

XGE 10Gb Ethernet Software Support Overview

VxWorks 6.x, Linux 2.6.x

Abstract

The XGE 10G series of 10 GbE NICs provides dual port 10 Gigabit Ethernet (10GbE) connectivity for embedded systems with the high performance characteristics that are essential for data intensive real-time systems, yet maintaining 100% interoperability and compatibility with standard Ethernet infrastructures.

TCP and UDP performance of up to 800 MB/s under VxWorks and 1200 MB/s under Linux can be achieved using XGE 10G drivers.

This paper describes the XGE 10G drivers available for VxWorks 6.x and Linux 2.6.x.

Critical I/O 10 Gb Ethernet Software Support

VxWorks 6.x, Linux 2.6.x

The XGE 10G series of 10 GbE NICs provides dual port 10 Gigabit Ethernet (10GbE) connectivity to embedded systems with the high performance characteristics that are essential for data intensive real-time systems, yet maintaining 100% interoperability and compatibility with standard Ethernet infrastructures.

The XGE 10G provides a balanced architecture which offers hardware acceleration for bulk data transfers with the flexibility of a programmable protocol processor to significantly improve network performance. It reduces the CPU cycles and burden required for 10 GbE networking, maximizing I/O bandwidth without sacrificing host CPU efficiency.

This paper describes the XGE 10G drivers available for VxWorks 6.x and Linux 2.6.x.

XGE 10G Driver Models

The Critical I/O XGE 10G VxWorks and Linux Drivers allow user access to the 10GbE network interface through two different methods; using the standard VxWorks or Linux sockets API, or using a special high performance UDP Streaming API. The standards socket API accesses the NIC through the VxWorks/Linux network stack, similar to a normal NIC driver, while the UDP Streaming model completely bypasses the VxWorks/Linux sockets layer, instead using a special API that directly accesses the XGE 10G hardware.

Standard Sockets API Model

The standard socket API model connects the XGE 10G driver through the VxWorks or Linux sockets interface. This allows new and existing user developed socket applications and standard network applications like FTP, Telnet, NFS etc. to make use of the XGE interface. Network performance and CPU loading is excellent, but rates are limited somewhat due to the interaction of the XGE 10G hardware with the VxWorks or Linux O/S.

Streaming UDP API Model

UDP Streaming provides a high performance data transfers models which leverage the offload capabilities of the XGE 10G hardware. As the standard BSD Socket datagram send/receive API is very limited, access to the UDP streaming functionality is provided via a specialized UDP streaming send/receive API. This specialized API provides very high performance UDP sends and receives with low host CPU loading.

For sends, the UDP streaming interface is used to send application supplied blocks of data as a stream of UDP datagrams, with the datagram size being a user specified value. Datagrams must be sized to fit within the current Ethernet frame size. The XGE offload hardware will break the application supplied blocks of data into a sequence of UDP datagrams, which relieves the host processor from the overhead of doing multiple individual datagram sends. Thus the application may pass very large blocks of data to the UDP streaming API to be sent, with no CPU involvement needed to perform that datagram sends, other than the initial send setup.

For receives, the application provides a large data buffer to the UDP streaming API that is to be filled with received datagrams for a defined IP/port. The offload hardware will fill the buffer with received datagrams. The application may pass very large receive buffers to the UDP streaming API to be filled, with negligible CPU involvement required to fill the buffers with data after the initial receive setup. The data stream is delivered to the application via a series of UDP datagrams that are written directly into application data buffers after stripping off the datagram header information.

VxWorks XGE 10G Driver

The VxWorks XGE 10G Driver allows access to the XGE 10GbE network interface via standard VxWorks network API, as well as supporting special high performance offload data transfer modes via specialized APIs such as the UDP Streaming API. The XGE 10G driver can be connected to the VxWorks network stack like any other NIC driver. This allows new and existing user developed socket applications and standard network applications like FTP, Telnet, NFS etc. to make use of the XGE interface. The driver also provides high performance data transfer APIs that allow users access to data transfer modes that take advantage of the offload capabilities of the XGE interface.

