libmodbus(7)

============

NAME

----

libmodbus - a fast and portable Modbus library

SYNOPSIS

--------

\*#include <modbus.h>\*

\*cc\* \`pkg-config --cflags --libs libmodbus` 'files'

DESCRIPTION

-----------

libmodbus is a library to send/receive data with a device which respects the

Modbus protocol. This library contains various backends to communicate over

different networks (eg. serial in RTU mode or Ethernet in TCP/IPv6). The

http://www.modbus.org site provides documentation about the protocol at

http://www.modbus.org/specs.php.

libmodbus provides an abstraction of the lower communication layers and offers

the same API on all supported platforms.

This documentation presents an overview of libmodbus concepts, describes how

libmodbus abstracts Modbus communication with different hardware and platforms

and provides a reference manual for the functions provided by the libmodbus

library.

Contexts

~~~~~~~~

The Modbus protocol contains many variants (eg. serial RTU or Ehternet TCP), to

ease the implementation of a variant, the library was designed to use a backend

for each variant. The backends are also a convenient way to fulfill other

requirements (eg. real-time operations). Each backend offers a specific function

to create a new 'modbus\_t' context. The 'modbus\_t' context is an opaque

structure containing all necessary information to establish a connection with

others Modbus devices according to the selected variant.

You can choose the best context for your needs among:

RTU Context

^^^^^^^^^^^

The RTU backend (Remote Terminal Unit) is used in serial communication and makes

use of a compact, binary representation of the data for protocol

communication. The RTU format follows the commands/data with a cyclic redundancy

check checksum as an error check mechanism to ensure the reliability of

data. Modbus RTU is the most common implementation available for Modbus. A

Modbus RTU message must be transmitted continuously without inter-character

hesitations (extract from Wikipedia, Modbus, http://en.wikipedia.org/wiki/Modbus

(as of Mar. 13, 2011, 20:51 GMT).

The Modbus RTU framing calls a slave, a device/service which handle Modbus

requests, and a master, a client which send requests. The communication is

always initiated by the master.

Many Modbus devices can be connected together on the same physical link so

before sending a message, you must set the slave (receiver) with

linkmb:modbus\_set\_slave[3]. If you're running a slave, its slave number will be

used to filter received messages.

The libmodbus implementation of RTU isn't time based as stated in original

Modbus specification, instead all bytes are sent as fast as possible and a

response or an indication is considered complete when all expected characters

have been received. This implementation offers very fast communication but you

must take care to set a response timeout of slaves less than response timeout of

master (ortherwise other slaves may ignore master requests when one of the slave

is not responding).

Create a Modbus RTU context::

linkmb:modbus\_new\_rtu[3]

Set the serial mode::

linkmb:modbus\_rtu\_get\_serial\_mode[3]

linkmb:modbus\_rtu\_set\_serial\_mode[3]

linkmb:modbus\_rtu\_get\_rts[3]

linkmb:modbus\_rtu\_set\_rts[3]

linkmb:modbus\_rtu\_set\_custom\_rts[3]

linkmb:modbus\_rtu\_get\_rts\_delay[3]

linkmb:modbus\_rtu\_set\_rts\_delay[3]

TCP (IPv4) Context

^^^^^^^^^^^^^^^^^^

The TCP backend implements a Modbus variant used for communications over

TCP/IPv4 networks. It does not require a checksum calculation as lower layer

takes care of the same.

Create a Modbus TCP context::

linkmb:modbus\_new\_tcp[3]

TCP PI (IPv4 and IPv6) Context

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

The TCP PI (Protocol Independent) backend implements a Modbus variant used for

communications over TCP IPv4 and IPv6 networks. It does not require a checksum

calculation as lower layer takes care of the same.

Contrary to the TCP IPv4 only backend, the TCP PI backend offers hostname

resolution but it consumes about 1Kb of additional memory.

Create a Modbus TCP context::

linkmb:modbus\_new\_tcp\_pi[3]

Common

^^^^^^

Before using any libmodbus functions, the caller must allocate and initialize a

'modbus\_t' context with functions explained above, then the following functions

are provided to modify and free a 'context':

Free libmodbus context::

linkmb:modbus\_free[3]

Set slave ID::

linkmb:modbus\_set\_slave[3]

Enable debug mode::

linkmb:modbus\_set\_debug[3]

Timeout settings::

linkmb:modbus\_get\_byte\_timeout[3]

linkmb:modbus\_set\_byte\_timeout[3]

linkmb:modbus\_get\_response\_timeout[3]

linkmb:modbus\_set\_response\_timeout[3]

Error recovery mode::

linkmb:modbus\_set\_error\_recovery[3]

Setter/getter of internal socket::

linkmb:modbus\_set\_socket[3]

linkmb:modbus\_get\_socket[3]

Information about header::

linkmb:modbus\_get\_header\_length[3]

Macros for data manipulation::

- MODBUS\_GET\_HIGH\_BYTE(data), extracts the high byte from a byte

- MODBUS\_GET\_LOW\_BYTE(data), extracts the low byte from a byte

- MODBUS\_GET\_INT64\_FROM\_INT16(tab\_int16, index), builds an int64 from the four

first int16 starting at tab\_int16[index]

- MODBUS\_GET\_INT32\_FROM\_INT16(tab\_int16, index), builds an int32 from the two

first int16 starting at tab\_int16[index]

- MODBUS\_GET\_INT16\_FROM\_INT8(tab\_int8, index), builds an int16 from the two

first int8 starting at tab\_int8[index]

- MODBUS\_SET\_INT16\_TO\_INT8(tab\_int8, index, value), set an int16 value into

the two first bytes starting at tab\_int8[index]

- MODBUS\_SET\_INT32\_TO\_INT16(tab\_int16, index, value), set an int32 value into

the two first int16 starting at tab\_int16[index]

- MODBUS\_SET\_INT64\_TO\_INT16(tab\_int16, index, value), set an int64 value into

the four first int16 starting at tab\_int16[index]

Handling of bits and bytes::

linkmb:modbus\_set\_bits\_from\_byte[3]

linkmb:modbus\_set\_bits\_from\_bytes[3]

linkmb:modbus\_get\_byte\_from\_bits[3]

Set or get float numbers::

linkmb:modbus\_get\_float\_abcd[3]

linkmb:modbus\_set\_float\_abcd[3]

linkmb:modbus\_get\_float\_badc[3]

linkmb:modbus\_set\_float\_badc[3]

linkmb:modbus\_get\_float\_cdab[3]

linkmb:modbus\_set\_float\_cdab[3]

linkmb:modbus\_get\_float\_dcba[3]

linkmb:modbus\_set\_float\_dcba[3]

linkmb:modbus\_get\_float[3] (deprecated)

linkmb:modbus\_set\_float[3] (deprecated)

Connection

