LibXil Net (v2.00.a)

EDK 8.2i, April, 2006

Summary

This document describes the network library for Embedded processors, libXilNet (v2.00.a). This library provides a new easy to use interface with the relevant parameters. The older libXilNet(v1.00.a) library is deprecated and is not MLD compatible with the new version (v2.00.a). The new xilnet v2.00.a library is code backwards compatible with the older version v1.00.a. Hence any application compile with xilnet v1.00.a will still compile and work the same under v2.00.a.

The library includes functions to support the TCP/IP stack and the higher level application programming interface (Socket APIs).

The document contains the following sections.

Overview

The Embedded Development Kit (EDK) networking library, **libXilNet**, allows a processor to connect to the internet. LibXilNet includes functions for handling the TCP/IP stack protocols. It also provides a simple set of Sockets Application Programming Interface (APIs) functions enabling network programming. Lib Xil Net supports multiple connections (through Sockets interface) and hence enables multiple client support. This chapter describes the various functions of LibXilNet.

LibXilNet Functions

LibXilNet Function Summary

Table 1 provides a list of LibXilNet functions with links to detailed descriptions of each.

Table 1: **LibXilNet Functions**

Functions
`int xilsock_init (void)`
`void xilsock_rel_socket (int sd)`
`int xilsock_socket (int domain, int type, int proto)`
`int xilsock_bind (int sd, struct sockaddr* addr, int addrlen)`

Table 1: **LibXilNet Functions *(Continued)***

Functions
int xilsock_listen (int sd, int backlog)
int xilsock_accept (int sd, struct sockaddr* addr, int *addrlen)
int xilsock_recvfrom (int s, unsigned char* buf, int len, struct sockaddr* from, unsigned int fromlen)
int xilsock_sendto (int s, unsigned char* buf, int len, struct sockaddr* to, unsigned int tolen)
int xilsock_recv (int s, unsigned char* buf, int len)
int xilsock_send (int s, unsigned char* buf, int len)
void xilsock_close (int s)
void xilnet_mac_init (unsigned int baseaddr)
void xilnet_eth_init_hw_addr (unsigned char* addr)
int xilnet_eth_recv_frame (unsigned char* frame, int len)
void xilnet_eth_send_frame (unsigned char* frame, int len, unsigned char* dipaddr, void *dhaddr, unsigned short type)
void xilnet_eth_update_hw_tbl (unsigned char* frame, int proto)
void xilnet_eth_add_hw_tbl_entry (unsigned char* ip, unsigned char* hw)
int xilnet_eth_get_hw_addr (unsigned char* ip)
void xilnet_eth_init_hw_addr_tbl (void)
int xilnet_arp (unsigned char* buf, int len)
void xilnet_arp_reply (unsigned char* buf, int len)
void xilnet_ip_init (unsigned char* ip_addr)
int xilnet_ip (unsigned char* buf, int len)
void xilnet_ip_header (unsigned char* buf, int len, int proto)
unsigned short xilnet_ip_calc_chksum (unsigned char* buf, int len, int proto)
int xilnet_udp (unsigned char* buf, int len)
void xilnet_udp_header (struct xilnet_udp_conn conn, unsigned char* buf, int len)
unsigned short xilnet_udp_tcp_calc_chksum (unsigned char* buf, int len, unsigned char* saddr, unsigned char* daddr, unsigned short proto)
void xilnet_udp_init_conns (void)
int xilnet_udp_open_conn (unsigned short port)
int xilnet_udp_close_conn (struct xilnet_udp_conn *conn)
int xilnet_tcp (unsigned char* buf, int len)

Table 1: **LibXilNet Functions *(Continued)***

Functions
`void xilnet_tcp_header (struct xilnet_tcp_conn conn, unsigned char* buf, int len)`
`void xilnet_tcp_send_pkt (struct xilnet_tcp_conn conn, unsigned char* buf, int len, unsigned char flags)`
`void xilnet_tcp_init_conns (void)`
`int xilnet_tcp_open_conn (unsigned short port)`
`int xilnet_tcp_close_conn (struct xilnet_tcp_conn *conn)`
`int xilnet_icmp (unsigned char* buf, int len)`
`void xilnet_icmp_echo_reply (unsigned char* buf, int len)`

Protocols Supported

LibXilNet supports drivers and functions for the Sockets API and protocols of TCP/IP stack. The following list enumerates them.

