



EasyLad梯形图STC32G12K128

使
用
手
册

(V 1 . 6)



前言

- 本资料包括 EasyLad 梯形图语言在 STC32G12K128 单片机的使用说明和注意事项。
- 有关 EasyLad 梯形图语言的各种指令的解释及编程方法请参照《EasyLad 梯形图语言编程手册》
- 有关梯形图编辑环境的使用方法请参照《EasyLad 梯形图集成编辑器使用手册》

声明

在不予通知的情况下保留对本手册进行修改的权利。

目 录

第一章	芯片的管脚分配和软硬件配置.....	1
1.1	普通开关量输入输出的管脚分配.....	1
1.2	特殊功能的管脚分配.....	1
1.3	芯片特殊功能寄存器预定义文件.....	3
1.4	特殊功能寄存器读写指令.....	3
1.5	设置 IO 端口工作模式.....	4
1.6	CPU 电源管理.....	5
1.7	PWMA 中断.....	5
1.8	如何添加函数库连接.....	7
1.9	芯片 ADC 模数转换.....	8
第二章	编程口 ModBus-RTU 通讯说明.....	9
2.1	编程口 ModBus-RTU 协议格式说明.....	9
2.2	Modbus-RTU 协议的设备地址和 PLC 内部元件对应表.....	9
2.3	Modbus-RTU 协议读写 PLC 内部的双字数据存储器.....	10
第三章	扩展通讯口通讯说明.....	12
3.1	自由协议通讯.....	12
3.2	扩展通讯口 ModBus-RTU 从机配置.....	22
3.3	扩展通讯口 ModBus-RTU 主机配置.....	23
第四章	以太网通讯说明.....	27
4.1	以太网 ModBusTCP 服务器通讯配置.....	27
第五章	脉冲串定位控制应用说明.....	29
5.1	配置 Y0 定位控制功能.....	29
5.2	绝对定位函数 DRVA0.....	30
5.3	相对定位函数 DRVI0.....	31
5.4	电机停止函数 DRVS0.....	32
5.5	读出电机的当前位置函数 RdCPos0.....	32
5.6	变速函数 DRVV0.....	33
5.7	点动函数 JOG0.....	33
5.8	读脉冲串内部寄存器函数 GetPTOR.....	34
5.9	设置 Y0 的剩余脉冲个数寄存器函数 SetPTOR0.....	34
5.10	设置 Y1 的剩余脉冲个数寄存器函数 SetPTOR1.....	35
5.11	设置脉冲频率的输出范围和分辨率.....	35
5.12	电机的回零点.....	36
第六章	SPI 接口数据块操作函数库.....	37
6.1	数据块大端格式 SPI 输出函数 BBSOUT.....	37
6.2	数据块小端格式 SPI 输出函数 BLSOUT.....	37
6.3	数据块大端格式 SPI 输入输出函数 BBSIO.....	38
6.4	数据块小端格式 SPI 输入输出函数 BLSIO.....	38
6.5	字符串变量大端 SPI 输出(为 0 字节不输出)函数 VSTRBSO.....	38
6.6	字符串变量小端 SPI 输出(为 0 字节不输出)函数 VSTRLSO.....	39
6.7	字符串变量大端 SPI 输出(为 0 字节输出)函数 VSTRWBSO.....	39
6.8	字符串变量小端 SPI 输出(为 0 字节输出)函数 VSTRWLSO.....	40

6.9	字符串常数大端 SPI 输出(为 0 字节不输出)函数 CSTRBSO.....	40
6.10	字符串常数小端 SPI 输出(为 0 字节不输出)函数 CSTRLSO.....	41
6.11	字符串常数大端 SPI 输出(为 0 字节输出)函数 CSTRWBSO.....	41
6.12	字符串常数小端 SPI 输出(为 0 字节输出)函数 CSTRWLSO.....	41
第七章	CAN 自由通讯接口使用说明	43
7.1	CAN 通讯口设置函数 CAN_Set.....	43
7.2	CAN 通讯口发送数据函数 CAN_TX.....	43
7.3	读 CAN 通讯口发送缓冲器状态标记函数 CAN_TXF.....	44
7.4	CAN 通讯口禁止发送函数 CAN_DisTX.....	45
7.5	CAN 通讯口允许发送函数 CAN_EnTX	46
7.6	CAN 通讯口重新发送数据函数 CAN_ReTX	46
7.7	读 CAN 通讯口最先接收数据的缓冲器编号函数 CAN_RXB.....	47
7.8	CAN 通讯口接收数据函数 CAN_RX	47
7.9	读 CAN 通讯口总线错误状态函数 CAN_ESR.....	49

第一章 芯片的管脚分配和软硬件配置

1.1 普通开关量输入输出的管脚分配

普通开关量输入输出对应的芯片管脚分配如下表：

芯片管脚	PLC 的开关量输入 X 和输出 Y
P00~P07	X00~X07, Y00~Y07
P60~P67	X10~X17, Y10~Y17
P20~P27	X20~X27, Y20~Y27
P70~P77	X30~X37, Y30~Y37
P40~P47	X40~X47, Y40~Y47
P50~P57	X50~X57, Y50~Y57

说明：

上表中芯片的 IO 端口即可作为 PLC 的开关量输入 X，也可作为 PLC 的开关量输出 Y，当作为 PLC 的开关量输入 X 时，要把对应的 IO 端口模式设置为高阻输入或准双向口模式，当作为 PLC 的开关量输出 Y 时，要把对应的 IO 端口模式设置为推挽输出或准双向口模式，芯片复位后这些 IO 端口都默认为高阻输入模式。

如果输入管脚为低电平，则对应的 X 继电器为 OFF，如果输入管脚为高电平，则对应的 X 继电器为 ON。

如果 Y 继电器为 OFF，则对应的输出管脚为低电平，如果 Y 继电器为 ON，则对应的输出管脚为高电平。

1.2 特殊功能的管脚分配

特殊功能的芯片管脚分配如下表：

芯片管脚	特殊功能
P00~P05	ADC 输入通道 ADC0~ADC5
P06	ADC 输入通道 ADC6，高速计数器 C2 输入
P07	SPI 主机接口片选线 SPI-S6
P00	CAN-RX ₂ 端
P01	CAN-TX ₂ 端
P02	PLC 扩展通讯口 COM1 的 RXD ₂
P03	PLC 扩展通讯口 COM1 的 TXD ₂
P10	EEPROM 芯片 24C256 接口 ROM-SDA
P11	EEPROM 芯片 24C256 接口 ROM-SCL
P13	SPI 主机接口数据输出 SPI-SDO
P14	SPI 主机接口数据输入 SPI-SDI
P15	SPI 主机接口时钟输出 SPI-SCK
P16	32MHz 无源晶振 XTALO
P17	32MHz 无源晶振 XTALI
P20	PLC 的多功能高速输入 HX5
P21	PLC 的多功能高速输入 HX6

P22	PLC 的外部中断输入 HX4, 高速计数器 C1 的 B 相输入
P23	PLC 的外部中断输入 HX3, 高速计数器 C0 的 B 相输入
P24	用户可编程 IIC2 接口的 IIC2-SDA
P25	用户可编程 IIC2 接口的 IIC2-SCL
P30	PLC 编程通讯口 COM 的 RXD
P31	PLC 编程通讯口 COM 的 TXD
P32	高速计数器 C0 的 A 相输入或单相输入
P33	高速计数器 C1 的 A 相输入或单相输入
P34	脉冲串输出或 PWM 输出 PTO-Y1 或 SPI 主机接口片选线 SPI-S7
P35	脉冲串输出或 PWM 输出 PTO-Y0 或 SPI 主机接口片选线 SPI-S8
P36	PLC 编程通讯口 COM 的收发控制端, 0-接收, 1-发送
P37	掉电检测的比较器正输入端, 和芯片内部 1.19 伏基准比较
P40	SPI 主机接口片选线 SPI-S0
P41	SPI 主机接口片选线 SPI-S1
P42	SPI 主机接口片选线 SPI-S2
P43	SPI 主机接口片选线 SPI-S3
P44	SPI 主机接口片选线 SPI-S4
P45	用户可编程 IIC 接口的 IIC-SDA
P46	用户可编程 IIC 接口的 IIC-SCL
P47	PLC 扩展通讯口 COM1 的收发控制端, 0-接收, 1-发送
P50	CAN-RX 端
P51	CAN-TX 端
P52	PLC 扩展通讯口 COM1 的 RXD
P53	PLC 扩展通讯口 COM1 的 TXD
P54	ADC 输入通道 ADC 7, SPI 主机接口片选线 SPI-S5

说明:

CPU 的工作时钟使用外部无源晶振, 频率为 32MHz。

EEPROM 芯片 24C256 为系统提供数据掉电保持存储, 页缓冲区应在 64 字节以上。

掉电检测使用芯片内置的比较器, 当掉电检测管脚输入电压低于 1.19 伏时为掉电状态。

用户要通过分压电阻设置掉电检测电压。**注意若上电时掉电检测管脚输入电压低于 1.19 伏, 则不会运行用户梯形图程序, 直到高于 1.19 伏后才会运行**, 因此若用户不使用掉电检测功能, 也不要让该管脚悬空而要接到 VCC 上。

用作 ADC 输入功能的管脚要为高阻输入模式。其他特殊功能用作输入的管脚可为高阻输入或准双向口模式。高速计数器 C0 和 C1 的 A 相输入由系统设置为高阻输入模式。

脉冲串输出管脚 PTO-Y0 和 PTO-Y1 由系统自动设置为准双向口模式, 且在无脉冲输出时保持为高电平。

编程通讯口由系统自动设置 TXD 管脚和收发控制管脚为推挽输出模式, RXD 管脚为准双向口模式。

扩展通讯口当有效时, 由系统自动设置 TXD 管脚和收发控制管脚为推挽输出模式, RXD

管脚为准双向口模式。

CAN 通讯口当有效时，由系统自动设置 CAN-TX 管脚为推挽输出模式，CAN-RX 管脚为准双向口模式。

SPI 主机接口片选线当执行对应的功能时，由系统自动设置为推挽输出模式，用户在初始化 IO 模式时该管脚可为高阻输入模式，若在初始化 IO 模式时把用作 SPI 主机接口片选线的管脚设置为推挽输出，则要注意初始化后管脚输出为低电平（对应的 Y 继电器状态）。

用户可编程 IIC 接口的管脚当执行对应的功能时，由系统自动设置为准双向口模式，用户在初始化 IO 模式时该管脚可为高阻输入模式，若在初始化 IO 模式时把用作 IIC 接口的管脚设置为输出模式，则要注意初始化后管脚输出为低电平（对应的 Y 继电器状态）。

用户可编程 IIC2 接口使用芯片内置的硬件 IIC 接口，用户可通过操作芯片硬件 IIC 的 XFR 寄存器来进行 IIC 通讯，注意不要使能 IIC 中断。

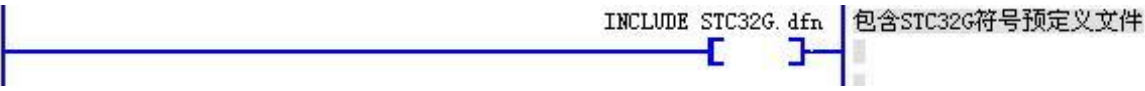
P32、P33、P10、P11、P14、P35、P34、P13 管脚状态可由指令 SFRRD DMx, HX_IW 读到 DMx 中，这些管脚状态按顺序依次为位 0-位 7，不使用特殊功能时也可作为输入使用。

1.3 芯片特殊功能寄存器预定义文件

用户必须在梯形图程序开头使用 INCLUDE 指令包含芯片特殊功能寄存器预定义文件 STC32G.dfn，该文件包含了芯片的 IO 端口模式寄存器、XFR 寄存器等特殊功能寄存器的符号名定义。具体操作步骤如下：

首先找到 STC32G.dfn 文件，把该文件复制到用户的梯形图程序文件夹中。

然后在梯形图程序开头放置直连于左母线的指令线圈，指令为“INCLUDE STC32G.dfn”。梯形图例子如下：



1.4 特殊功能寄存器读写指令

● SFRRD OUT, IN

指令步数：3 步。

操作数	可使用的元件
OUT	D、RM、RY、RX、RC、RT、DM、变址寻址
IN	特殊功能寄存器

特殊功能寄存器字读指令。

若该指令被接通，则把 IN 的内容（1 个字）传送到 OUT 中，对于 XFR 寄存器，

指令中的特殊功能寄存器所在的地址为高字节，地址+1 为低字节。若该指令被断开，则不执行任何操作。

例：SFRRD DM500, PWMA_CCR1

● SFRWR IN, OUT

指令步数：3 步。

操作数	可使用的元件
IN	常数、D、RM、RY、RX、RC、RT、DM、变址寻址
OUT	特殊功能寄存器

特殊功能寄存器字写指令。

若该指令被接通，则把 IN 的内容（1 个字）传送到 OUT 中，对于 XFR 寄存器，指令中的特殊功能寄存器所在的地址为高字节，地址+1 为低字节。若该指令被断开，则不执行任何操作。

