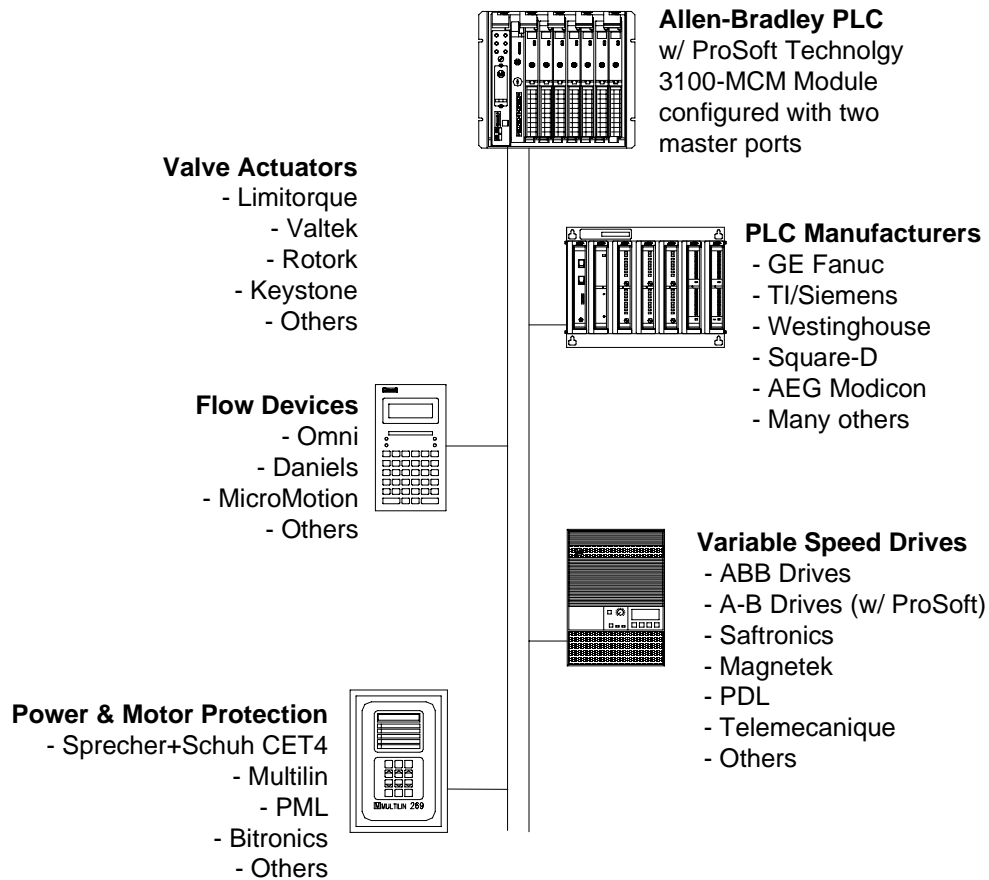


3100/3150-MCM Modbus 通讯

修订 2
2002 年 9 月



用户手册

ProSoft Technology, Inc.
1675 Chester Avenue, 4th Floor
Bakersfield, CA 93301
(661) 716-5100
(661) 716-5101 (fax)

E-mail 地址: prosoft@prosoft-technology.com

网址: <http://www.prosoft-technology.com>

请阅读以下注意事项

成功的应用这个模块需要对 **Allen-Bradley PLC/SLC** 硬件知识和现场应用方式有充分的了解。因此，对于负责完成应用的工作人员，了解应用需求并确保人员和设备不处于不安全或不适当的工作环境是非常重要的。

这本手册是用作帮助用户。我们力求提供的每个信息都是准确的，而且如实的反映产品的安装要求。为确保对本产品操作的完全理解，用户必须阅读有关 A-B 硬件操作的所有 Allen-Bradley 应用文档。

在任何条件下，ProSoft Technology, Inc.都不负责间接的或由用户使用或应用本产品而造成的损害。

在没有得到 ProSoft Technology 许可的情况下，禁止任何对本手册内容整体或部分性的复制。

本手册内容如有更改，恕不通知。ProSoft Technology, Inc.并不承担这个义务。并会随时改进和/或更改此文档或产品。这些更改会阶段性的进行，以更改技术的不准确和印刷排版错误。

修订 2 中的新特性

MCM 产品的修订 2 版本是对这个产品的第一次主要升级。产品集合了许多新特性，其中一些源自我们目前用户的建议，还有一些是我们在考察了我们的用户在使用这个产品上总结出来的。其中一些新特性是：

Modbus 主站驱动

命令状态位

在返回到梯形逻辑程序的信息中增加了命令“完成”和“错误”位。这些位使梯形逻辑程序可以跟踪每个 Modbus 主站命令执行和成功或失败。结合新的**命令控制模式**（见后），梯形程序现在可以控制每个命令的执行。

命令轮询时间

每个命令现在可以设置一个“轮询时间（Polling Time）”参数来有效的控制命令执行频率。其计时器的分辨率是 1 秒，最多可以设置到 65535 秒。

命令控制模式

命令控制模式大概是我们要被最多要求添加的功能。在这个模式下，Modbus 主站驱动仅到接收到来自梯形逻辑“使能（Enable）”位时，才执行命令列表内的命令。模块在定期传送的命令状态信息里加入“完成”和“错误”位，返回到梯形逻辑（参考上面的讨论），这就能让命令按照需要只发送一次。

事件初始写命令

MCM 模块中的 Modbus 主站驱动支持执行**事件驱动写命令**（Event Driven Write Commands）。这种支持提供了梯形逻辑程序除传统写命令（如说明书中陈述）和命令控制模式之外，其它的方法发送 Modbus 写命令到从站设备（FC 5, 6, 15, 16）。

Modbus 从站驱动

Pass-through 模式

Pass-Through 模式使 Modbus 从站端口直接把从主站那里接收到的写命令，通过背板发送给梯形逻辑处理。尽管这种特性需要更多的梯形逻辑来完成处理，但确实一些环境下非常适用。这些环境状况包括：

1. 当从站需要知道何时被写入时
2. 当需要某些条件才能接受数据时
3. 当主站的写数据寄存器必须和读寄存器空间重叠时

路由模式

路由模式使 Modbus 从站端口把从主站接收到的命令“路由”到主站端口。在设置表中可以输入多达 6 个路由从站地址。从站端口无论何时接收到主站命令（读或者写），只要这些命令符合 6 个地址中的任意一个。这些命令就会被路由到主站端口发出，而且回应也会路由回从站端口。

快速启动实施指南

如果按照下列步骤实施，把 MCM 模块集成到 PLC 或 SLC 应用中会变得更加轻松。为了帮助我们产品的新用户更快地上手操作，我们提出这个循序渐进的实施指南。



新用户

尽管下列步骤能帮助您应用这个模块，我们仍建议您在安排您的应用前，先尝试使用我们随模块提供的样例程序，这些样例程序也可以从我们的 FTP 上下载。这个步骤可以让您在决定那些因素会影响长期成功安装之前，先了解模块是如何工作的。

从 MCM 模块附带光盘上提供的程序之一起步，完成下列步骤：

如果进入的是 SLC 梯形逻辑程序，记得下列步骤：

- 把插槽设置成 5/02 模式下的 1746-BAS 模块
- 请确认如样例梯形逻辑那样，输入传送使能（Transfer Enable）和完成位

- 按照应用的需要修改光盘上提供的梯形逻辑程序（参考章节 3.0）
 - 校验程序中槽架和槽的位置
 - 按需要修改梯形指令地址
- 设置通讯配置参数（参考章节 4.2）
 - 确定每个端口的通讯配置需求：
 - 主站或从站，奇偶校验，停止位，波特率，RTS 延迟需求
 - 确认内存映射需求
 - 设置读数据，写数据和命令块个数参数
 - 设置应用所需的从站和主站错误表指针
- 如果是设置主站，需配置命令列表（参考章节 4.4）
 - 注意检查从站设备的寄存器映射，方便建立最有效的内存映射
- 确认模块跳线需求（参考附录 D）
- 制作通讯电缆（参考章节 8）。无论在任何一种连接下都要确保跳线满足 CTS 信号的要求。通常情况下，这个信号都会短接到 RTS。
- 把处理器切换到运行模式
- 在数据表中查看主站和从站错误状态数值（参考章节 5.1）

'ProSoft Tested' 测试文档

经过我们“ProSoft Tested”计划的努力，我们得出了一份我们已知可以和我们的模块通讯的设备列表，而且列表的内容也在不断增加。此外，我们也记录了一些我们测试过的设备。如需了解详情，请访问我们的下列网站：

<http://www.prosoft-technology.com>
选择 'Web Site Index'
选择 'MCM Connectivity Listing'
选择 'Test Document' 中需要的产品

目录

1	功能概述	1
1.1	总体	1
1.2	硬件概述	1
1.3	基本概念	2
1.3.1	模块上电和重启	3
1.3.2	主循环逻辑	3
1.3.3	模块的数据空间	4
1.3.4	背板数据传输过程	5
1.3.5	块传输的联锁	8
1.3.6	SLC 处理器设置	9
1.4	数据流	9
1.4.1	基本概念	9
1.4.2	从模块读取数据	10
1.4.3	向模块写数据	10
1.4.4	主站端口驱动	10
1.4.5	从站端口驱动	10
1.4.6	从站端口 - 普通模式	11
1.4.7	从站端口 - Pass Through 模式	11
1.4.8	从站端口 - 路由模式	12
1.5	Modbus 寻址	13
1.5.1	Modbus 寻址概念	13
1.5.2	MCM 支持的 Modbus 功能	13
1.5.3	把 Modbus 地址映射到梯形数据地址	14
2	启动 - 一个按部就班的步骤	15
3	梯形逻辑概述	16
3.1	运作概述	16
3.2	梯形逻辑	16
4	写入模块	17
4.1	块传输到模块	17
4.2	通讯设置 [BTW 块 ID 255]	18
4.3	写入模块数据内存 [BTW 块 ID 代码 0-79]	23
4.3.1	写数据到模块的梯形逻辑	23
4.3.2	块传输数据结构	25
4.4	命令列表设置 - 主站模式 [BTW 块 ID 代码 80-99]	25
4.4.1	命令列表梯形逻辑	25
4.4.2	命令列表结构	26
4.4.3	修改命令列表	28
4.5	命令控制模式 - 主站模式	28
4.5.1	BTW 块结构	28
4.5.2	控制命令	29
4.5.3	命令列表示例	29
4.6	事件触发命令 - 主站模式 [BTW 块 ID 代码 100 到 119]	29
4.6.1	梯形逻辑	30
4.6.2	BTW 块结构	30
5	读取模块	32
5.1	从模块传输数据 [BTR 块 ID 0 到 79]	32
5.1.1	读数据块结构	32
5.1.2	从模块移动数据到处理器	33
5.1.3	读模块数据的梯形逻辑	33
5.1.4	从站错误代码表	34
5.1.5	主站错误代码表	35
5.1.6	错误状态代码	36

5.2	Pass-Through 模式 - 从站模式 [BTR 块 ID 256 到 259]	37
5.2.1	块结构	37
5.2.2	接收寄存器写 [BTR 块 ID 256 和 257]	38
5.2.3	接收一个位写 [BTR 块 ID 258]	38
5.2.4	接收多个位写 [BTR 块 ID 259]	38
5.3	解码命令完成和命令错误位 - 主站模式	39
5.3.1	块结构	39
5.3.1	梯形逻辑	39
6	Modbus 命令设置	41
6.1	Modbus 命令	41
6.2	浮点数支持	43
6.2.1	ENRON 浮点数支持	44
7	诊断和故障排除	45
7.1	3100 PLC 平台 LED 指示	45
7.2	3150 SLC 平台 LED 指示	45
7.3	纠错 - 常规	46
8	接线连接示意图	49
A	支持, 服务和保修	51
B	产品规格	53
C	Modbus 协议规格	55
D	跳线设置	61
E	产品修订历史	63
F	读, 写和命令块计数数值使用	64
G	样例梯形逻辑程序	65

(请参考在应用样例说明书中单独页面提供的样例梯形程序)

1 功能概述

这个章节的目的是给读者提供 MCM 模块运作的总体概念。有关梯形逻辑程序和背板数据传输的具体内容会在接下来的章节和附录中讨论。

1.1 总体

MCM 产品是单插槽式的模块，它被设计成把 Modbus 通讯接口紧密的集成到 Allen-Bradley 1771 和 1746 I/O 平台。产品支持下列处理器：

3100-MCM 基于 1771 平台

PLC 5 系列
PLC 2 系列
PLC 3 系列

3150-MCM 基于 1746 平台

SLC 5/02, 5/03, 5/04

模块既可以在本地槽架和处理器一同工作，也可以安装到远程槽架使用远程 I/O 通讯连接槽架（PLC），或者是安装到扩展槽架（SLC）。

产品的两种外形如下图所示：



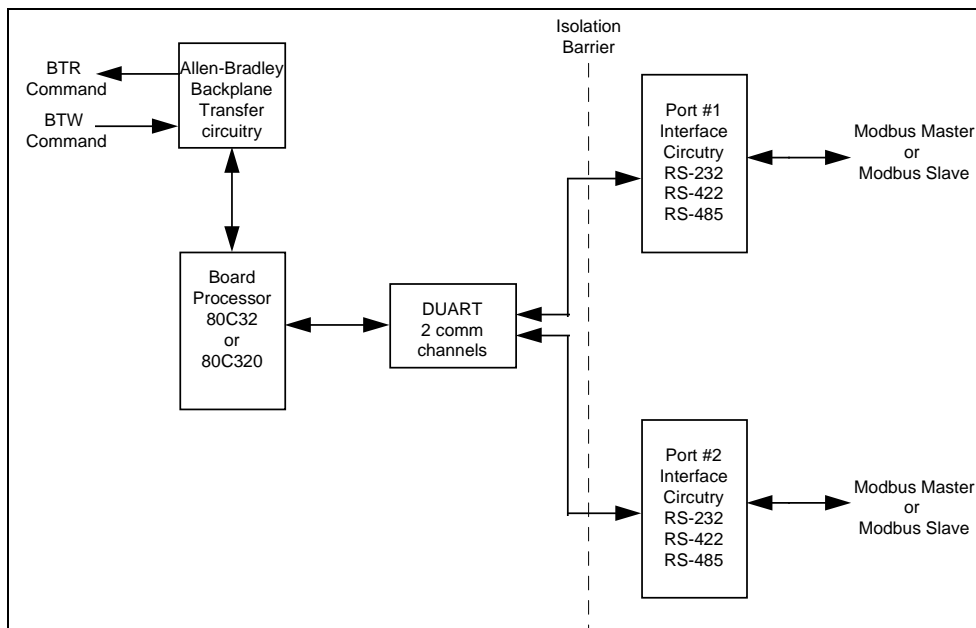
3100 Module
1771 Platform



3150 Module
1746 Platform

1.2 硬件概述

MCM 模块在两个硬件平台上的设计非常相似。在下面的说明中，除特别指出外，将适用于 3100 和 3150 平台。下图说明了模块的功能组成构成。



3100/3150 模块的硬件布局图

板上主要的功能部分有：

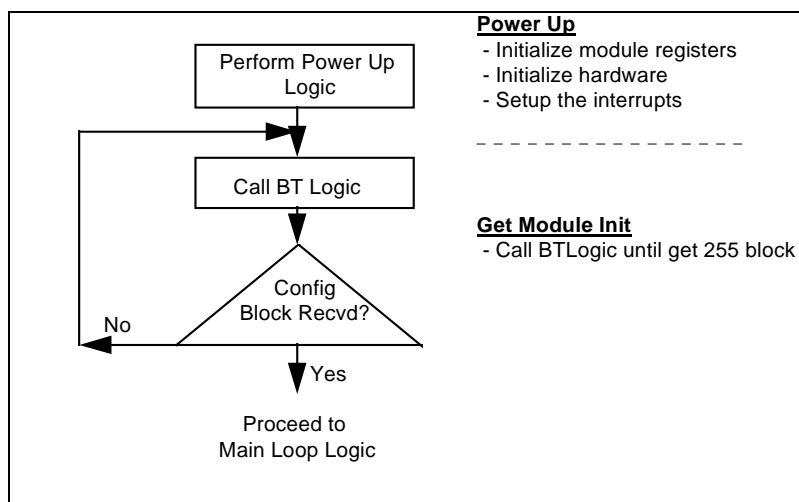
- 一个微控制器，负责主板上所有运作，包括：
 - 和 Allen-Bradley 处理器的背板通讯
 - 从模块向 PLC 传送数据
 - 接收来自 PLC 的数据，放置到模块内
 - 服务 DUART 通讯
 - LED 状态指示
- 一个 Allen-Bradley 背板芯片负责服务模块和 A-B 处理器之间的通讯。芯片包含了 A-B 版权的专利技术，设计成：
 - 在 PLC 中，芯片设计成使用块传输（Block Transfer）命令和背板通讯，每次传输 64 个字
 - 在 SLC 中，芯片设计成使用 M0/M1 文件和背板进行通讯。由于在 SLC 中并没有真正的块传输功能，我们通过使用 I/O 表来控制模块和处理器的硬件握手，从而完成一种形似块传输的功能。每次可传输 64 个字。如下所示，假设模块在 1 槽中，是这些位：
 - I:1/0 传输使能（Transfer Enable）
这个位由模块来写，梯形逻辑程序使用这个位来允许背板上的数据移动
 - O:1/0 传输完成（Transfer Done）
这个位由梯形逻辑程序来写，和模块通讯并告知梯形程序已经完成了数据传输
- 端口接口电路提供到真实世界的物理接口。端口和接口电路和卡的其它部分是光隔离的，特别是背板，这提供了对 A-B 处理器的高级别保护。两个端口都能支持：
 - RS-232
 - RS-422, 也称作 4 线连接
 - RS-485, 也称作 2 线连接

1.3 基本概念

下面的讨论包括了一些对于了解 ProSoft 模块运作非常重要的概念。

1.3.1 模块上电和重启

上电时，或按重启按钮后（仅 3100），模块开始执行其逻辑功能。下面的图表显示了这些功能



这包括：

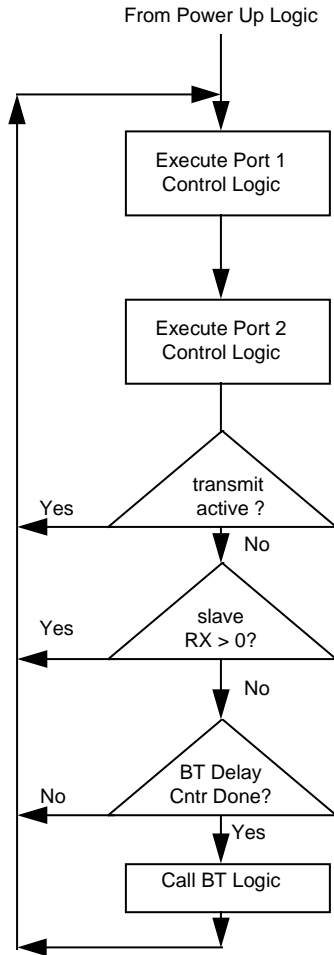
1. 初始化硬件
 - 初始化背板
 - 初始化 DUART
2. 初始化模块寄存器
 - 清除模块数据块
 - 清除命令列表
 - 清除错误状态表
 - 重置常量

在寄存器空间初始化后，模块就开始和梯形逻辑进行块传输。从模块发送出的第一个块传输会启动设置进程，使梯形逻辑程序把代码为 255 的块传送到模块。当模块已设置后，模块就会开始主逻辑循环。

1.3.2 主循环逻辑

完成上电设置过程后，模块会跳到一个无限循环中，执行以下功能：

1. 端口 1 和端口 2 处理程序
 - 检测讯息结束条件
 - 调用讯息处理程序
 - 初始化命令（主站）
2. 块传输
 - 检测 CTS 信号，确保模块没有处于发送模式
 - 检测从站端口缓冲器，确保没有接收
 - 检测块传输延迟计数
 - 如果一切正常，就开始块传输



Port 1 Control Logic

- if port in RX mode, then test for message received
- if port in TX mode, then test if message transmit has completed
- If port is a master and ready for new command then create new command

Port 2 Control Logic

- if port in RX mode, then test for message received
- if port in TX mode, then test if message transmit has completed
- If port is a master and ready for new command then create new command

Test Transmit Status

- If either port is in process of transmitting then do not execute BT logic. The module uses the CTS pin status to detect the transmit status

Test Slave port for characters

- If either slave port is in the process of receiving characters then skip BT logic. This is done to assure timely response from the slave

Test the Block Transfer Delay Counter

- If the Block Transfer Delay counter has incremented beyond the counter preset (set in config) then go ahead and perform block transfer

Execute Block Transfer Logic

- Calls BT Logic which executes the BTR and BTW logic

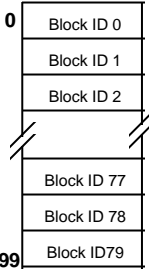
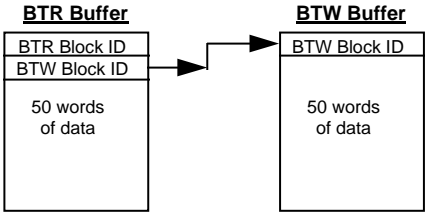
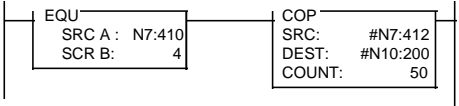
1.3.3 模块的数据空间

有助于理解模块的一个重要概念是理解模块内部数据空间和这部分数据如何在模块和处理器交换之间的关系。

下面的讨论解释模块的数据结构，以及这些数据如何在模块和梯形逻辑程序交换。一些理解要点：

要点	描述
模块数据寄存器空间尺寸	模块包含一个 4000 个字的数据空间，可以应用于数据存储 <div style="text-align: center;"> <p>Module Memory</p> <p>4000 word block of 16 bit registers</p> <p>Addresses : 0 to 3999</p> </div>

4000 个字的模块数据空间如何划分 | 这 4000 个字的数据块逻辑划分为 80 个 50 字长的

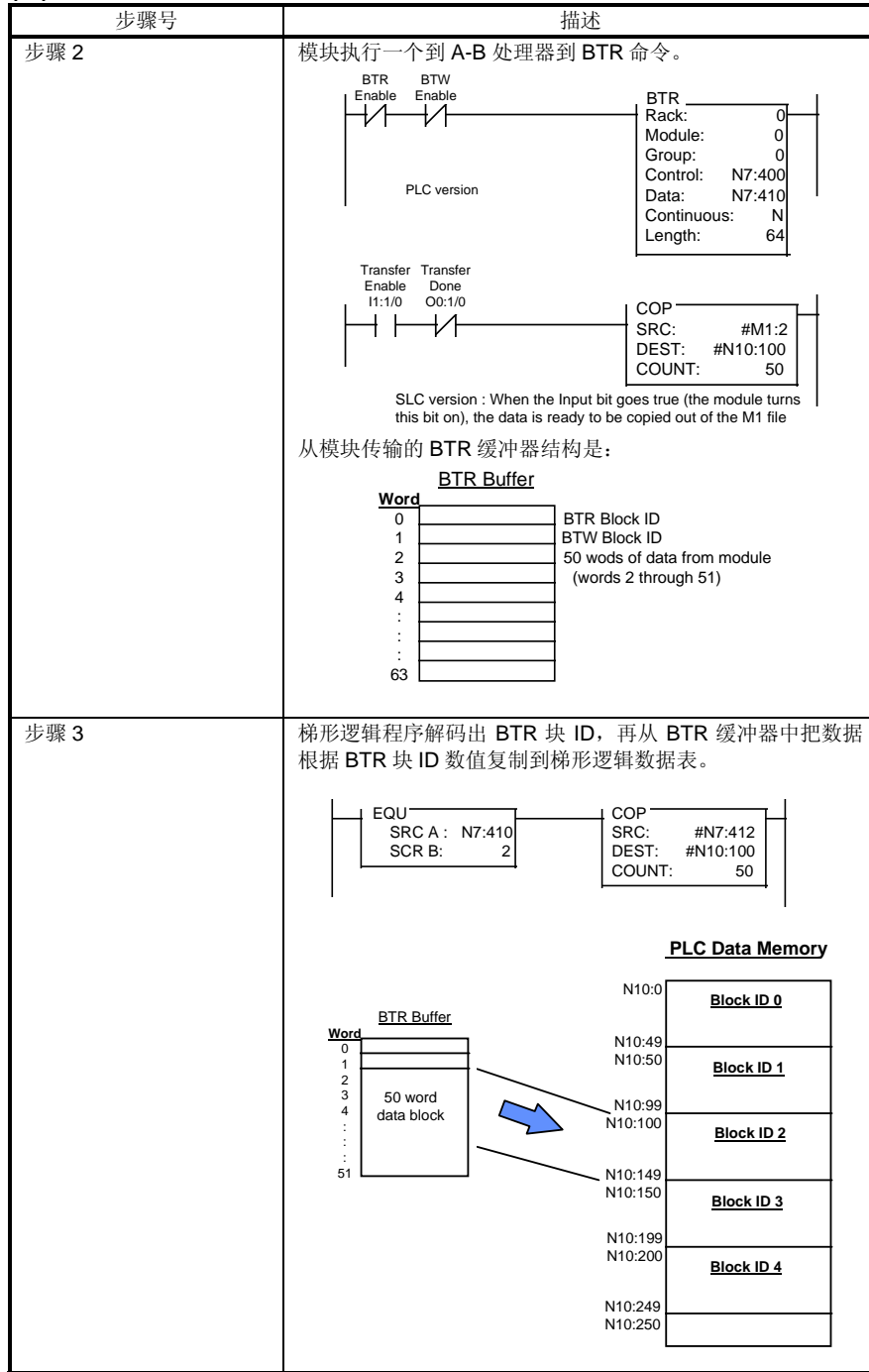
到块中	<p>块:</p> <p>80 块 x 50 字/块 = 4000 字</p> <p>Module Memory 4000 word block of 16 bit registers Block ID 0 to 79 Addresses : 0 to 3999</p> 
数据在模块和处理器之间如何分页传输	<p>按照后面章节内描述的数据传输次序，50 字长的数据块（或“页”）在模块和处理器之间双向传输。</p> 
数据“页”如何放置到处理器数据表中	<p>在 PLC/SLC 处理器中数据的放置受控于用户和应用梯形逻辑程序。处理器中任何可利用的数据文件都可以作为模块的数据源或作为源自于模块的数据目标。</p> 

1.3.4 背板数据传输过程

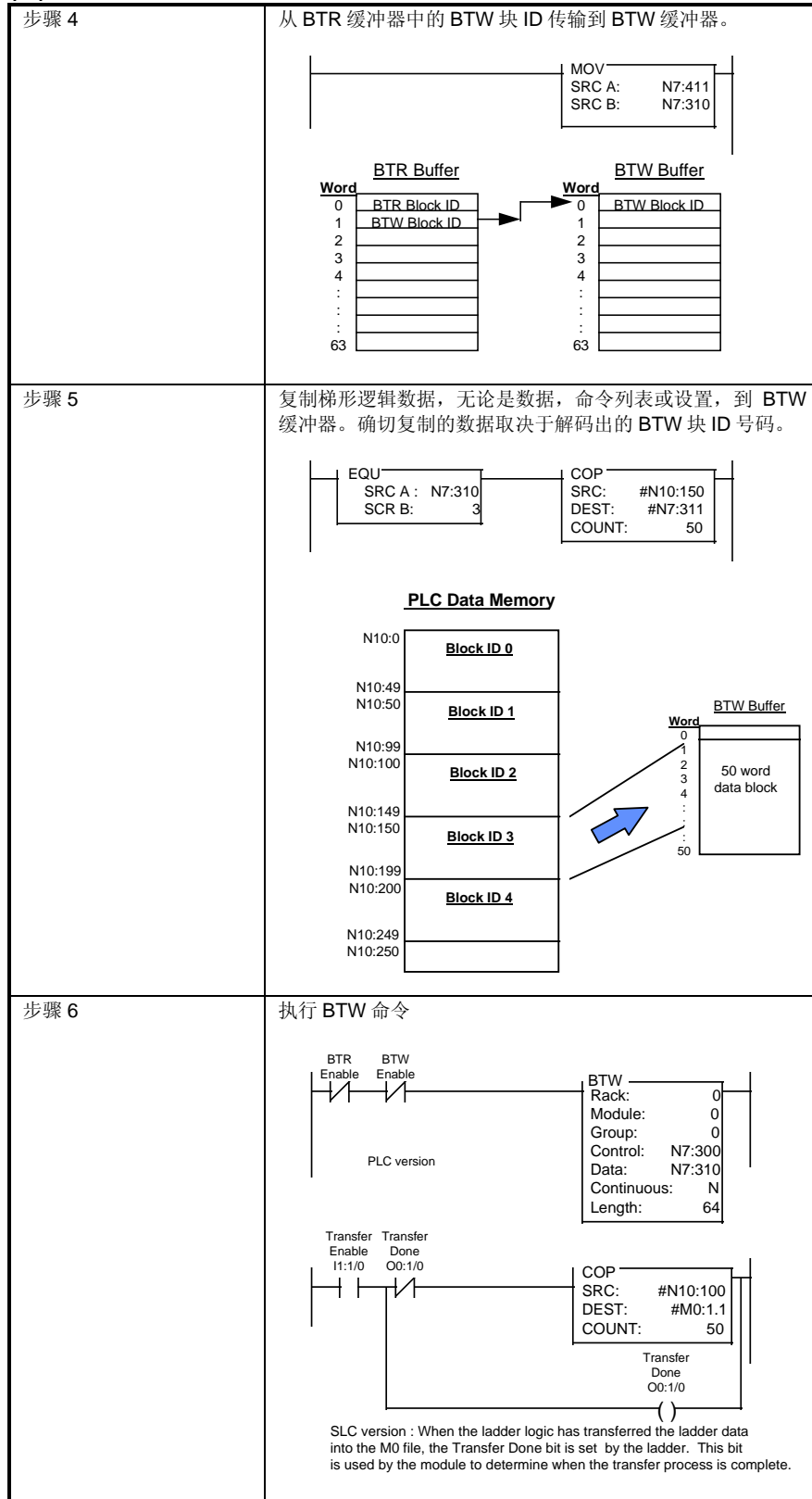
下面的表格给出了 A-B 处理器和模块之间数据传输的概况。处理器中的梯形逻辑程序有效的控制了这个过程。通过下面的内容可以了解模块和梯形逻辑中完成一次成功传输的步骤。参考附录中的样例程序可以看到一个实际的过程。

步骤号	描述
步骤 1	<p>模块根据下面的逻辑生成 BTR 和 BTW 块 ID 号码:</p> <p>BTW Block ID</p> <pre> if (BTW Block ID >= Write Block Cnt) then BTW Block ID = 80 elseif(BTW Block ID >= 80 + Command Block Cnt) then BTW Block ID = Write Block Start else BTW block ID = BTW block ID + 1 </pre> <p>BTR Block ID</p> <pre> if (BTR Block ID >= Read Block Cnt) then BTR Block ID = Read Block Start else BTR block ID = BTR block ID + 1 </pre>

(续)



(续)



(续)

步骤 7	模块接收 BTW 数据。在解码出 BTW 块 ID 号码后，模块会把 BTW 缓冲器中的数据传送到模块内存中正确的位置。
------	--------------------------------------------------------------

1.3.5 块传输的联锁

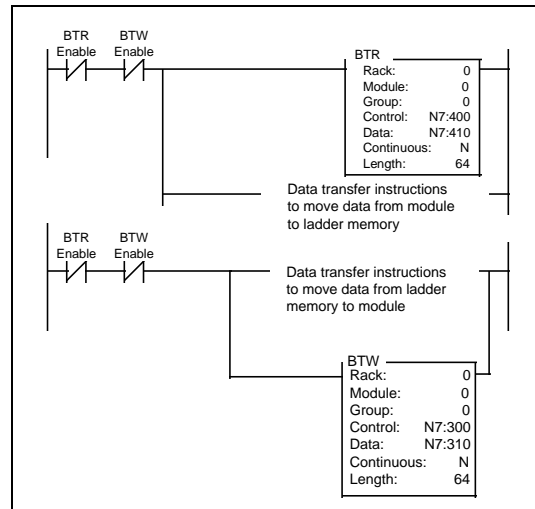
一个基本性的假设是模块认为每个 BTW 命令只有一个 BTR 命令。在模块内部，完成 BTR 指令后，模块会立刻跳转到 BTW 指令。如果程序员以我们的样例程序为基础编程序，就不会有大的问题。

但是当梯形逻辑程序试图完成不符合模块块传输预期的工作时就会有问题。这里特别指出，当为模块编程序时必须符合以下几点：

使用 BTR/BTW 指令的 PLC 编程

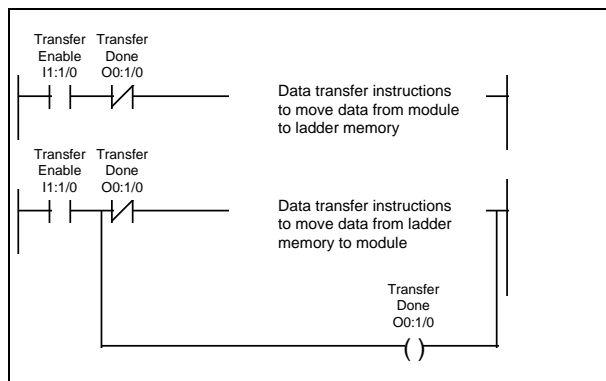
在 1771 类型的处理器中（PLC 2, PLC 3 和 PLC 5），必须写 BTR 和 BTW 的 Enable 位才能触发块传输指令。这种类型的编程保证了 PLC 不会同时执行两个块传输指令，也确保了 BTR 和 BTW 指令相互切换。

在 A-B 文档中有大量这种类型的块传输编程说明，同样在附录的样例程序中也有这样的编程示例。



使用 M0/M1 指令的 SLC 编程

在 SLC 处理器内并没有类似 PLC 平台那样真正能够保障完整数据块传输的机制。因此，我们开发了一种握手机制，确保 M0 和 M1 文件中所有的字和谐地传输。只有利用这种机制，我们才能保证根据块 ID 传输相应的数据。完成在 SLC 中的应用程序至少要有这样的基础结构：



1.3.6 SLC 处理器设置

在初始设置 SLC 程序文件，或当把模块从一个槽位到另一个槽位时，用户笔削设置这个槽位来接受 MCM 模块。

非常重要，容纳 ProSoft 模块的槽位必须经过如下设置：

- 1746-BAS 模块，5/02 或更高的配置
- 或输入 13106 作为模块 ID 代码
- 设置 M0/M1 文件成 64 个字
- 设置 8 个字的 I/O

下面是使用 Allen-Bradley APS 软件设置这些文件的步骤。其它软件包的用户应该依照类似的步骤。

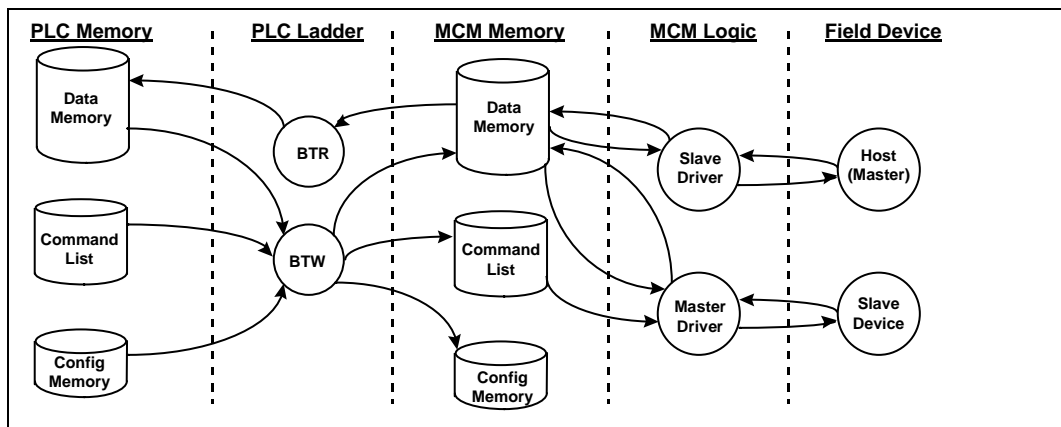
从主菜单：

- 1) 选择正确的处理器程序和 F3（离线编程）
 - 2) F1，处理器功能
 - 3) F1，更改处理器
如果需要，在这里修改处理器（注意：MCM 模块只能适配 5/02 或更高级别的处理器）
 - 4) F5，设置 I/O
配合 SLC 5/02 或更高级别处理器选择 1746-BAS 模块，或输入 13106 的模块代码
 - 5) F9，当选择到正确的槽位时设置 SPIO
 - 6) F5，高级设置
 - 7) F5，设置 M0 文件长度—输入 64，按回车
 - 8) F6，设置 M1 文件长度—输入 64，按回车
- Esc 退出和保存设置

1.4 输入流

1.4.1 基本概念

要彻底理解模块的运作，重要的是理解在梯形逻辑程序，模块和主站/从站驱动之间的数据流。



下面的讨论涉及在不同阶段内的数据流动。在后面的章节中会有对不同端口运行模式下的数据流的讨论。

1.4.2 从模块读取数据

模块包括一个 4000 块的数据内存。这个内存包含：

1. 主站端口的处理结果
2. 通过从站端口移动数据到模块
3. 从站端口状态数据
4. 模块修订信息
5. 主站端口状态数据

在把数据从模块传送到 PLC 的过程中，梯形逻辑程序可以读取到这些信息。

1.4.3 向模块写数据

模块，根据端口的设置，需要 3 个基本类型的数据才能正确工作。这 3 种类型的内存是可以传送到模块的，包括：

1. 设置数据。这些数据包含所有模块需要用来设置串口和设置模块和梯形逻辑程序建数据传输的参数
2. 命令列表。这个数据集包含所有模块需要用来构造有效命令的参数，这些命令从主站端口发出到其它的 Modbus 从站设备。最多有 20 个命令列表块可以发送到模块，相当于 100 条命令
3. 数据内存。这种内存向模块移动，向从站端口提供数据服务读请求（比如，主站作为读命令的结果所需要的数据），或向主站端口服务写请求（比如，写到从站的数据）。

1.4.4 主站端口驱动

在一般的应用中，主站端口被用来发送读命令到从站设备，从而作为一个数据集合器，接着发送从梯形逻辑程序读取的数据。

模块使用 命令列表 (Command List) 来编制有效的 Modbus 命令。当执行每个命令时，模块扫描命令列表中下一个条目。如果主站端口发送的是一个读命令，读的结果存放到数据内存中。如果主站端口发送的是一个写命令，数据内存中的数据被写入到从站设备。

对应每个模块执行的命令，命令的状态都可以在主站错误表 (Master Error Table) 中找到。这个表可以安排在数据内存块中任意的的位置，同时作为常规数据传输过程的一部分被读回到梯形逻辑程序。

1.4.5 从站端口驱动

从站端口的运作完全是从站端口设置的功能。端口可以运作在 3 种模式下：

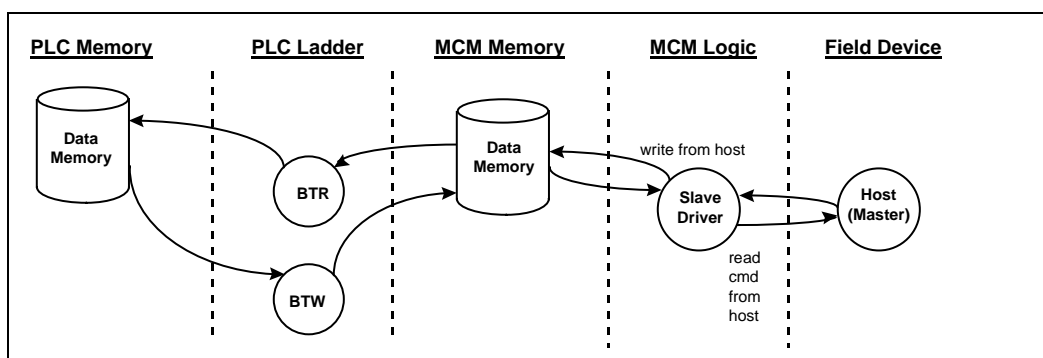
1. 普通模式。在这个模式下，直接用数据内存块来服务主站读命令，主站写命令数据也直接发送到数据内存
2. Pass Through 模式。在这个模式下，同普通模式一样直接用数据内存服务主站读命令。而原则性的区别在于主站写命令被发送到 BTR 缓冲器，用于梯形逻辑程序处理。
3. 路由模式。此模式只应用于一个端口设置为主站，其它端口设置为从站的情况。在这个模式下，从站驱动根据路由表（用户设置）中的 6 个条目来检查每个命令的从站地址。如果互相符合，从站端口接收到命令被路由到主站端口发出。模块接收到从站的响应，就发送到从站端口从而发送给主站。

下面的章节提供了一些数据流的图表，帮助理解各种模式。

1.4.6 从站端口-普通模式

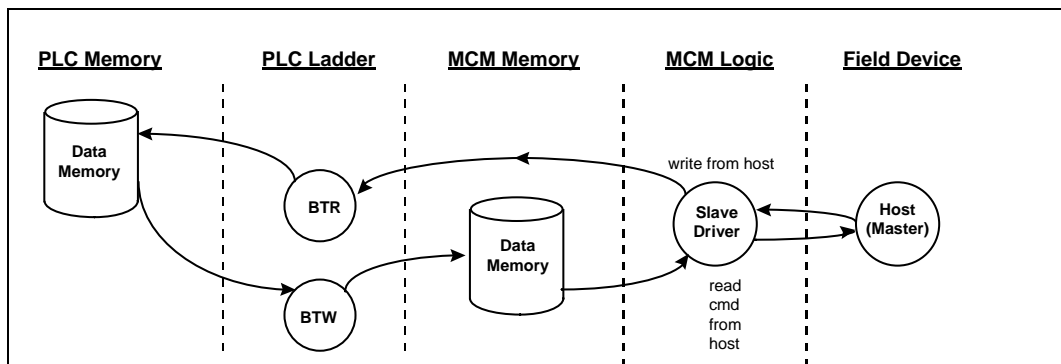
在普通模式下（比如不开启任何特殊功能）从站端口使用数据内存来服务所有收到的读和写请求。这种模式最大的好处是设置简单，仅需要很少的梯形逻辑程序就可以完成应用。

最主要的要求是主站写进的数据空间不能和梯形逻辑程序写进的数据空间不能重叠。在多数情况下，这种限制不会引起任何问题，这也是设置模块的最佳模式。



1.4.7 从站模式 - Pass Through 模式

Pass Through 模式具有能克服普通模式所引起的限制的能力。如前文所述，从站端口接收到的主站写命令不能和梯形逻辑程序写进的数据空间重叠，否则不能使用普通模式。



在 Pass Through 模式里，模块接收的所有写命令（目标地址是本站站地址或广播命令）都传输到 BTR 缓冲器，由梯形逻辑程序处理。这种模式有很多优点：

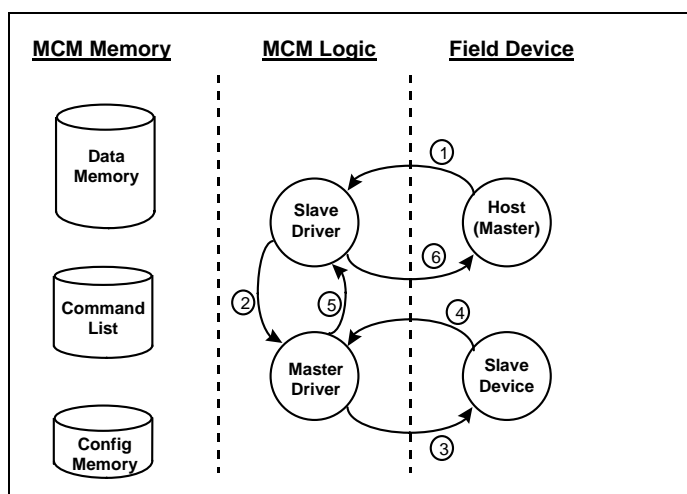
1. 主站命令可以直接读写到梯形逻辑程序数据表
2. 通过编写梯形逻辑程序可以有条件的接收所有写命令，比如地址范围，数值范围，等
3. 梯形逻辑程序能够确定何时从站端口从主站那里接收到写命令。在普通模式下，梯形逻辑程序不能区分处理过的命令的类型。

应用 Pass Through 模式唯一要考虑的事情就是和普通模式相比，需要编写多一点梯形逻辑程序。

1.4.8 从站模式 – 路由模式

如果主站设备需要定期的读写连接到模块主站端口上的从站设备的数据，那路由模式将是一个强有力的工具。这在一些 SCADA 应用中是一种普遍的需求。在这些 SCADA 系统中，有一些其它的 Modbus 设备连接到主站端口（比如流量计算机），这些设备有很多的数据对于主机非常有用，但本地的 PLC/SLC 确不需要这些设备。

当从站驱动检测接收到命令，而且命令的目标设备和模块路由表中的地址一致（对模块进行设置时一同设置），下列步骤开始执行：



1. 主机发送读或写命令到从站，而且从站地址和 MCM 模块路由表中的地址之一一致。
2. MCM 从站驱动检测到其和路由表的条目一致时，把命令发送到主站驱动
3. 主站驱动以最快的速度执行这条命令（比如，一旦当前所有正在执行的命令结束之后）
4. 目标从站回应这条命令
5. 从站的响应从主站驱动传递到从站驱动

6. 从站驱动把回应返回到主机

1.5 Modbus 寻址

在应用 MCM 模块时，无论作为主站或从站，Modbus 寻址的问题在任何层次上都是非常重要的。本章节初步介绍了 Modbus 寻址，让新的 Modbus 用户熟悉 Modbus 的术语和概念。如需了解 Modbus 的详细内容，可以参考附录中的精简的 Modbus 协议说明。

1.5.1 Modbus 寻址概念

Modbus 寻址体系是早期从 Modicon PLC 的数据表和 I/O 结构。因此，Modbus 协议支持读写 Modicon PLC 的不同数据空间。

到目前为止，最常用的数据空间是使用功能代码 3、6 和 16 的 4xxxx 空间。此空间用来传输 16 位寄存器数值，浮点数数值，和偶位映射数据。使用正式的 Modbus 寻址术语，这个数据空间真正从 40001 地址开始（Modbus 是从 1 开始的，而 MCM 数据表寻址是从 0 开始的）。

读写到不同的数据空间是根据所使用的功能代码决定的。后面的表格显示了 4 种不同类型的数据空间，这些空间的数字范围，和读写到这些数据空间所使用的功能代码。

Register Space			Read Command	Write Command
0XXXX	Coils (Discrete Out)	1 9999	1	5,15
1XXXX	Inputs (Discrete In)	1 9999	2	N/A
3XXXX	Input Registers (Analog In)	1 9999	4	N/A
4XXXX	Holding Registers (Memory Regs)	1	3	6,16

Modbus 功能代码和 Modbus 寻址体系的关系

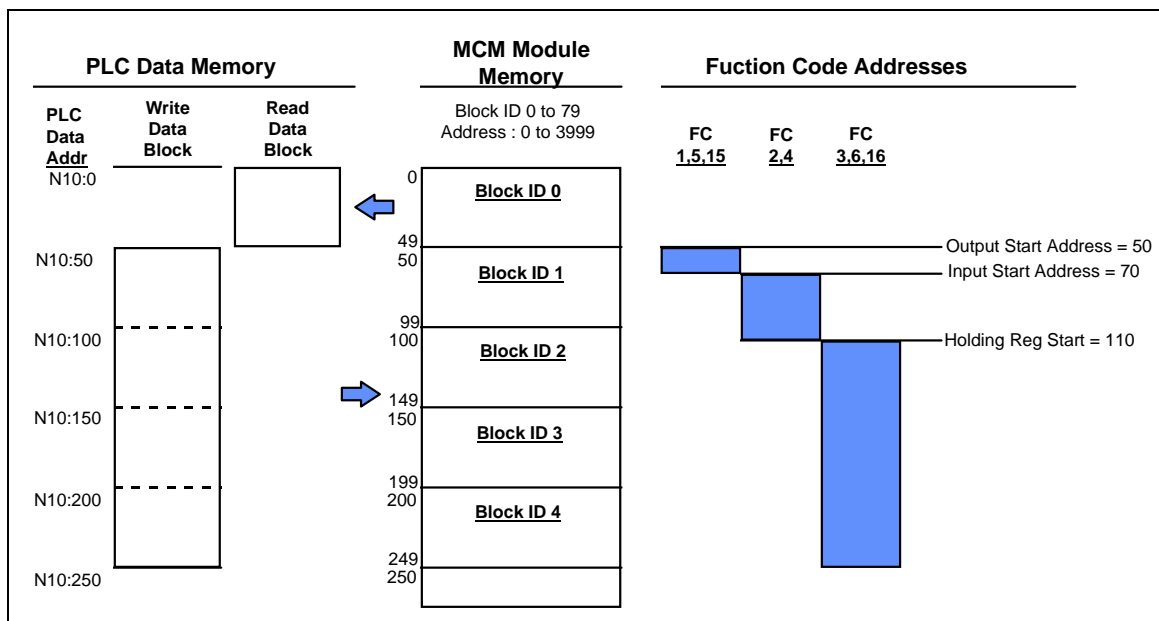
1.5.2 MCM 支持的 Modbus 功能

MCM 模块支持所有数据传输常用的 Modbus 功能代码。下表记录了支持的功能代码和寻址范围（详细内容参考第 6 章）。

功能代码	描述	地址范围
1	Read Output Status	0001 - 9999
2	Read Input Status	10001 - 29999
3	Read Holding Regs	40001 - 49999
4	Read Input Regs	30001 to 39999
5	Force Single Coil	00001 - 9999
6	Preset Single Reg	40001 - 49999
8	Loopback Test	
15	Force Multiple Coils	0001 - 9999
16	Preset Multiple Regs	40001 - 49999

1.5.3 把 Modbus 地址映射到梯形数据地址

梯形逻辑程序和 Modbus 寻址体系的关系受控于应用程序的编制。下面的图表就是一个应用模块时可以设置成的映射关系。



在这个样例中，MCM 设置成 Modbus 从站，响应主机的请求。主机会从模块的寄存器 50 开始读数据。下面表格真实的反映了这种数据排布：

功能代码	模块地址	Modbus 地址
1,5,15	50 到 69	1 到 320
2, 4	70 到 109	10001 到 10640 30001 到 30040
3, 6, 16	110 到 249	40001 到 40140

2 启动 – 一个按部就班的步骤

安装 3100/3150-MCM 模块是很简单的。安装到系统中只需要几个步骤。下面就是实际应用的步骤：

步骤	示例	用户应用
1. 判断机架位置	机架 0 组 2 槽 0	机架：____ 组：____ 槽：____
2. 确认使用的 PLC 数据文件	BT 缓冲器: N7 BT 控制: N7 设置文件: N7 数据文件 : N10	BT 缓冲器: N____ BT 控制: N____ 设置文件: N____ 数据文件 : N____
3. 梯形逻辑	光盘和附录中有样例程序 (多个样例可供选择)	选择最接近您应用的程序，并根据您的需要进行修改
4. 依据机架位置修改梯形逻辑	<u>PLC</u> BTR - Rung 2:0 BTW - Rung 2:1 <u>SLC</u> I:x.0 addresses O:x.0 addresses M0:x addresses M1:x addresses	根据机架的位置需求修改这些指令。在 SLC 中要设置槽位
5. 依据使用的数据文件修改梯形逻辑	模块使用 N7 和 N10 作为数据空间	创建文件，并根据 N7 和 N10 更改注释
6. 把卡安装到机架中	关闭机架电源，安装模块	关闭电源，安装模块
7. 连接通讯电缆到模块的前端	依据应用的需要选择电缆的类型	
9. 系统上电，切换 PLC 到 RUN 状态	监视状态文件和模块前端的 LED	

当硬件已经安装，必要的程序已经下载到处理器后，系统就已经准备好了（假设其它所有系统组成部分已经安全的准备好了）。

3 梯形逻辑概述

处理器和 ProSoft Technology 模块之间的数据传输在 PLC 中使用快传输命令，在 SLC 中使用 M0/M1 数据传输。这些命令每次能传输 64 个物理的寄存器。而逻辑数据的长度取决于数据传输的功能。

下面的章节详细讨论了在 ProSoft Technology 模块和处理器之间传输的不同类型数据所使用的数据结构。在后面的叙述中我们使用“块传输”这个词语来概括描述处理器和 ProSoft Technology 模块之间的数据传输。尽管在 SLC 中不存在真正意义上的块传输功能，但我们使用了一种假块传输功能来确保数据在块层面上的完整性。附录中包含了 PLC 和 SLC 的样例程序。



为了能让 ProSoft Technology 模块发挥作用，PLC/SLC 必须处于运行模式，或处于远程运行模式。如果处于其它模式（故障/编程），模块会停止所有的通讯直到块传输继续开始

3.1 运作概述

上电时，模块将 255 写入 BTR 数据文件的第一个字。这是个信号，表明模块在继续处理前需要接收设置数据。当模块接收到设置，就开始和处理器之间进行数据交换，传输的数据取决于设置的读和写块的个数。当这些都完成后，如果设置了命令块，模块就会传输这些命令块。

3.2 梯形逻辑

对模块编程的方式定义了梯形逻辑的运行。预期的梯形逻辑运行应该像这样：

读行

1. 从模块读数据。在 PLC 中，模块数据被传输到 BTR 缓冲器中。在 SLC 中，模块数据直接由 M1 文件读取
2. 解码出 BTR 块 ID 数字。根据 BTR 块 ID 的数值，复制模块数据到相应的梯形逻辑数据表的正确位置
3. 将 BTR 缓冲器中第一个字中的 BTW 块 ID 数移动到 BTW 缓冲器的第 0 个字。在 SLC 中实际上是把 M1 文件中的第一个字传输到 M0 文件的第 0 个字。如果需要，BTW 块 ID 数要正确处理，以确保模块中的数据不会写重叠（在样例程序中 LIM 指令完成这个功能）
4. 事件触发命令测试和模块设置

写行

1. 解码出 BTW 块 ID 数字，依据其数值移动数据数值，命令列表数值或设置数值到 BTW 缓冲器（SLC 中的 M0 文件）
2. 如果设置传输使能，则清除设置使能位
3. 如果事件触发命令使能，则清除此使能位
4. 执行 BTW 传输。在 PLC 中是使能 BTW 指令。在 SLC 中是置传输完成位（在模块设计时已经为此功能分配了一个输出位）

4 写入模块

本章节详细介绍了从 PLC/SLC 处理器到 MCM 模块的数据传输。这中类型的传输让梯形逻辑发送设置，命令列表和数据到模块。

4.1 块传输到模块

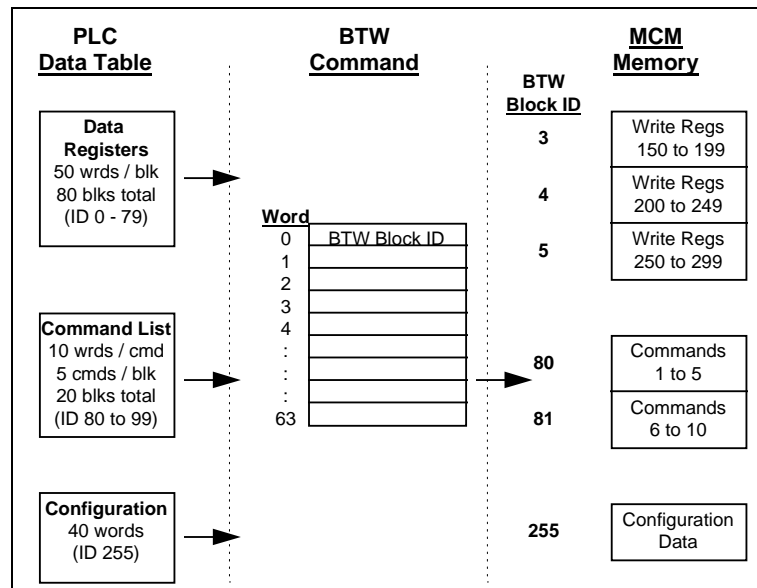
从处理器到模块的数据传输是通过块传输写功能来实现的。传输不同类型的数据需要不同的，但区别不大的数据块结构。但基础结构是：

字	名称	描述										
0	BTW 块 ID	块的标识代码。ProSoft 模块使用这个代码来判断如何处理数据块。有效代码是： <table border="1"> <thead> <tr> <th>BTW 代码</th> <th>描述</th> </tr> </thead> <tbody> <tr> <td>0-79</td> <td>模块数据内存</td> </tr> <tr> <td>80-99</td> <td>命令列表</td> </tr> <tr> <td>100-120</td> <td>事件驱动写</td> </tr> <tr> <td>255</td> <td>模块通讯设置</td> </tr> </tbody> </table>	BTW 代码	描述	0-79	模块数据内存	80-99	命令列表	100-120	事件驱动写	255	模块通讯设置
BTW 代码	描述											
0-79	模块数据内存											
80-99	命令列表											
100-120	事件驱动写											
255	模块通讯设置											



尽管不会使用到数据缓冲器中全部的物理的 64 个字，但 BTW 和 M0 长度必须设置到 64 个字，否则模块的运作会不可预料。

1 到 63	数据	需要写到模块的数据。数据结构取决于块 ID 代码。下面的章节提供不同结构的详细内容。
--------	----	--------------------------------------------



PLC 到 MCM 的数据传输：数据数值和命令列表项“分页”传送到 MCM 模块。数据类型和位置的写入依据 BTW 块 ID 数字。BTW 块 ID 数字受控于 MCM 模块，后面的章节会接续讨论。

4.2 通讯设置 [BTW 块 ID 255]

当模块首次上电时，或参数更改时，都必须设置一次 ProSoft Technology firmware 通讯参数。

上电

上电时，模块进入逻辑循环，等待从处理器接收设置数据。在等待时，模块设置 BTR 缓冲器 (BTW 块 ID) 的第二个字成 255，告知处理器模块在继续工作前需要被设置。模块会连续的执行块传输，直至接收到通讯设置参数。接收完毕后，模块开始执行如果存在的命令列表，或从处理器寻求命令列表。

运行中改变设置

可以在任何时间更改设置表中的数据。模块在“重新设置”过程触发前不会接收任何更改。可以通过很多方式来重新设置，包括：

1. 机架重新上电
2. 按模块上的重启按钮 (仅 3100)
3. 在 BTW 块 ID 位置写 255 (参考样例逻辑何时写位 B3/0)

在此过程中，“CFG”灯会更改状态，形象的显示模块已经收到了设置块。



传送通讯设置参数到模块会强制重启通讯端口，同时降低 DTR 信号 200ms 以重置任何连接的硬件

设置数据块必须以以下的结构从处理器传送到模块：

BTW 缓冲器	数据地址	名称	示例数值
0		BTW 块 ID	255
		端口 1 设置	
1	N[]:0	端口设置字	0 - 主站 1 - 从站
2	N[]:1	端口从站地址	1
3	N[]:2	波特率	5
4	N[]:3	RTS 到 TxD 延迟	0
5	N[]:4	RTS Off 延迟	0
6	N[]:5	响应超时	0
7	N[]:6	字间延迟	0
8	N[]:7	设置参数 #1	0
9	N[]:8	设置参数 #2	0
10	N[]:9	设置参数 #3	0
		端口 2 设置	
11	N[]:10	端口设置字	0 - 主站 1 - 从站
12	N[]:11	端口从站地址	1
13	N[]:12	波特率	5
14	N[]:13	RTS 到 TxD 延迟	0
15	N[]:14	RTS Off 延迟	0
16	N[]:15	响应超时	0
17	N[]:16	字间延迟	0
18	N[]:17	设置参数 #1	0
19	N[]:18	设置参数 #2	0
20	N[]:19	设置参数 #3	0

(Cont'd)

		系统设置	
21	N[]:20	读块个数	3
22	N[]:21	写块个数	1
23	N[]:22	命令块个数	2
24	N[]:23	从站错误指针	100
25	N[]:24	主站错误指针	120
26	N[]:25	BT 延迟	0
27	N[]:26	浮点偏移	0
28	N[]:27	读块 ID 开始	0
29	N[]:28	写块 ID 开始	0
30	N[]:29	空	0
31 - 36	N[]:30 到 N[]:35	路由模式从站 1 到 6	0

端口 1 和 2 设置

数据地址	名称	描述
N[]:0 N[]:10	端口设置字	<p>此寄存器包含一些通讯设置参数，这写参数编码成一个字。 参数如下：</p> <p><u>协议模式</u>：端口的协议模式按这些位来选择：</p> <p>位 210</p> <p>000 Modbus 主站 - RTU 模式 001 Modbus 从站 - RTU 模式 010 Modbus 主站 - ASCII 模式 7 bit 011 Modbus 从站 - ASCII 模式 7 bit 100 Modbus 主站 - ASCII 模式 8 bit 101 Modbus 从站 - ASCII 模式 8 bit</p> <p><u>Pass Through 模式</u>：从站端口运作模式可以通过这个位来选择：</p> <p>位 3</p> <p>0 Pass Through 禁止 1 Pass Through 开启</p> <p><u>Routing 模式</u>：开启从站到主站路由模式：</p> <p>位 4</p> <p>0 路由模式禁止 1 路由模式开启</p> <p><u>停止位</u>：使用的停止位的个数，用以下方式定义：</p> <p>位 13 12</p> <p>0 0 一个停止位 0 1 两个停止位 1 x 无效的端口设置</p> <p><u>奇偶校验</u>：模块使用下列方式来定义奇偶校验模式：</p> <p>位 15 14</p> <p>0 0 无校验 0 1 奇校验 1 0 偶校验 1 1 无效的端口设置</p>

(续)

N[]:1 N[]:11	从站地址	当端口设置成从站模式运行，在此寄存器内输入数值作为 Modbus 从站地址。有效的数值范围是 1 到 247。																		
N[]:2 N[]:12	波特率	<p>端口运作的波特率。可以设置的数值是：</p> <table border="1"> <thead> <tr> <th>数值</th> <th>波特率</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>300 Baud</td> </tr> <tr> <td>1</td> <td>600 Baud</td> </tr> <tr> <td>2</td> <td>1200 Baud</td> </tr> <tr> <td>3</td> <td>2400 Baud</td> </tr> <tr> <td>4</td> <td>4800 Baud</td> </tr> <tr> <td>5</td> <td>9600 Baud</td> </tr> <tr> <td>6</td> <td>19200 Baud</td> </tr> <tr> <td>7</td> <td>38400 Baud</td> </tr> </tbody> </table> <p>模块的两个端口的最高波特率被限制到 19200 或 38400。模块不能被设置成一个端口是 19200，另一个端口是 38400。如果尝试将模块设置成这样，模块会返回一个端口设置错误。</p>	数值	波特率	0	300 Baud	1	600 Baud	2	1200 Baud	3	2400 Baud	4	4800 Baud	5	9600 Baud	6	19200 Baud	7	38400 Baud
数值	波特率																			
0	300 Baud																			
1	600 Baud																			
2	1200 Baud																			
3	2400 Baud																			
4	4800 Baud																			
5	9600 Baud																			
6	19200 Baud																			
7	38400 Baud																			
N[]:3 N[]:13	RTS 至 TXD 延迟	<p>这个数值表示了以 1ms 为增量的时间。此时间是触发 RTS 到真正传输数据之间的间隔。延迟时间如果大于 CTS 的硬件延迟时间，则会覆盖 CTS，直至超时时间到。</p> <p>这个参数用于和 modem 类的设备连接，此时必须在数据传输之前衰减信号线的干扰，或是减慢数据传输。 有效数值范围是 0 到 65535 (0xffff)。</p>																		
N[]:4 N[]:14	RTS Off 延迟	<p>这个数值表示了以 1ms 为增量的时间。此时间是在最后一个字符传输后，RTS 信号降低前之间的间隔。模块自动插入一个字符长的下降延迟，假设 RTS 不会下降直到最后一个字符完整的被发出。除非在非常条件下，这个数值通常设置成 0。</p> <p>有效数值范围是 0 到 65535 (0xffff)。</p>																		
N[]:5 N[]:15	讯息响应超时	<p>这个寄存器表示了以 1ms 为增量的讯息响应超时时间间隔。这个时间是端口设置成主站时，在没有收到从站响应后重新发送命令时等待的时间。这个数值设置取决于从站预期响应的的时间。</p> <p>允许的数值范围是 0 到 65535(0xffff)。如果输入数值 0，模块会设置成默认时间 1 秒 (1000 ms)。</p>																		
N[]:6 N[]:16	字符间超时	<p>这个寄存器用于那些讯息字符的结束超时延迟必须超过通常的 3.5 个字符长度的情况。此数值表示毫秒级别的“无传输”间隔，在接收一个讯息前会计数到这个时间的长度。这个参数适用在卫星或包电台应用中，在这些应用中单个数据传输会被分成两个包传送。增大此数值超过系统包处理时间会消除超时错误。</p> <p>有效数值范围是 0 到 65535 (0xffff)</p>																		

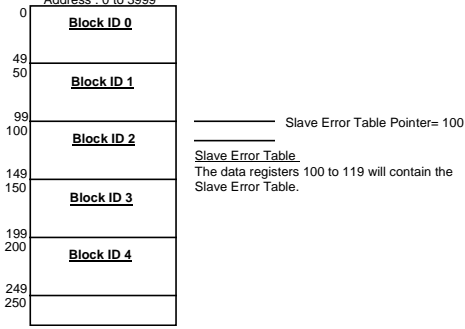
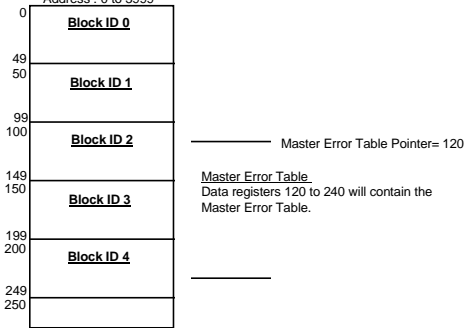
(续)

N[]:7 N[]:17	设置参数 #1 - 主站模式: 未使用 - 从站模式: 输入内存起始地址	<u>Modbus 从站模式:</u> 这个数值定义在 4000 个字数据空间中的偏移地址, MCM 从站端口使用这个地址来响应功能代码 2 和 4 的命令。例如, 如果输入镜像地址要从第 150 个字开始, 就输入 150。以地址零开始的功能代码 2 或 4 的命令 (10001 或 30001) 会从第 150 个字开始读。 有效数值范围是 0 到 3999。
N[]:8 N[]:18	设置参数 #2 - 主站模式: 未使用 - 从站模式: 输出内存起始地址	<u>Modbus 从站模式</u> 这个数值定义在 4000 个字数据空间中的偏移地址, MCM 从站端口使用这个地址来响应功能代码 1, 5 或 15 的命令。例如, 如果输出镜像地址要从第 100 个字开始, 就输入 100。以地址 0 (1) 开始的功能代码 1 的命令会从第 100 个地址开始。 有效数值范围是 0 到 3999。
N[]:9 N[]:19	设置参数 #3 - 主站模式: 未使用 - 从站模式: 保持寄存器起始地址	<u>Modbus 从站模式</u> 这个数值定义在 4000 个字数据空间中的偏移地址, MCM 从站端口使用这个地址来响应功能代码 3, 6 或 16 的命令。例如, 如果定义地址 40001 为模块的第 100 个字, 就输入 100。以地址 0 (40001) 开始的功能代码 3 的命令会从第 100 个字开始读。 有效数值范围是 0 到 3999。

系统设置

数据地址	名称	描述
N[]:20	读数据块数目	<p>这个数值表示 50 字长的数据块的个数, 这些数据块从 MCM 模块传送到处理器。模块返回的块从块 0 开始到这个数目。最大块数目是 80。</p> <p>例如, 数值 5 会返回 BTR 块 ID 数据块 0, 1, 2, 3 和 4 或模块寄存器 0 到 249。</p> <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 10px auto;">如果输入数值超过 80, 会引发系统设置错误</div>
N[]:21	写数据块数目	<p>这个数值表示 50 字长的数据块的个数, 这些数据块从处理器传送到模块。模块会使用这个数值来返回 BTW 块 ID 数目到处理器。梯形逻辑使用这个数值来判断哪些数据通过块传输写来移动到 MCM。最大块数目是 80。</p> <p>例如, 如果输入数值 5, MCM 会返回 BTW 块 ID 号 0, 1, 2, 3 和 4 到梯形逻辑 (参阅 4.2)。</p> <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 10px auto;">如果输入数值超过 80, 会引发系统设置错误</div>

(续)

<p>N[]:22</p>	<p>命令块数目</p>	<p>这个数值表示 50 字长的命令块的个数，这些数据块从处理器传送到 MCM 模块。模块如果不设置成主站端口，这个数值就是 0。参阅章节 4.1.3 详细了解需要的命令块。最大块数目是 20。</p> <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 10px auto;"> <p>如果输入数值超过 20，会引发系统设置错误</p> </div>
<p>N[]:23</p>	<p>从站错误块指针</p>	<p>这个数值表示模块数据表中 Modbus 从站错误数据块存放的起始地址。从站错误表是一个 20 个字的块，包含从站端口状态和一些通讯计数。错误数据可以被放置在模块数据空间（0 到 3999）内的任何地方。这样，错误表中的内容可以作为常规寄存器数据来读取。</p> <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 10px auto;"> <p>如果输入数值超过 3980，会引发系统设置错误</p> </div> <p>MCM Module Memory Block ID 0 to 79 Address : 0 to 3999</p>  <p>Slave Error Table Pointer= 100 Slave Error Table The data registers 100 to 119 will contain the Slave Error Table.</p>
<p>N[]:24</p>	<p>主站错误块指针</p>	<p>这个数值表示模块数据寄存器表中主站错误数据块存放的起始地址。错误块（长度为 120 个字）能放置在模块数据空间（0 到 3999）的任何位置。这样，错误表中的内容可以作为常规寄存器数据来读取。</p> <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 10px auto;"> <p>如果输入数值超过 3980，会引发系统设置错误</p> </div> <p>MCM Module Memory Block ID 0 to 79 Address : 0 to 3999</p>  <p>Master Error Table Pointer= 120 Master Error Table Data registers 120 to 240 will contain the Master Error Table.</p>

(续)

N[]:25	块传输延迟数目	这是个经验数值，模块使用这个数值来平衡模块进行块传输耗用的时间量和模块进行端口通讯耗用的时间量。输入的数值在模块内作为循环计数器，每次循环计数减少 1。当计数等于块传输延迟数目，块传输序列就启动。数值范围是 0 到 255。 例：在远程机架使用模块的主站模式应用中，可以输入 75 到 150 来改善命令执行的频率。这个数值必须依据经验判断。
N[]:26	浮点数偏移	当寻址寄存器 > 7000（通常称作 Enron 版 Modbus 协议），模块从站端口驱动使用此数值来支持浮点寄存器的读和写寻址。模块按照下面的方式使用偏移数值： MCM 寄存器地址 = 浮点数偏移 + (寄存器地址 - 7000) * 2 浮点数指针偏移在 Pass Through 模式寻址中不使用。在 Pass Through 模式中，传递到梯形逻辑的地址按下面的方式计算： 地址 = 寄存器地址 - 7000
N[]:27	读块 ID 开始	这个数值决定返回到模块的起始 BTR 块 ID 号。例如，如果梯形逻辑需要从模块接收块 2 到 5，参数应设置成“2”，读数据数目应设置成“4”。有效数值范围从 0 到 79。
N[]:28	写块 ID 开始	这个数值决定模块返回到梯形逻辑的起始 BTW 块 ID 号。例如，如果梯形逻辑要写块 4 到块 5 到模块，这个参数应设置成“4”，写块个数应设置成“2”。有效数值范围从 0 到 79。
N[]:30 to N[]:35	路由模式从站地址 #1 - #6	当 MCM 模块设置成路由模式时，需要提供这 6 个地址。在这种模式下，从站端口接收到命令中如果包含的地址符合路由地址，那么这些命令会被再从主站端口发送出去。从站的响应会通过从站端口路由到主机。

4.3 写入模块数据内存 [BTW 块 ID 代码 0-79]

写入 MCM 寄存器数据空间是通过使用 BTW 块 ID 代码 0 到 79 的块传输写来实现。每个块是 50 个字。

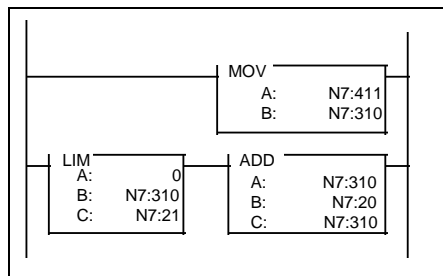


实际操作中要注意内存的排布，以避免 MCM 读写的数据和同梯形逻辑交换的数据发生重叠。Modbus 数据 **不能** 移动到 50 字的块，而这个块同时也被处理器刷新。附录中的梯形逻辑也指出这个要点。

4.3.1 写数据到模块梯形逻辑

移动数据到模块所需要的梯形逻辑完全是一系列的 EQU - COP 分支的简单组合，或者使用间接寻址的方式编写。在我们所有的样例梯形逻辑中，数据传输到模块的方式都是通过两个步骤实现的：

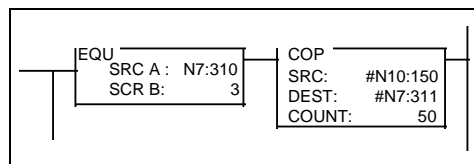
步骤 1: 在 BTR 过程中, 模块会在 BTR 数据缓冲中的第二个字中“喂”给梯形逻辑一个 BTW 块 ID 号。梯形逻辑接收这个数值, 根据需要对其整理, 然后把数值一定到真正的 BTW 块 ID 位置。梯形逻辑程序如下:



设置 BTW 块 ID 号

在 BTR 的最底行 (Rung 0), 这行程序把从模块接收到的 BTW 块 ID 号, 通过读块 ID 数目 (N7:20) 进行偏移, 这就确保了 PLC 数据不会覆盖正在从模块返回到 PLC 的数据。细节内容参阅附录中的逻辑程序。

步骤 2: 在处理 BTW 过程中, 梯形逻辑会对 BTW 块 ID 寄存器中的数值进行检查, 并根据这个数值, 从数据表复制数据到 BTW 块传输缓冲。这个过程需要每个 BTW 块 ID 都能够在一个逻辑分支中被处理。下面是所需程序的例子:



检查 BTW 块 ID 并移动数据到 BTW 缓冲

这个分支, 位于 BTW 行 (rung 1), 是所需的移动每个数据块到模块的程序。细节内容参阅附录中的逻辑程序。

4.3.2 块传输数据结构

下表的的就是写数据到模块的块传输缓冲:

字	名称	描述												
0	BTW 块 ID	<p>块识别号使 MCM 模块能解码出将要写入的是模块 4000 字数据空间的哪个“50 字页”。被写的数据空间通过用 50 乘 BTW 块 ID 号来决定。结果是“页”的第一个字。例如:</p> <table border="0"> <tr> <td>BTW</td> <td></td> </tr> <tr> <td>块 ID</td> <td>数据空间</td> </tr> <tr> <td>0</td> <td>0 到 49</td> </tr> <tr> <td>1</td> <td>50 到 99</td> </tr> <tr> <td>10</td> <td>500 到 549</td> </tr> <tr> <td>20</td> <td>1000 到 1049</td> </tr> </table> <p>通过分页传送不同的数据块到模块，处理器就能够控制模块的数据内存内容。</p>	BTW		块 ID	数据空间	0	0 到 49	1	50 到 99	10	500 到 549	20	1000 到 1049
BTW														
块 ID	数据空间													
0	0 到 49													
1	50 到 99													
10	500 到 549													
20	1000 到 1049													
1 to 50	数据	要写到模块的数据。												

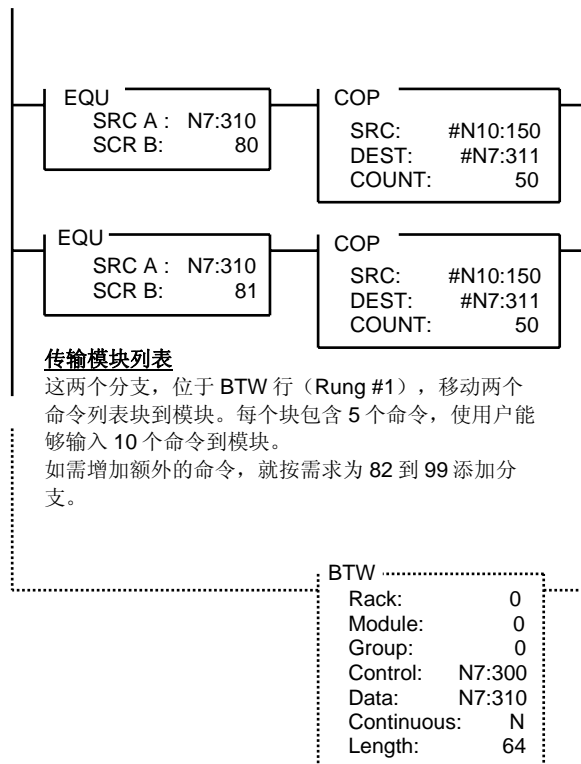
4.4 命令列表设置 - 主站模式 [BTW 块 ID 代码 80-99]

MCM Modbus 主站端口建立通讯，根据用户设置的命令列表的数据执行各种通讯功能。命令列表包含 100 个独立设置命令数据块（每个命令保留 10 个字），并被两个端口所共享（当模块设置成两个主站端口）。

4.4.1 命令列表梯形逻辑

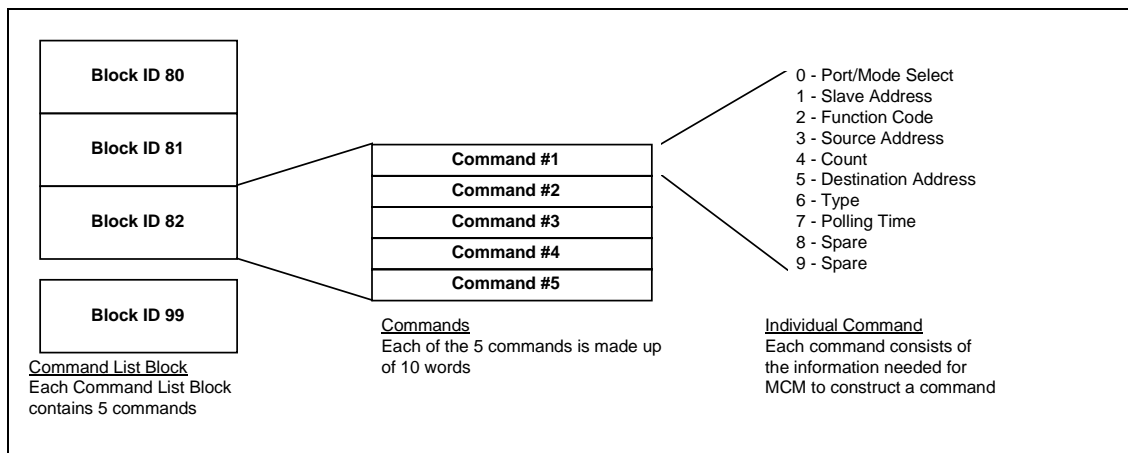
这个列表，在处理器数据表中输入，使用 BTW 块 ID 代码 80-99 传输到模块的内存，每个代码表示 50 个字的块，或 5 个命令。

移动命令到模块的程序示例如下:



4.4.2 命令列表结构

包含命令列表的块结构如下表所示：



设置 Modbus 命令参阅第 6 章

名称	描述																
端口/模式选择	<p>端口/模式选择参数让程序选择使用哪个 MCM 模块端口来执行命令，和命令连续执行或检测跳变信号（条件），或直接受控于梯形逻辑控制（控制）。有效数值是：</p> <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>端口/模式</th> <th>描述</th> </tr> </thead> <tbody> <tr><td>0</td><td>禁止命令</td></tr> <tr><td>1</td><td>端口 1 连续命令</td></tr> <tr><td>2</td><td>端口 2 连续命令</td></tr> <tr><td>5</td><td>端口 1 条件命令</td></tr> <tr><td>6</td><td>端口 2 条件命令</td></tr> <tr><td>9</td><td>端口 1 控制命令</td></tr> <tr><td>10</td><td>端口 2 控制命令</td></tr> </tbody> </table> <p>连续 vs. 条件：功能代码 5, 6, 15, 和 16</p> <p>在命令列表中设置写命令时，MCM 主站驱动可以支持两种类型的数据写命令：连续和条件。二者的区别是：</p> <p>连续：按此种方式输入的命令在每次扫描模块命令列表时都执行一次。</p> <p>条件：条件命令仅当检测到要写的数据块发生变化才执行。每次扫描命令列表时，模块会把要写的数据和最后一次写的数据进行比较。如果检测到任何数值，位或字发生变化，模块就会写出这些受控的整个数据块。</p> <p>控制命令模式</p> <p>在控制命令模式里，命令仅当命令使能位（参见章节 4.5）从 0 变换到 1 时才执行。每次转变时命令就执行一次（例如，模块执行一些单次逻辑以保证命令只执行一次）。说明的是，模块内的每个单次执行，命令使能位都必须从 1 回到 0。</p>	端口/模式	描述	0	禁止命令	1	端口 1 连续命令	2	端口 2 连续命令	5	端口 1 条件命令	6	端口 2 条件命令	9	端口 1 控制命令	10	端口 2 控制命令
端口/模式	描述																
0	禁止命令																
1	端口 1 连续命令																
2	端口 2 连续命令																
5	端口 1 条件命令																
6	端口 2 条件命令																
9	端口 1 控制命令																
10	端口 2 控制命令																

(续)

从站地址	从站地址表示命令所指向的从站的 Modbus 从站地址。地址应以十进制形式输入。																		
功能代码	<p>表格中输入的功能代码告知 MCM 模块要执行什么命令。各种选择参阅第 6 章，大致如下：</p> <table border="0"> <thead> <tr> <th><u>功能代码</u></th> <th><u>描述</u></th> </tr> </thead> <tbody> <tr> <td>1</td> <td>读输出状态</td> </tr> <tr> <td>2</td> <td>读输入状态</td> </tr> <tr> <td>3</td> <td>读多数据寄存器</td> </tr> <tr> <td>4</td> <td>读输入寄存器</td> </tr> <tr> <td>5</td> <td>强制单 Coil (置位/清位)</td> </tr> <tr> <td>6</td> <td>预制 (写) 单数据寄存器</td> </tr> <tr> <td>15</td> <td>多 Coil 置位/清位</td> </tr> <tr> <td>16</td> <td>预制 (写) 多数据寄存器</td> </tr> </tbody> </table>	<u>功能代码</u>	<u>描述</u>	1	读输出状态	2	读输入状态	3	读多数据寄存器	4	读输入寄存器	5	强制单 Coil (置位/清位)	6	预制 (写) 单数据寄存器	15	多 Coil 置位/清位	16	预制 (写) 多数据寄存器
<u>功能代码</u>	<u>描述</u>																		
1	读输出状态																		
2	读输入状态																		
3	读多数据寄存器																		
4	读输入寄存器																		
5	强制单 Coil (置位/清位)																		
6	预制 (写) 单数据寄存器																		
15	多 Coil 置位/清位																		
16	预制 (写) 多数据寄存器																		
源地址	<p>此数值表示读和写命令获得数据的寄存器或位地址。二者的区别如下：</p> <ul style="list-style-type: none"> - 发布一个读命令时，源寄存器地址是命令在从站中开始获取数据的寄存器地址 - 发布一个写命令时，源寄存器地址是命令要写到从站的，模块中的开始地址。 																		
数目	Modbus 命令读和写的字或位的数目。不同命令所需要的字和位长度请参阅第 6 章。																		
目标地址	<p>此数值表示读和写命令目标地址的寄存器或位地址。二者的区别如下：</p> <ul style="list-style-type: none"> - 发布一个读命令时，目标地址是命令开始放置从从站获得的数据的寄存器地址。 - 发布一个写命令时，目标地址是命令开始在从站放置数据的寄存器地址。 																		
类型	<p>仅对功能代码 3 (多寄存器读) 相关。这个参数告知模块在接收这种类型的命令时对数据进行交换处理。</p> <p>这对于从一些仪器 (某些仪器把自身的浮点数数据存储成和处理器方向相反的字) 读取浮点数 (每个数值两个字) 非常有用。在这些情况下，交换后的数据可以直接使用梯形逻辑 COP 命令复制整数文件到浮点数文件。可选项是：</p> <table border="0"> <thead> <tr> <th><u>类型</u></th> <th><u>描述</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>默认数值。不执行交换处理。</td> </tr> <tr> <td>1</td> <td>在命令中把从从站读取的数据每对字进行交换</td> </tr> <tr> <td>2</td> <td>交换接收的每对字，然后再交换每个字内的字节</td> </tr> <tr> <td>3</td> <td>交换每个字内吊顶字节 (不交换字)</td> </tr> </tbody> </table> <p>如果使用功能 6 或 16，而且目标寄存器大于 47000，把这个参数设置 1 可以禁止 Enron Extension。</p>	<u>类型</u>	<u>描述</u>	0	默认数值。不执行交换处理。	1	在命令中把从从站读取的数据每对字进行交换	2	交换接收的每对字，然后再交换每个字内的字节	3	交换每个字内吊顶字节 (不交换字)								
<u>类型</u>	<u>描述</u>																		
0	默认数值。不执行交换处理。																		
1	在命令中把从从站读取的数据每对字进行交换																		
2	交换接收的每对字，然后再交换每个字内的字节																		
3	交换每个字内吊顶字节 (不交换字)																		

预置轮询时间	预置轮询时间数值能让每个命令都有一个可设置的执行频率。在模块内，每个命令都有一个维持计时器。每秒钟，计时器减一，直到计时器计时到0。当计时器计时到0时，命令就允许执行，同时计时器复位到预置轮询时间数值。轮询计时器的分辨率是1秒。有效数值是0到65535 (0xffff)。
--------	------------------------------------------------------------------------------------------------------------------------------------------

4.4.3 修改命令列表

输入命令列表其实就是向 PLC 数据表中输入正确的数值。使用梯形逻辑编程软件输入需要的数值来设置一个或更多的有效命令。



提示

当首次设置命令列表时我们建议您从一个命令开始。如果其它一切都 OK (例如梯形逻辑, 电缆已链接, 等等) 模块就会开始传输。一旦模块开始传输, 模块就会尝试可从站通讯, 然后在数据其它需要的命令。

下面是一个命令列表的例子。注意, 命令按行输入, 而且在理解列的定义后, 查看命令列表就会非常容易。

	0	1	2	3	4	5	6	7
	端口	从站	功能	源		目标		轮询
	号	地址	代码	地址	个数	地址	类型	时间
N7:50	1	3	1	0	10	100	0	0
N7:60	1	3	4	50	20	100	0	0
N7:70	2	2	16	200	10	137	0	6
N7:80	6	2	16	210	10	150	0	0

命令列表例子

在下面的表格显示的是多讯息设置数据块的样例 (为了显示清楚, 列 8-9 没有显示)。

4.5 命令控制模式 - 主站模式

在一些特殊运作条件下, 需要梯形逻辑能紧密的合作和控制命令列表中命令执行。为适应这种需求, MCM 模块支持一种称作命令控制模式的功能。

当设置成命令控制模式, 梯形逻辑在每个命令列表输入的基础上提供命令使能控制。此外, 当和命令完成位 (参阅章节 5.3) 一同使用时, 梯形逻辑能按照需求有效的控制单次执行每条命令。

4.5.1 BTW 块结构

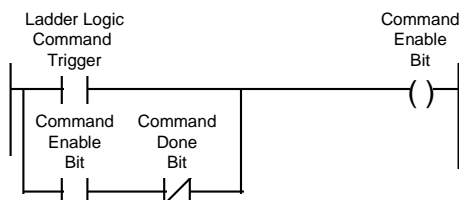
使能位写入到模块的 BTW 块传输结构如下表所示:

字	名称	描述														
0	BTW 块 ID	在每次 BTW 传输中, 命令使能位被移动到模块内。因此, 所有有效的 BTW 块 ID 号都能在这里使用														
1-50	数据	模块数据和命令列表, 如前所述														
51-56	命令使能位	这些寄存器包含了命令列表中每个命令的命令使能位, 共前 96 个命令。使能位根据它们在命令列表中相关联的位置被映射到这些字里: <table style="margin-left: 40px; border: none;"> <tr> <td style="text-align: center;">字</td> <td style="text-align: center;">命令</td> </tr> <tr> <td style="text-align: center;">51</td> <td style="text-align: center;">1 to 16</td> </tr> <tr> <td style="text-align: center;">52</td> <td style="text-align: center;">17 to 32</td> </tr> <tr> <td style="text-align: center;">53</td> <td style="text-align: center;">33 to 48</td> </tr> <tr> <td style="text-align: center;">54</td> <td style="text-align: center;">49 to 64</td> </tr> <tr> <td style="text-align: center;">55</td> <td style="text-align: center;">65 to 80</td> </tr> <tr> <td style="text-align: center;">56</td> <td style="text-align: center;">81 to 96</td> </tr> </table> 例子: 字 51 位 0 时第一个命令的使能位	字	命令	51	1 to 16	52	17 to 32	53	33 to 48	54	49 to 64	55	65 to 80	56	81 to 96
字	命令															
51	1 to 16															
52	17 to 32															
53	33 to 48															
54	49 to 64															
55	65 to 80															
56	81 to 96															

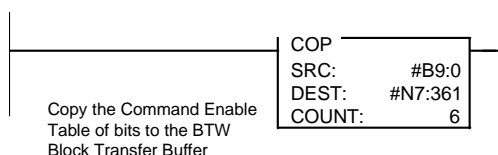
4.5.2 控制命令

当命令设置成命令控制模式，而且到模块检测到命令使能位状态从 0 变化到 1，模块会尝试执行这条命令（会尝试执行命令三次）。如果命令成功发送，命令完成位会置位。如果命令发送过程中出错，命令错误位会置位。

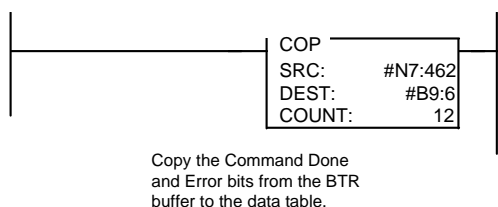
下面是一个控制命令的梯形逻辑例子：



最简单的方法是保留一个命令使能位的二进制表，再复制到每次传输的 BTW 缓冲。下面的分支程序可以添加到 BTW 行（传输数据到模块）：



然后，命令完成和错误位就可以复制到同一个二进制文件，并在梯形逻辑中引用。下面的指令可以添加到 BTR 行（从模块读取数据）：



4.5.3 命令列表示例

命令可以通过设置命令使能来控制

	0	1	2	3	4	5	6	7
	端口	从站	功能	源		目标		轮询
	号	地址	代码	地址	个数	地址	类型	时间
N7:50	9	3	1	0	10	100	0	0
N7:60	10	3	4	50	20	100	0	0

命令列表示例

示例中，N7:50 中的命令设置成端口 1 的控制命令模式，而 N7:60 命令是端口 2 的。

4.6 事件触发命令 - 主站模式 [BTW 块 ID 代码 100 到 119]

除了可以在命令列表中设置连续使能命令外，MCM 模块也支持事件触发命令。这些事件命令可以用作有条件的读/写数据到从站。样例应用可能包括在从站中设置时间，复位一批计数器，等。

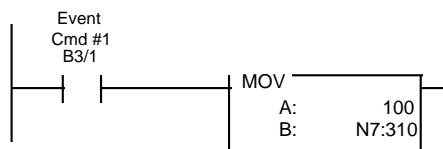


提示

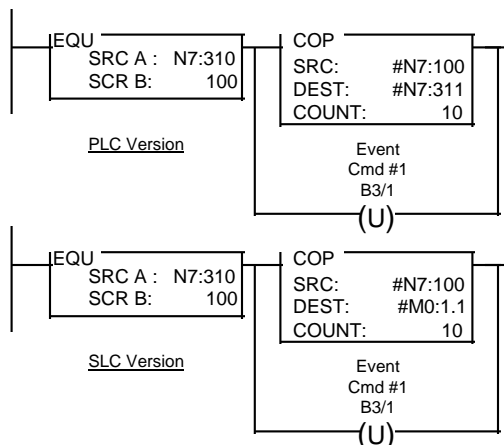
使用事件触发写命令（功能代码 5, 6, 15, 16）的好处之一是能够保证写到从站的数据内容和命令的执行相一致。注意这不一定是执行命令列表外命令的情况。

4.6.1 梯形逻辑

执行一个事件触发命令就是把数据传输到 BTW 缓冲，块 ID 号是在 100 到 119 之间。传输的数据块，下面将会详细讨论，包含模块用于编码一个有效命令的数据。支持这种功能的梯形逻辑就如下所示：



This branch is added to the Read rung, just above the MOV 255 to N7:310 branch. The B3/1 bit, selected here for example purposes only, is one-shot set in the ladder logic



This branch is added to the BTW rung, and serves to copy the Event Initiated Command block structure to the module and then Unlatches the command enable bit which was set in the ladder program.

4.6.2 BTW 块结构



设置 Modbus 命令的详细内容参阅第 6 章

包含事件触发命令的块结构如下表所示：

BTW 字	数据偏移	名称	描述
0		BTW 块 ID	模块使用 BTW 块 ID 来判断块传输缓冲包含的是一个事件触发命令。有效的数值范围是 100 到 119。这个数值定义了主站错误表中的相对位置。

(Cont'd)

BTW 字	数据偏移	名称	描述																		
1	N[:0]	端口号	此参数让用户选择在哪个端口上执行命令的操作。输入数值为： <table border="0"> <tr> <td><u>端口/模式</u></td> <td><u>描述</u></td> </tr> <tr> <td>1</td> <td>Port 1</td> </tr> <tr> <td>2</td> <td>Port 2</td> </tr> </table>	<u>端口/模式</u>	<u>描述</u>	1	Port 1	2	Port 2												
<u>端口/模式</u>	<u>描述</u>																				
1	Port 1																				
2	Port 2																				
2	N[:1]	从站地址	此参数表示命令所指向的 Modbus 从站地址。地址数值要以十进制方式输入。																		
3	N[:2]	功能代码	此参数告知 MCM 模块执行何种命令。可供选择的命令有很多，具体内容参阅第 6 章，下面作个概述。 <table border="0"> <tr> <td><u>功能代码</u></td> <td><u>描述</u></td> </tr> <tr> <td>1</td> <td>Read Output Status</td> </tr> <tr> <td>2</td> <td>Read Input Status</td> </tr> <tr> <td>3</td> <td>Read Multiple Data Registers</td> </tr> <tr> <td>4</td> <td>Read Input Registers</td> </tr> <tr> <td>5</td> <td>Force Single Coil (Latch/Unlatch)</td> </tr> <tr> <td>6</td> <td>Preset (Write) Single Data Register</td> </tr> <tr> <td>15</td> <td>Multiple Coil Latch/Unlatch</td> </tr> <tr> <td>16</td> <td>Preset (Write) Multiple Data Register</td> </tr> </table>	<u>功能代码</u>	<u>描述</u>	1	Read Output Status	2	Read Input Status	3	Read Multiple Data Registers	4	Read Input Registers	5	Force Single Coil (Latch/Unlatch)	6	Preset (Write) Single Data Register	15	Multiple Coil Latch/Unlatch	16	Preset (Write) Multiple Data Register
<u>功能代码</u>	<u>描述</u>																				
1	Read Output Status																				
2	Read Input Status																				
3	Read Multiple Data Registers																				
4	Read Input Registers																				
5	Force Single Coil (Latch/Unlatch)																				
6	Preset (Write) Single Data Register																				
15	Multiple Coil Latch/Unlatch																				
16	Preset (Write) Multiple Data Register																				
4	N[:3]	源地址	对于读命令，此参数表示数据源的寄存器或位地址。 <ul style="list-style-type: none"> 当执行读命令时，源地址是指命令开始读取数据的寄存器地址 当执行一个事件写命令时，这个数值将没有任何意义																		
5	N[:4]	个数	Modbus 命令读或写的寄存器或位的个数。参阅第 5 章有关不同命令中字节和位的长度确定。																		
6	N[:5]	目标地址	对于读和写命令，此参数表示数据要写入的寄存器或位地址。两者的区别如下所述： <ul style="list-style-type: none"> 执行读命令时，目标地址是指命令把从站那里读回的数据放置在模块内的起始地址 执行写命令时，目标地址是指命令在从站内开始放置所写数据的起始地址。 																		
7	N[:6]	类型	这个参数仅和功能代码 3 相关（读多个寄存器）。此参数告知模块对命令接收到的数据执行字交换。参阅章节 4.4.2 对于这个参数的完整叙述																		
8-51	N[:7]	写数据	依据每个功能代码的选择，这些寄存器包含将要发送到从站的写数据。																		

	<u>0</u> 端口号	<u>1</u> 从站地址	<u>2</u> 功能代码	<u>3</u> 目标地址	<u>4</u> 个数	<u>5</u> 目标地址	<u>6</u> 类型	<u>7</u> 数据	<u>8</u> 数据	<u>9</u> 数据
N7:100	1	3	1	0	10	100	0	0	0	0
N7:110	1	3	6	0	1	1	0	1267		

事件触发写命令例子

第一个命令发送 FC1 到从站地址 3，从位 0 读 10 个位放置到模块的寄存器 100 内。第二条命令写数值 1267 到从站的寄存器 1。

5 读取模块

本章节详细叙述了从 PLC/SLC 到 MCM 模块的数据传输。这种传输使梯形逻辑能够发送设置，命令列表和数据到模块。

5.1 从模块传输数据 [BTR 块 ID 0 到 79]

当主站端口驱动向从站读取数据或当主机写至从站端口驱动，结果数据被放置到 ProSoft 模块的数据空间内（地址 0 到 3999）。正如上面的讨论，这个数据空间同样也可以被 PLC/SLC 来写数据。

处理器通过使用 Block Transfer Read 功能来读取 ProSoft Technology 模块的数据。下面的章节详细论述如何处理这些读数据。



经管数据缓冲中的全部 64 个物理字不会全部使用，但 BTR 和 M1 的长度必须设置为 64 个字，否则模块的运行将会不可预料。

5.1.1 读数据块结构

BTR 缓冲定义如下：

字	名称	描述										
0	BTR 块 ID	<p>梯形逻辑依据此数值来判断 BTR 缓冲中的数据内容。在使用一些条件判断的梯形逻辑后，模块数据被放置在 PLC/SLC 的数据表中。</p> <div style="text-align: center;"> </div> <p>BTR 块 ID 号码和寄存器表之间的关系可以通过下面这个等式来表达： 起始寄存器地址 = 块 ID 号码 * 50</p> <p>有效数值是 0 到 79。</p>										
1	BTW 块 ID	<p>模块返回此数值给处理器，用作移动寄存器数据和命令列表块到模块。模块根据在 Block 255 中参数 21 和 22 的设置计算出 BTW 块 ID 号码。这个数值只是作意见参考和降低梯形逻辑编程需要。如果需要开发出一套不与此不同的传输系列，则可以简单的通过梯形逻辑来实现。</p> <p>有效数值是：</p> <table border="0" style="width: 100%;"> <tr> <td style="text-align: center;">BTW 代码</td> <td style="text-align: center;">描述</td> </tr> <tr> <td style="text-align: center;">0-79</td> <td style="text-align: center;">模块数据</td> </tr> <tr> <td style="text-align: center;">80-99</td> <td style="text-align: center;">命令列表</td> </tr> <tr> <td style="text-align: center;">100-119</td> <td style="text-align: center;">事件触发命令</td> </tr> <tr> <td style="text-align: center;">255</td> <td style="text-align: center;">模块设置</td> </tr> </table>	BTW 代码	描述	0-79	模块数据	80-99	命令列表	100-119	事件触发命令	255	模块设置
BTW 代码	描述											
0-79	模块数据											
80-99	命令列表											
100-119	事件触发命令											
255	模块设置											

(续)

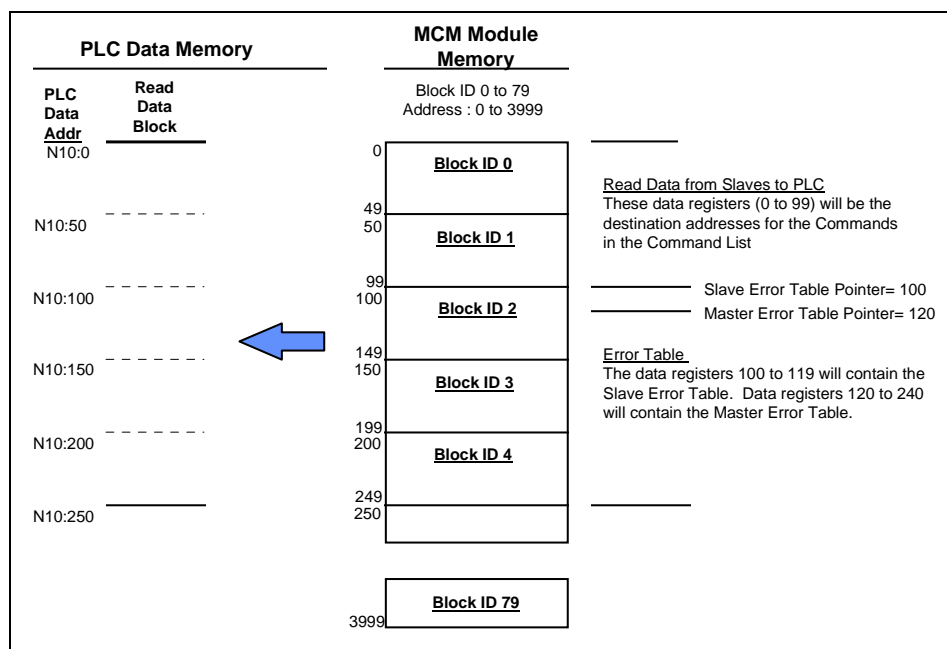
2 to 51	数据	模块寄存器数据空间 (0—3999) 的内容。此数据包含来自于从站的数据，来自于处理器的数据和从站、主站错误表。这些数值都是 16 位的寄存器，必须放置在整形文件中。注意是用户的应用程序来控制数据寄存器的放置和使用。
52 to 63	命令完成和错误位	参见章节 5.3

5.1.2 从模块移动数据到处理器

自从站设备读取的数据（主站驱动），或自主机写入的数据（从站驱动）被放置在模块的 4000 个字的寄存器表中。这个表格的寻址地址是从 0 到 3999。

为了克服 BTR 指令 64 个物理字的限制，从模块到梯形逻辑的数据传输被设计成分页式。分页传输的原理在前文中也曾简单提及过，但重要的是理解页号（BTR 块 ID 号）和模块内寄存器地址之间的关系。

下面的图标就是一个示例。需要指出的是在系统设置“读块个数”（Read Block Cnt）中设置的数值决定了模块返回到梯形逻辑的块个数。在这个示例中，我们假设读块个数（Read Block Count）的数值值是 5。

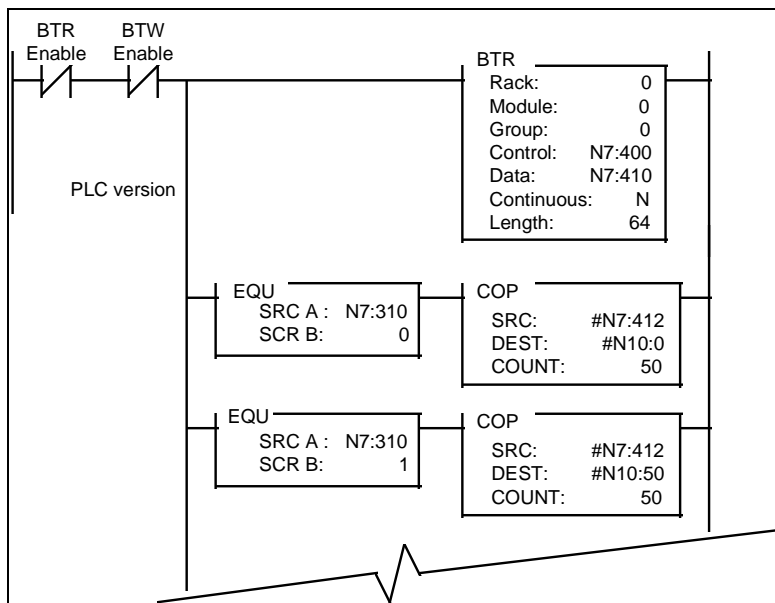


来自 MCM 模块的读数据块

注意在上图中的读数据块个数（Read Block Count）是 5，因此从模块返回寄存器 0 到 249。此设置数值可以根据具体的应用调整。

5.1.3 读模块数据的梯形逻辑

需要编写一些梯形逻辑来检查 BTR 缓冲，解码中间的一些字，然后采取行动。下面就是这些梯形逻辑的示例：



从模块传输数据的样例逻辑

这些逻辑显示了如何从模块向 PLC 数据表移动数据。

5.1.4 从站错误代码表

MCM 模块会监视所有从站端口命令的状态。这些状态数据以从站错误代码表（Slave Error Code Table）的方式传送到处理器。



上电后，和模块每次接收 255 设置数据块时，从站错误表会初始化为零。

从站错误代码表是一个 20 个字的块。错误代码表的位置由设置块中从站错误代码表指针（Slave Error Table Pointer）来决定。数据块的机构如下表所示：

端口 1 状态代码

字	样例地址	名称	描述
0	N10:100	端口当前状态	此数值表示端口当前错误代码的数值，仅当端口被设置为从站时有效。可能出现的数值及解释会在后文中叙述。
1	N10:101	上次传送错误	此数值是上次由从站端口发送到主站的错误代码。错误代码可能会是 0, 1, 2, 3 和 6。只有当从新设置模块（块 ID255）时，这个数值才会被清除。
2	N10:102	发送到此从站的讯息数目	此数值表示了在这个端口上接收到的，符合此从站地址的讯息数目，无论这些讯息是正确（值得响应）或错误。

端口 1 状态代码 (续)

3	N10:103	从站的所有响应	此数值表示这个端口上从站已经发送到主站的正确（无错）响应数目。假设从站有响应，而且讯息正确。
4	N10:104	从站接收到的所有讯息数目	此数值表示这个端口上从站接收到的所有命令个数，不考虑从站地址。

端口 2 从站代码

字	样例地址	名称	描述
5	N10:105	端口当前状态	描述如上
6	N10:106	上次传送错误	描述如上
7	N10:107	发送到此从站的讯息数目	描述如上
8	N10:108	从站的所有响应	描述如上
9	N10:109	从站接收到的所有讯息数目	描述如上

系统讯息

字	样例地址	名称	描述
10-11	N10:110 N10:111	产品名称 (ASCII)	这两个字表示了模块的产品名称（以 ASCII 方式）。如果是 MCM 产品，当把编程软件设置为 ASCII 字符格式时，就会显示字符“MCM”。
12-13	N10:112 N10:113	修订号 (ASCII)	这两个字表示产品的固件版本号（以 ASCII 方式）。比如，当把编程软件设置为 ASCII 字符格式时，会显示“1.45”。
14-15	N10:114 N10:115	操作系统号 (ASCII)	这两个字表示产品的内部操作系统的版本级别（以 ASCII 方式）。
16-17	N10:116 N10:117	产品运行号 (ASCII)	这个数字代表此产品芯片的生产批次（以 ASCII 方式）。
18-19	N10:118 N10:119	空	

在从站错误代码表中所有的计数在到达 65535 会从 0 重新开始计数

5.1.5 主站错误代码表

MCM 模块监测所有主站端口命令的状态。这些状态数据以主站错误代码表（Master Error Code Table）的方式传送到处理器。表格存储的位置取决于在通讯设置中主站错误代码表指针（Master Error Table Pointer）的设置。每个主站命令都会生成一个错误代码供用户使用。

当上电时，或每次模块接收到 255 设置数据块时，主站错误代码表会初始化为零。

错误代码表是一个 120 个字长的数据块。命令在错误代码表中的代码位置就相对应于命令在命令列表中的位置。

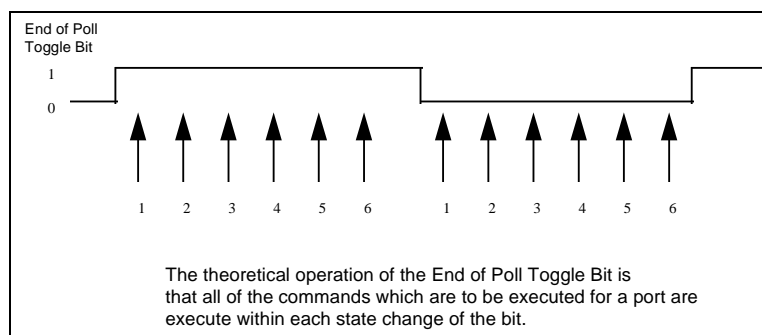
获取主站错误代码表的最简单的方法就是把它定位在应用数据块的最后部分，然后把它作为普通数据读回 PLC/SLC 的数据表。主站错误代码表的结构如下所示：

字	描述
0	命令列表循环扫描结束状态
1	命令 #1 错误状态
2	命令 #2 错误状态
-	
98	命令 #98 错误状态
99	命令 #99 错误状态
100-120	留用

这里:

命令列表循环扫描结束状态: 这个寄存器用来指示主站已经完成一次命令列表的循环扫描。此状态以如下形式表示每个主站端口的状态:

位	
0	主站端口 1
1	主站端口 2



命令错误状态: 此错误状态代码，或者源自于从站，或者由模块产生，放置于表格中。参阅下一个章节来获取错误代码解释。这个数值是一个 16 位的数值，并可以放置在一个整数文件里。注意是用户的应用程序来控制这些寄存器的存放位置和使用方式。

错误代码表示例										
主站错误表指针 = 120										
	Wrd 0	Wrd 1	Wrd 2	Wrd 3	Wrd 4	Wrd 5	Wrd 6	Wrd 7	Wrd 8	Wrd 9
N10:120	0	0	8	0	0	0	0	0	0	0
N10:130	0	0	0	0	0	0	0	0	0	0
N10:140	0	0	0	0	0	0	0	0	0	0
N10:150	0	0	0	0	0	0	0	0	0	0
N10:160	0	0	0	0	0	0	0	0	0	0
N10:170	0	0	0	0	0	0	0	0	0	0
N10:180	0	0	0	0	0	0	0	0	0	0
N10:190	0	0	0	0	0	0	0	0	0	0
N10:200	0	0	0	0	0	0	0	0	0	0
N10:210	0	0	0	0	0	0	0	0	0	0
N10:220	0	0	0	0	0	0	0	0	0	0
N10:230	0	0	0	0	0	0	0	0	0	0

这些寄存器仅对应于本手册后面的 PLC-5 样例程序中使用的寄存器。实际应用需要用户自己特定的程序。在此样例中，命令 2 产生了代码为 8 的错误—其它所有的命令执行都没有错误。第 0 列表明主站端口已经到达了命令列表结束，并开始执行命令列表中最前面的命令。

5.1.6 错误状态代码

在从站和主站错误代码表中返回的错误代码反映了模块执行命令和响应的结果。请注意，在任何情况下，返回数值是 0 则表明没有错误。有效的错误代码如下:

代码	名称	描述
0	全部 OK	模块正常运行
1	非法命令	正在尝试执行一条非法功能代码请求
2	错误数据地址	主站请求的地址或地址范围超出了通讯的范围
3	错误数据数值	命令数据区域的数值无效。
4	检测到不完整的响应	这个错误表示在主站发出请求后，收到一个不完整的响应。这通常是因为从站可能响应过快或线路干扰太多。
6	模块忙	当主站发出的一条写命令还没完成，又接收到一条写命令时，模块会返回错误代码模块忙
8	超时错误	和从站的通讯不成功，因为没有接收到从站的响应。主站端口会尝试 3 次重试，然后接着执行下一条命令。
10	缓冲溢出	接收缓冲溢出，并将字符计数复零。如果出现此状况，请减少每次命令读的数据
16	端口设置错误	如模块返回此错误代码，则表明一个或两个串口设置有误。请检查以下参数，判断错误原因： - 奇偶校验设置 - 停止位设置 - 波特率设置 - 输入寄存器开始地址 - 输出寄存器开始地址
18	系统设置错误	模块返回此错误代码，表明至少有一个系统设置参数超出范围。请检查以下数值，判断错误原因： - 读数据块个数 <= 80 - 写数据块个数 <= 80 - 命令块个数 <= 20 - 从站错误指针 <= 3850 - 主站错误指针 <= 3880
254	校验错误	从站判断出通讯信息校验有误，从而忽略了这条信息
255	TX 硬件超时	传输超时表明模块没有能传输命令。请确认端口的 RTS-CTS 跳线已经连接

5.2 Pass-Through 模式 – 从站模式 [BTR 块 ID 256 到 259]

当从站端口被设置成 Pass-Through 模式，任何目标地址为当地从站地址写命令都会被传送到背板，由梯形逻辑来处理。附录中的梯形逻辑示例表明了如何对这些命令进行解码。

5.2.1 块结构

Pass-Through 模式下 BTR 缓冲定义是：

字	名称	描述										
0	BTR 块 ID	BTR 块 ID 的数值代表了从主站那里接收到的写命令的类型。有效数值为： <table border="0" style="margin-left: 20px;"> <tr> <td>BTR ID</td> <td>描述</td> </tr> <tr> <td>256</td> <td>写寄存器</td> </tr> <tr> <td>257</td> <td>写寄存器 – Enron 浮点数</td> </tr> <tr> <td>258</td> <td>写一个位</td> </tr> <tr> <td>259</td> <td>写多个位</td> </tr> </table>	BTR ID	描述	256	写寄存器	257	写寄存器 – Enron 浮点数	258	写一个位	259	写多个位
BTR ID	描述											
256	写寄存器											
257	写寄存器 – Enron 浮点数											
258	写一个位											
259	写多个位											
1	BTW 块 ID	内容同上										
2-62	数据	这些寄存器的内容就是 BTR 块 ID 号的功能(例如从主站										

	那里接收到的命令)
--	-----------

5.2.2 接收写寄存器 [BTR 块 ID 256 和 257]

BTR 缓冲定义如下:

字	名称	描述										
0	BTR 块 ID	BTR 块 ID 寄存器的数值代表的是从主站那里接收到的写命令的类型。有效代码为: <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>BTR ID</th> <th>描述</th> </tr> </thead> <tbody> <tr> <td>256</td> <td>写寄存器</td> </tr> <tr> <td>257</td> <td>写寄存器 – Enron 浮点数</td> </tr> <tr> <td>258</td> <td>写一个位</td> </tr> <tr> <td>259</td> <td>写多个位</td> </tr> </tbody> </table>	BTR ID	描述	256	写寄存器	257	写寄存器 – Enron 浮点数	258	写一个位	259	写多个位
BTR ID	描述											
256	写寄存器											
257	写寄存器 – Enron 浮点数											
258	写一个位											
259	写多个位											
1	BTW 块 ID	内容同上										
2	个数	主站写寄存器的个数。有效的接收范围是 1 到 60										
3	目标地址	梯形逻辑使用此数值来判断从哪里开始写处理器数据表										
4-62	数据	主站写入的数据。数值为 16 位的寄存器										

5.2.3 接收一个位写 [BTR 块 ID 258]

BTR 缓冲定义为:

字	名称	描述				
0	BTR 块 ID	BTR 块 ID 寄存器的数值代表的是从主站那里接收到的写命令的类型。有效代码为: <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>BTR ID</th> <th>描述</th> </tr> </thead> <tbody> <tr> <td>258</td> <td>写一个位</td> </tr> </tbody> </table>	BTR ID	描述	258	写一个位
BTR ID	描述					
258	写一个位					
1	BTW 块 ID	内容同上				
2	位地址	代表要动作的位				
3	控制动作	主站命令的动作。当数值为 0, 目标地址需要复位。当数值为 1, 目标地址要被置位				

5.2.4 接收多个位 [BTR 块 ID 259]

BTR 缓冲定义是:

字	名称	描述				
0	BTR 块 ID	BTR 块 ID 寄存器的数值代表的是从主站那里接收到的写命令的类型。有效代码为: <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>BTR ID</th> <th>描述</th> </tr> </thead> <tbody> <tr> <td>259</td> <td>写多个位</td> </tr> </tbody> </table>	BTR ID	描述	259	写多个位
BTR ID	描述					
259	写多个位					
1	BTW 块 ID	描述同上				
2	字个数	代表在数据块中包含有效位数值字的个数。有效数值范围是 1 到 30 (这也就限定了一个命令能写的位个数是 30 * 16)。				
3	字开始地址	代表写数据块开始写的字偏移地址。当主站寻址一个写位, 它会发出起始位地址。模块根据起始位地址来生成这个字开始地址 (位地址 / 16)				
4-33	数据	这些寄存器包含了从主站那里接收到的写位数据。注意模块也会接受部分长度的写位命令。使用掩码和一些梯形逻辑来保护一个普通字中的为被寻址的位。				
34-63	掩码	这些字掩盖为被寻址的位。这就可以使起始地址不一定是字交界位, 长度也不一定终止于字的交界位。在俘虜				

	中有示例梯形逻辑.
--	-----------

5.3 解码命令完成位和命令错误位 – 主站模式

在每次数据块传输时（BTR 块 ID0 到 79）时，命令完成和命令错误位都会被返回到梯形逻辑中以便使用。梯形逻辑可以使用这些位来记录命令的执行或者禁止命令执行，如果这个命令被设置为**命令控制模式**（参阅章节 4.5）。

5.3.1 块结构

BTR 块传输缓冲中返回的完成和错误位结构如下所示：

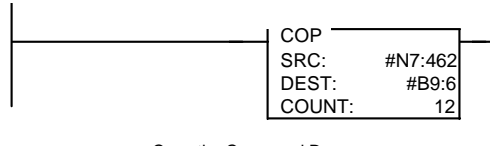
字	名称	描述														
0	BTR 块 ID	当 BTR 块 ID 在 0 到 79 之间时，BT 缓冲包含命令完成和命令错误位														
1	BTW 块 ID	内容同上														
2-51	数据	模块数据如前所述														
52-57	命令完成位	<p>这些寄存器包含在命令列表中前 96 个命令，每个命令的完成位标识。这些完成位以位的形式，根据其在命令列表中相应的位置映射到字中。映射结构如下：</p> <table border="0"> <thead> <tr> <th>字</th> <th>命令</th> </tr> </thead> <tbody> <tr> <td>52</td> <td>1 to 16</td> </tr> <tr> <td>53</td> <td>17 to 32</td> </tr> <tr> <td>54</td> <td>33 to 48</td> </tr> <tr> <td>55</td> <td>49 to 64</td> </tr> <tr> <td>56</td> <td>65 to 80</td> </tr> <tr> <td>57</td> <td>81 to 96</td> </tr> </tbody> </table> <p>示例：字 52 位 0 是命令 #1</p>	字	命令	52	1 to 16	53	17 to 32	54	33 to 48	55	49 to 64	56	65 to 80	57	81 to 96
字	命令															
52	1 to 16															
53	17 to 32															
54	33 to 48															
55	49 to 64															
56	65 to 80															
57	81 to 96															

(续)

字	名称	描述														
58-63	命令错误位	<p>这些寄存器包含在命令列表中前 96 个命令，每个命令的错误位标识。这些错误位以位的形式，根据其在命令列表中相应的位置映射到字中。映射结构如下：</p> <table border="0"> <thead> <tr> <th>字</th> <th>命令</th> </tr> </thead> <tbody> <tr> <td>58</td> <td>1 to 16</td> </tr> <tr> <td>59</td> <td>17 to 32</td> </tr> <tr> <td>60</td> <td>33 to 48</td> </tr> <tr> <td>61</td> <td>49 to 64</td> </tr> <tr> <td>62</td> <td>65 to 80</td> </tr> <tr> <td>63</td> <td>81 to 96</td> </tr> </tbody> </table> <p>示例：字 52 位 0 是命令 #1</p>	字	命令	58	1 to 16	59	17 to 32	60	33 to 48	61	49 to 64	62	65 to 80	63	81 to 96
字	命令															
58	1 to 16															
59	17 to 32															
60	33 to 48															
61	49 to 64															
62	65 to 80															
63	81 to 96															

5.3.1 梯形逻辑

只需要简单的梯形逻辑就可以把完成和错误位从 BTR 缓冲中移动到 PLC/SLC 数据表中。举例如下：



Copy the Command Done
and Error bits from the BTR
buffer to the data table.

6 Modbus 命令设置

ProSoft Technology MCM Modbus 主站从站通讯驱动支持多个数据读和写命令。当设置主站端口时，命令的选择取决于需要查询的数据类型和从站设备支持的 Modbus 层次。当设置为从站时，重要的是理解 Modbus 命令功能，从而决定如何创建数据结构。

我们在附录中摘录了 Modbus 协议规格以帮助您彻底的了解 Modbus 协议。

6.1 Modbus 命令

MCM 模块支持支持 Modbus 协议中主要的读写数据功能。下面的章节详细介绍模块支持的命令。虽然这些叙述主要根据主站端口来描述，但中间的大部分讨论可以帮助你应用从站端口。

功能代码	命令	地址范围	从站驱动	主站驱动
1	Read Output Status	Coil 0001 到 9999	模块根据“Output Status”中返回二进制数据。模块每条命令支持 125 个字。	<p><u>源地址</u>:从站数据读取处的<u>位地址</u>。输入 0 代表 coil 0001</p> <p><u>个数</u>:读位的个数（长度最大到 125 个字）</p> <p><u>目标地址</u>: 模块中用来存放读取回来数据的寄存器内存地址，从 0 开始</p>
2	Read Input Status	位 10001 到 29999	模块根据“Input Status”中返回二进制数据。模块每条命令支持 125 个字。	<p><u>源地址</u>:从站数据读取处的<u>位地址</u>。输入 0 代表位地址 10001</p> <p><u>个数</u>:读位的个数（长度最大到 125 个字）</p> <p><u>目标地址</u>: 模块中用来存放读取回来数据的寄存器内存地址，从 0 开始</p>

(续)

功能代码	命令	地址范围	从站驱动	主站驱动
3	Read Multiple Registers	寄存器 40001 到 47999	模块从寄存器空间返回字。模块的 4000 个字构成寄存器空间，供主站寻址。模块的字 0 对应于 Modbus 地址 40001。模块在一条命令中支持 125 个字。	<p><u>源地址</u>: 从站数据读取处的寄存器地址。输入 0 代表从站的寄存器地址 40001</p> <p><u>个数</u>: 读取字或数值的个数 (长度最大到 125 个字)</p> <p><u>目标地址</u>: 数据存放在模块寄存器内存中的起始地址。</p> <p><u>类型</u>: 读取浮点数时, 控制字节和字的换位 (具体内容参考第 4 章)。</p>
4	Read Input Registers	寄存器 30001 到 39999	模块返回其中的“Input Register”空间返回数据。模块在一条命令中支持 125 个字。	<p><u>源地址</u>: 从站数据读取处的寄存器地址。输入 0 代表从站的寄存器地址 30001</p> <p><u>个数</u>: 读取字或数值的个数 (长度最大到 125 个字)</p> <p><u>目标地址</u>: 数据存放在模块寄存器内存中的起始地址。</p>
5	Single Bit/Coil Write	Coil 0001 到	<p><u>一般模式</u>: 写入模块数据空间的位</p> <p><u>Pass-Through 模式</u>: 写的位会被传送到 PLC/SLC, 由梯形逻辑处理</p>	<p><u>源地址</u>: MCM 中用以判断置位/复位操作命令的起始地址</p> <p><u>个数</u>: 未使用, 默认为 1</p> <p><u>目标地址</u>: 从站中被置位或复位的位地址</p>
6	Single Register Write	寄存器 40001 到 47999	<p><u>一般模式</u>: 写入模块数据空间的寄存器</p> <p><u>Pass-Through 模式</u>: 写的寄存器会被传送到 PLC/SLC, 由梯形逻辑处理</p>	<p><u>源地址</u>: MCM 中用以判断数据源的起始寄存器地址</p> <p><u>个数</u>: 未使用, 默认为 1</p> <p><u>目标地址</u>: 从站中数据写入的寄存器地址。输入地址 0 代表从站中寄存器地址 40001</p>

(续)

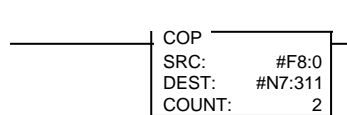
15	Multiple Bit/Coil Write		<p><u>一般模式</u>: 写入模块数据空间的位。最大到 125 个字的位个数。</p> <p><u>Pass-Through 模式</u>: 写的位会被传送到 PLC/SLC, 由梯形逻辑处理。最大到 30 个字的位个数。</p>	<p><u>源地址</u>: MCM 中用以判断置位/复位操作命令的起始位地址</p> <p><u>个数</u>: 写位的数目 (最大到 125 个字)</p> <p><u>目标地址</u>: 从站中数据写入的起始位地址。输入 0 代表从站中 coil 0001</p>
16	Multiple Register Write		<p><u>一般模式</u>: 写入模块数据空间的寄存器。最多到 125 个字。</p> <p><u>Pass-Through 模式</u>: 写的寄存器会被传送到 PLC/SLC, 由梯形逻辑处理。最多 60 个字。</p>	<p><u>源地址</u>: MCM 中用以判断数据源的起始寄存器地址</p> <p><u>个数</u>: 写字或数值的数目 (最多 125 个字)</p> <p><u>目标地址</u>: 从站中数据写入的寄存器地址。输入地址 0 代表从站中寄存器地址 40001</p>

6.2 浮点数支持

只要设备支持 IEEE754 浮点数格式, 在 MCM 模块和其他设备之间传送浮点数数据就非常方便。这个 IEEE 格式是 32 位单精度浮点数格式。

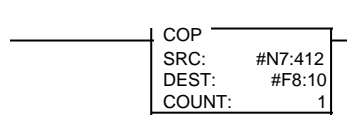
移动浮点数的编程方法是使用 PLC 和 SLC 中的 COP 命令。PLC/SLC 中的 COP 指令是个非常特别的数据移动指令, 因为它是无数据类型的指令。这就意味着在数据移动的时候, 指令不进行数据转换 (例如, 它是一种镜像复制, 而不是数值复制)。

用 COP 指令把一个浮点数移动到一个整数文件 (你需要这个操作来把浮点数移动到模块) 如下所示:



这个命令把一个浮点数移动到两个 16 位的整数镜像中。对于多个浮点数, 只需要增加移动的个数就可以了, 个数要每两个递增。

使用 COP 指令把整数文件移动到浮点数文件（你需要这个操作来接收来自模块的浮点数数据）如下所示：



这个命令会把两个 16 位的整数寄存器（包含了一个浮点数数值镜像）移动到浮点数文件。对于多个数值，只需要增加移动个数就可以了。

6.2.1 ENRON 浮点数支持

许多生产厂商在他们的设备中集成了一些特殊的功能支持，比如我们通常说的 Enron 版本的 Modbus 协议。在这种功能中，寄存器地址 > 7000 的数据是浮点数数据。这样做的好处在于，在命令的“个数”参数就变成了“数值个数”。在浮点数格式中，每个数值代表 2 个字。MCM 使用下面的方式来应用这种功能：

功能代码	主站驱动	从站驱动
3 - Register Read	如果目标地址 ≥ 7000 ，模块会认为每个数值由两个字组成。	如果主站的寄存器寻址 ≥ 7000 ，模块会使用下面这个表达式来判断 MCM 寄存器地址读取的位置： MCM 地址 = 浮点数偏移 + (寄存器地址 - 7000) * 2
6/16 - Register Write	如果目标地址 ≥ 7000 ，模块会对每个个数发送两个字（个数数值代表数值的个数，而不再是字的个数）	如果主站寻址的寄存器地址 ≥ 7000 ，模块会使用下面这个表达式来判断 MCM 寄存器地址写入的位置： MCM 地址 = 浮点数偏移 + (寄存器地址 - 7000) * 2

7 诊断和故障排除

一些硬件诊断信息可以使用模块前端的 LED 指示灯来获取。下面的章节解释了 PLC 和 SLC 平台的每个 LED 的含义。

7.1 3100 PLC 平台 LED 指示灯

PLC 平台的 MCM 产品是基于 ProSoft CIM 硬件平台。下面的表格记录 3100-MCM 硬件的 LED，并解释了 LED 的运作。

ProSoft CIM Card		
ACTIVE	○ ○	FLT
CFG	○ ○	BPLN
ERR1	○ ○	ERR2
TXD1	○ ○	TXD2
RXD1	○ ○	RXD2

ProSoft CIM	颜色	状态	指示
ACT	绿	闪烁 (快)	<u>一般状态</u> : 模块正常运作, 并且成功的和 PLC 进行块传输
		On	模块从背板接收到了供电, 但仍然存在其他问题
		Off	模块尝试和 PLC 进行块传输, 但失败了。PLC 也许处于 PGM 模式, 或出现故障
FLT	红	Off	<u>一般状态</u> : 后台检测中没有系统故障
		On	后台检测中发现系统故障。请联系生产厂
CFG	绿	Off	<u>一般状态</u> : 此时没有于设置相关的操作发生
		闪烁	每次从处理器梯形逻辑接收到模块设置块 (ID = 255) 时都会闪烁一次
		On	当设置有错误时, 这个指示灯会长亮。问题可能时端口设置数据或系统设置数据。具体内容参阅第 4 章
BPLN	红	Off	<u>一般状态</u> : 如果此灯熄灭并且 ACT 等快速闪烁, 模块正在和 PLC 进行块传输
		On	表明 PLC 和模块之间的块传输失败 (在第一代产品中无此含义)
ERR1 ERR2	琥珀色	Off	<u>一般状态</u> : 当 error LED 熄灭并且相关端口正在传送数据时, 说明没有通讯错误
		闪烁	在数据通讯中存在周期性的通讯错误。参阅第 4 章来判断错误条件
		On	此 LED 会在一些条件下长亮: <ul style="list-style-type: none"> • CTS 输入未满足要求 • 端口设置错误 • 系统设置错误 • MCM 从站通讯失败 • 返回错误条件到 MCM 主站
Tx1 Tx2	绿	闪烁	端口正在传送数据
Rx1 Rx2	绿	闪烁	端口正在接收数据

7.2 3150 SLC 平台 LED 指示灯

下面的表格记录 3150-MCM 硬件上的 LED，并解释这些 LED 的含义。

COMMUNICATIONS			
■ ACT	■ FAULT		
■ CFG	■ BPLN		
■ PRT1	■ ERR1		
■ PRT2	■ ERR2		

LED 名称	颜色	状态	指示
ACT	绿	闪烁 (快)	<u>一般状态</u> : 模块正常运行, 并且成功的和 SLC 进行块传输
		On	模块从背板接收到了供电, 但仍然存在其他问题
		Off	模块尝试和 SLC 进行块传输, 但失败了。SLC 也许处于 PGM 模式, 或出现故障
FLT	红	Off	<u>一般状态</u> : 后台检测中没有系统故障
		On	后台检测中发现系统故障。请联系生产厂
CFG	绿	Off	<u>一般状态</u> : 此时没有于设置相关的操作发生
		闪烁	每次从处理器梯形逻辑接收到模块设置块 (ID = 255) 时都会闪烁一次
		On	当设置有错误时, 这个指示灯会长亮。问题可能时端口设置数据或系统设置数据。具体内容参阅第 4 章
BPLN	红	Off	<u>一般状态</u> : 如果此灯熄灭并且 ACT 等快速闪烁, 模块正在和 SLC 进行块传输
		On	表明 SLC 和模块之间的块传输失败
ERR1 ERR2	琥珀色	Off	<u>一般状态</u> : 当 error LED 熄灭并且相关端口正在传送数据时, 说明没有通讯错误
		闪烁	在数据通讯中存在周期性的通讯错误。参阅第 4 章来判断错误条件
		On	此 LED 会在一些条件下长亮: <ul style="list-style-type: none"> • CTS 输入未满足要求 • 端口设置错误 • 系统设置错误 • MCM 从站通讯失败 • 返回错误条件到 MCM 主站
PRT1 PRT2	绿	闪烁	端口正在通讯, 传送或者接收数据

7.3 故障排除 - 常规

为了帮助你对模块进行故障排除, 我们整理了下面这个表格。请使用下面这些帮助来使用模块, 但如果你有其他需求或者问题请别犹豫联系我们。

下面的内容时根据模块上电后最可能发生问题的顺序来排列。

问题描述	对应步骤
BPLN 灯长亮 (SLC)	<p>当模块认为 SLC 未处于运行模式时，BPLN 灯长亮（例如 SLC 处于 PGM 或者故障）。如果 SLC 处于运行模式，请检查下列事项：</p> <ul style="list-style-type: none"> • 检查 SLC Status File，确认这个槽位处于允许状态 • 传输使能/完成位 (模块槽位 I/O 的位 0) 应受控于梯形逻辑。请参考第 2。 • 如果为模块服务的梯形逻辑在子程序中，请确认有 JSR 命令调用这个子程序
上电后 CFG 灯不熄灭 (ERR 未亮)	模块没有检测到 BTW 块 ID 号 255。这可能时因为块传输失败 (PLC) 或者梯形逻辑存在错误，阻止把 255 移动到 BTW 缓冲中
上电后 CFG 灯不熄灭 (ERR 亮)	如果 BPLN 灯熄灭，说明模块检测到一些端口和系统设置错误。请查看错误状态表来确定决定具体原因。
CFG 灯明暗交替	在通常情况下，CFG 指示灯会在接收完设置后立即熄灭。如果指示灯明暗交替，说明梯形逻辑中向 BTW 缓冲写 255 块 ID 号的程序条件没有清除。请检查梯形逻辑确保赋值 255 的逻辑条件没有被保持为真。
模块没有传输数据	<p>假设处理器处于运行模式，检查下列事项：</p> <ul style="list-style-type: none"> • CTS 输入未符合要求（检查 RTS/CTS 跳线） • 检查错误状态代码是否是 255。如果是此数值，请参考下个问题 • 在从站模式下，请确认从站地址是否正在被主站查询 • 在主站模式下，请确认命令列表设置，而且命令列表被传送到模块（例如，检查命令块个数和相关的梯形逻辑）
状态表中出现错误代码 255	这只会是一个原因，端口丢失 CTS 输入。如果连接到端口的一条电缆，请检查 RTS 和 CTS 之间是否有跳线。如果有跳线，那模块可能出现硬件故障
重复写数据块	如果没有注意到 BTW 块 ID 的数值是由模块控制的，而且是从 0 开始的，那很可能发生这种情况。请确保模块的设置（读和写块个数）没让 PLC/SLC 的数据覆盖模块返回的数据。检查这个问题的简单有效办法就是根据 BTW 块 ID 寄存器绘制一个图表
数据位置颠倒（仅 3100）	在一些条件下模块中的数据会出现位置颠倒的现象。这种颠倒的现象通常和模块后面的 8/16 点跳线相关。请确认跳线处于 8 点的位置
模块不接受新的设置数值	<p>为了让新的数值移动到模块，必须传送块 ID 号为 255 的写传输块到模块。样例程序中的 'User Config Bit' 用来完成此功能。在梯形逻辑中，这个位需要手动置位，或者模块重新上电或复位。</p> <p>若想在 PGM 模式切换到 RUN 模式时下载设置，只需要简单的添加一行程序，根据首次扫描状态位 (S1:1/15) 对 'User Config Bit' 置位</p>
问题描述	对应步骤
在无命令位置上返回错误代码（主站设置）	<p>确保命令块个数设置正确。在写程序行中，每个写入的命令块都应该有一个对应的分支（例如，命令块个数为 2 时，梯形逻辑中要有两个分支来处理 BTW 块 ID 80 和 81。</p> <p>如果命令块个数设置数值超过梯形逻辑中的分支个数，命令列表会不经意的被复制。想解决这个问题，要么添加更多的分支或减少命令块个数以符合 BTW 逻辑分支的个数。</p>

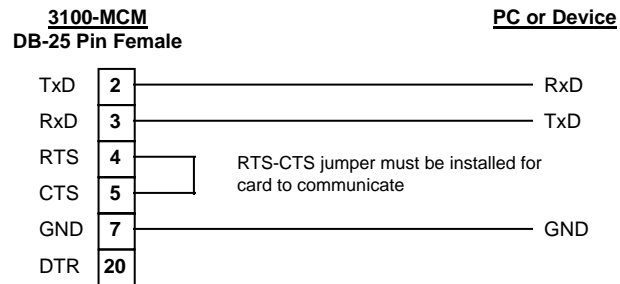
RX1 或 RX2 长亮 (仅 3100)	模块的 TX 和 RX 指示灯直接反映端口的硬件状态 (也就是说其不受控于固件)。当 RX 指示灯长亮, 这表明连接到端口的电缆连接极性颠倒。 特别是在 RS-485 和 RS-422 模式。

8 电缆连接示意图

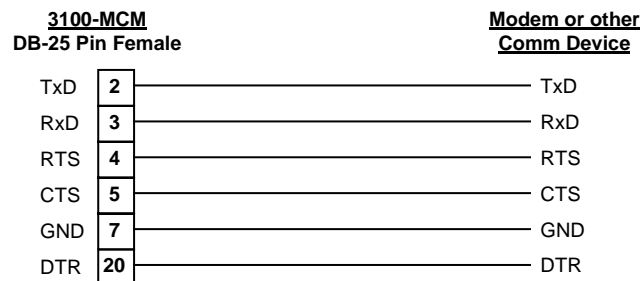
下面的示意图是连接到 3100 和 3150 端口的接线方法。

3100-MCM 模块

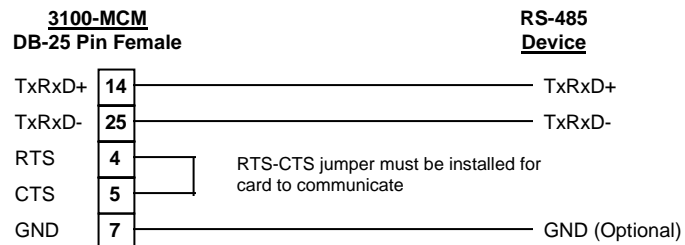
RS-232 无硬件握手
和另一通讯端口的连接



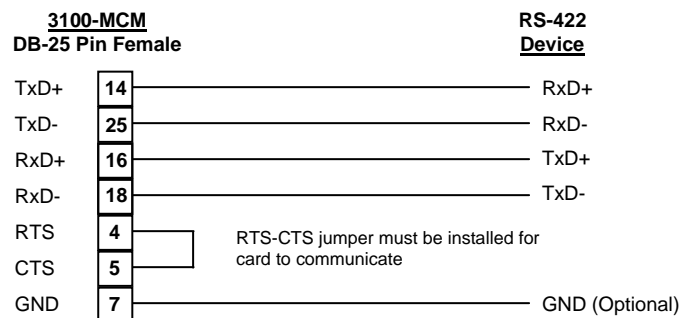
RS-232 硬件握手
端口连接到 Modem 或类似设备



RS-485/2-线连接
对于所有的 2 线应用，模块上的跳线
必须设置到 RS-485 位置

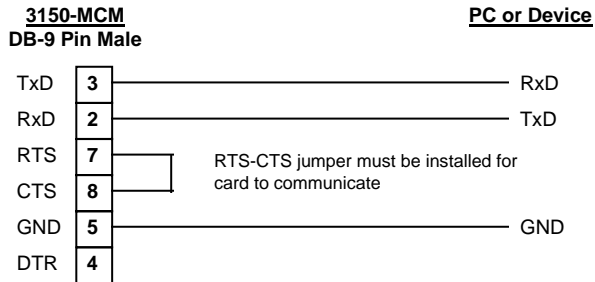


RS-422/4-线连接
对于所有的 4 线应用，模块上的跳线
必须设置到 RS-482 位置

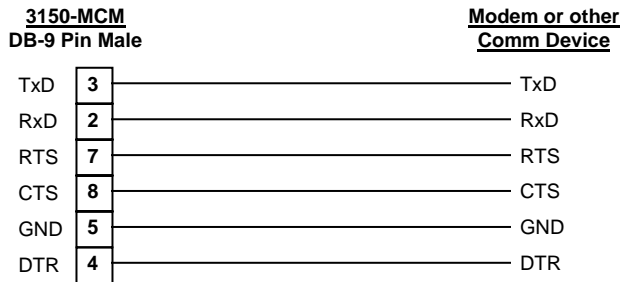


3150-MCM 模块

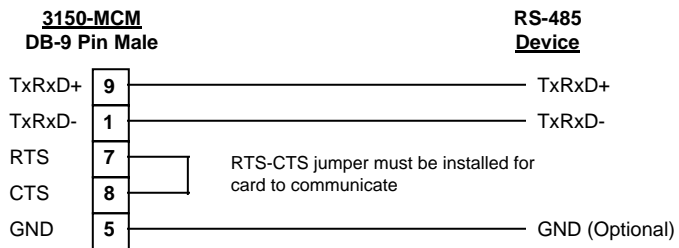
RS-232 无硬件握手
和另一通讯端口的连接



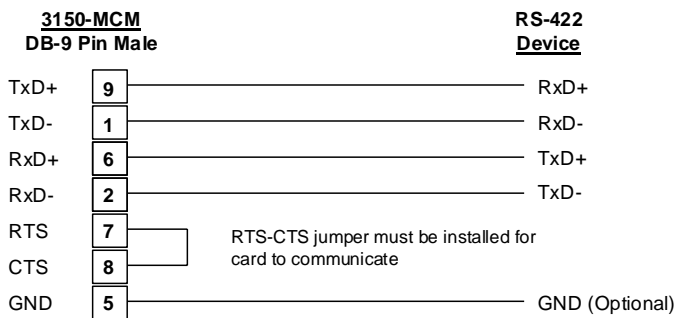
RS-232 硬件握手
端口连接到 Modem 或类似设备



RS-485/2-线连接
对于所有的 2 线应用，模块上的跳线
必须设置到 RS-485 位置



RS-422/4-线连接
对于所有的 4 线应用，模块上的跳线
必须设置到 RS-482 位置



RS-485 和 RS-422 提示

如果 RS-422/RS-485 模式下的第一次通讯失败了，在进行其他所有尝试前，先尝试交换终端极性。一些制造商对于 +/- 和 A/B 极性有自己的定义。

A 支持, 服务和保修

技术支持

ProSoft Technology 的生存依赖于其为用户提供有意义服务的能力。如果有任何问题和疑问, 那就请联系我们。地址: :

Factory/Technical Support
ProSoft Technology, Inc.
1675 Chester Avenue, 4th Floor
Bakersfield, CA 93301
(661) 716-5100
(661) 716-5101 (fax)

E-mail 地址: prosoft@prosoft-technology.com

网址: <http://www.prosoft-technology.com>

在打电话寻求支持前, 请先准备好。为了提供可能的最好和最快的支持, 我们通常会问讯以下信息 (您也可以在打电话前把这些信息发传真给我们):

1. 产品版本号
2. 系统级别
3. 模块设置和 MCM.CFG 文件的内容
4. 模块运作
 - 设置/调试状态信息
 - LED 状态
5. 根据 RSLogix5000 和处理器上 LED 状态, 处理器和数据区的信息
6. 串行网络的详细内容

在非工作时间内, 一个应答系统 (根据 Bakersfield 号码) 能够在任何时候寻呼到我们资深的技术和/或应用支持来回答您的重要问题。

模块服务和维修

MVI56-MCM 插卡是一个电子产品。其被设计和制造成能够在一些不利的条件下正常工作。但对于任何产品, 在经过老化, 误用或任何可能的问题的之后, 模块可能需要维修。从 ProSoft Technology 购买后, 根据保修条款指定的限制, 模块具有一年的部件和人力保修。更换和/或返回都应联系产品购买的分销商。如果你需要返回模块维修, 就要从 ProSoft Technology 得到一个 RMA 号码。请致电工厂告知这个号码, 并把这个号码显著的标识在用于包装返回模块的包装上。

常规保修政策

ProSoft Technology, Inc. (下文简称 ProSoft) 保证此产品遵从于并符合公布的技术规格和附带的文字资料, 还应在材料和工艺上无缺陷。保证时间是从接收到产品开始的保修期。

这种保修受限于维修和/或替换, 这基于 ProSoft 对缺陷或不一致产品的选择。并且, ProSoft 不应负责产品在执行指定功能时的失败, 或任何其它不一致引起的或归因于: (a) 任何产品的错误应用或错误使用; (b) 用户未遵守任何 ProSoft 规格或用法说明; (c) 产品的疏忽, 滥用或意外事故; 或 (d) 任何非 ProSoft 提供的相关或补充的设备或软件。

有限保修服务可以通过发送产品到 ProSoft 并提供购买证明或接收日期来获得。用户同意对产品保险或承担运输中遗失或损坏的风险, 并预先向 ProSoft 支付运费, 使用原包装盒或相等物。获取更多信息, 请联系 ProSoft 用户服务。

有限义务

除了这里表明的以外, PROSOFT 不担保任何, 明显或不言而喻的, 对应于任何依照这个协议的设备, 部件或服务, 这包括但不限于对某个特殊用途的明显的产品效率和适应性担保。PROSOFT 和其经销商不负责其它任何损失, 包括但不限于直接, 间接, 意外, 特殊或因果关联损坏, 不论这个合同内的动作或民事侵权行为 (包括疏忽和严格赔偿责任), 比如, 但不限于, 预期利润的损失和, 起源于或基于或有关, 使用或装备这个设备, 部件或服务的性能使用或无效的损失, 经管 PROSOFT 或其经销商的完整义务超过这个产品的价格。

受州法律指导, 上述例外和限制的部分内容在一些州中并不适用。保修提供了指定的合法权利; 其它由于州的不同也会有所不同。此保修不应应用于那些被任何联邦, 州或市政法律规定不能优先取得的部分条款

硬件产品保修细则

保修期: ProSoft 硬件产品保修期为 1 年。

保修程序: 在 ProSoft 接收到返回的硬件产品时, ProSoft 会, 按期选择, 维修或更换产品且不收取任何费用, 预付运费, 除去以下内容。维修部件和更换产品将会装备在调换基础上, 并会是可使用的或新的。所有更换下的产品和部件都将是 ProSoft 的财产。如果 ProSoft 判断出产品超出保修范围, 则会根据用户的选择, 按照 ProSoft 的部件和人力的标准收费等级维修产品, 和运回产品。

B 产品规格

3100/3150-MCM (“Modbus 通讯模块”) 产品系列让 Allen-Bradley 1771 和 1746 I/O 兼容处理器方便的和其他 Modbus 协议兼容设备进行通讯, 既可作为 Modbus 主站也可作为 Modbus 从站。

MCM 产品包括下列标准特性:

一般规格

- 两个可完全设置的串口, 每个都可以支持 Modbus Master 或 Modbus Slave。设置内容包括:

端口设置	端口 1	端口 2
主站-主站	主站	主站
主站-从站	主站	从站
从站-从站	从站	从站

- 支持存储, 传输 4000 个寄存器到 PLC /SLC 数据表
- 支持移动位, 整数, ASCII 和浮点数等数据类型
- 用户可通过数据表设置来定义内存映射
- 具有 RS-232C 握手, 可应用于 SCADA 电台/modem 应用
- 具有 RS-422/RS-485 接口, 可应用于多节点应用, 每个端口支持 32
- 每个端口可设置字符间超时, 支持卫星和报文电台

- 软件设置 (从处理器梯形逻辑)

从站地址	:	1 to 247 (0 is broadcast)
奇偶校验	:	无, 奇校验或者偶校验,
停止位	:	1 或 2
波特率	:	300 到 38,400
RTS 至 TxD	:	0-65535 ms, 精度 1 ms
RTS Off	:	0-65535 ms, 精度 1 ms
超时	:	0-65535 ms, 精度 1 ms

- 响应时间

Modbus 主站和从站协议驱动是由高级语言编写, 编译而成。因此, Modbus 驱动全面的使用了硬件中断功能以减少延迟, 优化程序性能

Modbus 从站规格

- 协议模式:

RTU 模式, CRC-16 错误校验
ASCII 模式, LRC 错误校验 (7 和 8 数据位格式)

- 支持的 Modbus 功能代码:

1	Read Output Status
2	Read Input Status
3	Read Multiple Data Registers
4	Read Input Registers
5	Force Single Coil (Latch/Unlatch)
6	Preset (Write) Single Data Register
8	Loopback Test (Test 0 only)
15	Multiple Coil Latch/Unlatch
16	Preset (Write) Multiple Data Register

- 支持主站的广播命令
- 错误状态和通讯状态返回至梯形处理器
- Pass Through 模式 – 可设置使能
可选择把来自主站的写命令传输到梯形逻辑处理。支持有条件的接受梯形逻辑的写数据
- 命令路由模式
支持 6 个从站地址的从站到主站路由。允许从站端口的监视器从路由从站处获取数据

Modbus 主站规格

- 协议模式:

RTU 模式, CRC-16 错误校验
ASCII 模式, LRC 错误校验 (7 和 8 数据位格式)

- 支持的 Modbus 功能代码:

1	Read Output Status
2	Read Input Status
3	Read Multiple Data Registers
4	Read Input Registers
5	Force Single Coil (Latch/Unlatch)
6	Preset (Write) Single Data Register
15	Multiple Coil Latch/Unlatch
16	Preset (Write) Multiple Data Register
- 支持 100 条命令输入，每条命令都可以设置以下参数:
 - 端口/模式选择
 - 从站地址
 - 功能代码
 - 源/目标数据地址
 - 传输数值的数目
 - 轮训时间
- 命令控制模式
 - 允许单个命令有条件的执行，梯形逻辑可以根据 PLC/SLC 的时间来触发命令
- 可读取每条命令的“完成”和“错误”位
- 支持梯形逻辑发出的“事件驱动”写命令
- 每个命令的错误状态代码返回到梯形逻辑处理器
- 支持到从站的广播命令

硬件规格

- 背板电流负载:

3100	: 0.65 A
3150	: 0.15 A , 5 V
	0.04 A , 24 V
- 工作温度: 0 to 60 °C
- 存储温度: -40 to 85 °C
- 连接:

3100	: 2 - DB25 孔形连接器
3150	: 2 - DB9 针形连接器

C Modbus 协议规格

Read Output Status (功能代码 01)

请求

用户使用此命令可以从目标从站读取逻辑线圈的 ON/OFF 状态，这些线圈用于控制开关量输出。此功能代码不支持广播模式。除了从站地址和功能代码区域，通讯帧中还需要包含用以询问状态信息的读线圈起始地址和数目。

每个请求中支持最多到 2000 个线圈；但特定的从站设备也许会有地址的最大限制。线圈地址从 0 计数；（线圈号 1=0，线圈号 2=1，线圈号 3=2，等等）。

图 C1 是 read output status 请求的示例，从从站设备 11 中读取线圈 0020 到 0056。

ADR	FUNC	DATA START PT HO	DATA START PT LO	DATA # OF PTS HO	DATA # OF PTS LO	ERROR CHECK FIELD
11	01	00	13	00	25	CRC

图 C1 Read Output Status 请求帧

应答

图 C2 是一个应答 Read Output Status 的示例。每个线圈被封装到位。应答中包含从站地址，功能代码，数据字符的数目，数据字符和错误校验。数据打包，每个位对应一个线圈 (1 = ON, 0 = OFF)。第一个字符的最低位包含寻址目标的线圈，后面的依次排列。如果线圈个数不是 8 的偶数倍，最后一个字符会从高端位填零。字符的数目特指 RTU 字符的数目，例如 RTU 或 ASCII 下，数目是相等的。

因为从站接口设备是在控制器扫描结束时才响应通讯，数据反映的也是扫描结束时线圈的状态。一些从站会限制每次扫描提供的线圈数目。对于多线圈数目，必须根据顺序扫描的线圈状态，通过经过多次 PC 处理。

ADR	FUNC	BYTE COUNT	DATA COIL STATUS 20-27	DATA COIL STATUS 28-35	DATA COIL STATUS 36-43	DATA COIL STATUS 44-51	DATA COIL STATUS 52-56	ERROR CHECK FIELD
11	01	05	CD	6B	B2	OE	1B	CRC

图 C2 Read Output Status 应答帧

线圈 20-27 的状态以此表达 CD(HEX) = 1100 1101 (Binary)。从左到右读，说明线圈 27, 26, 23, 22, 和 20 都是 on。其它线圈数据字节以同样的顺序编码。根据请求的线圈状态数目，最后的数据区域，也就是 1B (HEX) = 0001 1011 (Binary)，仅包含 5 个线圈 (52-56)，而不是 8 个线圈。最左端的 3 个位填零来填充 8-位格式。

Read Input Status (功能代码 02)

请求

用户使用此命令可以从目标从站读取开关量输入的 ON/OFF 状态。此功能代码不支持广播模式。除了从站地址和功能代码区域，通讯帧中还需要包含用以询问状态信息的输入起始地址和数目。

每个请求中支持最多到 2000 个输入；但特定的从站设备也许会有地址的最大限制。输入地址从 0 计数；（输入 1=0，输入 2=1，输入 3=2，等等）。

图 C3 是 read input status 请求的示例，从从站设备 11 中读取输入 10197 - 10218。

ADR	FUNC	DATA START PT HO	DATA START PT LO	DATA #OF PTS HO	DATA #OF PTS LO	ERROR CHECK FIELD
11	02	00	C4	00	16	CRC

图 C3 Read Input Status 请求帧

应答

图 C4 是一个应答 Read Input Status 的示例。每个输入被封装到位。应答中包含从站地址，功能代码，数据字符的数目，数据字符和错误校验。每个位对应一个输入 (1 = ON, 0 = OFF)。第一个字符的最低位包含寻址目标的输入，后面的依次排列。如果输入个数不是 8 的偶数倍，最后一个字符会从高端位填零。字符的数目特指 RTU 字符的数目，例如 RTU 或 ASCII 下，数目是相等的。

因为从站接口设备是在控制器扫描结束时才响应通讯，数据反映的也是扫描结束时输入的状态。一些从站会限制每次扫描提供的输入数目。对于多输入数目，必须根据顺序扫描的输入状态，通过经过多次 PC 处理。

ADR	FUNC	BYTE COUNT	DATA DISCRETE INPUT 10197-10204	DATA DISCRETE INPUT 10205-10212	DATA DISCRETE INPUT 10213-10218	ERROR CHECK FIELD
11	02	03	AC	DB	35	CRC

图 C4 Read Input Status 应答帧

输入状态 10197-10204 以此表达 AC (HEX) = 10101 1100 (binary)。从左到右读，说明输入 10204, 10202, 和 10199 都是 on。其它输入数据字节以同样的顺序编码。根据请求的输入状态数目，最后的数据区域，也就是 35 HEX = 0011 0101 (binary) 仅包含 6 个输入(10213-10218) 的状态而不是 8 个输入。最左端的 2 个位填零来填充 8-位格式。

Read Holding Registers (功能代码 03)**请求**

用户使用 Read holding registers (03) 来读取目标从站 4xxxx 寄存器的二进制数据。这些寄存器用以存储相关计时器和计数器的数值来驱动外部设备。每个请求中支持最多 125 个寄存器；但特定的从站设备也许会有地址的最大限制。寄存器从 0 开始计数 (40001 = 0, 40002 = 1, 等等)。不支持广播模式。The broadcast mode is not allowed.

下面的例子从从站 11 中读取寄存器 40108 到 40110。

ADR	FUNC	DATA START REG HO	DATA START REG LO	DATA #OF REGS HO	DATA #OF REGS LO	ERROR CHECK FIELD
11	03	00	6B	00	03	CRC

图 C5 Read Holding Register 请求帧

应答

目标从站的应答包括它的地址和功能代码，后面是信息区。信息区包含的一个字节，描述返回的数据字节的数目。所请求的寄存器内容 (DATA)，每个有两个字节的内容，表示每两个字符的二进制内容。第一个字节包含高位，第二个字节是低位。

因为从站接口设备是在控制器扫描结束时才响应通讯，数据反映的也是扫描结束时寄存器的状态。一些从站会限制每次扫描提供的寄存器数目；对于大的寄存器数目，必须通过连续多次传输来读取。

下面的例子中，寄存器 40108-40110 包含十进制数值 555, 0, 和 100。

ADR	FUNC	BYTE CNT	H.O. DATA	L.O. DATA	H.O. DATA	L.O. DATA	H.O. DATA	L.O. DATA	ERROR CHECK FIELD

11	03	06	02	2B	00	00	00	64	CRC
----	----	----	----	----	----	----	----	----	-----

图 C6 Read Holding Register 响应帧

Read Input Registers (功能代码 04)**请求**

用户使用功能代码 04 来读取控制 3xxxx 的输入寄存器。这些地址从连接到 I/O 结构的设备获取数值，只能被读取，不能改变。每个请求中支持最多 125 个寄存器；但特定的从站设备也许会有地址的最大限制。寄存器从 0 开始计数（30001 = 0, 30002 = 1, 等等）。不支持广播模式。

下面的例子从从站 11 中读取寄存器 3009 的内容。

ADR	FUNC	DATA START REG HO	DATA START REG LO	DATA #OF REGS HO	DATA #OF REGS LO	ERROR CHECK FIELD
11	04	00	08	00	01	CRC

Figure C7 Read Input Register Query Message

应答

目标从站的应答包括它的地址和功能代码，后面是信息区。信息区包含的一个字节，描述返回的数据字节的数目。所请求的寄存器内容（DATA），每个有两个字节的内容，表示每两个字符的二进制内容。第一个字节包含高位，第二个字节是低位。

因为从站接口设备是在控制器扫描结束时才响应通讯，数据反映的也是扫描结束时寄存器的状态。一些从站会限制每次扫描提供的寄存器数目；对于大的寄存器数目，必须通过连续多次传输来读取。

下面的例子中，寄存器 3009 中的数值为 0。

ADR	FUNC	BYTE COUNT	DATA INPUT REG HO	DATA INPUT REG LO	ERROR CHECK FIELD
11	04	02	00	00	E9

图 C8 Read Input Register 应答帧

Force Single Coil (功能代码 5)**请求**

此命令对一个线圈置位或复位。控制器内的任何线圈都可以被置位或复位。但是，因为控制在动态的扫描，除非线圈被禁止，控制器仍然可以改变线圈的状态。线圈从 0 开始计数（线圈 0001 = 0, 线圈 0002 = 1, 等等）。数值 65,280 (FF00 HEX) 把线圈置位，数值 0 可以把线圈关闭；其它所有的数值都无效，不会对线圈产生动作。

若使用从站地址 00（广播模式）会强制所有连接的从站更改目标线圈。

注意

只有功能 5, 6, 15, 和 16 可以作为广播模式发送。

下面例子从从站 11 中请求打开线圈 0173。

ADR	FUNC	DATA COIL # HO	DATA COIL # LO	DATA ON/OFF IND	DATA	ERROR CHECK FIELD
11	05	00	AC	FF	00	CRC

图 C9 Force Single Coil 请求帧

响应

对命令请求的正常应答是，在线圈状态改变后把接收的帧再发送一遍。

ADR	FUNC	DATA COIL # HO	DATA COIL # LO	DATA ON/ OFF	DATA	ERROR CHECK FIELD
11	05	00	AC	FF	00	CRC

图 C10 Force Single Coil 应答帧

不论线圈是否有效，通过 MODBUS 功能 5 来动作线圈都会完成。（对于 ProSoft 产品，仅当存在必须的梯形逻辑时，线圈才会被动作）。

注意

Modbus 协议不包含检测或改变开关量输入和输出禁止状态的标准功能。但，设备自身的某些命令确可以完成此功能。（对于 ProSoft 产品，这是通过梯形逻辑来实现）。

在处理器逻辑程序中重新编制的线圈不会上电的时候自动清除。因此，如果这个线圈被功能代码 5 置位，并且（甚至数月以后），连接到线圈的输出会非常的“热”。

Preset Single Register (功能代码 06)**请求**

用户使用功能代码 06 来改变一个保持寄存器的内容。控制器内的任何保持寄存器都可以通过此命令来改变其内容。但因为控制器一直在扫描，它可以在任何时候改变任何保持寄存器的内容。数值以二进制形式表示，直至控制器的最大容量。未使用的高位必须设置为 0。如果和从站地址 0 一起使用（广播模式），所有的从站控制器都会把指定的寄存器赋以响应的数值。

注意

只有功能 5, 6, 15, 和 16 可以作为广播模式发送。

ADR	FUNC	DATA START REG HO	DATA START REG LO	DATA #OF REGS HO	DATA #OF REGS LO	ERROR CHECK FIELD
11	06	00	01	00	03	CRC

图 C11 Preset Single Register 请求帧

应答

对于一个 preset single register 的请求应答就是在寄存器改变之后，重新传输请求的帧。

ADR	FUNC	DATA REG HI	DATA REG LO	DATA INPUT REG HO	DATA INPUT REG LO	ERROR CHECK FIELD
11	06	00	01	00	03	CRC

图 C12 Preset Single Register 应答帧

Force Multiple Coils (功能代码 15)**请求**

此命令对一个连续线圈块中的每个线圈置位或复位。控制器内的任何线圈都可以被置位或复位。但是，因为控制在动态的扫描，除非线圈被禁止，控制器仍然可以改变线圈的状态。线圈从 0 开始计数（线圈 0001 = 0, 线圈 0002 = 1, 等等）。每个线圈的目标状态在数据区内被封装，每个位对应每个线圈（1= ON, 0= OFF）。使用从站地址 0（广播模式）会强制所有连接的从站改变目标线圈。

注意

只有功能 5, 6, 15, 和 16 可以作为广播模式发送（除循环诊断检测）。

下面的例子强制从地址 20(13 HEX)开始的 10 个线圈。这两个数据区，CD =1100 和 00 = 0000 000，表示线圈 27, 26, 23, 22, 和 20 被置位。

ADR	FUNC	H.O. ADD	L.O. ADD	QUANTITY		BYTE CNT	DATA COIL STATUS 20-27	DATA COIL STATUS 28-29	ERROR CHECK FIELD
11	0F	00	13	00	0A	02	CD	00	CRC

图 C15 Force Multiple Coils 请求帧

应答

通常的应答是回复从站地址，功能代码，起始地址和动作线圈的数目。

ADR	FUNC	H.O. ADDR	L.O. ADDR	QUANTITY		ERROR CHECK FIELD
11	0F	00	13	00	0A	CRC

图 C16 Force Multiple Coils 应答帧

不论线圈是否有效，通过 MODBUS 功能 15 来动作线圈都会完成。

在处理器逻辑程序中重新编制的线圈不会上电的时候自动清除。因此，如果一个线圈被功能代码 15 置位，并且（甚至数月以后），连接到线圈的输出会非常的“热”。

Preset Multiple Registers (功能 16)**请求**

用户可以使用此命令来改变控制器中的保持寄存器（最多 60 个寄存器）。但因为控制器一直在扫描，它可以在任何时候改变任何保持寄存器的内容。数值以二进制形式表示，直至控制器的最大容量（184/384 和 584 是 16 位）；未使用的高位必须设置为 0。

注意

只有功能 5, 6, 15, 和 16 可以作为广播模式发送。

ADR	FUNC	H.O. ADD	L.O. ADD	QUANTITY		BYTE CNT	H.O. DATA	L.O. DATA	H.O. DATA	L.O. DATA	ERROR CHECK FIELD
11	10	00	87	00	02	04	00	0A	01	02	CRC

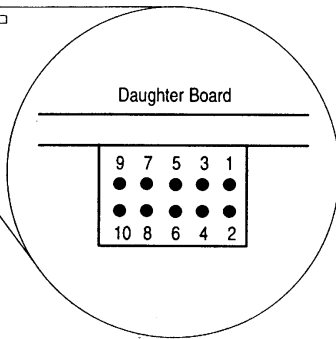
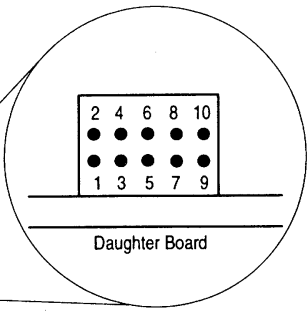
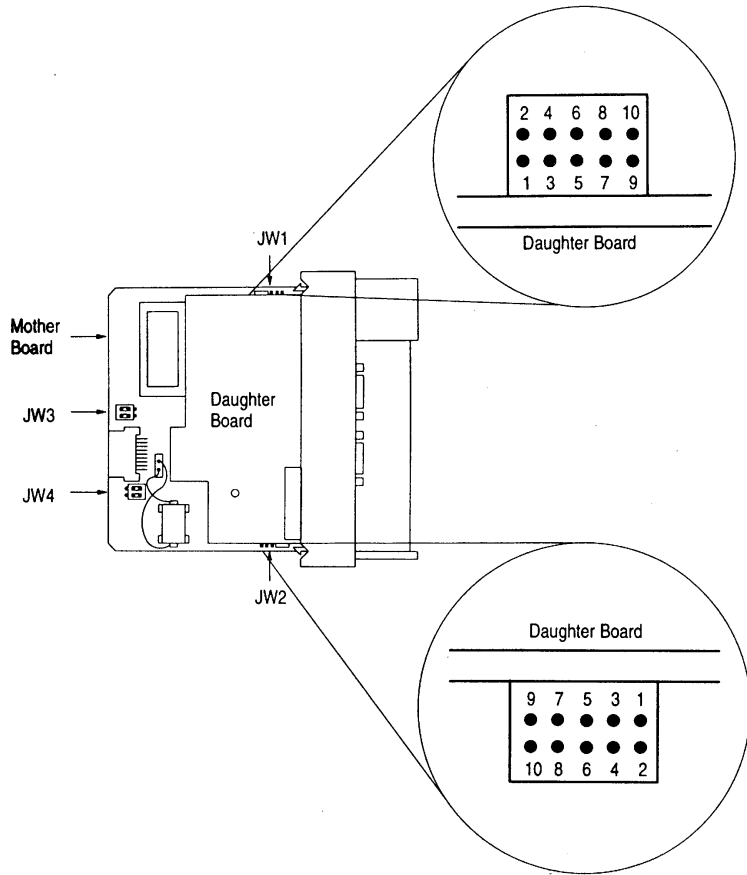
图 C13 Preset Multiple Coils 请求帧

应答

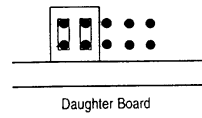
对于一个功能 16 请求的一般应答是重复讯息，包含地址，功能代码，起始地址和寄存器的数目。

ADR	FUNC	H.O. ADDR	L.O. ADDR	QUANTITY		ERROR CHECK FIELD
11	10	00	87	00	02	56

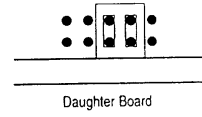
图 C14 Preset Multiple Registers 应答帧



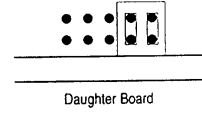
Jumper JW1 Settings



RS-232

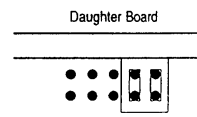


**RS-422
4-wire**

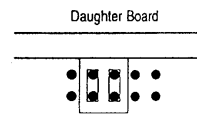


**RS-485
2-wire**

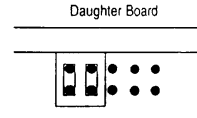
Jumper JW2 Settings



RS-232



**RS-422
4-wire**



**RS-485
2-wire**

E 产品修订历史

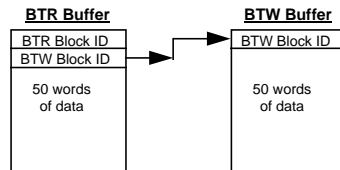
- 05/24/95 Revision 1.40 - 2
- 产品首次发布
- 07/04/95 Revision 1.41 - 3
- 修正大于 8 个位的 FC15 写命令。
- 添加端口上循环操作的支持
- 07/23/95 Revision 1.42 - 4
- 修正在多主站模式下，可能关闭所有端口的问题
- 09/01/96 Revision 1.44
- 更改 RTS 延迟支持字数值，而不再是字节数值。现在范围可以到 65534。
- 重新连接 1.42，支持 1771-DB 固件升级
- 添加主站端口的事件驱动写能力
- 添加从站端口的 pass-through 模式
- 支持浮点数变量类型 (功能代码 3, 6, 16，寄存器地址大于 47001)
- 在帧的起始阶段集成更优良的抗噪逻辑
- 添加支持 > 255 字符的缓冲，以满足 ASCII 模式的需要 s
- 添加 6 个从站地址的主站-从站路由能力
- 添加读块开始和写块开始设置数值
- 在主站模式下，3 次重试后仅返回错误 8
- 12/31/96 Revision 2.0 发布

F 读，写和命令块计数数值使用

作为设置的过程，用户可以在通讯设置数据块中设置一些参数来调整模块和 PLC/SLC 之间的数据传输

概要

如本手册的第 4 和第 5 章所述，BTR 缓冲中包含 BTR 和 BTW 块 ID 号。BTR 块 ID 用来鉴别数据内容，而 BTW 块 ID 号被梯形逻辑用来判断该移动哪些数据到模块。可以用图的形式表达这种关系，如下：



设置参数

对于传输数据非常重要的 3 个参数是：s

Read 块数目：此数值代表有多少个 50 字长的数据块要从 MCM 模块传送到处理器。从模块返回的块从输入到 Read_Block_Start 寄存器内的数值开始，并从此递增

Write 块数目：此数值代表有多少个 50 字长的数据块要从处理器传送到 MCM 模块。

Command 块数目：T 此数值代表有多少个 50 字长的命令块要从处理器传送到 MCM 模块。

模块使用这些数值来判断如何产生 BTW 和 BTR 块 ID 代码。模块提供的部分功能就是控制 BTR 和 BTW 块 ID 代码的增加和复位。这样做的原因是尽量减少用以服务模块的梯形逻辑。

模块运作

作为输入参数结果，模块会在 BTW 和 BTR 块的范围内循环。这个循环会依据下面的等式：

BTW 块 ID

```

if ( BTW Block ID >= Write_Block_Cnt ) then BTW Block ID = 80
elseif( BTW Block ID >= 80 + Command_Block_Cnt) then BTW Block ID = Write_Block_Start
else BTW block ID = BTW block ID + 1
  
```

BTR 块 ID

```

if ( BTR Block ID >= Read_Block_Cnt ) then BTR Block ID = Read_Block_Start
else BTR block ID = BTR block ID + 1
  
```

举例说明，假设我们有下面这样的设置：

```

Read_Block_Cnt      4
Write_Block_Cnt     1
Command_Block_Cnt   2
Read_Block_Start    1
Write_Block_Start    0
  
```

这样的设置数据会导致下面的块 ID 代码循环：

BTW 块 ID	BTR 块 ID
0	1
80	2
81	3
0	4
80	1
81	2
0	3

请注意在 BTW 和 BTR 块 ID 的绝对数值间没有固定的联系。

G 样例梯形逻辑程序

我们提供了下面这些样例程序来帮助您更有效的开发您的应用。在一本名叫样例梯形逻辑的说明书中提供了这些例子（有两本这样的手册，一本是 PLC，一本是 SLC 样例程序）

从站模式样例

样例 #1：从站模式带 Pass-Thru – 最低设置

MCM5EX1S	PLC 5
MCM3EX1S	SLC 5/03

样例 #2：从站模式带 Pass-Thru – 扩展应用

MCM5EX2S	PLC 5
MCM3EX2S	SLC 5/03

主站模式样例

样例#1：主站模式 – 基础应用

MCM5EX1M	PLC 5
MCM3EX1M	SLC 5/03

样例#2：主站模式带命令控制

MCM5EX2M	PLC 5
MCM3EX2M	SLC 5/03