- Ethernet Encapsulation (RFC 894)
- Address Resolution Protocol (ARP - RFC 826)
- Internet Protocol (IP - RFC 791)
- Internet Control Management Protocol (ICMP - RFC 792)
- Transmission Control Protocol (TCP - RFC 793)
- User Datagram Protocol (UDP - RFC 768)
- Sockets API

Library Architecture

Figure 1 gives the architecture of libXilNet. Higher Level applications like HTTP server, TFTP (Trivial File Transfer Protocol), PING etc., uses API functions to use the libXilNet library

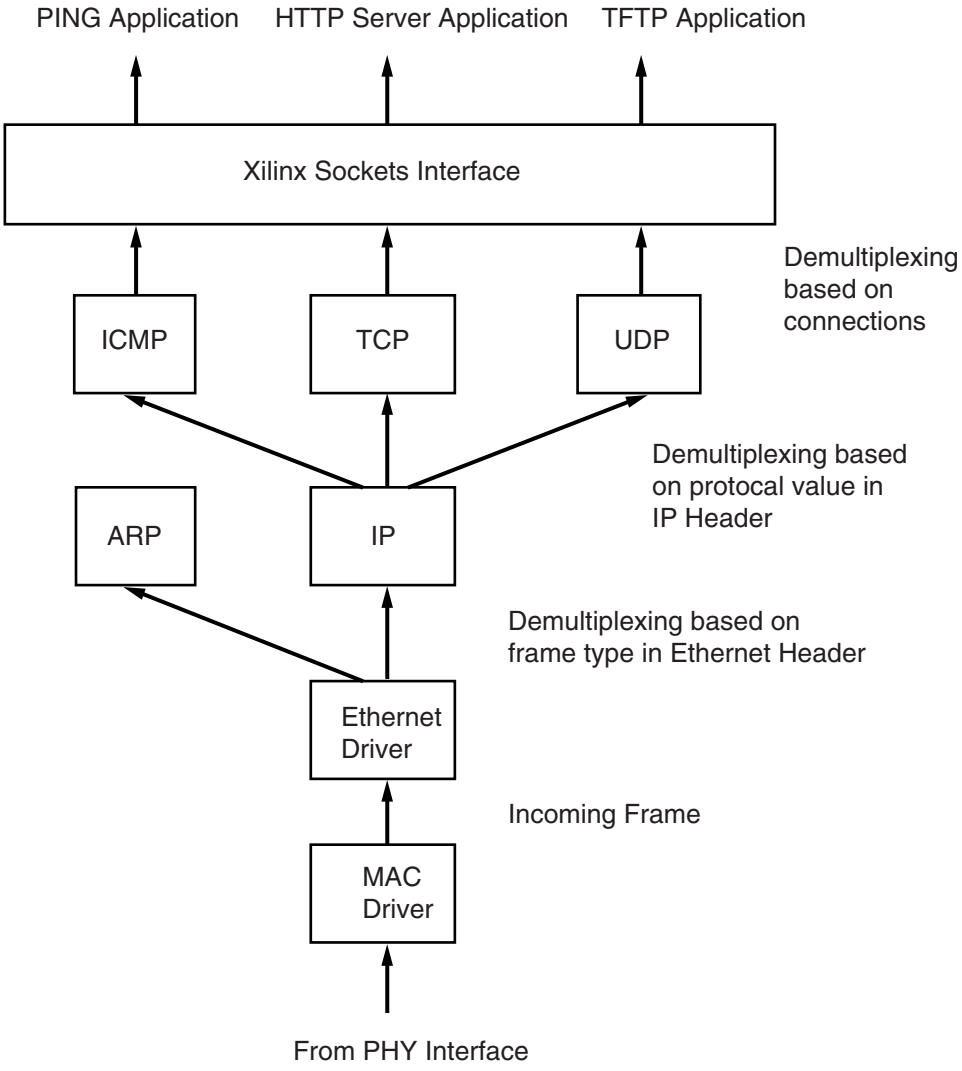

Figure 1: **Schematic Diagram of LibXilNet Architecture**

Protocol Function Description

A detailed description of the drivers and the protocols supported is given below.

Media Access Layer (MAC) Drivers Wrapper

MAC drivers wrapper initializes the base address of the mac instance specified by the user.This base address is used to send and receive frames. This initialization must be done before using other functionality of LibXil Net library. The details of the function prototype is defined in the section"Functions of LibXilNet."

