


原

Modbus学习总结

2017年09月08日 11:41:01 byxdaz 阅读数：22600

 版权声明：本文为博主原创文章，未经博主允许不得转载。 <https://blog.csdn.net/byxdaz/article/details/77892778>

一、介绍

Modbus 协议是应用于电子控制器上的一种通用语言。通过此协议，控制器相互之间、控制器经由网络（例如以太网）和其它设备之间可以通信。Modbus协议定义了控制器能认识使用的消息结构,而不管它们是经过何种网络进行通信的。它描述了一控制器请求访问其它设备的过程，如果回应来自其它设备的请求，以及怎样侦测错误并记录。它制定了

Modbus 是一个请求/应答协议。也叫做Slave和Master与Server和Client。

同一种设备在不同领域的不同叫法。

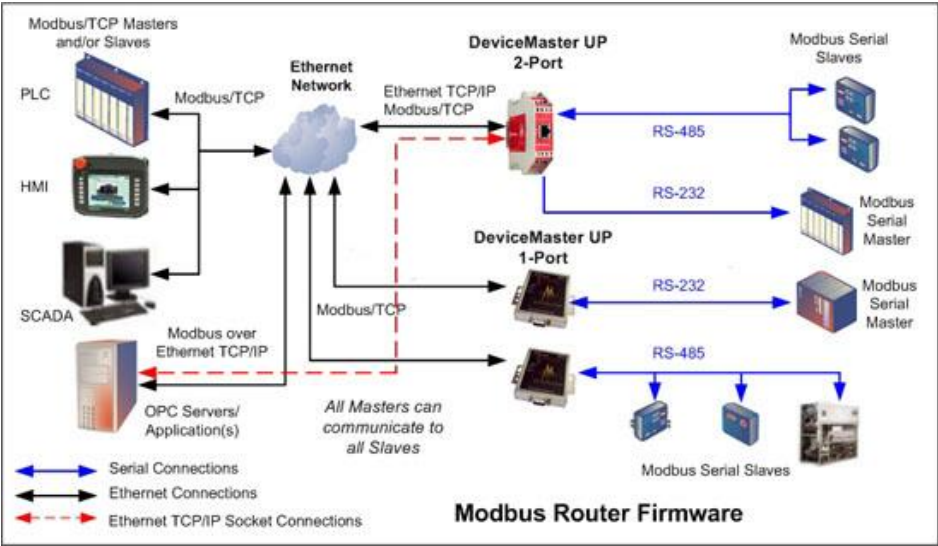
Slave：工业自动化用语；响应请求；

Master：工业自动化用语；发送请求；

Server：IT用语；响应请求；

Client：IT用语；发送请求；

在Modbus中，Slave和Server意思相同，Master和Client意思相同。



modbus结构示意图

二、协议分类

ModBus协议是应用层报文传输协议（OSI模型第7层），它定义了一个与通信层无关的协议数据单元（PDU），即PDU=功能码+数据域。

ModBus协议能够应用在不同类型的总线或网络。对应不同的总线或网络，Modbus协议引入一些附加域映射成应用数据单元（ADU），即ADU=附加域+PDU。目前，

Modbus有下列三种通信方式：

- 1.以太网，对应的通信模式是MODBUS TCP。
- 2.异步串行传输（各种介质如有线RS-232-/422/485/；光纤、无线等），对应的通信模式是MODBUS RTU或MODBUS ASCII。
- 3.高速令牌传递网络，对应的通信模式是Modbus PLUS。

ASCII模式

RTU模式

| | | | | | | | | |
|----|------|------|-----|-----|-----|--------|---|----|
| 地址 | 功能代码 | 数据数量 | 数据1 | ... | 数据n | CRC高字节 | (| 字节 |
|----|------|------|-----|-----|-----|--------|---|----|

所选的ASCII或RTU方式仅适用于标准的Modbus网络，它定义了在这些网络上连续传输的消息段的每一位，以及决定怎样将信息打包成消息块。何解码。“数据数量”字段只有在响应包中才有。

三、Modbus消息帧

两种传输模式中（ASCII或RTU），传输设备以将Modbus消息转为有起点和终点的帧，这就允许接收的设备在消息起始处开始工作，读地址分配，判断哪一个设备被选中（广播方式则传给所有设备），判知何时信息已完成。部分的消息也能侦测到并且错误能设置为返回结果。

1、ASCII帧

使用ASCII模式，消息以冒号（:）字符（ASCII码 3AH）开始，以回车换行符结束（ASCII码 0DH,0AH）。
其它域可以使用的传输字符是十六进制的0...9,A...F。网络上的设备不断侦测“:”字符，当有一个冒号接收到时，每个设备都解码下个域（地址域）来判断是否发给自己的。

消息中字符间发送的时间间隔最长不能超过1秒，否则接收的设备将认为传输错误。一个典型消息帧如下所示：

| | | | | | |
|------|------|------|------|-------|------|
| 起始位 | 设备地址 | 功能代码 | 数据 | LRC校验 | 结束符 |
| 1个字符 | 2个字符 | 2个字符 | n个字符 | 2个字符 | 2个字符 |

2、RTU帧

使用RTU模式，消息发送至少要以3.5个字符时间的停顿间隔开始。在网络波特率下多样的字符时间，这是最容易实现的(如下图的T1-T2-T3-T4所示)。传输的第一个域是设备地址。可以使用的传输字符是十六进制的0...9,A...F。网络设备不断侦测网络总线，包括停顿间隔时间内。当第一个域（地址域）接收到，每个设备都进行解码以判

整个消息帧必须作为一连续的流传输。如果在帧完成之前有超过1.5个字符时间的停顿时间，接收设备将刷新不完整的消息并假定下一字节为新消息的地址域。同样地，如果一个新消息在小于3.5个字符时间内接着前个消息开始，接收的设备将认为它是前一消息的延续。这将导致一个错误，因为在最后接收的值不可能是正确的。一典型的消息帧如下所示：

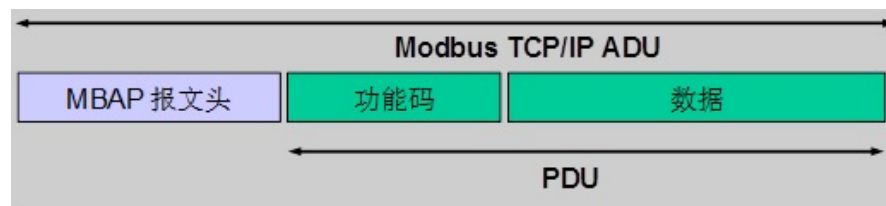
| 起始位 | 设备地址 | 功能代码 | 数据 | CRC校验 | 结束符 |
|-------------|------|------|--------|-------|-----------|
| T1-T2-T3-T4 | 8Bit | 8Bit | n个8Bit | 16Bit | T1-T2-... |

消息帧的地址域包含两个字符（ASCII）或8Bit（RTU）。可能的从设备地址是0...247（十进制）。单个设备的地址范围是1...247。主设备通过放入消息中的地址域来选通从设备。当从设备发送回应消息时，它把自己的地址放入回应的地址域中，以便主设备知道是哪一个设备作出回应。地址0是用作广播地址，以使所有的从设备都能认识。当Modbus协议用于更高水准的网络，广播可能不允许或以其它方式代替。

