

**SEARI**

上電科

可編程控制器 VPC1

# VPC1 编程手册

## 目录

软元件的资源和功能	8
数据类型和存储区说明	14
编程指令	19
使用中断功能	163
使用 PLC 串行通信口	165
Modbus-RTU 协议简介	191
系统寄存器一览表 (M 区)	205
特殊数据寄存器一览表 (S 区)	218
错误代码	221

# 前 言

上海电器科学研究所（集团）有限公司是致力于电器设备研发和生产的高科技企业。上海电科凭借在工业控制领域十余年的开发经验和对 PLC 产品的深刻理解，利用我们所掌握的 PLC 软、硬件核心技术，开发了与国际先进水平同步的高品质 PLC。上海电科以 100% 自有知识产权、优良的品质、体贴到位的服务，研发和生产适合国内用户需求的小型一体化 PLC 产品。

VPC1 系列 PLC 产品在交通、电力控制设备、纺织机械、塑料机械、数控机床、小型包装机械以及单一过程控制装置等方面有着广泛的应用。本产品性能稳定、性价比高，得到用户的一致认可。VPC1 系列 PLC 在产品推广应用的过程中，得到了广大客户的支持，在此表示衷心感谢！

VX-Pro 是对 VPC1 系列 PLC 产品进行编程的软件，为了方便用户使用 VPC1 系列 PLC，我们编写了此手册，对 VX-Pro 软件进行了系统介绍。

我们力求手册内容详尽，但由于时间仓促，书中难免有疏漏不足之处，敬请广大用户指正，我们将不胜感激。

若您需要更进一步了解产品和获得更多的技术支持，请通过以下方式与我们联系：

地 址： 上海市武宁路 505 号 8 号楼 2 楼

邮 编： 200063

电 话： 021-62574990-476（市场）

021-62574990-474（技术）

传 真： 021-62166010

E-mail: [tech\\_support@seari.com.cn](mailto:tech_support@seari.com.cn)

网 址: <http://fileldbus.seari.com.cn>

# 目 录

第 1 章 软元件的资源和功能.....	8
1.1 VPC1 系列 PLC 内存区域划分.....	8
1.2 软元件编号一览.....	8
1.3 输入输出继电器.....	9
1.3.1 输入输出继电器的编号.....	9
1.3.2 带扩展模块时的地址编号规则.....	10
1.3 中间继电器.....	10
1.4 数据寄存器.....	11
1.5 特殊继电器 (S 区).....	11
1.6 特殊寄存器(M 区).....	12
1.7 定时器.....	12
1.8 计数器.....	13
第 2 章 数据类型和存储区说明.....	14
2.1 支持的数据类型.....	14
2.1.1 数据的种类.....	14
2.1.2 变量取值范围.....	14
2.2 直接地址与内存单元之间的映射.....	15
2.2.1 位地址:.....	16
2.2.2 字节地址:.....	16
2.2.3 字地址:.....	17
2.2.4 双字地址:.....	17
2.3 数字直接量.....	17
第 3 章 编程指令.....	19
3.1 综述.....	19
3.2 位指令.....	19
3.2.1 常开触点指令.....	19
3.2.2 常闭触点指令.....	20
3.2.3 普通线圈、复位线圈、置位线圈.....	21
3.2.4 边沿(上升沿、下降沿)微分.....	22
3.2.5 保持(KEEP).....	24
3.2.6 块与、块或.....	24
3.3 移动指令.....	25
3.3.1 MOV (字移动).....	25
3.3.2 MOVB (字节移动).....	26
3.3.3 MOVD (双字移动).....	27
3.3.4 MOVR (实数移动).....	27
3.3.5 BMW (字型块移动).....	28
3.3.6 BMB (字节型块移动).....	29
3.3.7 BMD (双字型块移动).....	29
3.3.8 SWAP (交换字节).....	30
3.4. 比较指令.....	31
3.4.1 GT (大于).....	31

