TITLE

AUTHOR
Version 1.2
CREATEDATE

Table of Contents

Table of contents

Todo List

Member Modbus::query (modbus_t telegram)
finish function 15

Module Index

Modules

Here is a list of all modules:

Class Index

Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

File Index

File List

Here is a list of all files with brief descriptions:

Module Documentation

Modbus Object Instantiation/Initialization

Functions

- **Modbus::Modbus** ()
  Default Constructor for Master through Serial.
- void **Modbus::begin** (long u32speed)
  Initialize class object.
- void **Modbus::setID** (uint8_t u8id)
  write new ID for the slave
- uint8_t **Modbus::getID** ()
  get slave ID between 1 and 247
- void **Modbus::setTimeOut** (uint16_t u16timeout)
  write communication watch-dog timer

Detailed Description

Function Documentation

void Modbus::begin (long *u32speed*)

Initialize class object.

Sets up the serial port using specified baud rate. Call once class has been instantiated, typically within setup().

See Also:

`http://arduino.cc/en/Serial/Begin#.Uy4CJ6aKlHY`

Parameters:

speed	baud rate, in standard increments (300..115200)
config	data frame settings (data length, parity and stop bits)

Definition at line 250 of file ModbusRtu.h.

uint8_t Modbus::getID ()

get slave ID between 1 and 247

Method to read current slave ID address.

Returns:

u8id current slave address between 1 and 247

Definition at line 323 of file ModbusRtu.h.

Modbus::Modbus ()

Default Constructor for Master through Serial.

Definition at line 204 of file ModbusRtu.h.

void Modbus::setID (uint8_t *u8id*)

write new ID for the slave

Method to write a new slave ID address.

Parameters:

u8id	new slave address between 1 and 247

Definition at line 310 of file ModbusRtu.h.

void Modbus::setTimeOut (uint16_t *u16timeOut*)

write communication watch-dog timer

Initialize time-out parameter.

Call once class has been instantiated, typically within setup(). The time-out timer is reset each time that there is a successful communication between Master and Slave. It works for both.

Parameters:

time-out	value (ms)

Definition at line 338 of file ModbusRtu.h.

Modbus Object Management

Functions

- boolean **Modbus::getTimeOutState** ()
  get communication watch-dog timer state
- int8_t **Modbus::query** (**modbus_t** telegram)
  only for master
- int8_t **Modbus::poll** ()
  cyclic poll for master
- int8_t **Modbus::poll** (uint16_t *regs, uint8_t u8size)
  cyclic poll for slave

Detailed Description

Function Documentation

boolean Modbus::getTimeOutState ()

get communication watch-dog timer state

Return communication Watchdog state. It could be usefull to reset outputs if the watchdog is fired.

Returns:

TRUE if millis() > u32timeOut

Definition at line 350 of file ModbusRtu.h.

int8_t Modbus::poll ()

cyclic poll for master

*** Only for **Modbus** Master *** This method checks if there is any incoming answer if pending. If there is no answer, it would change Master state to COM_IDLE. This method must be called only at loop section. Avoid any delay() function.

Any incoming data would be redirected to au16regs pointer, as defined in its **modbus_t** query telegram.

nothing

Returns:

errors counter

Definition at line 513 of file ModbusRtu.h.

int8_t Modbus::poll (uint16_t * *regs*, uint8_t *u8size*)

cyclic poll for slave

*** Only for **Modbus** Slave *** This method checks if there is any incoming query Afterwards, it would shoot a validation routine plus a register query Avoid any delay() function !!!! After a successful frame between the Master and the Slave, the time-out timer is reset.

Parameters:

**regs*	register table for communication exchange
u8size	size of the register table

Returns:

0 if no query, 1..4 if communication error, >4 if correct query processed

Definition at line 588 of file ModbusRtu.h.

int8_t Modbus::query (modbus_t *telegram*)

only for master

*** Only **Modbus** Master *** Generate a query to an slave with a **modbus_t** telegram structure The Master must be in COM_IDLE mode. After it, its state would be COM_WAITING. This method has to be called only in loop() section.

See Also:

modbus_t

Parameters:

modbus_t	modbus telegram structure (id, fct, ...)

Todo:

finish function 15

Definition at line 425 of file ModbusRtu.h.

Modbus Buffer Management

Functions

- uint16_t **Modbus::getInCnt** ()
  number of incoming messages
- uint16_t **Modbus::getOutCnt** ()
  number of outcoming messages
- uint16_t **Modbus::getErrCnt** ()
  error counter
- uint8_t **Modbus::getState** ()
- uint8_t **Modbus::getLastError** ()
  get last error message

Detailed Description

Function Documentation

uint16_t Modbus::getErrCnt ()

error counter

Get errors counter value This can be useful to diagnose communication.

Returns:

errors counter

Definition at line 386 of file ModbusRtu.h.

uint16_t Modbus::getInCnt ()

number of incoming messages

Get input messages counter value This can be useful to diagnose communication.