Ethernet Drivers

Ethernet drivers perform the encapsulation/removal of ethernet headers on the payload in accordance with the RFC 894. Based on the type of payload (IP or ARP), the drivers call the corresponding protocol callback function. A Hardware Address Table is maintained for mapping 48-bits ethernet address to 32-bits IP address.

ARP (RFC 826)

Functions are provided for handling ARP requests. An ARP request (for the 48-bit hardware address) is acknowledged with the 48-bit ethernet address in the ARP reply. Currently, ARP request generation for a desired IP address is not supported. The Hardware address table is updated with the new IP/Ethernet address pair if the ARP request is destined for the processor.

IP (RFC 791)

IPv4 datagrams are used by the higher level protocols like ICMP, TCP, and UDP for receiving/sending data. A callback function is provided for ethernet drivers which is invoked whenever there is an IP datagram as a payload in an ethernet frame. Minimal processing of the source IP address check is performed before the corresponding higher level protocol (ICMP, TCP, UDP) is called. Checksum is calculated on all the outgoing IP datagrams before calling the ethernet callback function for sending the data. An IP address for a Embedded Processor needs to be programmed before using it for communication. An IP address initializing function is provided. Refer to the table describing the various routines for further details on the function. Currently no IP fragmentation is performed on the outgoing datagrams. The Hardware address table is updated with the new IP/Ethernet address pair if an IP packet was destined for the processor.

ICMP (RFC 792)

ICMP functions handling only the echo requests (ping requests) are provided. Echo requests are issued as per the appropriate requirements of the RFC (Requests For Comments).

UDP (RFC 768)

UDP is a connectionless protocol. The UDP callback function, called from the IP layer, performs the minimal check of source port and strips off the UDP header. It demultiplexes from the various open UDP connections. A UDP connection can be opened with a given source port number through Socket functions. Checksum calculation is performed on the outgoing UDP datagram. The number of UDP connections that can be supported simultaneously is configurable.

TCP (RFC 793)

TCP is a connection-oriented protocol. Callback functions are provided for sending and receiving TCP packets. TCP maintains connections as a finite state machine. On receiving a TCP packet, minimal check of source port correctness is done, before demultiplexing the TCP packet from the various TCP connections. Necessary action for the demultiplexed connection is taken based on the current machine state. A status flag is returned to indicate the kind of TCP packet received to support connection management. Connection management has to be done at the application level using the status flag received from TCP. Checksum is calculated on all outgoing TCP packets. The number of TCP connections that can be supported simultaneously is configurable.

Sockets API

Functions for creating sockets (TCP/UDP), managing sockets, sending and receiving data on UDP and TCP sockets are provided. High level network applications need to use these functions for performing data communication. Refer to Table 1 for further details.

Buffer Management

XiNet stack is geared to work with smaller FPGA devices and hence minimal buffer management is provided. The stack uses two global buffers - **sendbuf, recvbuf** - to send and receive ethernet frames. User code can either allocate a buffer or use **sendbuf** to send packets from the application. When **sendbuf** is used to transmit packets, user code is responsible to place the application data at the right offset from start of **sendbuf** accounting for all layers of stack starting from ethernet header.

Current Restrictions

Certain restrictions apply to the EDK libXilNet library software. These are

- Only server functionality for ARP. This means ARP requests are not being generated from the processor
- Only server functionality in libXilNet. This means no client application development support provided in libXilNet.
- No timers in TCP. Since there are no timers used, every “send” over a TCP connection waits for an “ack” before performing the next “send”.

Functions of LibXilNet

The following table gives the list of functions in libXilNet and their descriptions

---

`int` **`xilsock_init`** `(void)`

Parameters	None
Returns	1 for success and 0 for failure
Description	Initialize the xilinx internal sockets for use.
Includes	xilsock.h

`void` **`xilsock_rel_socket`** `(int` *`sd`*`)`

Parameters	*sd* is the socket to be released.
Returns	None
Description	Free the system level socket given by the socket descriptor *sd*
Includes	xilsock.h

int **xilsock_socket** (int *domain*, int *type*, int *proto*)

