

使用入门 •11/2014

S7-1500 Modbus-RTU 使用快 速入门 S7-1500、ET 200MP、ET 200SP、Modbus-RTU

http://support.automation.siemens.com/CN/view/zh/105784261

Copyright © Siemens AG Copyright year All rights reserved

目录

S7-1500 Modbus-RTU 功能简介	3
S7-1500 Modbus-RTU 系统实例	5
该实例所用的软件及硬件	6
硬件组态步骤	7
软件编写	9
注意事项	23
常见错误	24
本例程下载地址	25
	S7-1500 Modbus-RTU 功能简介 S7-1500 Modbus-RTU 系统实例

1

S7-1500 Modbus-RTU 功能简介

S7-1500 可以在主机架或分布式 IO 站使用点对点通信模块来实现 Modbus-RTU 通信,如需在 S7-1500 的分布式 IO 站上实现 Modbus-RTU 通信,推荐通过 Profinet 或者 Profibus 的方式扩展 ET 200MP 或者 ET 200SP 站,通过在 ET 200MP 或者 ET 200SP 上配置 HF(高性能)的点对点通信模块来实现。

根据实际使用通信物理接口的不同,S7-1500 主机架和 ET 200MP 可使用的模 板分 CM PtP RS232 HF(6ES7 541-1AD00-0AB0)和 CM PtP RS422/485 HF

(6ES7 541-1AB00-0AB0) 两种,普通的点对点通信模块(型号 BA 结尾)是 无法通过本文档实现 Modbus-RTU 通信;而 ET 200SP 的点对点通信模块只有 一个,即 CM PtP(6ES7 137-6AA00-0BA0),该模块物理接口支持 RS-232/422/485。

不同的接口类型其端子接线并不相同,详见该模块接线图,链接如下:

S7-1500 CM PtP RS232 HF:

http://support.automation.siemens.com/CN/view/en/59057160/0/zh

S7-1500 CM PtP RS422/485 HF:

http://support.automation.siemens.com/CN/view/en/59061372/0/zh

ET 200SP CM PtP:

http://support.automation.siemens.com/CN/view/en/59061378/0/zh

以上介绍的模板,无论是安装在 S7-1500 主机架的模板还是分布式 IO 站的模板,都可做 Modbus-RTU 的主站或从站,通过直接调用 Modbus-RTU 相关程序块即可实现 Modbus-RTU 通信,而无需硬件狗。

S7-1500 Modbus-RTU 的特点还在于:无论点对点通信模块安装在 S7-1500 主 机架还是分布式 IO 站;也无论分布式 IO 站是通过 ET 200MP 还是 ET 200SP 来实现 Modbus-RTU 通信,其组态步骤、方法、调用的程序块及注意事项等均 完全相同。

S7-1500 支持的 Modbus 功能代码如下表 1 和表 2。

表 1. 用于读取数据的功能: 读取分布式 I/O

Modbus 功能代码	用于读取从站(服务器)数据的功能-标准寻址
01	读取输出位: 每个请求 1 至 2000/1992 ¹⁾ 位
02	读取输入位: 每个请求 1 至 2000/1992 ¹⁾ 位
03	读取保持寄存器: 每个请求 1 至 125/124 ¹⁾ 字
04	读取输入字: 每个请求 1 至 125/124 ¹⁾ 字

1) 用于扩展寻址

表 2. 用于写入数据的功能: 更改分布式 I/O 和程序数据

Modbus 功能代码	用于向从站(服务器)写入数据的功能-标准寻址
05	写入一个输出位: 每个请求 1 位
06	写入一个保持寄存器: 每个请求 1 个字
15	写入一个或多个输出位:每个请求 1 至 1960 位
16	写入一个或多个保持寄存器:每个请求 1 至 122 个字

2

S7-1500 Modbus-RTU 系统实例

本文以示例项目为例,阐述 S7-1500 实现 Modbus-RTU 通信的配置步骤、程序 编写及注意事项,系统配置如下图 1。



图 1 实例系统构成

该实例中, Modbus-RTU 主站为安装在 S7-1500 主机架上的 CM PtP RS422/485 HF, Modbus-RTU 从站模块(站地址为 2)为安装在 ET 200SP 分 布式 IO 上的 CM PtP,接口类型为 RS485,通信波特率为 9600bit/s,无奇偶校 验。 3

Copyright © Siemens AG Copyright year All rights reserved

该实例所用的软件及硬件

SIMATIC 部件

Component	Qty.	MLFB / Order number	Note
CPU 1516-3 PN/DP	1	6ES7516-3AN00-0AB0	FW1.6
存储卡,24 MB	1	6ES7954-8LF01-0AA0	以实际容量为准
通信模块, CM PtP, RS422/485, 高性能型	1	6ES7541-1AB00-0AB0	
IM 155-6 PN ST, 带服务器模块和 总线适配器 2xRJ45 (6ES7193- 6AR00-0AA0)	1	6ES7155-6AA00-0BN0	
CM PtP(自由口,3964R,USS, Modbus RTU)	1	6ES7137-6AA00-0BA0	
BU A0 型, 16 个直插式端子, 2 个 单独馈电端子(数字量/模拟量, 最 高 24VDC/10A)	1	6ES7193-6BP00-0DA0	
PC	1		

软件

Component	Qty.	MLFB / Order number	Note
SIMATIC STEP 7 Professional V13	1	6ES7822-1AA03-0YA5	Step7 V13 Update 6