The VxWorks XGE 10G Driver supports two distinct modes of operation, which may be used concurrently. The END (Enhanced Network Device) mode of operation uses the standard VxWorks network stack, thus provides full compatibility with all VxWorks network applications, and any other applications that use the sockets API. Most of the hardware offload capabilities of the XGE 10G interfaces are not enabled in this mode. The high performance streaming API modes of operation use specialized data transfer APIs that allow full use of the offload capabilities of the XGE 10G hardware. In both modes of operation, everything that “goes on the wire” is always standard IP/Ethernet traffic that is fully compatible with standard Ethernet networks and standard Ethernet NICs. Note that the high performance offload mode of operation is only compatible with VxWorks kernel applications; it is not available for VxWorks Real Time Process (RTP) applications.

For VxWorks, two sub-modes of operation are also available using the standard sockets (END) model. The *Moderated* mode provides very good network performance combined with lowest CPU loading through the use of interrupt moderation and coalescing techniques. The tradeoff is slightly lower peak performance, and slightly higher transfer

latencies. The *Non-Moderated* mode focuses on achieving the highest possible data rates and the lowest possible transfer latencies, but at the expense of higher CPU loading.

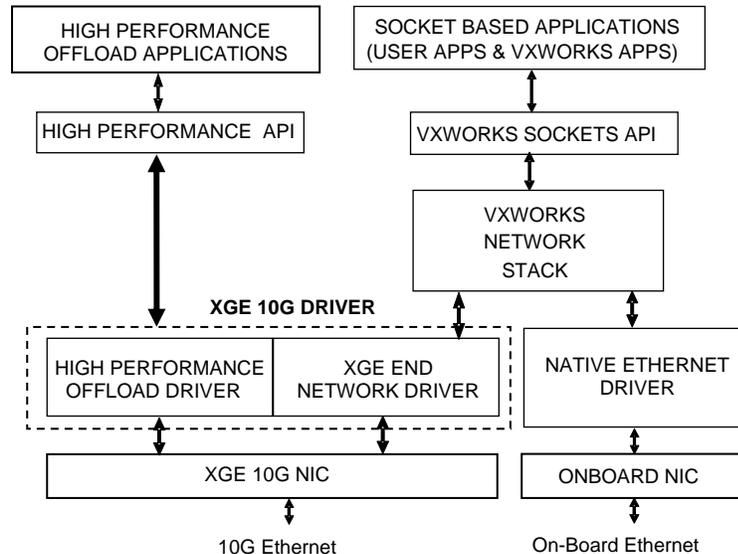


Figure 1. VxWorks XGE 10G Driver Architecture

As shown in **Error! Reference source not found.**, the XGE 10G Driver consists of two parts: a standard VxWorks END model driver and a High Performance Offload driver (e.g. UDP Streaming) model driver for higher performance data transfer modes. These two parts of the XGE 10G Driver coexist, so socket applications using the network stack and applications using the High Performance API can concurrently access the XGE network interfaces. The XGE 10G Driver through its END driver path provides for a full generic Ethernet NIC capabilities, while the High Performance API and Offload Driver path support applications requiring highest performance data transfers.

VxWorks UDP Streaming Support

UDP Streaming is a high performance data transfers modes which leverage the offload capabilities of the XGE hardware. As the standard BSD Socket datagram send/receive API is very limited, access to the UDP streaming functionality is provided via a specialized UDP streaming send/receive API. This specialized API provides very high performance UDP sends and receives with low host CPU loading.