Parameters	*domain:* Socket Domain *type*: Socket Type *proto*: Protocol Family
Returns	On success, return socket descriptor On failure, return -1
Description	Create a socket of type, domain and protocol proto and returns the socket descriptor. The type of sockets can be: SOCK_STREAM (TCP socket) SOCK_DGRAM (UDP socket) *domain* value currently is AF_INET *proto* refers to the protocol family which is typically the same as the *domain*.
Includes	xilsock.h

int **xilsock_bind** (int *sd*, struct sockaddr* *addr*, int *addrlen*)

Parameters	*sd:* Socket descriptor *addr:* Pointer to socket structure *addrlen*: Size of the socket structure
Returns	On success, return 1 On failure, return -1
Description	Bind socket given the descriptor *sd* to the ip address/port number pair given in structure pointed to by *addr* of len *addrlen*. *addr* is the typical socket structure.
Includes	xilsock.h

int **xilsock_listen** (int *sd*, int *backlog*)

Parameters	*sd:* Socket descriptor *backlog*: Number of simultaneous connections that can try to connect to server port
Returns	On success, return 1
Description	Listen on the socket descriptor *sd* for new connections. Currently this is a dummy call and is not implemented.
Includes	xilsock.h

int **xilsock_accept** (int *sd*, struct sockaddr* *addr*, int *addrlen)

Parameters

sd: Socket descriptor

addr: Pointer to socket structure

addrlen: Pointer to the size of the socket structure

Returns

On success, return socket descriptor

On failure, return -1

Description

Accepts new connections on socket *sd*. If a new connection request arrives, it creates a new socket *nsd*, copies properties of *sd* to *nsd*, returns *nsd*. If a packet arrives for an existing connection, returns 0 and sets the xilsock_status_flag global variable. The various values of the is flag are:

XILSOCK_NEW_CONN

XILSOCK_CLOSE_CONN

XILSOCK_TCP_ACK

for new connection, closed a connection and acknowledgment for data sent for a connection respectively.

This function implicitly polls/waits on a packet from MAC. Arguments *addr* and *addrlen* are in place to support the standard Socket accept function signature. At present, they are not used in the accept function.

Includes

xilsock.h

int **xilsock_recvfrom** (int *s*, unsigned char* *buf*, int *len*, struct sockaddr* from, unsigned int fromlen)

Parameters

s: UDP socket descriptor

buf: Buffer to receive data

len: Buffer size

from: Socket address of sender to be copied

fromlen: Length of the socket *from*

Returns

Number of bytes received if data received for this socket else -1

Description

Receives data (maximum length of *len*) from the UDP socket for socket *s* in *buf* and returns the number of bytes received. The address of the sender is copied onto socket *from* and the length of socket *from* is copied onto *fromlen*

Includes

xilsock.h

int **xilsock_sendto** (int *s*, unsigned char* *buf*, int *len*, struct sockaddr* to, unsigned int tolen)

Parameters	*s:* UDP socket descriptor *buf*: Buffer containing data to be sent *len*: Buffer size *to:* Send data to socket represented by address *to* *tolen:* Length of the socket *to*
Returns	Number of bytes received
Description	Sends data of length *len* in *buf* from the UDP socket *s* to destination *to* and returns the number of bytes sent.
Includes	xilsock.h

int **xilsock_recv** (int *s*, unsigned char* *buf*, int *len*)

Parameters	*s:* TCP socket descriptor *buf*: Buffer to receive data *len*: Buffer size
Returns	Number of bytes received
Description	Receives data (maximum length of *len*) from the TCP socket *s* in *buf* and returns the number of bytes received.
Includes	xilsock.h

int **xilsock_send** (int *s*, unsigned char* *buf*, int *len*)

Parameters	*s:* TCP socket descriptor *buf*: Buffer containing data to be sent *len*: Buffer size
Returns	Number of bytes received
Description	Sends data of length *len* in *buf* on the UDP socket *s* and returns the number of bytes sent.
Includes	xilsock.h

void **xilsock_close** (int *s*)

Parameters	*s:* socket descriptor
Returns	None
Description	Closes the socket connection given by the descriptor *s*. This function has to be called from the application for a smooth termination of the connection after a connection is done with the communication.
Includes	xilsock.h

void **xilnet_mac_init** (unsigned int *baseaddr*)

Parameters	*baseaddr*: Base address of the MAC instance used in a system
Returns	None
Description	Initialize the MAC base address used in the libXil Net library to *baseaddr*. This function has to be called at the start of a user program with the base address used in the MHS file for ethernet before starting to use other functions of libXil Net library.
Includes	mac.h

void **xilnet_eth_init_hw_addr** (unsigned char* *addr*)

