[1 设计依据 2](#_Toc512120736)

[2 数据传输协议 2](#_Toc512120737)

[2.1 地址域 2](#_Toc512120738)

[2.2 功能码 2](#_Toc512120739)

[2.3 数据 3](#_Toc512120740)

[2.4 差错校验 3](#_Toc512120741)

[2.5 串口通讯----RTU传输形式 4](#_Toc512120742)

[2.5.1 主机对从机读数据操作 4](#_Toc512120743)

[2.5.2 主机对从机写数据操作 5](#_Toc512120744)

[2.6 串口通讯----ASCII传输形式 6](#_Toc512120745)

[2.7 网络通讯----TCP/IP传输形式 6](#_Toc512120746)

[2.7.1 主机读操作 7](#_Toc512120747)

[2.7.2 主机写操作 8](#_Toc512120748)

[3 Modbus协议传输要点 8](#_Toc512120749)

[3.1 通讯参数相关概念 8](#_Toc512120750)

[3.2 字符时间间隔 9](#_Toc512120751)

[3.3 关于传输浮点数的问题 9](#_Toc512120752)

[4 模块详细设计 9](#_Toc512120753)

[4.1 底层应用模块设计 9](#_Toc512120754)

[4.1.1 通讯参数模块 9](#_Toc512120755)

[4.1.2 串口与采集设备通讯 11](#_Toc512120756)

[4.1.3 TCP长连接与采集设备通讯 11](#_Toc512120757)

[4.2套接字模块设计 11](#_Toc512120758)

[4.3数据转换模块设计 11](#_Toc512120759)

[4.3.1 进制转换算法设计 12](#_Toc512120760)

[4.3.2 计算公式算法设计 12](#_Toc512120761)

[4.4 数据分发模块 12](#_Toc512120762)

[4.5 定时任务模块设计 12](#_Toc512120763)

[4.6 协议转换模块设计 12](#_Toc512120764)

[4.7 报警处理模块设计 13](#_Toc512120765)

[4.8 web应用模块设计 13](#_Toc512120766)

[4.8.1 PC端的应用模块 13](#_Toc512120767)

[4.8.2 移动端应用 13](#_Toc512120768)

[5 数据库详细设计 14](#_Toc512120769)

[5.1 底层应用数据库设计 14](#_Toc512120770)

[5.2 web应用数据库设计 15](#_Toc512120771)

[6 系统技术路线方案 15](#_Toc512120772)

# 1 设计依据

《现场工控系统软件概要设计手册》

《Modbus协议中文版.pdf》

《污染物在线监控（监测）系统数据传输标准(HJ 212-2017代替HJT 212-2005)》

# 2 数据传输协议

在具体设计软件模块之前，先了解系统的数据传输协议。

《现场工控系统软件》属于上位机，采集设备属于下位机，上位机与下位机进行通讯，必须遵循一定的协议，这种协议就是**Modbus协议**，它是工控行业的标准协议，Modubus协议在工业上的数据传输模式一般有两种，一种是“串行链路”模式，即**串口通讯**，另一种是“Modbus TCP/IP ”模式，即**网络通讯**。

**串口通讯**的数据传输模式包括**ASCII**和 **RTU**两种传输模式，再加上**网络通讯**（TCP/IP），那么Modbus传输数据就有3中形式了。接下来就这三种传输模式传输的数据格式进行详细说明，因为在硬件模块中，涉及到基于Modbus数据传输，整个系统也就是基于Modbus协议读写数据的。

要注意的是，**串口通讯**有主从通讯之分，Modbus 通信总是由主节点发起。从节点在没有收到来自主节点的请求时，从不会发送数据。这里的主节点、子节点是相对的，也就是说，采集设备可能是主节点也可能是从节点，工控电脑可能是主节点也可能是从节点，另外在数据传输过程中，Modbus协议报文是先传输低位字节，后传输高位字节的。

基于Modbus协议传输的数据帧格式如下图：



一个数据帧最大为256 字节。

## 2.1 地址域

主节点没有地址，只有子节点必须有一个地址，0为广播地址，但一般情况下不会设为0的，因为广播请求一般用于写命令，所有设备必须接受广播模式的写功能。地址0 是专门用于表示广播数据的。