应答包中，数据包括了数据字节长度+数据值，请求包中数据只包含数据值。

Modbus TCP/IP数据帧结构

Modbus TCP/IP数据帧除了TCP已经有的包头外，还有modbus TCP协议数据单元（ADU），包括MBAP帧头以及RTU数据内容相同的应用数据单元（PDU），地址码除外。



其中与单纯的TCP/IP或是modbus-RTU相比，多的内容就是一个MBAP报文头：

MBAP 报文头包括下列域:

| 域 | 长度 | 描述 | 客户机 | 服务器 |
|--------|-------|-----------------------|-----------|----------------|
| 事务元标识符 | 2 个字节 | MODBUS 请求/响应事务处理的识别码 | 客户机启动 | 服务器从接收的请求中重新复制 |
| 协议标识符 | 2 个字节 | 0=MODBUS 协议 | 客户机启动 | 服务器从接收的请求中重新复制 |
| 长度 | 2 个字节 | 以下字节的数量 | 客户机启动（请求） | 服务器（响应）启动 |
| 单元标识符 | 1 个字节 | 串行链路或其它总线上连接的远程从站的识别码 | 客户机启动 | 服务器从接收的请求中重新复制 |

12

MBAP报文头定义

可以看出来，MBAP报文头主要添加了以下附加信息，为了识别是请求还是响应而设置的**事务元标识符**(2个字节，通常为0，客户端发出信息，服务器端只是需要将这两个字节的内容复制以后再放到回复报文的相应位置就可以)、为了判断协议类型设置的**协议标识符**(2个字节，0=MODBUS协议)、为了区分可变长度数据帧结束的**数据帧长度**(从下一个字节起至结束的长度，2个字节)、还有用于标识从站地址的**单元标识符**(1个字节，即从站地址)，与RTU不同的是，从站地址放在了MBAP帧头里。这个里面的2个字节默认都是大端对齐（高字节、低字节）。

PDU单元与MODBUS RTU数据内容基本相同，由于有TCP/IP和链路层（以太网）校验和机制所以去掉了CRC校验码，从站地址也放在了MBAP帧头里。

另外Modbus TCP/IP默认端口为502。

例子：

Modbus 控制命令为：

00 01 00 00 00 09 04 10 00 00 00 01 02 00 01

MBAP

PDU

上述命令可简单的解释为：00 01（事务标识符）00 00（协议标识符）00 09（后续字节数）04（设备标识符，即从站地址）10（功能码，写多个保持寄存器值）00 00（第一个地址，即地址1）00 01（写寄存器的个数，1个）02（后续所写数据的长度）00 01（具体写的数据）。

Modbus RTU与Modbus TCP读指令对比

[Python学习路线](#)

[转型AI人工智能指南](#)

[2019人工智能发展趋势](#)

[IT 巨头的敏捷之路](#)

[登录](#)

[注册](#)

×

| | | | | | | |
|------------|----------------------|----|----|-------|-------|-------|
| Modbus RTU | 无 | 01 | 03 | 01 8E | 00 04 | 25 DE |
| Modbus TCP | 00 00 00 00 00 06 00 | 无 | 03 | 01 8E | 00 04 | 无 |

指令的涵义：从地址码为01（TCP协议单元标志为00）的模块0x18E(01 8E)寄存器地址开始读（03）四个（00 04）寄存器。

Modbus RTU与Modbus TCP写指令对比

| | MBAP报文头 | 地址码 | 功能码 | 寄存器地址 | 寄存器数量 | 数据长度 | CRC校验 |
|-----|----------------------|-----|-----|-------|-------|------|-------|
| RTU | 无 | 01 | 10 | 01 8E | 00 01 | 02 | A8 7E |
| TCP | 00 00 00 00 00 09 00 | 无 | 10 | 01 8E | 00 01 | 02 | 无 |

指令的涵义：从地址码为01（TCP协议单元标志为00）的模块0x18E(01 8E)寄存器地址开始写（10）一个（00 01）寄存器，具体数据长度为2个字节（02），数据正文内容为00 00（00 00）。

MODBUS ASCII和RTU两种模式的区别、优缺点

MODBUS ASCII协议和RTU协议的比较：

| 协议 | 开始标记 | 结束标记 | 校验 | 传输效率 | 程序处理 |
|-------|--------|--------|-----|------|-----------|
| ASCII | : (冒号) | CR, LF | LRC | 低 | 直观，简单，易调试 |
| RTU | 无 | 无 | CRC | 高 | 稍复杂 |

MODBUS的ASCII协议和RTU协议相比，MODBUS ASCII协议拥有开始和结束标记，而MODBUS RTU却没有，所以ASCII协议的程序中对数据包的处理能更加方便。MODBUS ASCII协议的DATA域传输的都是可见的ASCII字符，因此在调试阶段就显得更加直观，另外它的LRC校验程序也比较容易编写，这些都是MODBUS ASCII的优点。MODBUS ASCII的主要缺点是传输效率低，因为它传输的都是可见的ASCII字符，原来用RTU传输的数据每一个字节，用ASCII的话都要把这个字节拆分两个字节，比如RTU传输一个十六进制数0xF9，ASCII就需要传输字符'F'和字符'9'，对应的ASCII码0x46和0x39两个字节，这样它的传输的效率肯定就比RTU低。所以一般来说，如果所需要传输的数据量较小可以考虑使用ASCII协议，如果所需传输的数据量比较大，最好能使用RTU协议。另外，由于ASCII协议有开始标志和结束标志，所以一个数据包之间的各字节间的传输间隔时间可以大于1秒，而MODBUS RTU方式下，由于没有规定开始和结束标记，所以协议规定每两个字节之间发送或者接收的时间间隔不能超过3.5倍字符传输时间。如果两个字符时间间隔超过了3.5倍的字符传输时间，就认为一帧数据已经接收，新的一帧数据传输开始，所以RTU方式下两个字节间传输间隔有时间要求。MODBUS的ASCII和RTU两种协议的这一区别可能决定某些应用场合只能选用其中一

下表列出MODBUS支持的部分功能代码：以十进制表示。

表1.1 MODBUS部分功能码

| 代码 | 中文名称 | 寄存器PLC地址 | 位操作/字操作 | 操 | 12 |
|----|----------|-------------|---------|----|----|
| 01 | 读线圈状态 | 00001-09999 | 位操作 | 单 | 个 |
| 02 | 读离散输入状态 | 10001-19999 | 位操作 | 单 | 个 |
| 03 | 读保持寄存器 | 40001-49999 | 字操作 | 单 | 个 |
| 04 | 读输入寄存器 | 30001-39999 | 字操作 | 单 | 个 |
| 05 | 写单个线圈 | 00001-09999 | 位操作 | 单 | |
| 06 | 写单个保持寄存器 | 40001-49999 | 字操作 | 单个 | |
| 15 | 写多个线圈 | 00001-09999 | 位操作 | 多个 | |
| 16 | 写多个保持寄存器 | 40001-49999 | 字操作 | 多个 | |

modbus协议的数据主要分为四类：离散量输入、线圈状态、输入寄存器、保持寄存器。离散量输入对应开入（遥信），线圈状态对应哪开出（遥控），输入寄存器对应只读的模拟量（遥测），保持寄存器对应可读可写的模拟量（遥调）。