例：SFRWR DM500, PWMA_CCR1

● SFRWRB IN, OUT

指令步数：3 步。

操作数	可使用的元件
IN	常数、D、RM、RY、RX、RC、RT、DM、变址寻址
OUT	特殊功能寄存器

特殊功能寄存器字节写指令。

若该指令被接通，则把 IN 低字节的内容传送到 OUT 中，OUT 所指定的地址为字节地址，该指令只能写 XFR 寄存器。若该指令被断开，则不执行任何操作。

例：SFRWRB 0X11, PWMA_CCER1

1.5 设置 IO 端口工作模式

用户可通过 IO 端口模式寄存器来设置 IO 端口工作模式，IO 端口模式寄存器为字寄存器（16 位），具体寄存器名称及说明如下：

P0M: P0 口工作模式寄存器，其高字节为 P0M1，低字节为 P0M0

P2M: P2 口工作模式寄存器，其高字节为 P2M1，低字节为 P2M0

P4M: P4 口工作模式寄存器，其高字节为 P4M1，低字节为 P4M0

P5M: P5 口工作模式寄存器，其高字节为 P5M1，低字节为 P5M0

P6M: P6 口工作模式寄存器，其高字节为 P6M1，低字节为 P6M0

P7M: P7 口工作模式寄存器，其高字节为 P7M1，低字节为 P7M0

可使用 SFRRD 和 SFRWR 指令对上述寄存器进行读写。上电复位后上述端口都为高阻输入模式，若工作于输出口，则必须设置为推挽输出或准双向口模式。

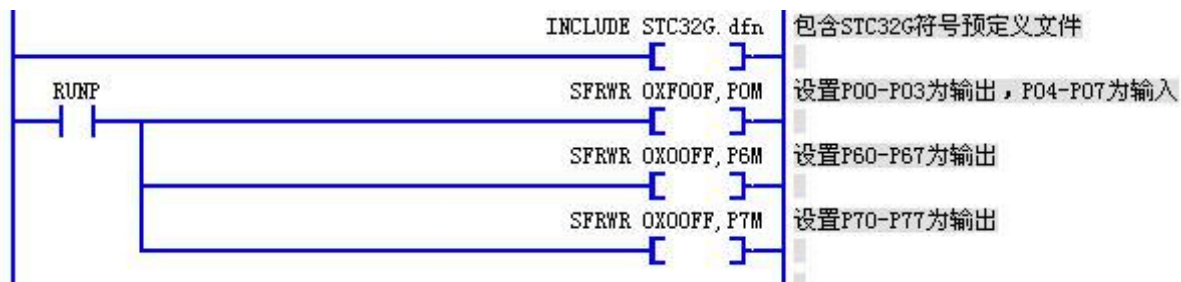
寄存器中的各个位和工作模式的对应关系如下：

PnM1.x	PnM0.x	Pn.x 口工作模式
0	0	准双向口
0	1	推挽输出
1	0	高阻输入
1	1	开漏模式

设置 IO 端口模式数值的快速计算方法为：把每组端口（8 个位，例如 P0）中为输出的位设为 1，为输入的位设为 0，计算出的数值为端口模式数值的低字节，把计算出数值的反码为端口模式数值的高字节，把端口模式数值（16 位）送给端口模式寄存器即可。

若某组端口全为输入，则该组端口不用设置 IO 端口模式。

设置 IO 端口模式梯形图例子如下：



1.6 CPU 电源管理

执行指令“SFRWR 2, PCON”后 CPU 进入睡眠省电模式，唤醒后将从该指令下面的指令开始执行。

1.7 PWMA 中断

PWMA 的某些中断源被映射到梯形图中断中，具体中断入口指令如下：

中断入口指令	对应的 PWMA 中断源
INT I_PWMA_UI	更新事件中断
INT I_PWMA_CC1	输入捕捉/输出比较 1 中断
INT I_PWMA_CC2	输入捕捉/输出比较 2 中断
INT I_PWMA_CC3	输入捕捉/输出比较 3 中断
INT I_PWMA_CC4	输入捕捉/输出比较 4 中断

各个中断源在 EI 指令（开中断）和 DI 指令（关中断）中的控制位如下：

PWMA 中断源在 EI 指令和 DI 指令中的位：

位	14	13	12	11	10
中断源	PWMA_CC4	PWMA_CC3	PWMA_CC2	PWMA_CC1	PWMA_UI

若要允许中断，除了使用 EI 指令开中断外，还要设置使能芯片 XFR 寄存器中对应的中断源。注意执行 DI 指令关某个中断源中断后，系统将不在记录这个中断源的中断事件（关全局中断不影响记录）。

1.8 如何添加函数库连接

若要使用函数库,用户必须在自己的程序中添加函数库连接,连接要使用的函数库文件,使自己的梯形图程序所在的文件夹中有该文件。在编程软件 EasyLad 中的操作如下:

- 点工程结构图中的“函数库连接”→“添加”,如图:



- 在弹出的“添加函数库连接”对话框中,点“其他函数库”按钮:



- 在弹出的“打开文件”对话框,找到需要的函数库文件打开即可,同时会自动把该

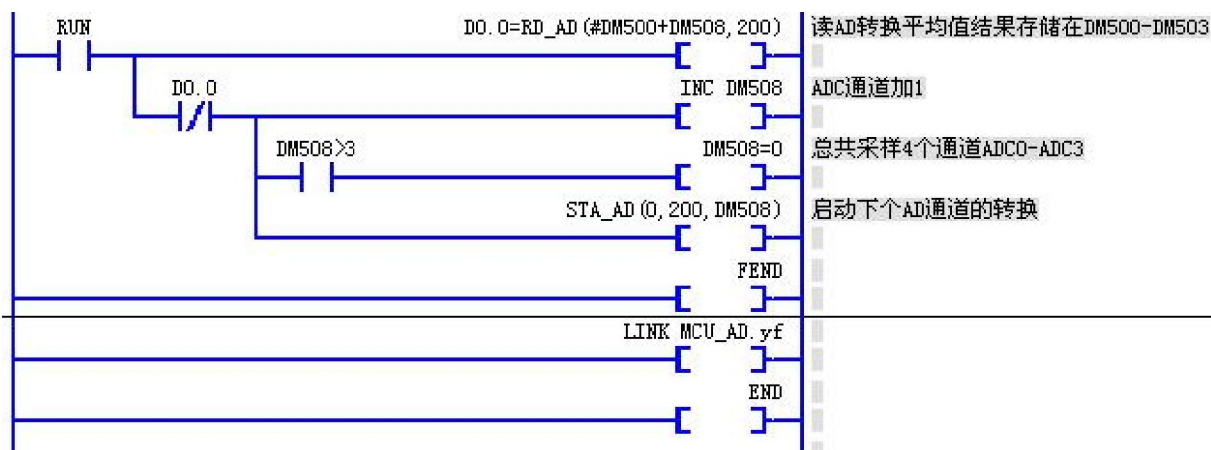
函数库文件复制到梯形图程序所在的文件夹中，若该函数库有所需的全局符号定义文件（扩展名为 `dfn` 的同名文件），则把该文件中的全局符号定义自动导入到用户的全局符号表中。

1.9 芯片 ADC 模数转换

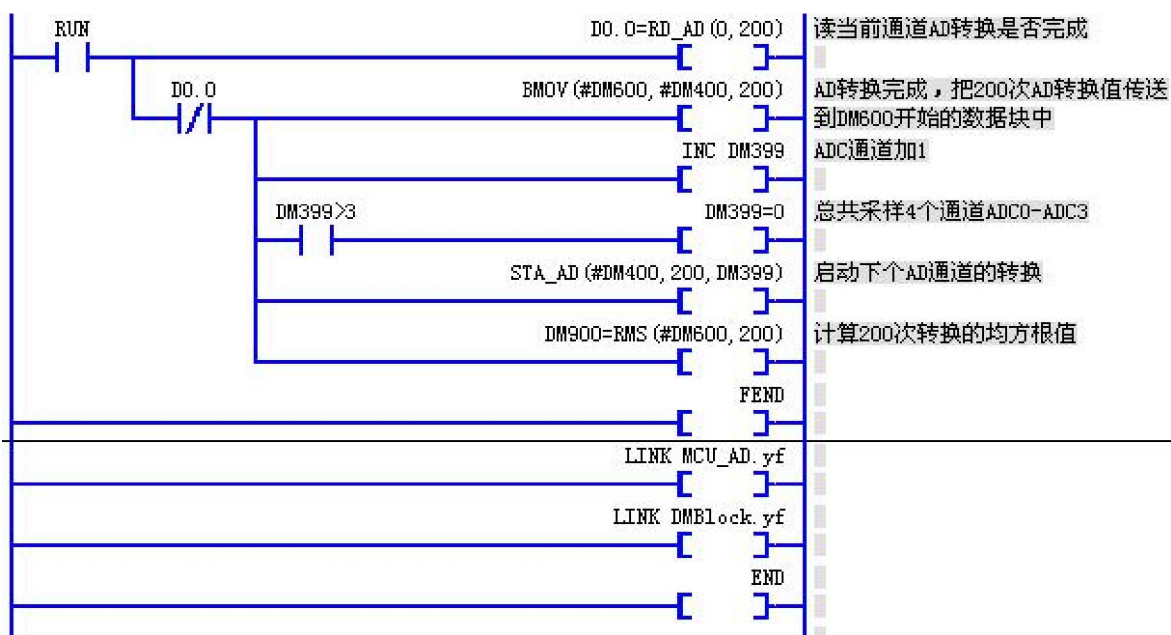
若要使用芯片 ADC 模数转换功能，则要连接芯片 ADC 模数转换函数库 `MCU_AD.yf`。

芯片 ADC 模数转换使用 PWMA 触发，若用户没有使用 PWMA，则由系统自动设置 PWMA 触发 ADC 所需要的配置，若用户使用了 PWMA，则由用户编程设置 PWMA 触发 ADC 所需要的配置。

按每通道采样 200 次平均值，读取 4 通道的梯形图例子如下：



按每通道采样 200 次进行存储，读取 4 通道的梯形图例子如下：



第二章 编程口 ModBus-RTU 通讯说明

2.1 编程口 ModBus-RTU 协议格式说明

1、字符格式

- 1 个起始位
- 8 个数据位
- 1 个偶校验位
- 1 个停止位

2、波特率

- 波特率默认为 115200，也可使用指令选择为 9600、19200、38400。注：编程口和扩展口的波特率始终保持相同，改变编程口的波特率也会改变扩展口的波特率。

3、帧格式

帧格式为标准的 Modbus-RTU 格式，一个帧中的最大寄存器个数为 64 个字、最大线圈个数为 256 个线圈。

注意：使用编程口 Modbus-RTU 协议时 PLC 的通讯地址不能设置为 2

用编程口和触摸屏通讯时，若触摸屏一次读写最大位和字数量能分开设置，则可把最大位数量设为 256 个位（16 个字）、最大字数量设为 64 个字，若触摸屏一次读写最大位和字数量不能分开设置，则一次读写最大数量应设为 16 个字。

2.2 Modbus-RTU 协议的设备地址和 PLC 内部元件对应表

设备地址与 PLC 元件对应一览表：

ModBus-RTU 设备		PLC 元件
设备类型	设备地址	
0X	1	M0
0X	2	M1
0X	3	M2
0X
0X	208	M207
0X	209	Y0
0X	210	Y1
0X	211	Y2
0X
0X	256	Y57

0X	257	X0
0X	258	X1
0X	259	X2
0X
0X	304	X57
4X	1	DM1024
4X	2	DM1025
4X	3	DM1026
4X
4X	256	DM1279
4X	257	DM256
4X	258	DM257
4X	259	DM258
4X
4X	1980	DM1979

1X 和 3X 不使用！

对于 YF0H 系列的非易失性数据存储器，也使用 4X 来读写，非易失性数据存储器 DM0～DM255 对应于 Modbus 中 4X 的设备地址 4X16385～4X16640，非易失性数据存储器的地址 K16640～…对应于 Modbus 中 4X 的设备地址 4X16641～…。