3.4.2 GTB (字节型大于)	32
3.4.3 GTD (双字型大于)	33
3.4.4 GTR (实型大于)	33
3.4.5 GE (大于等于)	34
3.4.6 GEB (字节型大于等于)	35
3.4.7 GED (双字型大于等于)	35
3.4.8 GER (实型大于等于)	36
3.4.9 EQ (等于)	37
3.4.10 EQB (字节型等于)	37
3.4.11 EQD (双字型等于)	38
3.4.12 EQR (实型大于)	39
3.4.13 NE (不等于)	40
3.4.14 NEB (字节型不等于)	40
3.4.15 NED (双字型不等于)	41
3.4.16 NER (实型不等于)	42
3.4.17 LT (小于)	42
3.4.18 LTB (字节型小于)	43
3.4.19 LTD (双字型小于)	44
3.4.20 LTR (实型小于)	45
3.4.21 LE (小于等于)	46
3.4.22 LEB (字节型大于等于)	46
3.4.23 LED (双字型小于等于)	47
3.4.24 LER (实型小于等于)	48
3.5 逻辑运算	48
NOT (二进制数取反)	48
ANDW (字与)	49
ANDB (字节与)	50
ANDD(双字与)	51
ORW (字或)	51
ORB (字节或)	52
ORD (双字或)	53
XOR (异或)	54
XORW (字异或)	54
XORB (字节异或)	55
XORD (双字异或)	56
INW (反转字)	57
INVB (反转字节)	57
INVD (反转双字)	58
3.6 移位指令	58
SHL (左移)	58
SHLB (字节左移)	59
SHLD (双字左移)	60
ROL (循环左移)	61
ROLB (字节循环左移)	61

ROL (双字循环左移) .....	62
SHR (右移) .....	63
SHRB (字节右移) .....	64
SHRD (双字右移) .....	65
ROR (循环右移) .....	66
RORB (字节循环右移) .....	66
RORD (循环右移) .....	67
3.7 类型转换.....	67
B2I (BCD 转整型) .....	67
I2B (整型转 BCD) .....	68
ASC (ASCII 代码转换) .....	69
BTI (BYTE 转 INT) .....	71
ITA (整型转换为 ASCII) .....	71
ITB (INT 转 BYTE) .....	72
ITD (INT 转 DINT) .....	73
DTI (DINT 转 INT) .....	74
DTR (DINT 转 REAL) .....	74
ROUND (实型转为双整型) .....	75
TRUNC (实型转为双整型) .....	76
SEG (7 段数码管显示) .....	76
ATH (ASCII 字符串转换为十六进制数) .....	77
HTA (十六进制数转换为 ASCII 字符串) .....	78
DTA (双字型整数转换为 ASCII 字符串) .....	79
DECO (解码) .....	80
ENCO (编码) .....	81
3.8 数学运算.....	82
ADD_I (加法) .....	82
ADDB (字节型加法) .....	83
ADDD (双字型加法) .....	84
ADDR (实型加法) .....	84
SUB_I (减法) .....	85
SUBB (字节型减法) .....	86
SUBD (双字型减法) .....	87
SUBR (实型减法) .....	88
MUL_I (乘法) .....	89
MULB (字节型乘法) .....	89
MULD (双字型乘法) .....	90
MULR (实型乘法) .....	91
DIV_I (除法) .....	92
DIVB (字节型除法) .....	93
DIVD (双字型除法) .....	93
DIVR (浮点型除法) .....	94
INC_I (自增) .....	95
INCB (字节型自增) .....	95

INCD (双字型自增)	96
DEC_I (自减)	97
DECB (字节型自减)	97
DECD (双字型自减)	98
SIN (正弦函数)	99
COS (余弦函数)	100
TAN (正切函数)	100
LN (工业对数函数)	101
EXP (幂指数函数)	102
SQRT (开平方函数)	102
3.9 程序控制	103
JMP (条件跳转)	104
NOP (空操作)	105
END (程序结束)	106
STOP (终止运行)	106
CRET (子程序有条件返回)	107
RET (子程序无条件返回)	108
FOR/NEXT (FOR/NEXT 循环)	109
SCR (工序控制)	111
3.10 中断指令	113
ATCH (中断关联)	113
DTCH (中断关联解除)	113
ENI (全局中断使能)	114
DISI (全局中断禁能)	114
STIM (时间间隔定时器)	114
3.11 通讯指令	116
TXD (串口输出)、RXD (串口输入)	116
NETR (网络读)、NETW (网络写)	119
MBAR,MBARX (添加非周期 Modbus 请求)	121
MBCR,MBCRX (添加周期 Modbus 请求)	123
MBDR (删除周期 Modbus 请求)	125
3.12 表格指令	125
TBL (向表格中添加数据)	125
FIFO (先入先出)	126
LIFO (后入先出)	127
FILL (内存填充)	128
FND (表格查找)	129
SCL (比例缩放)	130
SCL2 (比例缩放 2)	133
SCL3 (比例缩放 3)	136
3.13 定时器	139
TON (打开延迟定时器)	139
TOF (关闭延迟定时器)	140
TP (脉冲延迟定时器)	14138

