Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

zilog
Embedded in Life
An IXYS Company

eZ80® Family of Microprocessors

Zilog TCP/IP Software Suite Programmer's Guide

Reference Manual

RM004114-1211

www.zilog.com

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concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded.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.

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
Dec 2011	14	Globally updated for the ZTP v2.4.0 release.	All
Aug 2010	13	Globally updated for the ZTP v2.3.0 release; modified configwlan; added keyIndex and pass-phrase commands.	163, 165, 166
Nov 2008	12	Globally updated for the ZTP v2.2.0 release; added Configuring PPP, 2, scan, join, configwlan, setipparams sections. Updated Build Options for the ZDSII Environment, Common Libraries, Figure 1, 1, User Configuration Details, 3, 4, Configuring the SHELL, Configuring the Management Information Base, Configuring the Simple Network Management Protocol, Connecting to a Remote Host Across a Network, How to Use DNS, How to Use PPP, How to Use SNMP, SNMP Objects, Adding Objects to the MIB, Using SNMP to Manipulate Leaf Objects in the MIB, How to Add a Table to the MIB, The SNMP_GET_FUNC Support Routine, The SNMP_SET_FUNC Support Routine, Updating SNMP Values, 1 sections. Added Appendix B. Guidelines to Porting SNMP and PPP Applications.	3, 4, 7, 14, 15, 16, 24, 28, 29, 31, 34, 58, 66, 69, 77, 81, 84, 87, 89, 92, 94, 98, 103, 105,160, 161, 162,163
Jul 2007	11	Globally updated for branding.	All
Jul 2007	10	Globally updated for the ZTP v2.1.0 release.	All

Date	Revision Level	Description	Page
Jun 2007	09	Updated for style. Added User Configuration Details. Updated SNMP Objects, How to Use SMTP, 1, 8, Configuring PPP, How to Use SNMP, cd, gettime, settime, sleep, Stub Library sections. Removed ZTP Resource Usage, Operating system Overview, Protocol Overview, and ZTP HTTP Server Overview. Removed Build Operations for IAR Embedded Workbench Environment, Understanding SNMP, and Getting started with ZTP sections.	All
Jul 2006	08	Globally updated for the ZTP v2.0.0 release.	All

Table of Contents

Introduction

This reference manual describes the architecture of the Zilog TCP/IP (ZTP) Software Suite, which features a set of TCP/IP software libraries, board support packages (BSPs), application protocols and a version of the Zilog Real-Time Kernel (RZK) for Zilog's eZ80 microprocessors and eZ80Acclaim! microcontrollers. The ZTP libraries require minimum memory and transform these devices into efficient embedded webservers.

Note: This document describes the ZTP Software Suite v2.3.0 and later. If you are using ZTP Software Suite v1.3 or a prior version, refer to the Zilog TCP/IP Software Suite v1.3.4 Programmer's Guide (RM0008), which is available free for download from the Zilog website.

About This Manual

Zilog recommends that you read and understand the complete manual before using this product to develop code. This manual describes how to develop software using the ZTP Software Suite. For additional information regarding the ZTP Software Suite, please refer to the Zilog TCP/IP Stack API Reference Manual (RM0040).

Intended Audience

This document is written for Zilog customers who have exposure to microprocessors and networking fundamentals.

Manual Organization

This reference manual is organized into the following chapters and appendices.

Product Overview

This chapter describes the product overview of ZTP.

ZTP Configuration

This chapter discusses the details about ZTP's configurable parameters, and the build options for ZDS II environment.

Using ZTP

This chapter describes how to use the various protocols available in the ZTP Software Suite.

ZTP Shell Command Reference

This chapter describes the ZTP shell commands.

Appendix A. Creating ZTP Shell Commands

This appendix provides an example of how to create your own shell commands.

Appendix B. Guidelines to Porting SNMP and PPP Applications

This appendix describes the differences between ZTP v2.1.0 (and earlier) and ZTP v2.2.0 and later releases.

Software Release Versions

Software release versions in this manual are represented as `<version>`, which denotes the current release of the ZTP software available on

www.zilog.com. Version numbers are expressed as X.Y.Z, in which X is the major release number; Y is the minor release number, and Z is the revision number.

Safeguards

It is important that you understand the following safety terms.