1.1功能码说明

功能码可以分为位操作和字操作两类。位操作的最小单位为BIT，字操作的最小单位为两个字节。

【位操作指令】 读线圈状态01H，读(离散)输入状态02H，写单个线圈06H和写多个线圈0FH。

【字操作指令】 读保持寄存器03H，写单个寄存器06H，写多个保持寄存器10H。

1.2Modbus数据模型

Modbus中，数据可以分为两大类，分别为Coil和Register，每一种数据，根据读写方式的不同，又可细分为两种（只读，读写）。

Modbus四种数据类型：

Python学习路线

转型AI人工智能指南

2019人工智能发展趋势

IT 巨头的敏捷之路

| | | | | |
|-------------------|----------|----|----|--|
| Discretes Input | 位变量 | 只读 | | |
| Coils | 位变量 | 读写 | 12 | |
| Input Registers | 16-bit类型 | 只读 | | |
| Holding Registers | 16-bit类型 | 读写 | | |

表1.3 MODBUS寄存器地址分配

| 寄存器PLC地址 | 寄存器协议地址 | 适用功能 | 寄存器种类 | 读写状态 | 描述 |
|-------------|-------------|-----------------|---------|------|----------------------------------|
| 00001-09999 | 0000H-FFFFH | 01H 05H 0FH | 线圈状态 | 可读可写 | |
| 10001-19999 | 0000H-FFFFH | 02H | 离散输入状态 | 可读 | |
| 30001-39999 | 0000H-FFFFH | 04H | 输入寄存器 | 可读 | 每个寄存器表示一个16-bit无符号整数 (0~65535) |
| 40001-49999 | 0000H-FFFFH | 03H 06H 0FH | 保持寄存器 | 可读可写 | 两个连续16-bit寄存器表示一个浮点数 |
| 50001-59999 | 0000H-FFFFH | 03H 04H 06H 0FH | ASCII字符 | 可读可写 | 每个寄存器表示两个ASCII字符 |

表1.4 MODBUS寄存器种类说明

| 寄存器种类 | 说明 | PLC类比 | 举例说明 |
|-------|-------------------------------------|-------------------|--------------------------------------|
| 通用寄存器 | 用于存放数据，可由用户程序任意读写。 | PLC的V区（变量存储器） | 例如：M0（启动按钮）的常开触点连接到线圈，线圈驱动M1（启动继电器）。 |
| 特殊寄存器 | 用于存放特殊数据，如系统参数、I/O地址等，通常由系统或特定程序访问。 | PLC的I/O地址、系统参数存储器 | 例如：PLC的I地址（如I0.0）和Q地址（如Q0.0）。 |
| 累加器 | 用于存放运算结果，通常由CPU或特定指令访问。 | PLC的累加器（ACC） | 例如：PLC的累加器（ACC）用于存放运算结果。 |

| | | | |
|--------|--------------------------------|-------------|--------------------------|
| 离散输入状态 | 输入端口。通过外部设定改变输入状态，可读但不可写。 | DI 数字量输入 | 拨码开关，接近开关等。 |
| 保持寄存器 | 输出参数或保持参数，控制器运行时被设定的某些参数。可读可写。 | AO 模拟量输出 | 模拟量输出设定值，PID运行参数，报警上限下限。 |
| 输入寄存器 | 输入参数。控制器运行时从外部设备获得的参数。可读但不可写。 | AI 模拟量输入 | 模拟量输入 |

1.5 PLC地址和协议地址区别

PLC地址可以理解为协议地址的变种，在触摸屏和PLC编程中应用较为广泛。

1.5.1寄存器PLC地址

寄存器PLC地址指存放于控制器中的地址，这些控制器可以是PLC，也可以使触摸屏，或是文本显示器。PLC地址一般采用10进制描述，共有5位，其中第一位代码寄存器类型。第一位数字和寄存器类型的对应关系如表1所示。PLC地址例如40001、30002等。

1.5.2寄存器协议地址

寄存器协议地址指通信时使用的寄存器地址，例如PLC地址40001对应寻址地址0x0000，40002对应寻址地址0x0001，寄存器寻址地址一般使用16进制描述。再如，PLC寄存器地址40003对应协议地址0002，PLC寄存器地址30003对应协议地址0002，虽然两个PLC寄存器寄存器通信时使用相同的地址，但是需要使用不同的命令访问，所以访问时不存在冲突。

五、注意事项

1、MODBUS 3.5T是如何计算的？

1、1600高地址为起始位，数据位，奇偶校验，停止位，波特率

Python学习路线

转型AI人工智能指南

2019人工智能发展趋势

IT 巨头的敏捷之路

登录

注册

×

如果你的通讯方式是：波特率115200（表示每秒传输多少字符），数据位8，无奇偶校验。那么你发送一个字符的时间是： $T=1000 \times (1\text{起始位}+8\text{数据位}+0\text{奇偶校验}+1\text{停止位})/115200=0.087\text{ms}$ 。

发送端：发送一帧后延时 $7 \times T$ （其中 $3.5T$ 是停止时间， $3.5T$ 是起始时间）再发送第二帧，保证一帧数据里头各字节间间隔延时不能超过 $1.5T$ 。

接收端：接收一个字节，查询 $2T$ 时间，是否有接收到下一个字节，有则这帧数据未完，继续循环接收；没有则默认这帧已经接收完毕。

2、串口超时时间设置

COMMTIMEOUTS timeout;

//填充timeout结构

GetCommTimeouts(m_h232Port,&timeout);

timeout.ReadIntervalTimeout=100;//两字符之间最大的延时

timeout.ReadTotalTimeoutConstant=500;

timeout.ReadTotalTimeoutMultiplier=0;//读取每字符间的超时

timeout.WriteTotalTimeoutConstant=2000;

timeout.WriteTotalTimeoutMultiplier=60;//写入每字符间的超时

BOOL error=SetCommTimeouts(m_hPort,&timeout);

ReadIntervalTimeout：两字符之间最大的延时，当读取串口数据时，一旦两个字符传输的时间差超过该时间，读取函数将返回现有的数据。设置为0表示该参数不起作用。

ReadTotalTimeoutMultiplier：读取每字符间的超时。

ReadTotalTimeoutConstant：一次读取串口数据的固定超时。所以在一次读取串口的操作中，其超时为ReadTotalTimeoutMultiplier乘以读取的字节数再加上

ReadTotalTimeoutConstant。将ReadIntervalTimeout设置为MAXDWORD，并将ReadTotalTimeoutMultiplier和ReadTotalTimeoutConstant设置为0，表示读取操作将立即返回存放在输入缓冲区的字符。

WriteTotalTimeoutMultiplier：写入每字符间的超时。

WriteTotalTimeoutConstant：一次写入串口数据的固定超时。所以在一次写入串口的操作中，其超时为WriteTotalTimeoutMultiplier乘以写入的字节数再加上

WriteTotalTimeoutConstant。

| |
|----|
| 12 |
| |
| |
| |
| |
| |

C++ class for Win32 serial ports : <http://www.naughter.com/serialport.html>

CSerialPort : <https://github.com/itas109/CSerialPort>

4、如何写稳定的modbus代码？

需要从稳定性和易读性来考虑，如果稳定性较差会造成系统控制故障，如果易读性差就会造成难以维护，这些控制指令之间差别很小，如果一个一个单独写命令，容易出错。对应举措如下：

