

# **JX-300XP**

多串口多协议通讯卡

**XP248**

使用手册

**IM21H75-C**






## 声 明

- 严禁转载本手册的部分或全部内容。
- 在不经预告和联系的情况下，本手册的内容有可能发生变更，请谅解。
- 本手册所记载的内容，不排除有误记或遗漏的可能性。如对本手册内容有疑问，请与我公司联系，联系邮箱：SMS@supcon.com。

## 商 标














中控、SUPCON、SPlant、Webfield、ESP-iSYS、MultiF、InScan、SupField 以上商标或标识均是浙江中控技术股份有限公司已经注册或已经申请注册或正在使用的商标和标识，拥有以上商标的所有权，未经浙江中控技术股份有限公司的书面授权，任何个人及企业不得擅自使用上述商标，对于非法使用我司商标的行为，我司将保留依法追究行为人及企业的法律责任的权利。

## 文档标志符定义

	<p><b>警告：</b>标示有可能导致人身伤亡或设备损坏的信息。</p> <p><b>WARNING:</b> Indicates information that a potentially hazardous situation which, if not avoided, could result in serious injury or death.</p>
	<p><b>电击危险：</b>标示有可能产生电击危险的信息。</p> <p><b>RISK OF ELECTRICAL SHOCK:</b> Indicates information that Potential shock hazard where HAZARDOUS LIVE voltages greater than 30V RMS, 42.4V peak, or 60V DC may be accessible.</p>
	<p><b>防止静电：</b>标示防止静电损坏设备的信息。</p> <p><b>ESD HAZARD:</b> Indicates information that Danger of an electro-static discharge to which equipment may be sensitive. Observe precautions for handling electrostatic sensitive devices</p>
	<p><b>注意：</b>提醒需要特别注意的信息。</p> <p><b>ATTENTION:</b> Identifies information that requires special consideration.</p>
	<p><b>提示：</b>标记对用户的建议或提示。</p> <p><b>TIP:</b> Identifies advice or hints for the user.</p>

# 设备安全警示标志

下表列出了在设备中使用的安全警示标志符号及描述。

编号	符号	描述
1		直流（电）。文档可使用缩写 DC Direct current
2		交流（电）。文档可使用缩写 AC Alternating current
3		工作接地端子 Ground（Earth）terminal
4		保护接地端子 Protective earth（ground）terminal
5		抗干扰接地端子 Reference ground （Earth）terminal
6		机架或机箱端子。 Frame or chasis
7		等电位。 Equipotentiality
8		通（电源）。 On（power）
9		断（电源）。 Off（power）
10		警告，电击危险。 Caution,risk of electric shock
11		警告，热表面。 Caution,hot surface
12		警告，危险。 Caution,risk of danger
13		静电敏感器件（ESD） Electrostatic sensitive devices。

# 目 录

多串口多协议通讯卡XP248（V3.0） .....	1
1 基本说明 .....	1
1.1 性能说明 .....	1
1.2 网络结构 .....	2
2 硬件使用说明 .....	2
2.1 结构简图 .....	2
2.2 面板指示灯说明 .....	3
2.3 网络连接 .....	3
2.4 地址跳线 .....	3
2.5 掉电保护 .....	4
2.5.1 电池装卸说明 .....	4
2.6 通讯方式跳线 .....	5
2.7 卡件安装 .....	5
2.8 冗余方式 .....	5
3 组态配置 .....	7
3.1 名词解释 .....	8
3.2 SControl内置的XP248 功能块说明 .....	8
3.2.1 GW_SETCOM功能块 .....	8
3.2.2 GW_MODBUS_RTU功能块 .....	9
3.2.3 GW_MODBUS_SLAVE功能块 .....	10
3.2.4 GW_HOSTLINK功能块 .....	10
3.2.5 GW_GETCOMINFO功能块 .....	11
3.2.6 GW_GETCMDINFO功能块 .....	12
3.2.7 GW_GETBYTE功能块 .....	12
3.2.8 GW_GETBOOL功能块 .....	12
3.2.9 GW_GETUINT功能块 .....	13
3.2.10 GW_GETULONG功能块 .....	13
3.2.11 GW_GETFLOAT功能块 .....	13
3.2.12 GW_GETMORE功能块 .....	14
3.2.13 GW_SETBYTE功能块 .....	14
3.2.14 GW_SETBOOL功能块 .....	14
3.2.15 GW_SETUINT功能块 .....	15
3.2.16 GW_SETULONG功能块 .....	15
3.2.17 GW_SETFLOAT功能块 .....	15
3.2.18 GW_SETMORE功能块 .....	16
3.2.19 GW_SNDMSG功能块 .....	16
3.2.20 GW_RCVMSG功能块 .....	16

3.2.21 GW_SNDRCV功能块 .....	17
4 自定义协议编程 .....	17
5 工程应用说明举例 .....	20
5.1 XP248 与SCnet II 网络接口 .....	20
5.2 通信协议 .....	20
5.3 数据采集周期 .....	21
5.4 命令执行时间以及Cycle设置说明 .....	21
5.5 单块XP248 可连接设备数量 .....	21
5.6 使用Modbus进行通讯的组态示例 .....	22
5.7 运行调试 .....	22
5.8 编写自定义协议 .....	23
5.9 XP248 使用步骤 .....	25
5.10 实例一MODBUS主机通讯模式 .....	26
5.10.1 实例情况及分析 .....	26
5.10.2 XP248 卡的硬件设置 .....	26
5.10.3 和PLC的串口连接。 .....	26
5.10.4 组态软件设置 .....	27
5.10.5 SCControl软件通讯组态 .....	27
5.10.6 XP248 卡保存编译并且下载 .....	31
5.11 实例二MODBUS从机通讯模式 .....	32
5.11.1 实例情况及分析 .....	32
5.12 实例三自定义协议主机通讯模式 .....	33
5.12.1 实现自定义通讯条件 .....	33
5.12.2 实例情况和分析 .....	34
5.12.3 组态软件设置 .....	34
5.12.4 SCControl软件通讯组态 .....	34
5.13 应用注意事项 .....	38
6 资料版本说明 .....	39

# 多串口多协议通讯卡 XP248

## 1 基本说明

XP248 多串口多协议通讯卡（亦称网关卡）是 DCS 系统与其它智能设备（如 PLC、变频器、称重仪表等）互连的网间连接设备，是 SCnet II 网络节点之一，在 SCnet II 网络中处于与主控制卡同等的地位。其功能是将用户智能系统的数据通过通讯的方式连入 DCS 系统中，通过 SCnet II 网络实现数据在 DCS 系统中的共享。

### 1.1 性能说明

- XP248 支持 Modbus 协议、HostLink 协议以及自定义通讯协议。支持 Modbus 协议的主机模式和从机模式。通过 SCControl 功能块实现通讯组态。
- XP248 通讯卡支持 4 路串口的并发工作，每路串口支持 RS-232 和 RS-485 两种通讯方式。4 个串口可同时运行不同的协议。每一串口可以挂接的设备数量由运行的协议决定，但最多不超过 32 个。
- XP248 具备通道冗余功能及卡件冗余功能，四路串口 COM0-COM1，COM2-COM3 可以配置为互为冗余的串行通道，并可配合卡件冗余功能实现多种冗余方案。
- 通讯波特率支持（1200~19200）bps，数据位（5~8）位，停止位（1~2）位，校验方式：无校验、偶校验、奇校验、空格校验、标志校验。
- 安装方式：按 I/O 卡件安装方式安装于机柜的机笼中，占用两个 I/O 槽位；
- 接线方式：接线端子；
- 输入电压：（5.0~5.8）V；功耗：< 5W；
- 供电方式：机柜内 5V 供电。
- SCnet II 通讯：100M 以太网通讯，冗余配置（SCnet II A，SCnet II B）。支持与操作员站、服务器的数据交互以及与 SCnet II 网络其他控制站（IP 地址范围 2~62）的站间数据交互。

## 1.2 网络结构

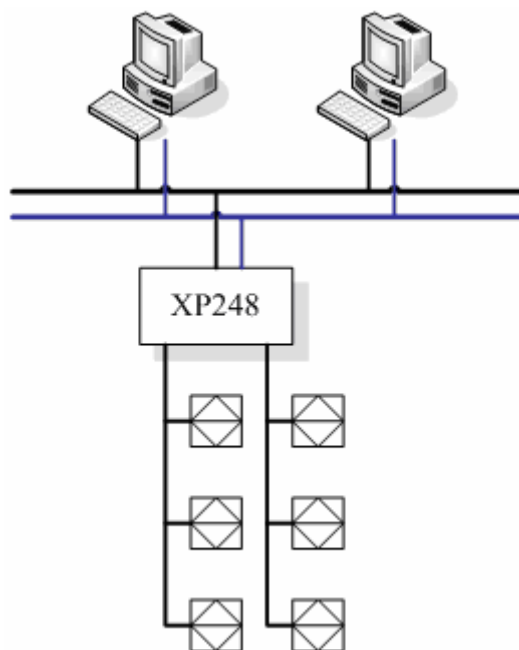


图 1-1 XP248 网络结构

## 2 硬件使用说明

### 2.1 结构简图

XP248 卡件的外观如下图所示（尺寸：187mm×145mm）：

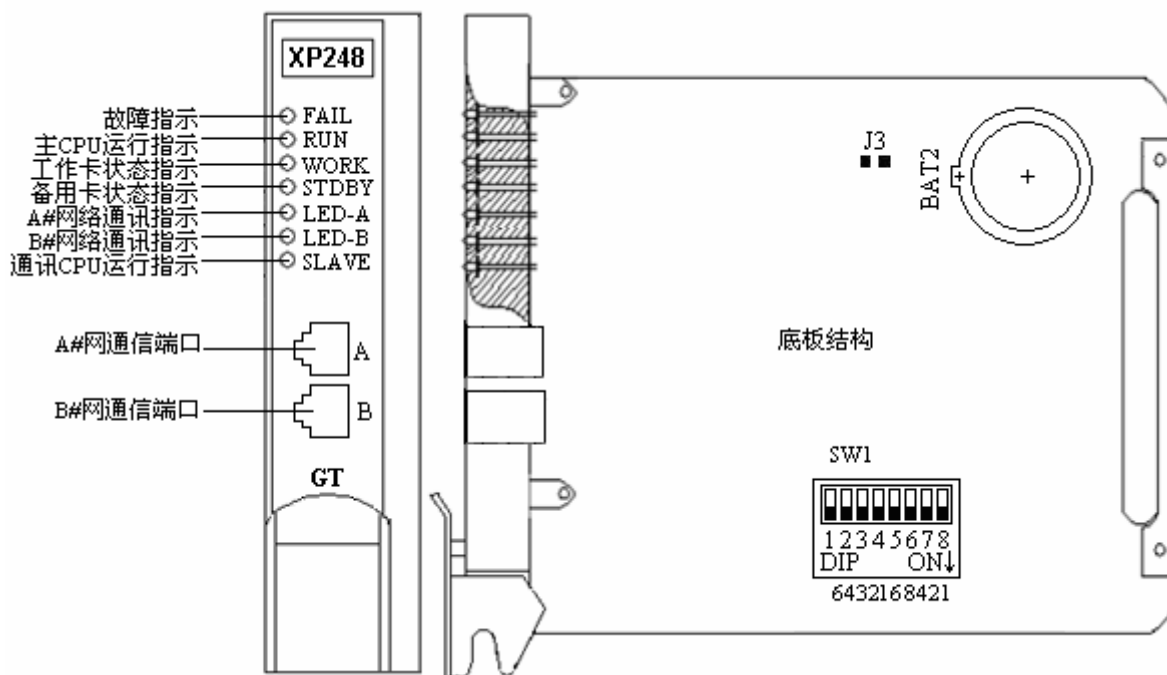


图 2-1 XP248 底板结构图

- SW1：卡件地址拨码开关；

- J3：掉电保护跳线。插上短路块，选择掉电保护功能。
- BAT2：备用电池，型号 CR2032。

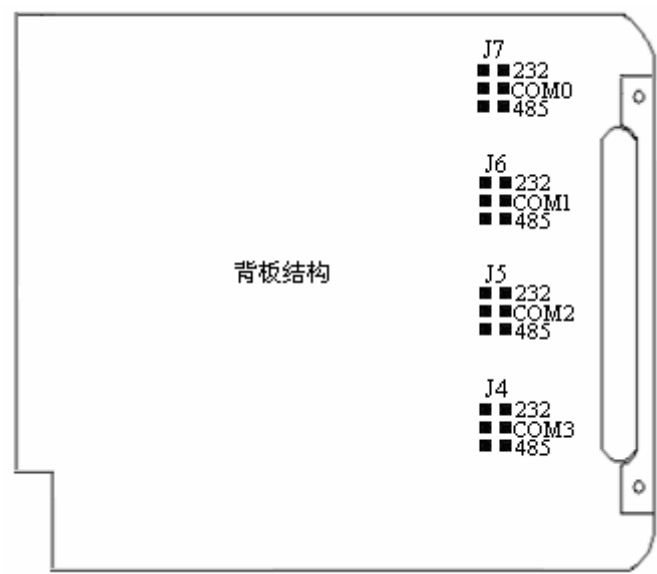


图 2-2 XP248 背板结构图

2.2 面板指示灯说明

如图 2-1 所示XP248 卡件具有 7 个LED指示灯，各指示灯说明如下表所示。

表 2-1 面板指示灯列表

LED 指示灯		常灭	闪烁	常亮
FAIL（红）	故障指示	正常	单网故障	无组态；无监控；网络交错或全断
RUN（绿）	主 CPU 运行指示灯	主 CPU 运行故障	主 CPU 运行正常	—
WORK（绿）	工作卡状态指示灯	此卡为备用卡	—	此卡为工作卡
STDBY（绿）	备用卡状态指示灯	此卡为工作卡	—	此卡为备用卡
LED-A（绿）	A#网络通讯指示灯	网络已断开	网络已连接	—
LED-B（绿）	B#网络通讯指示灯			
SLAVE（绿）	通讯 CPU 运行指示灯	通讯 CPU 运行故障	通讯 CPU 运行正常	—

2.3 网络连接

在 XP248 的面板上有两个互为冗余的 SCnet II 网络端口，分别为 SCnet II A 和 SCnet II B：

- SCnet II A：SCnet II 通讯端口 A，与冗余网络 SCnet II 的 A#网络相连。
- SCnet II B：SCnet II 通讯端口 B，与冗余网络 SCnet II 的 B#网络相连。

2.4 地址跳线

如图 2-1 所示，XP248 底板下方SW1 为拨码地址开关，对应卡件在SCnet II 网络中的地址。在



JX-300XP系统中，本卡件拨码地址范围为 2~63，即拨码 1 和 2 应设置为OFF状态，拨码开关状态与地址对应关系如下表所示。冗余配置时，互为冗余的两块卡件拨码地址应为 2n和 2n+1（n为 1~31 的整数）；单卡工作时拨码地址应为 2n。

表 2-2 拨码地址列表

拨码开关 SW1								地址
1	2	3	4	5	6	7	8	
OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF	02
OFF	OFF	OFF	OFF	OFF	OFF	ON	ON	03
OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF	04
OFF	OFF	OFF	OFF	OFF	ON	OFF	ON	05
.....	.....	.....	.....	.....	.....	.....	.....	.....
OFF	OFF	ON	ON	ON	ON	ON	ON	63

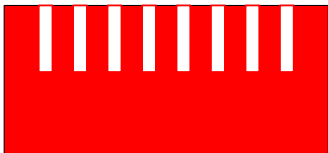


图 2-3 拨码地址开关

2.5 掉电保护

XP248 具有掉电保护功能，可通过底板右上角掉电保护功能选择跳线J3，如 图 2-1 所示，来选择是否启用该功能。

使用掉电保护功能应确保如下几点：

- 1. BAT2 电池座内已安装纽扣电池，电池型号为 CR2032、3V、220mAh 的锂电池。
- 2. 功能选择跳线 J3 处已插上短路块。

2.5.1 电池装卸说明

电池安装

- 1. 戴上防静电手腕；
- 2. 取出 XP248 卡，平放在桌面上；
- 3. 将锂电池正极朝上放入到电池槽中；
- 4. 用螺丝刀按压电池上方，使电池被电池槽中的卡口卡牢。

电池拆卸

- 1. 戴上防静电手腕；
- 2. 取出 XP248 卡，平放在桌面上；
- 3. 用螺丝刀朝外拨动电池槽左边的卡口簧片，直到锂电池从电池槽中弹起；
- 4. 取出锂电池。

## 2.6 通讯方式跳线

XP248 卡件每个通道都支持RS-232 及RS-485 通讯方式，如图 2-2 所示，背板上有 4 个通讯方式跳线，可以对通讯方式进行选择。跳线方式与通讯方式对应关系如下图所示。

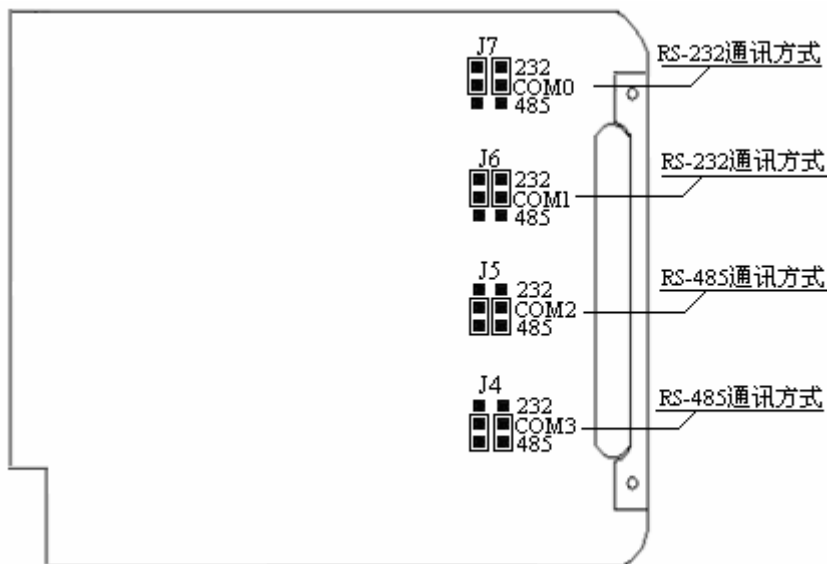


图 2-4 通讯方式选择跳线举例：第 1、2 路为 RS-232 通讯方式，第 3、4 路为 RS-485 通讯方式

## 2.7 卡件安装

XP248 卡件安装于机笼中，占用相邻的两个 I/O 槽位（槽位号为  $2n$ ， $2n+1$ ； $n$  为 0~7 的整数）。

## 2.8 冗余方式

XP248 卡件配合 XP526 端子板使用，支持通道冗余和卡件冗余，因此可以实现多种冗余方案。以下列举其中三种冗余方式，具体接线方法见《XP526 使用手册》。

### 单卡通道冗余

XP248 卡件的COM0 与COM1 通道、COM2 与COM3 通道可配置成互为冗余通道（配置方法见“组态配置”一节），冗余方式如图 2-5 所示。

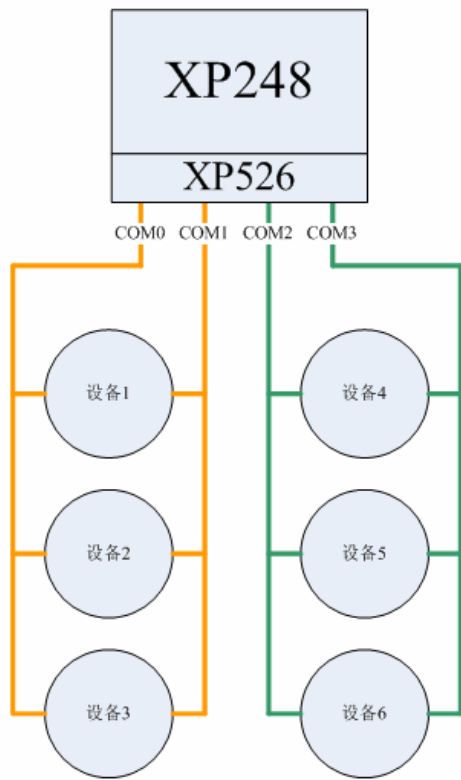


图 2-5 单卡通道冗余

双卡冗余

XP248 卡件提供双卡冗余功能，在主控制卡组态界面将冗余功能选项勾选上即可，如 图 2-6 所示。

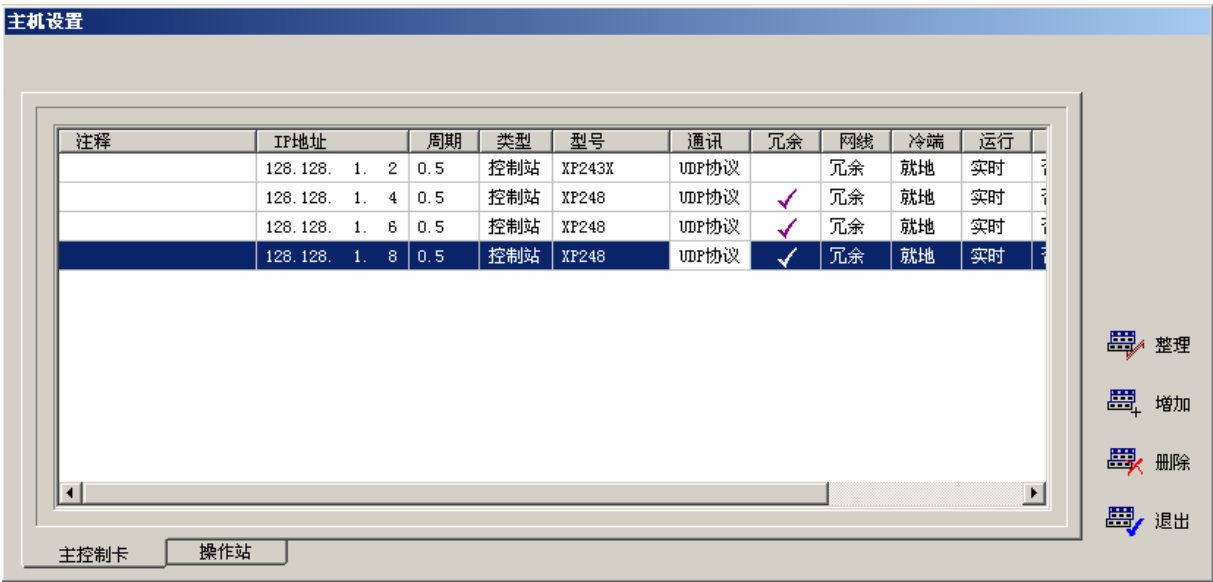


图 2-6 设置双卡冗余功能

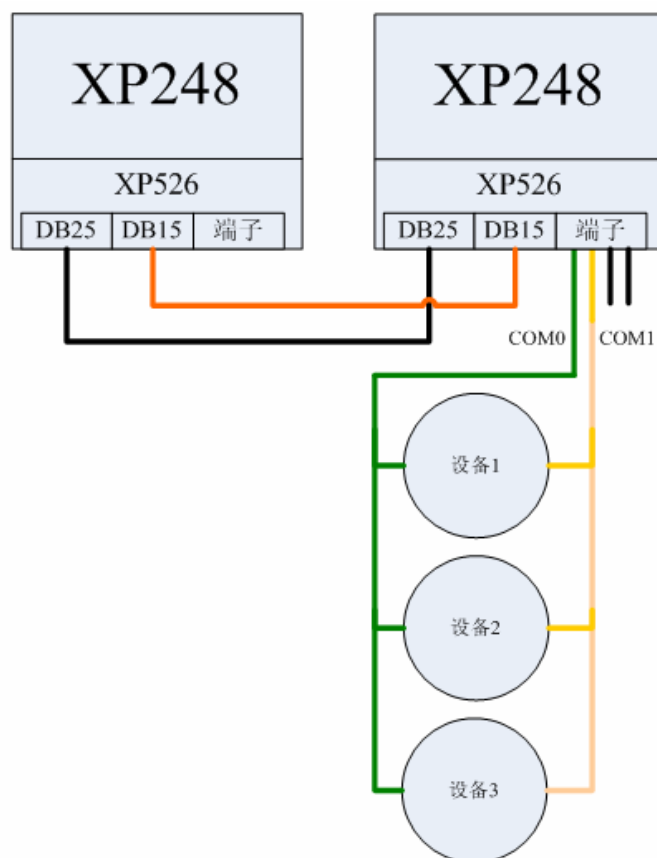


图 2-7 双卡冗余方式

## 双卡通道冗余

在如图 2-5 所示的单卡通道冗余基础上,按如图 2-7 所示方式配置双卡冗余,即可实现双卡通道冗余。

### 3 组态配置

XP248 通过 SCControl 图形化编程软件进行通讯组态。SCControl 软件中已经集成了通讯设置功能块以及 Modbus RTU 功能块、HostLink 功能块等。利用 SCControl 软件提供的数值或逻辑运算功能块，XP248 可以根据需要将智能模块输出的数据实现复杂的转换。（不支持 SCControl 中的 I/O 功能块以及 PID 控制算法功能块。）

XP248 组态主要由几个部分:

首先是 SCnet 组态。由于 XP248 与主控制卡都挂接在 SCnet II 网络上，所以也占用 SCnet II 网络的 IP 地址。XP248 的组态方法与主控制卡相同，设置 IP 地址（拨码）、控制周期默认为 500ms。卡件冗余方式由用户选择。

其次是自定义位号组态。从下挂设备读出或要写入下挂设备的数据都存放在自定义位号中，XP248 通过这些自定义位号与控制系统的操作员站/服务器进行数据交互。

最后是通讯组态。通讯组态也分为三个部分，一是对串口的通讯参数组态，包括波特率、校验方式等；二是命令组态，包括具体的 Modbus 通讯协议，例如读线圈、写寄存器等；三是读数或置数模块，将命令执行后的数据读到自定义位号或将自定义位号的数据写到命令的数据缓冲区。该部分组态必须按照先组串口，然后组命令，最后组取数或置数模块的顺序进行。请注意，串口设置以

及通讯命令设置模块的输入引脚只在组态时有效，在运行时不可改变，具体请参考模块的相关说明。

另外，如果从智能设备取得的数据需要再进行处理，可以直接在 **SCControl** 软件中利用各种丰富的功能函数进行再次计算。

为方便现场调试，**SCControl** 软件还提供了串口和命令诊断函数进行通讯诊断，通过这些功能块可以在 **SCControl** 软件的调试环境中，获取串口的收、发次数，命令的收、发次数，命令的通讯质量码等信息。

以下两节为 **SCControl** 软件内置的 **XP248** 功能块说明以及文中可能涉及的名词解释。

## 3.1 名词解释

### ● **AltTim**

命令间隔时间，串口设置模块中包含该参数，表示两条命令之间需要插入的等待时间。某些 PLC 或智能设备不允许在执行完一条命令后紧接着执行下一条命令，需要等待一段时间后才能响应。兼顾一般情况，可以将该参数设置为 10ms。设置时间过长将使得命令执行周期加大。

### ● **Cycle**

命令执行周期，命令置模块中包含该参数，表示该命令是否被周期执行以及执行周期为多长。只有周期地执行通讯命令，才能将智能设备中的数据读入到 **XP248** 卡。**Cycle** 的单位是 ms。

读智能设备数据命令。**XP248** 卡按照设置的 **Cycle** 时间周期性地执行命令，当 **Cycle** 等于 0 时，表示该命令不是周期命令，只在组态启动时执行一次。所以一般情况下，该值可以设置为与 **XP248** 卡的控制周期相同。对于某些采样时间没有严格要求的数据命令，放大 **Cycle** 时间，可以在通讯量比较大的场合下，提高其他命令的执行速度。

读写智能设备数据命令。有两种触发方式：一是数据有变化才写（更新写），二是定时写（即周期写）。将 **Cycle** 设置为 0 则表示有变化才写；将 **Cycle** 设置为定时时间则表示定时写方式，但当发现需要下写的数据有更新时，**XP248** 卡仍会立刻执行该条写命令；**XP248** 卡启动时，自动地认为数据没有变化，避免在启动时将不需要的数据写到智能设备中。所以一般情况下，建议配置为更新写，即设置 **Cycle** 为 0。

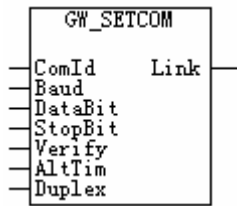
### ● **AckTim**

命令超时时间，单位 ms。设置为 0 时，**XP248** 卡内部自动调整。**XP248** 卡发送命令给智能设备后，某些智能设备并不会立刻回应，而需要等待一定时间后才回应。**XP248** 卡发出命令后，会启动一个超时定时器，定时时间为 **AckTim**，当规定时间到达仍没有收到智能设备的响应，则认为通讯超时，退出本次通讯，本次通讯错误。当用户将 **AckTim** 设置为 0 时，则 **XP248** 卡会在（0~200）ms 内自动调整。当用户将 **AckTim** 设置为非 0 时，按用户实际设置的数据设置从机的应答超时时间。一般情况下，可以设置为 0，让 **XP248** 卡自动修正（为了保证不错过返回的数据，可以把 **AckTim** 数值设置得大一些）。

## 3.2 **SCControl**内置的**XP248** 功能块说明

### 3.2.1 **GW\_SETCOM**功能块

功能块：**GW\_SETCOM**



功能：串口通讯参数设置模块

输入：

- ComId: BYTE （串口号 0~3）
- Baud: DWORD （波特率）
- DataBit: BYTE （数据位 5, 6, 7, 8）
- StopBit: BYTE （停止位 1, 2）
- Verify: BYTE （0: 无校验; 1: 偶校验; 2: 奇校验; 3: SPACE 空格; 4: MARK 标志）

Tim: WORD （两条命令之间的间隔时间，单位 ms），当该串口用于从机模式时，该值应该置为 0。

- Duplex: BOOL （OFF: 非冗余通道 ON: 冗余通道）

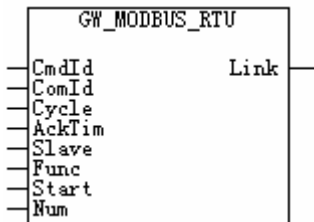
输出：Link: BYTE （用于后续链接，是 ComId 的复制）



注意：  
功能块引脚输入在运行时不能改变。（奇校验和偶校验位跟在一个字节之后，说明一个字节信息的特性；具体传输时，先从低到高位传信息字节，再传校验位）

3.2.2 GW\_MODBUS\_RTU功能块

功能块：GW\_MODBUS\_RTU



功能：MODBUS\_RTU 主机模块

输入：

CmdId: BYTE （命令号 0~255，要求每条命令不能重复。通道冗余情况下，占连续两个命令号，组态时只组偶数命令号）

- ComId: BYTE （串口号 0~3 ）

Cycle: WORD （执行周期，单位 ms）

AckTim: WORD （命令最大的超时时间，单位 ms，一般取 200ms）

Slave: BYTE （Modbus 从机地址）

Func: BYTE （功能号，支持 1, 2, 3, 4, 5, 6, 15, 16 号命令）Modbus 的命令

Start: WORD （数据起始编号 1~65535 ）

Num: WORD （该命令操作的寄存器或线圈等的个数，Modbus 协议对位变量（线圈或输入状态），一次最多可以存取 2040 个；对 16 位寄存器，一次最多可以存取 127 个。具体应用

的实际限制参见 5.9 节)

输出: Link:        BYTE (用于后续链接, 是 CmdId 的复制)



**注意:**  
功能块引脚输入在运行时不能改变。每一条命令的 ComId 都对应一个缓存区, 由此命令发出或收到的数据必然会在此缓存区内保留。串口作为主设备, 其下挂设备的设备号就是 Slave 值。

3.2.3 GW\_MODBUS\_SLAVE功能块

功能块: GW\_MODBUS\_SLAVE



功能: MODBUS\_SLAVE 从机模块

输入:

CmdId:        BYTE (命令号 0~255, 要求每条命令不能重复。通道冗余情况下, 占连续两个命令号, 组态时只组偶数命令号)

ComId:        BYTE (串口号 0~3 )

Add:         BYTE (本机在串口总线的从机地址号)

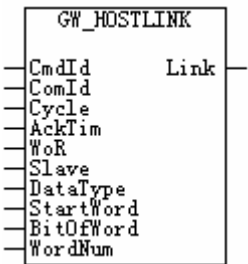
输出: Link:        BYTE (用于后续链接, 是 CmdId 的复制)



**注意:**  
功能块引脚输入在运行时不能改变。Add 值就是此串口作为从设备的设备号, 从机模式的 Add 要与主机方的 Slave 值一致。

3.2.4 GW\_HOSTLINK功能块

功能块: GW\_HOSTLINK



功能: 欧姆龙 PLC 的 HOSTLINK 命令模块 (主机模式)

输入参数:

CmdId:        BYTE (命令号 0~255, 要求每条命令不能重复。通道冗余情况下, 占连续两个命令号, 组态时只组偶数命令号)

ComId:        BYTE (串口号 0~3)

Cycle:        WORD (执行周期, 单位 ms, 等于 0 时为非周期命令, 只执行一次)

AckTim:       WORD (命令最大的超时时间, 单位 ms, 一般取 200ms)

WoR:         BYTE (1: 从 PLC 中读数据; 128: 向 PLC 写数据)

Slave:        BYTE (HOSTLINK 从机地址)

DataType:	BYTE（存取的 PLC 数据类型，相当于功能号）		
测试命令:	0	可读，	WORD（0：不通过；1：通过）
IR 寄存器:	1	可读可写，	WORD
HR 寄存器:	2	可读可写，	WORD
AR 寄存器:	3	可读可写，	WORD
LR 寄存器:	4	可读可写，	WORD
TC 状态:	5	可读可写，	BOOL，（ON=0xFF，OFF=0x00）
DM 数据:	6	可读可写，	WORD
ERROR 错误信息:	7	可读，	WORD
STATUS 信息:	8	可读可写，	WORD
TC 数据:	9	可读可写，	WORD
取消强迫命令:	10	需要用 ST_STARTCMD 手动触发执行	
设置强迫命令:	128 + （IR 或 HR、AR、LR）	只写，	BOOL，ON=0xFF，OFF=0x00

StartWord: WORD（起始 WORD 号）  
BitOfWord: BYTE（WORD 中位标号 0~15, 用于 Forced set/reset 命令）  
WordNum: BYTE（需要操作的 Word 数量）

输出: Link:        BYTE（用于后续链接，是 CmdId 的复制）

HOSTLINK 协议支持对 IR/HR/AR/LR 寄存器的读写；支持对 TC 的状态读写和计数值读写；支持强迫和取消强迫命令。DataType 引脚设置需要读写的数据类型，WoR 引脚设置为 1 表示读，WoR 设置为 128 表示写，例如，当 DataType 为 1，WoR 为 1 表示读 IR 寄存器。

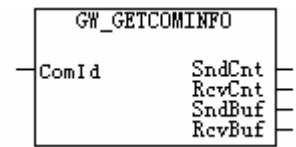
对于 FORCED CANCEL 取消强迫命令，需要用手动方式触发，即组态好一条命令后，用 GW\_STARTCMD（命令号）函数手动触发，上位机可以用变量控制该函数是否执行。



**注意：**  
功能块引脚输入在运行时不能改变。hostlink 功能块传输的数据为 ASCII 码，不是十六进制数。

3.2.5 GW\_GETCOMINFO功能块

功能块: GW\_GETCOMINFO



功能: 获取串口诊断数据

输入:

ComId:        BYTE （串口号 0~3）

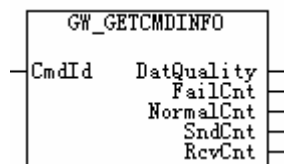
输出:

SndCnt:        WORD （发送次数）  
RcvCnt:        WORD （接收次数）  
SndBuf:        ULONG （发送缓冲区地址）  
RcvBuf:        ULONG （接收缓冲区地址）



### 3.2.6 GW\_GETCMDINFO功能块

功能块: GW\_GETCMDINFO



功能: 获取命令诊断数据

输入:

CmdId: BYTE (命令号 0~255)

输出:

DataQuality: BYTE (数据质量码 1: 正常 0: 异常)

FailCnt: BYTE (通讯连续失败次数)

NormalCnt: BYTE (通讯连续成功次数)

SndCnt: BYTE (发送次数)

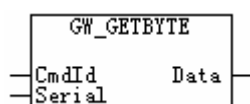
RcvCnt: BYTE (接收次数)

当连续通讯三次都出现故障的情况 ( $\text{FailCnt} \geq 3$ ), 数据质量码 DataQuality 置 0, 表示数据失效 (当前数据是三次通讯前的某次正确数据), 当连续三次通讯正常 ( $\text{NormalCnt} \geq 3$ ), 数据质量码 DataQuality 置 1, 表示数据有效。

特别地, 当该串口配置为冗余串口时, 即便此时该命令在这个串口执行错误 ( $\text{FailCnt} \geq 3$ ), 只要该命令在另一个冗余串口能正常通讯, 则数据质量码仍置 1, 表示当前该命令的数据有效。

### 3.2.7 GW\_GETBYTE功能块

功能块: GW\_GETBYTE



功能: 从命令的数据缓冲区读指定数据

输入:

CmdId: BYTE (命令号 0~255)

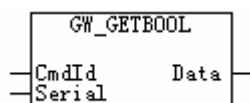
Serial: WORD (欲读取数据在命令的数据块中的 BYTE 类型顺序 (从 0 开始))

输出:

Data: BYTE (读出数据)

### 3.2.8 GW\_GETBOOL功能块

功能块: GW\_GETBOOL



功能: 从命令的数据缓冲区读指定数据

输入:

CmdId: BYTE (命令号 0~255)

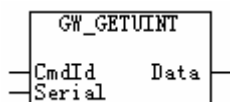
Serial: WORD (欲读取数据在命令的数据块中的 BOOL 类型顺序 (从 0 开始))

输出:

Data: BOOL (读出数据)

### 3.2.9 GW\_GETUINT功能块

功能块: GW\_GETUINT



功能: 从命令的数据缓冲区读指定数据

输入:

CmdId: BYTE (命令号 0~255)

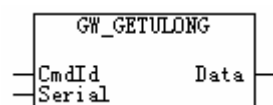
Serial: WORD (欲读取数据在命令的数据块中的 UINT 类型顺序 (从 0 开始))

输出:

Data: UINT (读出数据)

### 3.2.10 GW\_GETULONG功能块

功能块: GW\_GETULONG



功能: 从命令的数据缓冲区读指定数据

输入:

CmdId: BYTE (命令号 0~255)

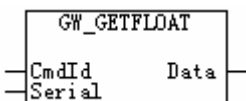
Serial: WORD (欲读取数据在命令的数据块中的 ULONG 类型顺序 (从 0 开始))

输出:

Data: ULONG (读出数据)

### 3.2.11 GW\_GETFLOAT功能块

功能块: GW\_GETFLOAT



功能: 从命令的数据缓冲区读指定数据

输入:

CmdId: BYTE (命令号 0~255)

Serial: WORD (欲读取数据在命令的数据块中的 FLOAT 类型顺序 (从 0 开始))

输出:

Data:        FLOAT（读出数据）

### 3.2.12 GW\_GETMORE功能块

功能块: GW\_GETMORE



功能: 将命令数据缓冲区中一批数据连续写到自定义变量区域, 方便操作。

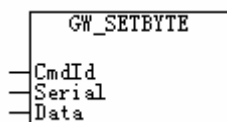
输入:

CmdId:        BYTE（命令号）  
 Serial:       WORD（数据在命令数据缓冲区块中的顺序（从 0 开始））  
 DataType:     BYTE（自定义变量类型: 1: 自定义 1 字节 2: 自定义 2 字节  
                  4: 自定义 4 字节 8: 自定义 8 字节）  
 VarId:        UNIT（自定义变量 ID 号）  
 Num:          UNIT（连续存取个数）

输出: 无

### 3.2.13 GW\_SETBYTE功能块

功能块: GW\_SETBYTE



功能: 设置命令的数据缓冲区中指定数据

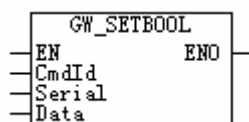
输入:

CmdId:        BYTE（命令号 0~255）  
 Serial:       WORD（欲设置数据在命令的数据块中的字节类型顺序（从 0 开始））  
 Data:         BYTE（欲设置数据）

输出: 无

### 3.2.14 GW\_SETBOOL功能块

功能块: GW\_SETBOOL



功能: 设置命令的数据缓冲区中指定数据

输入:

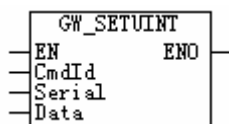
CmdId:        BYTE（命令号 0~255）

Serial: WORD （欲设置数据在命令的数据块中的 BOOL 类型顺序（从 0 开始））  
 Data: BOOL （欲设置数据）

输出：无

### 3.2.15 GW\_SETUINT功能块

功能块：GW\_SETUINT



功能：设置命令的数据缓冲区中指定数据

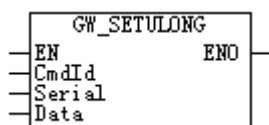
输入：

CmdId: BYTE （命令号 0~255）  
 Serial: WORD （欲设置数据在命令的数据块中的 UINT 类型顺序（从 0 开始））  
 Data: UINT （欲设置数据）

输出：无

### 3.2.16 GW\_SETULONG功能块

功能块：GW\_SETULONG



功能：设置命令的数据缓冲区中指定数据

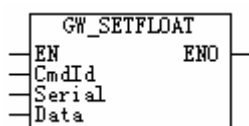
输入：

CmdId: BYTE （命令号 0~255）  
 Serial: WORD （欲设置数据在命令的数据块中的 ULONG 类型顺序（从 0 开始））  
 Data: ULONG （欲设置数据）

输出：无

### 3.2.17 GW\_SETFLOAT功能块

功能块：GW\_SETFLOAT



功能：设置命令的数据缓冲区中指定数据

输入：

CmdId: BYTE （命令号 0~255）  
 Serial: WORD （欲设置数据在命令的数据块中的 FLOAT 类型顺序（从 0 开始））  
 Data: FLOAT （欲设置数据）

输出：无

### 3.2.18 GW\_SETMORE功能块

功能块：GW\_SETMORE



功能：将自定义变量区域中一批数据写到命令数据缓冲区

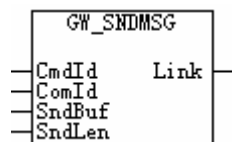
输入：

CmdId: BYTE (命令号)  
 Serial: WORD (数据在命令数据缓冲区块中的顺序 (从 0 开始))  
 DataType: BYTE (自定义变量类型：1: 自定义 1 字节 2: 自定义 2 字节  
 4: 自定义 4 字节；8: 自定义 8 字节。  
 VarId: UNIT (自定义变量 ID 号)  
 Num: UNIT (连续存取个数)

输出：无

### 3.2.19 GW\_SNDMSG功能块

功能块：GW\_SNDMSG



功能：串口发送命令。将 SndBuf 指定的发送缓冲区地址和 SndLen 长度数据发送出去。

输入：

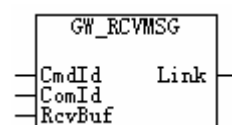
CmdId: BYTE (命令号 0~255)  
 ComId: BYTE (串口号 0~3)  
 SndBuf: ULONG (发送缓冲区指针，一般由 GW\_DEFSNDBUF (UINT Size) 定义函数定义)

SndLen: UINT (发送长度)

输出：Link: BYTE (用于后续链接，是 CmdId 的复制)

### 3.2.20 GW\_RCVMSG功能块

功能块：GW\_RCVMSG



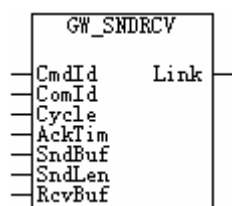
功能：串口接收命令。将收到的一帧数据存放到 RcvBuf 指定的缓冲区地址。

**输入:**

CmdId: BYTE (命令号 0~255)

ComId: BYTE (串口号 0~3)

RcvBuf: ULONG (接收缓冲区指针, 一般由 GW\_DEFRCVBUF (UINT Size) 定义函数定义)

**输出:** Link: BYTE (用于后续链接, 是 CmdId 的复制)**注意:** 功能块引脚输入在运行时不能改变。**3.2.21 GW\_SNDRCV功能块****功能块:** GW\_SNDRCV**功能:** 实现串口的发送和接收功能。首先将 SndBuf 缓冲区的数据发送出去后, 然后将接收到的数据存放到 RcvBuf 缓冲区。**输入:**

CmdId: BYTE (命令号 0~255, 要求每条命令不能重复。通道冗余情况下, 占连续两个命令号, 组态时只组偶数命令号)

ComId: BYTE (串口号 0~3)

Cycle: UINT (执行周期, 单位 ms, 等于 0 时为非周期命令, 只执行一次)

AckTim: UINT (命令最大的超时时间, 单位 ms, 一般取 100ms)

SndBuf: ULONG (发送缓冲区指针)

SndLen: UINT (发送长度)

RcvBuf: ULONG (接收缓冲区指针)

**输出:** Link: BYTE (用于后续链接, 是 CmdId 的复制)**注意:** 功能块引脚输入在运行时不能改变。**4 自定义协议编程**

XP248 提供标准的通讯协议, 例如 Modbus RTU、HostLink 等, 基本可以支持大部分的通讯应用, 但是对于某些采用了非标准通讯协议的特殊设备, 则需要利用 SCControl 软件提供的 ST 语言对 XP248 进行通讯协议编程, 以适应设备的通讯协议。

XP248 提供了 GW\_SNDMSG ()、GW\_RCVMSG ()、GW\_SNDRCV () 三个功能块实现串口的发送、接收、发送/接收功能。这些通讯功能块的基本作用是将指定的数据串发送出去, 并接收来自智能设备的返回数据串。数据串的具体内容由用户在 ST 语言中进行读写。

发送和接收的数据串需要存储在数组中, 由于 XP248 的通讯组态是在 SCControl 软件中进行, 而 SCControl 功能块的引脚不支持数组作为参数输入。所以 XP248 提供了 GW\_DEFSNDBUF(Size)、GW\_DEFRCVBUF (Size) 两个函数为每条命令定义发送和接收缓冲区, 缓冲区大小由 Size 决定。缓冲区一旦创建后将不能被修改, 除非再次下载组态。两个函数返回一个 ULONG 类型的数据用来

指向创建的缓冲区，称为缓冲区指针。

XP248 提供 GW\_WRITEBUF (MsgBuf, Serial, Char) 和 GW\_READBUF (MsgBuf, Serial) 两个函数用来对创建的缓冲区进行读或写操作。其中，MsgBuf 为缓冲区的指针，Serial 是缓冲区中的数据位置，对缓冲区的读写操作不会越过缓冲区边界。

GW\_SNDMSG ()、GW\_RCVMSG () 和 GW\_SNDRCV () 三个功能块其实与 Modbus 功能块类似，只是定义了一条通讯任务，具体的通讯操作并不在这些函数中实现，而是由 XP248 卡内部的调度程序完成。

XP248 提供了 GW\_GETRCVLEN (CmdId) 函数用来获取接收到的数据长度；GW\_GETRCVMSG (CmdId, MsgBuf) 函数将接收到的数据拷贝到 MsgBuf 指向的缓冲区。

此外，XP248 提供一些辅助函数，例如：GW\_CRC16 (MsgBuf, Len) 对数据串进行 CRC 检验；GW\_STARTCMD (CmdId) 函数强制执行一条命令；GW\_FIRSTRUN () 判断 XP248 卡当前是否是启动组态后的第一次运行。

自定义协议目前暂不支持冗余。

下面是以上函数的详细说明：

/\*\*\*\*\*\*

功能块： GW\_DEFSNDBUF

描述：定义命令的发送缓冲区

输入：

Size:            UINT （缓冲区大小，最大不超过 1024）

输出：            ULONG （缓冲区指针）

\*\*\*\*\*/

ULONG GW\_DEFSNDBUF( UINT Size )

/\*\*\*\*\*\*

功能块： GW\_DEFRCVBUF

描述：定义命令的接收缓冲区

输入：

Size:            UINT （缓冲区大小，最大不超过 1024）

输出：            ULONG （缓冲区指针）

\*\*\*\*\*/

ULONG GW\_DEFRCVBUF( UINT Size )

/\*\*\*\*\*\*

功能块： GW\_WRITEBUF

描述：向缓冲区写入一个元素

输入：

MsgBuf:        ULONG （缓冲区指针）

Serial:         UINT （写入位置）

Char:           BYTE （写入数据）

输出：

无，（注：如果 Serial 越限，则放弃操作）

\*\*\*\*\*/

void GW\_WRITEBUF( ULONG MsgBuf, UINT Serial, BYTE Char)

/\*\*\*\*\*\*

功能块: GW\_READBUF

描述: 从缓冲区读入一个元素

输入:

MsgBuf: ULONG (缓冲区指针)

Serial: UINT (写入位置)

输出:

BYTE (返回数组内指定位置数据, 如果 Serial 越限, 则返回 0)

\*\*\*\*\*/

BYTE GW\_READBUF(ULONG MsgBuf, UINT Serial)

/\*\*\*\*\*\*

功能块: GW\_ISCMDFINSHED

功能: 判断命令是否执行完毕

输入:

CmdId: BYTE (命令号 0~255)

输出: BYTE

0:正在执行

1:正常结束

2:异常结束

\*\*\*\*\*/

BYTE GW\_ISCMDFINSHED( BYTE CmdId )

/\*\*\*\*\*\*

功能块: GW\_GETRCVLEN

功能: 取得接收长度

输入:

CmdId: BYTE (命令号 0~255 )

输出: UINT (返回接收到信息的长度)

\*\*\*\*\*/

UINT GW\_GETRCVLEN(BYTE CmdId )

/\*\*\*\*\*\*

功能块: GW\_GETRCVMSG

描述: 拷贝接收到的数据到指定缓冲区

输入:

CmdId: BYTE (命令号 0~255 )

输出: ULONG (接收数据区指针)



```

*****/
ULONG  GW_GETRCVMSG( BYTE CmdId )

/*****
功能块: GW_CRC16
描述: 对缓冲区中指定长度元素进行 CRC 检验
输入:
    MsgBuf:    ULONG (缓冲区指针)
    Len:       WORD (指定长度)
输出:         WORD (16 位的 CRC 校验)
*****/
WORD GW_CRC16(ULONG MsgBuf, WORD Len)

/*****
功能块: GW_STARTCMD
功能: 手动启动命令执行
输入:
    CmdId:     BYTE (命令号 0~255)
输出: 无
*****/
void GW_STARTCMD ( BYTE CmdId )

/*****
功能块: GW_FIRSTRUN
描述: 判断是否首次运行
输入:
    Any :     BYTE 任何数
输出:       BYTE (0: 非首次运行, 1: 首次运行)
*****/
BYTE GW_FIRSTRUN(BYTE Any)

```

## 5 工程应用说明举例

### 5.1 XP248 与SCnet II 网络接口

XP248 通讯卡向上挂接在 SCnet II 网络上, 占用控制站地址, 可以用 SCKey 组态软件对通讯卡进行组态和下载, 操作方法和内部机制与 JX-300XP 系列主控制卡相同, 支持两块 XP248 的冗余配置。网络地址的设置同主控制卡, 拨号范围 2~63, 但地址不能与系统中的主控制卡冲突。

### 5.2 通信协议

XP248 卡支持标准的 Modbus RTU 协议 (主机或从机模式) 以及 HostLink 协议 (主机模式), 其他的通讯协议可以用自定义协议解决。

## 5.3 数据采集周期

XP248 通信周期（智能设备的循环采样的周期）与它所连接的智能设备数量、通信波特率、通信数据量、智能设备的特性、通信协议等都密切相关。其数据采集周期最短为 2ms。

## 5.4 命令执行时间以及Cycle设置说明

一般情况下，可以将命令的 Cycle 设置为与 XP248 组态时的控制周期一致。确定控制周期的依据主要是通讯量以及通讯速率，以下进行简单的说明。

一条命令的执行时间  $T_{cmd} = T1$ （命令发送时间）+  $T2$ （智能设备响应时间）+  $T3$ （智能设备数据回送时间）。

- $T1$ : 以 9600bps, 1 位停止位, 1 位起始位, 数据位 8 位为例, 收发 1 个字节的时间约为 1ms。Modbus 命令的帧头根据不同的命令具有不同的长度, 但一般可以计算为 6 个字节, 再加上 CRC 检验, 共 8 个字节。因此, 命令发送时间至少为 8ms。
- $T2$ : 智能设备的响应时间, 表示自命令发出后, 智能设备开始发送回应数据的时间。不同的设备有不同的相应时间, 一般取 50ms 计算。
- $T3$ : 智能设备数据回送时间。回应的数据帧长度也按 8 个字节计算。Modbus 的线圈占 1 位, 8 个 bit 合成一个字节, 超过部分也占用 1 个字节; Modbus 的寄存器占 2 个字节。对线圈、状态等按 Bit 操作的命令, 占用时间 =  $8 + (n/8 + 1) \text{ (ms)}$ , 其中  $n$  为线圈或状态的个数。寄存器由于占 2 个字节, 所以命令占用时间 =  $8 + 2 * n \text{ (ms)}$ , 其中  $n$  为寄存器个数。注意, 以上估计均以波特率为 9600bps 计算。

XP248 有 4 路串口, 每路串口都可以接若干设备, 对一个设备也可以有多条命令进行操作。XP248 的 4 路串口可以同时工作, 但同一路串口下命令必须逐条执行。因此, 对一路串口来说, 所有命令都执行一遍的时间为:

$$T_c (\text{所有命令执行时间}) = T_{cmd1} + T_{cmd2} + \dots + T_{cmdn} + n * T_{alt}$$

其中,  $T_{cmd1}$  为命令 1 的执行时间,  $T_{cmdn}$  为第  $n$  条命令的执行时间,  $T_{alt}$  是设置的两条命令间隔时间  $AltTim$ 。  $T_c$  则是该串口所有命令执行一遍的最小时间。

因此, 设置命令的执行周期 Cycle 时, 必须大于以上计算的  $T_c$ 。如果设置的 Cycle 时间小于实际运行时间  $T_c$ , 那么, XP248 卡将以最短的  $T_c$  时间运行, 每条命令的周期则被自动拉长, 但所有通讯仍会被执行且运行正常, 只不过每条命令的实际周期执行时间要比设置的大一些。因此, 虽然 Cycle 设置过小不会引起 XP248 卡工作异常, 但是此时的 Cycle 已经失真, 故设置 Cycle 时需要考虑命令的执行时间。

当 Cycle 设置为 0 时, 命令不是周期命令, 不会被周期执行。如果是读命令, 则只在启动组态后执行一次; 如果是写命令, 则需要在数据有变化后才触发命令执行。

## 5.5 单块XP248 可连接设备数量

XP248 在 RS-485 方式下, 每个串口最多可以驱动 32 台设备。但实际连入设备时, 还需考虑数据通讯量和数据刷新时间（命令执行周期）, 设置合适的命令周期, 如果实际的命令周期不能满足需要, 则需要减少挂接的设备, 或将设备移到其他串口。

因此, 与多台智能设备相连时, 每路串口配置的智能设备数量受以下两方面影响: (1) 通讯数据量: 数据量越大, 所带的智能设备越少。(2) 数据刷新时间: 数据刷新时间即命令的执行周期 Cycle,

要求的刷新时间越短，在其它条件相同的情况下所带的智能设备数量就会越少。

5.6 使用Modbus进行通讯的组态示例

XP248 的组态基本上分为三个步骤，首先是用 GW\_SETCOM 功能块对所有使用串口进行组态，然后组态通讯命令，最后用取数模块（如 GW\_GETBOOL）或置数模块（如 GW\_SETBOOL）将自定义位号与命令关联。

对于命令较多的组态，可以将一个串口所有组态（包括串口参数组态、命令组态）安排在一个 FBD 段落中，每个串口一个段落，增强可读性。下面是一个具体的例子。

COM0 组态为不冗余方式，波特率 9600bps，1 位停止位，8 位数据位，无检验。

串口 COM0 下的 0 号命令，读 1#PLC 中从 00001 开始的 16 个触点状态，并将 00003 号触点状态写到自定义变量 BOOL1 中。所有命令的 AckTim 响应超时时间设为 0，表示由 XP248 卡自动设置。

2 号命令则读 10001 号线圈到自定义 1 字节位号 BOOL2，由于 Cycle 设置为 0，故只在组态启动后运行一次。4 号命令将 BOOL3 写到 PLC 的 10001 号线圈，由于 Cycle 设置为 0，故只有在 BOOL3 变化后才将数据写到 PLC。

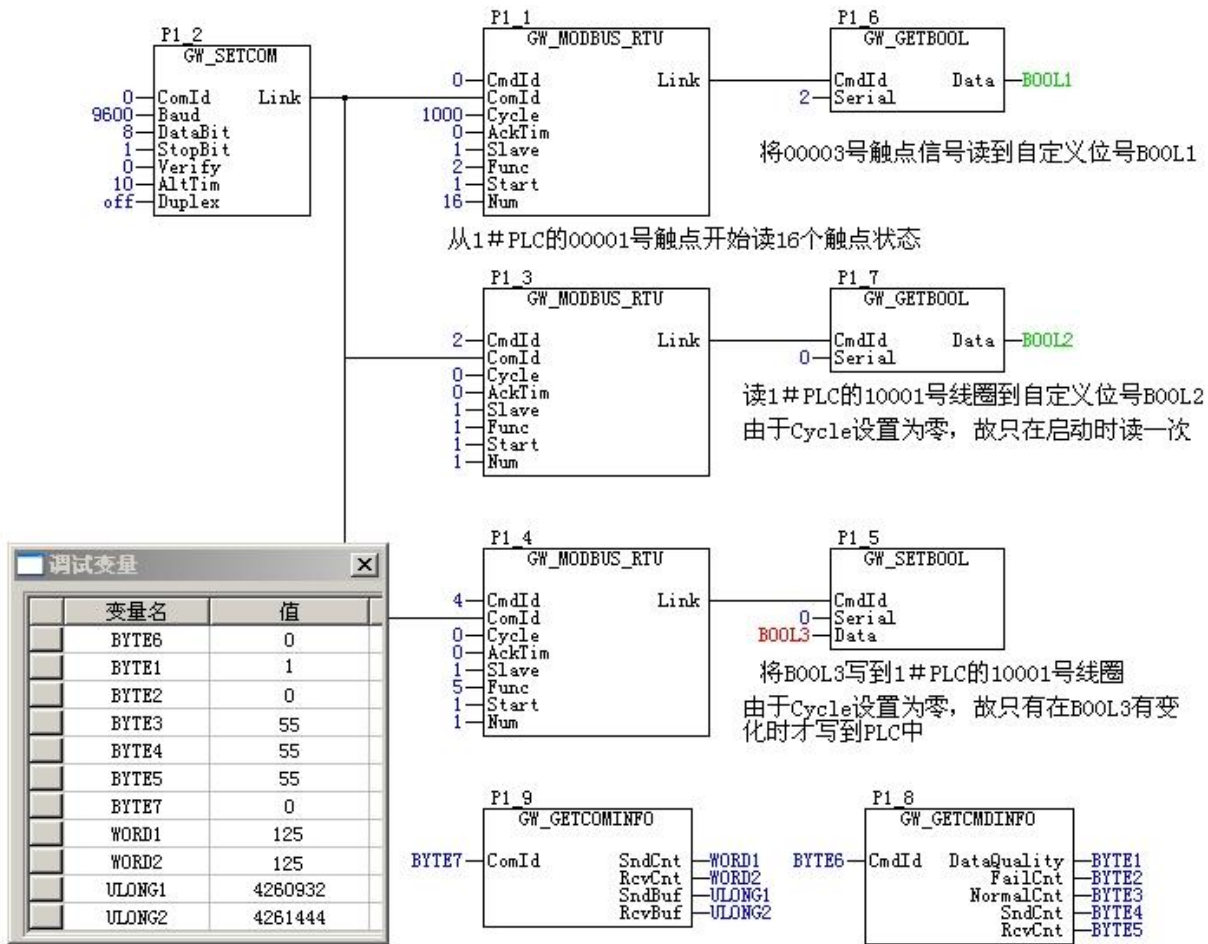


图 5-1 使用 Modbus 进行通讯的组态示例

5.7 运行调试

组态经过编译下载后，可以利用 SCControl 软件的调试功能进行在线调试。工程组态中，对 0

号命令 (CmdId=0) 利用 GW\_GETCMDINFO 功能块进行了诊断。调试变量画面显示了调试数据, 其中 BYTE6 是诊断命令的命令号, BYTE1 是该命令的数据质量码, 0 表示通讯故障数据失效, 1 表示通讯正常。BYTE2 是通讯连续错误次数, BYTE3 是通讯连续正常次数, BYTE4、BYTE5 分别表示该命令发送了几次, 收到了几次回应。

通过以上数据的实时变化, 可以判断通讯的情况。如果 SndCnt 发送次数在不断累加, 则说明该命令被不断执行, 如果此时 RcvCnt 没有变化, 说明命令发出后没有收到回应。如果 RcvCnt 和 FailCnt 不断累加则说明虽然能收到数据, 但收到的是错误的回应数据或命令不能被正常执行, 此时, 需要检查组态是否正确。当 NormalCnt 开始累加到 3 次以后, 表明已经连续 3 次通讯正常, 此时的数据质量码 DataQuality 变为 1, 表示数据有效。

实际应用时, GW\_GETCOMINFO 的引脚 ComId 和 GW\_GETCMDINFO 的引脚 CmdId 可以用一个自定义变量代替, 当在线调试时, 可以修改变量的值使之指向不同的串口或命令, 方便各串口或命令的调试, 整个项目只需要各用一个 GW\_GETCOMINFO 和 GW\_GETCMDINFO 功能块。

需要获取命令的质量码时可以用 GW\_GETCMDINFO 的 DataQuality 引脚获取, 可以将数据和质量码通过 SCnet II 网络传播到网络其他节点, 用于其他高级应用。

## 5.8 编写自定义协议

以下是用 ST 语言编写程序的方法实现 Modbus 协议中的读线寄存器命令。

编写自定义协议, 一般分为两步:

第一步: 编写 ST 段落, 该段落先用 GW\_DEFSNDBUF 和 GW\_DEFRCVBUF 函数定义发送和接收缓冲区 (由于缓冲区大小一旦定义完后不能被改变, 所以缓冲区应该大小合适, 既要适合最大通讯长度又要避免不必要的浪费)。缓冲区定义完后, 可以用 GW\_WRITEBUF 函数写入需要发送的命令数据, 利用 GW\_CRC 函数对输入的数据进行 CRC 检验, 并把校验后的数据写到发送数据的末尾。

将该 ST 段落的输出管脚 (发送缓冲区指针、接收缓冲区指针、发送长度) 连接到 GW\_SNDRCV 功能块的输入引脚。

第二步: 编写 ST 段落, 用 GW\_GETRCVLEN 函数检测串口是否有接收到数据, 如果有, 则用 GW\_GETRCVMSG 函数将接收到的数据拷贝到接收数据缓冲区 RcvBuf。然后对接收缓冲区中的数据进行解析。此处是先用 GW\_CRC 功能块进行校验, 如果校验正确, 则逐个读出缓冲区数据并作为模块的输出。该段落的输出是一个 2 字节模拟量。

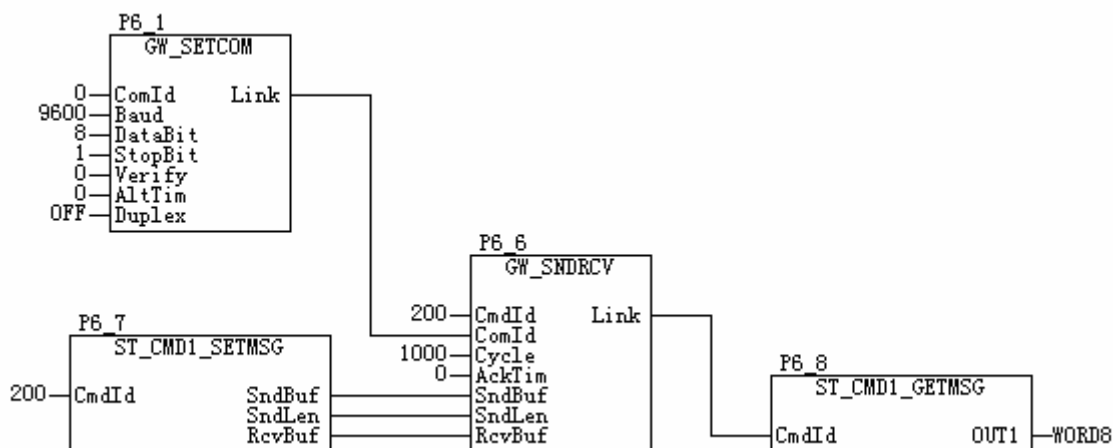


图 5-2 自定义协议的编写示例

以下是 ST 段落 GW\_CMD1\_SETMSG 的代码：

#### FUNCTION\_BLOCK ST\_CMD1\_SETMSG

##### VAR\_INPUT

CmdId: BYTE;

##### END\_VAR

##### VAR\_OUTPUT

SndBuf: ULONG;

SndLen: UINT;

RcvBuf: ULONG;

##### END\_VAR

##### VAR

crc16: WORD;

b1, b2: BYTE;

##### END\_VAR

SndBuf = GW\_DEFSNDBUF(CmdId, 128); (\* 定义发送缓冲区, 128 字节长度 \*)

GW\_WRITEBUF( SndBuf, 0, 1); (\* 写发送缓冲区 0 字节, PLC 地址为 1 \*)

GW\_WRITEBUF( SndBuf, 1, 4); (\* 写发送缓冲区 1 字节, 功能号为 4 \*)

GW\_WRITEBUF( SndBuf, 2, 0); (\* 写发送缓冲区 2 字节, 起始通道高位为 0 \*)

GW\_WRITEBUF( SndBuf, 3, 0); (\* 写发送缓冲区 3 字节, 起始通道低位为 0 \*)

GW\_WRITEBUF( SndBuf, 4, 0); (\* 写发送缓冲区 4 字节, 个数高位为 0 \*)

GW\_WRITEBUF( SndBuf, 5, 1); (\* 写发送缓冲区 5 字节, 个数低位为 1 \*)

crc16 = GW\_CRC16(SndBuf, 6); (\* CRC 校验 \*)

b1 = WORD\_TO\_BYTE(crc16);

b2 = WORD\_TO\_BYTE( crc16/256 );

GW\_WRITEBUF( SndBuf, 6, b1); (\* 末尾加 CRC \*)

GW\_WRITEBUF( SndBuf, 7, b2); (\* 末尾加 CRC \*)

SndLen = 8; (\* 发送长度 \*)

RcvBuf = GW\_DEFRCVBUF(CmdId, 128); (\* 定义接收缓冲区, 长度 128 字节 \*)

##### END\_FUNCTION\_BLOCK

先按照 Modbus 的数据结构，安排每个字节的内容。

进行 CRC 校验，再把数据发到缓冲区。

Modbus slave 是将计算机模拟成 PLC，按照 Modbus 协议的格式解析数据。此模块中，模仿它

的格式，替代 Modbus\_RTU 模块。

以下是 ST 段落 ST\_CMD1\_GETMSG 的代码：

```

FUNCTION_BLOCK ST_CMD1_GETMSG
VAR_INPUT
    CmdId: BYTE;
END_VAR

VAR_OUTPUT
    OUT1:WORD;
END_VAR

VAR
    RcvBuf:ULONG;
    crc16:WORD;
    n,Len:UINT;
    b1,b2,b3,b4:BYTE;
END_VAR

    Len = GW_GETRCULEN(CmdId);           (*获取接收长度*)
    IF Len>0 THEN
        RcvBuf = GW_GETRCUMSG(CmdId);    (*读取接收缓冲区数据*)
        n= Len - 2;
        crc16 = GW_CRC16(RcvBuf, n);     (*CRC校验*)
        b1 = WORD_TO_BYTE(crc16);
        b2 = WORD_TO_BYTE( crc16/256 );
        b3 = GW_READBUF(RcvBuf , n);     (*读缓冲区数据*)
        n = Len - 1;
        b4 = GW_READBUF(RcvBuf,n);
        IF (b3 = b1 ) AND (b2=b4) THEN
            b1 = GW_READBUF(RcvBuf, 3);
            b2 = GW_READBUF(RcvBuf, 4);
            OUT1 = BYTE_TO_WORD(b1)*256 + BYTE_TO_WORD(b2); (*输出*)
        END_IF;
    END_IF;

END_FUNCTION_BLOCK

```

读出缓冲区的数据，进行 CRC 校验。

若正确，则输出值 OUT1。

若没有这个模块，不影响发送和接受数据

自定义协议是针对非标准协议而言，是设备方遵守的特定协议。我们是根据其协议格式组模块，即第几个字节写什么，具体“什么”的含义由设备方所遵守的协议制定。

设备对数据进行解析，用自身所带的协议的格式，而不管数据的内容。

## 5.9 XP248 使用步骤

1. 了解需要连接的设备情况，包括使用的通讯模式是 RS-232 还是 RS-485，使用的协议是 MODBUS RTU、HOSTLINK 还是自定义协议，采用主机通讯还是从机通讯，串口的波特率、数据位、停止位、校验位，需要读取的数据的命令、地址和数量，通讯距离是否满足条件（在波特率为 9600bps 下大概 800 米范围）。
2. 检查 XP248 卡件版本是否正确，地址拨号是否正确。

3. 进行连接是否正确，包括卡件通过 DB25 电缆和端子板的连接、端子板的指定串口和第三方设备串口的连接。
4. 在 SCKey 软件中组态，选择 XP248 卡，并对 XP248 卡进行自定义算法图形编程。
5. 在 SCKey 软件中定义部分自定义 1、2、3 字节，用于保存获得的第三方实时数据或者提供实时数据给第三方设备。
6. 在 SCControl 软件中，首先选择 GM\_SETCOM 模块设置串口信息，该信息和连接的第三方设备串口一致，然后选择需要的功能块进行 MODBUS 或者 HOSTLINK 通讯。
7. 保存组态文件，编译并下载，完成 XP248 卡的组态。

## 5.10 实例一MODBUS主机通讯模式

### 5.10.1 实例情况及分析

某 PLC 提供 RS-485 进行通讯，支持 MODBUS RTU 从站协议，通讯地址为 10，通讯口设置成波特率为 9600bps，数据位 8 位，停止位 1 位，校验方式为偶校验，现在 ECS-100 控制系统需要定时读取 PLC 的线圈地址为 1 开始的 16 个开关量数据和向 PLC 的保持寄存器地址为 3001 开始的区域写入 1 个模拟量数据。

分析：由于 PLC 支持 MODBUS RTU 从站协议，通讯地址为 10，要读取 16 个开关量数据和 1 个模拟量数据。因此，我们配置一块 XP248 卡，设置通讯地址为 6（不能和 DCS 主控制卡地址重复），采用 MODBUS RTU 主站通讯方式进行通讯，获得的实时数据放在 16 个自定义 1 字节变量中，并且假设写入 PLC 的模拟量数据放在 1 个自定义 2 字节变量中。

实施步骤：整个实施可以分成四部份完成，分别为 XP248 卡的硬件设置、和 PLC 的串口连接、组态软件设置和 SCControl 软件通讯组态。

### 5.10.2 XP248 卡的硬件设置

首先对 XP248 卡进行跳线设置。

SW1 拨号设置：背板中标识为 SW1 的是 XP248 卡 ScnetII 网络通讯地址拨号开关，我们根据前面的分析选择通讯地址为 6（不和主控制卡地址重复，在 2~127 范围内任意选择），该地址即是 ScnetII 网络中的 IP 地址。因此将 SW1 中把标识为 6 和 7 的拨号开关向下拨，将其他拨号开关向上拨，设置 XP248 卡的 ScnetII 网络通讯地址为 6。（SW1 中标识为 1 的为高位，标识为 8 的为低位，采用 8421 编码方式。）

### 5.10.3 和 PLC 的串口连接。

XP248 卡需要通过 DB25 线和端子板连接，再通过端子板的串口和 PLC 的串口连接。

- XP248 卡和端子板的连接：将 XP248 卡插在控制柜 I/O 机笼右侧 16 个 I/O 卡插槽中的任一插槽中，占用 2 个槽位。通过 DB25 线和对应的端子板进行连接。
- 端子板和 PLC 的连接：在端子板中找到标识为 COM0 的串口，端子板串口都是通过接线柱的方式提供，由于采用 RS-485 通讯，因此将标有“—”的接线端子和 PLC 通讯口的 FB-（DATA-）连接；将标有“+”的接线端子和 PLC 通讯口的 FB+（DATA+）连接。这样就完成了端子板串口和 PLC 通讯口的连接。如果 PLC 通讯口只提供了 9 芯母头形式，则只要

在 PLC 端配置一个 9 芯公头，将 FB+（DATA+）和 FB-（DATA-）引线分别和端子板接线柱的“+”和“-”连接就可以了。

#### 5.10.4 组态软件设置

XP248 组态主要由几个部分：

- 首先是 SCnet 组态。由于 XP248 与主控制卡相同都是挂接在 SCnet II 网络上，所以也占用 SCnet II 网络的 IP 地址。XP248 的组态方法与主控制卡相同，设置好 IP 地址为 128.128.1.6（设定 XP248 卡地址为 6）、控制周期默认为 500ms。卡件冗余方式为不冗余（XP248 支持两块卡的冗余）。
- 其次是自定义位号组态。从下挂设备读出或写到下挂设备的数据都存放在自定义位号中，XP248 通过这些自定义位号与控制系统的操作员站/服务器进行数据交互。暂定义 16 个自定义 1 字节 TAGA01~TAGA16，1 个自定义 2 字节无符号整数 TAGB1。
- 最后是通讯组态。通讯组态也分为三个部分，一是对串口的通讯参数组态，包括波特率、校验方式等；二是命令组态，包括具体的 Modbus 通讯协议，例如读线圈、写寄存器等；三是读数或取数模块，将命令执行后的数据读到自定义位号或将自定义位号的数据写到命令的数据缓冲区。该部分组态必须按照先组串口，然后组命令，最后组取数或置数模块的顺序进行。具体的通讯组态在下一节中说明。

另外，如果从智能设备取得的数据需要再进行处理，可以直接在 SCControl 中利用各种丰富的功能函数进行再次计算。

#### 5.10.5 SCControl软件通讯组态

完成了前面的 XP248 卡组态和自定义位号的设置，就可以开始进行通讯组态了。通讯组态通过 SCControl 图形编程软件实现，在组态软件中选择“算法”按钮，在“自定义控制算法设置”对话框中选择主控制卡地址为 6（XP248 卡的地址），在图形编程中填入名字后点击“编辑”进行图形化编程，如下所示：

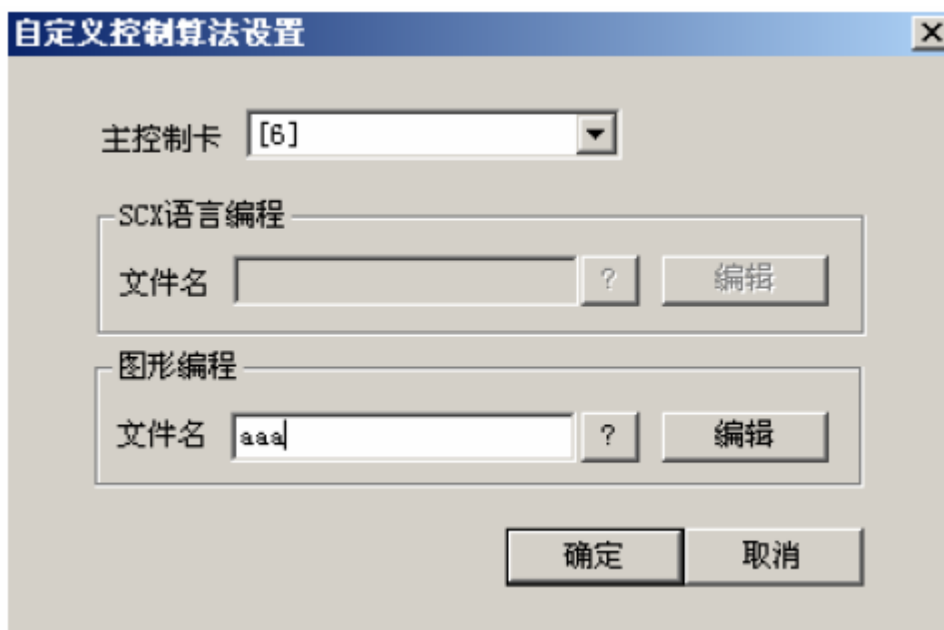


图 5-3 自定义控制算法设置



进入图形编程软件后，新建一个 FBD 段落。在模块选择对话框中选择附加库文件夹下的智能通讯卡模块中的功能块进行通讯组态，具体模块的选择顺序和使用如下：

步骤 1：选择通讯口设置模块，模块名称为 GW\_SETCOM，如下图所示：

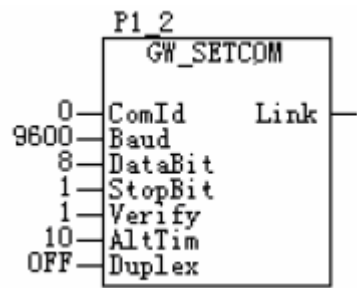


图 5-4 通讯口设置模块

说明：该模块选择了通讯的串口号和串口通讯设置。其中参数 ComId 指定 XP248 通讯的串口号，0~3 分别表示了 COM0~COM3 这 4 个串口。紧接着的 4 个参数分别和连接的设备通讯口设置一致，由于 PLC 的串口已设置成波特率 9600bps，数据位 8 位，停止位 1 位，校验为偶校验。因此，模块的相应参数分别设为 9600、8、1、1，其中参数 Verify 表示校验位，0 为无校验、1 为偶校验、2 为奇校验、3 为置 0、4 为置 1。AltTim 表示 2 条通讯命令间的间隔时间，一般设为 10ms。参数 Duplex 表示该串口是否冗余，ON 表示和 COM1 冗余，OFF 为不冗余。本例串口不冗余通讯。

步骤 2：选择通讯模块，支持 MODBUS 主站通讯、MODBUS 从站通讯、HOSTLINK 主站通讯。本例中是采用了 MODBUS 主站通讯，选择的模块名称为 GW\_MODBUS\_RTU，如下图所示：

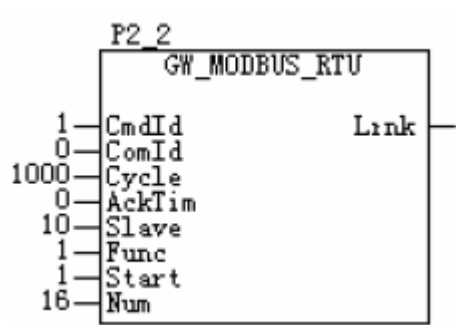


图 5-5 MODBUS 主站通模块

说明：该模块主要负责 XP248 卡和 PLC 的通讯。其中 CmdId 为一个 0~255 的任意数字（该数字和其他通讯模块的 CmdId 不重复），ComId 表示使用的串口号，和 SETCOM 模块中的 ComId 一致，因此也置 0，参数 Cycle 表示本条命令的通讯周期，一般设置成 XP248 卡的扫描周期 500ms，本例设置成 1000ms。参数 AckTim 表示通讯反馈时间，一般取 0 表示程序自动调节。参数 Slave 指定读取 PLC 的通讯地址。参数 Func 表示读取 PLC 的数据类型，其中 1 为线圈、2 为状态、3 为保持寄存器、4 为输入寄存器，由于我们读取线圈数据，所以置 1。参数 Start 表示起始地址，本例中从 1 开始。参数 Num 指定需要操作的位号数，本例为 16 个。

步骤 3：选择读取数据模块，将内存中的实时数据保存到自定义变量中，模块如下所示：

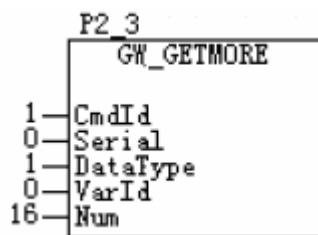


图 5-6 读取数据模块

说明：该模块主要负责将内存中的数据转换成 DCS 的位号变量。其中 CmdId 为步骤 2 中的模块使用的 CmdId。参数 Serial 为需要读取内存中的起始地址，也就是返回的数据包的起始数据序号。参数 DataType 为保存到自定义字节的类型，1 为自定义 1 字节、2 为自定义 2 字节、4 为自定义 4 字节、8 为自定义 8 字节。参数 VarId 为自定义字节的起始序号，在自定义变量的组态画面中可以查看。参数 Num 表示需要保存的自定义字节位号数量。这样就完成了 PLC 线圈实时数据传到 DCS 中显示的工作。

步骤 4：将一个模拟量数据写到 PLC 的寄存器中，在进行写入操作前必须先读取 PLC 寄存器的原始值，否则 XP248 卡运行显示的值和 PLC 中实际值不一致，引起检测错误。读取 PLC 初始值只在程序开始运行时执行一次，而软件没有现成的功能块可以控制只读一次数据，因此我们通过 ST 段落编写一个自定义功能块，延迟数个周期后输出 OFF 使读数据模块停止工作（否则软件将一直读数据，无法完成写数据的操作），ST 段落编写的自定义功能块代码如下，使用时不用修改代码。

```

FUNCTION_BLOCK READFLAG (*自定义模块名*)
VAR_INPUT
    IN1:BYTE;          (*输入管脚，无意义，取任意值*)
END_VAR

VAR_OUTPUT
    CANREAD:BOOL;      (*输出管脚*)
    DELAYCNT:BYTE;      (*输出控制*)
    (*延迟计算，做内部计算使用变量*)
END_VAR

VAR
    FIRSTFLAG:BYTE;    (*程序是否首次运行标志*)
END_VAR

FIRSTFLAG = GW_FIRSTRUN(1); (*本次是否首次运行，1首次；2非首次*)
IF FIRSTFLAG = 1 THEN      (*如果首次运行，置DELAYCNT变量为0*)
    DELAYCNT = 0;
ELSEIF DELAYCNT < 10 THEN  (*如果非首次运行，DELAYCNT每周期加一，直到10个周期*)
    DELAYCNT = DELAYCNT + 1;
END_IF;

IF DELAYCNT < 10 THEN      (*判断在10个周期内，CANREAD脚输出ON，否则输出OFF*)
    CANREAD = ON;
ELSE
    CANREAD = OFF;
END_IF;

END_FUNCTION_BLOCK

```

通过自定义功能块的 CANREAD 脚控制读模块的 EN 使能标志，如下所示：

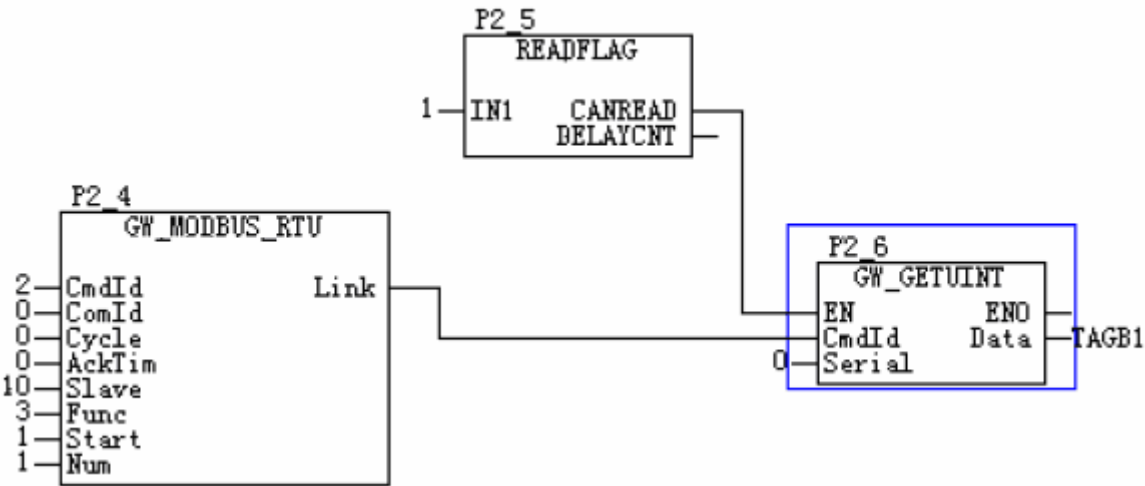


图 5-7 CANREAD 脚控制读模块 EN 使能标志

说明：该模块定义了一个新的 MODBUS RTU 主站通讯模块，命令号 CmdId 取为 2，ComId 串口仍旧使用 COM0，参数 Cycle 不再设置为周期读数据，这里设置成 0，表示第一次运行时才读取数据。其他参数表示读地址为 10 的 PLC，读取 PLC 的一个地址为 1 的寄存器数据，并且把这个数据存到 TAGB1 自定义 2 字节位号中。其中通过自定义模块 READFLAG 控制取数据模块 GW\_GETUINT，使取数据操作在程序刚开始运行时才执行，取 PLC 的初始值。

步骤 5：设置写模块，将 TAGB1 的数据写到 PLC 寄存器中，选择的模块如下所示：

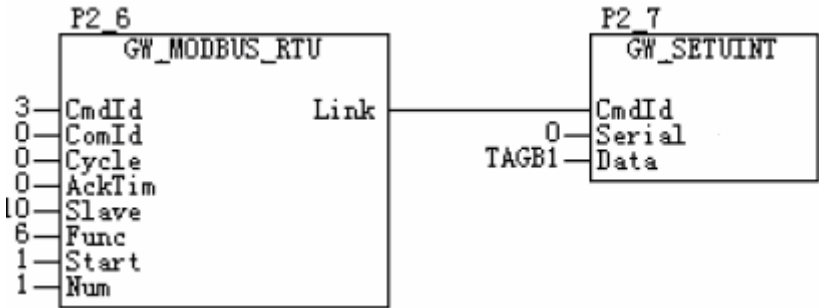


图 5-8 写数据至 PLC 寄存器

说明：该模块再定义了一个新的 MODBUS RTU 主站通讯模块进行写操作，命令号 CmdId 取为 3，ComId 串口仍旧使用 COM0，参数 Cycle 不再设置为周期读数据，这里设置成 0，表示只有 TAGB1 数据改变才进行写通讯操作。其他参数表示读地址为 10 的 PLC，写 PLC 的一个地址为 1 的寄存器数据。整个完整的程序如下图所示：

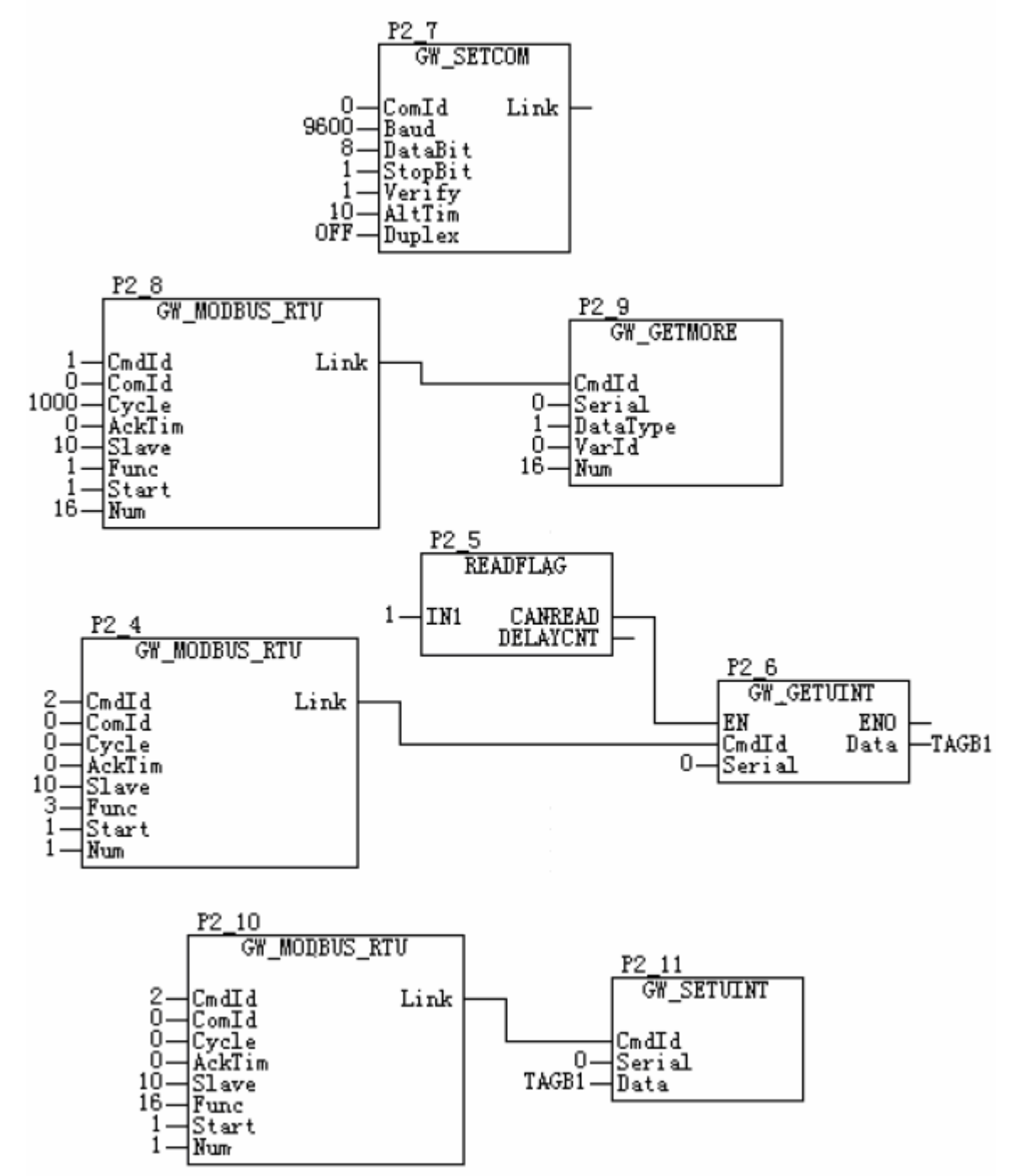


图 5-9 完整程序

5.10.6 XP248 卡保存编译并且下载

通过以上几个步骤后完成对 PLC 数据读取的编程工作。在组态软件中保存工程和全体编译，并下载到 XP248 卡中。完成整个 XP248 卡的设置。

## 5.11 实例二MODBUS从机通讯模式

### 5.11.1 实例情况及分析

某 PLC 提供 RS-485 通讯，支持 MODBUS RTU 主站协议，通讯地址为 10，通讯口设置成波特率为 9600bps，数据位 8 位，停止位 1 位，校验方式为偶校验，假设该 PLC 需要实时获得 ECS-100 控制系统的 8 个模拟量实时数据。

分析：由于 PLC 只支持 MODBUS RTU 主站协议，欲读取 8 个 DCS 的模拟量数据。假设 PLC 的通讯地址为 10。我们配置一块 XP248 卡，设置通讯地址为 6（不能和 DCS 主控制卡地址重复），采用 MODBUS RTU 从站通讯方式进行通讯，欲读取的自定义字节位号通过该自定义位号的序号进行识别，比如 PLC 发了一个 06 01 00 04 00 02 FD BD 的 MODBUS 命令，其中 06 表示 XP248 卡的通讯地址，01 表示读线圈（即读自定义 1 字节数据），00 04 表示读序号为 4 开始的自定义 1 字节数据，00 02 表示读 2 个自定义 1 字节数据（即序号为 4，5 的 1 字节数据），FD BD 为 CRC 校验码。以此类推对其他自定义字节的读写操作。 我们可以根据 PLC 的 MODBUS 命令，将需要的实时数据放置在指定的序号开始的自定义 2 字节位号中，等待 PLC 通过 MODBUS 协议进行读取。

整个实施和上个实例类似，也可以分成四部份完成，分别为 XP248 卡的硬件设置、和 PLC 的串口连接、组态软件设置和 SCControl 软件通讯组态。前面三个步骤可参考上个实施，只需修改 SCControl 软件通讯组态。主站模式和从站模式通过功能块的选择进行区分。

步骤四，通讯组态通过 SCControl 图形编程软件实现，在组态软件中选择“算法”按钮，在自定义控制算法对话框中选择主控制卡地址为 6（XP248 卡的地址），在图形编程中填入名字后点击“编辑”进行图形化编程。进入图形编程软件后，新建一个 FBD 段落。在模块选择对话框中选择附加库文件夹下的智能通讯卡模块中的功能块进行通讯组态，具体模块的选择顺序和使用如下：

选择通讯口设置模块，模块名称为 GW\_SETCOM，如下图所示：

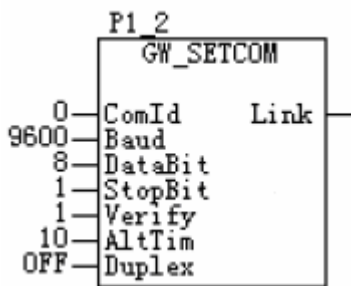


图 5-10 通讯口设置模块

说明：该模块选择了通讯的串口号和串口通讯设置。其中参数 COMID 指定 XP248 通讯的串口号，0~3 分别表示了 COM0~COM3 这 4 个串口。接着 4 个参数分别和连接的设备通讯口设置一致，由于 PLC 的串口已设置成波特率 9600bps，数据位 8 位，停止位 1 位，校验为偶校验。因此，模块的相应参数分别设为 9600、8、1、1，其中参数 VERIFY 表示校验位，其数值表示 0 为无校验、1 为偶校验、2 为奇校验、3 为置 0、4 为置 1。接着 2 个参数中 AltTim 表示 2 条通讯命令间的间隔时间，一般设为 10ms。参数 Duplex 表示该串口是否冗余，ON 表示和 COM1 冗余，OFF 为不冗余。本例串口不冗余通讯。

选择通讯模块，本例中是采用了 MODBUS 从站通讯，选择的模块名称为 GW\_MODBUS\_SLAVE，如下图所示：

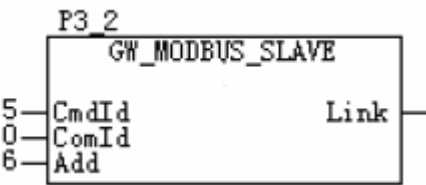


图 5-11 MODBUS 从站通讯模块

说明：该模块主要负责 XP248 卡和 PLC 的通讯。其中 CmdId 为一个 0~255 的任意数字（该数字和其他通讯模块的 CmdId 不重复），ComId 表示使用的串口号，和 SETCOM 模块中的 ComId 一致，因此也置 0。参数 ADD 表示为 XP248 卡的地址，本例为 6。通过该模块，ECS-100 系统把 PLC 需要的实时数据放到自定义 1、2、4 字节中，PLC 就可以通过 MODBUS 协议中的功能码 0x01 读 DCS 的自定义 1 字节变量，0x03 读自定义 2 字节变量，0x0f 写自定义 1 字节变量，0x10 写自定义 2 字节变量。需要通讯的自定义实时数据的序号需要和 PLC 发送的命令中起始地址和数量一致。本例中如果 PLC 要读取序号为 6 开始的 8 个自定义 2 字节数据，则 PLC 会发送如下命令格式 06 03 00 06 00 08 A5 BA，其中 06 表示 XP248 卡通讯地址，03 表示读自定义 2 字节数据，00 06 表示读序号为 6 开始的自定义 2 字节数据，00 08 表示读连续的 8 个数据，A5 BA 表示 CRC 校验，此时只要把实时数据放在序号为 6 开始的 8 个自定义 2 字节中就可以了。序号和数量根据 PLC 的 MODBUS 命令进行调整。

整个通讯功能块如下所示：

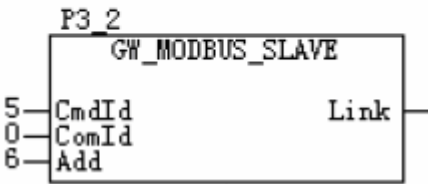
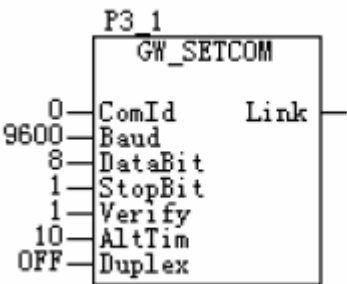


图 5-12 完整通讯功能块

步骤五，XP248 卡保存编译并且下载通过以上几个步骤即完成了对 PLC 数据读取的编程工作。在组态软件中保存工程和全体编译，并且下载到 XP248 卡中。完成整个 XP248 卡的设置。

## 5.12 实例三自定义协议主机通讯模式

### 5.12.1 实现自定义通讯条件

某记录仪提供电压的检测，支持 RS-485 从机通讯，即允许计算机或者其他设备通过自定义协议格式（记录仪提供）读取或者设置记录仪的参数和实时数据。

### 5.12.2 实例情况和分析

本例中我们通过 XP248 卡读取记录仪的 A 相电压限幅参数。

分析：由于该记录仪提供了自定义协议，并且支持 RS-485 从机通讯模式，通讯地址设为 9，因此可以使用 XP248 卡对计算机参数进行读写操作。根据记录仪说明书提供的自定义格式，我们知道对记录仪读操作的协议格式为：地址+93+命令号+CR，记录仪返回的协议格式为低位+高位+CR。其中地址为记录仪的通讯地址，命令号表示对哪个参数进行操作，我们设计的 A 相电压限幅参数的命令号为 6，低位和高位分别表示读取的数据（2 个字节表示），CR 为发送或者接收的字节的异或校验。

### 5.12.3 组态软件设置

XP248 卡的硬件设置和串口连接参考工程实例一，然后进行卡件组态。

XP248 组态主要由几个部分：

- 首先是 SCnet 组态。由于 XP248 与主控制卡都是挂接在 SCnet II 网络上，所以也占用 SCnet II 网络 IP 地址。XP248 的组态方法与主控制卡相同，设置 IP 地址为 128.128.1.6（设定 XP248 卡地址为 6）、控制周期默认为 500ms。卡件冗余方式为不冗余（XP248 支持两块卡的冗余）。
- 其次是自定义位号组态。从下挂设备读出或写到下挂设备的数据都存放在自定义位号中，XP248 通过这些自定义位号与控制系统的操作员站/服务器进行数据交互。我们暂定义 2 字节无符号整数 TAG1 和 TAG2，其中 TAG1 表示记录仪的 A 相限幅参数，TAG2 表示记录仪的 B 相限幅参数。
- 最后是通讯组态。通讯组态也分为二个部分，一部分为通过 ST 段落编写自定义通讯模式下的数据发送格式和解析接收到的数据，另一部分为通过 FBD 段落实现和记录仪的通讯。

另外，如果从智能设备取得的数据需要再进行处理，可以直接在 SCControl 中利用各种丰富的功能函数进行再次计算。

### 5.12.4 SCControl 软件通讯组态

由于 FBD 段落中已有的功能块不能满足自定义协议通讯的要求，因此需要通过 ST 段落编写自定义的功能模块，自定义模块包括 2 个，一个为数据发送格式的填写，一个为接收数据的解析。自定义协议的实现可以通过 SCControl 图形编程软件实现，在组态软件中选择“算法”按钮，在自定义控制算法对话框中选择主控制卡地址为 6（XP248 卡的地址），在图形编程中填入名字后点击“编辑”进行图形化编程，如下所示：



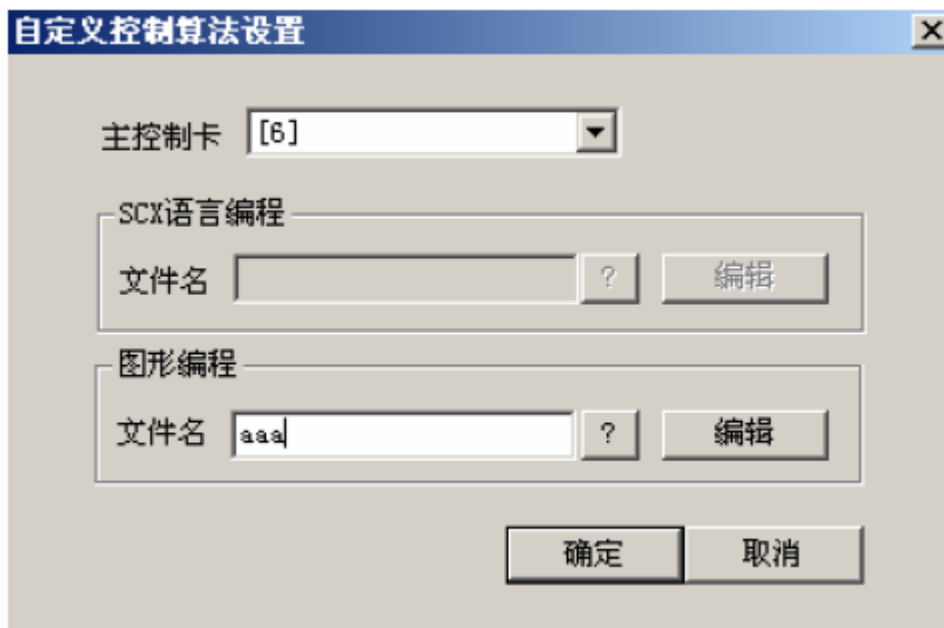


图 5-13 自定义控制算法对话框

进入图形编程软件后，新建一个 ST 段落，取名为 READMSGYB，提供读取参数的发送数据格式。根据自定义协议发送格式编写代码如下。

```

FUNCTION_BLOCK READMSGYB
VAR_INPUT           (*输入管脚定义*)
    CMDID: BYTE;     (*该通讯的命令码，0-255之间，不和其他通讯的命令码重复*)
    ADDR: BYTE;      (*记录仪的通讯地址*)
    CODE: BYTE;      (*命令号，指定需要操作的参数*)
END_VAR

VAR_OUTPUT          (*输出管脚定义*)
    SNDBUF: ULONG;   (*发送数据的缓冲区*)
    RCVBUF: ULONG;   (*接收数据的缓冲区*)
    SNDLEN: UINT;    (*发送的数据长度*)
END_VAR

VAR                (*临时变量定义*)
    CRC8: BYTE;      (*异或校验变量*)
    CRCT: BYTE;      (*异或校验临时变量*)
END_VAR

    SNDBUF = GW_DEFSNDBUF(CMDID, 20); (*定义一个20字节的发送缓冲区*)
    GW_WRITEBUF(SNDBUF, 0, ADDR);     (*缓冲区的第一个字节写入记录仪地址*)
    GW_WRITEBUF(SNDBUF, 1, 93);       (*缓冲区的第二个字节写入读命令93*)
    GW_WRITEBUF(SNDBUF, 2, CODE);     (*缓冲区的第三个字节写入参数命令号*)
    CRCT = XOR_BYTE(ADDR, 93);        (*第一个字节和第二个字节异或校验*)
    CRC8 = XOR_BYTE(CRCT, CODE);      (*和第三个字节异或校验*)
    GW_WRITEBUF(SNDBUF, 3, CRC8);     (*缓冲区的第四个字节写入校验码*)
    SNDLEN = 4;                       (*设置发送的字节数*)
    RCVBUF = GW_DEFRVBUFF(CMDID, 10); (*定义一个10字节的接收缓冲区，准备接收数据*)

END_FUNCTION_BLOCK

```

再新建一个 ST 段落，取名为 GETMSGYB，功能为对返回的数据进行解析，代码如下：



```
FUNCTION_BLOCK GETMSGVB
VAR_INPUT      (*输入管脚定义*)
    CMDID:BYTE; (*该通讯的命令码, 0-255之间, 不和其他通讯的命令码重复*)
END_VAR

VAR_OUTPUT     (*输出管脚定义*)
    LOWR: BYTE; (*输出2字节数据的低位数据*)
    HIGHR:BYTE; (*输出2字节数据的高位数据*)
END_VAR

VAR
    CRC8 :BYTE; (*异或校验*)
    TEMP1:BYTE; (*临时变量*)
    TEMP2:BYTE;
    TEMP3:BYTE;
    DATALEN:UINT;(*接收到的数据长度*)
    RCUBUF :ULONG;(*接收缓冲区*)
END_VAR

DATALEN = GW_GETRCULEN(CMDID); (*得到返回数据的长度*)
IF DATALEN > 0 THEN (*判断是否收到数据*)
    RCUBUF = GW_GETRCUMSG(CMDID); (*得到返回数据的缓冲区*)
    TEMP1 = GW_READBUF(RCUBUF, 0); (*从缓冲区中依次读取字节*)
    TEMP2 = GW_READBUF(RCUBUF, 1);
    TEMP3 = GW_READBUF(RCUBUF, 2);
    CRC8 = XOR_BYTE(TEMP1, TEMP2); (*对取到的数据进行异或校验*)
    IF TEMP3 = CRC8 THEN (*判断如果校验成功, 数据就正确, 赋值到输出管脚*)
        LOWR = TEMP1;
        HIGHR = TEMP2;
    END_IF;
END_IF;

END_FUNCTION_BLOCK
```

完成 2 个 ST 段落的编写，并且编译成功，此时 FBD 的自定义模块中会出现刚才编写的 2 个功能块，新建一个 FBD 段落，开始功能块编程。在模块选择对话框中选择附加库下的智能通讯卡模块中的串口设置功能块，如下：

步骤 1：在模块选择对话框中选择附加库文件夹下的智能通讯卡模块中的通讯口设置模块，模块名称为 GW\_SETCOM，如下图所示：

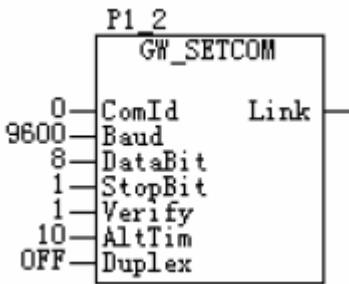


图 5-14 通讯口设置模块

说明：该模块选择了通讯的串口号和串口通讯设置。其中参数 COMID 指定 XP248 通讯的串口号，0~3 分别表示了 COM0~COM3 这 4 个串口。接着 4 个参数分别和连接的设备通讯口设置一致，由于 PLC 的串口已设置成波特率 9600bps，数据位 8 位，停止位 1 位，校验为偶校验。因此，模块的相应参数分别设为 9600、8、1、1，其中参数 VERIFY 表示校验位，其数值 0 为无校验、1 为偶校验、2 为奇校验、3 为置 0、4 为置 1。2 个参数中 AltTim 表示 2 条通讯命令间的间隔时间，一般设为 10ms。参数 Duplex 表示该串口是否冗余，ON 表示和 COM1 冗余，OFF 为不冗余。本例串口不

冗余通讯。

步骤 2：选择自定义模块库文件夹下的自定义模块中选择读模块，读取命令号为 6 的 A 相电压限幅参数，如下图所示：

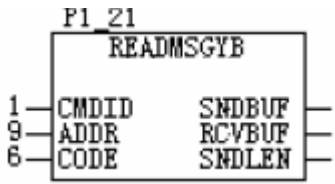


图 5-15 读模块

说明：该模块根据 ST 段落中的定义填入 CMDID 为 1（和其他通讯命令不重复），ADDR 为 9（记录仪的通讯地址），CODE 为 6（根据自定义协议读取 A 相电压限幅参数）。

步骤 3：在模块选择对话框中选择附加库文件夹下的智能通讯卡模块中的通讯模块，模块名称为 GW\_SNDRCV，模块如下所示：

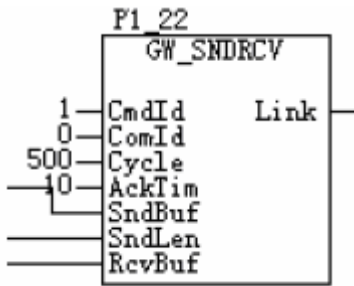


图 5-16 通讯模块

说明：该模块主要负责将发送缓冲区中的指定字节数发到记录仪，并且将记录仪返回的数据保存到指定的接收缓冲区中。

步骤 4：选择自定义模块库中的选择命令解析模块，对获取的数据进行解析和获得。

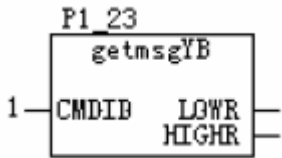


图 5-17 命令解析模块

说明：对得到的数据进行解析，通过 2 个字节在 LOWR 和 HIGHR 中输出数据。整个完整的程序如下图所示。

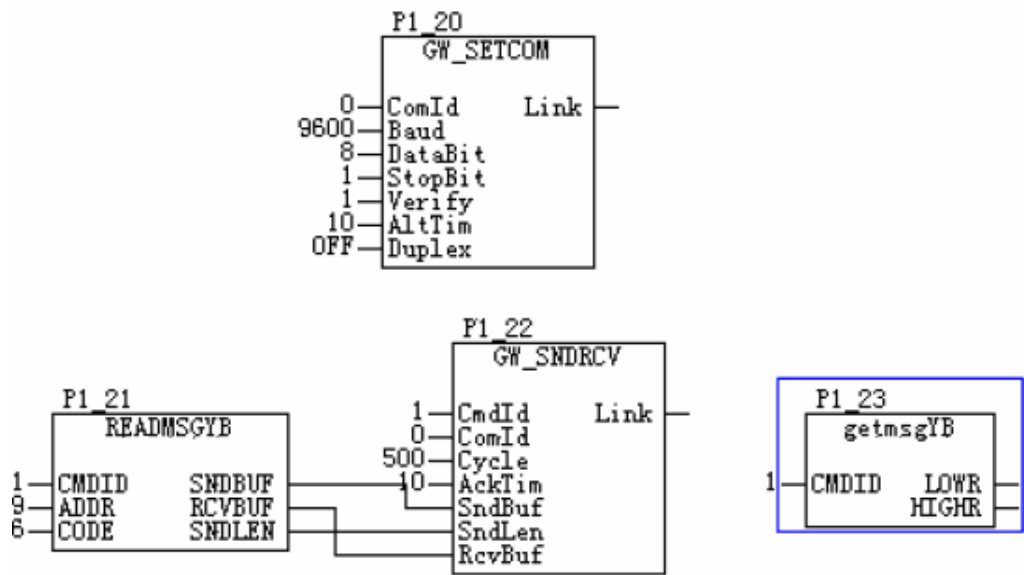


图 5-18 完整程序

将组态软件进行编译并下载,这样就完成了从记录仪中读取 A 相电压限幅参数,数据通过 LOWR 和 HIGHR 显示。

5.13 应用注意事项

- 在使用功能块 GW\_MODBUS\_RTU 时 Num 的最大值如下表所示:

表 5-1 GW\_MODBUS\_RTU 功能块 Num 最大值表

命令号	1	2	3	4	5	6	15	16
最大值 (个)	255	255	127	127	1	1	255	123

- 在使用功能块 GW\_MODBUS\_SLAVE 时,本卡件对主机模块协议能响应的读取数据的最大值如下表所示:

表 5-2 卡件响应主机模块协议的最大读取数据

命令号	1	2	3	4	5	6	15	16
最大值 (个)	2000	2000	125	125	1	1	2040	127

- XP248 (V3.0) 版本卡件须配套 AdvanTrol-Pro V2.65 软件 (含) 以上版本使用。

## 6 资料版本说明

表 6-1 版本升级更改一览表

资料版本	适用产品型号	更改说明
V1.0	XP248 V33.00.33.04	
V1.1		重新排版与修订
V1.2 (20140116)		新增 5.9~5.12 章节内容
V1.3 (20160520)	XP248 V31.32.00 及以上版本	调整格式
V1.4 (20160822)	XP248 V31.32.00 及以上版本	更改封面样式，增加文档编码