Parameters	*addr:* 48-bit colon separated hexa decimal ethernet address string
Returns	None
Description	Initialize the source ethernet address used in the libXil Net library to *addr*. This function has to be called at the start of a user program with a 48-bit, colon separated, hexa decimal ethernet address string for source ethernet address before starting to use other functions of libXil Net library. This address will be used as the source ethernet address in all the ethernet frames.
Includes	xilsock.h mac.h

int **xilnet_eth_recv_frame** (unsigned char* *frame*, int *len*)

Parameters	*frame*: Buffer for receiving an ethernet frame *len*: Buffer size
Returns	Number of bytes received
Description	Receives an ethernet frame from the MAC, strips the ethernet header and calls either *ip* or *arp* callback function based on frame type. This function is called from *accept /receive* socket functions. The function receives a frame of maximum length *len* in buffer *frame*.
Includes	xilsock.h mac.h

void **xilnet_eth_send_frame** (unsigned char* *frame*, int *len*, unsigned char* *dipaddr*, void **dhaddr*, unsigned short *type*)

Parameters	*frame*: Buffer for sending a ethernet frame *len*: Buffer size *dipaddr*: Pointer to the destination ip address *dhaddr*: Pointer to the destination ethernet address *type*: Ethernet Frame type (IP or ARP)
Returns	None
Description	Creates an ethernet header for payload *frame* of length *len*, with destination ethernet address *dhaddr*, and frame type, *type*. Sends the ethernet frame to the MAC. This function is called from *receive/send* (both versions) socket functions.
Includes	xilsock.h mac.h

void **xilnet_eth_update_hw_tbl** (unsigned char* *frame*, int *proto*)

Parameters	*frame*: Buffer containing an ethernet frame *proto*: Ethernet Frame type (IP or ARP)
Returns	None
Description	Updates the hardware address table with ipaddress/hardware address pair from the ethernet frame pointed to by *frame*. *proto* is used in identifying the frame (ip/arp) to get the ip address from the ip/arp packet.,
Includes	xilsock.h mac.h

void **xilnet_eth_add_hw_tbl_entry** (unsigned char* *ip*, unsigned char* *hw*)

Parameters	*ip*: Buffer contains ip address *hw*: Buffer containing hardware address
Returns	None
Description	Add an ip/hardware pair entry given by *ip/hw* into the hardware address table
Includes	xilsock.h mac.h

int **xilnet_eth_get_hw_addr** (unsigned char* *ip*)

Parameters	*ip*: Buffer containing ip address
Returns	Index of entry in the hardware address table that matches the *ip* address
Description	Receives an ethernet frame from the MAC, strips the ethernet header and calls either *ip* or *arp* callback function based on the frame type. This function is called from *accept /receive* socket functions. The function receives a frame of maximum length *len* in buffer *frame*.
Includes	xilsock.h mac.h

void **xilnet_eth_init_hw_addr_tbl** (void)

Parameters	None
Returns	None
Description	Initializes Hardware Address Table. This function must be called in the user program before using other functions of LibXilNet.
Includes	xilsock.h mac.h

int **xilnet_arp** (unsigned char* *buf*, int *len*)

Parameters	*buf*: Buffer for holding the ARP packet *len*: Buffer size
Returns	0
Description	This is the *arp* callback function. It gets called by the ethernet driver for *arp* frame type. The *arp* packet is copied onto the *buf* of length *len*.
Includes	xilsock.h

void **xilnet_arp_reply** (unsigned char* *buf*, int *len*)

Parameters	*buf*: Buffer containing the ARP reply packet *len*: Buffer size
Returns	None
Description	This function sends the *arp* reply, present in *buf* of length *len*, for *arp* requests. It gets called from the *arp* callback function for *arp* requests.
Includes	xilsock.h

void **xilnet_ip_init** (unsigned char* *ip_addr*)

Parameters	*ip_addr*: Array of four bytes holding the ip address to be configured
Returns	None
Description	This function initializes the ip address for the processor to the address represented in *ip_addr* as a dotted decimal string. This function must be called in the application before any communication.
Includes	xilsock.h

int **xilnet_ip** (unsigned char* *buf*, int *len*)