3.14 计数器.....	143
CTU (增计数器)、CTD (减计数器) .....	144
CTUD (增/减计数器) .....	145
3.15 字符串指令.....	147
SLEN (字符串长度) .....	147
SCPY (字符串复制) .....	148
SCAT (字符串连接) .....	149
SSCPY (从字符串复制子字符串) .....	149
SFND (字符串查找) .....	150
CFND (字符查找) .....	151
3.16 PID 指令 .....	152
PIDIN (PIN 输入转换) .....	152
PID (比例积分微分) .....	153
PIDOUT (PID 输出转换) .....	153
3.17 实时时钟指令.....	156
TODR (读实时时钟) .....	157
TODW (写实时时钟) .....	157
3.18 脉冲输出指令.....	158
PWM (脉宽调制输出) .....	158
PTO (脉冲连输出) .....	159
PWMSTOP(PWM 输出停止).....	161
PLSSTOP(PTO 输出停止).....	162
第 4 章 使用中断功能.....	163
4.1 中断程序中中断号的分配 .....	163
4.2 中断程序中中断号的收回 .....	163
4.3 中断的使能.....	163
4.4 中断的取消使能.....	163
4.5 时钟中断的使用.....	164
4.6 中断事件与中断号分配表.....	164
第 5 章 使用 PLC 串行通信口 .....	165
5.1. 上位机链接通信.....	165
5.2. PC-Net 自组网通信.....	166
5.2.1 PC-Net 自组网通信概述.....	166
5.2.2 用户配置说明.....	167
5.2.3 出错寄存器说明.....	167
5.2.4 PC-Link 说明.....	167
5.2.5 网络指令说明.....	168
5.2.6 实例说明.....	169
5.2.7 PC-Link 应用实例 .....	172
5.3 自由口通信.....	177
5.3.1 用户配置说明.....	177
5.3.2 出错寄存器说明.....	177
5.3.3 发送指令 (TXD) .....	177
5.3.4 接收指令 (RXD) .....	178

5.4	Modbus-RTU 从站通信 .....	185
5.4.1	用户配置说明 .....	185
5.4.2	Modbus 编址 .....	185
5.5	Modbus-RTU 主站通信 .....	186
5.5.1	添加非周期报文指令 (MBAR) .....	186
5.5.2	添加周期报文指令 (MBCR) .....	190
5.5.3	删除周期报文指令 (MBDR) .....	191
5.5.4	出错寄存器说明 .....	191
第 6 章	Modbus-RTU 协议简介 .....	191
6.1	Modbus-RTU 帧结构 .....	191
6.2	Modbus-RTU 命令介绍 .....	192
	功能码 01: 读线圈 (开关量输出) .....	192
	功能码 02: 读输入状态 (开关量输入) .....	193
	功能码 03: 读保持寄存器 (模拟量输出) .....	194
	功能码 04: 读输入寄存器 (模拟量输入) .....	195
	功能码 05: 写单个线圈 (开关量输出) .....	196
	功能码 06: 写单个保持寄存器 (模拟量输出) .....	197
	功能码 15: 写多个线圈 (开关量输出) .....	198
	功能码 16: 写多个保持寄存器 (模拟量输出) .....	200
6.3	异常回应 .....	201
6.4	CRC 校验算法 .....	202
	6.4.1 直接计算 CRC .....	202
	6.4.2 查表快速计算 CRC .....	203
第 7 章	系统寄存器一览表 (M 区) .....	205
第 8 章	特殊数据寄存器一览表 (S 区) .....	218
第 9 章	错误代码 .....	221
	9.1 严重错误代码 .....	221
	9.2 非严重错误代码 .....	222



