

SE300 – RS485 MODBUS RTU 通信協議

摘要：工業控制已從單機控制走向集中監控、集散控制，如今已進入網路時代，工業控制器連網也為網路管理提供了方便。Modbus 就是工業控制器的網路協定中的一種。

關鍵字：Modbus 協議；串列通信；LRC 校驗；CRC 校驗；RS-232C

一、Modbus 協議簡介

Modbus 協定是應用於電子控制器上的一種通用語言。通過此協定，控制器相互之間、控制器經由網路（例如以太網）和其他設備之間可以通信。它已經成為一通用工業標準。有了它，不同廠商生產的控制設備可以連成工業網路，進行集中監控。

此協定定義了一個控制器能認識使用的消息結構，而不管它們是經過何種網路進行通信的。它**描述了一控制器請求訪問其他設備的過程，如果回應來自其他設備的請求，以及怎樣偵測錯誤並記錄。它制定了消息欄位格局和內容的公共格式。**

當在一 Modbus 網路上通信時，此協議決定了每個控制器須要知道它們的設備位址，識別按位址發來的消息，決定要產生何種行動。如果需要回應，控制器將生成反饋資訊並用 Modbus 協定發出。在其他網路上，包含了 Modbus 協定的消息轉換為在此網路上使用的訊框或包結構。這種轉換也擴展了根據具體的網路解決節位址、路由路徑及錯誤檢測的方法。

1、在 Modbus 網路上轉輸

標準的 Modbus 口是使用一 RS-232C 相容串列介面，它定義了連介面的針腳、電纜、信號位元、傳輸串列傳輸速率、奇偶校驗。控制器能直接或經由 Modem 組網。

控制器通信使用主—從技術，即僅一設備（主設備）能初始化傳輸（查詢）。其他設備（從設備）根據主設備查詢提供的資料作出相應反應。典型的主設備：主機和可編程儀錶。典型的從設備：可編程控制器。

主設備可單獨和從設備通信，也能以廣播方式和所有從設備通信。如果單獨通信，從設備返回一消息作為回應，如果是廣播方式查詢的，則不作任何回應。**Modbus 協定建立了主設備查詢的格式：設備（或廣播）位址、功能代碼、所有要發送的資料、一錯誤檢測欄位。**

從設備回應消息也由 Modbus 協定構成，包括確認要行動的欄位、任何要返回的資料、和一錯誤檢測欄位。如果在消息接收過程中發生一錯誤，或從設備不能執行其命令，從設備將建立一錯誤消息並把它作為回應發送出去。

2、在其他類型網路上轉輸

在其他網路上，控制器使用對等技術通信，故任何控制都能初始和其他控制器的通信。這樣在單獨的通信過程中，控制器既可作為主設備也可作為從設備。提供的多個內部通道可允許同時發生的傳輸進程。

在消息位元，Modbus 協定仍提供了主—從原則，儘管網路通信方法是“對等”。如果一控制器發送一消息，它只是作為主設備，並期望從從設備得到回應。同樣，當控制器接收到一消息，它將建立一從設備回應格式並返回給發送的控制器。

3、查詢一回應週期

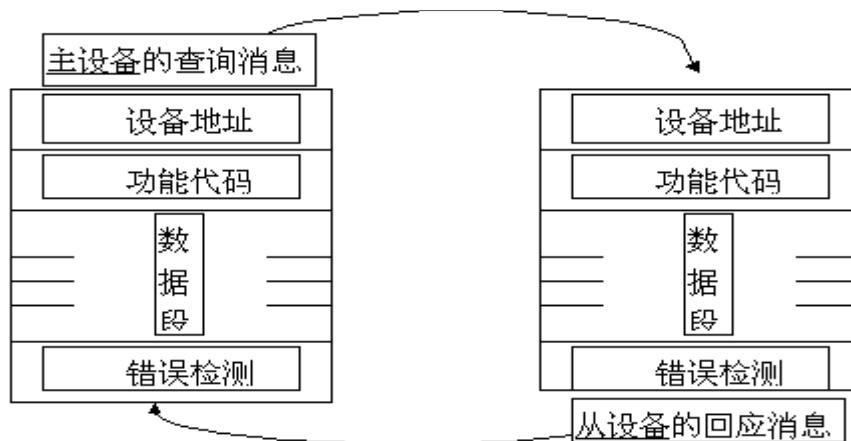


图 1 主—从 查询—回应周期表

(1) 查詢

查詢消息中的功能代碼告之被選中的從設備要執行何種功能。資料段包含了從設備要執行功能的任何附加資訊。例如功能代碼 03 是要求從設備讀保持暫存器並返回它們的內容。資料段必須包含要告之從設備的資訊：從何暫存器開始讀及要讀的暫存器數量。錯誤檢測欄位為從設備提供了一種驗證消息內容是否正確的方法。

(2) 回應

如果從設備產生一正常的回應，在回應消息中的功能代碼是在查詢消息中的功能代碼的回應。資料段包括了從設備收集的資料：象暫存器值或狀態。如果有錯誤發生，功能代碼將被修改以用於指出回應消息是錯誤的，同時資料段包含了描述此錯誤資訊的代碼。錯誤檢測欄位允許主設備確認消息內容是否可用。

二、兩種傳輸方式

控制器能設置為兩種傳輸模式（ASCII 或 RTU）中的任何一種在標準的 Modbus 網路通信。用戶選擇想要的模式，包括串口通信參數（串列傳輸速率、校驗方式等），在配置每個控制器的時候，在一個 Modbus 網路上的所有設備都必須選擇相同的傳輸模式和串口參數。

ASCII 模式

:	位 址	功 能 代 碼	資 料 數 量	數據 1	...	數據 n	LRC 高位 元組	LRC 低位 元組	回車	換行
---	--------	------------	------------	------	-----	---------	--------------	--------------	----	----

RTU 模式

位址	功 能 代 碼	資 料 數 量	數據 1	...	數據 n	CRC 高位 元組	CRC 低位 元組
----	------------	------------	------	-----	------	--------------	--------------

所選的 ASCII 或 RTU 方式僅適用於標準的 Modbus 網路，它定義了在這些網路上連續傳輸的消息段的每一位元，以及決定怎樣將資訊打包成消息欄位和如何解碼。

在其他網路上（象 MAP 和 Modbus Plus）Modbus 消息被轉成與串列傳輸無關的訊框。

1、ASCII 模式

當控制器設為在 Modbus 網路上以 ASCII（美國標準資訊交換代碼）模式通信，在消息中的每個 8Bit 位元組都作為兩個 ASCII 字元發送。這種方式的主要優點是字元發送的時間間隔可達到 1 秒而不產生錯誤。

代碼系統

- 十六進位，ASCII 字元 0...9，A...F
- 消息中的每個 ASCII 字元都是一個十六進位字元組成

每個位元組的位元

- 1 個起始位
- 7 個資料位元，最小的有效位先發送
- 1 個奇偶校驗位，無校驗則無
- 1 個停止位（有校驗時），2 個 Bit（無校驗時）

錯誤檢測欄位

- LRC(縱向冗長檢測)

2、RTU 模式

當控制器設為在 Modbus 網路上以 RTU（遠端終端單元）模式通信，在消息中的每個 8Bit 位元組包含兩個 4Bit 的十六進位字元。這種方式的主要優點是：在同樣的串列傳輸速率下，可比 ASCII 方式傳送更多的資料。

代碼系統

- 8 位元二進位，十六進位數 0...9，A...F
- 消息中的每個 8 位元欄位都是一個兩個十六進位字元組成

每個位元組的位元

- 1 個起始位
- 8 個資料位元，最小的有效位先發送
- 1 個奇偶校驗位，無校驗則無
- 1 個停止位（有校驗時），2 個 Bit（無校驗時）

錯誤檢測欄位

- CRC(循環冗餘核對)

三、Modbus 消息訊框

兩種傳輸模式中（ASCII 或 RTU），傳輸設備以將 Modbus 消息轉為有起點和終點的訊框，這就允許接收的設備在消息起始處開始工作，讀位址分配資訊，判斷哪一個設備被選中（廣播方式則傳給所有設備），判知何時資訊已完成。部分的消息也能偵測到並且錯誤能設置為返回結果。

3.1、訊框

3.1.1 ASCII 訊框

使用 ASCII 模式，消息以冒號(:)字元(ASCII 碼 3AH)開始，以回車換行符結束(ASCII 碼 0DH, 0AH)。其他欄位可以使用的傳輸字元是十六進位的 0...9, A...F。網路上的設備不斷偵測“:”字元，當有一個冒號接收到時，每個設備都解碼下個欄位（位址欄位）來判斷是否發給自己的。

消息中字元間發送的時間間隔最長不能超過 1 秒，否則接收的設備將認為傳輸錯誤。一個典型消息訊框如下所示：

起始位	設 備 位 址	功能代碼	數據	LRC 校驗	結束符
1 個字元	2 個字元	2 個字元	n 個字元	2 個字元	2 個字元

圖 2 ASCII 消息訊框

3.2.1 RTU 訊框

使用 RTU 模式，消息發送至少要以 3.5 個字元時間的停頓間隔開始。在網路串列傳輸速率下多樣的字元時間，這是最容易實現的(如下圖的 T1-T2-T3-T4 所示)。傳輸的第一個欄位是設備位址。可以使用的傳輸字元是十六進位的 0...9, A...F。網路設備不斷偵測網路匯流排，包括停頓間隔時間內。**當第一個欄位（位址欄位）接收到，每個設備都進行解碼以判斷是否發往自己的。**在最後一個傳輸字元之後，一個至少 3.5 個字元時間的停頓標定了消息的結束。一個新的消息可在此停頓後開始。

整個消息訊框必須作為一連續的傳輸。如果在訊框完成之前有超過 1.5 個字元時間的停頓時間，接收設備將刷新不完整的消息並假定下一位元組是一個新消息的位址欄位。同樣地，如果一個新消息在小於 3.5 個字元時間內接著前個消息開始，接收的設備將認為它是前一消息的延續。這將導致一個錯誤，因為在最後的 CRC 欄位的值不可能是正確的。一典型的消息訊框如下所示：

起始位	設備位址	功能代碼	數據	CRC 校驗	結束符
T1-T2-T3-T4	8Bit	8Bit	n 個 8Bit	16Bit	T1-T2-T3-T4

圖 3 RTU 消息訊框

3.2、欄位

3.2.1 位址欄位

消息訊框的位址欄包含兩個字元（ASCII）或 8Bit（RTU）。可能的從設備位址是 0...247（十進位）。單個設備的位址範圍是 1...247。主設備通過將要聯絡的從設備的位址放入消息中的位址欄來選定從設備。當從設備發送回應消息時，它把自己的位址放入回應的位址欄中，以便主設備知道是哪一個設備作出回應。

位址 0 是用作廣播位址，以使所有的從設備都能認識。當 Modbus 協定用於更高水準的網路，廣播可能不允許或以其他方式代替。

3.2.2 功能欄

消息訊框中的功能代碼欄包含了兩個字元（ASCII）或 8Bits（RTU）。可能的代碼範圍是十進位的 1...255。當然，有些代碼是適用於所有控制器，有些是應用於某種控制器，還有些保留以備後用。

當消息從主設備發往從設備時，功能代碼欄將告之從設備需要執行哪些行為。例如去讀取輸入的開關狀態，讀一組暫存器的資料內容，讀從設備的診斷狀態，允許調入、記錄、校驗在從設備中的程式等。

當從設備回應時，它使用功能代碼欄來指示是正常回應（無誤）還是有某種錯誤發生（稱作異議回應）。對正常回應，從設備僅回應相應的功能代碼。**對異議回應，從設備返回一等同於正常代碼的代碼，但將最高有效位元 (MSB) 設為邏輯 1。**

例如：一從主設備發往從設備的消息要求讀一組保持暫存器，將產生如下功能代碼：

0 0 0 0 0 0 1 1 （十六進位 03H）

對正常回應，從設備僅回應同樣的功能代碼。對異議回應，它返回：

1 0 0 0 0 0 1 1 （十六進位 83H）

除功能代碼因異議錯誤作了修改外，從設備將一獨特的代碼放到回應消息的資料欄中，這能告訴主設備發生了什麼錯誤。

主設備應用程式得到異議的回應後，典型的處理過程是重發消息，或者診斷發給從設備的消息並報告給操作員。

3.2.3 資料欄

資料欄是由兩個十六進位數集合構成的，範圍 00...FF。根據網路傳輸模式，這可以是由一對 ASCII 字元組成或由一 RTU 字元組成。

從主設備發給從設備消息的資料欄包含附加的資訊：從設備必須用於進行執行由功能代碼所定義的所為。這包括了像不連續的暫存器位址，要處理項的數目，欄位中實際資料位元組數。

例如，如果主設備需要從設備讀取一組保持暫存器（功能代碼 03），資料欄指定了起始暫存器以及要讀的暫存器數量。如果主設備寫一組從設備的暫存器（功能代碼 10 十六進位），資料欄則指明了要寫的要寫的起始暫存器以及要寫的暫存器數量，資料欄的資料位元組數，要寫入暫存器的資料。

如果沒有錯誤發生，從從設備返回的資料欄包含請求的資料。如果有錯誤發生，此欄位包含一異議代碼，主設備應用程式可以用來判斷採取下一步行動。

在某種消息中資料欄可以是不存在的（0 長度）。例如，主設備要求從設備回應通信事件記錄（功能代碼 0B 十六進位），從設備不需任何附加的資訊。

3.2.4 錯誤檢測欄位(CRC)

標準的 Modbus 網路有兩種錯誤檢測方法。錯誤檢測欄位的內容視所選的檢測方法而定。

ASCII

當選用 ASCII 模式作字元訊框，錯誤檢測欄位包含兩個 ASCII 字元。這是使用 LRC（縱向冗長檢測）方法對消息內容計算得出的，不包括開始的冒號符及回車換行符。LRC 字元附加在回車換行符前面。

RTU

當選用 RTU 模式作字元訊框，錯誤檢測欄位包含一 16Bits 值(用兩個 8 位元的字元來實現)。錯誤檢測欄位的內容是通過對消息內容進行循環冗餘核對(CRC)方法得出的。CRC 欄位附加在消息的最後，添加時先是低位元組然後是高位元組。故 CRC 的高位位元組是發送消息的最後一個位元組。

3.3、字元的連續傳輸

當消息在標準的 Modbus 系列網路傳輸時，每個字元或位元組以如下方式發送（從左到右）：

最低有效位 (LSB) -> 最高有效位 (MSB)

使用 ASCII 字元訊框時，位元的序列是：

有奇偶校驗

啓 始 位	1	2	3	4	5	6	7	奇 偶 位	停 止 位
----------	---	---	---	---	---	---	---	----------	----------

無奇偶校驗

啓 始 位	1	2	3	4	5	6	7	停 止 位	停 止 位
----------	---	---	---	---	---	---	---	----------	----------

圖 4. 位元順序 (ASCII)

使用 RTU 字元訊框時，位元的序列是：

有奇偶校驗

啓 位	始 位	1	2	3	4	5	6	7	8	奇偶位	停 止 位
--------	--------	---	---	---	---	---	---	---	---	-----	-------------

無奇偶校驗

啓 位	始 位	1	2	3	4	5	6	7	8	停 止 位	停 止 位
--------	--------	---	---	---	---	---	---	---	---	-------------	-------------

圖 4. 位元順序 (RTU)

四、錯誤檢測方法

標準的 Modbus 串列網路採用兩種錯誤檢測方法。奇偶校驗對每個字元都可用，訊框檢測 (LRC 或 CRC) 應用於整個消息。它們都是在消息發送前由主設備產生的，從設備在接收過程中檢測每個字元和整個消息訊框。

用戶要給主設備配置一預先定義的**超時時間間隔**，這個時間間隔要足夠長，以使任何從設備都能作為正常反應。如果從設備測到一傳輸錯誤，消息將不會接收，也不會向主設備作出回應。這樣超時事件將觸發主設備來處理錯誤。發往不存在的從設備的位址也會產生超時。

1、奇偶校驗

用戶可以配置控制器是奇或偶校驗，或無校驗。這將決定了每個字元中的奇偶校驗位元是如何設置的。

如果指定了奇或偶校驗，“1”的位數將算到每個字元的位元數中 (ASCII 模式 7 個資料位元，RTU 中 8 個資料位元)。例如 RTU 字元訊框中包含以下 8 個資料位元：

1 1 0 0 0 1 0 1

整個“1”的數目是 4 個。如果使用了偶校驗，訊框的奇偶校驗位將是 0，使得整個“1”的個數仍是 4 個。如果使用了奇校驗，訊框的奇偶校驗位將是 1，使得整個“1”的個數是 5 個。

如果沒有指定奇偶校驗位，傳輸時就沒有校驗位，也不進行校驗檢測。代替一附加的停止位填充至要傳輸的字元訊框中。

2、LRC 檢測

使用 ASCII 模式，消息包括了一基於 LRC 方法的錯誤檢測欄位。LRC 欄位檢測了消息欄位中除開始的冒號及結束的回車換行號外的內容。

LRC 欄位是一個包含一個 8 位元二進位值的位元組。LRC 值由傳輸設備來計算並放到消息訊框中，接收設備在接收消息的過程中計算 LRC，並將它和接收到消息中 LRC 欄位中的值比較，如果兩值不等，說明有錯誤。

LRC 方法是將消息中的 8Bit 的位元組連續累加，丟棄了進位。

LRC 簡單函數如下：

```
static unsigned char LRC(auchMsg, usDataLen)
unsigned char *auchMsg ; /* 要進行計算的消息 */
unsigned short usDataLen ; /* LRC 要處理的位元組的數量*/
{ unsigned char uchLRC = 0 ; /* LRC 位元組初始化 */
while (usDataLen--) /* 傳送消息 */
uchLRC += *auchMsg++ ; /* 累加*/
return ((unsigned char)(~((char_uchLRC))) ;
```

3、CRC 檢測

使用 RTU 模式，消息包括了一基於 CRC 方法的錯誤檢測欄位。**CRC 欄位檢測了整個消息的內容。**

CRC 欄位是兩個位元組，包含一 16 位元的二進位值。它由傳輸設備計算後加入到消息中。接收設備重新計算收到消息的 CRC，並與接收到的 CRC 欄位中的值比較，如果兩值不同，則有誤。

CRC 是先傳入所有值皆設為“1”的 16 位元暫存器，然後調用一副程式將消息中連續的 8 位元位元組各當前暫存器中的值進行處理。僅每個字元中的 8Bit 資料對 CRC 有效，起始位和停止位以及奇偶校驗位均無效。

CRC 產生過程中，每個 8 位元字元都單獨和暫存器內容做 OR 運算，結果向最低有效位方向移動，最高有效位以 0 填充。LSB 被提取出來檢測，如果 LSB 為 1，暫存器單獨和預置的值或一下，如果 LSB 為 0，則不進行。整個過程要重複 8 次。在最後一位（第 8 位）完成後，下一個 8 位元位元組又單獨和暫存器的當前值相或。最終暫存器中的值，是消息中所有的位元組都執行之後的 CRC 值。

CRC 添加到消息中時，低位元組先加入，然後高位元組。

CRC 簡單函數如下：

```
unsigned short CRC16(puchMsg, usDataLen)
unsigned char *puchMsg ; /* 要進行 CRC 校驗的消息 */
unsigned short usDataLen ; /* 消息中位元組數 */
{
unsigned char uchCRChi = 0xFF ; /* 高 CRC 位元組初始化 */
unsigned char uchCRCLo = 0xFF ; /* 低 CRC 位元組初始化 */
unsigned uIndex ; /* CRC 迴圈中的索引 */
while (usDataLen--) /* 傳輸訊息緩衝區 */
{
uIndex = uchCRChi ^ *puchMsg++ ; /* 計算 CRC */
uchCRChi = uchCRCLo ^ uchCRChi[uIndex] ;
uchCRCLo = uchCRCLo[uIndex] ;
}
return (uchCRChi << 8 | uchCRCLo) ;
}
```



```
/* CRC 高位位元組值表 */
```

```
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, 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  
}  
;
```

```
/* CRC 低位元位元組值表*/
```

```
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
} ;

ModBus 網路是一個工業通信系統，由帶智慧型終端的可編程式控制器和電腦通過公用線路或局部專用線路連接而成。其系統結構既包括硬體、亦包括軟體。它可應用於各種資料獲取和過程監控。下表 1 是 ModBus 的功能碼定義。

表 1 ModBus 功能碼

功能碼	名稱	作用
01	讀取線圈狀態	取得一組邏輯線圈的當前狀態（ON/OFF）
02	讀取輸入狀態	取得一組開關輸入的當前狀態（ON/OFF）
03	讀取保持暫存器	在一個或多個保持暫存器中取得當前的二進位值
04	讀取輸入暫存器	在一個或多個輸入暫存器中取得當前的二進位值
05	強置單線圈	強置一個邏輯線圈的通斷狀態
06	預置單暫存器	把具體二進值裝入一個保持暫存器
07	讀取異常狀態	取得 8 個內部線圈的通斷狀態，這 8 個線圈的位址由控制器決定，用戶邏輯可以將這些線圈定義，以說明從機狀態，短報文適宜於迅速讀取狀態
08	回送診斷校驗	把診斷校驗報文送從機，以對通信處理進行評鑒
09	編程（只用於 484）	使主機類比編程器作用，修改 PC 從機邏輯
10	控詢（只用於 484）	可使主機與一台正在執行長程式任務從機通信，探詢該從機是否已完成其操作任務，僅在含有功能碼 9 的報文發送後，本功能碼才發送
11	讀取事件計數	可使主機發出單詢問，並隨即判定操作是否成功，尤其是該命令或其他應答產生通信錯誤時
12	讀取通信事件記錄	可是主機檢索每台從機的 ModBus 事務處理通信事件記錄。如果某項事務處理完成，記錄會給出有關錯誤
13	編程（184/384 484 584）	可使主機類比編程器功能修改 PC 從機邏輯
14	探詢（184/384 484 584）	可使主機與正在執行任務的從機通信，定期控詢該從機是否已完成其程式操作，僅在含有功能 13 的報文發送後，本功能碼才得發送
15	強置多線圈	強置一串連續邏輯線圈的通斷
16	預置多暫存器	把具體的二進位值裝入一串連續的保持暫存器
17	報告從機標識	可使主機判斷編址從機的類型及該從機運行指示燈的狀態
18	（884 和 MICRO 84）	可使主機類比編程功能，修改 PC 狀態邏輯
19	重置通信鏈路	發生非可修改錯誤後，是從機重定於已知狀態，可重置順序位元組
20	讀取通用參數（584L）	顯示擴展記憶體檔中的資料資訊

21	寫入通用參數（584L）	把通用參數寫入擴展存儲檔，或修改之
22～64	保留作擴展功能備用	
65～72	保留以備用戶功能所用	留作用戶功能的擴展編碼
73～119	非法功能	
120～127	保留	留作內部作用
128～255	保留	用於異常應答

ModBus 網路只是一個主機，所有通信都由他發出。網路可支援 247 個之多的遠端從屬控制器，但實際所支援的從機數要由所用通信設備決定。採用這個系統，各 PC 可以和中心主機交換資訊而不影響各 PC 執行本身的控制任務。表 2 是 ModBus 各功能碼對應的資料類型。

表 2 ModBus 功能碼與資料類型對應表

代碼	功能	資料類型
01	讀	位
02	讀	位
03	讀	整型、字元型、狀態字、浮點型
04	讀	整型、狀態字、浮點型