理论上Modbus协议可以接247个从机（1~247），但若用于485接口上则由于485接口的限制，在没有中继情况下，最多可以接32个从机。

**地址域占1个字节**，这里的地址不是指寄存器地址。

## 2.2 功能码

这里的功能码是指常用的功能码，并非Modbus全部的功能码，如需详细了解，请祥看《Modbus协议中文版.pdf》，**功能码在数据帧中占1个字节。**

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01，读(离散)输入状态02，写单个线圈06和写多个线圈0F。

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

常用功能码（16进制）列表如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 代码 | 中文名称 | 寄存器PLC地址 | 位操作/字操作 | 操作数量 |
| 01 | 读线圈状态 | 00001-09999 | 位操作 | 单个或多个 |
| 02 | 读离散输入状态 | 10001-19999 | 位操作 | 单个或多个 |
| 03 | 读保持寄存器 | 40001-49999 | 字操作 | 单个或多个 |
| 04 | 读输入寄存器 | 30001-39999 | 字操作 | 单个或多个 |
| 05 | 写单个线圈 | 00001-09999 | 位操作 | 单个 |
| 06 | 写单个保持寄存器 | 40001-49999 | 字操作 | 单个 |
| 0F | 写多个线圈 | 00001-09999 | 位操作 | 多个 |
| 10 | 写多个保持寄存器 | 40001-49999 | 字操作 | 多个 |

表中功能码为标准Modbus中的功能码，生产厂商在实际中有可能不一样，应以实际的Modbus传输说明为准。

## 2.3 数据

数据即数据域，包括发送指令的数据域和返回指令的数据域，这两种指令的数据域所占的字节长度是不一样的，**所占字节为0 到 252字节**。

## 2.4 差错校验

差错校验指CRC校验或LRC校验，CRC校验用于RTU传输形式或TCP/IP传输形式，LRC校验用于ASCII传输形式，**差错校验在Modbus数据帧中占2个字节。**

如果返回的数据发生错误，则返回的数据格式为：

**地址+功能码+错误码+校验**

错误码（16进制）表：

|  |  |  |
| --- | --- | --- |
| 代码 | 名称 | 含义 |
| 01 | 非法功能 | 对于服务器（或从站）来说，询问中接收到的功能码是不可允许的操作，可能是因为功能码仅适用于新设备而被选单元中不可实现同时，还指出服务器（或从站）在错误状态中处理这种请求，例如：它是未配置的，且要求返回寄存器值。 |
| 02 | 非法数据地址 | 对于服务器（或从站）来说，询问中接收的数据地址是不可允许的地址，特别是参考号和传输长度的组合是无效的。对于带有100个寄存器的控制器来说，偏移量96和长度4的请求会成功，而偏移量96和长度5的请求将产生异常码02。 |
| 03 | 非法数据值 | 对于服务器（或从站）来说，询问中包括的值是不可允许的值。该值指示了组合请求剩余结构中的故障。例如：隐含长度是不正确的。modbus协议不知道任何特殊寄存器的任何特殊值的重要意义，寄存器中被提交存储的数据项有一个应用程序期望之外的值。 |
| 04 | 从站设备故障 | 当服务器（或从站）正在设法执行请求的操作时，产生不可重新获得的差错。 |
| 05 | 确认 | 与编程命令一起使用，服务器（或从站）已经接受请求，并且正在处理这个请求，但是需要长持续时间进行这些操作，返回这个响应防止在客户机（或主站）中发生超时错误，客户机（或主机）可以继续发送轮询程序完成报文来确认是否完成处理。 |
| 07 | 从属设备忙 | 与编程命令一起使用，服务器（或从站）正在处理长持续时间的程序命令，当服务器（或从站）空闲时，客户机（或主站）应该稍后重新传输报文。 |
| 08 | 存储奇偶性差错 | 与功能码20和21以及参考类型6一起使用，指示扩展文件区不能通过一致性校验。服务器（或从站）设备读取记录文件，但在存储器中发现一个奇偶校验错误。客户机（或主机）可重新发送请求，但可以在服务器（或从站）设备上要求服务。 |
| 0A | 不可用网关路径 | 与网关一起使用，指示网关不能为处理请求分配输入端口值输出端口的内部通信路径，通常意味着网关是错误配置的或过载的。 |
| 0B | 网关目标设备响应失败 | 与网关一起使用，指示没有从目标设备中获得响应，通常意味着设备未在网络中。 |

