Modbus 协议

2011-06-02

简介

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。它已经成为一通用工业标准。有了它,不同厂商生产的控制设备可以连成工业网络,进行集中监控。此协议定义了一个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。它制定了消息域格局和内容的公共格式。　　当在一 Modbus 网络上通信时,此协议决定了每个控制器须要知道它们的设备地址,识别按地址发来的消息,决定要产生何种行动。如果需要回应,控制器将生成反馈信息并用 Modbus 协议发出。在其它网络上,包含了 Modbus 协议的消息转换为在此网络上使用的帧或包结构。这种转换也扩展了根据具体的网络解决节地址、路由路径及错误检测的方法。

在 Modbus 网络上转输

标准的 Modbus 口是使用一 RS-232C 兼容串行接口,它定义了连接口的针脚、电缆、信号位、传输波特率、奇偶校验。控制器能直接或经由 Modem 组网。　　控制器通信使用主—从技术,即仅一设备（主设备）能初始化传输（查询）。其它设备（从设备）根据主设备查询提供的数据作出相应反应。典型的主设备：主机和可编程仪表。典型的从设备：可编程控制器。　　主设备可单独和从设备通信,也能以广播方式和所有从设备通信。如果单独通信,从设备返回一消息作为回应,如果是以广播方式查询的,则不作任何回应。Modbus 协议建立了主设备查询的格式：设备（或广播）地址、功能代码、所有要发送的数据、一错误检测域。　　从设备回应消息也由 Modbus 协议构成,包括确认要行动的域、任何要返回的数据、和一错误检测域。如果在消息接收过程中发生一错误,或从设备不能执行其命令,从设备将建立一错误消息并把它作为回应发送出去。

在其它类型网络上转输

在其它网络上,控制器使用对等技术通信,故任何控制都能初始和其它控制器的通信。这样在单独的通信过程中,控制器既可作为主设备也可作为从设备。提供的多个内部通道可允许同时发生的传输进程。
在消息位,Modbus 协议仍提供了主—从原则,尽管网络通信方法是“对等”。如果一控制器发送一消息,它只是作为主设备,并期望从从设备得到回应。同样,当控制器接收到一消息,它将建立一从设备回应格式并返回给发送的控制器。

查询—回应周期

（1）查询　　查询消息中的功能代码告之被选中的从设备要执行何种功能。数据段包含了从设备要执行功能的任何附加信息。例如功能代码 03 是要求从设备读保持寄存器并返回它们的内容。数据段必须包含要告之从设备的信息：从何寄存器开始读及要读的寄存器数量。错误检测域为从设备提供了一种验证消息内容是否正确的方法。

（2）回应　　如果从设备产生一正常的回应,在回应消息中的功能代码是在查询消息中的功能代码的

回应。数据段包括了从设备收集的数据：象寄存器值或状态。如果有错误发生,功能代码将被修改以用于指出回应消息是错误的,同时数据段包含了描述此错误信息的代码。错误检测域允许主设备确认消息内容是否可用。

两种传输方式

控制器能设置为两种传输模式（ASCII 或 RTU）中的任何一种在标准的 Modbus 网络通信。用户选择想要的模式,包括串口通信参数（波特率、校验方式等）,在配置每个控制器的时候,在一个 Modbus 网络上的所有设备都必须选择相同的传输模式和串口参数。 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域和如何解码。 在其它网络上（象 MAP 和 Modbus Plus）Modbus 消息被转成与串行传输无关的帧。

1、ASCII 模式 当控制器设为在 Modbus 网络上以 ASCII（美国标准信息交换代码）模式通信,在消息中的每个 8Bit 字节都作为两个 ASCII 字符发送。这种方式的主要优点是字符发送的时间间隔可达到 1 秒而不产生错误。 代码系统 · 十六进制,ASCII 字符 0...9,A...F · 消息中的每个 ASCII 字符都是一个十六进制字符组成 每个字节的位 · 1 个起始位 · 7 个数据位,最小的有效位先发送 · 1 个奇偶校验位,无校验则无 CRC 域是两个字节,包含一 16 位的二进制值。它由传输设备计算后加入到消息中。接收设备重新计算收到消息的 CRC,并与接收到的 CRC 域中的值比较,如果两值不同,则有误。 CRC 是先调入一值是全“1”的 16 位寄存器,然后调用一过程将消息中连续的 8 位字节各当前寄存器中的值进行处理。仅每个字符中的 8Bit 数据对 CRC 有效,起始位和停止位以及奇偶校验位均无效。

CRC 产生过程中,每个 8 位字符都单独和寄存器内容相或（OR）,结果向最低有效位方向移动,最高有效位以 0 填充。LSB 被提取出来检测,如果 LSB 为 1,寄存器单独和预置的值或一下,如果 LSB 为 0,则不进行。整个过程要重复 8 次。在最后一位（第 8 位）完成后,下一个 8 位字节又单独和寄存器的当前值相或。最终寄存器中的值,是消息中所有的字节都执行之后的 CRC 值。

CRC 添加到消息中时,低字节先加入,然后高字节。

CRC 简单函数如下： unsigned short CRC16(puchMsg, usDataLen) unsigned char *puchMsg ; /* 要进行 CRC 校验的消息 */
unsigned short usDataLen ; /* 消息中字节数 */
{
unsigned char uchCRCHi = 0xFF ; /* 高 CRC 字节初始化 */
unsigned char uchCRCLo = 0xFF ; /* 低 CRC 字节初始化 */
unsigned uIndex ; /* CRC 循环中的索引 */
while (usDataLen--) /* 传输消息缓冲区 */
{
uIndex = uchCRCHi ^ *puchMsgg++ ; /* 计算 CRC */
uchCRCHi = uchCRCLo ^ auchCRCHi[uIndex} ;
uchCRCLo = auchCRCLo[uIndex] ;
}
return (uchCRCHi << 8 uchCRCLo) ;

```
}
/* CRC 高位字节值表 */
static unsigned char auchCRCHi[] = {
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,  0x80, 0x41, 0x00,
0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,  0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1,
0x81, 0x40, 0x01, 0xC0,  0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40,  0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,  0x81, 0x
40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,  0x00, 0xC1, 0x81, 0x40, 0x
01, 0xC0, 0x80, 0x41, 0x00, 0xC1,  0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0x
C0, 0x80, 0x41,  0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
  0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,  0x01, 0xC0, 0x80,
 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,  0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
 0x00, 0xC1, 0x81, 0x40,  0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01,
 0xC0,  0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,  0x01, 0
xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,  0x80, 0x41, 0x01, 0xC0, 0
x80, 0x41, 0x00, 0xC1, 0x81, 0x40,  0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0
x41, 0x01, 0xC0,  0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,  0x80, 0x41, 0x
00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,  0x01, 0xC0, 0x80, 0x41, 0x00, 0x
C1, 0x81, 0x40, 0x01, 0xC0,  0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x
81, 0x40,  0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,  0x81,
 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,  0x00, 0xC1, 0x81, 0x40,
 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,  0x80, 0x41, 0x00, 0xC1, 0x81, 0x40
} ;
/* CRC 低位字节值表*/  static char auchCRCLo[] = {  0x00, 0xC0, 0xC1, 0x01, 0x
C3, 0x03, 0x02, 0xC2, 0xC6, 0x06,  0x07, 0xC7, 0x05, 0xC5, 0xC4, 0x04, 0xCC, 0
x0C, 0x0D, 0xCD,  0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09,
    0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A,  0x1E, 0xDE, 0
xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC, 0x14, 0xD4,  0xD5, 0x15, 0xD7, 0x17, 0x16,
 0xD6, 0xD2, 0x12, 0x13, 0xD3,  0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1,
 0x33, 0xF3,  0xF2, 0x32, 0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4,  0x
3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A,  0x3B, 0xFB, 0x39, 0x
F9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29,  0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x
2F, 0xEF, 0x2D, 0xED,  0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x
26,  0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60,  0x61, 0xA1,
 0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67,  0xA5, 0x65, 0x64, 0xA4, 0x6C,
 0xAC, 0xAD, 0x6D, 0xAF, 0x6F,  0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9,
 0xA8, 0x68,  0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA, 0xBE, 0x7E,  0
```

x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5, 0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0, 0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92, 0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C, 0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B, 0x99, 0x59, 0x58, 0x98, 0x88, 0x48, 0x49, 0x89, 0x4B, 0x8B, 0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C, 0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83, 0x41, 0x81, 0x80, 0x40 } ;

ModBus 网络是一个工业通信系统,由带智能终端的可编程序控制器和计算机通过公用线路或局部专用线路连接而成。其系统结构既包括硬件、亦包括软件。它可应用于各种数据采集和过程监控。表 1 是 ModBus 的功能码定义。

表 1 ModBus 功能码

01 READ COIL STATUS

02 READ INPUT STATUS

03 READ HOLDING REGISTER

04 READ INPUT REGISTER

05 WRITE SINGLE COIL

06 WRITE SINGLE REGISTER

15 WRITE MULTIPLE COIL

16 WRITE MULTIPLE REGISTER

ModBus 网络只是一个主机,所有通信都由他发出。网络可支持 247 个之多的远程从属控制器,但实际所支持的从机数要由所用通信设备决定。采用这个系统,各 PC 可以和中心主机交换信息而不影响各 PC 执行本身的控制任务。

（1）ModBus 的传输方式

在 ModBus 系统中有 2 种传输模式可选择。这 2 种传输模式与从机 PC 通信的能力是同等的。选择时应视所用 ModBus 主机而定,每个 ModBus 系统只能使用一种模式,不允许 2 种模式混用。一种模式是 ASCII（美国信息交换码）,另一种模式是 RTU（远程终端设备）这两种模式的定义(见表 3)

表 3 ASCII 和 RTU 传输模式的特性

ASCII 可打印字符便于故障检测,而且对于用高级语言（如 Fortan）编程的主计算机及主 PC 很适宜。RTU 则适用于机器语言编程的计算机和 PC 主机。 用 RTU 模式传输的数据是 8 位二进制字符。如欲转换为 ASCII 模式,则每个 RTU 字符首先应分为高位和低位两部分,这两部分各含 4 位,然后转换成十六进制等量值。用以构成报文的 ASCII 字符都是十六进制字符。ASCII 模式使用的字符虽是 RTU 模式的两倍,但 ASCII 数据的译玛和处理更为容易一些,此外,用 RTU 模式时报文字符必须以连续数据流的形式传送,用 ASCII 模式,字符之间可产生长达 1s 的间隔,以适应速度较快的机器。

（2）ModBus 的数据校验方式

CRC-16（循环冗余错误校验）

CRC-16 错误校验程序如下：

报文（此处只涉及数据位,不指起始位、停止位和任选的奇偶校验位）被看作是一个连续的二进制,其最高有效位（MSB）首选发送。报文先与 X↑16 相乘（左移 16 位）,然后看 X↑16+X↑15+X↑2+1 除,X↑16+X↑15+X↑2+1 可以表示为二进制数 11000000000000101。整数商位忽略不记,16 位余数加入该报文（MSB 先发送）,成为 2 个 CRC 校验字节。余数中的 1 全部初始化,以免所有的零成为一条报文被接收。经上述处理而含有 CRC 字节的报文,若无错误,到接收设备后再被同一多项式（X↑16+X↑15+X↑2+1）除,会得到一个零余数（接收设备核验这个 CRC 字节,并将其与被传送的 CRC 比较）。全部运算以 2 为模（无进位）。

习惯于成串发送数据的设备会首选送出字符的最右位（LSB-最低有效位）。而在生成 CRC 情况下,发送首位应是被除数的最高有效位 MSB。由于在运算中不用进位,为便于操作起见,计算 CRC 时设 MSB 在最右位。生成多项式的位序也必须反过来,以保持一致。多项式的 MSB 略去不记,因其只对商有影响而不影响余数。

生成 CRC-16 校验字节的步骤如下：

①装如一个 16 位寄存器,所有数位均为 1。

②该 16 位寄存器的高位字节与开始 8 位字节进行"异或"运算。运算结果放入这个 16 位寄存器。

③把这个 16 寄存器向右移一位。

④若向右（标记位）移出的数位是 1,则生成多项式 1010000000000001 和这个寄存器进行"异或"运算;若向右移出的数位是 0,则返回③。

⑤重复③和④,直至移出 8 位。

⑥另外 8 位与该十六位寄存器进行"异或"运算。

⑦重复③~⑥,直至该报文所有字节均与 16 位寄存器进行"异或"运算,并移位 8 次。

⑧这个 16 位寄存器的内容即 2 字节 CRC 错误校验,被加到报文的最高有效位。　　另外,在某些非 ModBus 通信协议中也经常使用 CRC16 作为校验手段,而且产生了一些 CRC16 的变种,他们是使用 CRC16 多项式 X↑16+X↑15+X↑2+1,单首次装入的 16 位寄存器为 0000;使用 CRC16 的反序 X↑16+X↑14+X↑1+1,首次装入寄存器值为 0000 或 FFFFH。　　LRC（纵向冗余错误校验）　　LRC 错误校验用于 ASCII 模式。这个错误校验是一个 8 位二进制数,可作为 2 个 ASCII 十六进制字节传送。把十六进制字符转换成二进制,加上无循环进位的二进制字符和二进制补码结果生成 LRC 错误校验（参见图）。这个 LRC 在接收设备进行核验,并与被传送的 LRC 进行比较,冒号（：）、回车符号（CR）、换行字符（LF）和置入的其他任何非 ASCII 十六进制字符在运算时忽略不计。