Caution: A procedure or file can be corrupted if you do not follow directions.

Warning: A procedure can cause injury or death if you do not follow directions.

Online Information

Visit Zilog's eZ80 and eZ80Acclaim! web pages for:

- Product information for eZ80 and eZ80Acclaim! devices
- Downloadable documentation describing the eZ80 and eZ80Acclaim! devices
- Source license information

Product Overview

The Zilog TCP/IP (ZTP) Software Suite includes a preemptive, multitasking real-time kernel, the Zilog Real-Time Kernel (RZK), which is an operating system developed by Zilog. ZTP contains a set of libraries that implement an embedded TCP/IP stack. In addition, ZTP also contains a number of application protocols.

System Features

The key features of ZTP include:

- Compact, preemptive, multitasking real-time kernel with interprocess communications (IPC) support and soft real-time attributes
- Complete TCP/IP stack
- Compatible with all members of the eZ80 family
- Implementation of the following standard network protocols:

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

- Interoperable with all RFC-compliant TCP/IP and Network Protocol implementations to provide seamless connectivity
- A board support package (BSP) containing an Ethernet Media Access Controller (EMAC) driver for the CrystalScan 8900A, the eZ80F91 integrated EMAC, and a WLAN driver for the Realtek 8711 chipset
- A serial driver

- Final stack size is link/time-configurable and determined by the protocols included in the build
- Application demonstrations

ZTP Software

The ZTP software is comprised of two planes.

1. The first plane represents the Zilog's RTOS, RZK; it is referred to as the *OS plane*. The OS plane includes a scheduler, a memory manager, and IPC services.
2. The second plane represents the embedded TCP/IP protocol stack; it is referred to as the *stack plane*. Modules in the stack plane typically require the services of the OS plane to ensure that they can coexist with other applications that compete for the processor.

Figure 1 displays the architecture of the Zilog TCP/IP protocol stack, which corresponds to the Open Systems Interconnect (OSI) model. This figure also displays the locations in which the application can interface to ZTP; these locations are denoted by the color teal.