On the send side, the UDP streaming interface is used to send application supplied blocks of data as a stream of UDP datagrams, with the datagram size being a user specified value. Datagrams must be sized to fit within the current Ethernet frame size. The XGE offload hardware will break the application supplied blocks of data into a sequence of UDP datagrams, which relieves the host processor from the overhead of doing multiple

individual datagram sends. Thus the application may pass very large blocks of data to the UDP streaming API to be sent, with no CPU involvement needed to perform that datagram sends, other than the initial send setup.

For example, if an application wishes to send a 1 MByte of data as a series of 8K datagrams, it creates a BUFLIST structure that points to the start of the 1 MByte block of data and calls the stream send function, which queues the send with the offload hardware. The offload hardware will then complete the send of 128 individual 8 KByte datagrams, directly from the application send buffer with no further host CPU involvement.

On the receive side, the data stream is delivered to the application via a series of UDP datagrams that are written directly into the application's data buffers, after stripping off the datagram header information. The application provides a large data buffer to the UDP streaming API that is to be filled with received datagrams for a defined IP/port. The offload hardware will fill the buffer with received datagrams. The application may pass very large receive buffers to the UDP streaming API to be filled, with negligible CPU involvement required to fill the buffers with data after the initial receive setup.

For example, if an application wishes to receive a 1 MByte of data as a series of 8K datagrams, it creates a BUFLIST structure that points to the start of the 1 MByte application receive buffer and calls the stream receive function, which queues the receive with the offload hardware. The offload hardware will then complete the receive of 128 individual 8 Kbyte datagrams directly into the application receive buffer (stripping off all of the Ethernet/IP/UDP headers) with virtually no further host CPU involvement.

VxWorks UDP Streaming API

The UDP Streaming API provides the application interface to send and receive streams of UDP datagrams. The functions available within this API are:

XgeStreamUdpSendSetup	- Set up a socket for UDP stream sends
XgeStreamUdpSendMulti	- Perform a UDP stream send
XgeStreamSendClose	- Close a stream send socket
XgeUdpStreamReceiveSetup	- Set up a socket for UDP stream receive
XgeStreamUdpReceiveMulti	- Perform a UDP stream receive
XgeStreamReceiveClose	- Close a stream receive socket

VxWorks Version Support

The XGE 10G Driver currently supports VxWorks versions 6.3, 6.5, 6.7 and 6.8. It is supplied in the form of a VxWorks object archive, which can be linked with the user's **boot ROM** (*Note: boot ROM integration not available in this version*) or the user's bootable **VxWorks image** project. Like any END driver the XGE 10G Driver is loaded into the system by making an entry in the BSP END Device Table. The XGE ports are available for network data transfer only after they are attached to the stack and configured with an appropriate network address and a netmask.

Linux XGE 10G Driver

The Critical I/O Linux XGE 10G Driver allows easy access to the XGE 10GbE network interface via standard Linux network API, as well as supporting special high performance offload data transfer modes via specialized APIs such as the UDP Streaming API. The XGE 10G driver can be connected to the Linux network stack like any other NIC driver. This allows new and existing user developed socket applications and standard network applications like FTP, Telnet, NFS etc. to make use of the XGE interface. The driver also provides high performance data transfer APIs that allow users access to data transfer modes that take advantage of the offload capabilities of the XGE interface.

The XGE 10G Driver supports two distinct modes of operation, which may be used concurrently. The network driver mode of operation uses the standard Linux network stack, thus provides full compatibility with all Linux network applications, and any other applications that use the sockets API. Most of the hardware offload capabilities of the XGE 10G interfaces are not enabled in this mode. The High Performance API modes of operation use specialized data transfer APIs that allow full use of the offload capabilities of the XGE 10G hardware. In both modes of operation, everything that “goes on the wire” is always standard IP/Ethernet traffic that is fully compatible with standard Ethernet networks and standard Ethernet NICs. Note that the High Performances mode of operation is only compatible with socket API.