## 2.5 串口通讯----RTU传输形式

### 2.5.1 主机对从机读数据操作

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 地址 | 功能码 | 起始寄存器  地址 | | 寄存器个数 | | CRC16校验 | |
| 09 | 03 | 04 | 00 | 03 | 00 | CA | 01 |

**报文解析：**

**主机告诉09号从机，要从第5个寄存器起，连续读取3个寄存器的数值。整个报文占8个字节，其中起始寄存器地址、寄存器个数、CRC16校验均占2个字节，且高位在前，低位在后。**

**从机响应：**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 地址 | 功能码 | 数据长度 | 寄存器1 | | 寄存器2 | | 寄存器3 | | CRC16校验 | |
| 09 | 03 | 06 | 03 | 00 | 04 | 00 | 05 | 00 | 5F | 3C |

**报文解析：**

**从机按照主机的指令返回了连续3个寄存器的数据，共有6个字节。整个报文占N\*2+6个字节，N为取数值的寄存器个数，且N\*2+6不大于256。其中数据长度=取数值的寄存器个数\*2。**

读单个寄存器也具有类似的操作。

**读取失败：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 地址 | 功能码 | 错误码 | CRC16校验 | |
| 09 | 03 | 04 | CA | 01 |

**报文解析：**

**09号从机告诉主机在执行读保持寄存器数据时发生了错误，错误代码为04（即从机发生故障）。**

### 2.5.2 主机对从机写数据操作

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 地址 | 功能码 | 寄存器地址 | | 寄存器个数 | | 写字节数 | 写入值 | | | | | | CRC16校验 | |
| 09 | 10 | 04 | 00 | 03 | 00 | 06 | 05 | 00 | 07 | 00 | 22 | 00 | 5F | 3C |

**报文解析：**

**主机告诉09号从机，要从第5寄存器起，连续写入3个寄存器的值，共6个字节，写入的值依次是0x0005，0x0007，0x0022。整个报文占N\*2+9个字节，且N\*2+9不大于256，N为写入数值的寄存器个数。**

**从机响应：**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 地址 | 功能码 | 寄存器地址 | | 寄存器个数 | | 写字节数 | 写入值 | | | | | | CRC16校验 | |
| 09 | 10 | 04 | 00 | 03 | 00 | 06 | 05 | 00 | 07 | 00 | 22 | 00 | 5F | 3C |

**报文解析：**

**写入成功后从机原样返回主机发送到从机的指令。**

写单个寄存器也具有类似的操作。

**写入失败：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 地址 | 功能码 | 错误码 | CRC16校验 | |
| 09 | 03 | 07 | CA | 01 |

**报文解析：**

**09号从机告诉主机在执行读保持寄存器数据时发生了错误，错误代码为07（即从机繁忙）。**

## 2.6 串口通讯----ASCII传输形式

ASCII传输形式其实就是将RTU模式的数据全部拆分为2个字节，加上头尾标识，再传输。头标识为输入法处于半角状态下的冒号对应的16进制(3A)，结尾标识不可见的回车换行字符(0D 0A)，**注意数据传输仍然高位在前，低位在后**。

参照2.5.2中主机对从机发送的03读取指令，转换为ASCII传输模式就是：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 地址域 | | 功能码 | | 数据域 | | | | | | | | | | | | 校验区 | |
| **3A** | 30 | 39 | 30 | 33 | 30 | 34 | 30 | 30 | 30 | 33 | 30 | 30 | 43 | 41 | 30 | 31 | **0D** | **0A** |