易失性数据存储器 DM256～…中的各个位可使用 0X 来读写，位地址为 **字号×16 + 位号 (0～15)**，对应于 Modbus 中 0X 的设备地址为 **位地址+1**。例如 DM300.5，位地址为 $300 \times 16 + 5 = 4805$ ，对应于 0X 的设备地址为 0X4806。

注：0X 使用功能码“01”来读取，可使用功能码“05”或“15”来修改。4X 使用功能码“03”来读取，使用功能码“06”或“16”来修改，或者使用功能码“23”来读取和修改。

2.3 Modbus-RTU 协议读写 PLC 内部的双字数据存储器

当用 Modbus 协议来读写 PLC 内部的双字整型数据存储器时，牵涉到一个存储格式问题，有 2 种存储格式：大端格式和小端格式。

大端格式为双字整型数据的高字存储在编号小的单元，低字存储在编号大的单元。例如 DM300 存高字，DM301 存低字。

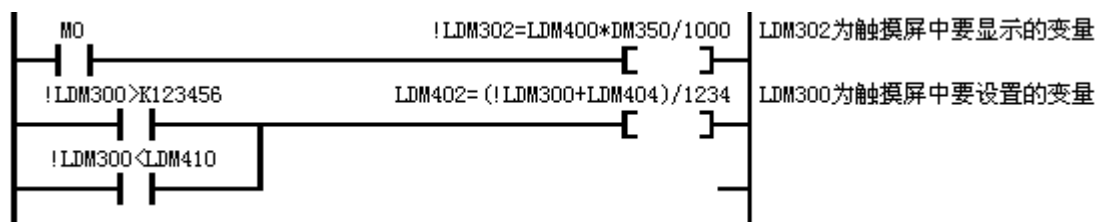
小端格式为双字整型数据的低字存储在编号小的单元，高字存储在编号大的单元。例如 DM300 存低字，DM301 存高字。

YF 系列 PLC 内部的双字整型数据存储器默认都采用大端格式，例如 LDM300 中，DM300

存储高字，DM301 存储低字。但在**表达式、函数和比较触点**中，也可采用小端格式来访问和存储，方法是使用前缀“!” + 双字数据存储器元件，例如“!LDM300”表示以小端格式来访问和存储 LDM300。

有许多触摸屏和文本显示器的 Modbus 协议都是以小端格式来读写双字整型数据，当与 YF 系列 PLC 连接时，触摸屏中所用到的双字整型数据，在 PLC 的程序中要使用前缀“!” + 双字数据存储器元件（例如“!LDM300”）。

例如触摸屏要设置 DM300、DM301 中的双字整型数据，同时要显示出 DM302、DM303 中的双字整型数据，则在 PLC 中的梯形图程序例子如下：



第三章 扩展通讯口通讯说明

3.1 自由协议通讯

扩展通讯口内置有 255 字节发送和 255 字节接收缓冲区（地址编号为 0~254），每帧数据最大可发送或接收 255 字节，CPU 模块通过向发送缓冲区写数据来设置要发送的内容，通过读接收缓冲区来读取接收到的数据。

用户需在自己的程序中连接扩展通讯口基础操作函数库 MCU_COMMx.yf。在该函数库中提供了使用扩展通讯口自由协议通讯所需要的函数。

在该函数库中提供的函数如下：

● 通讯设置函数：COMMxSet

函数定义：

FUN I, DEV_Addr As D0, Mode As D2, RxTim As D3

函数功能：

通讯口初始化、设置工作模式、波特率和接收字符间隔超时时间。

输入参数：

DEV_Addr: 设备地址（COM1 为 1）。

Mode: 工作模式和波特率选择，**位 2--0:** 波特率选择，011—9600，100—19200，101—38400，111—115200；**位 4--3:** 校验位选择，00—无校验，01—偶校验，10—奇校验，11—固定为 1；**位 5:** 数据位位数，0 为 8 位数据，1 为 7 位数据；**位 7:** 通讯口管脚选择，0 为 P5.2/P5.3，1 为 P0.2/P0.3；停止位固定为 1 位。若没有执行过该函数，则通讯口默认的波特率为 115200，无校验。

RxTim: 接收字符间隔超时时间（0~254），当设置为 0 时表示没有接收字符间隔超时检测。当接收到数据时将启动接收字符间隔超时检测，若在该时间内没有接收到下个数据，则置位接收字符间隔超时标记。

该函数调用例子：COMMxSet (1, H03, 10)

● 读通讯模块的状态字：COMMxFlag

函数定义：

FUN I, DEV_Addr As D0

函数功能：

读取通讯模块内的标记位：接收缓冲区校验比较结果标记、发送数据标记、接收字符间隔超时标记等。

输入参数：

DEV_Addr: 设备地址（COM1 为 1）。

返回值：

通讯模块内的各个标记位（状态字）：

位 0：发送数据标记，为 1 表示正在发送数据块，为 0 表示发送端口空闲，数据块发送完成或无数据发送。

位 1：接收缓冲区数据块校验比较结果标记，当执行 RXxSum、RXxXor 或 RXxCRC 函数后，为 1 表示接收缓冲区数据块的校验结果与收到的校验数据不相等，为 0 表示相等。该标记在执行 RXxSum、RXxXor 或 RXxCRC 函数后才有效。

位 2：接收字符间隔超时标记。若接收字符间隔超时时间不为 0，则当接收到数据时将启动接收字符间隔超时检测，若在该时间内没有接收到下个数据，则该标记置位。当执行接收数据指针复位函数 RXxRST(0)后该标记复位。

该函数调用例子：D0 = COMMxFlag (1)

- **读校验计算结果寄存器的值：COMMxVerW**

函数定义：

FUN I, DEV_Addr As D0

函数功能：

读取通讯模块内的校验计算结果寄存器的值。

输入参数：

DEV_Addr: 设备地址（COM1 为 1）。

返回值：

校验计算结果寄存器的值（字）：

说明：

当执行了数据块校验计算函数（TXxSum、TXxXor、TXxCRC、RXxSum、RXxXor、RXxCRC）后，会把计算结果（SUM 和 CRC-16 校验为 1 个字有效；XOR 校验为低字节有效、高字节为 0）放到校验计算结果寄存器中，用户使用该函数即可获得校验计算结果。

该函数调用例子：D0 = COMMxVerW (1)

- **发送缓冲区写字节数据（字符）：TXxChar**

函数定义:

FUN I, DEV_Addr As D0, TxBufAddr As D2, Val As D3

函数功能:

向发送缓冲区中指定的地址写一个字节的数据。

输入参数:

DEV_Addr: 设备地址 (COM1 为 1)。

TxBufAddr: 发送缓冲区中要写的地址 (0~254)。

Val: 要写入的数据, 其低字节被写入到发送缓冲区中。

该函数调用例子: TXxChar(1, 0, 123)

● **发送缓冲区写字数据 (2 个字节): TXxWord**

函数定义:

FUN I, DEV_Addr As D0, TxBufAddr As D2, Val As D3

函数功能:

向发送缓冲区中指定的地址写一个字 (2 个字节) 的数据。

输入参数:

DEV_Addr: 设备地址 (COM1 为 1)。

TxBufAddr: 发送缓冲区中要写的首地址 (0~254)。

Val: 要写入的数据, 其高字节被写入到 TxBufAddr 的地址, 低字节被写入到 TxBufAddr+1 的地址。

该函数调用例子: TXxWord (1, 0, 1234)

● **发送缓冲区写双字数据 (4 个字节): TXxDINT**

函数定义:

FUN I, DEV_Addr As D0, TxBufAddr As D2, Val As LDM256

函数功能:

向发送缓冲区中指定的地址写一个双字 (4 个字节) 的数据。

输入参数:

DEV_Addr: 设备地址 (COM1 为 1)。

TxBufAddr: 发送缓冲区中要写的首地址 (0~254)。

Val: 要写入的双字数据, 其高字的高字节被写入到 TxBufAddr 的地址, 高字的低字节被写入到 TxBufAddr+1 的地址, 低字的高字节被写入到 TxBufAddr+2 的地址, 低字的低字

节被写入到 TxBufAddr+4 的地址。

该函数调用例子：TXxDINT (1, 0, 12345678)

● 发送缓冲区写数据块：TXxData

函数定义：

FUN I, DEV_Addr As D0, TxBufAddr As D2, DM_Addr As D1, Len As D3

函数功能：

向发送缓冲区中指定的起始地址写一个 DM 数据块的数据。

输入参数：

DEV_Addr: 设备地址 (COM1 为 1)。

TxBufAddr: 发送缓冲区中要写的首地址 (0~254)，位 15 为 0：大端格式 (字数据高字节写入低地址、低字节写入高地址)、为 1：小端格式 (字数据低字节写入低地址、高字节写入高地址)。

DM_Addr: 要写入的 DM 数据块的首地址。

Len: 要写入的 DM 数据块的长度 (字数)。

该函数调用例子：TXxData(1,0,#DM300,10)

● 发送缓冲区求累加和校验：TXxSum

函数定义：

FUN I, DEV_Addr As D0, TxBufAddr As D2, Len As D3, OutAddr As D4

函数功能：

把发送缓冲区中指定的数据块进行累加和校验计算，计算结果放入到发送缓冲区中指定的地址和校验计算结果寄存器中。

输入参数：

DEV_Addr: 设备地址 (COM1 为 1)。

TxBufAddr: 发送缓冲区中要参与累加和计算的数据块的首地址 (0~254)。

Len: 发送缓冲区中要参与累加和计算的数据块的长度。

OutAddr: 校验计算结果 (低字节) 要放入到发送缓冲区中指定的地址 (0~254)。

该函数调用例子：TXxSum (1, 0, 10, 10)

● 发送缓冲区求异或和校验：TXxXor

函数定义：

FUN I, DEV_Addr As D0, TxBufAddr As D2, Len As D3, OutAddr As D4

函数功能:

把发送缓冲区中指定的数据块进行异或和校验计算, 计算结果放入到发送缓冲区中指定的地址和校验计算结果寄存器中。

输入参数:

DEV_Addr: 设备地址 (COM1 为 1)。

TxBufAddr: 发送缓冲区中要参与异或和计算的数据块的首地址 (0~254)。

Len: 发送缓冲区中要参与异或和计算的数据块的长度。

OutAddr: 校验计算结果 (低字节) 要放入到发送缓冲区中指定的地址 (0~254)。

该函数调用例子: TXxXor (1, 0, 10, 10)

● **发送缓冲区求 CRC-16 校验: TXxCRC**

函数定义:

FUN I, DEV_Addr As D0, TxBufAddr As D2, Len As D3, OutAddr As D4

函数功能:

把发送缓冲区中指定的数据块进行 CRC-16 (ModBus-RTU) 校验计算, 计算结果放入到发送缓冲区中指定的地址和校验计算结果寄存器中。

输入参数:

DEV_Addr: 设备地址 (COM1 为 1)。

TxBufAddr: 发送缓冲区中要参与 CRC-16 计算的数据块的首地址 (0~254)。

Len: 发送缓冲区中要参与 CRC-16 计算的数据块的长度。

OutAddr: 校验计算结果 (1 个字) 要放入到发送缓冲区中指定的地址 (0~254), 高字节放入到 OutAddr 的地址, 低字节放入到 OutAddr+1 的地址。

该函数调用例子: TXxCRC (1, 0, 10, 10)

● **发送数据: TXxOut**

函数定义:

FUN I, DEV_Addr As D0, TxBufAddr As D2, Len As D3

函数功能:

把发送缓冲区中指定的数据块的数据从通讯端口发送出去。

输入参数:

DEV_Addr: 设备地址 (COM1 为 1)。

TxBufAddr: 要发送的数据块的首地址 (0~127)。从首地址开始依次按字节发送。

Len: 要发送的数据块的长度（按字节）。

注：若 TxBufAddr 为 -1 并且 Len 为 0，则按上次发送的首地址和数据长度重新发送，例如 TXxOut (1, -1, 0) 重新发送上次发送的数据。

该函数调用例子：TXxOut (1, 0, 12)

- **读接收端口接收到的字节数：RXxNum**

函数定义：

FUN I, DEV_Addr As D0

函数功能：

读取接收端口接收到的字节数。

输入参数：

DEV_Addr: 设备地址（COM1 为 1）。

返回值：

接收端口接收到的字节数，为 0 表示还没接收到数据，为其他表示已经接收到的数据的字节数。该数据可通过执行 RXxRST 函数复位为 0。

注：接收端口接收到的数据按先后顺序依次存放在从地址 0 开始的接收缓冲区中。

该函数调用例子：D0 = RXxNum (1)

