

DFRobot / DFRobot_RTU Public

MIT license

11 stars 7 forks Branches Tags Activity

Star

Notifications

<> Code

Issues

Pull requests

Actions

Projects

Security

Insights

master 1 Branch 6 Tags

Go to file

Go to file

Code

Arya11111 解决0数组在esp32板上, 如果将arduino IDE编译警告设置为严格(所有), RTU编译报错数组... 6 months ago

examples	解决0数组在esp32板上, 如果将arduino IDE编...	6 months ago
python/raspberrypi	修改readme表格格式, 将 改成空格加换行, ...	3 years ago
src	解决0数组在esp32板上, 如果将arduino IDE编...	6 months ago
LICENCE	V1.0	3 years ago
README.md	Update README.md	2 years ago
README_CN.md	add depin notes	2 years ago
keywords.txt	V1.0	3 years ago
library.properties	Solve the rs485 delay problem	8 months ago

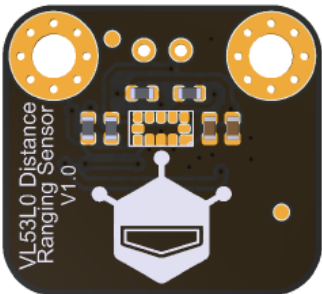
README MIT license

DFRobot_RTU

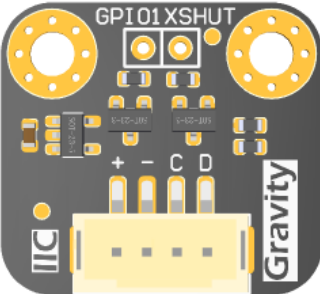
- [中文版](#)

Modbus RTU library for Arduino. The supported modbus commands are as follows :

- 0x01: Read one or multiple coils register;
- 0x02: Read one or multiple discrete inputs register;
- 0x03: Read one or multiple holding register;
- 0x04: Read one or multiple input register;
- 0x05: Write a coils register;
- 0x06: write a holding register;
- 0x0F: Write multiple coils register;
- 0x10: Write multiple holding register;



Front



Back

Product Link (链接到英文商城)

Table of Contents

- [Summary](#)
- [Connected](#)
- [Installation](#)
- [Calibration](#)
- [Methods](#)
- [Compatibility](#)
- [History](#)
- [Credits](#)

Summary

This is a modbus RTU library for Arduino by DFRobot.

Connected

Hardware conneted table

Sensor	MCU
VCC	5V
GND	GND
RX	connected to the UART TX pin of MCU
TX	connected to the UART RX pin of MCU

Installation

To use this library, first download the library file, paste it into the \Arduino\libraries directory, then open the examples folder and run the demo in the folder.