硬件组态步骤

按照系统配置图,首先组态 S7-1516 CPU 主机架,在主机架上配置点对点通信 模块,并将该模块的协议设置为" Modbus",如下图 2。

1500 Modbus-RTU 🕨	PLC_1 [CPU 1516-3 PN/DP]	@ ■ × _ 硬件目录	
		🦉 拓扑视图 📥 网络视图 📑 设备视图 选件	
df PLC_1	 H G 11 0% 		
	A.	△	
	SAL	<搜索>	
	C APR	= ☑过滤	
	Sp. On		
	0 1 2 3 4 5 6 7 15 23 31	▶ Im DI	
是朝 0		> 🛅 DQ	
0.04		< S DIDQ	
	8 16 24		
	🔤 🚎 📷 📷	▼ ■ 通信模块	
		PROFINET/Ethernet	
		▶ Marketers	
		CM PtP RS232 B/	A
		• CM PtP R5422/48	85 BA
		- I CM PtP RS232 H	.F
<	11	> 1 CM PtP R5422/48	85 HF
CM PtP RS422/485 HF_	_1 [CM PtP RS422/485 HF]	◎属性 14信息 1 2 诊断 □ = ▼ ■ 6ES7 541-1A	800-0A80
常規 10 变量	系统常数 文本	▶ ■ 上乙煨状	
 常規 		▲ ~ 信息	
• 模块参数	KS422/485 按口		
- RS422/485 接口	端口组态		
硬件标识符	协议		
₩O 地址			
	协议: Modbus	CM PtP R542	2/485 HF
	: 对于 MODBUS 协议, 必须使用 Modbus_Comm_Load 描号	?在 PLC 程序中进行组态	
	·	✓ 订货号: 6ES7 541-1A	-B00-0AB0
	一種性振辺袋	版本: V1.0	
		说明:	
	現什你院付	具有 RS422/485 接口的通信都	莫块
	硬件标识符: 257	(Freeport、3964(R)、USS、 :15 轩 D-sub 插孔、115.2 ki	(bps

图 2 S7-1500 CPU 主机架点对点通信模块组态

然后激活 CPU 的"系统和时钟存贮器功能",如下图 3。(本例程中使用了 CPU 首次扫描位来实现 Modbus 的初始化,使用 CPU 的时钟信号来控制发送频 率,该方法供参考。)



Copyright © Siemens AG Copyright year All rights reserved 4

图 3 激活 CPU 的系统和时钟存储器

然后组态 ET 200SP 分布式 IO 站,并在该分布式 IO 上组态点对点通信模块,并 将该模块的通信协议设置为" Modbus",如下图 4。



图 4 ET200 SP 分布式 IO 站点对点模块组态

至此硬件配置已完成。有关 ET 200SP 分布式 IO 组态详细步骤及如何为 Profinet 设备分配 Device Name,本使用入门不再阐述,如需帮助,请参阅下载 中心文档:《ET 200SP 使用快速入门》,链接如下:

<u>http://www.ad.siemens.com.cn/download/docMessage.aspx?ID=7363&loginID=</u> <u>&srno=&sendtime=</u>

软件编写

S7-1500 实现 Modbus-RTU 功能,需要调用以下指令,见下表 3,其中" Modbus_Comm_Load"指令用于通信模块的组态,"Modbus_Master" 指令 和"Modbus_Slave" 指令分别实现 Modbus 主站通信和 Modbus 从站通信。 "Modbus_Comm_Load"指令和"Modbus_Master"/"Modbus_Slave" 指令是通过"Modbus_Comm_Load"指令的"MB_DB"参数来实现关联的。 表 3. Modbus-RTU 相关指令

指令	含义
Modbus_Comm_Load	指令 Modbus_Comm_Load 允许组态 Modbus RTU 的通信模 块端口。
Modbus_Master	Modbus_Master 指令允许通过 PtP 端口作为 Modbus 主站进行通信。
Modbus_Slave	Modbus_Slave 指令允许通过 PtP 端口作为 Modbus 从站进行 通信。

在此,首先编写 Modbus 主站程序,添加一个新FB,将其命名为" Modbus-Master" ,如下图 5。

添加新块		×
名称: Modbus-Master]
■ <mark>●</mark> ● 组织块	语言: LAD ▼ 编号: ① 手动 ④ 自动	1
王 函数块	描述: 函数块是将自身的值永久存储在背景数据块中的	的代码块:从而在块执行后这些值仍然可用。
王子 函数		
DB 数据块 (DB)	更多信息	
> 其它信息		
☑新增并打开(0)		确定取消

图 5 添加 Modbus-Master 功能块

在该 FB 中以多重背景方式调用"Modbus_Comm_Load"指令,该指令在指令目录下"通信—)通信处理器—)Modbus(RTU)"下,如下图 6。

Copyright © Siemens AG Copyright year All rights reserved 5



					Totally Integrated Au	tomation
					Totally Integrated Au	PORTA
500 Modbus-RTU > PLC	_1 [CPU 1516-3	PN/DP] ▶ 程序块	Modbus-Master [FB1]	_ # =×	指令	- II (
					选件	
S 🔊 🖻 🍬 🖿 🗖	🔳 💬 🗃 ± 🖓	• 🖃 🗊 🍋 🖕	8 GH 49 L 1 & 8 10 II			
Modbus-Master					✔ 收藏夹	
名称	数据类型	默认值	保持性 可从 HMI 在 HMI.			
🕣 👻 Input		_		A 1		-
■ <新增>			-			
<				>		
IF +/F -0- 121 -→ -	±					
也提照						
· ············ 注释						
an de rin 4					> 基本指令	_
桂 序段1:					✓ 扩展指令	
汪祥	调用选项			×	名称	抽述
		多重背景			▼ 上艺	1++ 1E
		界面中的名	称 dbus_Comm_Load_Instanc		治称	油还
	DB	调用的函数抉择	, 相关,我们们们们们们们们们们们们们们们们们们们们们们们们们们们们们们们们们们们们	ia l	< / 通信	_
	単个 立例	是自己的背景数	1据块中。这可以让您将背景数据集中在一个1 结小的非易为提供或预测。	央中	▲称 ▶ ┣┓ с⁊ 涌 信	
	<u>×</u> 0	- 并通过程序中	49(1)目常到描述就规。		▶ □ 开放式用户诵信	
程序段 2:					▶ 🗀 WEB 服务器	
注释					▶ 🗀 其他	
1	▲DB 久舌				▼ □ 通信处理器	
		更多信息			▶ PTP Communication	
					MODBUS (RTU)	
					Hodbus_Comm_Lo	ad
1			福宁即湖		Modbus_Master	
			明 1 4 7 7		Modbus_Slave	
					▶ LI200S 串行接口	
					Simauciver CP	

图 6 调用" Modbus_Comm_Load"