# 第 1 章 软元件的资源和功能

在本章中，对 VPC1 系列 PLC 中使用的内存区域划分、输入输出继电器、中间继电器、特殊寄存器、计数器、定时器等各种软元件的资源进行了说明。

## 1.1 VPC1 系列 PLC 内存区域划分

VPC1 系列 PLC 的内存区域划分如下表所示：

表 1.1

No.	存储区	符号	访问方式	大小	地址
1	物理输入	I	位/字节/ 字/双字	100 字	%I0.0~%I99.15
2	物理输出	Q		100 字	%Q0.0~%I99.15
3	中间继电器	W		200 字	%W0.0~%W199.15
4	数据寄存器	D	字/双字	2048 字	%D0~%D2047
5	数据寄存器间接寻址	P	字	2048 字	%PW0~%PW2047
6	特殊继电器	S	位	128 位	%S0~%S127
7	特殊寄存器	M	字	256 字	%MW0~%MW255
8	定时器	T	字	64 字	%TX0~%TX63
9	计数器	C	字	64 字	%CX0~%CX63

注：P 是指向数据寄存器(D 区)的指针，使用方法举例如下：

如 %DW0=1，此时 %PW0 指向 %DW(0+1)，即 %PW0=%DW1

如 %DW10=9，此时 %PW10 指向 %DW(10+9)，即 %PW10=%DW19

## 1.2 软元件编号一览

PLC 主机的软元件资源分配如下表 1.1 所示。

另外，但 PLC 连接扩展模块时，扩展模块的输入输出软元件标号请参考该扩展模块的硬件手册。

表 1.1

软元件名称	内容			参照
输入输出继电器				
输入继电器	%I0.0~物理输入点数	对应主机和扩展模块的物理输入、输出点		1.3 节
输出继电器	%Q0.0~物理输入点数			1.1 节
中间继电器				
一般用	%W0.0~%W199.15	200 字	也可作字节、字或双字用，如: %WB0=%W0.0~%W0.7	1.3 节
数据寄存器				
一般用(16 位)	%DW0~%DW1023	1024 字	作为双字使用时，总计 512 个,即: %DD0~%DD511	
软元件名称	内容			参照
特殊继电器 (S 区)				
一般用	%S0~%S127	128 位	对应系统特殊功能位信息	
特殊寄存器 (M 区)				
一般用	%MW0~%MW255	256 字	对应系统特殊功能位信息	
定时器				
1ms 定时器	%TX0	1 字	0.001~32.767 秒	1.7 节
10ms 定时器	%TX1~%TX4	4 字	0.01~327.67 秒	1.7 节
100ms 定时器	%TX5~%TX63	59 字	0.1~3276.7 秒	1.7 节
计数器				
一般用(32 位)	%CX0~%CX63	64 字		

注:

使用掉电保持区域时，需通过编程软件设置，详细见编程软件手册 6.1 章节。

使用掉电保持区域时，请选装锂电池。PLC 本身自带的<sup>1</sup>数据维持时间大约为 15 天。

扩展模块所占用的输入输出资源请参照该模块的硬件手册。

## 1.3 输入输出继电器

### 1.3.1 输入输出继电器的编号

具体如下:

输入输出继电器的编号是 PLC 基本单元所有的，每个地址对应 PLC 上的具体接线端子。

VPC1 系列 可编程控制器	型号	VPC1-32M	VPC1-60M
	输入	%I0.0~%I0.11 %I1.0~%I1.7	%I0.0~%I0.11 %I1.0~%I1.11 %I2.0~%I2.11
	输出	%Q0.0~%Q0.7 %Q1.0~%Q1.5	%Q0.0~%Q0.7 %Q1.0~%Q1.7 %Q2.0~%Q2.7