Methods

```
/**
 * @brief DFRobot_RTU abstract class constructor. Construct serial port.
 * @param s: The class pointer object of Abstract class, here you can fill in the pointer to the serial port object.
 * @param dePin: RS485 flow control, pull low to receive, pull high to send.
 */
DFRobot_RTU(Stream *s,int dePin);
DFRobot_RTU(Stream *s);
~DFRobot_RTU()

/**
 * @brief Set receive timeout time, unit ms.
 * @param timeout: receive timeout time, unit ms, default 100mss.
 */
void setTimeoutTimeMs(uint32_t timeout = 100);

/**
 * @brief Read a coils Register.
 * @param id: modbus device ID. Range: 0x00 ~ 0xF7(0~247), 0x00 is broadcast address, which all slaves will process broadcast |
 * @n but will not answer.
 * @param reg: Coils register address.
 * @return Return the value of the coils register value.
 * @n true: The value of the coils register value is 1.
 * @n false: The value of the coils register value is 0.
 */
```

```

    bool readCoilsRegister(uint8_t id, uint16_t reg);

/**
 * @brief Read a discrete input register.
 * @param id: modbus device ID. Range: 0x00 ~ 0xF7(0~247), 0x00 is broadcast address, which all slaves will process broadcast |
 * @n          but will not answer.
 * @param reg: Discrete input register address.
 * @return Return the value of the discrete input register.
 */
uint16_t readDiscreteInputsRegister(uint8_t id, uint16_t reg);

/**
 * @brief Read a holding Register.
 * @param id: modbus device ID. Range: 0x00 ~ 0xF7(0~247), 0x00 is broadcast address, which all slaves will process broadcast |
 * @n          but will not answer.
 * @param reg: Holding register address.
 * @return Return the value of the holding register value.
 */
uint16_t readHoldingRegister(uint8_t id, uint16_t reg);

/**
 * @brief Read a input Register.
 * @param id: modbus device ID. Range: 0x00 ~ 0xF7(0~247), 0x00 is broadcast address, which all slaves will process broadcast |
 * @n          but will not answer.
 * @param reg: input register address.
 * @return Return the value of the input register value.
 */
uint16_t readInputRegister(uint8_t id, uint16_t reg);

/**
 * @brief Write a coils Register.
 * @param id: modbus device ID. Range: 0x00 ~ 0xF7(0~247), 0x00 is broadcast address, which all slaves will process broadcast |
 * @n          but will not answer.
 * @param reg: Coils register address.
 * @param flag: The value of the register value which will be write, 0 ro 1.
 * @return Exception code:
 * @n      0 : sucess.
 * @n      1 or eRTU_EXCEPTION_ILLEGAL_FUNCTION : Illegal function.
 * @n      2 or eRTU_EXCEPTION_ILLEGAL_DATA_ADDRESS: Illegal data address.
 * @n      3 or eRTU_EXCEPTION_ILLEGAL_DATA_VALUE: Illegal data value.
 * @n      4 or eRTU_EXCEPTION_SLAVE_FAILURE: Slave failure.
 * @n      8 or eRTU_EXCEPTION_CRC_ERROR: CRC check error.
 * @n      9 or eRTU_RECV_ERROR: Receive packet error.
 * @n      10 or eRTU_MEMORY_ERROR: Memory error.
 * @n      11 or eRTU_ID_ERROR: Broadcast address or error ID
 */
uint8_t writeCoilsRegister(uint8_t id, uint16_t reg, bool flag);

/**
 * @brief Write a holding register.
 * @param id: modbus device ID. Range: 0x00 ~ 0xF7(0~247), 0x00 is broadcast address, which all slaves will process broadcast |
 * @n          but will not answer.
 * @param reg: Holding register address.
 * @return Exception code:
 * @n      0 : sucess.
 * @n      1 or eRTU_EXCEPTION_ILLEGAL_FUNCTION : Illegal function.
 * @n      2 or eRTU_EXCEPTION_ILLEGAL_DATA_ADDRESS: Illegal data address.
 * @n      3 or eRTU_EXCEPTION_ILLEGAL_DATA_VALUE: Illegal data value.
 * @n      4 or eRTU_EXCEPTION_SLAVE_FAILURE: Slave failure.
 * @n      8 or eRTU_EXCEPTION_CRC_ERROR: CRC check error.
 * @n      9 or eRTU_RECV_ERROR: Receive packet error.
 * @n      10 or eRTU_MEMORY_ERROR: Memory error.
 * @n      11 or eRTU_ID_ERROR: Broadcast address or error ID
 */
uint8_t writeHoldingRegister(uint8_t id, uint16_t reg, uint16_t val);

/**
 * @brief Read multiple coils Register.
 * @param id: modbus device ID. Range: 0x00 ~ 0xF7(0~247), 0x00 is broadcast address, which all slaves will process broadcast |
 * @n          but will not answer.
 * @param reg: Coils register address.
 * @param regNum: Number of coils Register.
 * @param data: Storage register worth pointer.