Parameters	*buf*: Buffer for holding the IP packet *len*: Buffer size
Returns	0
Description	This is the ip callback function. It gets called by the ethernet driver for ip frame type. The *ip* packet is copied onto the *buf* of length *len*. This function calls in the appropriate protocol callback function based on the protocol type.
Includes	xilsock.h

void **xilnet_ip_header** (unsigned char* *buf*, int *len*, int *proto*)

Parameters	*buf:* Buffer for the ip packet *len:* Length of the ip packet *proto:* Protocol Type in IP packet
Returns	None
Description	This function fills in the ip header from the start of *buf*. The ip packet is of length *len* and *proto* is used to fill in the protocol field of ip header. This function is called from the *receive/send* (both versions) functions.
Includes	xilsock.h

unsigned short **xilnet_ip_calc_chksum** (unsigned char* *buf*, int *len*, int *proto*)

Parameters	*buf*: Buffer containing ip packet *len*: Length of the ip packet
Returns	checksum calculated for the given ip packet
Description	This function calculates the checksum for the ip packet *buf* of length *len*. This function is called from the ip header creation function.
Includes	xilsock.h

int **xilnet_udp** (unsigned char* *buf*, int *len*)

Parameters *buf*: Buffer containing the UDP packet

len: Length of the UDP packet

Returns Length of the data if packet is destined for any open UDP connections else returns 0

Description This is the *udp* callback function which is called when ip receives a udp packet. This function checks for a valid udp port, strips the udp header, and demultiplexes from the various UDP connections to select the right connection.

Includes xilsock.h

void **xilnet_udp_header** (struct xilnet_udp_conn *conn*, unsigned char* *buf*, int *len*)

Parameters conn: UDP connection

buf: Buffer containing udp packet

len: Length of udp packet

Description This function fills in the *udp* header from the start of *buf* for the UDP connection *conn*. The udp packet is of length *len*. This function is called from the *receivefrom/sendto* socket functions.

Includes xilsock.h

unsigned short **xilnet_udp_tcp_calc_chksum** (unsigned char* *buf*, int *len*, unsigned char* *saddr*, unsigned char* *daddr*, unsigned short *proto*)

Parameters	*buf*: Buffer containing UDP/TCP packet *len*: Length of udp/tcp packet *saddr*: IP address of the source *daddr*: Destination IP address *proto*: Protocol Type (UDP or TCP) Returns the
Returns	Checksum calculated for the given udp/tcp packet
Description	This function calculates and fills the *checksum* for the *udp/tcp* packet *buf* of length *len*. The source ip address (*saddr*), destination ip address(*daddr*) and protocol (*proto*) are used in the checksum calculation for creating the pseudo header. This function is called from either the udp header or the tcp header creation function.
Includes	xilsock.h

void **xilnet_udp_init_conns** (void)

Parameters	None
Returns	None
Description	Initialize all UDP connections so that the states of all the connections specify that they are usable.
Includes	xilsock.h

int **xilnet_udp_open_conn** (unsigned short *port*)

Parameters	port: UDP port number
Returns	Connection index if able to open a connection. If not returns -1.
Description	Open a UDP connection with port number *port*.
Includes	xilsock.h

int **xilnet_udp_close_conn** (struct xilnet_udp_conn *conn)

Parameters	*conn*: UDP connection
Returns	1 if able to close else returns -1.
Description	Close a UDP connection *conn*.
Includes	xilsock.h

int **xilnet_tcp** (unsigned char* *buf*, int *len*)

Parameters	*buf*: Buffer containing the TCP packet *len*: Length of the TCP packet
Returns	A status flag based on the state of the connection for which the packet has been received
Description	This is the *tcp* callback function which is called when ip receives a tcp packet. This function checks for a valid tcp port and strips the tcp header. It maintains a finite state machine for all TCP connections. It demultiplexes from existing TCP open/listening connections and performs an action corresponding to the state of the connection. It returns a status flag which identifies the type of TCP packet received (data or ack or fin).
Includes	xilsock.h

void **xilnet_tcp_header** (struct xilnet_tcp_conn *conn*, unsigned char* *buf*, int *len*)

Parameters	conn: TCP connection *buf*: Buffer containing tcp packet *len*: Length of tcp packet
Returns	None
Description	This function fills in the *tcp* header from the start of *buf* for the TCP connection *conn*. The tcp packet is of length *len*. It sets the flags in the tcp header.
Includes	xilsock.h

void **xilnet_tcp_send_pkt** (struct xilnet_tcp_conn *conn*, unsigned char* *buf*, int *len*, unsigned char *flags*)