~~~~~~~~~~

The following functions are provided to establish and close a connection with

Modbus devices:

Establish a connection::

linkmb:modbus\_connect[3]

Close a connection::

linkmb:modbus\_close[3]

Flush a connection::

linkmb:modbus\_flush[3]

Client

~~~~~~

The Modbus protocol defines different data types and functions to read and write

them from/to remote devices. The following functions are used by the clients to

send Modbus requests:

Read data::

linkmb:modbus\_read\_bits[3]

linkmb:modbus\_read\_input\_bits[3]

linkmb:modbus\_read\_registers[3]

linkmb:modbus\_read\_input\_registers[3]

linkmb:modbus\_report\_slave\_id[3]

Write data::

linkmb:modbus\_write\_bit[3]

linkmb:modbus\_write\_register[3]

linkmb:modbus\_write\_bits[3]

linkmb:modbus\_write\_registers[3]

Write and read data::

linkmb:modbus\_write\_and\_read\_registers[3]

Raw requests::

linkmb:modbus\_send\_raw\_request[3]

linkmb:modbus\_receive\_confirmation[3]

Reply an exception::

linkmb:modbus\_reply\_exception[3]

Server

~~~~~~

The server is waiting for request from clients and must answer when it is

concerned by the request. The libmodbus offers the following functions to

handle requests:

Data mapping:

linkmb:modbus\_mapping\_new[3]

linkmb:modbus\_mapping\_free[3]

Receive::

linkmb:modbus\_receive[3]

Reply::

linkmb:modbus\_reply[3]

linkmb:modbus\_reply\_exception[3]

ERROR HANDLING

--------------

The libmodbus functions handle errors using the standard conventions found on

POSIX systems. Generally, this means that upon failure a libmodbus function

shall return either a NULL value (if returning a pointer) or a negative value

(if returning an integer), and the actual error code shall be stored in the

'errno' variable.

The \*modbus\_strerror()\* function is provided to translate libmodbus-specific

error codes into error message strings; for details refer to

linkmb:modbus\_strerror[3].

MISCELLANEOUS

-------------

The \_LIBMODBUS\_VERSION\_STRING\_ constant indicates the libmodbus version the

program has been compiled against. The variables 'libmodbus\_version\_major',

'libmodbus\_version\_minor', 'libmodbus\_version\_micro' give the version the

program is linked against.

AUTHORS

-------

The libmodbus documentation was written by Stéphane Raimbault

<stephane.raimbault@gmail.com>

RESOURCES

---------

Main web site: <http://www.libmodbus.org/>

Report bugs on the issue tracker at

<http://github.com/stephane/libmodbus/issues>.

COPYING

-------

Free use of this software is granted under the terms of the GNU Lesser General

Public License (LGPL v2.1+). For details see the file `COPYING.LESSER` included

with the libmodbus distribution.