**3A**：半角状态下的冒号的ASCII码

**30**：30是数字"0"的ASCII码

**39**：39是数字"9"的ASCII码

**30**：30是数字"0"的ASCII码

**33**：39是数字"3"的ASCII码

…

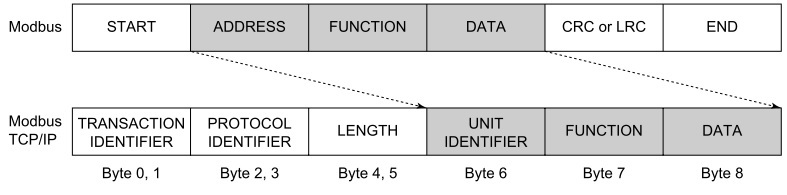
**0D**：回车符的ASCII码

**0A**：换行符的ASCII码

其他写入、返回数据的格式也同比操作。

## 2.7 网络通讯----TCP/IP传输形式

Modbus TCP协议则是在RTU协议上加一个MBAP报文头，由于TCP是基于可靠连接的服务，RTU协议中的CRC校验码就不再需要，所以在Modbus TCP协议中是没有CRC校验码。主站通常称为Client，从站称为Server。Modbus默认采用的TCP端口号是502。**注意数据传输仍然高位在前，低位在后。**



与RTU传输形式相比，TCP/IP传输形式有如下改变：

1、取消了校验位。数据链路层上就进行了CRC-32的校验，TCP/IP是面向连接的可靠性的协议，因此没必要再加上校验位。

2、Slave地址换成了Unit Identifier。当网络里的设备全是使用TCP/IP，这个地址是没有意义的，因为IP就能进行路由寻址。如果网络里还有串行通讯的设备，则需要网关来实现Modbus TCP到Modbus RTU或ASCII之间的协议转换，这时用Unit Identifier(**单元标识**)来标识网关后面的每个串行通讯设备。

3、Length是指后面的字节总数。实际上数据区的长度是能确定的，有的功能码就可以确定数据区的长度，有的功能码虽不能确定数据区长度，但是数据区有字节计数，参见上文举的从站应答的例子。表头增加的Length是为了应对有些情况下TCP/IP协议会将应用层的数据拆包传输。

4、Transaction Identifier(**传输标识**)和Protocol Identifier(**协议标识**)由Client生成，Server的响应将复制这些参数。

TCP/IP传输中的MBAP报头：

|  |  |  |  |
| --- | --- | --- | --- |
| 传输标识 | 协议标识 | 长度 | 单元标识 |
| 2字节，标志某个Modbus询问/应答的传输，可视为ID，客户端生成。默认0。 | 2字节，0=Modbus协议，1=UNI-TE协议，客户端生成。默认0。 | 2字节，后续字节计数，客户端生成。 | 1字节，定义连接于目的，节点的其它设备，客户端生成。 |

### 2.7.1 主机读操作

参照2.5.2中主机对从机发送的03读取指令，转换为TCP/IP传输模式就是：

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 传输标识 | | 协议标识 | | 长度 | | 单元标识 | 功能码 | 寄存器地址 | | 寄存器数量 | |
| 00 | 00 | 00 | 00 | 06 | 00 | 00 | 03 | 04 | 00 | 03 | 00 |

**报文解析：**

**主机告诉从机09，要从第5个寄存器起，连续读取3个寄存器的数值，其中0x0006(即6个字节)是指紧随其后的“00 03 04 00 03 00”这6个字节。**

**返回的数据格式也类似。**

### 2.7.2 主机写操作

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 传输标识 | | 协议标识 | | 长度 | | 单元标识 | 功能码 | 寄存器地址 | | 寄存器数量 | | 数据长度 | | 写入值 | | | | | |
| 00 | 00 | 00 | 00 | 0E | 00 | 00 | 03 | 04 | 00 | 03 | 00 | 06 | 00 | 01 | 00 | 02 | 00 | 03 | 00 |

**报文解析：**

**主机告诉从机09，要从第5个寄存器起，连续设置3个寄存器的数值，设置的数据长度为6个字节，设置的数值依次为：0x0001，0x0002，0x0003。其中0x000E(即14个字节)=7\*2。**

**返回的数据格式也类似。**

# 3 Modbus协议传输要点

在实际RTU模式传输中通常是定时发指令进行数据传输的，这个**定时**是有规定的，不能随便规定一个定时就可以的。

正确发送两组指令的前提“至少3.5个字符传输时间”。在解析“至少3.5个字符传输时间”之前先来说明通讯格式中的参数**波特率，检验方式，数据位，停止位**。

## 3.1 通讯参数相关概念

波特率：一秒钟传送的位数，也就是通讯速率；比如波特率为9600，即一秒种可以传送9600个位数。

校验方式：指奇校验或偶校验或无校验，目的是判断传输过程中是否有错误！它只是用于判断一个字符（比如八个位或是七个位组成一个字符）传输是否有错误。但是它并不能完全能够判断传输是否有错。