Parameters	conn: TCP connection *buf*: Buffer containing TCP packet *len*: Length of tcp packet
Returns	The checksum calculated for the given udp/tcp packet
Description	This function sends a tcp packet, given by *buf* of length *len*, with *flags* (ack/rst/fin/urg/psh) from connection *conn*.
Includes	xilsock.h

void **xilnet_tcp_init_conns** (void)

Parameters	None
Returns	None
Description	Initialize all TCP connections so that the states of all the connections specify that they are usable.
Includes	xilsock.h

int **xilnet_tcp_open_conn** (unsigned short *port*)

Parameters	port: TCP port number
Returns	Connection index if able to open a connection. If not returns -1.
Description	Open a TCP connection with port number *port*.
Includes	xilsock.h

int **xilnet_tcp_close_conn** (struct xilnet_tcp_conn *conn)

Parameters	*conn*: TCP connection
Returns	1 if able to close else returns -1.
Description	Close a TCP connection *conn*.
Includes	xilsock.h

int **xilnet_icmp** (unsigned char* *buf*, int *len*)

Parameters	buf: Buffer containing ICMP packet len: Length of the ICMP packet
Returns	0
Description	This is the icmp callback function which is called when ip receives a icmp echo request packet (ping request). This function checks only for a echo request and sends in an icmp echo reply.
Includes	xilsock.h

void **xilnet_icmp_echo_reply** (unsigned char* *buf*, int *len*)

Parameters	buf: Buffer containing ICMP echo reply packet len: Length of the ICMP echo reply packet
Returns	None
Description	This functions fills in the icmp header from the start of buf. The icmp packet is of length *len*. It sends the icmp echo reply by calling the ip, ethernet send functions. This function is called from the icmp callback function.
Includes	xilsock.h

Configuring XilNet v2.00.a in EDK

In EDK, the XilNet library is configured through XPS (Platform Studio) GUI. From the tree view of XPS, right-click on any peripheral and select **S/W Settings...** option. This launches the Software Platform Settings dialog box. The bottom half of the Software Platform panel displays the libraries available in EDK. Select **xilnet** library version 2.00.a in this panel. Click on the Library/OS Parameters tab to configure xilnet library. Table 2 lists the configurable parameters for xilnet library.

Table 2: **Configurable Parameters for XilNet in MSS**

Parameter	Description
emac_instname	Name of the EMAC instance to be used with xilnet
no_of_tcp_conns	Number of Open TCP Connections
no_of_udp_conns	Number of Open UDP Connections

The *emac_instname* parameter should be set to the ethernet core (regular ethernet core of the lite version, ethernetlite) that is to be used with xilnet library. Setting *no_of_tcp_conns* lets the stack reserve space for the number of open TCP connections. Similarly, *no_of_udp_conns* will let the stack reserve space for the number of open UDP connections. Configuring the xilnet library in XPS for the above parameters will result in the following snippet in the MSS file.

```
BEGIN LIBRARY
PARAMETER LIBRARY_NAME = xilnet
PARAMETER LIBRARY_VER = 2.00.a
PARAMETER emac_instname = Ethernet_MAC
PARAMETER no_of_tcp_conns = 5
PARAMETER no_of_udp_conns = 5
END
```

Using XilNet in Application

Libgen generates configuration files xilnet_config.h, xilnet_config.c based on the parameter selection in MSS file. These files are built into the xilnet library when Libgen is run.

In order to use the XilNet functions in your application, you need to do the following initialization:

- Define "`#include <net/xilsock.h>`" in your C-file.
- Initialize Ethernet Hardware Table by calling the following function in the application:
  - xilnet_eth_init_hw_addr_tbl();
- Setup MAC and IP addresses using the following functions :
  - xilnet_eth_init_hw_addr("00:00:00:00:22:38");
  - xilnet_ip_init("149.199.6.108");
- Initialize Ethernet/EthernetLite drivers as instructed in the driver. For example, the emac driver call for initialization would be:

```
XEmac_mSetMacAddress(XPAR_ETHERNET_MAC_BASEADDR, mb_hw_addr);
where hw_addr is a character array representing the 48 bit MAC address
XEmac_mEnable(XPAR_ETHERNET_MAC_BASEADDR);
```