```

```

* @param size: Cache size of data.
* @return Exception code:
* @n      0 : sucess.
* @n      1 or eRTU_EXCEPTION_ILLEGAL_FUNCTION : Illegal function.
* @n      2 or eRTU_EXCEPTION_ILLEGAL_DATA_ADDRESS: Illegal data address.
* @n      3 or eRTU_EXCEPTION_ILLEGAL_DATA_VALUE: Illegal data value.
* @n      4 or eRTU_EXCEPTION_SLAVE_FAILURE: Slave failure.
* @n      8 or eRTU_EXCEPTION_CRC_ERROR: CRC check error.
* @n      9 or eRTU_RECV_ERROR: Receive packet error.
* @n     10 or eRTU_MEMORY_ERROR: Memory error.
* @n     11 or eRTU_ID_ERROR: Broadcast address or error ID
*/
uint8_t readCoilsRegister(uint8_t id, uint16_t reg, uint16_t regNum, uint8_t *data, uint16_t size);

/**
* @brief Read multiple discrete inputs register.
* @param id: modbus device ID. Range: 0x00 ~ 0xF7(0~247), 0x00 is broadcast address, which all slaves will process broadcast but will not answer.
* @param reg: Discrete inputs register. address.
* @param regNum: Number of coils Register.
* @param data: Storage register worth pointer.
* @param size: Cache size.
* @return Exception code:
* @n      0 : sucess.
* @n      1 or eRTU_EXCEPTION_ILLEGAL_FUNCTION : Illegal function.
* @n      2 or eRTU_EXCEPTION_ILLEGAL_DATA_ADDRESS: Illegal data address.
* @n      3 or eRTU_EXCEPTION_ILLEGAL_DATA_VALUE: Illegal data value.
* @n      4 or eRTU_EXCEPTION_SLAVE_FAILURE: Slave failure.
* @n      8 or eRTU_EXCEPTION_CRC_ERROR: CRC check error.
* @n      9 or eRTU_RECV_ERROR: Receive packet error.
* @n     10 or eRTU_MEMORY_ERROR: Memory error.
* @n     11 or eRTU_ID_ERROR: Broadcast address or error ID
*/
uint8_t readDiscreteInputsRegister(uint8_t id, uint16_t reg, uint16_t regNum, uint8_t *data, uint16_t size);

/**
* @brief Read multiple Holding register.
* @param id: modbus device ID. Range: 0x00 ~ 0xF7(0~247), 0x00 is broadcast address, which all slaves will process broadcast but will not answer.
* @param reg: Holding register.
* @param data: Storage register worth pointer.
* @param size: Cache size.
* @return Exception code:
* @n      0 : sucess.
* @n      1 or eRTU_EXCEPTION_ILLEGAL_FUNCTION : Illegal function.
* @n      2 or eRTU_EXCEPTION_ILLEGAL_DATA_ADDRESS: Illegal data address.
* @n      3 or eRTU_EXCEPTION_ILLEGAL_DATA_VALUE: Illegal data value.
* @n      4 or eRTU_EXCEPTION_SLAVE_FAILURE: Slave failure.
* @n      8 or eRTU_EXCEPTION_CRC_ERROR: CRC check error.
* @n      9 or eRTU_RECV_ERROR: Receive packet error.
* @n     10 or eRTU_MEMORY_ERROR: Memory error.
* @n     11 or eRTU_ID_ERROR: Broadcast address or error ID
*/
uint8_t readHoldingRegister(uint8_t id, uint16_t reg, void *data, uint16_t size);

/**
* @brief Read multiple Input register.
* @param id: modbus device ID. Range: 0x00 ~ 0xF7(0~247), 0x00 is broadcast address, which all slaves will process broadcast but will not answer.
* @param reg: Input register.
* @param data: Storage register worth pointer.
* @param regNum: register numbers.
* @return Exception code:
* @n      0 : sucess.
* @n      1 or eRTU_EXCEPTION_ILLEGAL_FUNCTION : Illegal function.
* @n      2 or eRTU_EXCEPTION_ILLEGAL_DATA_ADDRESS: Illegal data address.
* @n      3 or eRTU_EXCEPTION_ILLEGAL_DATA_VALUE: Illegal data value.
* @n      4 or eRTU_EXCEPTION_SLAVE_FAILURE: Slave failure.
* @n      8 or eRTU_EXCEPTION_CRC_ERROR: CRC check error.
* @n      9 or eRTU_RECV_ERROR: Receive packet error.
* @n     10 or eRTU_MEMORY_ERROR: Memory error.
* @n     11 or eRTU_ID_ERROR: Broadcast address or error ID
*/

