COM-5403SOFT
IP/TCP CLIENT/UDP/ARP/PING STACK for GbE
VHDL SOURCE CODE OVERVIEW

Overview

Gigabit-speed IP protocols like TCP/IP can demand a high level of computation on processors. The trend has been to move the implementation of these fast but highly repetitive tasks to a TCP offload engine (TOE) to free the application processor from frequent interrupts.

The COM-5403SOFT is a generic Internet protocol stack (including the **VHDL source code**) designed to support 1Gbps speed on low-cost FPGAs. It is designed to achieve 950+ Mbps (UDP) or 450+ Mbps (per TCP client) throughputs on Gigabit Ethernet medium.

The following protocols are implemented in modular VHDL components: TCP client[1], UDP frames, ARP, PING, IP to MAC address routing table and DHCP client. Ancillary components are also included for streaming, test signal generation and bit error rate measurement.

These components can be instantiated as needed for the application. For a UDP receive-only application, one must instantiate *packet_parsing.vhd*, *arp.vhd*, *udp_rx.vhd*. The maximum number of concurrent TCP connections and UDP ports can be adjusted prior to VHDL synthesis depending on the available FPGA resources.

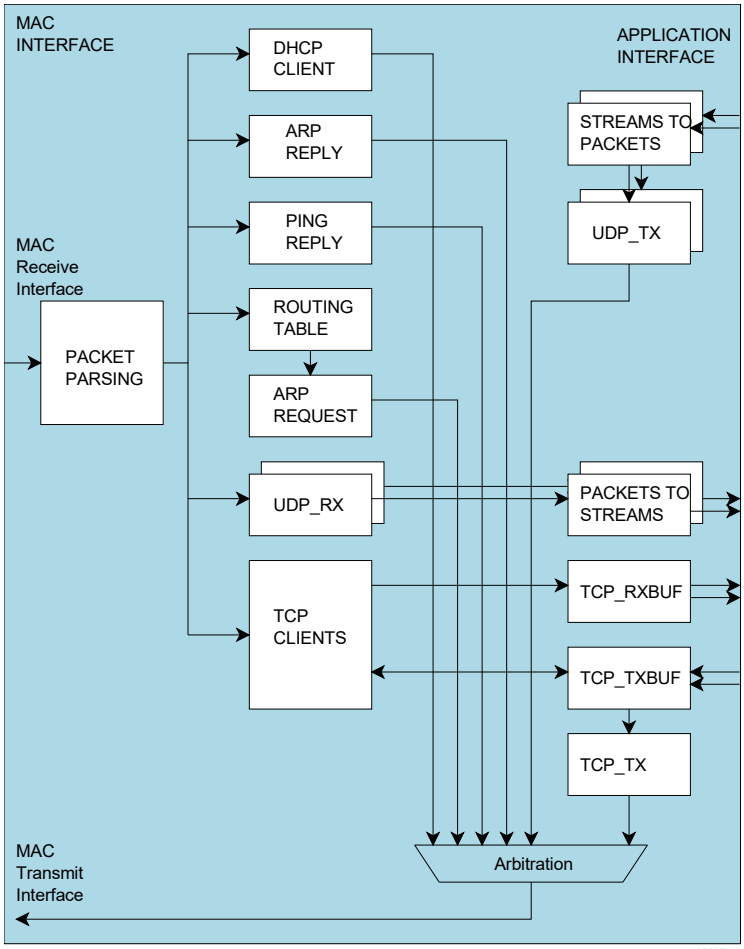
Wireshark Libpcap network capture files can be used as receiver input for simulation purposes.

The code is written specifically for IEEE 802.3 Ethernet packet encapsulation (RFC 894), IPv4 protocols.

The code interfaces seamlessly with the COM-5401SOFT Tri-mode 10/100/1000 Mbps Ethernet MAC for the MAC / PHY layers implementation. However, the MAC interface is generic and simple enough to interface with any Ethernet MAC component with minimum glue logic.

The component's very efficient implementation makes it suitable for multiple concurrent TCP and UDP streams instantiations within a small FPGA. For example, it can be configured for 2 PHYs, 2 TCP clients and 2 UDP streams in a small Spartan-6 XC6SLX16 FPGA.

Block Diagram



[1] See COM-5402SOFT for TCP server.

MSS • 845 Quince Orchard Boulevard Ste N • Gaithersburg, Maryland 20878-1676 • U.S.A.
Telephone: (240) 631-1111 Facsimile: (240) 631-1676 www.ComBlock.com
 Issued 7/23/2016

Target Hardware

The code is written in generic VHDL so that it can be ported to a variety of FPGAs. The code was developed and tested on a Xilinx Spartan-6 FPGA.

It can be easily ported to any Xilinx series 7, Virtex-6, Spartan-6, Virtex-5 FPGAs and other FPGAs capable of running at 125 MHz or above.

Device Utilization Summary

Device: Xilinx Spartan-6

	TCP client (1) and ancillary protocols
Flip Flops	2666
LUTs	4198
RAMB16s	10
DSP48A1s	0
GCLKs	3
DCMs/PLLs	0

	TCP clients (2) and ancillary protocols
Flip Flops	3125
LUTs	5162
RAMB16s	14
DSP48A1s	0
GCLKs	3
DCMs/PLLs	0

	TCP client (1) and ancillary protocols, UDP tx (1), UDP rx (1)
Flip Flops	3115
LUTs	5163
RAMB16s	11
DSP48A1s	0
GCLKs	3
DCMs/PLLs	0

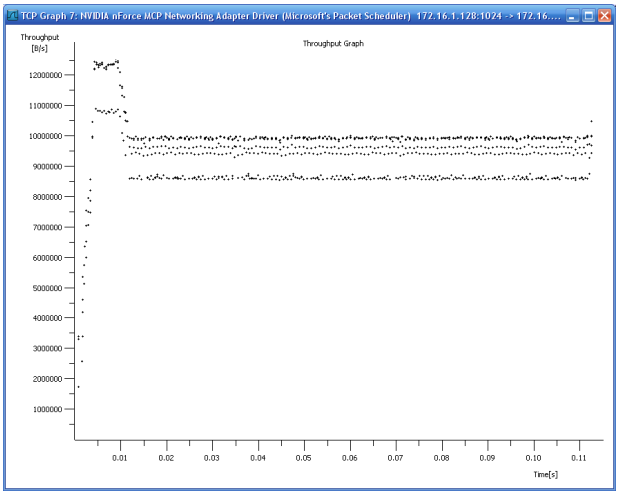
Throughput Performance Examples

Test setup1:
1 bidirectional connection between TCP server and TCP client over Gigabit Ethernet. 120 MHz FPGA processing clock. Measured sustained throughput: 452 Mbits/s concurrently in each direction.

Test setup2:
512-byte UDP packets sent point-to-point over a LAN cable. Xilinx Spartan-6 FPGA –2 speed grade.

Measured: 0 bit error, payload throughput 878.5 Mbits/s. This matches the theoretical throughput (accounting for Ethernet, IP and UDP overhead and slower (120 MHz) user clock). The maximum throughput for this UDP frame size is 915 Mbits/s when user clock is 125 MHz or above.

Test setup 3:
TCP server transmit throughput on 100 Mbps LAN Wireshark measurement on receive PC.
Average throughput 93 Mbps.



TCP Latency Performance Examples

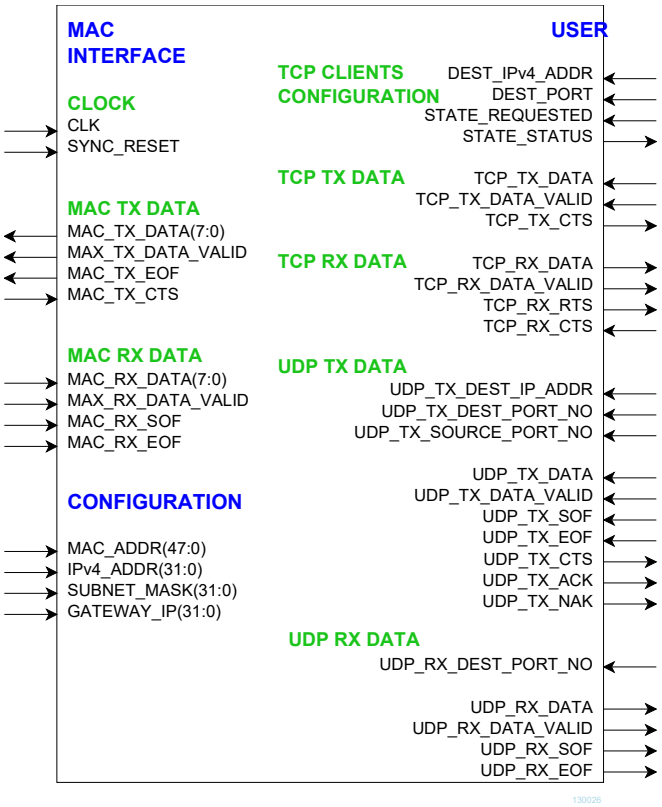
The transmit and receive latency depend on the frame length. For a maximum frame length of 1460 bytes, FPGA 125 MHz processing clock:

- Transmit latency (from the 1st byte of payload data input to the 1st byte of payload data output to the Ethernet MAC): 23.9µs
- Receive latency (from the 1st byte of Ethernet MAC input to the 1st byte of payload data output): 12.2µs

If latency is more important than throughput, the transmit segmentation threshold can be reduced to X payload bytes. In this more general case,

- Transmit latency (from the 1st byte of payload data input to the 1st byte of payload data output to the Ethernet MAC): 0.5 + 2X/125 µs
- The receive latency (from the 1st byte of Ethernet MAC input to the 1st byte of payload data output): 0.5 + X/125 µs

- Interfaces



User Interface

This interface comprises three primary signal groups: MAC interface (direct connection to COM-5401SOFT MAC core or equivalent), TCP streams, UDP frames or UDP streams to/from the user application.

All signals are clock synchronous with a user-selected clock CLK (it does not have to be the same as the PHY clock). To guarantee a 1 Gbps throughput, a minimum 125 MHz clock speed is required.

The user interface is buffered by internal elastic buffers in both tx/rx directions.

Configuration

The key configuration parameters are brought to the interface so that the user can change them dynamically at run-time. Other, more arcane, parameters are fixed at the time of VHDL synthesis.

Pre-synthesis configuration parameters

The following configuration parameters are set prior to synthesis in the *com5402pkg.vhd* package or at the top level component *com5403.vhd*.

Configuration parameters in com5402pkg.vhd	*Description*
Number of concurrent TCP streams	NTCPSTREAMS. Each additional TCP stream requires additional resources (RAM block, logic).
Number of transmit UDP ports enabled	NUDPTX
Number of receive UDP ports enabled	NUDPRX
Configuration parameters in com5403.vhd	*Description*
DHCP client enable	DHCP_CLIENT_EN '1' to instantiate a DHCP client within. DHCP is a protocol used to dynamically assign IP addresses at power up from remote DHCP servers, like a gateways. '0' when a fixed (static) IP address is defined by the user.
TCP port numbers	A block of ports is allocated, one for each client, starting at SOURCE_PORT_BASE, incremented by one for each client.
Elastic buffer size	Customized I/O elastic buffer sizes for various TCP and UDP components. Expressed as an integer number NBUFS of 16Kbits RAM blocks. NBUFS is typically restricted to 1,2,4 or 8 (see code comments). Defined as generic parameters in the following components: *tcp_rxbufndemux.vhd, tcp_txbuf.vhd, udp_tx, stream_2_packets.vhd*.

Configuration parameters in arp_cache2.vhd	*Description*
Routing table refresh period REFRESH_PERIOD(19:0)	Refresh period for this routing table. Expressed as an integer multiple of 100ms. Default value is 3000 (5 minutes).
Configuration parameters in stream_2_packets.vhd	*Description*
Maximum packet size when segmenting a stream to packets MAX_PACKET_SIZE	When segmenting a transmit stream, a packet will be sent out as soon as MAX_PACKET_SIZE bytes are collected. The recommended size is 512 bytes for a low overhead.
Inactive input stream timeout TX_IDLE_TIMEOUT	When segmenting a transmit stream, a packet will be sent out with pending data if no new data was received within the specified timeout. Expressed as integer multiple of 4µs.
Retransmission timer TX_RETRY_TIMEOUT	A re-transmission attempt will be made periodically until routing information is available and the transmit path to the MAC is available. The retry period is expressed as an integer multiple of 4µs.

Run-time configuration parameters

The user can set and modify the following controls at run-time. All controls are synchronous with the user-supplied global CLK.

Run-time configuration	*Description*
MAC address MAC_ADDR(47:0)	This network node 48-bit MAC address. The user is responsible for selecting a unique 'hardware' address for each instantiation. Natural bit order: enter x0123456789ab for the MAC address 01:23:45:67:89:ab It is essential that this input matches the MAC address used by the MAC/PHY.
Dynamic vs static IP DYNAMIC_IP	'1' for dynamic addressing '0' for static IP address. The device IP address can be assigned dynamically by an external DHCP server, or defined as static address by the user. Dynamic addressing requires instantiating a DHCP client: set the generic parameter DHCP_CLIENT_EN = '1'.
IPv4 address REQUESTED_IPv4 _ADDR(31:0)	Static address when DYNAMIC_IP = '0' Last dynamically assigned address when DYNAMIC_IP = '1'. Address 0.0.0.0 can also be used in conjunction with dynamic addressing if the user does not 'remember' the last dynamic IP address. 4 bytes for IPv4. Byte order: (MSB)192.68.1.30(LSB)
Subnet Mask SUBNET_MASK(31:0)	Subnet mask to assess whether an IP address is local (LAN) or remote (WAN) Byte order: (MSB)255.255.255.0(LSB)
Gateway IP address GATEWAY_IP(31:0)	One gateway through which packets with a WAN destination are directed. Byte order: (MSB)192.68.1.1(LSB)
Transmit UDP destination IP address UDP_TX_DEST_ IP_ADDR	The UDP destination IP address can be modified dynamically on a frame-by-frame basis.
Transmit UDP destination port number UDP_TX_DEST_ PORT_NO	The UDP destination port number can be modified dynamically on a frame-by-frame basis.
Transmit UDP source port number UDP_TX_SOURCE_ PORT_NO	The UDP source port number can be modified dynamically on a frame-by-frame basis.
Receive UDP port number UDP_RX_DEST_ PORT_NO	Local UDP port listening for incoming UDP frames. Receive and transmit UDP streams can use identical or different ports at the user's discretion.
Client TCP destination IP address DEST_IPv4_ADDR(I)	For each TCP client, define a remote server IP address to connect to. It should be set prior to triggering a connection.
Client TCP destination port DEST_PORT(I)	For each TCP client, define a remote server TCP port to connect to. It should be set prior to triggering a connection.
Client TCP connection control STATE_REQUESTED(I)	For each TCP client, set to 1 to request a connection to a remote server, 0 to clear any such connection.

Limitations

This software does not support the following:

- IEEE 802.3/802.2 encapsulation, RFC 1042, only the most common Ethernet encapsulation.

Only one gateway is supported at any given time.

Software Licensing

The COM-5403SOFT is supplied under the following key licensing terms:

1. A nonexclusive, nontransferable license to use the VHDL source code internally, and
2. An unlimited, royalty-free, nonexclusive transferable license to make and use products incorporating the licensed materials, solely in bitstream or synthesized (e.g. .ngc) format, on a worldwide basis.

The complete VHDL/IP Software License Agreement can be downloaded from http://www.comblock.com/download/softwarelicense.pdf

Configuration Management

The current software revision is 2.

Directory	Contents
/	Project files for various Xilinx ISE versions.
/doc	Specifications, user manual, implementation documents when applicable.
/src	.vhd source code, .ucf constraint files, .pkg packages. One component per file.
/sim	Testbenches
/bin	.ngc, .bit, .mcs configuration files
/use_example	use example, .ngc for Spartan-6 and instantiation template Test components (pseudo random binary sequence generator, bit error rate measurement, stream to packets segmentation, etc) are in directory \use_example\src

Key file:

Xilinx ISE project file: com-5403_ISE144.xise

VHDL development environment

The VHDL software was developed using the following development environment:

(a) Xilinx ISE 14.4 with XST as synthesis tool

(b) Xilinx ISE Isim as VHDL simulation tool

Ready-to-use Hardware

Use examples are available to run on the following Comblock hardware modules:

- COM-1500 FPGA + ARM + USB2.0+ DDR2 SODIMM SOCKET + NAND development platform
  http://www.comblock.com/com1500.html
- ComBlock COM-5102 1-Port 10/100/1000 Mbps Ethernet Transceiver
  http://www.comblock.com/com5102.html

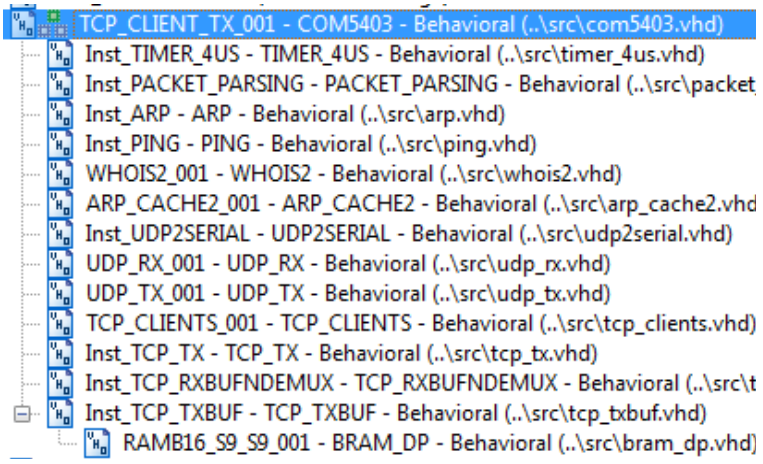
All hardware schematics are available in this CD.

Xilinx-specific code

The VHDL source code is written in generic VHDL with few Xilinx primitives. No Xilinx CORE is used. The Xilinx primitives are:

- IBUF
- IBUFG
- BUFG (global clocks)
- RAM block: RAMB16

Top-Level VHDL hierarchy



The code is stored with one, and only one, component per file.

The root entity (highlighted above) is *COM5403.vhd*. It contains instantiations of the IP protocols and a transmit arbitration mechanism to select the next packet to send to the MAC/PHY.

The root also includes the following components:

- The *PACKET_PARSING.vhd* component parses the received packets from the MAC and efficiently extracts key information relevant for multiple protocols. Parsing is done on the fly without storing data. Instantiated once.
- The *ARP.vhd* component detects ARP requests and assembles an ARP response Ethernet packet for transmission to the MAC. Instantiated once.
- The *PING.vhd* component detects ICMP echo (ping) requests and assembles a ping echo Ethernet packet for transmission to the MAC. Instantiated once.

- The *WHOIS2.vhd* component generates an ARP request (broadcast) packet requesting that the target identified by its IP address responds with its MAC address. Instantiated once. Required only if *ARP_CACHE2.vhd* is instantiated.
- The *ARP_CACHE2.vhd* component is a shared routing table that stores up to 128 IP addresses with their associated 48-bit MAC addresses and a 'freshness' timestamp. An arbitration circuit is used to arbitrate the routing request from multiple transmit instances. Instantiated once. Required if either *TCP_CLIENTS.vhd* or *UDP_TX.vhd* components are enabled.
- The flexible *UDP_TX.vhd* component encapsulates a data packet into a UDP frame addressed from any port to any port/IP destination. Instantiated once, irrespective of the number of source or destination UDP ports.
- The *UDP_RX.vhd* component validates received UDP frames and extracts the data packet within. As the validation is performed on the fly (no storage) while received data is passing through, the validity confirmation is made available at the end of the packet. The calling application should therefore be able to 'backtrack' upon receiving an invalid packet. Instantiated once, irrespective of the number of UDP ports being listened to. Although this component is written for one port, it can very easily be modified to accommodate several ports (follow the PORT_NO signal). Therefore, there is never any need to instantiate more than one component.
- The *TCP_CLIENTS.vhd* component is the heart of the TCP protocol. It is written parametrically so as to support NTCPSTREAMS concurrent TCP connections. It essentially handles the TCP state machines of several TCP clients: user-initiated connection request to user-specified remote server IP/Port address, establishing and tearing down the connections and managing flow control while the bi-directional connections are established.
- The *TCP_TX.vhd* component formats TCP tx frames, including all layers: TCP, IP, MAC/Ethernet. It is common to all concurrent streams.
- The *TCP_TXBUF.vhd* component stores TCP tx payload data in individual elastic buffers, one for each transmit stream. The buffer size is configurable prior to synthesis as NBUFS*16Kbits RAM blocks.
- The *TCP_RXBUFNDEMUX*.vhd component demultiplexes several TCP rx streams. It does not include any elastic buffer. It is expected that the user will instantiate elastic buffers if the application requires it. Data bytes are received in sequence without gaps or backtracking.

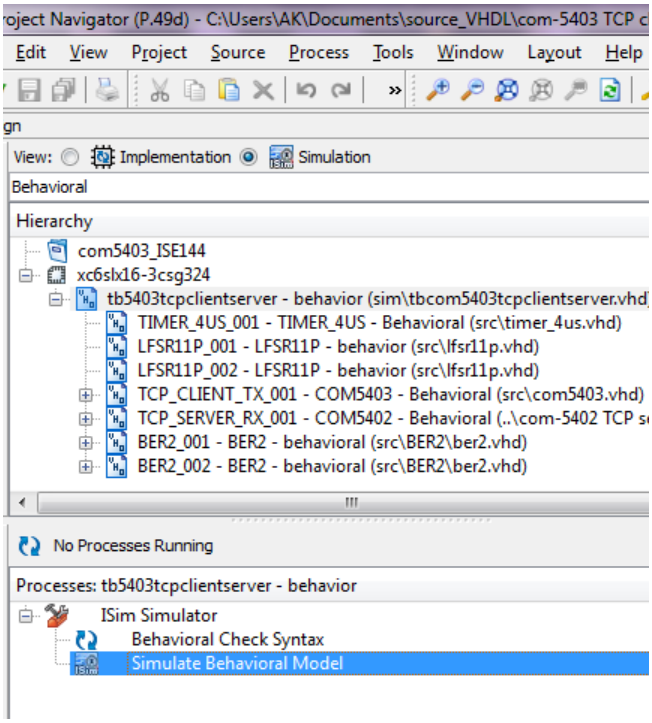
Additional components are also provided for use during system integration or tests.

- *STREAM_2_PACKETS.vhd* segments a continuous data stream into packets. The transmission is triggered by either the maximum packet size or a timeout waiting for fresh stream bytes. Useful when sending a continuous data stream over UDP.
- *PACKETS_2_STREAM.vhd* reassembles a data stream from received valid packets while discarding invalid packets. The packet's validity is assessed at the end of packet. It is designed to connect seamlessly with the *TCP_RX.vhd* component. Useful when receiving a continuous data stream over UDP.
- *LFSR11P.vhd* generates a pseudo-random binary stream PRBS11 for use during throughput and bit error rate tests. It is capable of generating 1 Gbps (8 bit per clock @ 125 MHz).
- *BER2.vhd* synchronizes with a received data stream and counts bit errors. It is also capable of working at 1 Gbps.

VHDL simulation

A testbench (tb*.vhd), located in the /sim directory, can be used to validate the source code through VHDL simulation. When applicable, ancillary files such as Wireshark LAN capture files (.cap) and Xilinx ISIM simulator configuration files (.wcfg) are also in the same /sim directory for ready-to-start simulations.

Quick start:
In the Xilinx ISE, open the com5403_ISE144.xise project. Click on the View Simulation button. The available testbenches will be displayed as illustrated below. Start the ISIM simulator.



Clock / Timing

The software uses one synchronous clock CLK. The clock should be at least 125 MHz in order to take full advantage of the Gbit Ethernet speed. The code can operate properly at less than 125 MHz, albeit at reduced throughput.

The code is written to run at 125 MHz on a Xilinx Spartan-6 –2 speed grade with 2 concurrent TCP streams instantiated.

ComBlock Compatibility List

FPGA development platform
COM-1500 FPGA + DDR2 SODIMM socket + ARM development platform
Network adapter
COM-5102 1-port 10/100/1000 Mbps Ethernet Transceiver
COM-5401 4-port 10/100/1000 Mbps Ethernet Transceivers
Software
COM-5401SOFT Tri-mode 10/100/1000 Mbps Ethernet MAC. VHDL source code.
COM-5402SOFT IP/UDP/TCP SERVER/ARP/PING stack. VHDL source code.
COM-5403SOFT IP/UDP/TCP CLIENT/ARP/PING stack. VHDL source code.

ComBlock Ordering Information

COM-5403SOFT IP/TCP CLIENT/ UDP/ARP/PING PROTOCOL STACK, VHDL SOURCE CODE

ECCN: 5E001.b.4

MSS • 845 Quince Orchard Boulevard Ste N • Gaithersburg, Maryland 20878-1676 • U.S.A.
Telephone: (240) 631-1111
Facsimile: (240) 631-1676
E-mail: sales@comblock.com