

## Features of Modbus Software Libraries

### What is Modbus Protocol?

The Modbus Protocol messaging structure was developed by Modicon in 1979 to establish master-slave/client-server communication between intelligent devices. It has since become an open protocol. The Modbus-IDA Organization has been formed to provide users and suppliers an opportunity to obtain and share information about the Modbus communication protocol. In addition, the Modbus TCP protocol specification has been recently submitted to the Internet Engineering Task Force (IETF).

Modbus is an application layer messaging protocol, positioned at level 7 of the OSI model that provides client/server communication between devices connected on different types of buses or networks. Modbus is a request/reply protocol and offers services specified by function codes. Modbus function codes are elements of Modbus request/reply PDUs.

Several implementations of Modbus are available. The Modbus Serial ASCII implementation is readable on a standard terminal window. The Modbus Serial RTU implementation is a more compact, binary protocol. Modbus Plus uses a proprietary high-speed token passing network, while Modbus TCP uses standard, inexpensive Ethernet cards.

### What are Triangle MicroWorks, Inc. Software Libraries?

Triangle MicroWorks' Software Libraries provide a cost-effective means of supporting industry-standard protocols in your device. Incorporating our royalty-free Software Libraries in your products will shorten development time, freeing internal resources to work on company proprietary aspects of your products.

Triangle MicroWorks Software Libraries are available in two formats: .NET Protocol Components for incorporation in Windows .NET-based products, and ANSI-Standard C Source Code Libraries for all other platforms.

### *Features Common to All of Our Modbus Software Libraries*

- Conforms to Modbus Application Protocol Specification V1.1b.
- Supports any physical communication network including RS 232/485 (for RTU and ASCII), Modbus Plus, and TCP.
- Can be used in event-driven or non-event-driven environments.
- Supports binary data (coils and discrete inputs) and analog data (holding registers and input registers).
- Supports function codes for read, write, and read/write multiple registers.
- Simple configuration for big-endian or little-endian byte order.
- Extensive, built-in (but removable) diagnostics including a **protocol analyzer** used to visually decipher protocol messages. The diagnostic and analyzer strings can be directed to any target system display device, even a serial port or RAM buffer.
- Records communication protocol errors such as "Unsupported function code", "Data base errors", "Address range errors", "Exception response, FC = xxx, Exception Code = xxx".
- No royalty fees per unit sold.

### ***Features of Our Modbus ANSI-Standard C Source Code Libraries***

- Written in ANSI-Standard C Source Code, under a strict corporate coding standard.
- Designed to be processor and operating system independent, using any ANSI-Standard C compiler.
- Simple configuration for big-endian or little-endian byte order.
- Can be used with or without a Real Time Operating System (RTOS).
- Database interface supports any database, ranging from direct I/O input with no storage to complex, relational databases.
- Includes sample applications and source code for Low-Level Target Interface for Linux and Windows (see *Design Details for Implementation*).
- Typical product integration times are less than three weeks.
- Easy installation through "Triangle" Source Code Library to Target Application interface (see Design Details for Implementation).

### ***Features of Modbus .NET Protocol Components***

- Based on Triangle MicroWorks, Inc. industry-proven Source Code Library design.
- Supports all .NET Languages (C#, J#, Managed C++, VB .NET, etc.) and tools.
- Compatible with .NET 2.0 Framework.
- Integrates with Visual Studio Help.
- Available as single-use or redistributable with source code.
- Source code version includes corresponding ANSI-Standard C Source Code Library.
- Includes built-in simple database with save/restore capabilities; also supports user-defined database.
- Ideal for quick development of products and tools requiring Modbus support.
- Scalable for large implementations.
- Typical product integration time of less than one week.

### ***Modbus Slave Software Library Features***

- Interoperability is maximized by making it easy to attach virtually all possible interoperability configuration settings to run-time variables or function calls.
- Supplies data to an unlimited number of host devices through an unlimited number of communication ports.
- Example Database Interface implementations are provided for testing, illustration, and as templates to be used for developing final Database Interface.

### ***Modbus Master Software Library Features***

- An unlimited number of remote devices can be configured on an unlimited number of communication ports, and new remote devices can be added at runtime.
- Multiple devices can be assigned to the same communication port to support multiple network communication topologies.
- Database manager maps received Modbus data into Target Application data points (coils, discrete input registers, holding registers, input registers, etc.).