在该 FB 中以多重背景方式调用"Modbus_Master"指令,该指令在指令目录下 "通信一〉通信处理器一〉Modbus(RTU)"下,如下图 7。

Woodbus-RTU PLC	_1 [CPU 1516-3 I	M/DP] / 在序り	< Modbus-Ma	aster [FB1]			18 文 叶丹				
🗸 _st _st 💿 🛌 🖼		• 🗔 💽 🗛 🖉	All Con 475 1	1 0.000	eta.	-	达计				-
on ≝r ≝r <mark>≪a</mark> ⊫⊨ ⊟ Nochus-Mastar	<u>∎</u> [2] - 4 = - 4 =	• 🖃 🔛 🖤 🖗 🛛	(e ve 😵 'e	₩ 0° 5	1183		⊶ ।kr	黄士			
名称	数据类型	默认值	保持性	可从 HMI	在 HMI	设	* 12	鴉大	_	17	
🛛 👻 Input						^			- 12?	→	-1
■ <新增>											
■ ◆ Output ■ 《新憎》											
∎ ▼ InOut											
■ <新增>						~					
- Static						>	∖ Jt	木指会			
	±						・至くが	年指マート			
							名称	× HI ≫			描述
t你題: 『							~ I	艺			
*							名称				描述
·程序段 1:							✔ 通	信			
程序段 2:							名称	17 42			
注释	用田井西					-		57 週信 日初式日	白通信	r	
	可用述项	女手北目			,			NEB 服务	.器	-	
		夕里 月京 男面巾的夕秒	Modbu	r Mactar Inct	ance 💌		• 🗖 :	其他			
	DB	77回中的石松				-	- 7	通信处理 DtD Cor	봅	lion	
	单合	调用的函数块将具 是自己的背景数据	数据保存在调用函数 快中。 这可以让您将	时间 亲 数 据 与 行 景 数 据 集 中 7	9年, 而不 在一个块中			USS iff	信	lion	
程序路 3 ·	头1例	· 并通过程序中较;	少的背景数据块获取	•		E	-	MODB	JS (RTU))	
注释								Hoo Moo	lbus Cor	nm Lo	ad
(L)T						E .		Mod	ibus_Ma	ster	
	A 重						•	ET200	5串行接	€□	
	背景	更多信息				-	•	Simatio	NET CP		

图 7 调用" Modbus_Master" 指令

然后对" Modbus_Comm_Load" 指令进行参数化,由于该指令参数较多,在此 只列出必须要关注的参数,如下表 4 所示,其它参数解释见手册或在线帮助。

表 4. Modbus_Comm_Load 主要参数列表

参数	声明	数据类 型	缺省 值	说明
REQ	IN	Bool	FALSE	当此输入出现上升沿时,启动该指令。
PORT	IN	Port	0	CM 端口值,即"硬件 ID"(Hardware ID)。符号端 口名称在 PLC 变量表的"系统常数"(System constants)选项卡中指定。
BAUD	IN	UDInt	9600	选择数据传输速率 有效值为: 300,600,1200,2400,4800,9600, 19200,38400,57600,76800,115200 bit/s.
PARITY	IN	UInt	0	选择奇偶校验: • 0 - 无 • 1 - 奇校验 • 2 - 偶校验

MB_DB	IN/OUT	MB_BASE		对 Modbus_Master 或 Modbus_Slave 指令的背景 数据块的引用。 MB_DB 参数必须与 Modbus_Master 或 Modbus_Slave 指令的静态参数 MB_DB 关联。
MODE	Static	USInt	0	工作模式 有效的工作模式包括: • 0 = 全双工 (RS232) • 1 = 全双工 (RS422) 四线制操作(点对点) • 2 = 全双工 (RS 422) 四线制模式(多点主, CM PtP (ET 200SP)) • 3 = 全双工 (RS 422) 四线制模式(多点从, CM PtP (ET 200SP)) • 4 = 半双工 (RS485) 二线制模式

首先要为" Modbus_Comm_Load" 指令指定端口,即该指令是针对哪个点对点 模块进行参数化的。在硬件配置中,每个硬件均有一个硬件标识符,该硬件标识 符在硬件属性中可以查看到,如下图 8。

CM PtP RS422/485 HF_	1 [CM PtP RS422/485 HF]	☑属性 凶信息 8 型诊断	
常规 10 变量	系统常数 文本		
▶ 常规 ▶ 模块参数	RS422/485 接口		
▼ RS422/485 接口	端口组态		
·····································	协议		
	协议:	Modbus	-
		对于 MDBUS 协议:必须使用 Modbus_Comm_Load 指令在 PLC 程序中进行组态	~
	硬件标识符		
	硬件标识符 硬件标识符:	257	
	, //o 地址		
	, 输入地址		
	起始地址: 结束地址: 组织块: 过程映像:	0 7 ···(自动更新) · · 自动更新 ·	

图 8 在硬件属性中查看模块硬件标识符

同样,该硬件标识符也可以在"PLC变量—〉显示所有变量—〉系统变量"下可以查看到,如下图 9。

opyright © Si AG Copyrigh All rights rese

150	0 M	odbus-RTU 🔸 PLC_1 [CPU 15	16-3 PN/DP] 🕨	PLC 变量 🗕 🖬 🖬	×								
			变量 ■ 用户	🖙 🗐 🕫 系统常量									
			11										
P	PLC 变量												
		名称	数据类型	值	-								
45		端口_1[PN](1)	Hw_Interface	73	~								
46	E	OB_Main	OB_PCYCLE	1	1								
47	E	CM PtP RS422 485 HF 1[DI]	Hw SubModule	257									
48	P	PROFINET_IO-System[IOSystem]	Hw_loSystem	258									
49	Ę	IO_device_1[Proxy]	Hw_SubModule	264									
50	1	IO_device_1[IODevice]	Hw_Device	262									
51	F	PROFINET_接口	Hw_Interface	265									
52	F	Port_1[PN]	Hw_Interface	259									
53	F	Port_2[PN]	Hw_Interface	260									
54	.=	IO device 1[Head]	Hw SubModule	261									
55	P	CM_PtP_1[AI]	Hw_SubModule	266									
56	Ţ	服务器模块_1	Hw_SubModule	267									

图 9 PLC 变量表中查看系统常量

所以可以通过如图 10 所示方法,通过拖拽的方式,将 Modbus 主站接口的硬件标识符拖至" Modbus_Comm_Load"指令的" Port"接口参数处,如下图 10。



图 10 为" Modbus_Comm_Load" 指定端口

接下来,定义端口的工作模式,本示例中,点对点模块的工作模式为 RS485, 所以需要将"Modbus_Comm_Load"背景数据中静态变量的"MODE"参数赋 值为 4,赋值既可以通过"Move"指令来完成,也可以通过直接修改该静态变 量的默认值来实现,本实例使用后一种方法,参见下图 11。