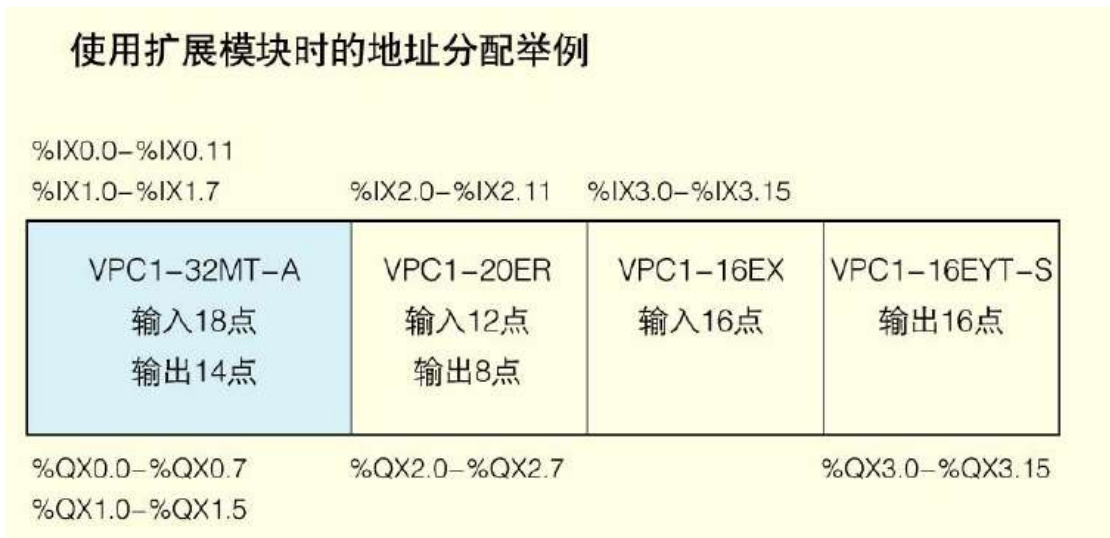
以上输入输出地址也可以使用字节(**IB**%,8 位), 字(**IW**,16 位), 双字(**ID**,32 位)进行访问, 如: **%IB0=%I0.7~%I0.1** **%IB1=%I0.15~%I0.8** (注: 地址大的是高位)

**%IW0=%IB1~%IB0**

**%ID0=%IW1~%IW0=%IB3~%IB0=%I0.31~%I0.0**

### 1.3.2 带扩展模块时的地址编号规则

带扩展模块时的起始地址取决于该扩展模块所在的位置。下例给出了 VPC1-32M 带 3 个扩展模块是的地址编号:



当扩展中有模拟量模块、通讯模块时, 后面的模块地址依次排列。如上图中 VPC1-20ER 替换为 VPC1-4AD 时, 通过查 VPC1-4AD 的硬件手册得知其占用 4 个字输入、2 个字输出的资源, 因此有:

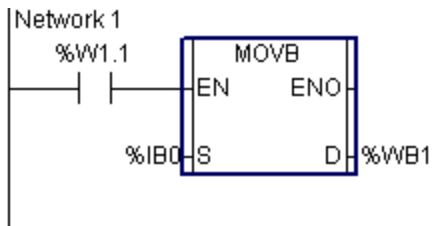
	主机(VPC1-32M)	扩展 1 (VPC1-4AD)	扩展 2 (VPC1-20ER)	说明
输入	%I0.0~%I0.11 (%IW0) %I1.0~%I1.7 (%IW1)	%IW2~%IW5	%I6.0~%I6.11 (%IW6)	1. 位地址的“X”可以省略, %Ix0.0=%I0.0 2. %IW 为字输入 %QW 为字输出, 如: %IW0=%I0.15~%I0.0
输出	%Q0.0~%Q0.7 (%QW0) %Q1.0~%Q1.5 (%QW1)	%QW2~%QW3	%Q4.0~%Q4.7 (%QW5)	

## 1.3 中间继电器

VPC1 PLC 的中间继电器的地址范围是: **%W0.0~%W199.15**

以上地址也可以使用字节(**WB**%,8 位), 字(**WW**,16 位), 双字(**WD**,32 位)进行访问, 其规则请参见 1.2.1 章节。

使用举例:

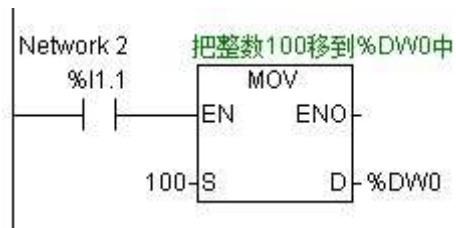


## 1.4 数据寄存器

VPC1 PLC 的数据寄存器的地址范围是：%DW0.0~%DW2047

数据寄存器也可使用双字指令访问，此时地址范围是：%DW0.0~%DW1023

使用举例 1：



使用举例 2：



## 1.5 特殊继电器（S 区）

特殊继电器用以存放 PLC 一些特殊功能的位(bool)信息。地址范围: %S0~%S80。

具体每个地址的功能定义请参考第 8 章 特殊数据寄存器一览表（S 区）章节

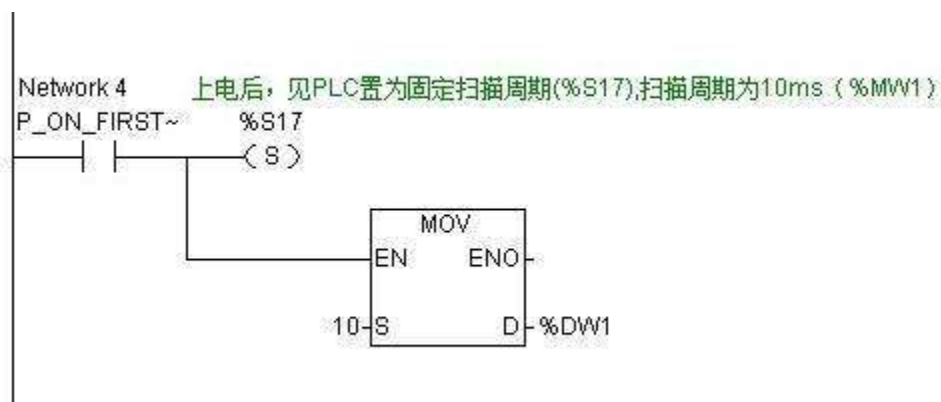
为了方便使用，在编程软件中对一些常用的特殊继电器做了变量定义，具体如表 1.5 所示。（例如：P\_ON 和 %S01 是等效的）

表 1.5

变量名称	特殊继电器地址	说明
P_ON	%S 01	常闭
P_OFF	%S 02	常开
P_ALTERNATE	%S 03	每扫描周期变换一次状态
P_ON_FIRST_CYC	%S 04	PLC 首次扫描时为 ON，之后为 OFF
P_OFF_FIRST_CYC	%S 05	PLC 首次扫描时位 OFF，之后为 ON
P_1S	%S 06	1 秒变换一次状态
P_0.5S	%S 07	0.5 秒变换一次状态
P_30S	%S 08	30 秒变换一次状态
P_0.1S	%S 09	0.1 秒变换一次状态
P_10S	%S 11	10 秒变换一次状态
P_0.02S	%S 12	0.02 秒变换一次状态

## 1.6 特殊寄存器(M 区)

特殊继电器用以存放 PLC 一些特殊功能的字信息。地址范围: %MW00~%MW8000  
 具体每个地址的功能定义请参考第 7 章 系统寄存器一览表 (M 区) 章节  
 使用举例:



## 1.7 定时器

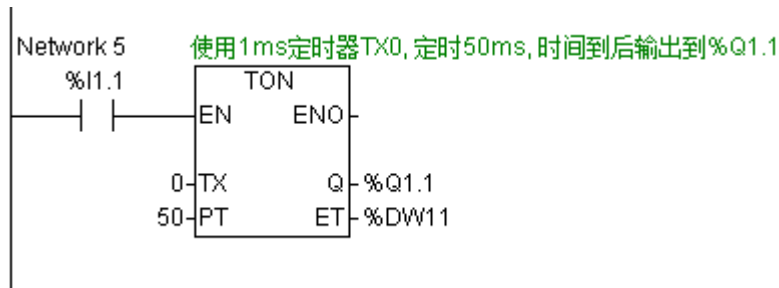
PLC 的定时器资源如下表所示:

定时器地址	精度	时间范围(秒)	说明
%TX0	1ms	0.001~32.767	
%TX1~%TX4	10ms	0.01~327.67	
%TX5~%TX63	100ms	0.1~3276.7	