## Efficient Installation, Testing, and Verification

### Design Objective:

Our primary design objective is to provide our customers with an ANSI Standard C **Source Code Library (SCL)** with a **Target Application (TA)** Interface that can be implemented in less than three man weeks. To accomplish this, our design divides the interface into “entry-points” from TA-to-SCL, and “calls” back into the TA from the SCL.

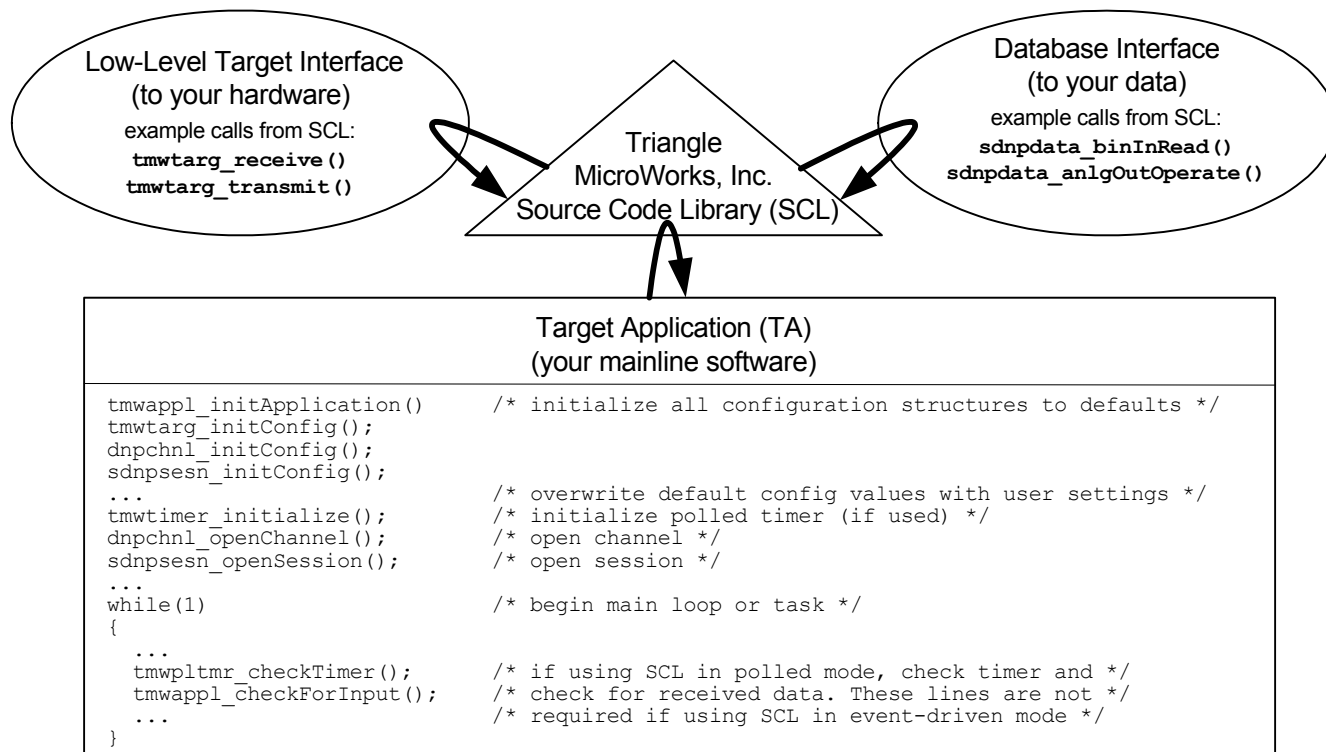
### The “Triangle” Approach to Interfacing between the SCL and TA:

The interface between the SCL and TA can be viewed as a three-sided, or “triangle” interface. Two sides represent calls back into the TA from the SCL Interface. Each of these sides is organized into individual, well-documented modules or header files. These files are the only recommended customer-editable (or platform-specific) files. All other files are protocol-specific and should not need to be modified by the customer.

The three sides of the interface are shown in the diagram below:

- 1) **TA-to-SCL Entry Points:** The entry points are limited to a few SCL initialization functions and a single process function; the process function can be called regularly as part of a Target Application main loop, or as an event-driven task in a real-time operating system environment. Master Source Code Libraries also provide C function calls to build and send request messages to remote Slave devices.
- 2) **Low-Level Target Interface:** Provides access to hardware components such as communication channels, timers, and clocks.
- 3) **Database Interface:** Provides customized fit to TA data, and allows extraction and insertion of individual database profiles, including any of our included simulated database profiles used for testing.