  数据位：传输一个字符由几个位组成。

停止位：传输一个字符有几个停止位，用于判断某个字符是否传输结束，以便开始接收下一个字符。

传输1个字符的比特位数量与规定的通讯参数有关：

规定8个数据位，1个校验位（奇或偶校验位），1个停止位，则传输一个字符有1+8+1+1=11个位（第一个1固定表示开始位）。

规定8个数据位，没有校验位，2个停止位，则传输一个字符有1+8+2=11个位（第一个1固定表示开始位）。

规定7个数据位，1个校验位（奇或偶校验位），1个停止位，则传输一个字符有1+7+1+1=10个位（第一个1固定表示开始位）。

规定7个数据位，没有校验位，2个停止位，则传输一个字符有1+7+2=10个位（第一个1固定表示开始位）。

**关于奇偶校验：**

奇校验就是看2进制数的1的个数为奇数 就在校验位填0 偶数就填1 使1的个数变成奇数个，偶校验相反 把1的个数变成偶数，然后传送数据，接收后再做相应的奇偶校验判断是否有位传送错误。

## 3.2 字符时间间隔

通过3.1的说明，可以知道，传输一个字符有11个bit或10个bit，那么3.5个字符就是38.5个bit或35个bit，设定波特率为X，则传输3.5个字符所需要的时间T为：T=38.5/(X/1000)毫秒或T=35/(X/1000)毫秒。

“至少3.5个字符传输时间”包括了开始和结束。

**如果在连续发送字符的过程中，中间停顿时间超过T毫秒，接收方就认为已经发送完了这组消息，开始处理了，这是“至少3.5个字符传输时间的结束”。**

**如果发送的太连续，下一组消息与上一组消息之间的间隔时间没超过T毫秒，接收方就认为这些字符是一组消息，按一组消息去处理。所以，您发送结束一组命令后，必须间隔T毫秒才能发送下一组命令，这是“至少3.5个字符传输时间的开始”。**

一般为了简单起见，可以将传输45 bit的时间四舍五入后的整型值作为两个数据帧之间的时间间隔，并以此来判断报文接收的完整性。

## 3.3 关于传输浮点数的问题

因为一般浮点数都是32位，而Modbus总线中只能传输最多16位的数据，在Modbus中，利用两个整形数传送一个浮点数（即将一个32位的二进制数分割成两个16位的二进制数），传送完以后，再将两个整形数整合为一个浮点数。

特别注意的是，在Modbus总线中，数据传输时是低位字节先传送，传送过来后如果不进行交换，二进制的数值排列是反向的，即原来的低字节在高位，高字节在低位，所以得到的数据和原数据相差甚远。

# 4 模块详细设计

## 4.1 底层应用模块设计

这里的底层应用模块是指与采集设备通讯的模块，是《现场工控系统软件》所用数据的来源，结合《现场工控系统软件概要设计手册》中的“数据流图”，从整体上来看该系统与采集设备分为两种情况，一种是通过串口与采集设备通讯，另一种是通过TCP长连接与采集设备通讯。

### 4.1.1 通讯参数模块

不管是通过串口进行数据交换还是通过TCP进行数据交换，还是web管理通过http进行数据交换，系统必须包含配置文件，配置文件中可以手动指定实际需要的参数。

结合“数据流图”和Modbus数据传输协议，从大的方面来看，配置文件包含底层硬件通讯参数、web层应用参数。为了简单明了，本系统规定配置文件一律采用json文件。