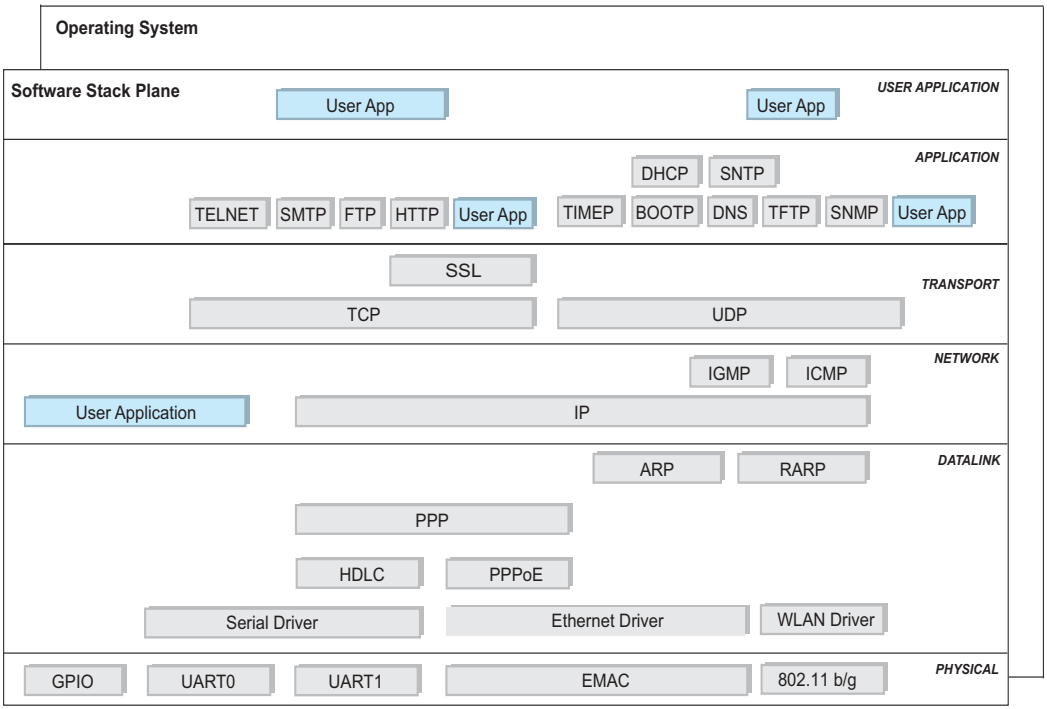

Figure 1. Architecture of the Zilog TCP/IP Protocol Stack

Many TCP/IP protocols are designed to operate on a client-server model. Therefore, Table 1 lists the complete name of each ZTP protocol and also indicates whether ZTP implements a client or a server for each of the application protocols displayed in Figure 1. Protocols that implement the Transport, Network, and Datalink layers typically operate in Peer-to-Peer mode, requiring both a client component and a server component to allow interoperability. These protocols are designated as Peer in Table 1.

Table 1. ZTP Protocol Layers

Protocol	Expansion	Client, Server, or Peer
ARP	Address Resolution Protocol	Peer
DHCP	Dynamic Host Configuration Protocol	Client
DNS	Domain Name Server	Client
FTP	File Transfer Protocol	Client and Server
HTTP	Hyper Text Transfer Protocol	Server
ICMP	Internet Control Message Protocol	Peer
IGMP	Internet Group Management Protocol	Peer
IP	Internet Protocol	Peer
PPP	Point-to-Point Protocol	Peer
RARP	Reverse Address Resolution Protocol	Peer
SMTP	Simple Mail Transfer Protocol	Client
SNMP	Simple Network Management Protocol	Server
SSL	Secure Socket Layer	Server
TCP	Transmission Control Protocol	Peer
Telnet	Telnet	Client and Server
TFTP	Trivial file Transfer Protocol	Client
TIMEP	Time Protocol	Client
UDP	User Datagram Protocol	Peer
SNTP	Simple Network Time Protocol	Client
PPPoE	Point-to-Point Protocol over Ethernet	Client

ZTP Configuration

ZTP is highly configurable and scalable. This chapter discusses the details about its configurable parameters. ZTP configuration is divided into the following three main modules:

1. RZK configuration
2. BSP configuration
3. Network configuration

For more information about RZK and BSP configurations, refer to the Zilog Real-Time Kernel User Manual (UM0075).

Network-Configurable Parameters

This section discusses the configurable parameters of the Datalink Layer, which includes the point-to-point protocol (PPP), the network stack, and the shell.

Datalink Layer Configuration

Table 1 lists a number of device configurations that are used by ZTP in the Datalink Layer. In the table, the default system values are identified by an asterisk. To modify these default values, you must include the corresponding file in the project workspace and modify it according to project requirements.

Table 1. Datalink Layer Configurable Parameters

Component Configuration	File To Modify	Variable/Macro To Modify	Valid Configuration Values
EMAC Driver	RZK\Conf\ emac_conf.c	f91_mac_addr	EMAC address (default values in hexadecimal): 0x00, 0x90, 0x23, 0x00, 0x04, 0x04
		F91_emac_config (valid only for the eZ80F91 platform)	Structure that contains the following values for initializing the EMAC device: txBufSize = 0–1368* mode = F91_10_HD; 10 Mbps Half Duplex mode = F91_10_FD; 10 Mbps Full Duplex mode = F91_100_HD; 100 Mbps Half Duplex mode = F91_100_FD; 100 Mbps Full Duplex mode = F91_AUTO; Autosense bufSize = 0-32*
UART Driver	RZK\Conf\ uart_conf.c	serparams	Structure that contains the following values for initializing the UART device: baud = 2400, 9600 or 19200, 38400, 57600*, or 115200 databits = 7 or 8* stopbits = 1* or 2 parity = PAREVEN, PARODD, or PARNONE*

Note: *Default value: for example, 1368 is default value for upper limit of txBufSize.

Table 1. Datalink Layer Configurable Parameters (Continued)

Component Configuration	File To Modify	Variable/Macro To Modify	Valid Configuration Values
UART Driver	RZK\Conf\uart_conf.c	serparams (cont'd)	Settings can also contain combinational values with logical OR (|)operation: SERSET_DTR_ON* (UART1): Assert data terminal ready (DTR) on open, reset it on close. SERSET_RTSCTS* (UART1): Use RTS/CTS hardware flow control. SERSET_DTRDSR: Use DTR/DSR hardware flow control. SERSET_XONXOFF: Use XON/XOFF SW flow control. SERSET_ONLCR* (UART0): Map NL to CR-NL on output. SERSET_SYNC* (UART0): Use Synchronous routines instead of interrupts. SERSET_IGNHUP* (UART0): Ignore Hangup (CD drop).