1) 避免每个指令写一部分代码，需要统一处理，比如校验函数，发送函数，接收函数等。

通信协议已知，从中可以知道通信的实际数据长度（不包含包头包尾和校验的部分），所以可以控制读写多少个字节，并且可以知道什么时候启动校验，那参与校验计算。

2) 为指令建立指令列表，这样后来需要添加功能，就可以直接把指令字加入列表即可。

3) 适当抽象。为每个指令的动作写回调函数，这样就可以在应用层，用一句简单的回调函数指针直接操作具体的动作函数，而不是应用逻辑层操作具体的驱动层面的接口。

4) 超时，必须有超时机制。通信失败怎么处理？通信之后一般线断了怎么处理？不能让系统死等后面的几个字节发过来。

5) 接收超时机制，不能依靠数传输的字节个数来停止接收来和区分帧间隔，因为可能通信就是断掉了，所以要按照协议，串口通信情况下3~5个T空闲就认为一帧结束。也可以通过协议事先计算出接收数据的总时间（字符数*（字符时间+字符间间隔时间）），使用该使用作为等待超时时间，但最长超时时间不能超过5T时间。

6) 响应超时机制，Modbus是请求/应答式通信，那么主机就需要知道到底多久从机才会应答，主机等待从机应答的最长时间就是从机的最大回复间隔，超过这个时间后从机即使已经完成计算也不能回复，因为此时主机可能已经开始给其他从机发送数据了。

7) 如果通信都是由你发起的，那么无论此次通信是完成还是超时或失败，都应该关闭通信端口。

8) 如果是线程中操作串口，可以提升线程优先级，让操作系统在一定程度上提高串口读写性能。

9) ModbusRTU设备参数设置如下：

(1) 内部属性：单击“查看设备内部属性”，点击按钮进入内部属性

(2) 最小采集周期：组态软件对设备进行操作的时间周期，单位为ms，默认为100ms，根据采集数据量的大小,设置值可适当调整

(3) 设备地址：必须和实际设备的地址相一致，范围为0-255，默认值为0。

(4) 通讯等待时间：通讯数据接收等待时间，默认设置为200ms,根据采集数据量的大小,设置值可适当调整。

Python学习路线

转型AI人工智能指南

2019人工智能发展趋势

IT巨头的敏捷之路

登录

注册

×

16 位整数解码顺序 举例：0x0001

0-12 表示字元件高低字节不颠倒（默认值）表示 1

1-21 表示字元件高低字节颠倒 表示 256

（7）32位整数解码顺序：调整双字元件的解码顺序，对于Modicon PLC，请设置为“2-3412”顺序解码。

32 位整数解码顺序 举例：0x0000 0001

0-1234 表示双字元件不做处理直接解码（默认值）表示 1

1-2143 表示双字元件高低字不颠倒，但字内高低字节颠倒 表示256

2—3412表示双字元件高低字颠倒，但字内高低字节不颠倒 表示65536

3—4321表示双字元件内4个字节全部颠倒 表示 16777216

（8）32位浮点数解码顺序：调整双字元件的解码顺序，对于Modicon PLC，请设置为“2-3412”顺序解码。

32 位浮点数解码顺序 举例：0x3F80 0000

0-1234 表示双字元件不做处理直接解码（默认值）表示 1.0

1-2143 表示双字元件高低字不颠倒，但字内高低字节颠倒 表示-5.78564e-039

2—3412表示双字元件高低字颠倒，但字内高低字节不颠倒 表示2.27795e-041

3—4321表示双字元件内4个字节全部颠倒 表示 4.60060e-041

（9）校验方式：选择LRC校验值的组合方式，对于 Modicon PLC及标准 PLC 设备，使用默认设置即可。

0—LH[低字节，高字节]：校验结果为2个字节，低字节在前，高字节在后。

1—HL[高字节，低字节]：校验结果为2个字节，高字节在前，低字节在后。默认值。

（10）分块采集方式：驱动采集数据分块的方式，对于Modicon PLC及标准 PLC设备，使用默认设置可以提高采集效率。

0—按最大长度分块：采集分块按最大块长处理,对地址不连续但地址相近的多个分块,分为一块一次性读取,以优化采集效率。

1—按连续地址分块：采集分块按地址连续性处理,对地址不连续的多个分块,每次只采集连续地址,不做优化处理。

例如：有4区寄存器地址分别为 1~5，7，9~12的数据需采集，如果选择“0 - 按最大长度分块”，则两块可优化为地址1~12的数据打包1次完成采集；如果选择“1 - 按连续地址分块”，则需要采集 3 次。

（11）4区16 位写功能码选择：写 4 区单字时功能码的选择，这个属性主要是针对自己制作设备的用户而设置的，这样的设备4区单字写可能只支持 0x10 功能码，而不支持0x06 功能码。

0—0x06：单字写功能码使用0x06。

1—0x10：单字写功能码使用0x10。

注意：

（1）.“解码顺序”及“校验方式”设置：主要是针对非标准 ModbusRTU 协议的不同解码及校验顺序。当用户通过本驱动软件与设备通讯时，如果出现解析数据值不对，或者通讯校验错误

| |
|----|
| 12 |
| |
| |
| |
| |
| |

[Python学习路线](#)[转型AI人工智能指南](#)[2019人工智能发展趋势](#)[IT 巨头的敏捷之路](#)[登录](#)[注册](#)[×](#)

(2). “分块采集方式” 设置：主要是针对非标准 ModbusRTU协议设备。当用户通过本驱动软件与设备通讯时，如果按默认 “0 - 按最大长度分块” 时，出现读取连续地址正常，而不连续地址不正常时，可与厂家咨询，并设置为 “1 - 按连续地址分块方式” 尝试是否可正常通讯。而对于 Modicon PLC 及支持标准 ModbusRTU 的 PLC 及控制器等设备，直接使用默认设置即可，这样可以提高采集效率。