#### 4.1.1.1 底层硬件通讯参数

底层硬件通讯参数顶层通讯参数一定是通讯方式，以0代表串口通讯，以1代表TCP通讯，默认1。

同样是顶层配置参数的还有主从模式、定时间隔、数据类型、量程下限、量程上限、线性系数、修正截距、污染源类型、监测因子个数。

对于主从模式，以1代表主动模式，以0代表被动模式，默认1。

对于定时间隔，其单位是毫秒，如果是主动模式，必须指定，但指定了也不见得就被采用，系统会根据波特率、校验方式、数据位、停止位处理并判断，如果指定的时间间隔太小，系统将会使用默认值60\*1000作为定时间隔。

对于数据类型，以0代表电流值，1代表电压值，2代表实测值，实测值即无须转换直接存入数据可的数据，默认电流值。

对于量程下限，默认4，不管是电流值还是电压值，如果读取的数据是实测值，量程上下限失去作用，系统将忽略这两个设定。

对于量程上限，默认20，不管是电流值还是电压值。

对于线性系数，默认1。

对于修正截距，默认0。

对于污染源类型，严格要求按照《污染物在线监控（监测）系统数据传输标准(HJ 212-2017代替HJT 212-2005)》中“6.6.1.2 系统编码方法”指定的编码。

对于监测因子个数，默认8。

在程序启动时中，通过参数配置处理工具直接读取json文件，再一一获取对应的参数即可。

##### 4.1.1.1.1 串口通讯参数

串口通讯参数包括**COM口**、**主从模式**、**波特率，检验方式，数据位，停止位，定时间隔。**

**COM口：**必须是手动有效指定，不然系统无法正确取到数据，默认com1。

**波特率：**必须指定，与定时间隔有关，默认9600，这从“3.5个字符时间间隔”可得知。

**检验方式**：必须指定，与定时间隔有关，以0表示偶校验，1表示奇校验，2表示无校验，默认2。

**数据位**：必须指定，与定时间隔有关，以0表示7位数据位，1表示8位数据位，默认1。

**停止位**：必须指定，与定时间隔有关，以1表示1位停止位，2表示2位停止位，默认2，与校验方式必须对应。

##### 4.1.1.1.2 TCP通讯参数

TCP通讯参数包括通讯IP、端口。配置文件中默认的IP为127.0.0.1，端口是502。

#### 4.1.1.2 web管理配置参数

web管理配置参数一般是启动的服务端口，可能还有其他的缓存配置参数。

### 4.1.2 串口与采集设备通讯

根据“数据流图”，不管被动接收还是主动接收，都是接收动作，因而抽象出一个接收的接口作为接口层，之后通过被动、主动两个实体结构实现这个接口层。接口中包含接收、发送、打包、解包的方法。

根据Modbus的协议有RTU和ASCII传输模式，那么两个实体结构也就对应于RTU和ASCII了，每一个实体类有各自的校验方法。

此外，因为交互的数据都是16进制或者二进制数据，不是常用的10进制数据，因此还有数据进制转换公用工具。

同时，通讯中有功能码、错误码，所以需要定义公共常量。

### 4.1.3 TCP长连接与采集设备通讯

TCP长连接独立一个实体结构，但也是利用4.1.2中的接口、常量、公用工具。因为TCP/IP数据传输有特殊的情况，即“3个标识1个长度”，所以在实体结构中定义内部变量。

## 4.2套接字模块设计

套接字模块关键之处就是一个永久运行的TCP服务器，利用go编程语言的TCP相关编程手段，可以极速实现这个模块。

## 4.3数据转换模块设计

数据转换模块包括两部分，第一部分是处理从寄存器读出来的将16进制数据转为10进制或者将10进制数据处理成16进制数据写入寄存器的算法，另一部分是将从寄存器读出来的已经过处理的10进制数据按照主要污染物浓度计算公式生成的算法。

### 4.3.1 进制转换算法设计

从数据流图中可以看到，将数据接回来后是要处理的，因为接回来的数据是电流值或者电压值或者实测值。

数据转换算法的目的就是将电流或电压转化为实测值，但这个实测值并不知道具体是什么监测因子的实测值，它在数据库中只有一个代号，需要web管理页面赋予含义。

转换算法分为两部分，一部分是将读到的地址16进制整数转为10进制整数，另一部分是将读到的16进制整数电流或电压值转为10进制的浮点数，或将16进制整数的实测值转为10进制的浮点数。