Note: *Default value: for example, 1368 is default value for upper limit of txBufSize.

Configuring PPP

The `PPP_CONF.C` file must be configured to enable communication over Point-to-Point (PPP) protocol. PPP supports HDLC and PPPoE as the lower layer interfaces. If the PPPoE macro is defined in the project workspace, then the project will be configured for PPPoE.

Changes to PPP parameters can be made by changing a parameter in the PPP_CONF structure in PPP_CONF.c file.

Note: ZTP supports PPPoE only as a client.

The members of the PPP_CONF structure is explained below.

```
struct PppConf {
   INT8            *myuser;
   INT8            *mypassword;
   UINT16          auth;
   UINT16          MRU;
   UINT16          ConfigTimer;
   UINT16          MaxConfigRequest;
   UINT8           ppp_mode;
};
```

INT8 *myuser. A pointer to a user name string that can be used for authentication. If ZTP performs as a PPP client, then this user name is sent to the peer for authentication and if the ZTP performs as a PPP server, then the user name can be used to authenticate the connecting peer.

INT8 *mypassword. A pointer to a password string that can be used for authentication. If ZTP performs as a PPP client then this is the password that will be sent to the peer for authentication and if ZTP performs as a PPP server, then the password can be used to authenticate the connecting peer.

UINT16 auth. A value that specifies the authentication protocol to use. A value of 0 means that the peer is not authenticated. A value of ZTP_PPP_PAP requires the remote to authenticate using the PAP protocol or a value of ZTP_PPP_CHAP requires the remote to authenticate using CHAP protocol.

UINT16 MRU. The maximum receive unit (MRU) specifies the largest packet size that can be received from the peer.

UINT16 ConfigTimer. The time interval between two PPP configuration request packets.

UINT16 MaxConfigRequest. A value that specifies the maximum number of times a configuration request packet is sent and if either unanswered or rejected before the connection is terminated.

UINT8 ppp_mode. Mode of operation of the PPP layer (`PPP_CLIENT` or `PPP_SERVER`).

The members of the `PppNetworkConf` structure is explained below.

```
struct PppNetworkConf {
   INT8            *myaddress;
   INT8            *peeraddress;
   INT8            *PrimaryDns;
   INT8            *SecondaryDns;
   INT8            *PrimaryNbns;
   INT8            *SecondaryNbns;
};
```

INT8 *myaddress. A string that contains the four-octet IP address that is used for the local end of the connection. The value 0 indicates that the local IP is obtained by negotiation from the other end of the connection.

INT8 *peeraddress. A string that contains the four-octet IP address that is used for the remote end of the connection. The value 0 indicates that the remote IP is obtained by negotiation from the other end of the connection. If a value is specified, the connection is established only if the remote end negotiates the same address.

INT8 *PrimaryDns. A string that contains the four-octet primary DNS server IP address that is used for the remote end of the connection. The value 0 indicates that the DNS server IP is obtained through negotiation from the other end of the connection. If a value is specified, then the DNS server IP address is offered to the peer, else it is not offered.

INT8 *SecondaryDns. A string that contains the four-octet secondary DNS server IP address that is used for the remote end of the connection. If a value is specified, then the secondary DNS server IP address is offered

to the peer. Otherwise, if the value is 0, then the secondary DNS server IP address is not offered to the peer.

INT8 *PrimaryNbns. A string that contains the four-octet primary NBNS server IP address that is used for the remote end of the connection. If a value is specified, then the secondary DNS server IP address is offered to the peer. Otherwise, if the value is 0, then the secondary DNS server IP address is not offered to the peer.

INT8 *SecondaryNbns. A string that contains the four-octet secondary NBNS server IP address that is used for the remote end of the connection. If a value is specified, then the secondary DNS server IP address is offered to the peer. Otherwise, if the value is 0, then the secondary DNS server IP address is not offered to the peer.