- **接收端口接收数据指针复位函数：RXxRST**

函数定义：

FUN I, DEV_Addr As D0

函数功能：

把接收数据存放指针和接收字符间隔超时标记复位，同时也把接收到的字节数复位为 0，表示接收缓冲区要从地址 0 开始重新接收数据。

输入参数：

DEV_Addr: 设备地址（COM1 为 1）。

该函数调用例子：RXxRST (1)

- **按字节读接收缓冲区中的数据：RXxChar**

函数定义：

FUN I, DEV_Addr As D0, RxBufAddr As D2

函数功能：

按字节读取接收缓冲区中指定位置的数据。

输入参数:

DEV_Addr: 设备地址 (COM1 为 1)。

RxBufAddr: 接收缓冲区中要读的数据的地址 (0~254)。

返回值:

接收缓冲区中指定位置的数据 (1 个字节)。

该函数调用例子: D0 = RXxChar (1, 0)

● **按字读接收缓冲区中的数据: RXxWord**

函数定义:

FUN I, DEV_Addr As D0, RxBufAddr As D2

函数功能:

按字读取接收缓冲区中指定位置的数据。

输入参数:

DEV_Addr: 设备地址 (COM1 为 1)。

RxBufAddr: 接收缓冲区中要读的字数据的首地址 (0~254)。

返回值:

接收缓冲区中指定位置的数据 (1 个字), 地址 RxBufAddr 中的数据为高字节, 地址 RxBufAddr+1 中的数据为低字节。

该函数调用例子: D0 = RXxWord (1, 10)

● **按双字读接收缓冲区中的数据: RXxDINT**

函数定义:

FUN I, DEV_Addr As D0, RxBufAddr As D2

函数功能:

按双字读取接收缓冲区中指定位置的数据。

输入参数:

DEV_Addr: 设备地址 (COM1 为 1)。

RxBufAddr: 接收缓冲区中要读的双字数据的首地址 (0~254)。

返回值:

接收缓冲区中指定位置的数据 (双字), 地址 RxBufAddr 中的数据为其高字的高字节, 地址 RxBufAddr+1 中的数据为高字的低字节, 地址 RxBufAddr+2 中的数据为低字的高字节, 地址 RxBufAddr+3 中的数据为低字的低字节。

该函数调用例子：LDM300 = RXxDINT (1, 10)

● 接收缓冲区读数据块：RXxDATA

函数定义：

FUN I, DEV_Addr As D0, RxBufAddr As D2, DM_Addr As D1, Len As D3

函数功能：

读取接收缓冲区中指定起始位置的数据块到 DM 数据块中。

输入参数：

DEV_Addr: 设备地址（COM1 为 1）。

RxBufAddr: 接收缓冲区中要读数据块的首地址（0~254），位 15 为 0：大端格式（低地址为字的高字节、高地址为字的低字节）、为 1：小端格式（低地址为字的低字节、高地址为字的高字节）。

DM_Addr: 要读到的 DM 数据块的首地址。

Len: 要读到的 DM 数据块的长度（字数）。

该函数调用例子：RXxDATA(1,0,#DM300,10)

● 接收缓冲区求累加和校验：RXxSUM

函数定义：

FUN I, DEV_Addr As D0, RxBufAddr As D2, Len As D3, CMPAddr As D4

函数功能：

把接收缓冲区中指定的数据块进行累加和校验计算，计算结果放入到校验计算结果寄存器中，同时和接收缓冲区中指定的位置的接收到校验数据进行比较。

输入参数：

DEV_Addr: 设备地址（COM1 为 1）。

RxBufAddr: 接收缓冲区中要参与累加和计算的数据块的首地址（0~254）。

Len: 接收缓冲区中要参与累加和计算的数据块的长度。

CMPAddr: 校验计算结果（低字节）要比较的校验数据的地址（0~254）。比较结果可由 Comm1Flag 函数读取。

该函数调用例子：RXxSUM (1, 0, 10, 10)

● 接收缓冲区求异或和校验：RXxXOR

函数定义：

FUN I, DEV_Addr As D0, RxBufAddr As D2, Len As D3, CMPAddr As D4

函数功能:

把接收缓冲区中指定的数据块进行异或和校验计算, 计算结果放入到校验计算结果寄存器中, 同时和接收缓冲区中指定的位置的接收到校验数据进行比较。

输入参数:

DEV_Addr: 设备地址 (COM1 为 1)。

RxBufAddr: 接收缓冲区中要参与异或和计算的数据块的首地址 (0~254)。

Len: 接送缓冲区中要参与异或和计算的数据块的长度。

CMPAddr: 校验计算结果 (低字节) 要比较的校验数据的地址 (0~254)。比较结果可由 Comm1Flag 函数读取。

该函数调用例子: RXxXor (1, 0, 10, 10)

● **接收缓冲区求 CRC-16 校验: RXxCRC**

函数定义:

FUN I, DEV_Addr As D0, RxBufAddr As D2, Len As D3, CMPAddr As D4

函数功能:

把接收缓冲区中指定的数据块进行 CRC-16 (ModBus-RTU) 校验计算, 计算结果放入到校验计算结果寄存器中, 同时和接收缓冲区中指定的位置的接收到校验数据进行比较。

输入参数:

DEV_Addr: 设备地址 (COM1 为 1)。

RxBufAddr: 接收缓冲区中要参与 CRC 计算的数据块的首地址 (0~254)。

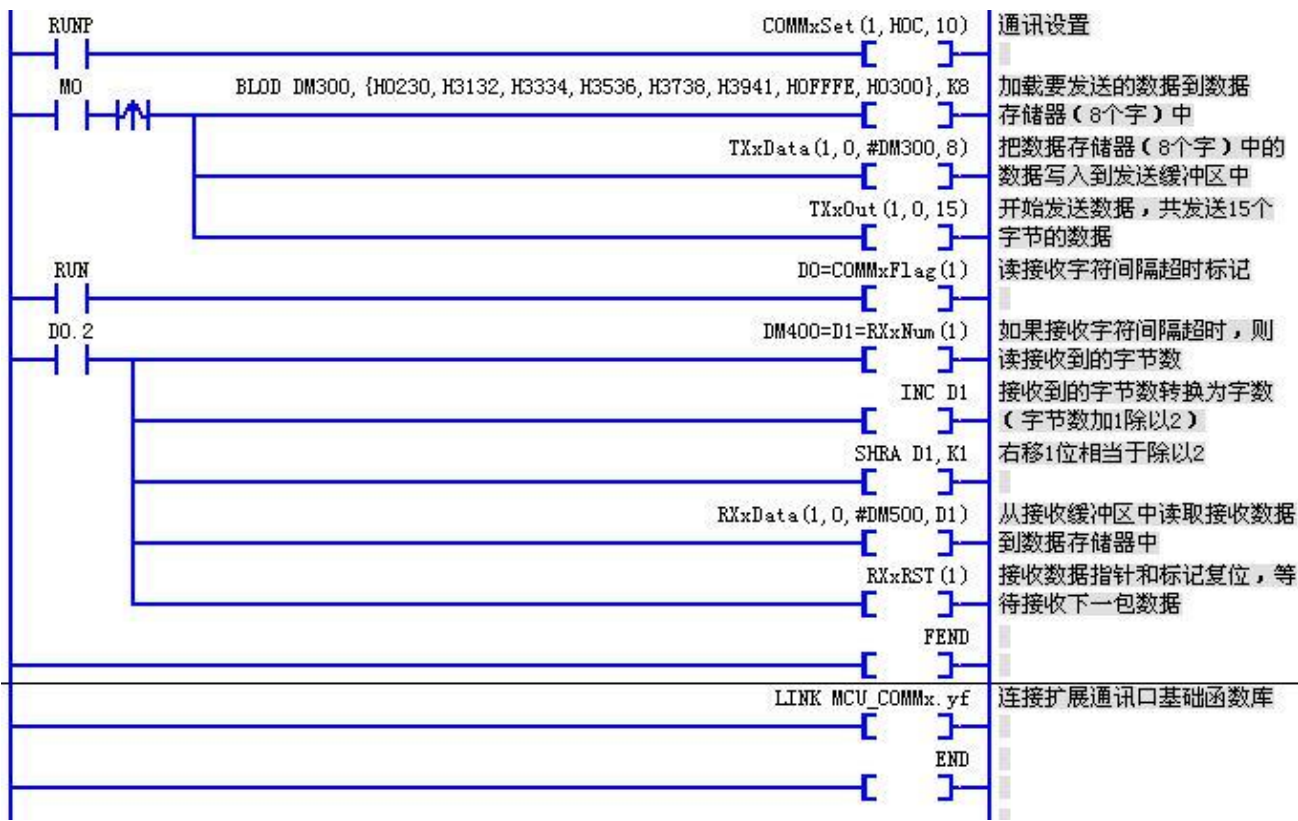
Len: 接送缓冲区中要参与 CRC 计算的数据块的长度。

CMPAddr: 校验计算结果 (字) 要比较的校验数据 (2 个字节) 的首地址 (0~254)。

比较结果可由 Comm1Flag 函数读取。

该函数调用例子: RXxCRC (1, 0, 10, 10)

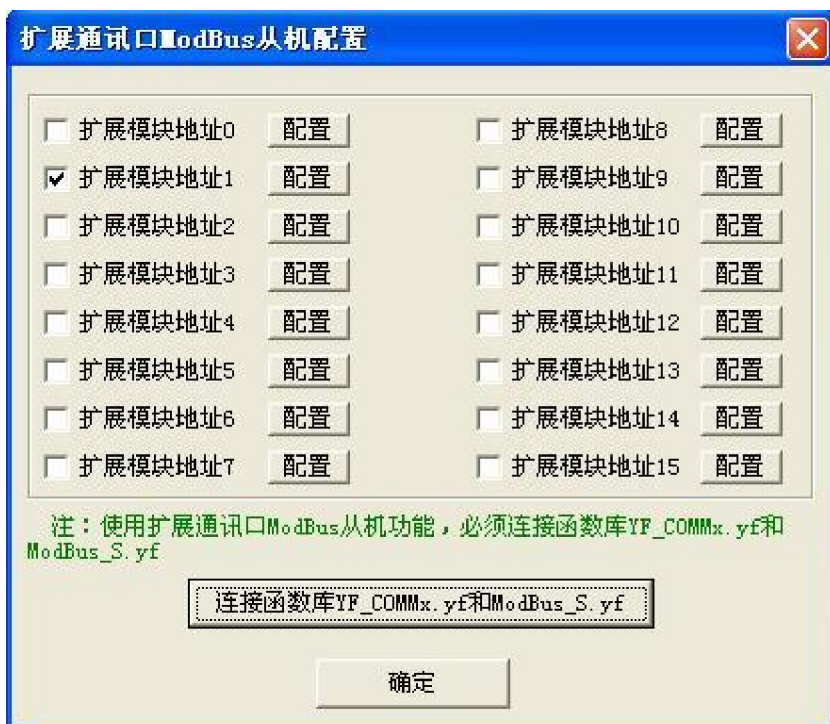
扩展通讯口自由协议通讯梯形图例子：



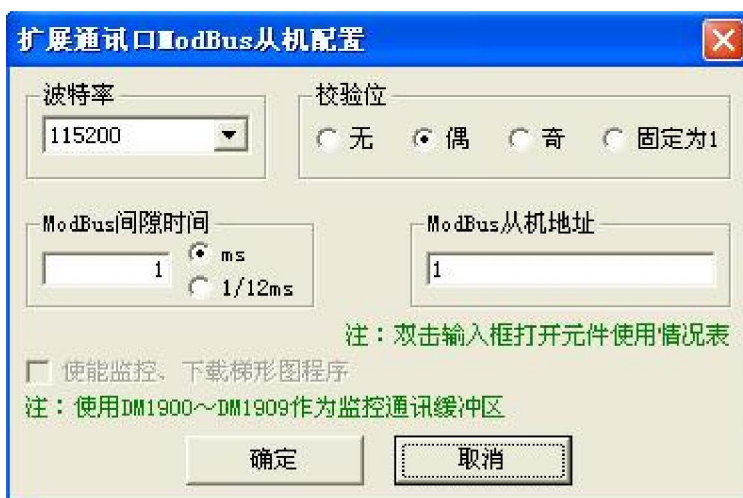
3.2 扩展通讯口 ModBus-RTU 从机配置

若要使用扩展通讯口 ModBus-RTU 从机通讯功能，需进行以下操作：

- (1) 连接函数库 MCU_COMMx.yf 和 MCU_ModBus_S.yf。
- (2) 在梯形图编程软件中点“配置”菜单→点其中的“扩展通讯口 ModBus-RTU 从机配置”菜单，弹出以下对话框：