Returns:

input messages counter

Definition at line 362 of file ModbusRtu.h.

uint8_t Modbus::getLastError ()

get last error message

Get the last error in the protocol processor

NO_REPLY = 255 Time-out

Returns:

EXC_FUNC_CODE = 1 Function code not available
EXC_ADDR_RANGE = 2 Address beyond available space for **Modbus** registers
EXC_REGS_QUANT = 3 Coils or registers number beyond the available space

Definition at line 409 of file ModbusRtu.h.

uint16_t Modbus::getOutCnt ()

number of outcoming messages

Get transmitted messages counter value This can be useful to diagnose communication.

Returns:

transmitted messages counter

Definition at line 374 of file ModbusRtu.h.

uint8_t Modbus::getState ()

Get modbus master state

Returns:

= 0 IDLE, = 1 WAITING FOR ANSWER

Definition at line 396 of file ModbusRtu.h.

Modbus Function Codes for Discrete Coils/Inputs

Detailed Description

Modbus Function Codes for Holding/Input Registers

Detailed Description

Class Documentation

Modbus Class Reference

Arduino class library for communicating with **Modbus** devices over USB/RS232/485 (via RTU protocol).
`#include <ModbusRtu.h>`

Public Member Functions

- **Modbus** ()
  Default Constructor for Master through Serial.
- **Modbus** (uint8_t u8id, uint8_t u8serno)
- **Modbus** (uint8_t u8id, uint8_t u8serno, uint8_t u8txenpin)
- void **begin** (long u32speed)
  Initialize class object.
- void **begin** ()
- void **setTimeOut** (uint16_t u16timeout)
  write communication watch-dog timer
- uint16_t **getTimeOut** ()
  get communication watch-dog timer value
- boolean **getTimeOutState** ()
  get communication watch-dog timer state
- int8_t **query** (**modbus_t** telegram)
  only for master
- int8_t **poll** ()
  cyclic poll for master
- int8_t **poll** (uint16_t *regs, uint8_t u8size)
  cyclic poll for slave
- uint16_t **getInCnt** ()
  number of incoming messages
- uint16_t **getOutCnt** ()
  number of outcoming messages
- uint16_t **getErrCnt** ()
  error counter
- uint8_t **getID** ()
  get slave ID between 1 and 247
- uint8_t **getState** ()
- uint8_t **getLastError** ()
  get last error message
- void **setID** (uint8_t u8id)
  write new ID for the slave
- void **end** ()
  finish any communication and release serial communication port

Detailed Description

Arduino class library for communicating with **Modbus** devices over USB/RS232/485 (via RTU protocol).

Definition at line 141 of file ModbusRtu.h.

Constructor & Destructor Documentation

Modbus::Modbus (uint8_t *u8id*, uint8_t *u8serno*)

Definition at line 218 of file ModbusRtu.h.

Modbus::Modbus (uint8_t *u8id*, uint8_t *u8serno*, uint8_t *u8txenpin*)

Definition at line 234 of file ModbusRtu.h.

Member Function Documentation

void Modbus::begin ()

Definition at line 299 of file ModbusRtu.h.

void Modbus::end ()

finish any communication and release serial communication port

uint16_t Modbus::getTimeOut ()

get communication watch-dog timer value

The documentation for this class was generated from the following file:

- **ModbusRtu.h**

modbus_t Struct Reference

Master query structure: This includes all the necessary fields to make the Master generate a **Modbus** query. A Master may keep several of these structures and send them cyclically or use them according to program needs.

```
#include <ModbusRtu.h>
```

Public Attributes

- uint8_t **u8id**
- uint8_t **u8fct**
- uint16_t **u16RegAdd**
- uint16_t **u16CoilsNo**
- uint16_t * **au16reg**

Detailed Description

Master query structure: This includes all the necessary fields to make the Master generate a **Modbus** query. A Master may keep several of these structures and send them cyclically or use them according to program needs.

Definition at line 48 of file ModbusRtu.h.

Member Data Documentation

uint16_t* modbus_t::au16reg

Pointer to memory image in master

Definition at line 53 of file ModbusRtu.h.

uint16_t modbus_t::u16CoilsNo

Number of coils or registers to access

Definition at line 52 of file ModbusRtu.h.

uint16_t modbus_t::u16RegAdd

Address of the first register to access at slave/s

Definition at line 51 of file ModbusRtu.h.

uint8_t modbus_t::u8fct

Function code: 1, 2, 3, 4, 5, 6, 15 or 16

Definition at line 50 of file ModbusRtu.h.

uint8_t modbus_t::u8id

Slave address between 1 and 247. 0 means broadcast

Definition at line 49 of file ModbusRtu.h.