<http://blog.csdn.net/gshgsh1228/article/details/51218306>

5、lrc和crc校验算法

```

1 //-----
2 // Constants
3 //-----
4 static const uint16_t modbus_crc_table[256] = {
5     0x0000, 0xc0c1, 0xc181, 0x0140, 0xc301, 0x03c0, 0x0280, 0xc241,
6     0xc601, 0x06c0, 0x0780, 0xc741, 0x0500, 0xc5c1, 0xc481, 0x0440,
7     0xcc01, 0x0cc0, 0x0d80, 0xcd41, 0x0f00, 0xcfc1, 0xce81, 0x0e40,
8     0x0a00, 0xcac1, 0xcb81, 0x0b40, 0xc901, 0x09c0, 0x0880, 0xc841,
9     0xd801, 0x18c0, 0x1980, 0xd941, 0x1b00, 0xdb81, 0xda81, 0x1a40,
10    0x1e00, 0xdec1, 0xdf81, 0x1f40, 0xdd01, 0x1dc0, 0x1c80, 0xdc41,
11    0x1400, 0xd4c1, 0xd581, 0x1540, 0xd701, 0x17c0, 0x1680, 0xd641,
12    0xd201, 0x12c0, 0x1380, 0xd341, 0x1100, 0xd1c1, 0xd081, 0x1040,
13    0xf001, 0x30c0, 0x3180, 0xf141, 0x3300, 0xf3c1, 0xf281, 0x3240,
14    0x3600, 0xf6c1, 0xf781, 0x3740, 0xf501, 0x35c0, 0x3480, 0xf441,
15    0x3c00, 0xfcc1, 0xfd81, 0x3d40, 0xff01, 0x3fc0, 0x3e80, 0xfe41,
16    0xfa01, 0x3ac0, 0x3b80, 0xfb41, 0x3900, 0xf9c1, 0xf881, 0x3840,
17    0x2800, 0xe8c1, 0xe981, 0x2940, 0xeb01, 0x2bc0, 0x2a80, 0xea41,
18    0xee01, 0x2ec0, 0x2f80, 0xef41, 0x2d00, 0xedc1, 0xec81, 0x2c40,
19    0xe401, 0x24c0, 0x2580, 0xe541, 0x2700, 0xe7c1, 0xe681, 0x2640,
20    0x2200, 0xe2c1, 0xe381, 0x2340, 0xe101, 0x21c0, 0x2080, 0xe041,
21    0xa001, 0x60c0, 0x6180, 0xa141, 0x6300, 0xa3c1, 0xa281, 0x6240,
22    0x6600, 0xa6c1, 0xa781, 0x6740, 0xa501, 0x65c0, 0x6480, 0xa441,
23    0x6c00, 0xacc1, 0xad81, 0x6d40, 0xaf01, 0x6fc0, 0x6e80, 0xae41,
24    0xaa01, 0x6ac0, 0x6b80, 0xab41, 0x6900, 0xa9c1, 0xa881, 0x6840,
25    0x7800, 0xb8c1, 0xb981, 0x7940, 0xbb01, 0x7bc0, 0x7a80, 0xba41,
26    0xbe01, 0x7ec0, 0x7f80, 0xbf41, 0x7d00, 0xbdc1, 0xbc81, 0x7c40,

```

12

Python学习路线

转型AI人工智能指南

2019人工智能发展趋势

IT 巨头的敏捷之路

登录

注册

✕

```

30 0x9601, 0x56c0, 0x5780, 0x9741, 0x5500, 0x95c1, 0x9481, 0x5440,
31 0x9c01, 0x5cc0, 0x5d80, 0x9d41, 0x5f00, 0x9fc1, 0x9e81, 0x5e40,
32 0x5a00, 0x9ac1, 0x9b81, 0x5b40, 0x9901, 0x59c0, 0x5880, 0x9841,
33 0x8801, 0x48c0, 0x4980, 0x8941, 0x4b00, 0x8bc1, 0x8a81, 0x4a40,
34 0x4e00, 0x8ec1, 0x8f81, 0x4f40, 0x8d01, 0x4dc0, 0x4c80, 0x8c41,
35 0x4400, 0x84c1, 0x8581, 0x4540, 0x8701, 0x47c0, 0x4680, 0x8641,
36 0x8201, 0x42c0, 0x4380, 0x8341, 0x4100, 0x81c1, 0x8081, 0x4040
37 };
38
39 //-----
40 // Modbus functions
41 //-----
42 uint8_t modbus_lrc_calc(uint8_t *data, uint16_t len)
43 {
44     uint8_t lrc = 0U;
45     int i;
46
47     for (i = 0U; i < len; i++)
48     {
49         lrc += data[i];
50     }
51     lrc = (0xFFU - lrc)+1U;
52     return(lrc);
53 }
54
55 uint16_t modbus_crc_calc(uint8_t *buffer, uint16_t size)
56 {
57     uint16_t crc = 0xFFFFU;
58     uint8_t nTemp;
59
60     while (size--)
61     {
62         nTemp = *buffer++ ^ crc;
63         crc >>= 8;
64         crc ^= modbus_crc_table[(nTemp & 0xFFU)];
65     }
66 }

```

12

Python学习路线

转型AI人工智能指南

2019人工智能发展趋势

IT 巨头的敏捷之路

登录

注册

✕

LRC算出来的是16进制的字节值，若发送的数据是字符串形式，不需任何转换，直接放在字符串最后位置就行了。若发送的数据是数组形式，则需要转换为ASCII。由1字节的16进制转换为2字节的ASCII值，4D应该转换为34 44。

如果你的发送指令是：com.output="xxxxx",就是字符串形式；

如果发送指令形式是：com.output=A[],就是数组形式。

但不管是哪种形式，ModBus ASCII模式最终在数据线上的数据都是ASCII值。字符串形式的数据，编译软件会自动转换。

6、调试工具

Modbus Poll是一个Modbus管理模拟器软件，帮助开发人员进行管理和监控的Mod bus数据区在同一时间和模拟Mod bus协议。支持Modbus RTU / ASCII和Modbus TCP协议。

下载地址：<http://www.downcc.com/soft/25945.html>

modbus slave是一款功能强大的modbus子设备模拟工具，可以帮助modbus通讯设备开发人员进行modbus通讯协议的模拟和测试，用于模拟、测试、调试modbus通讯设备。

下载地址：<http://www.ddooo.com/softdown/70166.htm>

7、modbus开源项目

<https://github.com/stephane/libmodbus>

vc2008 版本下载：<http://download.csdn.net/download/byxdaz/10011387>

参考资料

<http://www.cnblogs.com/luomingui/archive/2013/06/14/Modbus.html>



想对作者说点什么

Python学习路线

转型AI人工智能指南

2019人工智能发展趋势

IT 巨头的敏捷之路

登录

注册

×

MODBUS-RTU数据帧格式、报文实例

阅读数 4

MODBUS-RTU报文模型设备地址功能代码数据格式CRC校验LCRC校验H8bit8bitN*8bit8bit8bit一个报文就是一帧... 博文 来自： 欧阳鑫

Modbus协议

阅读数 1

(-)MODBUS规约MODBUS规约是MODICOM公司开发的一个为很多厂商支持的开放规约， Modbus 协议是应用于... 博文 来自： bmbm546的专栏

物联网RTU (Modbus TCP协议) Java接口开发及Modbus Slave仿真使用

阅读数 3852

在物联网体系中，经常用到RTU（远程终端单元），RTU是负责对现场信号、工业设备的监测和控制，通常由信... 博文 来自： 肖永威的专栏

modbus注意几点

阅读数 7361

1、在利用Modbus通讯的过程中，遇到这样一个问题，即浮点数的传输问题。因为一般浮点数都是32位，而Modbu... 博文 来自： ponydph的专栏

关于Modbus 3区、4区寄存器地址的理解以及Freemodbus中开始地址的设定

阅读数 2943

在Modbus实际应用中，我们对Modbus3区、4区的地址有的时候会出现混淆，尤其是类似于404097这种表达方式... 博文 来自： 猪哥的专栏

modbus协议中的寄存器理解

阅读数 5866

最近有用到modbus协议，就把之前原来收集的资料全都拿出来又复习了一遍。发现以前了解的也忘了差不多了... 博文 来自： bijinsong的博客

Modbus测试工具ModbusPoll与Modbus Slave使用方法

阅读数 3

Modbus测试工具ModbusPoll与ModbusSlave使用方法 博文 来自： 深之JohnChen的...

libmodbus中文手册详解

阅读数 4063

最近做libmodbus相关内容，因为中文没有libmodbus各个函数的详细解释，所以在此把要用的libmodbus的手册包... 博文 来自： 跃动的风的博客

Modbus协议(最全版)

04-12

MODBUS协议支持传统的RS-232、RS-422、RS-485和以太网设备。许多工业设备，包括PLC，DCS，智能仪表等都在使用Modbus协议作为...