在该对话框中选择扩展模块地址 1，点其右边的配置按钮（注意不要点“连接函数库 YF_COMMx.yf 和 ModBus_S.yf”按钮），弹出以下对话框：



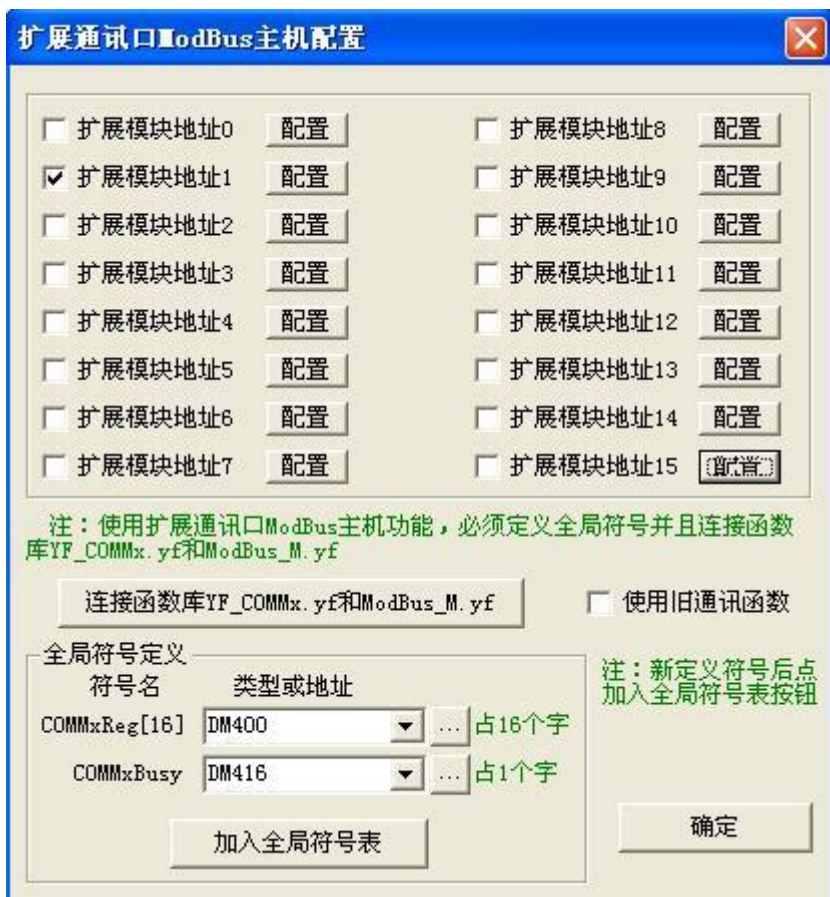
在该对话框中配置完点确定按钮即可。波特率可选 9600、19200、38400、115200，当改变该波特率时也会改变编程口的波特率，两者波特率始终相同。ModBus 间隙时间和波特

率有关，波特率 9600 时建议为 4ms，19200 时建议为 2 ms，38400 和 115200 时建议为 1 ms。注：ModBus 间隙时间中的选择按钮选择为 ms 时通讯管脚为 P5.2/P5.3，选择为 1/12ms 时通讯管脚为 P0.2/P0.3

3.3 扩展通讯口 ModBus-RTU 主机配置

若要使用扩展通讯口 ModBus-RTU 主机通讯功能，需进行以下操作：

- (1) 连接函数库 MCU_COMMx.yf 和 MCU_ModBus_M.yf。
- (2) 在梯形图编程软件中点“配置”菜单→点其中的“扩展通讯口 ModBus-RTU 主机配置”菜单，弹出以下对话框：



若程序中没有定义全局符号 COMMXReg[16]和 COMMXBusy（没定义则“类型或地址”输入框中显示空，已定义则会显示定义的内容），则要在上面的对话框中各自的“类型或地址”输入框中输入元件类型或元件名（均为 DM 元件或 WORD 类型），然后点“加入全局符号表”按钮把这两个符号定义加入到全局符号表中。若已经定义过这两个符号则不要再点“加入全局符号表”按钮。若要修改已经定义过的这两个符号的类型或地址，则要到全局符号表中修改。

在该对话框中选择扩展模块地址 1，点其右边的配置按钮（注意不要点“连接函数库 YF_COMMx.yf 和 ModBus_M.yf”按钮），弹出以下通讯配置对话框：

扩展通讯口ModBus主机配置

波特率

19200

校验位

☐ 无

☒ 偶

☐ 奇

☐ 固定为1

ModBus间隙时间

5

ms

☐ 1/12ms

循环动作控制位

通讯无响应设置

使用定时器

T0

无响应时间

100 × 10ms

无响应输出位

M0

报告当前正在执行的通讯命令对应的控制位地址

DM417

☒ 通讯无响应时重试

报告当前正在通讯的站号

☒ 通讯无响应时初始化通讯口设置

通讯命令表

参数说明：

控制位	站号	功能码	主机读数据块	读数据块长度	读地址	主机写数据块	写数据块长度	写地址	动作
M10	1	16				DM300	10	0X1000	1
M11	2	16				DM310	10	0X1000	1
M12	3	16				DM320	10	0X1000	1
M13	4	16				DM330	10	0X1000	1
M20	1	05				M100		10	1
M21	2	05				M101		10	1
M22	1	06				DM340		500	1
M23	2	06				DM341		500	1
M24	1	03	DM400	10	0X2000				0
M25	2	03	DM410	10	0X2000				0
M26	3	03	DM420	10	0X2000				0
M26	4	03	DM430	10	0X2000				0

添加

插入

删除

复制

粘贴

注：双击输入框打开元件使用情况表

确定

取消

注：该对话框中 ModBus 间隙时间中的选择按钮选择为 ms 时通讯管脚为 P5.2/P5.3，选择为 1/12ms 时通讯管脚为 P0.2/P0.3

各个参数说明如下：

【循环动作控制位】：为 M 继电器或 DMx.y 位格式，该位为 ON 时执行通讯命令动作，为 OFF 时暂停执行通讯命令动作。空表示通讯命令动作不受该参数控制。

通讯无响应设置参数：

【使用定时器】：为内部 T 定时器，用于通讯无响应定时。

【无响应时间】：为数值，超出该时间没收到正确的从机应答则认为从机无响应。

【无响应输出位】：为 M 继电器或 DMx.y 位格式，通讯无响应时使该位置位一个扫描周期。空表示没有无响应输出位。

【报告当前正在执行的通讯命令对应的控制位地址】：为 DM 存储器。空表示不需要该参数。若使用该参数，则通讯命令表中的触发动作通讯命令控制位不能为 M0，并且当通讯确认完成时控制位才清 0。

【报告当前正在通讯的站号】：为 DM 存储器。空表示不需要该参数。

通讯命令表参数：

【控制位】：为 M 继电器或 DMx.y 位（ $x < 4096$ ）格式，用于控制该条通讯命令，若几个控制位同时为 ON，则按上下顺序依次执行各个命令。若使用“报告当前正在执行的通讯命令对应的控制位地址”参数，则当该条通讯命令确认完成时控制位才清 0，若不使用，则当发送该条通讯命令时控制位清 0。

【站号】：为数值，从机的通讯地址。

【功能码】：01:读线圈，02:读输入线圈，03:读寄存器，04:读输入寄存器，05:写单线圈，06:写单寄存器，15:写多线圈，16:写多寄存器，23:读写多寄存器。

【主机读数据块】：为 DM 存储器。若功能码为 01 或 02，则该参数为主机读线圈的 DM 元件(占 1 个字)，为主机中存储读入的线圈的状态。若功能码为 03 或 04 或 23，则该参数为主机读数据块的起始 DM 元件，起始元件～以后元件为主机中存储读入的从机数据块的数据。其他功能码不需要该参数。

【读数据块长度】：为数值。若功能码为 01 或 02，则该参数为主机读线圈的线圈个数，最多为 16 个。若功能码为 03 或 04 或 23，则该参数为读数据中的主机 DM 数据块的长度(最大为 121)，该值为要从从机中读的数据个数(字)，例如要从从机中读 8 个字则要把该参数设为 8。其他功能码不需要该参数。

【读地址】：为数值或 DM 存储器，也可为 16 进制数值格式 H0xxxx。若功能码为 01 或 02，则该参数为读线圈中的从机线圈起始地址。若功能码为 03 或 04 或 23，则该参数为读数据中的从机寄存器起始地址。其他功能码不需要该参数。

【主机写数据块】：根据功能码不同可为数值、M 继电器或 DMx.y 位格式、DM 存储器。若功能码为 05，则该参数为写单线圈的线圈值（0 或 1 或 M 继电器或 DMx.y 位格式）。若功能码为 06，则该参数为写单寄存器的寄存器值（数值或 DM 存储器）。若功能码为 15，则该参数为主机写多线圈的 DM 元件(占 1 个字)，为主机中存储要写入的多线圈的状态。若功能码为 16 或 23，则该参数为主机写数据块的起始 DM 元件，起始元件～…为主机中要写入到从机的数据块数据。其他功能码不需要该参数。

【写数据块长度】：为数值。若功能码为 15，则该参数为主机写多线圈的线圈个数，最

多为 16 个。若功能码为 16 或 23，则该参数为写数据中的主机 DM 数据块的长度（最大为 121），该值为要写入到从机中的数据个数(字)，例如要向从机中写入 8 个字则要把该参数设为 8。其他功能码不需要该参数。

【写地址】：为数值或 DM 存储器，也可为 16 进制数值格式 H0xxxx。若功能码为 05 或 15，则该参数为写线圈中的从机线圈起始地址。若功能码为 16 或 23，则该参数为写数据中的从机寄存器起始地址。其他功能码不需要该参数。

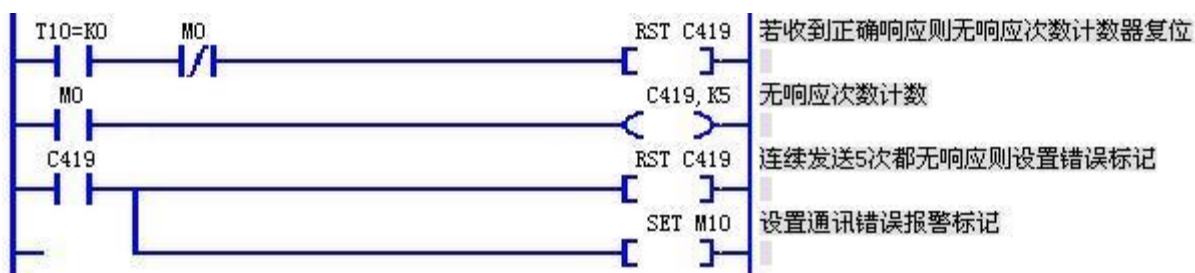
【动作】：为 0 或 1。通讯命令动作方式，0--循环，由系统控制周而复始自动执行，用户不需控制；1--触发，控制位为 ON 时执行 1 次，控制位自动复位。触发优先级高于循环，即若有触发动作的控制位为 ON，则暂停循环动作，去执行这些控制位为 ON 的触发动作，待这些触发动作都执行完后再继续执行循环动作。

通讯无响应重试：

若要检测通讯无响应，则必须要设置“使用定时器”和“无响应时间”两个参数。

若“通讯无响应时重试”选项没有选择，则当通讯无响应时不会重试该条通讯命令，若需要通讯无响应时重试该条通讯命令，则要选择“通讯无响应时重试”选项，并且设置“无响应输出位”和“报告当前正在执行的通讯命令对应的控制位地址”两个参数。

下面程序无论循环通讯还是触发通讯，每连续 5 次无响应时报警（程序中 M0 为“无响应输出位”，T10 为“通讯无响应定时器”）：



上面程序在通讯无响应时会一直重试下去，若需要 5 次重试后不再重试、继续执行下面的通讯命令，则应在无响应次数计数到时复位当前正在通讯的控制位（例如“报告当前正在执行的通讯命令对应的控制位地址”为 DM417，则执行函数 `WrCoil(0,DM417,1)`即可复位当前正在通讯的控制位）。

第四章 以太网通讯说明

4.1 以太网 ModBusTCP 服务器通讯配置

若要使用以太网 ModBusTCP 服务器通讯功能，需进行以下操作：

- (1) 连接函数库 MCU_ENET.yf 和 MCU_ModBusTCP_S.yf。
- (2) 在梯形图编程软件中点“配置”菜单→点其中的“以太网通讯模块配置”菜单，