```

```

uint8_t readInputRegister(uint8_t id, uint16_t reg, void *data, uint16_t size);

/**
 * @brief Read multiple Holding register.
 * @param id: modbus device ID. Range: 0x00 ~ 0xF7(0~247), 0x00 is broadcast address, which all slaves will process broadcast
 * @n but will not answer.
 * @param reg: Holding register.
 * @param data: Storage register worth pointer.
 * @param regNum: register numbers.
 * @return Exception code:
 * @n 0 : sucess.
 * @n 1 or eRTU_EXCEPTION_ILLEGAL_FUNCTION : Illegal function.
 * @n 2 or eRTU_EXCEPTION_ILLEGAL_DATA_ADDRESS: Illegal data address.
 * @n 3 or eRTU_EXCEPTION_ILLEGAL_DATA_VALUE: Illegal data value.
 * @n 4 or eRTU_EXCEPTION_SLAVE_FAILURE: Slave failure.
 * @n 8 or eRTU_EXCEPTION_CRC_ERROR: CRC check error.
 * @n 9 or eRTU_RECV_ERROR: Receive packet error.
 * @n 10 or eRTU_MEMORY_ERROR: Memory error.
 * @n 11 or eRTU_ID_ERROR: Broadcast address or error ID
 */
uint8_t readHoldingRegister(uint8_t id, uint16_t reg, uint16_t *data, uint16_t regNum);

/**
 * @brief Read multiple Input register.
 * @param id: modbus device ID. Range: 0x00 ~ 0xF7(0~247), 0x00 is broadcast address, which all slaves will process broadcast
 * @n but will not answer.
 * @param reg: Input register.
 * @param data: Storage register worth pointer.
 * @param regNum: register numbers.
 * @return Exception code:
 * @n 0 : sucess.
 * @n 1 or eRTU_EXCEPTION_ILLEGAL_FUNCTION : Illegal function.
 * @n 2 or eRTU_EXCEPTION_ILLEGAL_DATA_ADDRESS: Illegal data address.
 * @n 3 or eRTU_EXCEPTION_ILLEGAL_DATA_VALUE: Illegal data value.
 * @n 4 or eRTU_EXCEPTION_SLAVE_FAILURE: Slave failure.
 * @n 8 or eRTU_EXCEPTION_CRC_ERROR: CRC check error.
 * @n 9 or eRTU_RECV_ERROR: Receive packet error.
 * @n 10 or eRTU_MEMORY_ERROR: Memory error.
 * @n 11 or eRTU_ID_ERROR: Broadcast address or error ID
 */
uint8_t DFRobot_RTU::readInputRegister(uint8_t id, uint16_t reg, uint16_t *data, uint16_t size)

/**
 * @brief Write multiple coils Register.
 * @param id: modbus device ID. Range: 0x00 ~ 0xF7(0~247), 0x00 is broadcast address, which all slaves will process broadcast
 * @n but will not answer.
 * @param reg: Coils register address.
 * @param regNum: Numbers of Coils register.
 * @param data: Storage register worth pointer.
 * @param size: Cache size.
 * @return Exception code:
 * @n 0 : sucess.
 * @n 1 or eRTU_EXCEPTION_ILLEGAL_FUNCTION : Illegal function.
 * @n 2 or eRTU_EXCEPTION_ILLEGAL_DATA_ADDRESS: Illegal data address.
 * @n 3 or eRTU_EXCEPTION_ILLEGAL_DATA_VALUE: Illegal data value.
 * @n 4 or eRTU_EXCEPTION_SLAVE_FAILURE: Slave failure.
 * @n 8 or eRTU_EXCEPTION_CRC_ERROR: CRC check error.
 * @n 9 or eRTU_RECV_ERROR: Receive packet error.
 * @n 10 or eRTU_MEMORY_ERROR: Memory error.
 * @n 11 or eRTU_ID_ERROR: Broadcast address or error ID
 */
uint8_t writeCoilsRegister(uint8_t id, uint16_t reg, uint16_t regNum, void *data, uint16_t size);

/**
 * @brief Write multiple Holding Register.
 * @param id: modbus device ID. Range: 0x00 ~ 0xF7(0~247), 0x00 is broadcast address, which all slaves will process broadcast
 * @n but will not answer.
 * @param reg: Holding register address.
 * @param data: Storage register worth pointer.
 * @param size: Cache size.
 * @return Exception code:
 * @n 0 : sucess.
 * @n 1 or eRTU_EXCEPTION_ILLEGAL_FUNCTION : Illegal function.