下载

ModBus-RTU详解

阅读数 6

Modbus一个工业上常用的通讯协议、一种通讯约定。Modbus协议包括RTU、ASCII、TCP。其中MODBUS-RTU最... 博文 来自： brucezcg的专栏

Python学习路线

转型AI人工智能指南

2019人工智能发展趋势

IT 巨头的敏捷之路

登录

注册

×

[关注](#) 排名:947

[关注](#) 排名:千里之外

[关注](#) 排名:千里之外

[关注](#) 排名:千里之外

Modbus_master/slave_TCP/RTU

阅读数 199

Modbus是一种串行通信协议,是Modicon于1979年为使用可编程逻辑控制器PLC而发表的.MOVBUS是工业领域通信... 博文 来自: [aa70m1_xl的博客](#)

12

Modbus-RTU通信入门

阅读数 1

Modbus-RTU一、数据分析 两个设备 (单片机) 通讯,用的是Modbus协议。 在单片机中拿出一部分内存 (RA... 博文 来自: [大漠飞鹰lb的博客](#)

modbus 入门篇 , 不冗长 , 很好理解 ! (转自中国工控网)

阅读数 2260

先来简单分析一条MOVBUS-RTU报文,例如: 01 06 0001 0017 9804 01 06 0001 0017 ... 博文 来自: [lhq_215的专栏](#)

Swagger与SpringMvc集成生成Restful形式接口文档

阅读数 1

swagger提供的接口文档相比传统的文档方式更加直观也更加高效,但是在网上找了很多关于Swagger与SpringMvc... 博文 来自: [zjx2016的博客](#)

.NET和java的RSA互通 , 仅此而已

阅读数 3

RSA.netjva互通解决不能互通的问题

博文 来自: [lubiaopan的专栏](#)

推荐2本学习Modbus通信协议的资料书

阅读数 3400

基本上玩物联网、串口的都离不开modbus开发,很多人觉得简单,从网上随便找资料看看,其实不太系统,结果是... 博文 来自: [Modbus软件开发...](#)

ModBus RTU协议总结

阅读数 67

上一篇讲了ModBusTCP协议总结,有了上一篇的基础ModBusRTU协议理解起来就简单多了。ModBusTCP比Mod... 博文 来自: [Allone2333的博客](#)

Modbus功能码与数据类型

阅读数 2019

表1 ModBus功能码 功能码 名称 作用 01 读取线圈状态取得一组逻辑线圈的当前状态 (ON/OFF) 02 读取输... 博文 来自: [TNT的博客](#)

ModBus3.5字符的理解

阅读数 3469

假设你的通讯方式是: 波特率115200,数据位8,无奇偶校验。那么你发送一个字符的时间是: T=1/115.2*(1起始位+... 博文 来自: [SunCherryDream...](#)

最好用的modbus工具

09-10

[Python学习路线](#)

[转型AI人工智能指南](#)

[2019人工智能发展趋势](#)

[IT 巨头的敏捷之路](#)

[登录](#)

[注册](#)

[×](#)

modbus

阅读数 78

一.什么是modbus Modbus是由Modicon (现为施耐德电气公司的一个品牌) 在1979年发明的 , 是全球第一个真正... 博文 来自 : [zb313982521的博客](#)

Modbus

阅读数 77

UART : 最底层的协议、 ‘位’ 级别的协议Modbus : 字节级别协议 , 叫应用层通信协议控制仪器 控制PLCSCPI ... 博文 来自 : [辣手熊猫博客](#)

modbus通信协议中的功能码、异常功能码和错误码

阅读数 1

Modbus协议主要构成是地址码/标识码 , 功能码 , 寄存器地址 , 数据报文等内容。由于modbus协议是请求/应答通... 博文 来自 : [欧阳鑫](#)

初学Modbus

阅读数 8617

转载自 : <http://bbs.gkong.com/archive.aspx?id=340353>一个工业上常用的通讯协议、一种通讯约定。Modbus协... 博文 来自 : [hyx283的专栏](#)

Modbus 网络通讯协议

阅读数 5985

1、概述通信协议详细地描述了温湿度传感器的输入和输出命令、信息和数据 , 以便第三方使用和开发。1.1通信协议... 博文 来自 : [拥抱变化](#)

ModBus RTU和ModBus ASC

阅读数 2180

1.RTU的传输机制 RTU , 即远程传输单元。ModBus主机通过RTU方式发送数据帧给从机时 , 数据帧格式为 : ... 博文 来自 : [echo_bright_的博客](#)

浅谈-ModBus-发送报文

阅读数 1533

ModBus协议是什么 , 用于什么样的现场这些我就不介绍了 , 大家自行百度。我对协议本身简单的坐一些解释 , 可能... 博文 来自 : [z5201314100的博客](#)

Modbus RTU/TCP协议解析

阅读数 2

Modbus通信协议由Modicon公司 (现已经为施耐德公司并购 , 成为其旗下的子品牌) 于1979年发明的 , 是全球最... 博文 来自 : [zj20781的博客](#)

MODBUS协议学习的一点心得

阅读数 273

最近刚写完一个完整的MODBUS协议 (RTU) , 包括主站和从站。其实主站从站这个说法不全面 , 应该是请求方 (... 博文 来自 : [laocui1的博客](#)

Modbus全面学习总结带源码资源

03-12

转载 , 很好的学习资料还有源码下载链接 , 言简意赅 , 不浪费时间。 Modbus 协议是应用于电子控制器上的一种通用语言。通过此协议 , 控制...

下载

12

Python学习路线

转型AI人工智能指南

2019人工智能发展趋势

IT 巨头的敏捷之路

登录

注册

×

modbus规约学习

阅读数 200

modbus协议1.历史modbus是一种串行协议,是Modicon于1979年,为使用可编辑逻辑控制器(PLC)而发表的.事实上,...

博文 来自: liyang_nash的专栏

CAN和Modbus RS485总线协议对比总结

03-30

CAN和RS485总线协议对比一览表及各协议的特点说明,注:RS485的协议为Modbus协议

下载

CRC-16/MODBUS 校验位计算

阅读数 3228

实验指令:前23位表示信息头+信息内容。24,25位是待计算的校验位。26位是结束码7E0100000110020027000C...

博文 来自: wanyongtai的博客

Modbus通信CRC16校验程序

阅读数 2167

ModBus通信协议的CRC(冗余循环校验码)含2个字节,即16位二进制数。CRC码由发送设备计算,放置于所发送信息...

博文 来自: 睿思派克

MODBUS学习笔记——modbus tk modbus TCP主机实现

阅读数 4

modbus是一种"古老"但高效的应用层协议。在嵌入式和PC机领域有多种方法实现modbus协议栈,modbus又分为...

博文 来自: 物联网 IoT 经验分...

MODBUS协议整理——汇总

阅读数 2

Modbus是一种串行通信协议,是Modicon于1979年,为使用可编程逻辑控制器(PLC)而发表的。MODBUS是工...

博文 来自: 物联网 IoT 经验分...

寄存器PLC地址与寄存器modbus协议地址

阅读数 2

寄存器PLC地址指存放于控制器中的地址,这些控制器可以是PLC,也可以使触摸屏,或是文本显示器。PLC地址...

博文 来自: 深之JohnChen的...