弹出以下对话框：



以太网通讯模块配置

☒ 使能以太网通讯模块 连接函数库ENET.yf 注：使用以太网通讯模块，必须连接函数库ENET.yf

本机IP地址: 192. 168. 1. 253 网关IP地址: 192. 168. 1. 1 子网掩码: 255. 255. 255. 0

Socket0
☒ 使能 本机端口号: 502 目标IP地址: . . . 目标端口号: ModBusTcp 服务器配置

Socket1
☒ 使能 本机端口号: 502 目标IP地址: . . . 目标端口号: ModBusTcp 服务器配置

Socket2
☒ 使能 本机端口号: 502 目标IP地址: . . . 目标端口号: ModBusTcp 服务器配置

Socket3
☒ 使能 本机端口号: 502 目标IP地址: . . . 目标端口号: ModBusTcp 服务器配置

Socket4
☐ 使能 本机端口号: 502 目标IP地址: . . . 目标端口号: ModBusTcp 服务器配置

Socket5
☐ 使能 本机端口号: 502 目标IP地址: . . . 目标端口号: ModBusTcp 服务器配置

Socket6
☐ 使能 本机端口号: 502 目标IP地址: . . . 目标端口号: ModBusTcp 服务器配置

Socket7
☐ 使能 本机端口号: 1031 目标IP地址: . . . 目标端口号: ModBusTcp 服务器配置
☐ 使能监控、下载梯形图程序 用于监控通讯缓冲区的起始数据存储区(占10个字): DM1900

确定 取消

在该对话框中：

“使能以太网通讯模块”选项选中。不要点“连接函数库 ENET.yf”按钮。

设置本机 IP 地址、网关 IP 地址、子网掩码。这些参数可以为常数，也可以为掉电保持的 DM 数据存储区（若在运行中改变这些参数并立即生效，则在这些参数改变后执行 ENE T_Init 函数）。

需要连接几个客户端就要使能几个 Socket。

然后点使能的“ModBusTCP 服务器配置”按钮，弹出以下对话框：



在该对话框中，不要点“连接函数库 ModBusTCP_S.yf”按钮，设置空闲时间定时器（要选择没有使用过的 PLC 内部定时器），设置最大空闲时间，然后点确定按钮即可。

第五章 脉冲串定位控制应用说明

线性加减速脉冲串单轴输出最高 400KHz，双轴同时输出最高 200KHz。注：当使用了高速计数器 C0、C1 或外部中断时会降低脉冲串输出的最高频率。

使用脉冲串输出函数库 **PTO_0L.yf**（Y0 的线性加减速脉冲串输出）、**PTO_1L.yf**（Y1 的线性加减速脉冲串输出）来完成定位控制。

下面说明是对于 Y0 轴的，Y1 轴的类型（把说明中的 0 换成 1 即可）。

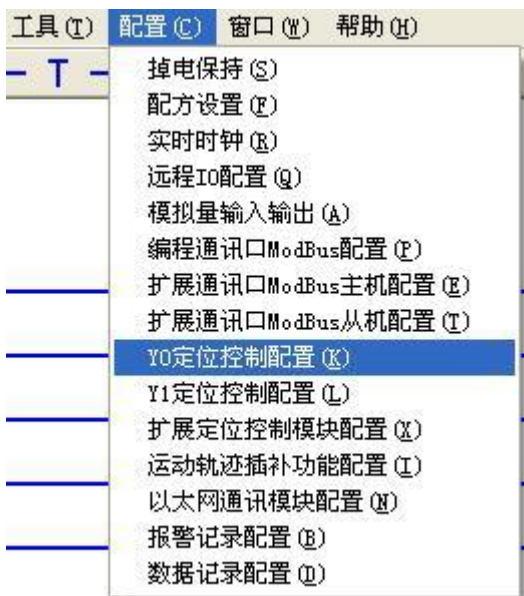
5.1 配置 Y0 定位控制功能

在梯形图中要进行以下操作：

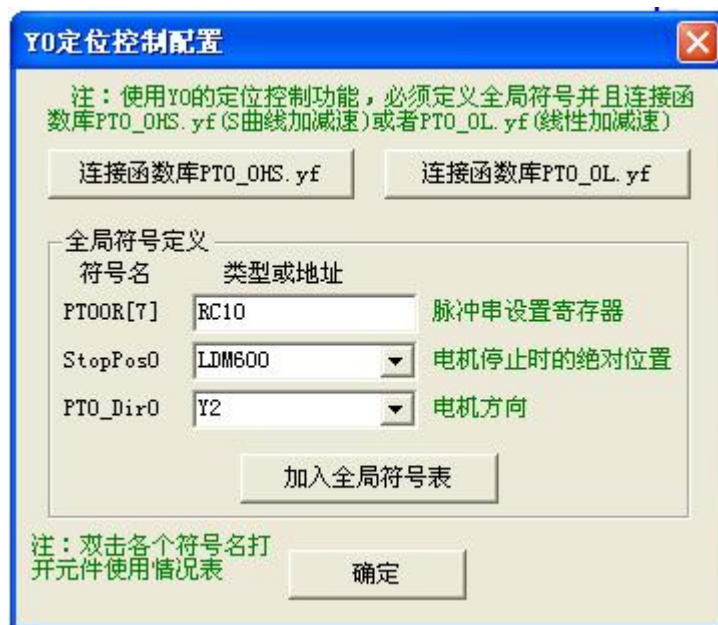
第 1 步：进入 Y0 定位控制配置界面

在编程软件 EasyLad 中的操作如下：

- 点击配置菜单，弹出以下菜单内容：



- 点击“Y0 定位控制配置”菜单，弹出 Y0 定位控制配置对话框，如下：



Y0 定位控制配置界面

第 2 步：连接 Y0 定位函数库并定义函数库所需的全局符号

首先在 Y0 定位控制配置对话框中点击“连接函数库 PTO_OL.yf”按钮，连接 Y0 的定位函数库。

然后在全局符号定义栏定义以下全局符号：

PTO0R[7]：类型为 RC。Y0 的脉冲串设置寄存器，占用 7 个单元，推荐使用默认值。

StopPos0：类型为 LDM 或 DINT。电机停止时的绝对位置（单位：脉冲个数）。

PTO_Dir0：类型为 Y。电机方向，为 OFF 远离原点，为 ON 接近原点（坐标 $0 \sim +\infty$ ）。

可使用 Y0~Y7，表示选择普通输出 Y0~Y7 作为 Y0 定位的方向控制。

全局符号定义设置完成后，点击“加入全局符号表”按钮，把定义的内容加入到全局符号表中。

用户若需要在停电后记住电机的绝对位置，则应对 StopPos0 进行掉电保存（必须在电机停止后断电才能准确的记住绝对位置）。

PTO_OL.yf 中有 6 个函数供用户定位控制使用（PTO_1L.yf 函数库类同）。如下：

5.2 绝对定位函数 DRVA0

函数定义：

FUN I, Pos As LDM256, RunFre As LDM258, Accel As D1

函数功能:

对电机按绝对位置来进行定位。

输入参数:

Pos: 电机要移动到的绝对位置（脉冲个数）。

RunFre: 电机的恒速运行脉冲频率（62~200000Hz）。

Accel: 加速度，单位：Hz/ms。

说明:

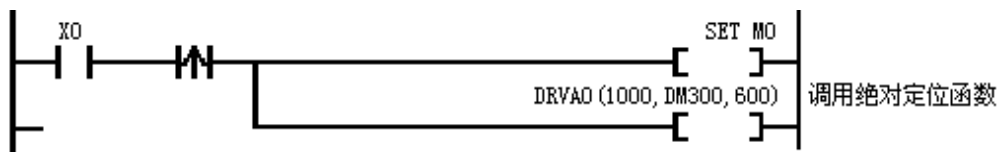
执行该函数后脉冲串输出完成标记 PTO0F 被复位，到位置后 PTO0F 被置位。

注：执行该函数时电机必须处于停止状态（无脉冲输出），执行该函数后电机立刻运行（输出脉冲）。

例如：

DRVA0(1000,5000,500)表示把电机移动到绝对位置 1000 处，恒速运行脉冲频率为 5000Hz，加速度为 500Hz/ms。

梯形图例子如下：



5.3 相对定位函数 DRVIO

函数定义:

FUN I, PNum As LDM256, RunFre As LDM258, Accel As D1

函数功能:

对电机按相对位置来进行定位。

输入参数:

Pos: 电机要移动到的相对位置（相对于当前位置的脉冲个数），为正数表示电机远离原点，为负数表示接近原点。

RunFre: 电机的恒速运行脉冲频率（62~200000Hz）。

Accel: 加速度，单位：Hz/ms。

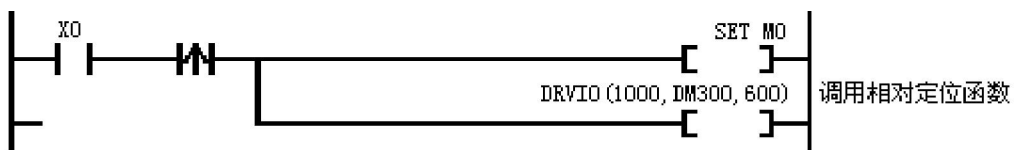
说明:

执行该函数后脉冲串输出完成标记 PTO0F 被复位，到位置后 PTO0F 被置位。

注：执行该函数时电机必须处于停止状态（无脉冲输出），执行该函数后电机立刻运行（输出脉冲）。

例如:

DRVIO(1000,5000,500)表示把电机移动到相对于当前位置 1000 处，恒速运行脉冲频率为 5000Hz，加速度为 500Hz/ms。

梯形图例子如下:

5.4 电机停止函数 DRVS0

函数定义:

FUN I, StopMode As D0

函数功能:

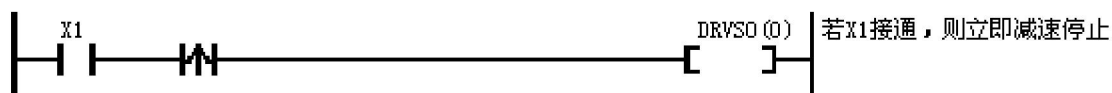
通过程序来使电机停止，执行该函数后电机立刻进行减速停止。

输入参数:

StopMode: 无意义，可为任意值。

说明:

当电机减速停止后脉冲串输出完成标记 PTO0F 被置位。

梯形图例子如下:

5.5 读出电机的当前位置函数 RdCPos0

函数定义:

FUN L, Void As D0

函数功能:

返回电机的当前绝对位置（脉冲个数），无论电机处于运行还是停止状态。

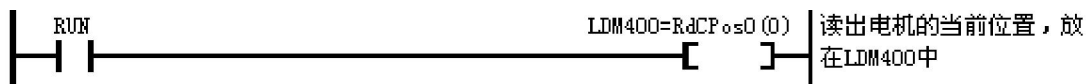
输入参数：

Void: 无意义，可为任意值。

返回值：

电机的当前绝对位置（脉冲个数）。

梯形图例子如下：



5.6 变速函数 DRVVO

函数定义：

FUN I, VelFre As LDM256

函数功能：

在电机运行过程中改变运行速度。

输入参数：

VelFre: 新的运行速度频率。

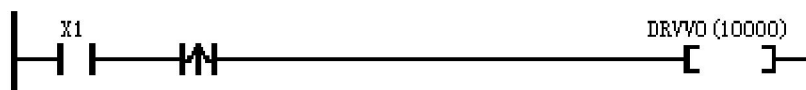
说明：

若新速度大于原速度则加速，若新速度小于原速度则减速，加速度为启动电机时设置的加速度。电机停止时会自动减速停止。

注：若电机在减速停止阶段时不能执行该函数。

例如：DRVVO(10000)表示把正在运行的 Y0 轴电机的速度变为 10000Hz，带加减速。

梯形图例子如下：



5.7 点动函数 JOG0

函数定义：

FUN I, JGF As D2.0, JGR As D3.0, RunFre As LDM256, Accel As D4

函数功能：

对电机进行点动。

输入参数:

JGF: 正转点动输入, ON 时正转, OFF 时停止。

JGR: 反转点动输入, ON 时反转, OFF 时停止。

RunFre: 电机的恒速运行脉冲频率 (62~200000Hz)。

Accel: 加速度, 单位: Hz/ms。

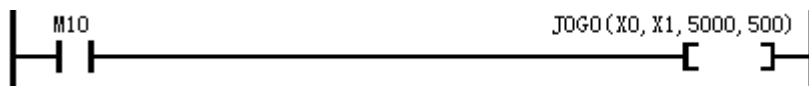
说明:

JGF 和 JGR 同时只能有一个为 ON, 当都为 OFF 时才使电机停止。该函数接通则执行点动功能, 断开则不执行点动功能。程序中可以多处使用, 但同时只能有一个接通。

例如:

JOG0(X0,X1,5000,500)表示对 Y0 轴电机点动, X0 为正向点动, X1 为反向点动, 运行频率为 5000Hz, 加速度为 500Hz/ms。

梯形图例子如下: (当 M10 为 ON 时允许点动)



5.8 读脉冲串内部寄存器函数 GetPTOR

函数定义:

FUN I, REG As D0

函数功能:

根据输入的寄存器号读对应的脉冲串内部寄存器的值作为函数返回值。

输入参数:

REG: 要读的脉冲串内部寄存器的编号, 0: Y0 的剩余脉冲个数, 1: Y1 的剩余脉冲个数, 2: Y0 的加速段脉冲个数, 3: Y1 的加速段脉冲个数, 4: 电机运行时加减速状态字(位 1:Y0 加速状态标记,位 2:Y0 减速状态标记,位 4:Y1 加速状态标记,位 5:Y1 减速状态标记)。

返回值:

要读的寄存器的值。

例如: LDM300=GetPTOR(0) 把 Y0 的剩余脉冲个数送到 LDM300 中。

5.9 设置 Y0 的剩余脉冲个数寄存器函数 SetPTOR0

函数定义:

FUN I, Val As LDM256

函数功能:

设置 Y0 的剩余脉冲个数寄存器。

输入参数:

Val: 要设置的剩余脉冲个数。

例如: SetPTOR0(1000000)。

5.10 设置 Y1 的剩余脉冲个数寄存器函数 SetPTOR1

函数定义:

FUN I, Val As LDM256

函数功能:

设置 Y1 的剩余脉冲个数寄存器。

输入参数:

Val: 要设置的剩余脉冲个数。

例如: SetPTOR1(1000000)。

5.11 设置脉冲频率的输出范围和分辨率

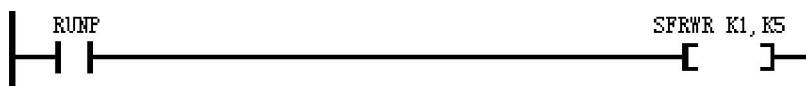
上电后默认的脉冲频率输出范围为 62~200000Hz, 分辨率为 0.25us, 也可以使用指令改变脉冲频率的下限值和分辨率, 如下:

指令 **SFRWR K1,K5** 设置脉冲频率输出范围为 21~200000Hz, 分辨率为 0.75us, 频率下限值较低, 但高频时频率误差较大。

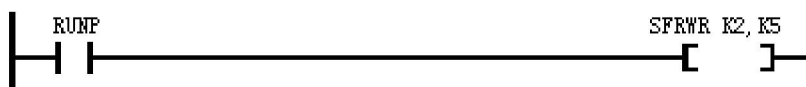
指令 **SFRWR K2,K5** 设置脉冲频率输出范围为 250~200000Hz, 分辨率为 0.0625us, 频率下限值较高, 但高频时频率误差较小。

梯形图例子如下:

设置脉冲频率输出范围为 21~200000Hz 梯形图例子:



设置脉冲频率输出范围为 250~200000Hz 梯形图例子:

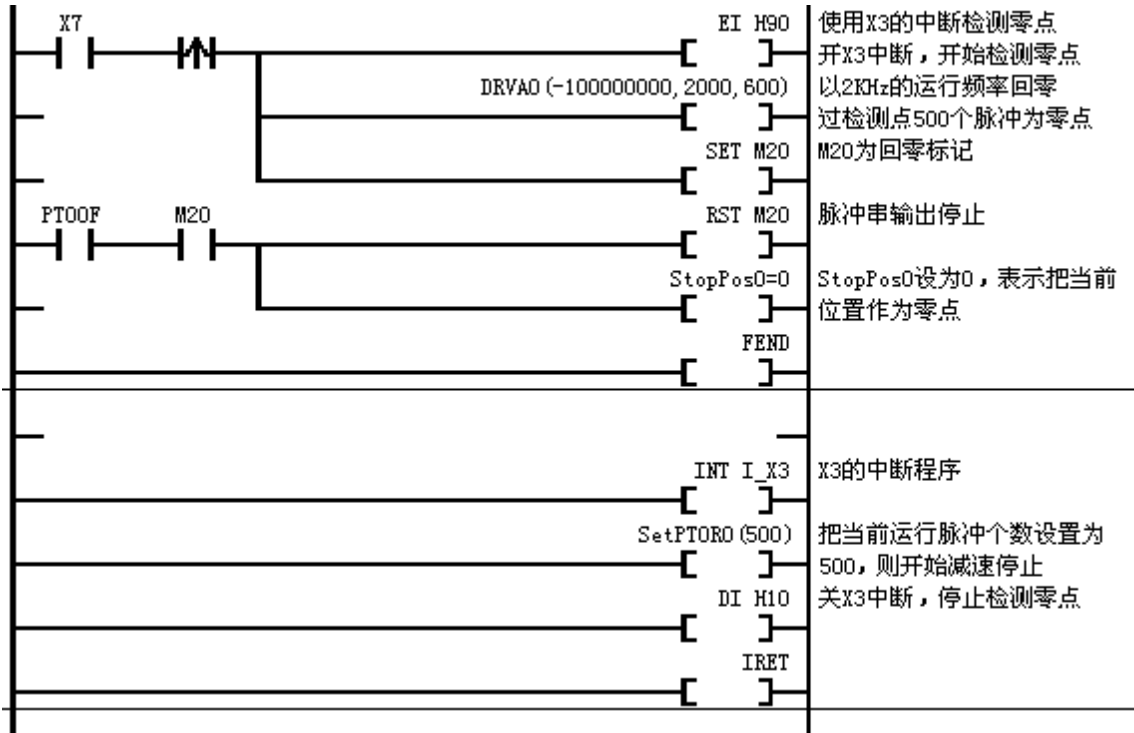


5.12 电机的回零点

方式 1:

只使用一个输入，把该输入的上升沿作为零点检测点，把过该检测点指定的距离（脉冲数，要大于减速段的脉冲个数）作为零点。

下面程序完成回零点功能，使用 X3 中断来检测，把过检测点 500 个脉冲作为零点



第六章 SPI 接口数据块操作函数库

SPI 接口除了可以使用字和字节的输入和输出指令外，还可以使用 SPI 数据块操作函数来完成数据块的输入和输出。

使用 SPI 数据块操作函数，用户需在自己的程序中连接 SPI 接口数据块操作函数库 **BSPI.yf**。

下面是该函数库中的各个函数说明。

6.1 数据块大端格式 SPI 输出函数 BBSOUT

函数定义：

FUN I, DM_Addr As D0, ByteLen As D2

函数功能：

把数据存储器块中的数据按大端格式（高字节先输出）从 SPI 输出指定的字节数。

输入参数：

DM_Addr: 用于输出数据的数据存储器首地址。

ByteLen: 要输出的数据的字节长度。

注：如果字节长度为奇数，则把最后一个数据存储器的低字节作为最后一个输出的数据。

例如：

BBSOUT(#DM300,20)

6.2 数据块小端格式 SPI 输出函数 BLSOUT

函数定义：

FUN I, DM_Addr As D0, ByteLen As D2

函数功能：

把数据存储器块中的数据按小端格式（低字节先输出）从 SPI 输出指定的字节数。

输入参数：

DM_Addr: 用于输出数据的数据存储器首地址。

ByteLen: 要输出的数据的字节长度。

注：如果字节长度为奇数，则把最后一个数据存储器的低字节作为最后一个输出的数据。

例如：

BLSOUT(#DM300,20)

6.3 数据块大端格式 SPI 输入输出函数 BBSIO

函数定义:

FUN I, DM_Addr As D0, ByteLen As D2

函数功能:

把数据存储器块中的数据按大端格式（高字节先输入输出）从 SPI 输入输出指定的字节数。

输入参数:

DM_Addr: 用于输入输出数据的数据存储器首地址。

ByteLen: 要输入输出的数据的字节长度。

注：如果字节长度为奇数，则把最后一个数据存储器的低字节作为最后一个输入输出的数据。

例如：

BBSIO(#DM300,20)

6.4 数据块小端格式 SPI 输入输出函数 BLSIO

函数定义:

FUN I, DM_Addr As D0, ByteLen As D2

函数功能:

把数据存储器块中的数据按小端格式（低字节先输入输出）从 SPI 输入输出指定的字节数。

输入参数:

DM_Addr: 用于输入输出数据的数据存储器首地址。

ByteLen: 要输入输出的数据的字节长度。

注：如果字节长度为奇数，则把最后一个数据存储器的低字节作为最后一个输入输出的数据。

例如：

BLSIO(#DM300,20)

6.5 字符串变量大端 SPI 输出(为 0 字节不输出)函数 VSTRBSO

函数定义:

FUN I, StrAddr As D0

函数功能:

把字符串变量中的字符按大端格式（高字节先输出）从 SPI 输出，为 0 的字节不输出，碰到为 0 的字结束。

输入参数:

StrAddr: 字符串变量的地址。

返回值:

从 SPI 接口输出的字节数。

例如:

VSTRBSO(#STR1)

6.6 字符串变量小端 SPI 输出(为 0 字节不输出)函数 VSTRLSO

函数定义:

FUN I, StrAddr As D0

函数功能:

把字符串变量中的字符按小端格式（低字节先输出）从 SPI 输出，为 0 的字节不输出，碰到为 0 的字结束。

输入参数:

StrAddr: 字符串变量的地址。

返回值:

从 SPI 接口输出的字节数。

例如:

VSTRLSO(#STR1)

6.7 字符串变量大端 SPI 输出(为 0 字节输出)函数 VSTRWBSO

函数定义:

FUN I, StrAddr As D0

函数功能:

把字符串变量中的字符按大端格式（高字节先输出）从 SPI 输出，为 0 的字节输出，碰到为 0 的字结束。

输入参数:

StrAddr: 字符串变量的地址。

返回值:

从 SPI 接口输出的字节数。

例如:

VSTRWBSO(#STR1)

6.8 字符串变量小端 SPI 输出(为 0 字节输出)函数 VSTRWLSO

函数定义:

FUN I, StrAddr As D0

函数功能:

把字符串变量中的字符按小端格式（低字节先输出）从 SPI 输出，为 0 的字节输出，碰到为 0 的字结束。

输入参数:

StrAddr: 字符串变量的地址。

返回值:

从 SPI 接口输出的字节数。

例如:

VSTRWLSO(#STR1)

6.9 字符串常数大端 SPI 输出(为 0 字节不输出)函数 CSTRBSO

函数定义:

FUN I, String As D0

函数功能:

把字符串常数中的字符按大端格式（高字节先输出）从 SPI 输出，为 0 的字节不输出，碰到为 0 的字结束。

输入参数:

String: 字符串常数。

返回值:

从 SPI 接口输出的字节数。

例如:

CSTRBSO("ABC 字符串 123")

6.10 字符串常数小端 SPI 输出(为 0 字节不输出)函数 CSTRLSO

函数定义:

FUN I, String As D0

函数功能:

把字符串常数中的字符按小端格式（低字节先输出）从 SPI 输出，为 0 的字节不输出，碰到为 0 的字结束。

输入参数:

String: 字符串常数。

返回值:

从 SPI 接口输出的字节数。

例如:

CSTRLSO("ABC 字符串 123")

6.11 字符串常数大端 SPI 输出(为 0 字节输出)函数 CSTRWBSO

函数定义:

FUN I, String As D0

函数功能:

把字符串常数中的字符按大端格式（高字节先输出）从 SPI 输出，为 0 的字节输出，碰到为 0 的字结束。

输入参数:

String: 字符串常数。

返回值:

从 SPI 接口输出的字节数。

例如:

CSTRWBSO("ABC 字符串 123")

6.12 字符串常数小端 SPI 输出(为 0 字节输出)函数 CSTRWLSO

函数定义:

FUN I, String As D0

函数功能:

把字符串常数中的字符按小端格式（低字节先输出）从 SPI 输出，为 0 的字节输出，碰

到为 0 的字结束。

输入参数:

String: 字符串常数。

返回值:

从 SPI 接口输出的字节数。

例如:

CSTRWLSO("ABC 字符串 123")

第七章 CAN 自由通讯接口使用说明

若要使用 CAN 自由通讯接口，则用户必须在自己的程序中连接 CAN 通讯基础操作函数库 “MCU_CAN.yf”，该函数库中提供有如下函数：

7.1 CAN 通讯口设置函数 CAN_Set

函数定义：

FUN I, PIN_S As D0, Baud As D2, ACR As LDM256, AMR As LDM258

函数功能：

初始化并设置 CAN 通讯口的管脚、波特率、滤波屏蔽寄存器及其报文验收滤波器。

输入参数：

PIN_S: CAN 管脚选择，为 0 选择 P0.0/P0.1，为 1 选择 P5.0/P5.1。

Baud: 波特率选择，0-1M，1-500K，2-250K，3-125K，4-100K，5-50K，6-25K，7-20K，8-10K。

ACR: 报文验收滤波器，单过滤器模式，只有 ID 和验收数据中滤波相关位值相同的报文才会接收。

AMR: 滤波屏蔽寄存器，为 1 的位为滤波无关位，为 0 的位为滤波相关位。

调用形式：

CAN_Set(PIN_S, Baud, ACR, AMR)