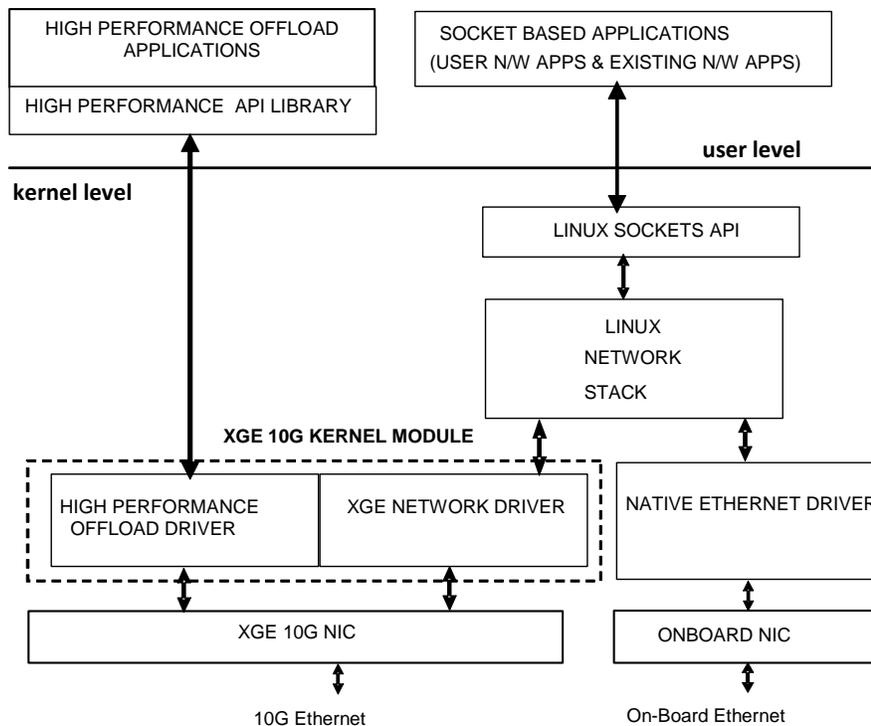


Figure 2. Linux XGE 10G Driver Architecture

As shown in figure 2, the XGE 10G Driver consists of two parts: a standard Linux network driver model and a High Performance Offload driver (e.g. UDP Streaming) model driver for higher performance data transfer modes. These two parts of the XGE 10G Driver coexist, so socket applications using the network stack and applications using the High Performance API can concurrently access the XGE network interfaces. The XGE 10G Driver through its network driver path provides for a full generic Ethernet NIC capabilities, while the High Performance API and Offload Driver path support applications requiring highest performance data transfers.

Linux UDP Streaming API

The UDP Streaming API provides the application interface to send and receive streams of UDP datagrams. The functions available within this API are:

xel_init	- Initialize the user level library
xel_end	- Clean up the user level library
xel_alloc_contig_block	- Allocate a contiguous block of memory
xel_map_contig_block	- Map a contiguous block of memory into application process's address space
xel_unmap_contig_block	- Unmap from memory contiguous memory region from application process's address space.
xel_udp_smsend_setup	- Set up a socket for UDP stream sends
xel_udp_smsend_multi	- Perform a UDP stream send
xel_udp_smsend_close	- Close a UDP stream send socket
xel_udp_smrecv_setup	- Set up a socket for UDP stream receive
xel_udp_smrecv_multi	- Perform a UDP stream receive
xel_udp_smrecv_close	- Close a UDP stream receive socket

Linux Version Support

The XGE 10G Driver standard *network* driver supports many variants of Linux base version 2.6.x, and is supplied as source code. The high performance *streaming* driver supports a smaller set of Linux variants, and is supplied as a combination of source files and an object archive. Note that the streaming driver also fully supports normal networking functionality.

Driver Performance

Running on an dual core Intel i7 platform, TCP and UDP performance of up to 800 MB/s under VxWorks and 1200 MB/s under Linux can be achieved using XGE 10G drivers.