Modbus协议学习(四)

阅读数 430

Modbus在TCP/IP上的实现一、客户机/服务器模型客户机/服务器模式是基于4种类型报文:1.MODBUS请求——客...

博文 来自: Jagaur_XY的博客

modbus协议(大一科研项目)

阅读数 3200

modbus最近学习了modbus相关的知识,做了一个小实验。在此,做一个小结:此试验最终就是利用单片机跟ds18...

博文 来自: 杨宗纬

485+MODBUS总结 第一章(完)

阅读数 4159

RS485+MODBUS总结第一章(完)

博文 来自: 米亚拉斯

Python学习路线

转型AI人工智能指南

2019人工智能发展趋势

IT巨头的敏捷之路

登录

注册

×

基于Netty框架开发的Modbus源代码。

01-18

基于Netty框架开发的Modbus源代码。支持 * READ COILS | 0x01 * READ DISCRETE INPUTS | 0x02 * READ HOLDING REGISTERS

下载

Modbus大小端

1375

//小端模式for(i=0;i

博文 来自： [xxgxxg的专栏](#)

modbus lrc crc

11-20

自己写的modbus小程序，适用与标注modbus协议也可以是自己定制de，通过注册回调函数来获取数据。

下载

Mod-bus协议数据结构

2949

Mod-bus协议数据结构Mod-bus协议规定了数据传输帧格式及传输过程，对本次设计来说，我们仅响应03命令，即...

博文 来自： [CyberLabs的专栏](#)

MUDBUS通讯协议中文版

11-07

MUDBUS通讯协议中文版 ASCII码和RTU两种方式，具体帧格式功能码

下载

Modbus通用数据读取工具设计及使用

718

读取标准Modbu协议设备的数据，支持Bit，short，int等类型数据的读取，并且支持TCPServer、TCPClie...

博文 来自： [dwx1005526886...](#)

转载一篇Modbus总结

59

https://blog.csdn.net/byxdaz/article/details/77892778

博文 来自： [bemodesty的博客](#)

485 ModbusRTU通讯协议(完整版)

04-19

485 ModbusRTU通讯协议(完整版)，这是我在网上找到的的最全的资料，和大家一块学习！

下载

STM32 上移植FreeModbus详细过程（学习总结）

8369

一、整体代码下面给出一个STM32平台上使用FREEMODBUS最简单的例子，操作保持寄存器，此时操作指令可以为...

博文 来自： [精诚所至](#)

【Unity3D Shader编程】之二 雪山飞狐篇：Unity的基本Shader框架写法&颜色、光照与材质

5万+

本篇文章中，我们学习了Unity Shader的基本写法框架，以及学习了Shader中Properties（属性）的详细写法，光照...

博文 来自： [【浅墨的游戏编程B...](#)

Python学习路线

转型AI人工智能指南

2019人工智能发展趋势

IT 巨头的敏捷之路

登录

注册

×

共同父域下的单点登录

阅读数 1万+

单点登录(Single Sign On), 简称为SSO, SSO不仅在企业级开发很常用, 在互联网中更是大行其道。随便举几个例...

博文 来自: [高爽|Coder](#)

12

机器学习模型评价(Evaluating Machine Learning Models)-主要概念与陷阱

阅读数 6万+

机器学习模型评价(Evaluating Machine Learning Models)-主要概念与陷阱 本文主要解释一些关于机器学习模...

博文 来自: [我和我追逐的梦~~~](#)

zouxy09博客原创性博文导航

阅读数 9万+

zouxy09博客原创性博文导航zouxy09@qq.comhttp://blog.csdn.net/zouxy09 一、基于计算机视觉的目标跟踪计...

博文 来自: [zouxy09的专栏](#)

Java设计模式学习06——静态代理与动态代理

阅读数 6116

一、代理模式为某个对象提供一个代理, 从而控制这个代理的访问。代理类和委托类具有共同的父类或父接口, 这样...

博文 来自: [小小本科生长之路](#)

如何查看自己的电脑端口被什么程序占用了

阅读数 5025

1.首先按windows+R键进入运行程序, 或者通过开始菜单进入运行, 输入cmd命令, 回车 2.输入netstat -ano回车...

博文 来自: [lianxue1986的专栏](#)

整数在计算机中的编码

阅读数 2652

整数在计算机中是使用补码表示的, 在讲解补码前, 先看一下相关概念。 机器数与真值 数值在计算机中的表现形式...

博文 来自: [gaoyi的专栏](#)

plsql的命令 (command) 窗口与sql窗口有什么区别20170620

阅读数 1万+

command窗口是命令窗口, 即为sqlplus窗口, 有命令提示符, 识别sqlplus命令, 基本的命令都可以执行 sql仅可执...

博文 来自: [Ape55的博客](#)

jquery/js实现一个网页同时调用多个倒计时(最新的)

阅读数 20万+

jquery/js实现一个网页同时调用多个倒计时(最新的) 最近需要网页添加多个倒计时. 查阅网络,基本上都是千篇一律的...

博文 来自: [websites](#)

linux上安装Docker(非常简单的安装方法)

阅读数 13万+

最近比较有空, 大四出来实习几个月了, 作为实习狗的我, 被叫去研究Docker了, 汗汗! Docker的三大核心概念: ...

博文 来自: [我走小路的博文](#)

ThreadLocal的设计理念与作用

阅读数 2万+

Java中的ThreadLocal类允许我们创建只能被同一个线程读写的变量。因此, 如果一段代码含有一个ThreadLocal变...

博文 来自: [u011860731的专栏](#)[Python学习路线](#)[转型AI人工智能指南](#)[2019人工智能发展趋势](#)[IT 巨头的敏捷之路](#)[登录](#)[注册](#)

✕

EasyUI - 一个简单的后台管理系统入门实例

阅读数 1万+

采用EasyUI 1.4.x 版本，默认default风格，异步加载页面，多Tab页展示，使用JSON文件模拟从后台动态获取数据... 博文 来自： 般若

Java中BIO、NIO和AIO的区别和应用场景

阅读数 5574

最近一直在准备面试，为了使自己的Java水平更上一个档次，拜读了李林峰老师的《Netty权威指南》，了解了Java... 博文 来自： 我的编程世界

localStorage的过期时间设置的方法？

阅读数 1万+

我们都知道localStorage不主动删除,永远不会销毁，那么如何设置localStorage的过期时间呢，今天我们来一起尝试... 博文 来自： gb4215287的博客

人脸检测工具face_recognition的安装与应用

阅读数 3万+

人脸检测工具face_recognition的安装与应用

博文 来自： roguesir的博客

[转]极线几何约束

阅读数 9090

博文 来自： volkswageos的专栏

python图片处理类之~PIL.Image模块(ios android icon图标自动生成处理)

阅读数 2万+

1.从pyCharm提示下载PIL包 http://www.pythonware.com/products/pil/ 2.解压后，进入到目录下 cd /Users/ji... 博文 来自： 专注于cocos+unit...