The following are the other configuration parameters:

UINT8 g_EnablePppDebug. Setting this variable to TRUE enables the debug prints on the console. The debug prints on the console provides the summary of all of the LCP, PAP/CHAP, IPCP options during PPP negotiations. Setting this variable to FALSE disables all of the debug prints in PPP.

UINT8 g_PppServerAutoInitialize. When a PPP connection is terminated (either by ZTP or by the peer), the ZTP PPP protocol reinitializes and accepts new connection requests from clients, but only if the PPP connection is configured as a server for dial-up or as a direct cable connection, and its variable is set to TRUE. If the variable is set to FALSE, then the application calls the PPP initialization routine (`ztpPPPInit`). If configured as a client, then this variable is not affected.

Configuring PPP with HDLC

Structures of the type `chatscript_t` contain chat scripts (character strings) that are used in exchanges between the modem and the PPP software to perform tasks such as, answering an incoming call (PPP server) or dialing a specific phone number (PPP client). There are four default `chatscript_t` scripts in the `PPP_CONF.c` file that can be used as a

starting point in creating your projects. Table 2 on page 11 lists the default modemchat scripts.

```
typedef struct chatscript {
 INT8            *SendScript;
 INT8            *ReceiveScript;
 UINT16          TimeOutValue;
}chatscript_t;
```

INT8 *SendScript. A pointer to a string that is sent to the modem. NULL is used if the string is not sent.

INT8 *ReceiveScript. A pointer to a string that is expected from the modem; use NULL if no response is expected.

UINT16 TimeOutValue. The maximum number of seconds to wait for an expected string from the external device. After sending a string, the modem control software sets a timer and waits for the expected string. If the expected string arrives before the time-out period, the timer is stopped and the next `modemchat` in the script is executed. However, a time-out occurs before the expected string is received, the PPP layer closes the serial port and abandons this connection attempt. If the time-out is specified as 0, the time-out period is set to an infinite value.

Table 2. Modemchat Scripts and their Description

Modemchat Scripts	Description
`DialServer`	The `DialServer` is used when the ZTP performs as a PPP server answering incoming calls from an external modem.
`DialClient`	The `DialClient` is used when the ZTP performs as a PPP client dialing outgoing calls using an external modem.

Table 2. Modemchat Scripts and their Description

Modemchat Scripts	Description
DccServer	The DccServer is used when the ZTP performs as a PPP server using Direct Cable Connect (DCC, NULL modem) to a client PC with Windows.
DccClient	The DccClient is used when the ZTP performs as a PPP client using Direct Cable Connect (DCC, NULL modem) to a server PC with Windows.

```
typedef struct HdlcConfig {
  INT8 *SerDevName;
  struct chatscript *chat;
  UINT16 nchat;
}HdlcConfig_t;
```

INT8 *SerDevName. Name of the serial device to which the modem is connected (SERIAL0 or SERIAL1).

struct chatscript *chat. Pointer to the chat script used (DialServer/DialClient/DccServer/DccClient).

UINT16 nchat. Number of entries in the chat script pointed by chatscript *chat structure.

Examples of the PPP settings for server and client appear in the code fragments are provided below.

PPP Server Settings

```
struct PppConf PPP_CONF = {
  "ez80",        /* User ID */
  "Zilog123",    /* Password */
  ZTP_PPP_PAP,   /* ZTP_PPP_CHAP,ZTP_PPP_CHAP
Authentication protocol*/
  1400,          /* MRU */
  3,             /* ConfigTimer-Time intervals b/w conf
                 /* requests */
```

```
  6,            /* MaxConfigRequest-Max No. of Conf
                /* requests */
  PPP_SERVER,   /*PPP mode-PPP_SERVER or PPP_CLIENT */
};

struct PppNetworkConf PppNwConf = {
   "192.168.2.1", /* My IP Address */
   "192.168.2.2", /* Peer IP Address */
   "192.168.2.10",/* Peer Primary DNS IP Address */
   0,             /* Peer Secondary DNS IP Address */
   0,             /* Peer Primary NBNS IP Address */
   0              /* Peer Secondary NBNS IP Address */
};
```