```

```
* @n      2 or eRTU_EXCEPTION_ILLEGAL_DATA_ADDRESS: Illegal data address.
* @n      3 or eRTU_EXCEPTION_ILLEGAL_DATA_VALUE:  Illegal data value.
* @n      4 or eRTU_EXCEPTION_SLAVE_FAILURE:  Slave failure.
* @n      8 or eRTU_EXCEPTION_CRC_ERROR:   CRC check error.
* @n      9 or eRTU_RECV_ERROR:   Receive packet error.
* @n     10 or eRTU_MEMORY_ERROR: Memory error.
* @n     11 or eRTU_ID_ERROR: Broadcastr address or error ID
*/
uint8_t writeHoldingRegister(uint8_t id, uint16_t reg, void *data, uint16_t size);

/**
 * @brief Write multiple Holding Register.
 * @param id: modbus device ID. Range: 0x00 ~ 0xF7(0~247), 0x00 is broadcast address, which all slaves will process broadcast |
 * @n      but will not answer.
 * @param reg: Holding register address.
 * @param data: Storage register worth pointer.
 * @param regNum: Number of coils Register..
 * @return Exception code:
 * @n      0 : sucess.
 * @n      1 or eRTU_EXCEPTION_ILLEGAL_FUNCTION : Illegal function.
 * @n      2 or eRTU_EXCEPTION_ILLEGAL_DATA_ADDRESS: Illegal data address.
 * @n      3 or eRTU_EXCEPTION_ILLEGAL_DATA_VALUE:  Illegal data value.
 * @n      4 or eRTU_EXCEPTION_SLAVE_FAILURE:  Slave failure.
 * @n      8 or eRTU_EXCEPTION_CRC_ERROR:   CRC check error.
 * @n      9 or eRTU_RECV_ERROR:   Receive packet error.
 * @n     10 or eRTU_MEMORY_ERROR: Memory error.
 * @n     11 or eRTU_ID_ERROR: Broadcastr address or error ID
 */
uint8_t writeHoldingRegister(uint8_t id, uint16_t reg, uint16_t *data, uint16_t regNum);
```

Compatibility

MCU	SoftwareSerial	HardwareSerial
Arduino Uno	√	X
Mega2560	√	√
Leonardo	√	√
ESP32	X	√
ESP8266	√	X
micro:bit	X	X
FireBeetle M0	X	√
raspberry	X	√

History

- Data 2021-07-17
- Version V1.0

Credits

Written by(xue.peng@dfrobot.com), 2021. (Welcome to our [website](#))

Releases

🏷 6 tags

Packages

No packages published

Contributors 5



Languages