一个ajax实现根据积分查询mysql获取用户等级的小demo

阅读数 337

一个小功能,临时用的时候写起来麻烦,所以整理一下,就是普通的ajax请求获取反馈,只是框架用久了,有点生疏. 数据库 ... 博文 来自： houzhyan-博客

Android源码解析之（十四）-->Activity启动流程

阅读数 3万+

好吧，终于要开始讲解Activity的启动流程了，Activity的启动流程相对复杂一下，涉及到了Activity中的生命周期方... 博文 来自： 一片枫叶的专栏

强连通分量及缩点tarjan算法解析

阅读数 34万+

强连通分量：简言之 就是找环（每条边只走一次，两两可达）孤立的一个点也是一个连通分量 使用tarjan算法 在... 博文 来自： 九野的博客

数据挖掘笔记-情感倾向点互信息算法

阅读数 1万+

点间互信息（PMI）主要用于计算词语间的语义相似度，基本思想是统计两个词语在文本中同时出现的概率，如果概... 博文 来自： PURSUE ONE PIECE

[Python学习路线](#)[转型AI人工智能指南](#)[2019人工智能发展趋势](#)[IT 巨头的敏捷之路](#)[登录](#)[注册](#)[×](#)

centos 查看命令源码

阅读数 2万+

yum install yum-utils 设置源: [base-src] name=CentOS-5.4 - Base src - baseurl=http://vault.ce...

博文 来自: [linux/unix](#)

12

DirectX修复工具强力修复实验包

阅读数 1万+

DirectX修复工具API Sets强力修复实验包下载地址: https://pan.baidu.com/share/init?surl=o9fQavS密码: 5h33...

博文 来自: [VBcom的专栏](#)**json.Marshal的小细节**

阅读数 3599

type User struct { id int name string age int class string }// 官网例子 type Color...

博文 来自: [wksw](#)**c#在output窗口输出调试信息**

阅读数 9738

System.Diagnostics.Debug.WriteLine("信息"); 参考: http://blog.csdn.net/lly20000/article/details/44979833...

博文 来自: [dragoo1的专栏](#)**DirectX修复工具增强版**

阅读数 182万+

最后更新: 2018-12-20 DirectX修复工具最新版: DirectX Repair V3.8 增强版 NEW! 版本号: V3.8.0.11638 大小: ...

博文 来自: [VBcom的专栏](#)**【OpenCV人脸识别入门教程之二】人脸检测**

阅读数 2万+

本篇文章主要介绍了如何使用OpenCV实现人脸检测的功能。要实现人脸识别功能,首先要进行人脸检测,判断出图...

博文 来自: [生活,哭泣着奔向...](#)**vsftpd匿名用户上传和下载的配置**

阅读数 1万+

看到很多朋友配置vsftpd时不能使用匿名用户上传和下载(创建目录或删除、重命名文件夹),本文主要解决vsf...

博文 来自: [九宫霓虹](#)**关于SpringBoot bean无法注入的问题(与文件包位置有关)**

阅读数 10万+

问题场景描述整个项目通过Maven构建,大致结构如下: 核心Spring框架一个module spring-boot-base service...

博文 来自: [开发随笔](#)**手把手教你整合最优雅SSM框架: SpringMVC + Spring + MyBatis**

阅读数 17万+

小疯手把手带你整合SpringMVC+Spring+MyBatis三大框架,俗称SSM,用它完全代替传统的SSH框架,把它们最...

博文 来自: [小疯的代码健身房](#)**加密算法介绍及加密算法的选择**

阅读数 1万+

加密算法介绍 一. 密码学简介 据记载,公元前400年,古希腊人发明了置换密码。1881年世界上的第一个电话保密专...

博文 来自: [leolewin的博客](#)[Python学习路线](#)[转型AI人工智能指南](#)[2019人工智能发展趋势](#)[IT 巨头的敏捷之路](#)[登录](#)[注册](#)

✕

Html5的video标签自动填充父div的大小

阅读数 3万+

想要video能自动填充慢父div的大小，只要给video标签加上style="width= 100%; height=100%; object-fit: fill"... 博文 来自： wuqingyou_w的专...

MAC下安装mysql-python方法（亲测可用）

阅读数 8623

mac装mysql-python有点坑，好在解决了，为了避免类似的问题耽误大家时间，我写下我安装的最终过程：我选择... 博文 来自： zhaoteng345的专栏

连续特征离散化和归一化

阅读数 9580

连续特征进行离散化处理。 博文 来自： hero_fantao的专栏

【结合实例】信息增益的计算

阅读数 1万+

参考文章：https://www.cnblogs.com/qcloud1001/p/6735352.html 信息增益原理介绍 介绍信息增益之前，首先... 博文 来自： guomutian911的...


Python(2) 基础语法

阅读数 3994


1. 模块1.1. 从某模块导入函数import somemodule from somemodule import somefunction from somemodul... 博文 来自： 清欢

相机标定总结 图像处理总结 双目视觉总结 3d相机标定总结 图像处理光流算法总结

bootstrap 学习总结 c++学生管理系统学习总结 dreamweaver的学习总结 android io流学习总结 c++ modbus python培训学习总结 pythonweb开发总结学习



byxdaz

 博客专家

关注

| | | | |
|-----|------|-----|------|
| 原创 | 粉丝 | 喜欢 | 评论 |
| 557 | 3428 | 237 | 1151 |

等级：  访问：482万+

- Python学习路线
- 转型AI人工智能指南
- 2019人工智能发展趋势
- IT 巨头的敏捷之路

登录 注册 ×

最新文章

bcd码

时间函数

gps nmea数据格式解析与生成

thrift rpc js使用

MFC中ListBox添加水平滚动条和多行提示类

个人分类

音视频40篇

ACE网络编程14篇

Android15篇

IOS1篇

C/C++27篇

展开

归档

2018年12月8篇

2018年11月5篇

2018年10月4篇

2018年9月4篇

2018年8月4篇

展开

| |
|----|
| 12 |
| |
| |
| |
| |
| |

Python学习路线

转型AI人工智能指南

2019人工智能发展趋势

IT 巨头的敏捷之路

登录

注册

×

BP神经网络基本原理

阅读数 101786

书籍 《世界是平的》

阅读数 92128

墨菲定律、二八法则、马太效应、手表定理、“不值得”定律、彼得原理、零和游

阅读数 84566

STL学习小结

阅读数 73006

网络编程学习小结

阅读数 65315

最新评论

Thrift之c++实例

ljt350740378：运行server程序和多个client程序时，client端没有结果，请问这是怎么回事？

寄存器PLC地址与寄存器modbu...

qq_44667643：施耐德%MD/%MW/%MB对应表出现错误

CCITT标准G726编解码实例

hhhlihao：楼主这个是用哪个库？avilib吗？

博客如何赚钱？

luhu124541：博主要不更新一下

海康、大华等IpCamera RT...

qq_41913924：你好,我问一下,实时预览流地址可以,回放的地址不可以这是是什么原因,可以指导一下吗

便民服务

携程网

Python学习路线

转型AI人工智能指南

2019人工智能发展趋势

IT 巨头的敏捷之路

登录

注册

×

天气预报
万年历查询
58同城
在线翻译
免费取名网站
GOOOGLE地图



微信客服



QQ客服

👤 QQ客服 ✉ kefu@csdn.net
💬 客服论坛 ☎ 400-660-0108
⌚ 工作时间 8:30-22:00

关于我们 招聘 广告服务 网站地图
🐼 百度提供站内搜索 京ICP证19004658号
©1999-2019 北京创新乐知网络技术有限公司

网络110报警服务 经营性网站备案信息
北京互联网违法和不良信息举报中心
中国互联网举报中心

Python学习路线

转型AI人工智能指南

2019人工智能发展趋势

IT 巨头的敏捷之路

登录

注册

×