既然有16进制转为10进制，肯定也有10进制转为16进制，甚至中间还有过渡的2进制与16进制或10进制数据的相互转换。

转换算法直接设计为一个实体结构，其包含上述的数据转换功能即可。

### 4.3.2 计算公式算法设计

计算公式参照《污染物在线监控（监测）系统数据传输标准(HJ 212-2017代替HJT 212-2005)》附录中的计算公式，并严格遵循计量单位的一致即可。

## 4.4 数据分发模块

数据分发模块主要指将数据发送到实时曲线、将数据保存到mysql数据库，将报警信息发送到指定工作人员，还有一个可选的数据发送----将数据发送到环保局数据中心或者企业内部中控系统。

实时曲线利用redis的订阅发布、socket-io.js、go-socketio来实现。

数据保存到mysql数据库将是一个很复杂的过程，经过接入数据、数据转换、数据衍生三个大层面的处理，最终才保存到数据库。

发送数据到环保局数据中心或者企业内部中控系统，这按照一定的协议将数据打包成协议规定的格式，再利用TCP/IP传输，将数据包发出去即可。

## 4.5 定时任务模块设计

定时任务主要是定时统计生成分、时、天、月、季、年的数据，利用go语言的定时包即可实现。

## 4.6 协议转换模块设计

协议转换在本系统里是指将modbus协议转为GB212协议或者企业内部自定义的中控数据包协议，根据协议文档，将接收回来的数据按照格式打包即可。

## 4.7 报警处理模块设计

报警处理模块就是建立判断标准，是每个排放因子的排放标准，判断接收回来的数据里每个排放因子是否超过了指定的标准，超过了则系统后台利用短信或者微信公众号发送报警信息到相关的工作人员手机，提醒他，下位机发生了状况。

## 4.8 web应用模块设计

上层应用模块包括PC端的应用模块，移动端的应用模块。

### 4.8.1 PC端的应用模块

PC端应用是基于电脑浏览器的web管理系统、web动态曲线系统，并且web管理系统设计的兼容目标仅限于Chrome浏览器，Firefox浏览器与Chrome浏览器兼容性不存在很大的问题，故web管理系统也可以用Firefox浏览器打开，但是不考虑IE浏览器以及其他小众的浏览器。

#### 4.8.1.1 web动态曲线系统

web动态曲线系统是单页面系统，页面仅仅是各个监测因子的实时曲线，具体来说就是将下位机的采集到的实时数据通过web页面的曲线图还原出来。实时曲线不考虑手机端的，只兼容PC端的浏览器，因为实时数据要求的性能和系统资源比较多，而手机的资源有限。

动态曲线以时间为x轴，监测因子的数据为y轴，每隔10秒中刷新一次。曲线图以监测设备为最小单位，每台设备自动生成一个曲线图，每个曲线图中共有若干条曲线（多少条取决于监测多少个因子，包括排放因子、某些排放参数）。

动态曲线图不设置查看权限，所有人都可以打开浏览器查看。

#### 4.8.1.2 web管理系统

web管理系统集成了单点登陆、历史数据查询、报表导出、图表显示、统计分析、报警信息模板定制、系统操作日志记录参数配置的功能。

### 4.8.2 移动端应用

移动端应用指apk应用和基于移动端浏览器的web app应用。因为移动端的屏幕尺寸原因，移动端应用的功能不可设计的非常复杂，越简单越好。按照这个理念，移动端的主要功能是查询数据，查看图形图表，发送短信报警。

# 5 数据库详细设计

系统数据的存储来自2个部分，一个web应用的（移动端，PC端），另一个是底层应用的（采集回来的数据）。上层应用的数据表包括用户表、权限表、角色表、因子表、系统配置表等，底层应用的数据表包括历史数据表，就一个，但表名是动态的，与年份有关。

因为设计的系统具有通用性，即不分废水、废气、噪音等监测工程都适用，下面就以废气的设计路线进行解读。系统的数据源头是下位机的寄存器，寄存器读取到的数据要保存在数据标记代号字段中，数据标记代号又要与监测因子存在映射关系。