The documentation for this struct was generated from the following file:

- **ModbusRtu.h**

File Documentation

ModbusRtu.h File Reference

Classes

- struct **modbus_t**
- *Master query structure: This includes all the necessary fields to make the Master generate a **Modbus** query. A Master may keep several of these structures and send them cyclically or use them according to program needs.* class **Modbus**

Arduino class library for communicating with Modbus devices over USB/RS232/485 (via RTU protocol). Macros

- #define **T35** 5
- #define **MAX_BUFFER** 64
  maximum size for the communication buffer in bytes

Enumerations

- enum { **RESPONSE_SIZE** = 6, **EXCEPTION_SIZE** = 3, **CHECKSUM_SIZE** = 2 }
- enum **MESSAGE** { **ID** = 0, **FUNC**, **ADD_HI**, **ADD_LO**, **NB_HI**, **NB_LO**, **BYTE_CNT** }
- *Indexes to telegram frame positions.* enum **MB_FC** { **MB_FC_NONE** = 0, **MB_FC_READ_COILS** = 1, **MB_FC_READ_DISCRETE_INPUT** = 2, **MB_FC_READ_REGISTERS** = 3, **MB_FC_READ_INPUT_REGISTER** = 4, **MB_FC_WRITE_COIL** = 5, **MB_FC_WRITE_REGISTER** = 6, **MB_FC_WRITE_MULTIPLE_COILS** = 15, **MB_FC_WRITE_MULTIPLE_REGISTERS** = 16 }
- ***Modbus** function codes summary. These are the implement function codes either for Master or for Slave.* enum **COM_STATES** { **COM_IDLE** = 0, **COM_WAITING** = 1 }
- enum **ERR_LIST** { **ERR_NOT_MASTER** = -1, **ERR_POLLING** = -2, **ERR_BUFF_OVERFLOW** = -3, **ERR_BAD_CRC** = -4, **ERR_EXCEPTION** = -5 }
- enum { **NO_REPLY** = 255, **EXC_FUNC_CODE** = 1, **EXC_ADDR_RANGE** = 2, **EXC_REGS_QUANT** = 3, **EXC_EXECUTE** = 4 }

Variables

- const unsigned char **fctsupported** []

Detailed Description

Version:
1.2

Date:
2014.09.09

Author:
Samuel Marco i Armengol `sammarcoarmengol@gmail.com`

Arduino library for communicating with **Modbus** devices over RS232/USB/485 via RTU protocol.

Further information: `http://modbus.org/` `http://modbus.org/docs/Modbus_over_serial_line_V1_02.pdf`

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Definition in file **ModbusRtu.h**.

Macro Definition Documentation

#define MAX_BUFFER 64

maximum size for the communication buffer in bytes

Definition at line 133 of file ModbusRtu.h.

#define T35 5

Definition at line 132 of file ModbusRtu.h.

Enumeration Type Documentation

anonymous enum

Enumerator

RESPONSE_SIZE
EXCEPTION_SIZE
CHECKSUM_SIZE

Definition at line 57 of file ModbusRtu.h.

anonymous enum

Enumerator

NO_REPLY
EXC_FUNC_CODE
EXC_ADDR_RANGE
EXC_REGS_QUANT
EXC_EXECUTE

Definition at line 113 of file ModbusRtu.h.

enum COM_STATES

Enumerator

COM_IDLE
COM_WAITING

Definition at line 99 of file ModbusRtu.h.

enum ERR_LIST

Enumerator

ERR_NOT_MASTER
ERR_POLLING
ERR_BUFF_OVERFLOW
ERR_BAD_CRC
ERR_EXCEPTION

Definition at line 105 of file ModbusRtu.h.

enum MB_FC

Modbus function codes summary. These are the implement function codes either for Master or for Slave.

See Also:

also **fctsupported**
also **modbus_t**

Enumerator

MB_FC_NONE null operator

MB_FC_READ_COILS FCT=1 -> read coils or digital outputs

MB_FC_READ_DISCRETE_INPUT FCT=2 -> read digital inputs

MB_FC_READ_REGISTERS FCT=3 -> read registers or analog outputs

MB_FC_READ_INPUT_REGISTER FCT=4 -> read analog inputs

MB_FC_WRITE_COIL FCT=5 -> write single coil or output

MB_FC_WRITE_REGISTER FCT=6 -> write single register

MB_FC_WRITE_MULTIPLE_COILS FCT=15 -> write multiple coils or outputs

MB_FC_WRITE_MULTIPLE_REGISTERS FCT=16 -> write multiple registers

Definition at line 87 of file ModbusRtu.h.

enum MESSAGE

Indexes to telegram frame positions.

Enumerator

ID ID field.

FUNC Function code position.

ADD_HI Address high byte.

ADD_LO Address low byte.

NB_HI Number of coils or registers high byte.

NB_LO Number of coils or registers low byte.

BYTE_CNT byte counter

Definition at line 68 of file ModbusRtu.h.

Variable Documentation

const unsigned char fctsupported[]

```
Initial value:= {
  MB_FC_READ_COILS,
  MB_FC_READ_DISCRETE_INPUT,
  MB_FC_READ_REGISTERS,
  MB_FC_READ_INPUT_REGISTER,
  MB_FC_WRITE_COIL,
  MB_FC_WRITE_REGISTER,
  MB_FC_WRITE_MULTIPLE_COILS,
  MB_FC_WRITE_MULTIPLE_REGISTERS
}
```

Definition at line 121 of file ModbusRtu.h.

Index

INDEX