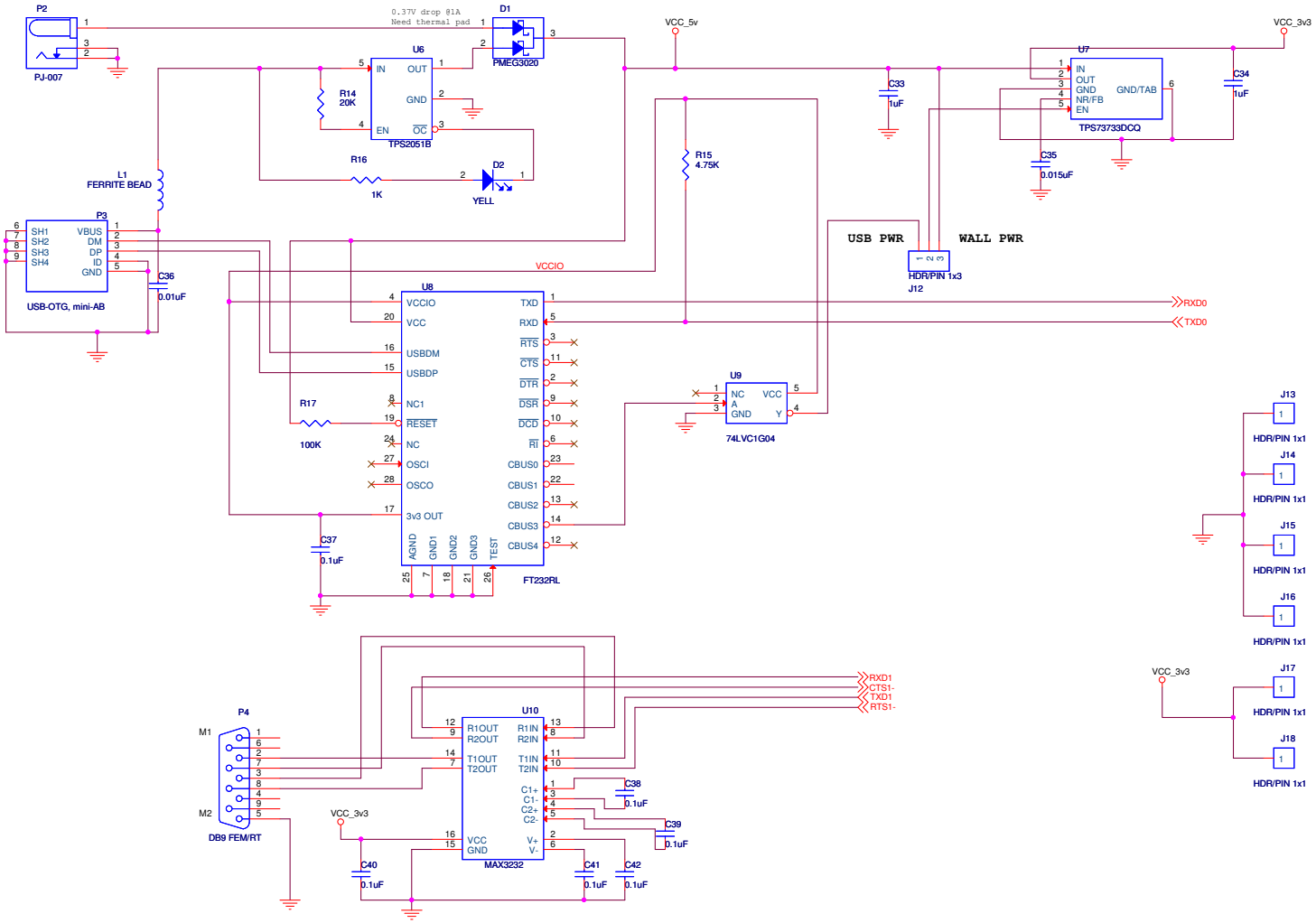


Figure 18. Schematic Diagram #1 of 4: USB and Serial Interfaces

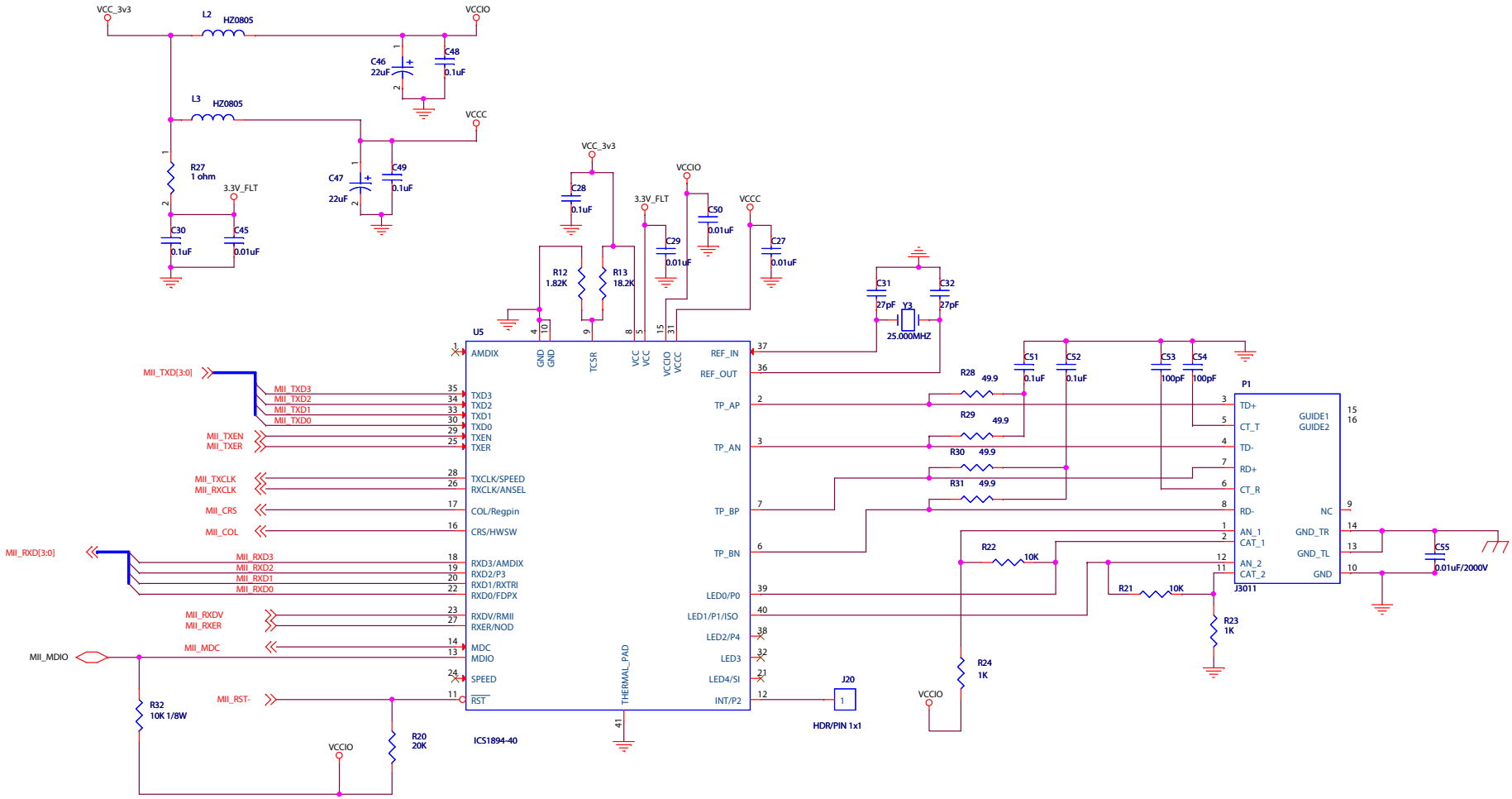


Figure 19. Schematic Diagram #2 of 4: EMAC Interface

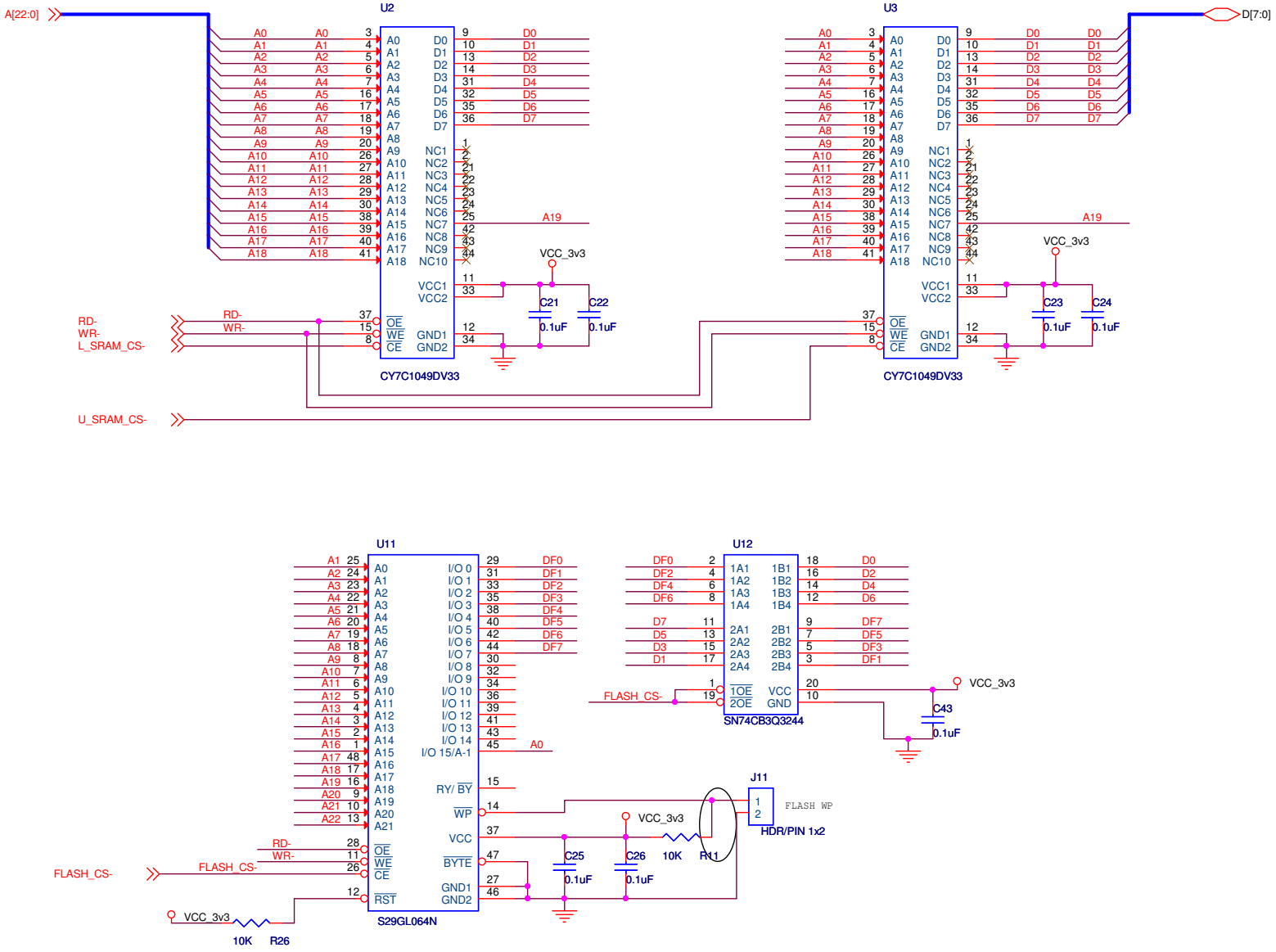


Figure 20. Schematic Diagram #3 of 4: Memory Interface

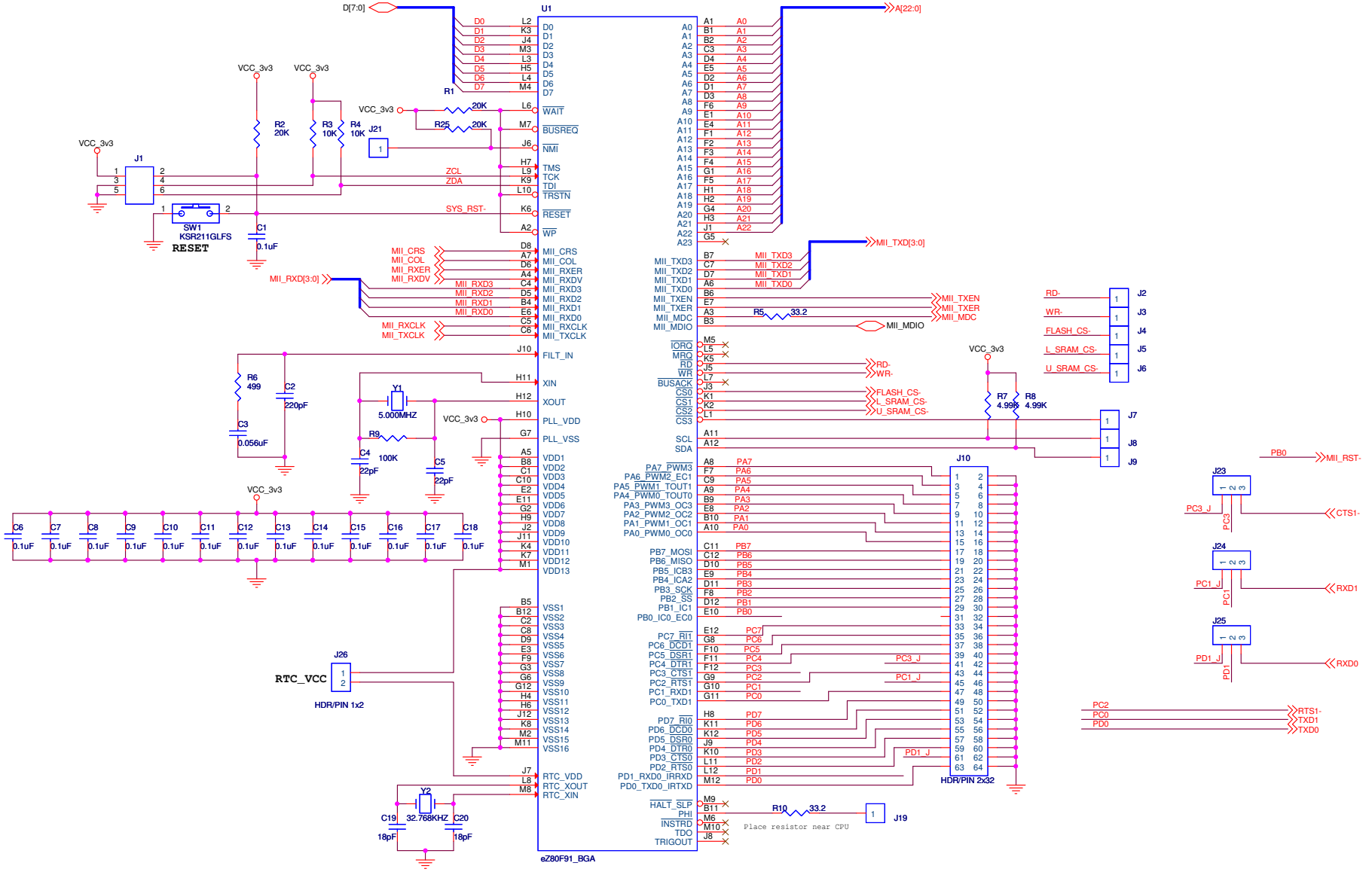


Figure 21. Schematic Diagram #4 of 4: eZ80F91 MCU

Appendix C. Related Documentation

The following documents are associated with the eZ80F91 MCU and are available free for download from the Zilog website.

- Zilog Developer Studio II – eZ80Acclaim! User Manual (UM0144)
- Zilog TCP/IP Software Suite Quick Start Guide (QS0049)
- ZTP Network Security SSL Plug-In Quick Start Guide (QS0059)
- eZ80F91 ASSP Product Specification (PS0270)

Customer Support

To share comments, get your technical questions answered, or report issues you may be experiencing with our products, please visit Zilog's Technical Support page at http://support.zilog.com.

To learn more about this product, find additional documentation, or to discover other facets about Zilog product offerings, please visit the Zilog Knowledge Base or consider participating in the Zilog Forum.

This publication is subject to replacement by a later edition. To determine whether a later edition exists, please visit the Zilog website at http://www.zilog.com.