05	寫	位
06	寫	整型、字元型、狀態字、浮點型
08	N/A	重複“回路反饋”資訊
15	寫	位
16	寫	整型、字元型、狀態字、浮點型
17	讀	字元型

（1）ModBus 的傳輸方式

在 ModBus 系統中有 2 種傳輸模式可選擇。這 2 種傳輸模式與從機 PC 通信的能力是同等的。選擇時應視所用 ModBus 主機而定，每個 ModBus 系統只能使用一種模式，不允許 2 種模式混用。一種模式是 ASCII（美國資訊交換碼），另一種模式是 RTU（遠端終端設備）這兩種模式的定義見表 3

表 3 ASCII 和 RTU 傳輸模式的特性

特性		ASCII (7 位)	RTU (8 位)
編碼系統		十六進位（使用 ASCII 可列）	二進位

(2) ModBus 的資料校驗方式

CRC-16 (迴圈冗餘錯誤校驗)

CRC-16 錯誤校驗程式如下：報文（此處只涉及資料位元，不指起始位、停止位和任選的奇偶校驗位）被看作是一個連續的二進位，其最高有效位元（MSB）首選發送。報文先與 X^{16} 相乘（左移 16 位），然後看 $X^{16}+X^{15}+X^2+1$ 除， $X^{16}+X^{15}+X^2+1$ 可以表示為二進位數字 1100000000000101。整數商位元忽略不記，16 位餘數加入該報文（MSB 先發送），成為 2 個 CRC 校驗位元組。餘數中的 1 全部初始化，以免所有的零成為一條報文被接收。經上述處理而含有 CRC 位元組的報文，若無錯誤，到接收設備後再被同一多項式（ $X^{16}+X^{15}+X^2+1$ ）除，會得到一個零餘數（接收設備核驗這個 CRC 位元組，並將其與被傳送的 CRC 比較）。全部運算以 2 為 MOD（無進位）。

習慣於成串發送資料的設備會首選送出字元的最右位元（LSB-最低有效位）。而在生成 CRC 情況下，發送首位應是被除數的最高有效位 MSB。由於在運算中不用進位，為便於操作起見，計算 CRC 時設 MSB 在最右位。生成多項式的位序也必須反過來，以保持一致。多項式的 MSB 略去不記，因其只對商有影響而不影響餘數。

生成 CRC-16 校驗位元組的步驟如下：

- ①裝如一個 16 位暫存器，所有數位均為 1。
- ②該 16 位元暫存器的高位位元組與開始 8 位元位元組進行“XOR”運算。運算結果放入這個 16 位暫存器。
- ③把這個 16 暫存器向右移一位。
- ④若向右（標記位元）移出的數位是 1，則生成多項式 1010000000000001 和這個暫存器進行“XOR”運算；若向右移出的數位是 0，則返回③。
- ⑤重複③和④，直至移出 8 位。
- ⑥另外 8 位與該十六位暫存器進行“XOR”運算。
- ⑦重複③~⑥，直至該報文所有位元組均與 16 位元暫存器進行“異或”運算，並移位 8 次。
- ⑧這個 16 位元暫存器的內容即 2 位元組 CRC 錯誤校驗，被加到報文的最高有效位。

另外，在某些非 ModBus 通信協議中也經常使用 CRC16 作為校驗手段，而且產生了一些 CRC16 的變種，他們是使用 CRC16 多項式 $X^{16} + X^{15} + X^2 + 1$ ，單首次裝入的 16 位暫存器為 0000；使用 CRC16 的反序 $X^{16}+X^{14}+X^2+1$ ，首次裝入暫存器值為 0000 或 FFFFH。

LRC (縱向冗餘錯誤校驗)

LRC 錯誤校驗用於 ASCII 模式。這個錯誤校驗是一個 8 位元二進位數字，可作為 2 個 ASCII 十六進位位元組傳送。把十六進位字元轉換成二進位，加上無迴圈進位元的二進位字元和二進位補數結果生成 LRC 錯誤校驗（參見圖）。這個 LRC 在接收設備進行核驗，並與被傳送的 LRC 進行比較，冒號（:）、回車符號（CR）、換行字元（LF）和置入的其他任何非 ASCII 十六進位字元在運算時忽略不計。

表 5 LRC 生成範例——讀取 02 號從機的前 8 個線圈

十六進位	二進位
------	-----

位址	0	2	0000	0010
功能碼	0	1	0000	0001
起始位址高位	0	0	0000	0000
起始位址低位	0	0	0000	0000
單元數量	0	0	0000	0000
	0	8	+	0000 1000
				0000 1011
			變成補數	1111 0101
錯誤校驗	F	5	F	5
				0000 0010
接受 PC 把所有收到的資				0000 0001
料位元組（包括最後的				0000 0000
LRC）加在一起，8 位元				0000 0000
應全部為 0（注意：和可				0000 0000
能超過 8 位，應略去最低				0000 1000
位）			錯誤校驗	1111 0101
			和	0000 0000

摘自：<http://www.chinakong.net>