图 11 定义 Modbus-RTU 主站端口工作模式为 RS485

接下来,通过对"Modbus_Comm_Load"指令的"MD_DB"参数赋值,将" Modbus_Comm_Load"指令与"Modbus_Master"指令进行关联,即将" Modbus_Master"指令的背景 DB 块中静态变量"MB_DB"赋值给" Modbus_Comm_Load"指令的"MD_DB",可以通过拖拽的方式来实现,拖 拽路径如下图 12。





图 12 对" Modbus_Comm_Load" 指令的" MD_DB" 参数赋值

除以上操作外,对于"Modbus_Comm_Load"指令的"REQ"参数,本实例使用PLC的首个扫描位来完成。其它参数如波特率,奇偶校验等,请根据实际使用情况对这些参数进行赋值,因本实例波特率为9600bit/s,无奇偶校验,所以以上参数使用缺省设置即可。

接下来,对指令"Modbus_Master"进行参数设置,该指令主要参数如下表 5 所示。

表 5. Modbus_Master 主要参数列表

参数	声明	数据 类型	缺省值	说明
REQ	IN	Bool	FALSE	FALSE = 无请求 TRUE = 请求向 Modbus 从站发送数据
MB_ADDR	IN	UInt	0	Modbus RTU 站地址: 标准地址范围(1到247以及0,用于 Broadcast) 扩展地址范围(1到65535以及0,用于 Broadcast) 值0为将帧广播到所有 Modbus 从站预 留。广播仅支持 Modbus 功能代码05、 06、15和16。
MODE	IN	USInt	0	模式选择: 指定请求类型(读取、写入或 诊断)。

DATA_ADDR	IN	UDInt	0	从站中的起始地址: 指定在 Modbus 从站中访问的数据的起始地址。
DATA_LEN	IN	UInt	0	数据长度: 指定此指令将访问的位或字的 个数。
DATA_PTR	IN/OUT	Variant		数据指针: 指向要进行数据写入或数据读 取的标记或数据块地址。

由于 Modbus 指令读取或写入的数据区必须为指针寻址,所以必须是有绝对地址 的区域方可访问,而 S7-1500 创建的 DB 块缺省为优化的 DB 块,变量没有绝对 地址,无法直接使用。本例中我们创建 DB 块,并在该 DB 块内创建一个名为" M_Data"的数组,类型为 WORD,长度为 100,即创建了 100 个字的存储空间 。然后在该 DB 块点右键,在属性中将"优化的块访问"前的勾去掉,重新编译 该 DB 块,该 DB 块就会生成,可以看到每个变量都有偏移地址了,如下图 13。



图 13 创建一个标准 DB 块

根据下表 6 所示的 Modbus 功能码,本实例需要 Modbus 主站读取 Modbus 从站保持寄存器从起始地址开始的 10 个字的内容到创建的" Master_Data"中,即 Modbus 功能码 03 的功能:

表 6. Modbus 功能码的选择

MOD E	DATA_/	ADDF is 地	२ 址)	DAT/ (数携	A_LE 张度	N)	Modbus 功能代 码	运行和数据					
0				每个	请求的	的位数	01	读取输出位:					
	1	到	9999	1	到	2000/1992 1	1	0	到	9998			
0		61 - C		每个	请求的	的位数	02	读取输入位:					
	10001	到	19999	1	1 到 2000/1992 1			0 到 99					
0				每个	请求的	的字数	03	读取保持寄存器:					
	40001	到	49999	1	到	125/124 1		0	到	9998			
	40000	到	465535	1	到	125/124 1		0	到	65534			
0				每个	请求的		04	读取输入字:	读取输入字:				
	30001 到 39		30001 到 39999 1 到 125/124 1					0	到	9998			
1			每个	请求自		05	写入一个输出位:	1					
	1 到 9999			1				0	到	9998			
1				每个	请求	 1 个字	06	写入一个保持寄送	存器。				
	40001	到	49999	1	11-14			0	到	9998			
	40000	到	465535	1				0	到	65524			
1	<u></u>			伝入	法式的	如你粉	15	(定) 2 个给电台。	_				
<u></u>	1	ठा	0000	т т Г	찌	1000/1000 1	15	与八多 一 棚山也:	ठा	0008			
	- 1	1 到 99		פפפפ ני≆ן ד		2 到 1968/1960 '			16	□ 1 9998 (三) 2 ∧ (には安方県)			
1		74		母个	府水日	Ŋ子数 ┃	16	与八多个保持奇	子裔:				
	40001	到	49999	2	到	123/122		0	到	9998			
	40000	到	465534	2	到	123/122 1		0	到	65534			
2 ²				每个	请求的	的位数	15	写入一个或多个!	写入一个或多个输出位:				
	1	到	9999	2	到	1968/1960 ¹		0	到	9998			
2 ²				每个	请求的	的字数	16	写入一个或多个	保持	寄存器:			
	40001	到	49999	1	到	123	1	0	到	9998			
	40000 1	到	465535	1	到	122 1]	0	到	65534			
11	此功能料 和 DAT	∟ 各_LE	A Modbus_ N 操作数。	Maste	n 的 I	DATA_ADDR	11	读取从站通信的料 数器。状态字表 不忙,0xFFF- 数器随着帧的每沿 增。	大态与示" 忙) 次成功	P和事件计 忙"(0- 。事件计 边处理而递			
80				每个	请求	1 个字	08	使用数据诊断代码 从站状态(回送测 请求的回应)	马 0x 则试 ·	0000 检查 · 从站返回			
	-			1			1	-					
81				每个	请求	1 个字	08	利用数据诊断代码设置从站事件计算	马Ox 数器	000A 重新			
				1			1	-					
3到 10, 12到 79, 82到	*			-				保留					

根据以上要求,则" Modbus_Master" 指令应按如下赋值:

- "REQ":本实例使用 PLC 时钟信号来完成,即下图中的 M0.5;
- " MB_ADDR": 2 //访问的从站地址;
- " MODE": 0; //与" DATA_ADDR" 参数一起决定 Modbus 功能码为 03
- " DATA_ADDR" :40001 //Modbus 地址
- " DATA_LEN": 10 //数据长度为 10 个字