#### Example flow diagram for an installation of the DNP3 Slave Source Code Library



## Typical Installation Sequence for All Source Code Libraries

We strongly recommend that before you develop any communications protocol, whether or not you use our Libraries, you should create a **C**onfiguration/**I**nteroperability (**C/I**) Guide that conforms to standard “device profile” documents for each protocol. The C/I Guide specifies how to configure the protocol operation of the device; for interoperability, it specifies the objects, variations, and protocol functions that will be implemented. Our Source Code Libraries come with C/I Guide templates, with much of the information already filled in.

After completing the C/I Guide, a typical sequence to install one of our Source Code Libraries is:

- 1) Edit low-level target interface file to attach serial port receive and transmit functions, add access to a free-running millisecond timer/counter, and to configure byte-order (most or least significant first).
- 2) Add TA-to-SCL entry point functions in Target Application software (as in the DNP3 example above).
- 3) Conduct initial testing, which verifies operation of the Source Code Library on the target hardware. Initial testing consists of comparing results of requests and polls with known values from either a simulated database (for Slave Libraries), or from a known slave device (for Master Libraries). Slave Libraries include simulated database profiles with pre-set initial values of database objects; no modification of the database interface is necessary for initial testing.
- 4) Attach SCL-to-TA calls to your database design by simple replacement of simulated database calls.
- 5) For Slave Libraries, install report-by-exception processing by adding access to a date/time clock, and configuring scan periods and event buffer sizes.
- 6) For Master Libraries, install Target Application request messaging by adding calls to build request message entry points from Target Application software.
- 7) Make final adjustments of the configuration interface to ensure that all user settable configuration parameters are mapped to SCL configuration structures.
- 8) Conduct final testing.

## Testing and Diagnostics

For testing Source Code Library installations, we provide the following tools:

- A **built-in protocol analyzer** allows you to visually decipher protocol messages to or from both Master and Slave devices.

### Sample Protocol Analyzer display

```
15:28:37.506: ...> slave      05 64 14 c4 04 00 03 00 c7 17
15:28:37.506: ...> slave      ca c9 01 3c 02 06 3c 03 06 3c 04 06 3c 01 06 f8
15:28:37.506:                  ae
15:28:37.506: ----> slave      Primary Frame - Unconfirmed User Data
15:28:37.506:                  LEN(20) DIR(1) PRM(1) FCV(0) FCB(0) DEST(4) SRC(3)
15:28:37.506:                  05 64 14 c4 04 00 03 00 c7 17
15:28:37.506:                  ca c9 01 3c 02 06 3c 03 06 3c 04 06 3c 01 06 f8 ae
15:28:37.506: ~~~> slave      Transport Header
15:28:37.506:                  FIR(1) FIN(1) SEQ# 10
15:28:37.506:                  ca c9 01 3c 02 06 3c 03 06 3c 04 06 3c 01 06
15:28:37.506: ==>> slave      Application Header, Read Request
15:28:37.506:                  FIR(1) FIN(1) CON(0) UNS(0) SEQ# 9
15:28:37.506:                  c9 01 3c 02 06 3c 03 06 3c 04 06 3c 01 06
15:28:37.506:                  Object 60(Class Data), variation 2, qualifier 0x06(All Points)
15:28:37.506:                  Object 60(Class Data), variation 3, qualifier 0x06(All Points)
15:28:37.506:                  Object 60(Class Data), variation 4, qualifier 0x06(All Points)
```

- Triangle MicroWorks also offers a Communication Protocol Test Harness to facilitate version release testing of your Source Code Library implementation. The Test Harness acts as a simple Master or Slave device and can also be programmed with automated test sequence scripts. Conformance Test Scripts are also available to perform the conformance test procedures published by the technical committees of each protocol. Please contact us for more information on this product or download a full 21-day evaluation from our website at <http://www.TriangleMicroWorks.com/downloads.htm>.

**2840 PLAZA PLACE, SUITE 205 • RALEIGH, NC 27612 • PH: 919.870.5101**  
**FAX: 919.870.6692 • WEB SITE: [www.TriangleMicroWorks.com](http://www.TriangleMicroWorks.com)**