## 5.1 底层应用数据库设计

a寄存器地址自由分配，具体是什么，上位机程序不关心，只要确保上位机将要读取的地址与下位机提供的可读地址一致即可。

b 要事先商议好从下位机读出的数据是模拟量还是数字量，这个条件决定了是否需要进行模拟量转数字量的过程。

c 如果判定是模拟量，则将读到的16进制数合成10进制数，再将10进制数按照转换算法转换为数字量，然后将该寄存器地址与转换后的数字量建立映射关系，如[0x40001:12.45]。

d 如果判定是数字量，则将读到的16进制数合成10进制数，然后将该寄存器地址直接与数字量建立映射关系，如[0x40001:12.45]。

e 在web管理页面配置好寄存器地址与dataX的映射关系，寄存器地址是任意指定的，但dataX个数是指定的，它对应与监测因子的个数，这样做的目的是不管寄存器地址是什么，总有一个dataX与寄存器地址对应。

f 在web管理页面配置好寄存器地址与监测因子的映射关系，如[0x40001:a21024]，[0x40001:a21026]，一个寄存器地址可以与任意监测因子建立映射关系，但一个寄存器地址只能映射一个监测因子，这样做的目的是不管因子是什么，总有一个因子对应某个寄存器地址。

g 经过两次web页面的设置，隐含地将dataX与监测因子建立了映射关系，根据这个隐式的映射关系，判断监测因子是否是排放因子、全局因子，进而计算排放因子的衍生字段值和流量字段值。

一个烟气监测系统中，流量字段有小时工况流量(hour\_wcf，单位m3/h)、小时标况流量(hour\_scf，单位m3/h)、日排放流量(day\_scf，单位m3/d),这几个流量都与烟气流速有关系。

h 排放因子的衍生字段有工况浓度(dataX，单位mg/m3)、标况浓度(dataX\_ssc，单位mg/m3)、干基浓度(dataX\_dbc，单位mg/m3)、折算浓度(dataX\_cnc，单位mg/m3)、分钟均值浓度(dataX\_mavg，单位mg/m3)、小时均值浓度(dataX\_havg，单位mg/m3)、日均值浓度(dataX\_davg，单位mg/m3)、小时排放速率(dataX\_her，单位kg/h)、日排放速率(dataX\_der，单位kg/d)。

i 计算过程中所涉及的浓度均是干基浓度。

j 综合考虑，这个烟气历史数据表里应有的字段如下：

dataid,datatime,hour\_wcf,hour\_scf,day\_scf,dataX,dataX\_flag,dataX\_ssc,dataX\_dbc,dataX\_cnc,dataX\_mavg,dataX\_havg,dataX\_davg,dataX\_her,dataX\_der。

k 需要指出的是：如果dataX是排放参数，那么在实测数据的记录中,dataX\_mavg,dataX\_havg,dataX\_davg是无数据的，因为这些是统计数据，dataX\_her,dataX\_der也是没数据的，因为只有排放因子才有排放率的说法，而dataid,datatime,hour\_wcf,hour\_scf,day\_scf不管什么类型的因子都是存在的。

如a~k的介绍，烟气监测对应的底层应用数据表基本形成，字段可能还漏了一些，比如某个统计时间段内的排放因子的最大值、最小值。

其他废水监测、噪音监测等也可以类比这样的设计，但因为废水监测中并没有折算值的说法，噪音监测中没有排放率、流量的概念等等，具体如何，应该根据污染源类型扩展监测因子的衍生字段，从《污染物在线监控（监测）系统数据传输标准(HJ 212-2017代替HJT 212-2005)》中的“主要要污染物计算方法”可以查得到。

## 5.2 web应用数据库设计

web应用数据库

# 6 系统技术路线方案

整套系统基于B/S构架，编程语言采用golang，用到的技术包括golang TCP/IP技术，golang串口编程技术，go-redis技术，go-socketio技术，go-mysql技术，beego快速开发框架，go-cron定时任务，golang接口设计，golang结构体设计，这些都是系统后台用到的技术，为前端提供API接口。

前端技术包括了移动端和PC端，系统按照前后端分离的模式开展。PC端采用Vue.js实现界面UI，移动端apk采用MUI实现，移动端web app采用BUI web app快速实现。