PPP Client Settings

```
struct PppConf PPP_CONF = {
  "ez80",       /* User ID */
  "Zilog123",   /* Password */
  ZTP_PPP_PAP,  /* ZTP_PPP_CHAP,ZTP_PPP_CHAP
                /* Authentication protocol*/
  1400,         /* MRU */
  3,            /* ConfigTimer-Time intervals b/w conf
                /* requests */
  6,            /* MaxConfigRequest-Max No. of Conf
                /* requests */
  PPP_CLIENT,   /*PPP mode-PPP_SERVER or PPP_CLIEN */
};
struct PppNetworkConf PppNwConf = {
0,              /* My IP Address */
0,              /* Peer IP Address */
0,              /* Peer Primary DNS IP Address */
0,              /* Peer Secondary DNS IP Address */
0,              /* Peer Primary NBNS IP Address */
0               /* Peer Secondary NBNS IP Address */
};
```

Sample PPP HDLC Server Settings Using An External Modem

```
chatscript_t DialServer[] = {
```

```
  {"ATE&F&K3&S1\r","OK",30},
  {NULL, "RING",0},
  {"ATA\r","CONNECT",60},
  };
HdlcConfig_t g_HdlcConf = {
   "SERIAL1",    /* Serial port on which modem */
   DialServer,
   3
};
```

Configuring PPPoE

The `PPPoE_conf.c` file contains the following variables which can be modified according to the requirements.

UINT8 g_PPPoE_PADI_RexmitCount. Maximum value of the retransmission count for PADI and PADO request, if no response is received.

UINT32 g_PPPoE_PADI_BlockTime. Maximum value of the block time in RZK ticks for a response from the server for PADI and PADO packets sent.

User Configuration Details

ZTP maintains a common list of the user name and password for FTP, SHELL, and Telnet. These details are stored in `ZTPUsrDtls.txt` file in Zilog File System if it is included in the project. If the project does not include ZFS then the user name and password are maintained as a linked list. `ZTPuserDetails.c` file has supporting routines to add/delete the user name and password pairs.

Table 3. ZTP Initialization Routines

File	Routine	Description
ZTPinit_Conf.c	RZK_KernelInit()	Initializes the RZK Kernel and all of the other objects used.
	Init_DataPersistence()	Initializes the data persistence structures and sets the values of the MAC addresses, IP addresses, gateway addresses, subnet mask, and DHCP enable/disable based on the values stored in Flash Information page.
	Init_Serial0_Device()	Adds the UART0 device to the RZK device driver frame work (DDF) and also execute the corresponding initialization routine.
	Init_Serial1_Device()	Adds the UART1 device to the RZK DDF and also execute the corresponding initialization routine.
	Init_RTC_Device()	Adds the RTC device to the RZK DDF and also execute the corresponding initialization routine.
	Init_EMAC_Device()	Adds the EMAC device to the RZK driver frame work and also execute the corresponding initialization routine.
	Init_TTY_Device()	Adds the TTY device to the RZK driver frame work and also execute the corresponding initialization routine.
	nifDriverInit()	Initializes network interfaces.
ZTPinit_Conf.c	DHCP_Init()	Initiates the DHCP processes.
	CreateZTPAppThread()	ZTPAppEntry() thread is created.
	RZK_KernelStart()	all of the threads that are created will start running.

Network Configuration

ZTP features a network stack configuration in which certain components can be included – or different stack parameters can be modified – based on system requirements. These variables and macros are located in the following filepath:

```
..\ZTP\Conf\
```

The variables and macros for the eZ80F91, eZ80F92, eZ80F93 and eZ80L92 devices are defined in the `ZTPConfig.c` configuration file; the configuration file for the eZ80F91 Mini configuration is named `ZTPConfig_mini.c`.

Table 4 lists the ZTP core configuration and the values for the variables or macros that are a part of the configuration.

Table 4. ZTP Core Configuration

File To Modify	Variable/Macro To Modify	Valid Configuration Values
ZTP\Conf\ ZTPConfig.c	MAX_IP_RX_BUFFH	Maximum number of IP receiver buffer size. Default value: 16.
	MAX_TCP_CONNECTIONS	Maximum number of TCP connections allowed at a specific point of time. Default value: 24.
	MAX_RX_BUFSIZE	Maximum TCP internal receiver buffer size. Default value: 4096.
	MAX_TX_BUFSIZE	Maximum TCP internal transmit buffer size. Default value: 4096.
	MAX_UDP_CONNECTIONS	Maximum number of UDP connections allowed at a specific point of time. Default value: 6.