"DATA_PTR": 该参数可以通过拖拽的方式,将创建的标准 DB内的变量"M_Data"拖拽到"DATA_PTR"处,拖拽路径如下图 14。



图 14 为" Modbus_Master" 指令进行参数赋值

接下来,编写 Modbus_RTU 从站程序,参照主站的程序,添加一个新 FB 2"Modbus-Slave",在该 FB 中以多重背景方式调用"Modbus_Comm_Load" 指令,同样为该指令选择 ET 200SP 的硬件地址,其它通信参数如波特率、奇偶 校验等与主站的"Modbus_Comm_Load" 指令相同,如下图 15。



图 15 在从站 FB 中调用" Modbus_Comm_Load" 指令

使用定义主站端口的工作模式相同的方法,将从站点对点模块的工作模式定义为 RS485,所以需要将"Modbus_Comm_Load"背景数据中静态变量的"MODE"参数赋值为 4,如下图 16。

요 말' 말' 🤹 🖿 🖻 🚍 🕎 🍓 Aodbus-Slave	± ≥ ≝ ≣ 🔛 🕫 😡	o (el v≝ 🛠 '	а <u>а</u> (6, 5) [10			
名称	数据类型	默认值	保持性	可从HMI	在 HMI	设置值	注释
🗉 = 🔽 Modbus_Comm_Load_Instan	ce Modbus_Comm_Lo	ad					
🗉 🔹 🕨 Input							
🖬 🔹 🕨 Output							
InOut							
Static							
ICHAR_GAP	Word	16#0	未保留	¥	 Image: A start of the start of		Inter-character gap
RETRIES	Word	16#0	未保留		 Image: A start of the start of		Number of retries in case of a timeout
MODE	USInt	1 4	未保留 💌				Duplex operation mode
LINE_PRE	USInt	16#00	未保留				Presetting of the receive line
BRK_DET	USInt	16#00	未保留				RS232 BREAK Handling
EN_DIAG_ALARM	Bool	false	未保留				Enable diagnostic alarm generation
EN_SUPPLY_VOLT	Bool	false	未保留				Enable supply voltage diagnostic
b_e_REQ	Bool	FALSE	未保留				Edge flag for REQ
y_state	SInt	16#00	禾保留				State of instruction
Send_Config	Send_Config			¥	Image: A start of the start		Local instance of the instruction Send_C.
Receive_Config	Receive_Config			¥	4		Local instance of the instruction Receive.
程序段1:							
1+ 52							
Para ()							
(Modl	#Modbus_ lomm_Load_ Instance						
(Modi	#Modbus_ lomm_Load_ Instance pus_Comm_Load	0			-		
EN Wood	#Modbus_ Comm_Load_ Instance Dus_Comm_Load ENG	0			-		
Modl EN "Ketscon" and	#Modbus_ Comm_Load_ Instance bus_Comm_Load ENG DON	0 E			-		
Modi EN "FirstScan" — REQ	#Modbus_ .omm_Load_ Instance pus_Comm_Load ENG DON ERRO	0 E R			-		
*FirstScan" – REQ 266	#Modbus_ Comm_Load_ Instance Dus_Comm_Load ENC DON ERRO STATU	0 E → R → IS →			-		
۲۰۰۰ ۲۰۰۰ ۲۰۰۰ ۲۰۰۰ ۲۰۰۰ ۲۰۰۰ ۲۰۰۰ ۲۰۰	#Modbus_ comm_Load_ instance bus_Comm_Load ENV DON ERRO STATU	0 E → R → IS →			-		
Modl EN "FirstScan" – REQ 266 "CM_PP_1AI] – PORT 9600 – BAUD	#Modbus_ Comm_Load_ Instance bus_Comm_Load ENN DON ERRO STATU	0 E → R → IS →			-		
*MM.0 *FirstScan — REQ *CM_PPP_1[AI]" — PORT 9600 — BAUD 9 = PORT	#Modbus_ Comm_Load Instance bus_Comm_Load ENV DON ERRO STATU	0 E R IS			1		
**************************************	#Modbus_ Comm_Load_ Instance bus_Comm_Load ENV DON ERRO STATU	0 E — R — IS —			-		
Mod *FirstCan ⁺ REQ *CM_PUP_1[A]' - REQ 9600 - RAUTY 0 - FLOW_CTR	#Modbus_ Comm_Load_ Instance bus_Comm_Load ENC DON ERRO STATU	0 E → R → S →			-		
%Mn.0 *FirstScan' — REQ 266 *CM_PPE_1/A)" — PORT 9600 — BAUD 0 — PARITY 0 — PLOW_CTE − LOW_CTE 0 — RTS_ON_D	#Modbus_ Comm_Load_ Instance bus_Comm_Load ENN DON ERRO STATU L	0 E R IS			-		
Modi EN "FirstScan" – REQ 266 "CM_PIP_1[A]" – PORT 9600 – BAUD 0 – FAUY 0 – FLOW_CTR 0 – RTS OFE 1 0 – RTS OFE 1	#Modbus_ Comm_Load_ Instance BNG DON ERRO STATU	0 E — R — IS —			Т		
%M1.0 EN "FirstScan" — REQ 266 "CM_PP_1[A]" — PORT 9600 — BAUD 0 — PARITY 0 — FLOW_CTR 0 — RTS_ON_D 0 — RTS_OFF_I 0 — RTS_OFF_I 1000 — BFSFD	#Modbus_ Comm_Load_ Instance bus_Comm_Load ENN DON ERRO STATU LY LY	0 E → R → IS →			-		

图 16 定义 Modbus-RTU 从站端口工作模式为 RS485

然后,以多重背景的方式调用从站指令"Modbus_Slave"指令,并设置 Modbus 从站地址为 2,并为从站创建一个标准的 DB 块"Slave Data",长度 根据实际情况定,本例中在"Slave Data"中创建了一个长度为 200 字的数组 ,并将该变量以拖拽的形式(也可以通过指针 P#的方式),将该变量填在 "Modbus_Slave"的"MB_HOLD_REG"参数处,如下图 17。