说明：

在报文发送过程中不要执行该函数。

例如：

CAN_Set(0,3,0,0xFFFFFFFF) 波特率为 125K，接收所有标准标识符和扩展标识符报文。

该函数的使用例子如下：



CAN_Set 函数使用例子

7.2 CAN 通讯口发送数据函数 CAN_TX

函数定义：

FUN I, TXBn As D1, ID As LDM256, DLC As D2, DM_Addr As D0

函数功能：

向指定的 CAN 发送缓冲器中的写发送数据并从端口发送出去。

输入参数:

SPI_Addr: 模块地址（本机中为 14）。

TXBn: 指定的发送缓冲器编号（0~15）。

ID: 标识符。左对齐格式：位 31-位 21 为标准标识符或者位 31-位 3 为扩展标识符。

DLC: 发送数据长度，位 7: 0 为标准帧、1 为扩展帧，位 6: 0 为数据帧、1 为远程帧，位 3-位 0: 数据长度。

DM_Addr: 要写入到发送缓冲器中的数据在 DM 存储器中的首地址（占 4 个字），低字节先发，高字节后发。注：首地址不能为 256 或 257。

调用形式:

CAN_TX(TXBn, ID, DLC, DM_Addr)

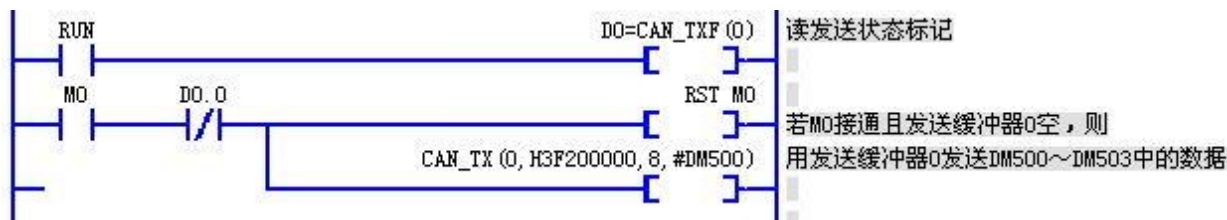
说明:

当执行该函数后将使该函数指定的发送缓冲器进入发送状态，该发送缓冲器的发送状态标记为 ON，当该发送缓冲器中的数据发送完成后其发送状态标记变为 OFF。

例如:

CAN_TX(0,H35800000,8,#DM300)

该函数的使用例子如下:



CAN_TX 函数的使用例子

7.3 读 CAN 通讯口发送缓冲器状态标记函数 CAN_TXF

函数定义:

FUN I, Void As D0

函数功能:

读出 CAN 的各个发送缓冲器的发送状态标记。

输入参数:

Void: 无意义，可为任何值。

返回值:

其中的各个位为各个发送缓冲器的发送状态标记（0 为无等待发送报文，1 为有等待发送报文）。位 0：发送缓冲器 0 的发送状态标记，位 1：发送缓冲器 1 的发送状态标记，…。

调用形式:

CAN_TXF(Void)

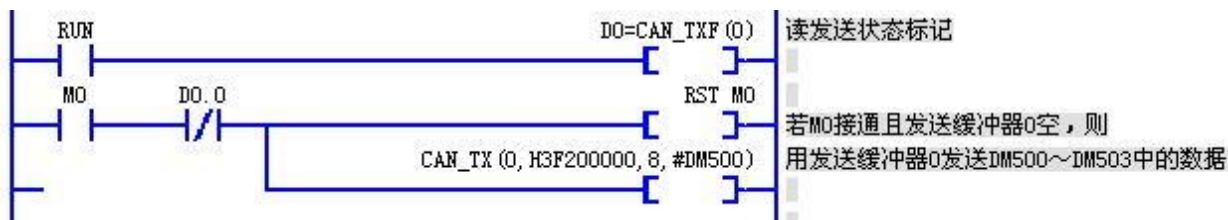
说明:

当执行发送函数 CAN_TX 后将使该函数指定的发送缓冲器的发送状态标记为 ON，当该发送缓冲器中的数据发送完成后其发送状态标记变为 OFF。

例如:

D0=CAN_TXF(0)

该函数的使用例子如下:



CAN_TXF 函数的使用例子

7.4 CAN 通讯口禁止发送函数 CAN_DisTX

函数定义:

FUN I, Void As D0

函数功能:

禁止 CAN 通讯口发送数据。

输入参数:

Void: 无意义，可为任何值。

调用形式:

CAN_DisTX(Void)

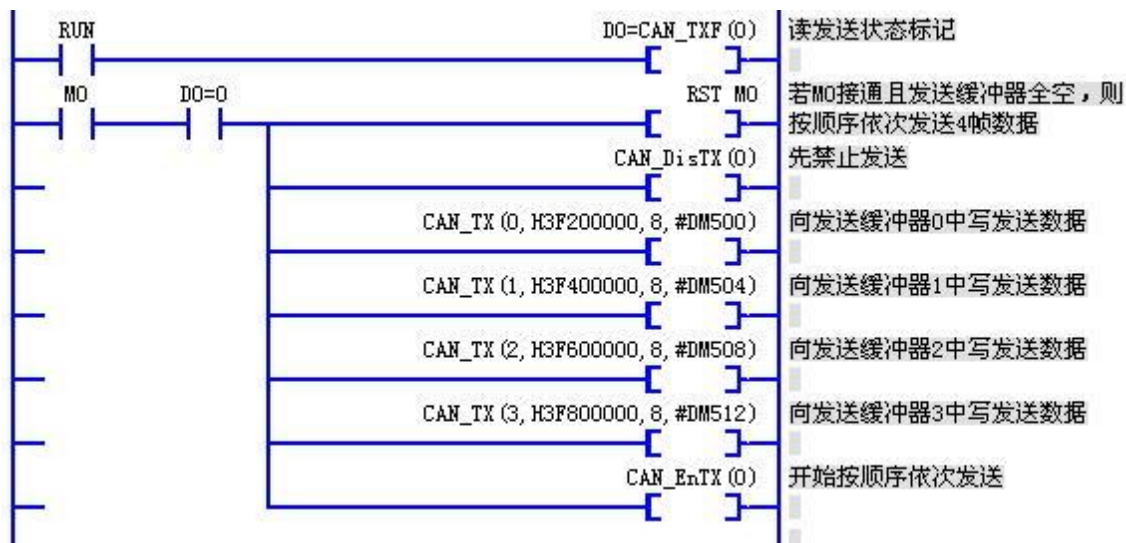
说明:

若希望从发送缓冲器 0 到发送缓冲器 15 按顺序依次发送，则可先使用该函数禁止发送，然后再向各个发送缓冲器中依次写入发送数据，然后再使用允许发送函数允许发送。

例如:

CAN_DisTX(0)

该函数的使用例子如下：



7.5 CAN 通讯口允许发送函数 CAN_EnTX

函数定义：

FUN I, Void As D0

函数功能：

允许 CAN 通讯口发送数据，若之前是禁止发送，则将从发送缓冲器 0 开始依次扫描发送。上电后是允许发送。

输入参数：

Void: 无意义，可为任何值。

调用形式：

CAN_EnTX(Void)

说明：

上电后是允许发送，因此若没执行禁止发送函数，则没必要使用该函数允许发送。

例如：

CAN_EnTX(0)

7.6 CAN 通讯口重新发送数据函数 CAN_ReTX

函数定义：

FUN I, TXBn As D2

函数功能:

重新发送指定的 CAN 发送缓冲器中的数据。

输入参数:

TXBn: 指定的发送缓冲器编号 (0~15)。

调用形式:

CAN_EnTX(TXBn)

例如:

CAN_ReTX (0)

7.7 读 CAN 通讯口最先接收数据的缓冲器编号函数 CAN_RXB

函数定义:

FUN I, Void As D0

函数功能:

读出 CAN 最先接收数据的缓冲器编号。

输入参数:

Void: 无意义, 可为任何值。

返回值:

最先接收数据的缓冲器编号, 位 7 为 1 表示接收缓冲器全空、没有接收数据。

说明:

每次读取接收缓冲器数据, 都要先使用该函数获得最先接收数据的缓冲器编号, 若位 7 为 0, 然后再使用 CAN_RX 函数读取该编号接收缓冲器中的数据。

调用形式:

CAN_RXB(Void)

例如:

D0=CAN_RXB(0)

7.8 CAN 通讯口接收数据函数 CAN_RX

函数定义:

FUN I, RXBn As D0, DM_Addr As D1

函数功能:

把指定的 CAN 接收缓冲器中的数据读入到 DM 存储器中。

输入参数:

RXBn: 接收缓冲器编号 (0~15), 该编号必须使用 CAN_RXB 函数获得。

DM_Addr: 要读入到的 DM 存储器首地址, 占 7 个字, 其中[0][1]:接收到的标识符 ID (左对齐格式: 位 31-位 21 为标准标识符或者位 31-位 3 为扩展标识符), [2]:数据长度 DLC (位 7: 0 为标准帧、1 为扩展帧, 位 6: 0 为数据帧、1 为远程帧, 位 3-位 0: 数据长度), [3]~[6]:接收数据 (先接收的为低字节, 后接收的为高字节)。

调用形式:

CAN_RX(RXBn, DM_Addr)

例如:

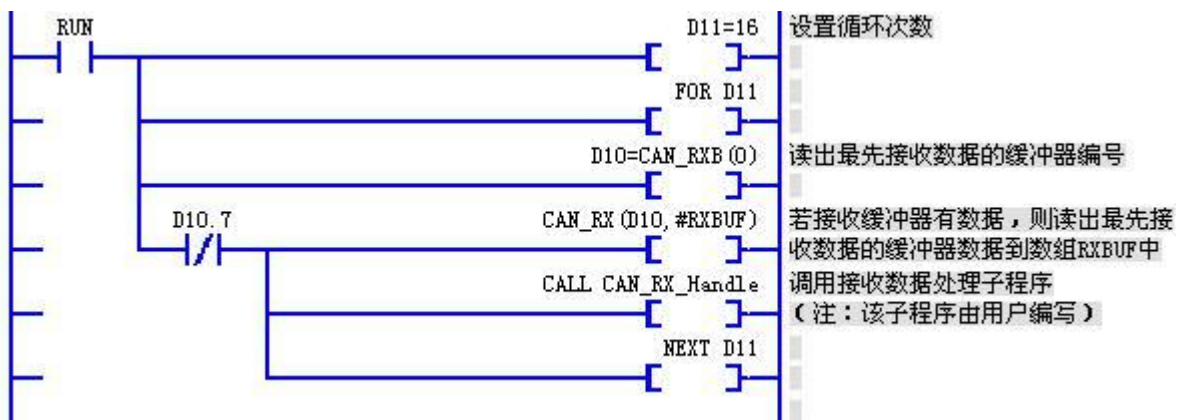
CAN_RX(0,#DM400)

说明:

先接收到的报文最先被读出。每次读取接收缓冲器数据, 都要先使用 CAN_RXB 函数获得最先接收数据的缓冲器编号, 若位 7 为 0, 然后再使用 CAN_RX 函数读取该编号接收缓冲器中的数据。

如果接收了新的报文, 但 16 个接收缓冲器全满, 此时可能会丢弃掉新接收的报文。为了防止丢失报文, 尽量在一个扫描周期内对接收缓冲器进行判断读取数据 16 次, 必要时可在一个扫描周期内多处放置报文接收程序。

报文接收程序例子如下:



报文接收程序

7.9 读 CAN 通讯口总线错误状态函数 CAN_ESR

函数定义:

FUN I, Void As D0

函数功能:

读出 CAN 通讯口的总线错误状态，若总线错误关闭则自动恢复。

输入参数:

Void: 无意义，可为任何值。

返回值:

位 0 为 1 表示 CAN 模块在总线错误关闭状态，位 2 为 1 表示 CAN 模块接收缓冲器溢出。

调用形式:

CAN_ESR(Void)

说明:

为了能在总线错误关闭后自动恢复，每隔一段时间要执行一次该函数。

例如:

CAN_ESR(0)

该函数的使用例子如下:

