第一章
Modbus 协议

- ☐ 介绍 Modbus 协议介绍
- ☐ 两种串行传输模式
- ☐ 信息帧
- ☐ 错误检查方法

Modbus 协议介绍

Modbus 可编程控制器之间可相互通讯，也可与不同网络上的其他设备进行通讯，支撑网络有 Modicon 的 Modbus 和 Modbus+工业网络。网络信息存取可由控制器内置的端口，网络适配器以及 Modicon 提供的模块选件和网关等设备实现，对 OEM(机械设备制造商)来说，Modicon 可为合作伙伴提供现有的程序，可使 Modbus+网络紧密地集成到他们的产品设计中去。

Modicon 的各种控制器使用的公共语言被称为 Modbus 协议，该协议定义了控制器能识别和使用的信息结构。当在 Modbus 网络上进行通讯时，协议能使每一台控制器知道它本身的设备地址，并识别对它寻址的数据，决定应起作用的类型，取出包含在信息中的数据和资料等，控制器也可组织回答信息，并使用 Modbus 协议将此信息传送出去。

在其他网络上使用时，数据包和数据帧中也包含着 Modbus 协议。如，Modbus+或 MAP 网络控制器中有相应的应用程序库和驱动程序，实现嵌入式 Modbus 协议信息与此网络中用子节点设备间通讯的特殊信息帧的数据转换。

该转换也可扩展，处理节点地址，路由，和每一个特殊网络的错误检查方法。如包含在 Modbus 协议中的设备地址，在信息发送前就转换成节点地址，错误检查区也用于数据包，与每个网络的协议一致，最后一点是需用 Modbus 协议，写入嵌入的信息，定义应处理的动作。

图 1 说明了采用不同通讯技术的多层网络中设备的互连方法。在信息交换中，嵌入到每个网络数据包中的 Modbus 协议，提供了设备间能够交换数据的公共的语言。

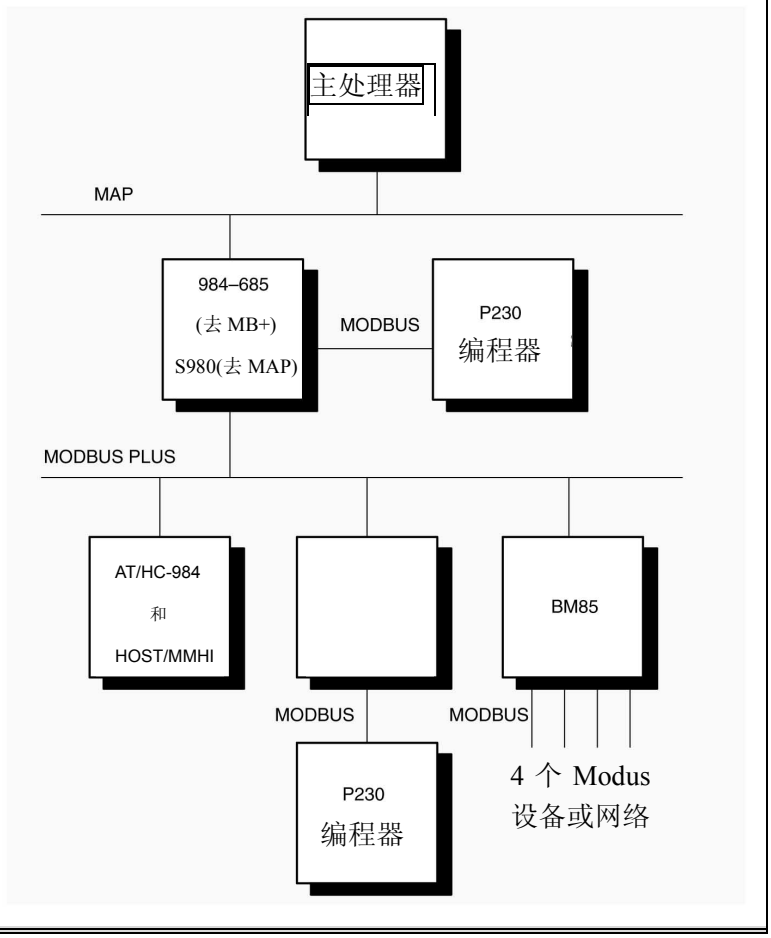

图 1：Modbus 协议应用示意图

* MB+为 Modbus

Modbus 上的数据传输

Modicon 控制器上的标准 Modbus 端口是使用一个 RS-232 兼容的串行接口，定义了连接器，接线电缆，信号等级，传输波特率，和奇偶校验，控制器可直接或通过调制解调器(以后简称 Modems)接入总线(网络)。控制器通讯使用主从技术，即主机能起动数据传输，称查询。而其它设备(从机)应返回对查询作出的响应，或处理查询所要求的动作。典应的主机设备应包括主处理器和编程器。典应的从机包括可编程控制器。

主机可对各从机寻址，发出广播信息，从机返回信息作为对查询的响应。从机对于主机的广播查询，无响应返回 Modbus 协议报据设备地址，请求功能代码，发送数据，错误校验码，建立了主机查询格式，从机的响应信息也用 Modbus 协议组织，它包括确认动作的代码，返回数据和错误校验码。若在接收信息时出现一个错误或从机不能执行要求的动作时，从机会组织一个错误信息。并向主机发送作为响应。

在其它总线上传输数据

除标准的 Modbus 功能外，有些 Modcon 控制器内置端口或总线适配器，在 Modbus+总线上实现通讯或使用网络适配器，在 MAP 网络上通讯。

在这些总线上，控制器间采用对等的技术进行通讯，即任意一个控制器可向其它控制器启动数据传送。因此，一台控制器既可作为从机，也可作为主机，常提供多重的内部通道，允许并列处理主机和从机传输数据

在信息级，尽管网络通讯方法是对等的，但 Modbus 协议仍采用主从方式，若一台控制器作为主机设备发送一个信息，则可从一台从机设备返回一个响应，类似，当一台控制器接受信息时，它就组织一个从机设备的响应信息，并返回至原发送信息的控制器。

查询响应周期：

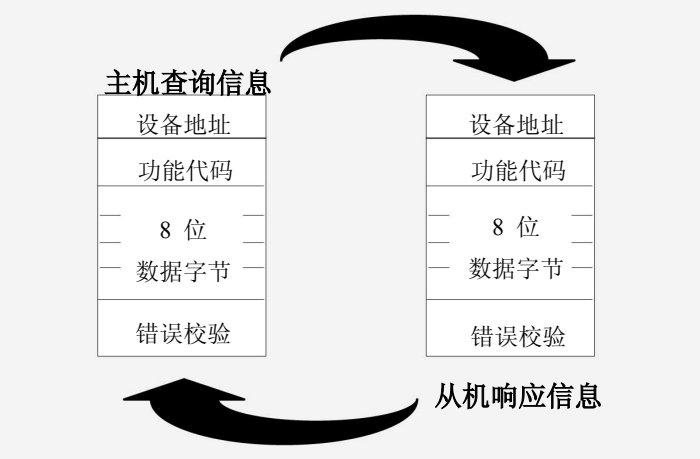

图 2：主从查询响应周期

查询：

查询中的功能代码为被寻址的从机设备应执行的动作类型。数据字节中包含从机须执行功能的各附加信息，如功能代码 03 将查询从机，并读保持寄存器。并用寄存器的内容作响应。该数据区必须含有告之从机读取寄存器的起始地址及数量，错误校验区的一些信息，为从机提供一种校验方法，以保证信息内容的完整性。

响应：

从机正常响应时，响应功能码是查询功能码的应答，数据字节包含从机采集的数据，如寄存器值或状态。如出现错误，则修改功能码，指明为错误响应。并在数据字节中含有一个代码，来说明错误，错误检查区允许主机确认有效的信息内容。

两种串行传输模式

控制器可使用 ASCII 或 RTU 通讯模式，在标准 Modbus 上通讯。在配置每台控制器时，用户须选择通讯模式以及串行口的通讯参数。(波特率，奇偶校验等)，在 Modbus 总线上的所有设备应具有相同的通讯模式和串行通讯参数。

选择 ASCII 或 RTU 模式用于标准的 Modbus 总线。它定义了总线上串行传输信息区的“位”的含义，决定信息打包及解码方法。
如在 MAP 和 Modbus+总线上时，Modbus 信息以帧的方式出现，并与串行传输无关，如请求读保持寄存器可以在 Modbus+上的两个控制器之间处理，而与使用的控制器的 Modbus 端口无关。

ASCII 模式

当控制器以 ASCII 模式在 Modbus 总线上进行通讯时，一个信息中的每 8 位字节作为 2 个 ASCII 字符传输的，这种模式的主要优点是允许字符之间的时间间隔长达 IS，也不会出现错误。

ASCII 码每一个字节的格式：

编码系统： 16 进制，ASCII 字符 0-9,A-F　　1 个 16 进制
数据位： 1 起始位
7 位数据，低位先送
奇/偶校验时 1 位；无奇偶校验时 0 位
（LRC）1 位带校验 1 停止位；无校验 2 止位
错误校验区： 纵向冗余校验

RTU 模式

控制器以 RTU 模式在 Modbus 总线上进行通讯时，信息中的每 8 位字节分成 2 个 4 位 16 进制的字符，该模式的主要优点是在相同波特率下其传输的字符的密度高于 ASCII 模式，每个信息必须连续传输。

RTU 模式中每个字节的格式：
编码系统：8 位二进制，十六进制 0-9，A-F

数据位： 1 起始位
8 位数据，低位先送
奇/偶校验时 1 位；无奇偶校验时 0 位
停止位 1 位(带校验);停止位 2 位(无校验)
带校验时 1 位停止位；无校验时 2 位停止位

错误校验区：循环冗余校验(CRC)

Modbus 信息帧

何论是 ASCII 模式还是 RTU 模式，Modbus 信息以帧的方式传输，每帧有确定的起始点和结束点，使接收设备在信息的起点开始读地址，并确定要寻址的设备 (广播时对全部设备)，以及信息传输的结束时间。可检测部分信息，错误可作为一种结果设定。

对 MAP 或 Modbus+协议可对信息帧的起始和结束点标记进行处理，也可管理发送至目的地的信息，此时，信息传输中 Modbus 数据帧内的目的地址已无关紧要，因为 Modbus+地址已由发送者或它的网络适配器把它转换成网络节点地址和路由。

ASCII 帧

在 ASCII 模式中，以(:)号(ASCII3AH)表示信息开始，以回撤一换行键(CRLF) (ASCII OD 和 OAH)表示信息结束。

对其它的区，允许发送的字符为 16 进制字符 0-9,A-F。网络中设备连续检测并接收一个冒号(:)时，每台设备对地址区解码，找出要寻址的设备。

字符之间的最大间隔为 1S,若大于 1S，则接收设备认为出现了一个错误。
典型的信息帧见下表

开始	地址	功能	数据	纵向冗余检查	结束
1 字符	2 字符	2 字符	n 字符	2 字符	2 字符

:					

图 3 ASCII 信息帧

例外：对于 584 和 984A/B/X 控制器，一个 ASCII 信息可在 LRC 区后正常终止，而不需发送 CRLF 字符，此时出现>IS 的时间间隔，控制器也将认为是正常中断。

RTU 帧

RTU 模式中，信息开始至少需要有 3.5 个字符的静止时间，依据使用的波特率，很容易计算这个静止的时间(如下图中的 T1-T2-T3-T4)。接着，第一个区的数据为设备地址。

各个区允许发送的字符均为 16 进制的 0-9,A-F。

网络上的设备连续监测网络上的信息，包括静止时间。当接收第一个地址数据时，每台设备立即对它解码，以决定是否是自己的地址。发送完最后一个字符号后，也有一个 3.5 个字符的静止时间，然后才能发送一个新的信息。

整个信息必须连续发送。如果在发送帧信息期间，出现大于 1.5 个字符的静止时间时，则接收设备刷新不完整的信息，并假设下一个地址数据。

同样一个信息后，立即发送的一个新信息，（若无 3。5 个字符的静止时间）这将会产生一个错误。是因为合并信息的 CRC 校验码无效而产生的错误。

开始	地址	功能	数据	校验	终止
T1-T2-T3-T4	8 B 位 S	8 B 位 S	N×8 B 位 S	16B 位 S	T1-T2-T3T-4

图 4 RTU 信息帧

Modbus 信息帧(Continued)

地址设置

信息地址包括 2 个字符(ASCII)或 8 位(RTU)，有效的从机设备地址范围 0-247,(十进制)，各从机设备的寻址范围为 1-247。主机把从机地址放入信息帧的地址区，并向从机寻址。从机响应时，把自己的地址放入响应信息的地址区，让主机识别已作出响应的从机地址。

地址 0 为于广播地址，所有从机均能识别。当 Modbus 协议用于高级网络时，则不允许广播或其它方式替代。如 Modbus+使用令牌循环，自动更新共享的数据库。

功能码设置

信息帧功能代码包括字符(ASCII)或 8 位(RTU)。有效码范围 1-225(十进制)，其中有些代码适用全部型号的 Modicon 控制器，而有些代码仅适用于某些型号的控制器。还有一些代码留作将来使用，有关功能代替码的设置将在第 2 章说明。

当主机向从句发送信息时，功能代码向从机说明应执行的动作。如读一组离散式线圈或输入信号的 ON/OFF 状态，读一组寄存器的数据，读从机的诊断状态，写线圈（或寄存器），允许下截、记录、确认从机内的程序等。当从机响应主机时，功能代码可说明从机正常响应或出现错误(即不正常响应)，正常响应时，从句简单返回原始功能代码；不正常响应时，从机返回与原始代码相等效的一个码，并把最高有效位设定为“1”。

如，主机要求从机读一组保持寄存器时，则发送信息的功能码为：

0000 0011 (十六进制 03)

若从机正确接收请求的动作信息后，则返回相同的代码值作为正常响应。发现错时，则返回一个不正常响信息：

1000 0011(十六进制 83)

从机对功能代码作为了修改，此外，还把一个特殊码放入响应信息的数据区中，告诉主机出现的错误类型和不正常响应的原因。主机设备的应用程序负责处理不正常响应，典型处理过程是主机把对信息的测试和诊断送给从机，并通知操作者。

数据区的内容

数据区有 2 个 16 进制的数据位，数据范围为 00-FF(16 进制)，根据网络串行传输的方式，数据区可由一对 ASCII 字符组成或由一个 RTU 字符组成。

主机向从机设备发送的信息数据中包含了从机执行主机功能代码中规定的请求动作，如离散量寄存器地址，处理对象的数目，以及实际的数据字节数等。

举例说明，若主机请求从机读一组寄存器（功能代码 03），该数据规定了寄存器的起始地址，以及寄存器的数量。又如，主机要在一从机中写一组寄存器，（则功能代码为 10H）。该数据区规定了要写入寄存区的起始地址，寄存器的数量，数据的字节数，以及要写入到寄存器的数据。

若无错误出现，从机向主机的响应信息中包含了请求数据，若有错误出现，则数据中有一个不正常代码，使主机能判断并作出下一步的动作。

数据区的长度可为“零”以表示某类信息，如，主机要求-从机响应它的通讯事件记录（功能代码 OBH）。此时，从机不需要其他附加的信息，功能代码只规定了该动作。

信息帧

错误校验

标准 Modbus 总线，有两类错误检查方法，错误检查区的内容按使用的错误检查方法填写。

SDCII

使用 ASCII 方式时，错误校验码为 2 个 ASCII 字符，错误校验字符是 LRC 校验结果。校验时，起始符为（:）冒号结束符为 CRLF 字符。

RTU

使用 RTU 方式时，错误校验码为一个 16 位的值，2 个 8 位字节。错误校验值是对信息内容执行 CRC 校验结果。CRC 校验信息帧是最后的一个数据，得到的校验码先送低位字节，后送高位字节，所以 CRC 码的高位字节是最后被传送的信息。

串行传送信息

在标准的 Modbus 上传送的信息中，每个字符或字节，按由左向右的次序传送：

最低有效位：（LSB）最高有效位：（MSB）

ASCII 数据帧位序：

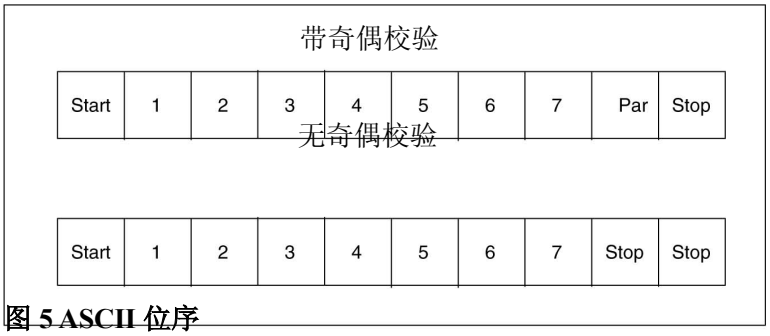

图 5 ASCII 位序

RTU 数据帧位序:

图 6 RTU 位序

错误校验方法

标准的 Modbus 串行通讯网络采用两种错误校验方法，奇偶校验(奇或偶)可用于校验每一个字符，信息帧校验(LRC 或 CRC)适用整个信息的校验，字符校验和信息帧校验均由主机设备产生，并在传送前加到信息中去。从机设备在接收信息过程中校验每个字符和整个信息。

主机可由用户设置的一个预定时间间隔，确定是否放弃传送信息。该间隔应有足够的时间来满足从机的正常响应。若主机检测到传输错误时，则传输的信息无效。从机不再向主机返回响应信息。此时，主机会产生一个超时信息，并允许主机程序处理该错误信号。注意：主机向实际并未存在的从机发送信息时也会引起超时出错信号。

在 MAP 或 Modbus+等其它网骆上使用时，采用比 Modbus 更高一级的数据帧校验方法。在这些网络中，不再运用 Modbus 中的 LRC 或 CRC 校验方法。当出现发送错误时，网络中的通讯协议通知发送设备有错误出现，并允许根据设置的情况，重试或放弃信息发送。若信息已发送，但从机设备未作响应，则主机通过程序检查后发出一个超时错误。

奇偶校验

用户可设置奇偶校验或无校验，以此决定每个字符发送时的奇偶校验位的状态。何论是奇或偶校验，它均会计算每个字符数据中值为“1”的位数，ASCII 方式为位数据；RTU 方式为 8 位数据。并根据“1”的位数值(奇数或偶数)来设定为“0”或“1”

如一个 RTU 数据帧中 8 位数据位为：

1100 0101

在该帧中，值为“1”的总位数为 4，即偶数。如采用奇校验方式时，则 “1”的总位数为奇数，即 5。

发送信息时，计算奇偶位，并加到数据帧中，接收设备统计位值为“1”的数量，若与该设备要求的不一致时产生一个错误。在 Modbus 总线上的所有设备必须采用相同的奇偶校验方式。

注意：奇偶校验只能检测到数据帧在传输过程中丢失奇数“位”时才产生的错误。如采用奇数校验方式时，一个包含 3 个“1”位的数据丢失 2 个“1”位时，其结果仍然是奇数。若无奇偶校验方式时，传输中不作实际的校验，应附加一个停止位。

LRC 校验

ASCII 方式时，数据中包含错误校验码，采用 LRC 校验方法时，LRC 校验信息以冒号“:”开始，以 CRLF 字符作为结束。它忽略了单个字符数据的奇偶校验的方法。

LRC 校验码为 1 个字节，8 位二进制值，由发送设备计算 LRC 值。接收设备在接收信息时计算 LRC 校验码。并与收到的 LRC 的实际值进行比较，若二者不一致，亦产生一个错误。

在梯形图中，CKSM 函数可计算数据信息中 LRC 的校验。用于主计算机时请查阅附录 C 中的一个实例，它详细说明 LRC 的校验的过程。

错误校验方法

CRC 校验

RTU 方式时，采用 CRC 方法计算错误校验码，CRC 校验传送的全部数据。它忽略信息中单个字符数据的奇偶校验方法。

CRC 码为 2 个字节，16 位的二进制值。由发送设备计算 CRC 值，并把它附到信息中去。接收设备在接收信息过程中再次计算 CRC 值并与 CRC 的实际值进行比较，若二者不一致，亦产生一个错误，校验开始时，把 16 位寄存器的各位都置为“1”，然后把信息中的相邻 2 个 8 位字节数据放到当前寄存器中处理，只有每个字符的 8 位数据用于 CRC 处理。起始位，停止位和校验位不参与 CRC 计算。

CRC 校验时，每个 8 位数据与该寄存器的内容进行异或运算，然后向最低有效位(LSB)方向移位，用零填入最高有效位(MSB)后，再对 LSB 检查，若 LSB=1，则寄存器与预置的固定值异或，若 LSB=0，不作异或运算。

重复上述处理过程，直至移位 8 次，最后一次(第 8 次)移位后，下一个 8 位字节数据与寄存器的当前值异或，再重复上述过程。全部处理完信息中的数据字节后，最终得到的寄存器值为 CRC 值。

CRC 值附加到信息时，低位在先，高位在后。在梯形图中，CKSM 函数计算信息中的 CRC 值。用于主计算机时，可查阅附录 C 中的一个实例，它详细说明了 CRC 的校验。

第二章
数据和控制功能

- ❒ Modbus 功能代码格式
- ❒ Modbus 功能代码总结

❒ Modbus 功能代码说明

数字值表达

若无特殊说明在此节文中用＋进制值表示，图中的数据区则用十六进制表示。

Modbus 信息中的数据地址

Modbus 信息中的所有数据地址以零作为基准，各项数据的第一个数据地址的编号为 0 如：

- ❒ 在可编程控制器中“coil 1”在 Modbus 信息中其地址值表示为 0000
- ❒ Coil 127(十进制)在 Modbus 信息中则为 007EH(126 十进制)
- ❒ 保持寄存器 40001，在信息中数据地址为寄存器 0000。功能代码区为保持寄存器类型规定的操作，因此，“4XXXX”是缺省的地址类型。
- ❒ 保持寄存器 40108 寻址寄存器地址为 006B hex(＋进制 107)

Modbus 信息中区内容

图 7 为一个例子，说明了 Modbus 的查询信息，图 8 为正常响应的例子，这两例子中的数据均是 16 进制的，也表示了以 ASCLL 或 RTU 方式构成数据帧的方法。主机查询是读保持寄存器，被请求的从机地址是 06，读取的数据来自地址从 40108 至 40110 3 个保持寄有器。注意，该信息规定了寄存器的起始地址为 0107 (006BH)。

从机响应返回该功能代码，说明是正常响应，字节数“Byle count”中说明有多少个 8 位字节被返回。因无论是 ASCII 方式还是 RTU 方式，它表明了附在数据区中 8 位字节的数量。ASCII 方式时，字节数为数据中 ASCII 字符实际数的一半，每 4 个位的 16 进制值需要一个 ASCII 字符表示，因此在数据中应由 2 个 ASCII 字符来表示一个 8 位的字节。

如 RTU 方式时，63H 用一个字节(01100011)发送，而用 ASCII 方式时，发送需 2 个字节，即 ASCII“6”(0110110)和 ASCII“3”(0110011)。8 个位为一个单位计算“字节数”，它忽略了信息帧用(ASCII 或 RTU)组成的方法。

字节数使用方法：当在缓冲区组织响应信息时，“字节数”区域中的值应与该信息中数据区的字节数相等。

QUERY

Field Name	Example (Hex)	ASCII Characters	RTU 8-Bit Field
Header		:(colon)	None
Slave Address	06	06	0000 0110
Function	03	03	0000 0011
Starting Address Hi	00	00	0000 0000
Starting Address Lo	6B	6B	0110 1011
No. of Registers Hi	00	00	0000 0000
No. of Registers Lo	03	03	0000 0011
Error Check		LRC (2 chars.)	CRC (16 bits)
Trailer		CR LF	None
	Total Bytes:	17	8

图 8 说明“字节数”区在一个贡型响应中的应用。

RESPONSE

Field Name	Example (Hex)	ASCII Characters	RTU 8-Bit Field
Header		:(colon)	None
Slave Address	06	0 6	0000 0110
Function	03	0 3	0000 0011
Byte Count	06	0 6	0000 0110
Data Hi	02	0 2	0000 0010
Data Lo	2B	2 B	0010 1011
Data Hi	00	0 0	0000 0000
Data Lo	00	0 0	0000 0000
Data Hi	00	0 0	0000 0000
Data Lo	63	6 3	0110 0011
Error Check		LRC (2 chars.)	CRC (16 bits)
Trailer		CR LF	None
	Total Bytes:	23	11

图 8：从机采用 ASCII/RTU 方式响应

Modbus+数据内容

在 Modbus+网络发送的 Modbus 信息应需嵌入到 LLC (逻辑连接控制)级数据帧，Modbus 信息区由 8 位字节的数据组成，类似于 RTU 中的信息组成。

由发送设备把从机地址转换成 Modbus+路由地址，CRC 数据不在 Modbus 信息中发送，因为会在更高级的数据链路控制层(HDLC)中进行 CRC 校验。

其余的信息与原标准格式一致，应用软件(控制器中的 MSTR 或主机中的 Modcom III)可将这些信息帧组成数据包。

图 9 示例说明了如何将读寄存器值的请求嵌入到＋Modbus 网络的数据帧中。

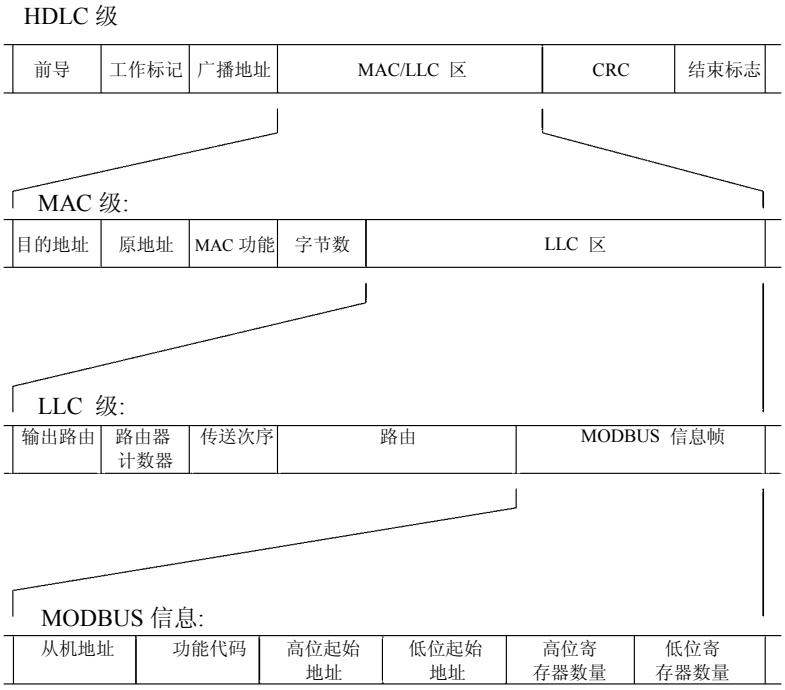

图 9：Modbus+数据内容

控制器支持的功能代码

下表列出 Modicon 控制器支持的功能代码：以十进制表示。

“Y”表示支持“N”表示不支持。

代码	名称	384	484	584	884	M84	984
01	读线圈状态	Y	Y	Y	Y	Y	Y
02	读输入状态	Y	Y	Y	Y	Y	Y
03	读线保持寄存器	Y	Y	Y	Y	Y	Y
04	读输入寄存器	Y	Y	Y	Y	Y	Y
05	强制单个线圈	Y	Y	Y	Y	Y	Y
06	预置单个寄存器	Y	Y	Y	Y	Y	Y
07	读不正常状态	Y	Y	Y	Y	Y	Y
08	诊断(见第 3 章)						
09	程序 484	N	Y	N	N	N	N
10	查询 484	N	Y	N	N	N	N
11	通讯事件控制	Y	N	Y	N	N	Y
12	通讯事件记录	Y	N	Y	N	N	Y
13	程序控制器	Y	N	Y	N	N	Y
14	查询控制器	Y	N	Y	N	N	Y
15	强制多个寄存器	Y	Y	Y	Y	Y	Y
16	预置多个寄存器	Y	Y	Y	Y	Y	Y
17	报告从机 ID	Y	Y	Y	Y	Y	Y
18	程序 884/M84	N	N	N	Y	Y	N

19	通讯链路复位	N	N	N	Y	Y	N
20	读通用参考值	N	N	Y	N	N	Y
21	写通用参考值	N	N	Y	N	N	Y
22	Mask Write 4X Register	N	N	N	N	N	(1)
23	Read/Write 4X Registers	N	N	N	N	N	(1)
24	Read FIFO 队列	N	N	N	N	N	(1)

（1）功能代码仅由 984-785 控制器支持

01 读线圈状态

描述

读从机离散量输出口的 ON/OFF 状态，不支持广播。附录 B 列出由不同控制器型号支持最大的参数清单。

查询

查询信息规定了要读的起始线圈和线圈量，线圈的起始地址为零，1-16 个线圈的寻址地址分为 0-15。例：请求从机设备 17 读 20-56 线圈。

QUERY

Field Name	Example (Hex)
Slave Address	11
Function	01
Starting Address Hi	00
Starting Address Lo	13
No. of Points Hi	00
No. of Roints Lo	25
Error Check (LRC or CRC)	—

图 10: 读线圈状态－查询

响应

响应信息中的各线圈的状态与数据区的每一位的值相对应，

1=ON; 0=OFF。第一个数据字节的 LSB 为查询中的寻址地址，其他的线圈按顺序在该字节中由低位向高位排列，直至 8 个为止，下一个字节也是从低位向高位排例。

若返回的线圈数不是 8 的倍数，则在最后的数据字节中的剩余位至字节的最高位全部填零，字节数区说明全部数据的字节数。

RESPONSE

Field Name	Example (Hex)
Slave Address	11
Function	01
Byte Count	05
Data (Coils 27-20)	CD
Data (Coils 35-28)	6B
Data (Coils 43-36)	B2
Data (Coils 51-44)	0E
Data (Coils 56-52)	1B
Error Check (LRC or CRC)	—

图 11: 读线圈状态－响应

线圈 27-20 的状态用 CDH 表示，二进制值为 11001101，该字节的 MCB 为线圈 27，LSB 为 20。线圈从左(27)向右(20)状态分别为 ON-ON-OFF-OFF-ON-ON-OFF-ON，因此第一个字节中的线圈从左到右应是 27-20。下一个字节的线圈应为 35 至 28。位数据串行转输从低位到高位，即 20…27，28…35。

最后一个数据字节中，56-52 线圈的状态为 1BH(或二进制 00011011)，线圈 56 是左数第 4 位，线圈 52 是该字节的最低位，所线圈 56 至 52 的状态分别为 ON-ON-OFF-ON-ON 注意 3 个剩余位(至最高位的数)全部填 0。

02 读输入位状态

说明

读从机离散量输入信号的 ON/OFF 状态。不支持广播。附录 B 列出各种型号控制器所支持的最大参数量。

查询

查询信息规定了要读的输入起始地址，以及输入信号的数量。输入起始地址为 0，1-16 个输入口的地址分别为 0-15。
例：请求读从机设备 17 的 10197-10218 的输入位状态。

QUERY	
Field Name	Example (Hex)
Slave Address	11
Function	02
Starting Address Hi	00
Starting Address Lo	C4
No. of Points Hi	00
No. of Roints Lo	16
Error Check (LRC or CRC)	—

图 12：读输入位状态－查询。

响应

响应信息中的各输入口的状态，分别对应于数据区中的每一位值，1 = ON; 0 = OFF，第一个数据字节的 LSB 为查询中的寻址地址，其他输入口按顺序在该字节中由低位向高位排列，直至 8 个位为止。下一个字节中的 8 个输入位也是从低位到高位排列。

若返回的输入位数不是 8 的倍数，则在最后的数据字节中的剩余位直至字节的最高位全部填零。字节的最高位，字节数区。说明了全部数据的字节数。
例：对查询作出响应(参见上页)。

RESPONSE	
Field Name	Example (Hex)
Slave Address	11
Function	02
Byte Count	03
Data (Inputs 10204-10197)	AC
Data (Inputs 10212-10205)	DB
Data (Inputs 10218-10213)	35
Error Check (LRC or CRC)	—

图 13：读输入位状态－响应。

输入位 10204-10197 的状态用 35H (或二进制 00110101) 表示。输入位 10218 为左数第 3 位，10213 输入位为 LSB，输入位 10218-10213 的状态分别为 ON-ON-OFF-ON-OFF-ON，注意最位还有 2 个剩余位需填零。

03 读保持寄存器

说明

读从机保持寄存器的二进制数据不支持广播，附录 B 列出了由各种型号控制器所支持的最大的参数量

查询

查询信息规定了要读的寄存器起始地址及寄存器的数量，寄存器寻址起始地址为 0000，寄存器 1-16 所对应的地址分别为 0-15

QUERY	
Field Name	Example (Hex)
Slave Address	11
Function	03
Starting Address Hi	00
Starting Address Lo	6B
No. of Points Hi	00
No. of Roints Lo	03
Error Check (LRC or CRC)	—

图 14：读保持寄存器-查询

响应

响应信息中的寄存器数据为二进制数据，每个寄存器分别对应 2 个字节，第一个字节为高位值数据，第二个字节为低位数据。

对 984-X8X 型控制器(如 984-685 等)，扫描数据的速率为每次 125 个寄存器。对其它控制器型号的扫描速率为每次 32 个寄存器，全部数据完成组合后返回响应信息。
例按查询要求返回响应。

RESPONSE	
Field Name	Example (Hex)
Slave Address	11
Function	03
Byte Count	06
Data Hi (Register 40108)	02
Data Lo(Register 40108)	2B
Data Hi(Register 40109)	00
Data Lo(Register 40109)	00
Data Hi(Register 40110)	00
Data Lo(Register 40110)	64
Error Check (LRC or CRC)	—

图 15：读寄存器-响应

寄存器 40108 的数据用 022BH 2 个字节(或用十进制 555)表示，寄存器 40109-40110 中的数据为 0000 和 0064H，(十进制时为 0 和 100)

04 读输入寄存器

说明

读从机输入寄存器(3X 类型)中的二进制数据，不支持广播

附录 B 列出了由各种型号控制器所支持的最大的参数量

查询

查询信息规定了要读的寄存器的起始地址及寄存器的数量，寻止起始地址为 0，寄存器 1-16 所对应的地址分别为 0-15。

例：请求读从机设备 17 中的 30009 寄存器。

QUERY	
Field Name	Example (Hex)
Slave Address	11
Function	04
Starting Address Hi	00
Starting Address Lo	08
No. of Points Hi	00
No. of Roints Lo	01
Error Check (LRC or CRC)	—

图 16：读输入寄存器-查询

响应

响应信息中的寄存器数据为每个寄存器分别对应 2 个字节，第一个字节为高位数据，第二个字节为低位数据。

对 984-X8X 型控制器(如 984-685 等)，扫描数据的速率为每次 125 个寄存器，对其它型号的控制器为每次 32 个寄存器。数据完成组合后，返回响应信息。

例按查询要求返回响应

RESPONSE	
Field Name	Example (Hex)
Slave Address	11
Function	04
Byte Count	02
Data Hi(Register 30009)	00
Data Lo(Register 30009)	0A
Error Check (LRC or CRC)	—

图 17：读寄存器-响应

寄存器 30009 中的数据用 000AH 2 个字节(或用十进制 10)表示

05 强制单个线圈

说明

强制单个线圈(0X 类型)为 ON 或 OFF 状态。广播时，该功能可强制所有从机中同一类型的线圈均为 ON 或 OFF 状态。

※ 注意：该功能可越过控制器内存的保护状态和线圈的禁止状态。线圈强制状态一直保持有效直至下一个控制逻辑作用于线圈为止。控制逻辑中无线圈程序时，则线圈处于强制状态。

附录 B 中列出了由各种型号控制器所支持的最大的参数量。

查询

查询信息规定了需要强制线圈的类型，线圈起始地址为 0，线圈 1 的寻址地址为 0

由查询数据区中的一个常量。规定被请求线圈的 ON/OFF 状态， FF00H 值请求线圈处于 ON 状态，0000H 值请求线圈处于 OFF 状态，其它值对线圈无效，不起作用。

例：强制从机设备 17 中的 173 线圈为 ON 状态

QUERY

Field Name	Example (Hex)
Slave Address	11
Function	05
Coil Address Hi	00
Coil Address Lo	AC
Force Data Hi	FF
Force Data Lo	00
Error Check (LRC or CRC)	—

图 18：强制单个线圈-查询

响应

线圈为强制状态后即返回正常响应

例：按查询要求返回响应

RESPONSE

Field Name	Example (Hex)
Slave Address	11
Function	05
Coil Address Hi	00
Coil Address Lo	AC
Force Data Hi	FF
Force Data Lo	00
Error Check (LRC or CRC)	—

图 19：强制单个线圈

06 预置单个寄存器

说明

把一个值预置到一个 4X 类型保持寄存器中。广播时，该功能把值预置到所有从机的相同类型的寄存器中。

※ **注意：**该功能可越过控制器的内存保护。使寄存器中的预置值保持有效。只能由控制器的下一个逻辑信号来处理该预置值。若控制逻辑中无寄存器程序时，则寄存器中的值保持不变。
附录 B 中列出了各种型号控制器所支持的最大的参数量

查询

查询信息规定了要预置寄存器的类型，寄存器寻址起始地址为 0，寄存器 1 所对应的地址为 0。

请求的预置值在查询数据区，M84 或 484 控制器使用一个 10 位二进制值，其中高 6 位设定为 0，而其它类型的控制器使用 16 位值。
例：请求把从机设备 17 中的 40002 寄存器预置为 0003H 值。

QUERY

Field Name	Example (Hex)
Slave Address	11
Function	06
Register Address Hi	00
Register Address Lo	01
Preset Data Hi	00
Preset Data Lo	03
Error Check (LRC or CRC)	—

图 20：预置单个寄存器-查询

响应

寄存器内容被预置后返回正常响应
例：按查询要求返回响应

RESPONSE

Field Name	Example (Hex)
Slave Address	11
Function	06
Register Address Hi	00
Register Address Lo	01
Preset Data Hi	00
Preset Data Lo	03
Error Check (LRC or CRC)	—

图 21：预置单个寄存器-响应

07 读不正常状态

说明

读从中机中 8 个不正常状态线圈的数据，某些线圈号已在不同型号的控制器中预定义，而其它的线圈由用户编程，作为有关控制器的状态信息，如“machine ON/OFF”，“heads retraced”，(缩回标题)，“safeties satisfied”(安全性满意)，“error conditions”(存在错误条件)或其它用户定义的标志等。该功能码不支持广播。

该功能代码为存取该类信息提供了一种简单的方法，不正常线圈的类型是已知的(在功能代码中不需要线圈类型) 预定义的不正常线圈号如下：

控制器型号	线圈	设定
M84,184/384,584,984	1-8	用户定义
484	257	电池状态
	258-264	用户定义
884	761	电池状态
	762	内存保护状态

763	R10 工况状态
764-768	用户预定义

查询

例请求读从机设备 17 中的不正常状态

QUERY	
Field Name	Example (Hex)
Slave Address	11
Function	07
Error Check (LRC or CRC)	—

图 22：读不正常状态-查询

响应

正常响应包含 8 个不正常的线圈状态，为一个数据字节，每个线圈一位。LSB 对应为最低线圈类型的状态。

例：按查询要求返回响应：

QUERY	
Field Name	Example (Hex)
Slave Address	11
Function	0B
Error Check (LRC or CRC)	—

图 23：读不正常状态—响应

该例子中，线圈数据为 6DH (二进制 0110 ,1101)，从左到右 (最高位至最低位) 的线圈状态分别为: OFF – ON – ON – OFF – ON – ON – OFF – ON。若控制器型号为 984，这些位表示线圈 8 至 1 的状态；若控制器型号为 484 则表示线圈 264 至 257 的状态。

11 (0B Hex) 取通讯事件计数器

说明

由从机通讯事件计数器返回一个状态字和事件数，依据一串信息前后读取的当前数值，由主机决定其信息是否已被从机正常处理，该功能代码不支持广播。

信息成功完成 1 次，使控制器的事件计数器加 1， 不正常响应，查询命令或取事件计数器命令等，对计数值不起作用。

通过诊断功能代码 (08)，(若重起动通讯选择子功能代码 0001) 或计数器和诊断寄存清零器代码 (000A) 等可对事件计数器复位。

查询

例：请求读取从机设备 17 的通讯事件计数器

QUERY	
Field Name	Example (Hex)
Slave Address	11
Function	0B
Error Check (LRC or CRC)	—

图 24：读取通讯事件计数器—查询

响应

正常响应含一个带 2 个字节的状态字和一个双字节的事件数，若从机还未处理完以前发出的程序值状态字中的各位。均为 1 (FFFFH)，处理完时，各位值均为 0(0000H)。

例：按查询要求返回响应

QUERY	
Field Name	Example (Hex)
Slave Address	11
Function	0B
Status Hi	FF
Status Lo	FF
Event Count Hi	01
Event Count Lo	08
Error Check (LRC or CRC)	—

图 25：读取通讯事件计数器—响应

该列子中，状态字是 FFFFH，说明从机还在处理程序，控制器计算的事件数为 264 (0108H)

12 (0C Hex) 读取通讯事件记录

说明

由从机返回一个状态字，事件数，信息数和一个事件的数据区。不支持广播

状态字和事件数与读取通讯事件计数器功能代码（11,OBH）返回值相同。信息计数器包含从机处理，(最后一次再起动，计数器清零操作，或通电)的信息量，该值与由诊断功能代码 (08)，总线信息数子功能代码 (11,0BH) 返回的值相同。事件数据区包含 0-64 个字节。每个字节对应 Modbus 送出的一个状态，或子机接收操作的一个状态。由子机把事件送到顺序排列的区域。字节 0 为最新的事件，最大新的确字节刷新该区域的最老的字节。

查询

例：请求从机设备 17 读取通讯事件记录

QUERY	
Field Name	Example (Hex)
Slave Address	11
Function	0C
Error Check (LRC or CRC)	—

图 26：读取通讯事件记录—查询

响应

正常响应含一个 2 个字节的状态字区，一个 2 个字节的事件数区和一个 2 个字节的信息数区，以及有 0-64 个字节的事件区，一个字节数区定义上述 4 个区的数据的总长度。

例：按查询要求返回响应

RESPONSE	
Field Name	Example (Hex)
Slave Address	11
Function	0C
Byte Count	08
Status HI	00
Status Lo	00
Event Count Hi	01
Event Count Lo	08
Message Count Hi	01
Message Count Lo	21
Event 0	20
Event 1	00
Error Check (LRC or CRC)	—

图 27：读取通讯事件记录—响应

在这例子中状态字为 0000H，说明从机已完成程序处理从机计算的事件数为 264 (0108H)，已处理的信息数为 289 (0121H)。 最近的通讯事件在 Event 0 字节中。数值 20H 表示该从机已最后进入了只听状态 (Listen Only Mode)。

以前的事件在 Event 1 字节中，数值 00H 表示该从机接收了一个通讯再起动事件（Communications Restart），响应事件的字节会在后面叙述。

12 (0C Hex) 读取通讯事件记录

事件字节内容

读取通讯事件记录功能代码返回的一个事件字节可为 4 种类型之一，每一个字节中的高 7 位定义该字节的类型，高 6 位可进一步说明该字节，见下面说明。

从机 Modbus 接收事件

从机接收查询信息时，储存事件字节的类型，并在处理前。储存该事件字节的类型。把高 7 位置 1 定义这事件，如果相应条件“真”（true），则其他位也置 1，字节中各位的含义如下。

Bit	Contents
0	未用
1	通讯错误
2	未用
3	未用
4	字符超限
5	只听模式
6	接收广播
7	1

从机 Modbus 发送事件

从机完成查询信息处理后返回正常（或不正常）响应，或无响应后，储存事件字节的类型。通过高 7 位置“0”，高 6 位置“1”定义事件，若相应的条件“真”（TRUE），则其他位置 “1”。
字节中各位含义如下：

Bit	**Contents**
0	读不正常发送 (不正常功能码 1-3)
1	从机放弃不正常发送 (不正常功能码 4)
2	从机放弃不正常发送 (不正常功能码 从 5-6)
3	从机程序 NAK 不正常发送 (不正常功能码 从 7)
4	从机程序 NAK 不正常发送，出现写入超时错误
5	从机程序 NAK 不正常发送，当前只听方式
6	1
7	0

从机进入只听模式

当从机进入只听模式时储存这类事件字节，事件由 04H 数据定义，如下：

Bit	**Contents**
0	0
1	0
2	1
3	0
4	0
5	0
6	0
7	0

从机初始化通讯再起动

通讯口再起动时由从机储存事件字节的类型，诊断功能代码 (08)再带起动通讯选择子功能代码 (0001) 使从机通讯再起动。该功能可把从机配置成错误时继读方式 (Continue on Error) 或错误时停止方式 (Stop on Error)。若从机配置成“错误时

继读方式”时，则事件字节加到已存在的事件记录中，若配置成“错误时停止”方式时，则把该字节加到记录中，并把其余的记录清零。

Bit	Contents
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0

15 (0F H) 强制多个线圈

说明

按线圈的顺序把各线圈 (0X 类型) 强制成 ON 或 OFF。广播时，该功能代码可对各从机中相同类型的线圈起强制作用。

※ 注意：该功能代码可越过内存保护和线圈的禁止状态线圈。保持强制状态有效，并只能由控制器的下一个逻辑来处理。若无线圈控制逻辑程序时，线圈将保持强制状态。

附录 B 中列出了各种型号控制器所支持的最大参数量。

查询

查询信息规定了被强制线圈的类型，线圈起始地址为 0，线圈 1 寻址地址为 0。

查询数据区规定了被请求线圈的 ON/OFF 状态，如数据区的某位值为“1”表示请求的相应线圈状态为 ON，位值为“0”，则为 OFF 状态。

下述例子为请求从机设备 17 中一组 10 个线圈为强制状态，起始线圈为 20 (则寻址地址为 19 或 13H)，查询的数据为 2 个字节，CD01H (二进制 11001101 0000 0001) 相应线圈的二进制位排列如下：

Bit:	1	1	0	0	1	1	0	1		0	0	0	0	0	0	0	1
Coll:	27	26	25	24	23	22	21	20	-	-	-	-	-	-	-	29	28

传送的第一个字节 CDH 对应线圈为 27-20, LSB 对应线圈 20，传送的第二个字节为 01H，对应的线圈为 29-28，LSB 为继圈 28，其余未使用的位均填“0”。

QUERY

Field Name	Example (Hex)
Slave Address	11
Function	0F
Coil Address Hi	00
Coil Address Lo	13
Quantity of Coils Hi	00
Quantity of Coils Lo	0A
Byte Count	02
Force Data Hi (Coils 27-20)	CD
Force Data Lo (Coils 29-28)	01
Error Check (LRC or CRC)	—

图 28：强制多个线圈 - 查询

响应

正常响应返回从机地址，功能代码，起始地址以及强制线圈数

例：对上述查询返回的响应

RESPONSE	
Field Name	Example (Hex)
Slave Address	11
Function	0F
Coil Address Hi	00
Coil Address Lo	13
Quantity of Coils Hi	00
Quantity of Coils Lo	0A
Error Check (LRC or CRC)	—

图 29：强制个多个圈—响应

16(10 Hex)预置多个寄存器

说明

把数据按顺序预置到各 (4×类型) 寄存器中，广播时该功能代码可把数据预置到全部从机中的相同类型的寄存器中。

※ 注意：该功能代码可越过控制器的内存保护，在寄存器中的预置值一直保持有效，只能由控制器的下一个逻辑来处理寄存器的内容，控制逻辑中无该寄存器程序时，则寄存器中的值保持不变。
附录 B 中列出了各种型号控制器所支持的最大参数量。

查询

信息中规定了要预置的寄存器类型，寄存器寻址的起始地址为 0，寄存器 1 寻址地址为 0。查询数据区中指定了寄存器的预置值，M84 和 484 型控制器使用 10 位二进制数据，2 个字节，剩余的高 6 位置 0。而其他类型的控制器使用一个 16 位二进制数据，每个寄存器 2 个字节。

例：请求在从机设备 17 中的 2 个寄存器中放入预置值，起始寄存器为 40002 ，预置值为 00 0AH 和 01 02H。

QUERY	
Field Name	Example (Hex)
Slave Address	11
Function	10
Starting Address Hi	00
Starting Address Lo	01
No. of Registers Hi	00
No. of Registers Lo	02
Byte Count	04
Data Hi	00
Data Lo	0A
Data Hi	01
Data Lo	02
Error Check (LRC or CRC)	—

图 30：预置多个寄存器

响应

正常响应返回从机地址，功能代码和起始地址和预置寄存器数。
例：按查询要求返回响应

PESPONSE	
Field Name	Example (Hex)
Slave Address	11
Function	10
Starting Address Hi	00
Starting Address Lo	01
No. of Registers Hi	00
No. of Registers Lo	02
Error Check (LRC or CRC)	—

图 31：预置多个寄存器—响应

17(11 Hex)报告从机 ID

说明

返回一个从机地址控制器的类型，从机的当前状态，以及有关从机的其他说明，不支持广播。

查询

例：请求报告从机设备 17 的 ID 和状态

QUERY	
Field Name	Example (Hex)
Slave Address	11
Function	11
Error Check (LRC or CRC)	—

图 32：报告从机 ID—查询

响应

正常响应格式见图 33，数据内容对应每台控制器的类型。
图 33 报告从机 ID—响应

从机 ID 总结

数据区第一个字节为 Modicon 控制器返回的从机 ID

Slave ID	Controller
0	Micro 84
1	484
2	184/384
3	584
8	884
9	984

17(11 Hex)报告从机 ID

184/384

控制器返回一个字节数 4 或 74 (4AH)，若控制器 J347 Modbus 从机接口已建立，内部 PIB 表正常，则字节数为 74，其他情况字节数为 4。

经常返回的 4 个字节是：

Byte	Contents
1	Slave ID (2 为 184/384)
2	运行指示器状态 (0 = OFF, FF = ON)
3,4	状态字
	Bit 0 = 0
	Bit 1 = 内存保护状态 (0 = OFF, 1 = ON)
	Bit 2,3 = 控制器类型：Bit 2 = 0 和 Bit 3 = 0 指示 184
	Bit 2 = 1 和 Bit 3 = 0 指示 384
	Bits 4-15 = 未用

J347 正确建立，PIB 表正常时返回附加的 70 个字节:

Byte	Contents
5,6 PIB	起始地址
7,8	控制器系列号
9,10	执行 ID
	字节 11-74 含 PIB 表，只有控制器工作，数据在有效 (字节 2)，PIB 表如下：
11,12	最大的输出线圈量
13,14	输出线圈允许表
15,16	输入线圈/运行表的地址
17,18	输入线圈量
19,20	输入线圈允许表
21,22	第一个获取数 (16 倍数)
23,24	最后一个获取数 (16 的倍数)
25,26	输入寄存器的地址
27,28	输入寄存器数量
29,30	输出和保持寄存器数量
31,32	用户逻辑地址
33,34	输出线圈 RAM 表地址
35,36	功能禁止屏蔽
37,38	扩展功能路由地址
39,40	数据传输路由
41,42	
43,44	未用
45,46	功能禁止屏蔽
47,48	A 模式历史表格地址
49,50	DX 打印机请求表
51,52	顺序组数量
53,54	顺序映像表
55,56	顺序 RAM 地址
57,58	50XX 寄存器数量
59,60	50XX 表地址
61,62	输出线圈 RAM 映像地址
63,64	输入 RAM 映像地址
65,66	延时输出起动组

67,68	延时输出结束组
69,70	看门狗线
71,72	自锁 RAM 地址
73,74	延时输出组数量

17 (11 Hex) 报告从机 ID

584

584 控制器返回一个字节数为 9 的信息如下：

Byte	Contents
1	从机 ID 号为 3
2	运行指示器状态 (0 = OFF, FF = 0N)
3	4K 0 页内存
4	1K 状态 RAM
5	用户逻辑段数量
6,7	机器状态字 (配置表字 101,65H) 字的构成如下： Byte 6: Bit 15 = 建立端口 1 Bit 14 = 建立端口 2 Bit 13 = 设定端口 1 地址 Bit 12 = 设定端口 2 地址 Bit 11 = 未用 Bit 10 = 恒定扫描状态 (0 = OFF, 1 = ON) Bit 9 = 单次扫描状态 (0 = OFF, 1 = ON) Bit 8 = 16/24-bit 节点 (0 = 24-bit 节点，1 = 16-bit 节点) *Byte 7:* Bit 7 = (MSB of byte 7) = 通电 ON (1 = ON, 不应为 0) Bit 6 = RUN 运行指示器状态 (0 = ON, 1 = OFF) Bit 5 = 内存保护状态 (0 = ON, 1 = OFF) Bit 4 = Battery OK (0 = OK, 1 = 不 OK) 电池 Bits 3-0 = 未用

8,9 机器停止码 (配置表格字 105,69H)
字的构成如下：

Byte 8:
Bit 15 (MSB) = 外设口停止 (可控停止)
Bit 14 = 未用
Bit 13 =
Bit 12 = 不合法的外设干涉
Bit 11 =
Bit 10 =
Bit 9 = 状态 RAM 测试失败
Bit 8 = 不停止逻辑检测或

Byte 9:
Bit 7 = (MSB) = 看门狗定时器终
Bit 6 = 实时时钟错误
Bit 5 = CPU 诊断失败
Bit 4 = 无效的
Bit 3 = 无效节点类型
Bit 2 = 逻辑检查出错
Bit 1 = 备用检查出错
Bit 0 = 不合法配置

17 (11 Hex) 报告从机 ID

984 型控制器

984 型控制器返回的 ID 数据共 9 个字节，如下：

Byte	Contents
1	控制器为 984 型时从机 ID 号为 9
2	运行指示器状态 (0 = OFF, FF = 0N)
3	4K 0 页内存
4	1K 状态 RAM
5	用户逻辑段数量
6,7	机器状态码 (配置表字 101,65H) 字的结构如下：

Byte 6:
Bit 15(MSB) = 未赋值
Bit 14-11 = 未赋值
Bit 10 = 恒定扫描状态 (0 = OFF, 1 = ON)
Bit 9 = 单次扫描状态 (0 = OFF, 1 = ON)
Bit 8 = 16/24-bit
Bit 10 = 恒定扫描状态 (0 = OFF, 1 = ON)
Bit 9 = 单次扫描状态 (0 = OFF, 1 = ON)
Bit 8 = 16/24-bit 节点 (0 = 24-位节点，1 = 16-位节点)

Byte 7:
Bit 7 (MSB) = 通电 (1 = ON, 不应为 0)
Bit 6 = 运行指示器状态 (0 = ON；1 = OFF)
Bit 5 = 内存保护状态(0 = ON；1 = OFF)
Bit 4 = 电池 OK (0 = OK, 1 = 不 OK)
Bit 3-1 = 未赋值
Bit 0 = 内存减少标志（0=不减少；1=减少）

Bit 0 = 内存减少标志

内存减少：机器状态字以配置表中的字 99,100 和 175 (63H, 64H 和 AFH) 定义使用内存下降值。若 bit0 = “1”下降值计算如下：

0 页（16 位字）= 字 99*4096)-(字 175 低字节*16)

状态表 (16 位字) = (字 100*1024)-(字 175 高字节 *16)

8,9 机器停止码 (配置表字 105,69H)

字的结构：

Byte 8:

Bit 15 (MSB) = 外设端口停止 (可控停止)

Bit 14 = (984A, B, X) = 扩展内存部分损坏

Bit 13 =

Bit 12 = 不合法的外设干涉

Bit 11 =

Bit 10 = 起动节点不能起动段

Bit 9 = 状态 RAM 测试失败

Bit 8 = 逻辑检测死循环或数据段数不正确

Byte 9:

Bit 7(MSB 9) = 看门狗定时器结束

Bit 6 = 实时时钟出错

Bit 5(984A, B, X) = CPU 诊断失败

Bit 5(984) = 使用表线圈不正确

Bit 4 = S908 远程 IO 标题坏

Bit 3 = 无效节点类型

Bit 2 = 逻辑检查出错

Bit 1 = 运行方式下禁止线圈

Bit 0 = 不合法配置

20(14Hex)读通用类型寄存器

说明

返回扩展内存文件中的 6X 类型寄存器内容，不支持广播。

该功能代码能读多组类型寄存器，组别之间的地址可分开，但组内的地址必须连续。

查询

查询信息包括，标准的 Modbus 从机地址，功能代码，字节数，以及错误检查区域。查询信息中还指定要读取的寄存器组或组的地址类型，每一组由一个单独的“子请求”区定义，它包括 7 个字节：

- 寄存器类型：1 字节，(指定为 6X 类型)
- 扩展内存文件号：2 字节(1-10 或 0001-000AH)
- 文件中寄存器起始地址：2 个字节
- 要读取的寄存器数量：2 个字节

要读的寄存器数量与预期响应的其它数据字节加在一起，不能超过 Modbus 所允许的 256 个字节的长度。

有效的扩展内存文件的数量取决于从机的配置和的安装的内存量，除最后一个外，其它的每个文件均含 10000 个寄存器，寻址地址为 0000-270FH(十进制 0000-9999)。

※ **注意：**6X 类型扩展寄存器的寻址地址与 4X 类型的保持寄存器不同。

扩展寄存器起始寻址地址为寄存器“0”(600000)

保持寄存器起始寻址地址为寄存器“1”(40001)

带扩展寄存器的 984-758 控制器中，最后文件中的最后一个(最高的)寄存器见下面 2 个表格。

984-785 带 AS-M785-032 内存：

用户逻辑	表态 RAM	扩展内存容量	最后文件	最后寄存器(十进制)
32K	32K	0	0	0
16K	64K	72K	8	3727

984-785 带 AS-M785-032 内存：

用户逻辑	表态 RAM	扩展内存容量	最后文件	最后寄存器(十进制)
48K	32K	24K	3	4575
32K	64K	96K	10	8303

带扩展寄存器的其它控制器，最后文件中的最后一个寄存器(最高位)如下：

扩展内存容量	最后文件数	最后一个寄存器(十进制)
16K	2	6383
32K	4	2767
64K	7	5535
96K	10	8303

20(14Hex) Read 读通用类型寄存器

例：请求读从机设备 17 中的两组类型寄存器。

1 组包括文件 4 的 2 个寄存器，寄存器起始地址 0001
2 组包括文件 3 的 2 个寄存器，寄存器起始地址 0009

QUERY Field Name	Example (Hex)
Slave Address	11
Function	14
Byte Count	0E
Sub–Req 1, Reference Type	06
Sub–Req 1, File Number Hi	00
Sub–Req 1, File Number Lo	04
Sub–Req 1, Starting Addr Hi	00
Sub–Req 1, Starting Addr Lo	01
Sub–Req 1, Register Count Hi	00
Sub–Req 1, Register Count Lo	02
Sub–Req 2, Reference Type	06
Sub–Req 2, File Number Hi	00
Sub–Req 2, File Number Lo	03
Sub–Req 2, Starting Addr Hi	00
Sub–Req 2, Starting Addr Lo	09
Sub–Req 2, Register Count Hi	00
Sub–Req 2, Register Count Lo	02
Error Check (LRC or CRC)	—

图 34：读通用类型寄存器

响应

正常响应为一串“子响应”，每一个“子响应”对应每个“子请求”，字节数区的值为所有“子响应”字节数之和，此外每个“子响应”中有一个区说明本身的字节数。

RESPONSE

Field Name	Example (Hex)
Slave Address	11
Function	14
Byte Count	0C
Sub–Res 1, Byte Count	05
Sub–Res 1, Reference Type	06
Sub–Res 1, Register Data Hi	0D
Sub–Res 1, Register Data Lo	FE
Sub–Res 1, Register Data Hi	00
Sub–Res 1, Register Data Lo	20
Sub–Res 2, Byte Count	05
Sub–Res 2, Reference Type	06
Sub–Res 2, Register Data Hi	33
Sub–Res 2, Register Data Lo	CD
Sub–Res 2, Register Data Hi	00
Sub–Res 2, Register Data Lo	40
Error Check (LRC or CRC)	—

图 35：读通用类型寄存器

21(15Hex)写通用类型寄存器

说明

在 6X 类型寄存器中，扩展内存文件，不支持广播。
该功能代码能写多组类型寄存器，组别之间地址可分开，但组力寄存器的地必须连续。

查询

查询信息，包括标准的 Modbus 从机地址，功能代码，字节数，以及错误校验区，查询信息还指定要写入的寄存器组或组的地址，每一组由单独的“子请求”区定义，它包括 7 个字节：

- 寄存器类型：1 字节，指定为 6X 类型
- 扩展内存文件号：2 字节(1-10 或 0001-000AH)
- 文件写入寄存器中的起始地址：2 个字节
- 寄存器数量：2 个字节
- 要写入的数据，每一个寄存器 2 个字节

写入文件需要的寄存器数，加上查询的其他数据其总长度不能超过 Modbus 所允许的 256 个字节。

有效的扩展内存文件数取决于从机控制器内存的扩展容量，除最后一个文件外，其它的每个文件均含 10,000 寄存器，寻址地址为 0000-270FH(十进制 0000-9999)

※ **注意：**6X 类型的扩展寄存器与 4X 类型的保持寄存器的寻址方式不同。

6X 类型的扩展寄存器寻址起始地址为 0。

4X 类型的保持寄存器寻址起始地址为 1。

带扩展寄存器的 984-758 控制器中最后一个文件中一个最后的(最高的)寄存器见下面 2 个表格。

984-785 带 AS-M785-032 内存：

用户逻辑	表态 RAM	扩展内存容量	最后一个文件	最后一个寄存器(十进制)
32K	32K	0	0	0

16K	64K	72K	8	3727

984-785 带 AS-M785-032 内存：

用户逻辑	表态 RAM	扩展内存容量	最后文件	最后寄存器(十进制)
48K	32K	24K	3	4575
32K	64K	96K	10	8303

带扩展寄存器的其它控制器，最后文件中的最后一个寄存器(最高位)如下：

扩展内存容量	最后文件数	最后一个寄存器(十进制)
16K	2	6383
32K	4	2767
64K	7	5535
96K	10	8303

21(15Hex) 写通用类型寄存器

例：请求把数据写入从机设备 17 中的一组寄存器。

文件 4 在一组 3 个寄存器中，寄存器起始地址为 7 (0007)

QUERY Field Name	Example (Hex)
Slave Address	11
Function	15
Byte Count	0D
Sub–Req 1, Reference Type	06
Sub–Req 1, File Number Hi	00
Sub–Req 1, File Number Lo	04
Sub–Req 1, Starting Addr Hi	00
Sub–Req 1, Starting Addr Lo	07
Sub–Req 1, Register Count Hi	00
Sub–Req 1, Register Count Lo	03
Sub–Req 1, Register Data Hi	06
Sub–Req 1, Register Data Lo	AF
Sub–Req 1, Register Data Hi	04
Sub–Req 1, Register Data Lo	BE
Sub–Req 1, Register Data Hi	10
Sub–Req 1, Register Data Lo	0D
Error Check (LRC or CRC)	—

图 36：写通用类型寄存器-查询

响应

正常响应为返回查询信息(即查询和响应内容一致)

RESPONSE	
Field Name	Example (Hex)
Slave Address	11
Function	15
Byte Count	0D
Sub–Req 1, Reference Type	06
Sub–Req 1, File Number Hi	00
Sub–Req 1, File Number Lo	04
Sub–Req 1, Starting Addr Hi	00
Sub–Req 1, Starting Addr Lo	07
Sub–Req 1, Register Count Hi	00
Sub–Req 1, Register Count Lo	03
Sub–Req 1, Register Data Hi	06
Sub–Req 1, Register Data Lo	AF
Sub–Req 1, Register Data Hi	04
Sub–Req 1, Register Data Lo	BE
Sub–Req 1, Register Data Hi	10
Sub–Req 1, Register Data Lo	0D
Error Check (LRC or CRC)	—

图 37：写通用类型寄存器-响应

22(16Hex)掩码写入 4X 类型寄存器

说明

通过 AND 掩码,OR 掩码和寄存器当前值来修改一个指定 4X 类型寄存器的内容，该功能代码用于设置和请除寄存器中的某一位，不支持广播。

该功能代码只支持 948-785 型控制器

查询

查询可指定要写入的 4X 类型寄存器，数据可作为 AND 掩码或 OR 掩码。

算法：

结果=(当前值 AND And_Mask) OR (or_Mask AND And_Msdk)

例：	Hex	Binary
当前值=	12	0001 0010
And_Mask=	F2	1111 0010
Or_Mask=	25	0010 0101
And_Mask=	0D	0000 1101
Result=	17	0001 0111

注意 Or 的掩码值为“0”，是寄存器当前值与 And,Mask 进行逻辑运算的结果，若 And_Mask 值为“0”其结果等于 Or_Mask 值。

注意：寄存器的内容可由功能代码 03(读保持寄存器)读出，由于可编程控制器可对用户的逻辑程序扫描，同此其值可改变。

例：把上面的掩码值。掩码写入从机设备 17 中的寄存器 5。

QUERY	
Field Name	Example (Hex)
Slave Address	11
Function	16
Reference Address Hi	00
Reference Address Lo	04
And_Mask Hi	00
And_Mask Lo	F2
Or_Mask Hi	00
Or–Mask Lo	25
Error Check (LRC or CRC)	—

图 38 掩码写入 4X 类型寄存器-查询

响应

正常响应是对查询的应答，写入寄存器后，返回响应。

RESPONSE	
Field Name	Example (Hex)
Slave Address	11
Function	16
Reference Address Hi	00
Reference Address Lo	04
And_Mask Hi	00
And_Mask Lo	F2
Or_Mask Hi	00
Or–Mask Lo	25
Error Check (LRC or CRC)	—

图 39 掩码写入 4X 类型寄存器-响应

23(17Hex)读/写 4X 类型寄存器

说明

Modbus 单次传送中执行一个读操作和一个写操作。该功能代码能把新的数据写入一组 4X 类型寄存器，然后返回另一组 4X 类型寄存器中的数据，不支持广播，该功能只支持 984-785 型控制器。

查询

查询指定要读寄存器组的起始地址及寄存器数量，也指定要写入的寄存器组的起始地址及寄存器的数量，字节数区指定了应写入数据区的字节数。

例：对从机设备 17 查询，读出起始地址为 5 的 6 个寄存器内容，并把数据写入起始地址为 16 的 3 个寄存器。

QUERY

Field Name	Example (Hex)
Slave Address	11
Function	17
Read Reference Address Hi	00
Read Reference Address Lo	04
Quantity to Read Hi	00
Quantity to Read Lo	06
Write Reference Address Hi	00
Write Reference Address Lo	0F
Quantity to Write Hi	00
Quantity to Write Lo	03
Byte Count	06
Write Data 1 Hi	00
Write Data 1 Lo	FF
Write Data 2 Hi	00
Write Data 2 Lo	FF
Write Data 3 Hi	00
Write Data 3 Lo	FF
Error Check (LRC or CRC)	—

图 40：读/写 4X 类型寄存器-查询

响应

正常响应包含已被读出的寄存器组中的数据，字节数区指定了数据区应读的字节数。

例：按查询要求返回响应

RESPONSE Field Name	Example (Hex)
Slave Address	11
Function	17
Byte Count	0C
Read Data 1 Hi	00
Read Data 1 Lo	FE
Read Data 2 Hi	0A
Read Data 2 Lo	CD
Read Data 3 Hi	00
Read Data 3 Lo	01
Read Data 4 Hi	00
Read Data 4 Lo	03
Read Data 5 Hi	00
Read Data 5 Lo	0D
Read Data 6 Hi	00
Read Data 6 Lo	FF
Error Check (LRC or CRC)	—

图 41：读/写 4X 寄存器-响应

24(18Hex)读 FIFO 查询数据

说明

读一个先进先出(FIFO)的 4X 类型寄存器中查询数据，该功能代码先返回查询的寄存器数，接着返回查询数据。最多读 32 个寄存器，即寄存器数加 31 个含有查询数据的寄存器。

只能读查询数据，但不能请除数据，不支持广播。

只有 984-785 型控制器支持该功能。

查询

查询指定读 4X 类型 FIFO 查询寄存器的起始地址，该地址作为指针指向控制器的 FIN 和 FOUT 的功能块，它包含查询的寄存器数，跟在这地址后的是 FIFO 数据的寄存器。

例：读从机设备 17 中的 FIFO 查询数据，起始地址指向 41247 寄存器(04DEH)。

QUERY

Field Name	Example (Hex)
Slave Address	11
Function	18
FIFO Pointer Address Hi	04
FIFO Pointer Address Lo	DE
Error Check (LRC or CRC)	—

图 42：读 FIFO 行列-查询

响应

正常响应中，字节数包括查询字节数和数据寄存器字节，但不包括错误校验区。

查询数是查询中数据寄存器数量，不包括该查询数寄存器。

若查询数超过 31，则返回一个不正常响应带(不合法数据值) 错误代码 03。

例：按查询要求返回正常响应：

RESPONSE	
Field Name	Example (Hex)
Slave Address	11
Function	18
Byte Count Hi	00
Byte Count Lo	08
FIFO Count Hi	00
FIFO Count Lo	03
FIFO Data Reg 1 Hi	01
FIFO Data Reg 1 Lo	B8
FIFO Data Reg 2 Hi	12
FIFO Data Reg 2 Lo	84
FIFO Data Reg 3 Hi	13
FIFO Data Reg 3 Lo	22
Error Check (LRC or CRC)	—

图 43：读 FIFO 查询数据-响应

该例中，返回被指向的 FIFO 寄存器地址(41247)带查询数 3，后跟 3 个数据寄存器，确良地址分别为 41248(十进制 440 或 01B8H);41249(十进制 4740 或 1284H);41250(十进制 4898 或 1322H)。

第三章
诊断子功能代码

- Modbus 功能代码 08-诊断
- 诊断子功能代码

功能代码 08-诊断

说明

功能代码 08 提供一系列试验，校验主机和从机间的通讯系统或检查从机中出现错误的各种条件，不支持广播。

该功能使用一个子功能代码（2 个字节），定义试验的类型。正常响应时，从机返回功能代码和子功能代码。

大多数诊断测试，使用 1 个 2 字节的数据区，向从机发送诊断数据和控制信息。有些诊断会产生需由从机返回的数据，放在正常响应的数据区。

诊断对从机设备的影响

一般来说，向从机设备发出的一个诊断功能，不会影响从机运行的用户程序。因为诊断不作用于用户的逻辑，如离散量控制和寄存器等。某些功能可对从机中出错的计数器复位。

可以使一台从机设备强置为“只听模式”，以便检测通讯系统，但不对信息作响应，此时，对用户应用程序的影响程度取决于和从机设备的数据交换量，通常采用强制方法的目的，是把有故障的从机设备排除在通讯系统之外。

查询

例：请求从机设备 17 返回查询数据，使用一个子功能代码“0”(0000H)，需返回的数据(A537H)在一个 2 字节的数据区中。

QUERY	
Field Name	Example (Hex)
Slave Address	11
Function	08
Subfunction Hi	00
Subfunction Lo	00
Data Hi	A5
Data Lo	37
Error Check (LRC or CRC)	—

图 44：诊断－查询

响应

正常响应返回的数据与查询的数据相同，包括功能代码和子功能代码。

RESPONSE	
Field Name	Example (Hex)
Slave Address	11
Function	08
Subfunction Hi	00
Subfunction Lo	00
Data Hi	A5
Data Lo	37
Error Check (LRC or CRC)	—

图 45：诊断－响应

对其它类型查询响应的数据区可含有错误数或子功能代码请求的其它信息。以上 2 个例子说明了查询和响应信息中的功能代码和子功能代码以及数据区的单元和位置。

控制器支持的诊断功能代码

Modicon 控制器支持的子功能代码列表如下：(代码为十进制)。
其中“Y”为支持，“N”为不支持。

代码	名称	384	484	584	884	M84	984
00	返回查询数据	Y	Y	Y	Y	Y	Y
01	再起动通讯选择	Y	Y	Y	Y	Y	Y
02	返回诊断寄存器	Y	Y	Y	Y	Y	Y
03	改变 ASCII 输入分隔符	Y	Y	Y	N	N	Y
04	强制只听模式	Y	Y	Y	Y	Y	Y
05-09	备用						
10	请除 Ctrs 和诊断标志	Y	Y	(1)	Y	Y	(1)
11	返回总线信息数	Y	Y	Y	N	N	Y
12	返回总线通讯错误数	Y	Y	Y	N	N	Y
13	返回总线不正常错误数	Y	Y	Y	N	N	Y
14	返回从机信息数	Y	Y	Y	N	N	N
15	返回从机不响应数	Y	Y	Y	N	N	N
16	返回从机 NAK 数	Y	Y	Y	N	N	Y
17	返回从机忙数	Y	Y	Y	N	N	Y
18	返回总线字符超限数	Y	Y	Y	N	N	Y
19	返回超限错误数	N	N	N	Y	N	N

20	清除超限计数器和标志	N	N	N	Y	N	N
21	获取/清除 Modbus+统计值	N	N	N	N	N	Y
22-up	备用						

注意:

（1）只清除记数器

00 返回查询数据

响应时返回查询区的数据，响应信息与查询信息相同。

子功能	数据区(查询)	数据(响应)
00 00	任意	返回查询数据

01 再起动通讯选择

对从机外设口初始化和再起动，对全部的通讯事件计数器清零。若当前的端口为“只听模式”，则无响应返回。该功能代码是唯一能撤除“只听模式”的功能代码，若端口不在“只听模式”工作，则在执行重起前返回正常响应。

值从机接收查询时，该功能对再起动和电源进行测试，只有顺利通过上述试验后，端口才能处于联机状态。

查询数据区 FF 00H 值清除端口通讯事件的记录，而 00 00H 值在重起动前清除记录。

子功能代码	数据区(查询)	数据(响应)
00 01	0000	00 响应查询数据
00 01	FF 00	响应查询数据

02 返回诊断寄存器

响应时，返回从机的 16 位诊断寄存器内容。

子功能代码	查询数据区	响应数据区
00 02	00 00	诊断寄存器内容

寄存器数据格式

Modicon 控制器诊断寄存器每位的赋值列表如下。

bit 15 为最高位，相应位置成“1”时，为（TRUE）

184/384 诊断寄存器

Bit	说明
0	连续出错
1	Run Light Failed
2	T-Bus 测试失败
3	异步总线测试失败
4	强制只听模式
5	不用
6	不用
7	ROM 芯片“0”测试失败
8	连续执行 ROM 测试
9	ROM 芯片 1 测试失败
10	ROM 芯片 2 测试失败
11	ROM 芯片 3 测试失败
12	RAM 芯片 5000-53FF 测试失败
13	RAM 芯片 6000-67FF 测试失败，偶地址
14	RAM 芯片 6000-67FF 测试失败，奇地址
15	定时器芯片测试失败

诊断寄存器

Bit	说明
0	连续
1	CPU 试验或运行指示故障
2	并行口测试失败
3	异步总线测试失败
4	定时器 0 测试失败
5	定时器 1 测试失败
6	定时器 2 测试失败
7	ROM 芯片 0000-07FF 测试失败
8	连续执行 ROM 测试
9	ROM 芯片 0800-0FFF 测试失败
10	ROM 芯片 1000-17FF 测试失败
11	ROM 芯片 1800-1FFF 测试失败
12	ROM 芯片 4000-40FF 测试失败
13	ROM 芯片 4100-41FF 测试失败

14	ROM 芯片 4200-42FF 测试失败
15	ROM 芯片 4300-43FF 测试失败

584/984 诊断寄存器

Bit	说明
0	不合法配置
1	高速 RAM 后备检测错误
2	逻辑检查错误
3	无效节点类型
4	无效 Traffic Cop 类型
5	CUP/Solve 诊断失败
6	实时时钟故障
7	看门狗定时器故障－扫描时间超过 250ms
8	未检测到逻辑节点的终点，或数段结束量与配置段量不匹配
9	静态 RAM 测试失败
10	
11	
12	
13	
14	
15	外设口停止，无出错。

884 诊断寄存器

Bit	说明
0	Modbus IOP 超过错误标志
1	Modbus 选择超过错误标志
2	Modbus IOP 故障
3	Modbus 选择故障
4	Ourbus IOP 故障
5	远程 IO 故障
6	主 CUP 故障
7	RAM 表检查故障
8	扫描任务超过时限用户逻辑太多
9	未用
10	未用
11	未用
12	未用
13	未用
14	未用
15	未用

03 改变 ASCII 输入分隔符

查询数据中的“CHAR”字符为信息结束的分隔符(替代缺省的 LF 字符)，ASCII 信息终了不用“LF”的作结束符时使用该功能码。

子功能代码	查询数据区	响应数据区
00 03	CHAR 00	返回查询数据

04 强置“只听模式”

强制从机为“只听模式”，该功能代码把此从机设备与网络上的其它设备隔离，使其不干扰其它设备的正常通讯，无响应返回。

从机进入只听模式后，关掉全部的通讯控制，终止看门狗定时器，封锁控制功能。只听模式时，从机检测接收的信息或广播信息，但不执行动作，不返回响应。

唯一能处理“只听模式”的功能代码为 08，子功能代码为 1(再起动通讯选择功能代码)。

子功能代码	查询数据区	响应数据区
00 04	00 00	不响应

10(0AH)计数器和诊断寄存器清零

对 584 或 984 型控制器，该功能代码只对计数器清零(计数器通电时也清零)，对其它类型的控制器时，该功能代码对全部的计数器和诊断寄存器清零。

子功能代码	查询数据区	响应数据区
00 0A	00 00	返回查询数据

484 诊断寄存器

Bit	说明
0	连续出错
1	CPU 试验和 Run Light 失败
2	Parallel Port Failed
3	异步总线测试失败
4	定时器 0 测试失败
5	定时器 1 测试失败
6	定时器 2 测试失败
7	ROM 芯片 0000-07FF 测试失败
8	连续执行 ROM 测试
9	ROM 芯片 0800-0FFF 测试失败
10	ROM 芯片 1800-17FF 测试失败
11	ROM 芯片 0000-1FFF 测试失败
12	RAM 芯片 4000-40FF 测试失败
13	RAM 芯片 4100-41FF 测试失败
14	RAM 芯片 4200-42FF 测试失败
15	RAM 芯片 4300-43FF 测试失败

584/984 诊断寄存器

Bit	说明
0	不合法配置
1	高速 RAM 后备检测错误

2	逻辑检查错误
3	无效节点类型
4	无效 Traffic Cop 类型
5	CPU/Solve 诊断失败
6	实时时钟故障
7	看门狗定时器故障-扫描时间超过 250ms
8	No End of Logic Node detected,or quantity of end of segment words(DOIO) does not match quantity of segments configured
9	表态 RAM 测试失败
10	Start of Network(SON)did not begin network
11	Bad Order of Solve Tacle
12	Illegal Peripheral lntervention
13	Dim Awareness Flag
14	Not Used
15	Peripheral Port Stop Executed,not an error.

11(0B H)返回总线信息数

响应数据区的主机返回再起动，计数器清零或通电后，从机通讯系统已检测的数据量。

子功能代码	查询数据区	响应数据区
00 0B	00 00	总的信息数

12(0C H)返回总线通讯错误数

响应数据区向主机返回再起动，计数器清零或通电后，从机所测测到的 CRC 校验错误数。

子功能代码	查询数据区	响应数据区
00 0C	00 00	CRC 错误数

13(0D Hex)返回总线不正常错误数

响应数据区向主机返回再起动，计数器清零或通电后，Modbus 不正常响应的数量。不正常响应详见附录 A。

子功能代码	查询数据区	响应数据区
00 0D	00 00	不正常错误数

14(0E Hex)返回从机信息数

响应数据区向主机返回再起动，计数器清零(或通电)后，从机已处理的被访问的(或广播)信息量。

子功能代码	查询数据区	响应数据区
00 0E	00 00	从机信息数

15(0F Hex)返回从机不响应数

数据区向主机返回再起动，计数器清零(或通电)后，从机被访问而无响应返回的信息量，(既可正常响应也可不正常响应)。

子功能代码	查询数据区	响应数据区
00 0F	00 00	从机不响应数

16(10Hex)返回从机 NAK 数

数据区向主机返回再通电，计数器清零(或通电)后，从机被访问，返回 NAK 不正常响应的信息量。

子功能	数据区(查询)	数据(响应)
00 10	00 00	从机 NAK 数

17(11Hex)从机返回忙的次数

响应数据区向主机返回再起动，计数器清零(或通电)后因从机设备忙而引起不正常响应的次数，不正常响应详见附录 A。

子功能代码	查询数据区	响应数据区
00 11	00 00	从机设备忙次数

18(12Hex)返回总线字符超限次数

响应数据区向主机返回再起动和计数器清零(或通电)后，从机因字符超限而无法处理信息的次数。字符超限是由于到达端口的字符速度高于从机能保存的速度，或由于硬件故障丢失字符而引起。

子功能代码	查询数据区	响应数据区
00 11	00 00	从机设备忙次数

19(13Hex)返回 IOP 超限次数(884)

响应数据区向主机返回，再起动，计数器清零(或通电)后，被访向从机因一台 884 IOP 超限条件而无法次理信息的次数，IOP 超限是由于到达端口字符的速度高于从机能保存的速度或由于硬件故障丢失字符而引起。该功能代码指定用于 884 型控制。

子功能代码	查询数据区	响应数据区
00 13	00 00	从机 IOP 超限次数

20(14Hex)超限计数器和标志清零(884)

对 884 超限错误计数器清零，对错误标志复位，标志的当前状态放在 884 诊断寄存器的 0 位。

子功能代码	查询数据区	响应数据区
00 14	00 00	返回查询数据

21(15Hex)获取/清除 Modbus+的统计值

响应数据区向主机返回信息量为 108 字节，54 个 16 位字数据。该功能与数据区的 2 个字节长度不同。数据中含有对从机设备中的确良 Modbus+对等。处理器的统计数，查询时除功能代码(08)和子功能代码(0015H)外，还有一个 2 字节的操作区，用于执行“获取统计”或“清除统计”的操作。“获取操作”不能清除统计值，清除操作在清除前不能回到“统计值”，统计值可在从机设备通电时清除。操作区后跟查询子功能区。

—00 03 代码指定“获取统计”操作
—00 04 代码指定“清除统计”操作

功能代码	子功能代码	操作	
08	00 15	00 03	(获取统计)
08	00 15	00 04	(清除统计)

获取统计响应：

功能代码	子功能代码	操作	字节数	数据
08	00 15	00 03	006C	字 00-53

清除统计响应：清除统计的正常响应是对查询的应答查询

功能代码	子功能代码	操作
08	00 15	00 04

Modbus+网络统计

字	位	含义
0	0	节点类型 ID：
	0	未知节点类型
	1	PLC 节点
	2	Modbus 桥路节点
	3	主机节点
	4	桥路加节点
	5	对等 1/0 节点
01	0...11	十六进制软件文本号 (从字中读 12-15 位)
	12...14	备用
	15	定义字 15 个错误计数器

MSB 定义字 15 错误计数器，高字节的低 4 位。
加上低字节 8 位，为(十六进制)软件版本

高字节	低字节
[ ]	[软件版本（十六进制）]

\\
最高有效位定义字 15 错误计数器。

02 该站的网络地址

03 状态变量：

0	通电状态
1	监视脱机状态
2	双工脱机状态
3	空间位
4	令牌状态
5	工作响应状态
6	传送牌
7	请求响应
8	检查通过状态
9	要求令牌状态
10	要求响应状态

04 对等状态(LED 代码)提供与网络有关单元的状态：

0	监视连接操作
32	正常连接操作
64	不取令牌
96	单工站
128	双工站

Modbus Plus 网络统计

字	位	含义
05		令牌通过计数器，通过令牌站一次，计数器加 2
06		令牌旋转时间(ms)
15		若 15 字的 1 位未设定，含义如下：
	LO	接收器冲突－放弃错误计数器
	HI	接收器排队错误计数器
		设定字 15 的 1 位后，含义如下：
	LO	电览 A 出错
	HI	电览 B 出错
16	LO	接收器 CRC 错误寄存器
	HI	捆包长度错误计数器
17	LO	连接地址错误计数器
	HI	发送绶冲器 DMA-underrun 错误计数器
18	LO	内部捆包长度错误计数器
	HI	MAC 功能代码错误计数器
19	LO	通讯再试计数器
	HI	故障(错误)计数器

08 诊断

Modbus Plus 网络统计

字	位	含义
05		令牌通过计数器，通过令牌站一次，计数器加 1
06		令牌旋转时间(ms)
07	LO	令牌循环期间数据主站失败
	HI	令牌循环期间程序主站失败
08	LO	数据主站令牌宿主位图
	HI	程序主站令牌宿主位图
09	LO	数据从站令牌宿主位图
	HI	程序从站令牌宿主位图
10	HI	数据从站/得到从机命令传送位图请求
11	LO	程序主机/得到主机请求发送位图
	HI	程序从机/得到从机命令请求发送位图

12	LO	程序主站连接状态状位图
	HI	程序从站自动退出请求位图
13	LO	提前发送延期错误计数器
	HI	接收缓冲器超时错误记数器
14	LO	重复命令接收记数器
	HI	帧错误记数器
15	若 15 字的 1 位未设定，含义如下：	
	LO	接收器冲突一放弃错误计数器
	HI	接收器排队错误计数器
	设定字 15 的 1 位后含义如下：	
	LO	电缆 A 出错
	HI	电缆 B 出错
16	LO	接收器 CRC 错误寄存器
	HI	捆包长度错误计数器
17	LO	连接地址错误计数器
	HI	发送冲器 DMA underrun 错误计数器

字	位	含义
20	LO	捆包成功计数器
	HI	无响应错误计数器
21	LO	不正常响应计数器
	HI	不正常通道计数器
22	LO	不正常响应计数器
	HI	忘记发送错误计数器
23	LO	有效站位图表，节点 1...8
	HI	有效站位图表，节点 9...16

24	LO	有效站位图表，节点 17...24
	HI	有效站位图表，节点 25...32
25	LO	有效站位图表，节点 33...40
	HI	有效站位图表，节点 41...48
26	LO	有效站位图表，节点 49...56
	HI	有效站位图表，节点 57...64
27	LO	令牌站位图表，节点 1...8
	HI	令牌站位图表，节点 9...16
28	LO	令牌站位图表，节点 17...24
	HI	令牌站位图表，节点 25...32
29	LO	令牌站位图表，节点 33...40
	HI	令牌站位图表，节点 41...48
30	LO	令牌站位图表，节点 49...56
	HI	令牌站位图表，节点 57...64
31	LO	全局数据位图表 1...8
	HI	全局数据位图表 9...16
32	LO	全局数据位图表 17...24
	HI	全局数据位图表 25...32
33	LO	全局数据位图表 33...40
	HI	全局数据位图表 41...48
34	LO	全局数据表 49...56
	HI	全局数据位图表 57...64

Modbus+网络统计

字	位	含义
35	LO	位图中由接收缓冲器，缓冲器 1...8
	HI	位图中由接收缓冲器，缓冲器 9...16
36	LO	位图中由接收缓冲器，缓冲器 17...24
	HI	位图中由接收缓冲器，缓冲器 25...32
37	LO	位图中由接收缓冲器，缓冲器 33...40
	HI	站管理命令开始计数器
38	LO	主机数据输出通道 1 命令开始计数器、
	HI	主机数据输出通道 2 命令开始计数器
39	LO	主机数据输出通道 3 命令开始计数器
	HI	主机数据输出通道 4 命令开始计数器
40	LO	主机数据输出通道 5 命令开始计数器
	HI	主机数据输出通道 6 命令开始计数器
41	LO	主机数据输出通道 7 命令开始计数器
	HI	主机数据输出通道 8 命令开始计数器
42	LO	从机数据输入通道 41 命令处理计数器
	HI	从机数据输入通道 42 命令处理计数器
43	LO	从机数据输入通道 43 命令处理计数器
	HI	从机数据输入通道 44 命令处理计数器
44	LO	从机数据输入通道 45 命令处理计数器
	HI	从机数据输入通道 46 命令处理计数器
45	LO	从机数据输入通道 47 命令处理计数器
	HI	从机数据输入通道 48 命令处理计数器
46	LO	主机数据输出通道 81 命令开始计数器
	HI	主机数据输出通道 82 命令开始计数器
47	LO	主机数据输出通道 83 命令开始计数器
	HI	主机数据输出通道 84 命令开始计数器
48	LO	主机程序命令开始计数器
	HI	主机程序输出通道 86 命令开始计数器
49	LO	主机程序输出通道 87 命令开始计数器
	HI	主机程序输出通道 88 命令开始计数器
50	LO	从机程序输入通道 C1 命令处理计数器
	HI	从机程序输入通道 C2 命令处理计数器
51	LO	从机程序输入通道 C3 命令处理计数器
	HI	从机程序输入通道 C4 命令处理计数器

52	LO	从机程序输入通道 C5 命令处理计数器
	HI	从机程序输入通道 C6 命令处理计数器
53	LO	从机程序输入通道 C7 命令处理计数器
	HI	从机程序输入通道 C8 命令处理计数器

附录 A
不正常响应

- 不正常响应
- 不正常代码

不正常响应

除广播外，主机向从机设备发送查询并希望有一个正常响应，主机查询中有可能产生 4 种事件：

- 从机接收查询，通讯错误正常处理信息，则返回一个正常响应事件。
- 由于通讯出错，从机不能接收查询数据，因而不返回响应。此时，主机依靠处理程序给出查询超时事件。
- 若从机接收查询，发现有（LRC 或 CRC）通讯错误，并返回响应，此时，依靠主机处理程序给出查询超时事件。
- 从机接收查询，无通讯错误，但无法处理(如读不存在的线圈和寄存器)时，向主机报告错误的性质。

不正常响应信息有 2 个与正常响应不相同的区域：

功能代码区：正常响应时，从机的响应功能代码区，带原查询的功能代码。所有功能代码的 MSB 为 0(其值低于 80H)。不正常响应时，从机把功能代码的 MSB 置为 1，使功能代码值大于 80H，高于正常响应的值。这样，主机应用程序能识别不正常响应事件，能检查不正常代码的数据区。

数据区：正常响应中，数据区含有(按查询要求给出的）数据或统计值，在不正常响应中，数据区为一个不正常代码，它说明从机产生不正常响应的条件和原因。

例：主机发出查询，从机不正常响应。（为十六进制数据）。

QUERY

Byte	Contents	Example
1	Slave Address	0A
2	Function	01
3	Starting Address Hi	04
4	Starting Address Lo	A1
5	No. of Coils Hi	00
6	No. of Coils Lo	01
7	LRC	4F

EXCEPTION RESPONSE

Byte	Contents	Example
1	Slave Address	0A
2	Function	81
3	Exception Code	02
4	LRC	73

图 46：主机发出查询，从机不正常响应

上例中，从机设备地址 10(0AH)，读线圈状态的功能代码(01)，主机请求线圈状态的地址为 1245(04A1H)。注意：只读一个指定线圈，地址为(0001).

若从机中不存在此线圈地址时，即以不正常代码(02)，向主机返回一个不正常响应。说明为不合法地址。

不正常代码

代码	名称	含义
01	不合法功能代码	从机接收的是一种不能执行功能代码。发出查询命令后，该代码指示无程序功能。
02	不合法数据地址	接收的数据地址，是从机不允许的地址。
03	不合法数据	查询数据区的值是从机不允许的值。
04	从机设备故障	从机执行主机请求的动作时出现不可恢复的错误。
05	确认	从机已接收请求处理数据，但需要较长　　的处理时间，为避免主机出现超时错误而发送该确认响应。主机以此再发送一个“查询程序完成”未决定从机是否已完成处理。
06	从机设备忙碌	从机正忙于处理一个长时程序命令，请求主机在从机空闲时发送信息。
07	否定	从机不能执行查询要求的程序功能时，该代码使用十进制 13 或 14 代码，向主机返回一个“不成功的编程请求”信息。主机应请求诊断从机的错误信息。
08	内存奇偶校验错误	从机读扩展内存中的数据时，发现有奇偶校验错误，主机按从机的要求重新发送数据请求。

附录 B
应用须知

- 本附录向您提供有关应用的资料和建议。
- ModiconPLC 的最大查询/响应参数。
- 估算串行传送时序
- 584 和 984A/B/X PLC 的应用须知

最大查询/响应参数

584

功能	说明	查询	响应
1	读线圈状态	2000 线圈	2000 线圈
2	读输入状态	2000 输入	2000 输入
3	读线圈状态	125 寄存器	125 寄存器
4	读输入状态	125 寄存器	125 寄存器
5	强置单线圈	1 线圈	1 线圈
6	预置单寄存器	1 寄存器	1 寄存器
7	读不正常状态	N/A	8 线圈
8	诊断	N/A	N/A
9	程序 484	不支持	不支持
10	查询 484	不支持	不支持
11	获取通讯事件控制	N/A	N/A
12	获取通讯事件记录	N/A	70 数据字节
13	编程控制器	33 数据字节	33 数据字节
14	查询控制器	N/A	33 数据字节
15	强置多线圈	800 线圈	800 线圈
16	预置多寄存器	100 寄存器	100 寄存器
17	报告从机	N/A	N/A
18	编程 884/M84	不支持	不支持
19	复位通讯连接 Limk	不支持	不支持
20	读通用类型寄存器	(1)	(1)
21	写通用类型寄存器	(1)	(1)

最大查询/响应参数

984

功能		说明	查询	响应
1	读线圈状态	2000 线圈	2000 线圈	
2	读输入状态	2000 输入	2000 输入	
3	读线圈状态	125 寄存器	125 寄存器	
4	读输入状态	125 寄存器	125 寄存器	
5	强置单线圈	1 线圈	1 线圈	
6	预置单寄存器	1 寄存器	1 寄存器	
7	读不正常状态	N/A	8 线圈	
8	诊断	N/A	N/A	
9	程序 484	不支持	不支持	
10	查询 484	不支持	不支持	
11	获取通讯事件控制	N/A	N/A	
12	获取通讯事件记录	N/A	70 数据字节	
13	编程控制器	33 数据字节	33 数据字节	
14	查询控制器	N/A	33 数据字节	
15	强置多线圈	800 线圈	800 线圈	
16	预置多寄存器	100 寄存器	100 寄存器	
17	报告从机	N/A	N/A	
18	编程 884/M84	不支持	不支持	
19	复位通讯连接 Limk	不支持	不支持	
20	读通用类型寄存器	(1)	(1)	
21	写通用类型寄存器	(1)	(1)	

估算串行处理时序

处理顺序

Modbus 串行传输时按如下顺序：括号中的字符。

1. Modbus 主机组织信息。
2. 检查主机设备 Modbus RTS 和 CTS 的状态。(A)
3. 向从机发送查询信息。(B)
4. 从机处理查询数据。(C) (D)
5. 从机计算一个错误校验区。(E)
6. 检查从机设备 RTS 和 CTS, modem 的状态。(A)
7. 向主机返回响应信息。(B)
8. 主机按从机响应的数据处理。

Timing Notes

(A) 若 RTS 和 CTS 跳接在一起，可忽略该时间。
对 J478 modem, 该时间约 5ms。

(B) 使用下述公式，估算传时间：

(C) 在 PLC 扫描结束时，处理 Modbus 信息，最坏情况延时是一个扫描时间，若控制器刚开始一个新扫描时间。

控制器扫描结束时，Modbus 端口的分配时间取决控制器的型号。

(D) Contiued:

584 和 984 型控制器，每个 Modbus 端口的工作时间约为 1.5ms,从端口 1 开始，顺序进行。

有些低档控制器(184/384)该时间按处理的数量的大小而变化。从 0.5ms 至 6.0ms(100 寄存器)或至 7.0ms(800 个线圈)。

(E) Modbus 功能代码 1-4，1.5 和 16，可在从机工作的 Modbus 端口的分配时间内允许主机请求从机处理更多的数据：若从机一时无法处理的话，可放入缓冲区。
Modbus 端口在一次服务期中能处理的数据量如下：

	点数	寄存器
Micro84	16	4
184/384	800	100
484	32	16
584	64	32
984A/B/X	64	32
984-X8X	1000	125

注意：‘984-X8X’指的是 984 单独使用型(984-835,-685,etc)。

(F) LRC 计算时间＜1ms,CRC 计算时间约 0.3ms(响应而返回的 8 位数据)

584 和 984A/B/X PLC 应用须知

本应用须知只适用 Modicon 584 和 984/A/B/X 控制器。

- **波特率：**用 Modbus 端口 1 和端口 2 时，最大允许的波特率为 19200。
- **端口封锁：**使用 ASCII,要送“零数据长度”的信息，或无设备地址的信息，如下为一个不合法信息。
  : CR LF (colon,CR,LF)
  出现这类信息时，会随机封死端口。
- ASCII 信息终止：ASCII 信息一般用，CRLF 终止。584 和 984A/B/X 型控制器的一个信息，可在 LRC 区后终止，而不需发送 CRLF 字符，如 LRC 区后出现>IS 时间间隔则控制器认为信息是正常终止。

附录 C
LRC/CRC 生成

- LRC 校验
- CRC 校验

LRC 纵向冗余校验

纵向冗余校验区为 1 个字节，8 位二进制数据，由发送设备计算 LRC 值，并把计算值附到信息中。接收设备在接收信息时，重新计算 LRC 值，并把计算值与 LRC 区中接收的实际值进行比较，若两者不相同，则产生一个错误。

信息中的相邻 2 个 8 位字节相加，丢弃进位，然后进行二进制补码，运算计算出 LRC 值。LRC 是一个 8 位数据区，因此每加一个新字符，会产生大于十进制 255 的数值而溢出，因为没有第 9 位，自动放弃进位。
产生 LRC 的过程：

1. 相加信息中的全部字节，包括起始“:”号和结束符 CRLF.并把结果送入 8 位数据区，放弃进位。
2. 由 FFH 减去最终的数据值，产生的补码。
3. 加“1”产生二进制补码。

把 LRC 放入信息中

发送 8 位 LRC(2 个 ASCII 字符)时，先送高位字符，后送低位字符，如:LRC 值为 61H(0110 0001):

Colon	Addr	Func	Data Count	Data	Data	Data	Data	LRC Hi	LRC Lo	CR
								6	1	

图 47LRC 字符顺序

例：

用一个C语言功能码产生LRC值。
该功能码用2个自变量：

`unsigned char *auchMsg ;` 为生成LRC值，把指针指向含有二进制数据的缓冲器

`unsigned short usDataLen ;` 缓冲器中的字节数。

该功能返回LRC作为一种类型“`unsigned char`”。

LRC产生的功能

```
static unsigned char LRC(auchMsg, usDataLen)
unsigned char *auchMsg ; /*按信息的字节计算LRC*/
unsigned short usDataLen ; /*按信息的字节计算LRC*/
{
   unsigned char uchLRC = 0 ; /*初始化LRC字符 */
   while (usDataLen--) /*通过数据缓冲器*/
         uchLRC += *auchMsg++ ; /*加缓冲器字节无进位*/
   return ((unsigned char)(-((char)uchLRC))) ; /*返回二进制补码*/
}
```

CRC 循环冗余校验

循环冗余校验CRC区为2字节，含一个16位二进制数据。由发送设备计算CRC值，并把计算值附在信息中，接收设备在接收信息时，重新计算CRC值，并把计算值与接收的在CRC区中实际值进行比较，若两者不相同，则产生一个错误。

CRC开始时先把寄存器的16位全部置成“1”,然后把相邻2个8位字节的数据放入当前寄存器中，只有每个字符的8位数据用作产生CRC，起始位，停止位和奇偶校验位不加到CRC中。

产生CRC期间，每8位数据与寄存器中值进行异或运算，其结果向右移一位(向LSB方向)，并用“0”填入MSB，检测LSB，若LSB为“1”则与预置的固定值异或，若LSB为“0”则不作异或运算。

重复上述处过程，直至移位8次，完成第8次移位后，下一个8位数据，与该寄存器的当前值异或，在所有信息处理完后，寄存器中的最终值为CRC值。
产生CRC的过程：

1. 把16位CRC寄存器置成FFFFH.
2. 第一个8位数据与CRC寄存器低8位进行异或运算，把结果放入CRC寄存器。
3. CRC寄存器向右移一位，MSB填零，检查LSB.
4. (若LSB为0):重复3，再右移一位。
   (若LSB为1):CRC寄存器与Aool H 进行异或运算
5. 重复3和4直至完成8次移位，完成8位字节的处理。
6. 重复2至5步，处理下一个8位数据，直至全部字节处理完毕。
7. CRC寄存器的最终值为CRC值。
8. 把CRC值放入信息时，高8位和低8位应分开放置。

把CRC值放入信息中

发送信息中的16 位CRC值时，先送低8位，后送高8位。
若CRC值为1241(0001 0010 0100 0001):

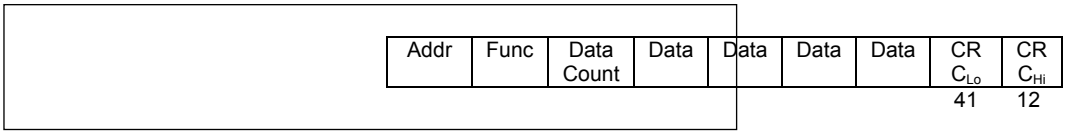

图48：CRC字节顺序

例：

各种可能的CRC值，按两列装入，一列在16 位CRC的高8位区，为(0-256的)CRC值，另一类为低8位区，为CRC的低位值。

用这种方法得到的CRC其执行速度快于计算缓冲器中每个新字符得到一个新CRC值的方法。

※ **注意：**该功能内部交换CRC中的高/低字节，返回的CRC值中，其字节已交换。

因此，由功能码返回的CRC值，能直接放在信息中传送。

例：

功能取2个自变量：

`unsigned char *puchMsg ;` 为生成CRC值，把指针指向含有二进制的数据的缓冲器

`unsigned short usDataLen ;` 缓冲器中的字节数。

该功能返回CRC作为一种类型“`unsigned short`”。

CRC产生的功能

```
unsigned short CRC16(puchMsg, usDataLen)

unsigned char *puchMsg ; /*按信息的字节数计算CRC */

unsigned short usDataLen ; /* quantity of bytes in message */

{

   unsigned char uchCRCHi = 0xFF ; /* 初始化高字节*/

   unsigned char uchCRCLo = 0xFF ; /* 初始化低字节*/

   unsigned uIndex ; /*把CRC表*/

   while (usDataLen--) /*通过数据缓冲器*/

        {

        uIndex = uchCRCHi ^ *puchMsgg++ ; /*计算CRC */

        uchCRCHi = uchCRCLo ^ auchCRCHi[uIndex] ;

        uchCRCLo = auchCRCLo[uIndex] ;

        }

    return (uchCRCHi << 8 | uchCRCLo) ;

}
```

高位字节表

```
/* Table of CRC values for high-order byte */

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, 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, 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, 0xC0, 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, 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, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40
} ;
```

低位字节表

```
/* Table of CRC values for low-order byte */

static char auchCRCLo[] = {
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07, 0xC7, 0x05, 0xC5, 0xC4,
0x04, 0xCC, 0x0C, 0x0D, 0xCD, 0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09,
0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A, 0x1E, 0xDE, 0xDF, 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, 0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A,
0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29, 0xEB, 0x2B, 0x2A, 0xEA, 0xEE,
0x2E, 0x2F, 0xEF, 0x2D, 0xED, 0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
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, 0x7F, 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
} ;
```