ví	Siemens - 1500 Modbus-RIU								<u> </u>	
I	页目(P) 编辑(E) 视图(V) 插入(1) 石	主线(O)	选项(N) 工具(T) 窗	iロ(W) 帮助(H)						
I	🛉 🎦 🖬 保存项目 🔳 🐰 🗐 🗎 🗙 🛎) ± (*	* 🖬 🖥 🛄 🛍 関 🛤	. 💋 转到在线 🖉 转到	高线 🔓 🖪 📕	× 🗄 🛄				
	项目树		U • PLC_1 [CPU 1516	i-3 PN/DP] > 程序块	Modbus-Slav	/e [FB2] 🗕		指令		
	设备							选件		
	1900	7	🖓 🖓 🖻 🔮 💺 🖿	🖻 🚍 💬 🖀 ± 🖴 ± [= 🦻 🧐 🖕 ở	8 49 L- I	•			
			Modbus-Slave					> 收藏夹		
민반	▼ ☐ PLC_1 [CPU 1516-3 PN/DP]	^	名称	数据类型	默认值	保持性	可从	> 基本指令		
섊	₩ 设备组态				τ		1240	✓ 扩展指令		
PLC	Q 在线和诊断		⊣⊢ ⊣/⊢ →/// →	_				名称	描述	版本
	▼ 雪 柱序 状		▶ 块标题:					▼ 工艺		
	Main [OB1]		▶ 程序段 1:					名称	描述	版本
	Modbus-Master [FB1]	=	▼ 程度限 2 ·				_	✔ 通信		
	Modbus-Slave [FB2]		1177PX				-	名称	描述	版本
	Master Data [DB1]		21.14					▶ 🛄 57 通信		V1.2
	● Slave Data [Db2]			#Modbus_S	lave_			▶ □ 开放式用户通信		<u>V3.1</u>
	▶ 및 工艺对象			Instanc	:e			▶ 🛄 WEB 服穷器 ➡ 🛅 甘 ⑾		
	▶ 圖 外部源文件			Modbus	Slave			MODBUS TCP		V3.1
	▼ C g B C g B C c c c c c c c c c c c c c c c c c c			EN	ENO			▼ 🗀 通信处理器		
	◎ 显示所有受重 ● 活加新充品表		2.	MB_ADDR	NDR	- 1		PtP Communication		<u>V2.1</u>
	■ / / / / / / / / / / / / / / / / / / /		PHDP2 DPV0.0		DR			▶ Duss 诵信		<u>V2.1</u>
	▶ PLC 数据类型	~	"Slave Data",S		ERROR	-i		MODBUS (RTU)	ood 细态 Ma	<u>V2.1</u> ▼ withour 的谜口 V2.0
	∨ 详细视图		Data -	MB HOLD REG	STATUS			Modbus_Comm_L	他为 Mc	dbus 于站进, V2.1
			1					Modbus_Slave	作为 Mc	dbus 从站进. <u>V2.1</u>
	名称 偏移 数据类型							▶ 🛅 ET2005 串行接口		V2.6
	S_Data 0.0 Array[099] of Word							Simatic NET CP		V3.3
			▼ 程序段3:							

图 17 调用" Modbus_Slave" 指令

然后对"Modbus_Comm_Load"指令的"MD_DB"参数赋值,将" Modbus_Comm_Load"指令与"Modbus_Slave"指令进行关联,即将" Modbus_Slave"指令的背景 DB 块中静态变量"MB_DB"赋值给" Modbus_Comm_Load"指令的"MD_DB",可以通过拖拽的方式来实现,拖 拽路径如下图 18。

	11 🔃 🖸 🖳 📮 🏓 转	帮助(H) 到在线 🖉 转到高线	<mark>ሐ? 🖪 🖪 </mark> 🗡					
500 Mc	odbus-RTU → PLC_1 [C	PU 1516-3 PN/DP] 🕨	程序块▶	Modbus-Slave	[FB2]			
Кы й	୬ ୬ ା ⊾ ⊟ ⊑ [፼ ૄ ±	¢° ⊊₀ ¢≣ ¢	a ⊉ I ₌ 1 ₌	e 😵 🛛	2		
Mode	bus-Slave							
名	4称	数据类型	默认值	保持性	可从HMI	在 HMI	设置值	注释
-01 =	 Modbus_Slave_Instan 	Modbus_Slave						
-01	Input							
-01	Output							
-01	InOut	1 🕞	1	v				
-01	 Static 	1						
-01	MB State	Word	16#0	在 IDB 中				Internal state of the Modbus slave op
-	 HR_Start Offset 	Word	16#0	在 IDB 中…				Specifies the starting address of the M
-01	 Extended Addr. 	Bool	false	在 IDB 中…	 Image: A start of the start of			Configures slave addressing: 1=doub
-00	Request Count	Word	16#0	在 IDB 中…				Number of all requests received by th
-01	Slave Message	Word	16#0	在 IDB 中				Number of requests received for this
-	Bad CRC Coun	t Word	16#0	在 IDB 中				Number of requests received that have
-00	Broadcast Court	Word	16#0	在 IDB 中				Number of broadcast requests receiv
-01	Exception Court	Word	16#0	在 IDB 中…				Number of Modbus specific errors that
-671	Success Count	Word	0	在IDB中				Number of successful transmission
	MR DR	P2P MR RASE		在IDB由				Modbus parameters
	SENID PTP	Sond P2P		Щ юю ч				Local instance of the instruction Sound
		#Modbus_ Comm_Load_ Instance						
		Modbus_Comm_Lo	ad					
			ENO				-	
_	EN							
-	%//1.0		DONE					
	"FirstStan" — REO							
	EN %/11.0 "FirstStan" — REQ							
	EN %/11.0 "FirstStan" — REQ 266		DONE ERROR STATUS					
3	EN %/11.0 "FirstStan" — REQ 266 "CM_PtP_1 [[N]" — PORT		DONE — ERROR — STATUS —					
3	EN "FirstStan" — REQ 266 "CM_PtP_1[N]" — PORT 9600 — BAUE	·	DONE ERROR STATUS					
	EN "FirstStan" — REQ 266 "CM_PtP_1 [N]" — PORT 9600 — BAUD 0 — PART) Y	DONE — ERROR — STATUS —					
	EN %/1.0 "FirstStan" — REQ 866 "CM_PtP_1[Ri]" — PORT 9600 — BAUE 0 — PART	7 7	DONE — ERROR — STATUS —					
2	*FirstStan" - REQ *FirstStan" - REQ *66 *CM_PtP_1[10]" - PORT 9600 - BAUE 0 - PART 0 - FLOV	γ Y_CTRL	DONE — ERROR — STATUS —					
2	*FirstStan* — REQ *G66 *CM_PtP_1[h]* — PORT 9600 — BAUD 0 — PART 0 — FLOU 0 — FLOU	r) YY V_CTRL ON_DLY	DONE — ERROR — STATUS —					
2	9600 - REQ 1000 - PARI 1000	TY V_CTRL ON_DLY OFF_DLY	DONE — ERROR — STATUS —					
2	Р. Р	Y NY V_CTRL ON_DLY OFF_DLY TO	DONE — ERROR — STATUS —					
	PM1.0 "FirstStan" — REQ 166 "CM_PP_1(N" — POR 9600 — BAUL 0 — PARI 0 — FLOV — RTS_ 100 ↓ RESP	ry v_ctrl on_dly off_dly _to	DONE ERROR STATUS					
#1	EN %41.0 "FirstStan" — REQ 66 "CM_PIP_11AI" — PORT 9000 — BAUE 0 — PARI 0 — FLOS 0 — RESP 10004 — RESP Modbus_Slave BDB 1 = 0	TY V_CTRL ON_DLY OFF_DLY _TO	DONE ERROR STATUS					

图 18 将" Modbus_Slave" 指令和" Modbus_Comm_Load" 指令关联

在 OB1 中分别调用 Modbus 主站程序块和 Modbus 从站程序块,并为其分配全 局 DB 块,如下图 19。



图 19 在 OB1 中分别调用主站程序和从站程序

至此,程序编写基本结束,建立 2 个变量监视表,分别用来监视 Modbus 主站数 据和 Modbus 从站数据,将该例程下载到 PLC 后,可以看到通信模块对应的收发 LED 指示灯在闪烁,表示端口正在发送/接收数据。将 DB 块"Slave Data"中的变量赋值,监视主站"Master Data"中的变量,可以看到主站已经读取到 从站的数据,如下图 20。

选项(送(项(N) 工具(T) 窗口(W) 帮助(H)											
Ū	1516-3 PN/DP] > 监控与强制表 >	Slave Data Mon	itor	_ 7	×	∍.	1516-3 PN/DP] ▸ 监控与强	制表)Ma	ster Data Moni	tor		×
			_									
							ALE E 4 4 40 00 00					-
N N				<u>.</u>	1411	Be .	er 💵 🕹 71 % 27 🎬 筑					
	名称	显示格式	监视值	修改值	8	1	名称	地址	显示格式	监视值	修改值	
1	"Slave Data".S_Data	十六进制 💌	16#00AA	16#00	^ 1		"Master Data".M_Data[0]	%DB1.DBW0	十六进制	16#00AA	0	^
2	"Slave Data".S_Data[1] %DB2.DBW2	十六进制	16#00BB	16#00	≣ 2		"Master Data".M_Data[1]	%DB1.DBW2	十六进制	16#00BB		=
3	"Slave Data".S_Data[2] %DB2.DBW4	十六进制	16#00CC	16#00	3		"Master Data".M_Data[2]	%DB1.DBW4	十六进制	16#00CC		
4	"Slave Data".S_Data[3] %DB2.DBW6	十六进制	16#00DD	16#00	4		"Master Data".M_Data[3]	%DB1.DBW6	十六进制	16#00DD		
5	"Slave Data".S_Data[4] %DB2.DBW8	十六进制	16#00EE	16#00EE	5		"Master Data".M_Data[4]	%DB1.DBW8	十六进制	16#00EE		
6	"Slave Data".S_Data[5] %DB2.DBW10	十六进制	16#00FF	16#00FF	6		"Master Data".M_Data[5]	%DB1.DBW10	十六进制	16#00FF		
7	"Slave Data".S_Data[6] %DB2.DBW12	十六进制	16#0011	16#00	7		"Master Data".M_Data[6]	%DB1.DBW12	十六进制	16#0011		
8	"Slave Data".S_Data[7] %DB2.DBW14	十六进制	16#0022	16#00	8		"Master Data".M_Data[7	%DB1.DBW14	十六进制 💌	16#0022		
9	"Slave Data".S_Data[8] %DB2.DBW16	十六进制	16#0033	16#00	9		"Master Data".M_Data[8]	%DB1.DBW16	十六进制	16#0033		
10	"Slave Data".S_Data[9] %DB2.DBW18	十六进制	16#0044	16#00	11	0	"Master Data".M_Data[9]	%DB1.DBW18	十六进制	16#0044		
11	"Slave Data".S_Data[10 %DB2.DBW20	十六进制	16#0055	16#00	1	1	"Master Data".M_Data[10]	%DB1.DBW20	十六进制	16#0055		
12	"Slave Data".S_Data[11 %DB2.DBW22	十六进制	16#0066	16#00	13	2	"Master Data".M_Data[11]	%DB1.DBW22	十六进制	16#0066		
13	"Slave Data".S_Data[12 %DB2.DBW24	十六进制	16#0077	16#00	1.	3	"Master Data".M_Data[12]	%DB1.DBW24	十六进制	16#0077		
14	"Slave Data".S_Data[13 %DB2.DBW26	十六进制	16#0088	16#00	1-	4	"Master Data".M_Data[13]	%DB1.DBW26	十六进制	16#0088		
15	"Slave Data".S_Data[14 %DB2.DBW28	十六进制	16#0099	16#00	13	5	"Master Data".M_Data[14]	%DB1.DBW28	十六进制	16#0099		
16	"Slave Data".S_Data[15 %DB2.DBW30	十六进制	16#0012	16#00	14	6	"Master Data".M_Data[15]	%DB1.DBW30	十六进制	16#0012		
17	"Slave Data".S_Data[16 %DB2.DBW32	十六进制	16#0013	16#00	13	7	"Master Data".M_Data[16]	%DB1.DBW32	十六进制	16#0013		
18	"Slave Data".S_Data[17 %DB2.DBW34	十六进制	16#0014	16#00	18	8	"Master Data".M_Data[17]	%DB1.DBW34	• 十六进制	16#0014		
19	"Slave Data".S_Data[18 %DB2.DBW36	十六进制	16#0014	16#00	19	9	"Master Data".M_Data[18]	%DB1.DBW36	- 十六进制	16#0014		
20	"Slave Data".S_Data[19 %DB2.DBW38	十六进制	16#0016	16#00	2	0	"Master Data".M_Data[19]	%DB1.DBW38	- 十六进制	16#0016		
21	"Slave Data".S_Data[2C %DB2.DBW40	十六进制	16#0017	16#00	2	1	"Master Data".M_Data[20]	%DB1.DBW40	十六进制	16#0017		
22	"Slave Data".S_Data[21 %DB2.DBW42	十六进制	16#0018	16#00	2.	2	"Master Data".M_Data[21]	%DB1.DBW42	十六进制	16#0018		
23	"Slave Data".S_Data[22 %DB2.DBW44	十六进制	16#0019	16#00	2	3	"Master Data".M_Data[22]	%DB1.DBW44	十六进制	16#0019		
24	"Slave Data".S_Data[23 %DB2.DBW46	十六进制	16#0020	16#00	2.	4	"Master Data".M_Data[23]	%DB1.DBW46	十六进制	16#0020		
25	"Slave Data".S_Data[24 %DB2.DBW48	十六进制	16#0020	16#00	2	5	"Master Data".M_Data[24]	%DB1.DBW48	十六进制	16#0020		
26	"Slave Data".S_Data[25 %DB2.DBW50	十六进制	16#0020	16#00	2	6	"Master Data".M_Data[25]	%DB1.DBW50	1 十六进制	16#0020		
27	"Slave Data".S_Data[26 %DB2.DBW52	十六进制	16#0020	16#00	2	7	"Master Data".M_Data[26]	%DB1.DBW52	1 十六进制	16#0020		
28	"Slave Data".S_Data[27 %DB2.DBW54	十六进制	16#0020	16#00	2	8	"Master Data".M_Data[27]	%DB1.DBW54	- 十六进制	16#0020		
29	"Slave Data".S_Data[28 %DB2.DBW56	十六进制	16#0020	16#00	2	9	"Master Data".M_Data[28]	%DB1.DBW56	- 十六进制	16#0020		
30	"Slave Data".S_Data[29 %DB2.DBW58	十六进制	16#0020	16#00	3	0	"Master Data".M_Data[29]	%DB1.DBW58	- 十六进制	16#0020		
31	"Slave Data".S_Data[3C %DB2.DBW60	十六进制	16#0020	16#00	3	1	"Master Data".M_Data[30]	%DB1.DBW60	1 十六进制	16#0020		
32	"Slave Data".S_Data[31 %DB2.DBW62	十六进制	16#0020	16#00	3.	2	"Master Data".M_Data[31]	%DB1.DBW62	十六进制	16#0020		
33	"Slave Data".S_Data[32 %DB2.DBW64	十六进制	16#0020	16#00	3.	3	"Master Data".M_Data[32]	%DB1.DBW64	- 十六进制	16#0020		
34	"Slave Data".S_Data[33 %DB2.DBW66	十六进制	16#0020	16#00	3	4	"Master Data".M_Data[33]	%DB1.DBW66	- 十六进制	16#0020		
35	"Slave Data".S_Data[34 %DB2.DBW68	十六进制	16#0020	16#00	3	5	"Master Data".M_Data[34]	%DB1.DBW68	- 十六进制	16#0020		
36	"Slave Data".S_Data[35 %DB2.DBW70	十六进制	16#0020	16#00	3	6	"Master Data".M_Data[35]	%DB1.DBW70	1 十六进制	16#0020		
37	"Slave Data".S_Data[36 %DB2.DBW72	十六进制	16#0020	16#00	3	7	"Master Data".M_Data[36]	%DB1.DBW72	十六进制	16#0020		
38	"Slave Data".S_Data[37 %DB2.DBW74	十六进制	16#0020	16#00	3	8	"Master Data".M_Data[37]	%DB1.DBW74	十六进制	16#0020		
39	"Slave Data".S_Data[38 %DB2.DBW76	十六进制	16#0020	16#00	3	9	"Master Data".M_Data[38]	%DB1.DBW76	- 十六进制	16#0020		
40	"Slave Data".S_Data[39 %DB2.DBW78	十六进制	16#0020	16#00	4	0	"Master Data".M_Data[39]	%DB1.DBW78	十六进制	16#0020		
41	"Slave Data".S_Data[40 %DB2.DBW80	十六进制	16#0020	16#00	4	1	"Master Data".M_Data[40]	%DB1.DBW80	十六进制	16#0020		
42	"Slave Data".S_Data[41 %DB2.DBW82	十六进制	16#0020	16#00	✓ 4.	2	"Master Data".M_Data[41]	%DB1.DBW82	十六进制	16#0020		*
<				3	>		¢	1111			>	

图 20 使用变量监视表测试, 主站侧已成功读取到从站数据

Modbus 其它功能码使用方法类似,请参照表 6 修改相应的变量即可,此处不再 ——举例。

也可以根据此例程,举一反三,编写 Modbus-RTU 轮询程序。

注意事项

Modbus 通信,不论是主站侧还是从站侧,其收发数据区必须是可通过指针寻址的,即必须是有确切地址的数据区;而优化的 DB 块由于其内部变量没有地址,所以收发数据区均不能使用优化的 DB 块,必须使用标准 DB,否则 PLC 运行会报错。

除 Modbus 收发数据区必须使用标准的 DB 块,其它部分如程序等均可使用优化 的程序块实现。

" Modbus_Comm_Load" 指令的初始化信号端" REQ" 和" Modbus_Master

"指令的发送/接收信号端"REQ"必须使用边沿信号触发,否则初始化、发送/ 接收都无法完成。

Copyright © Siemens AG Copyright year All rights reserved 6

7 常见错误

- ➢ 没有定义端口的工作模式。 如未定义,缺省模式为 RS232。
- ▶ 收发数据区使用了优化的 DB。 将优化的 DB 修改为绝对 DB。
- " Modbus_Comm_Load" 初始化未执行。
 重新执行初始化指令。
- "Modbus_Master"指令输入接口参数"DATA_LEN"和"DATA_PTR"不匹配,无法实现收发。
 - "DATA_LEN"必须小于等于"DATA_PTR"指向的数据存储区。
- ▶ 点对点通信模块的"TX"/"RX"或"TXD"/"RXD"灯无闪烁。
 检查"Modbus_Comm_Load"初始化参数,确保其被正确初始化;检查" Modbus_Master"参数和"Modbus_Slave"指令参数,确保参数正确。

Copyright ⊚ Siemens AG Copyright year All rights reserved 8

Copyright © Siemens AG Copyright year All rights reserved

本例程下载地址

http://support.automation.siemens.com/CN/view/zh/105784261