VPC1 系列 PLC 的定时器共有 TON(定时 ON)、TOF(定时 OFF)和 TP(定时脉冲输出)三种, 如 TP TX0, TON TX1, TOF TX2 分别表示使用了 3 个不同种类的定时器。

注：每个地址只能使用一种定时器，如 TP TX0 和 TON TX0 不可以同时使用。

使用举例：



上例中，TX 为定时器地址，PT 为设定的定时值，Q 为输出，ET 为定时器当前值。

## 1.8 计数器

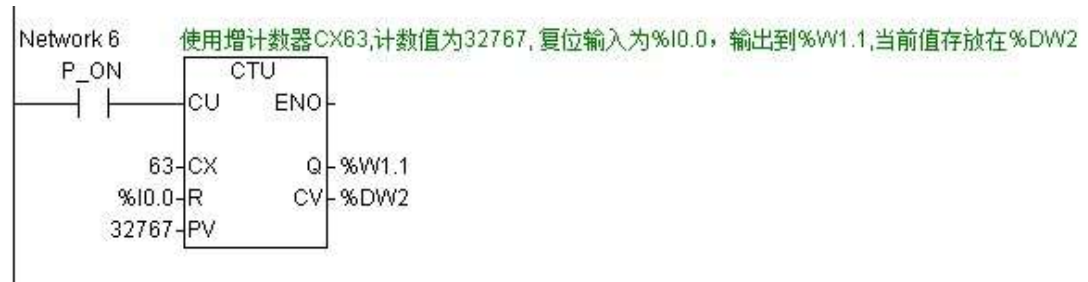
PLC 的计数器资源如下表所示：

定时器地址	计数频率	计数范围	说明
CX0~CX63	$\leq 50\text{Hz}$ (周期 $\geq 20\text{ms}$ )	0~32767	

VPC1 系列 PLC 的计数器共有 CTD(定时 ON)、CTU(定时 OFF)和 CTUD(定时脉冲输出增减计数)三种，如 TP TX0，TON TX1，TOF TX2 分别表示使用了 3 个不同种类的定时器。

注：每个地址只能使用一种计数器器，如 CTD CX0 和 CTUD CX0 不能同时使用。

使用举例：



# 第 2 章 数据类型和存储区说明

## 2.1 支持的数据类型

### 2.1.1 数据的种类

VPC1 系列 PLC 中, 根据用途和目的的不同, 数据可分为布尔型 (位)、8 位数据 (字节), 16 位数据 (字)、32 位数据 (双字) 等不同类型。具体如下表所示:

表 2.1.1

数据类型	数据长度 (位)	符号	说明	举例
BOOL	1	X 或空	布尔型	%I1.1 或 %Ix1.1
BYTE	8	B	字节型 (无符号)	%WB1
SINT	8	B	短整型 (有符号)	
WORD	16	W	字型 (无符号)	
INT	16	W	整型 (有符号)	
DWORD	32	D	双字型 (无符号)	
DINT	32	D	双字整型 (有符号)	
REAL	32	D	实型	
STRING	0~255 个字节	B	字符串	

**注:** STRING 类型只支持在字符串类型的指令参数中输入, 不支持自定义字符串型变量。

使用直接物理地址时, 分别用 X,B,W,D 代表不同的数据宽度。例如:

%Q0.0 (或 %Q0.0), 表示 Q 区第 0 个字的第 0 位。

%QB1, 表示 Q 区的第 1 个字节, 对应 “%Q1.0~%Q1.07”。

%QW2, 表示 Q 区的第 2 个字, 对应 “%QB2~%QB3”。

%QD3, 表示 Q 区的第 3 个双字, “%QW4~%QW5”。

### 2.1.2 变量取值范围

对于不同数据类型的变量, 其取值范围如下表所示:

表 2.1.2

数据类型	说明	长度 (位)	取值范围	缺省值
BOOL	布尔型	1	True(1), False(0)	False(0)
BYTE	字节型 (无符号)	8	0 ~ 255 ( $2^8-1$ )	0
SINT	短整型 (有符号)	8	$-2^7 \sim (2^7-1)$	0
WORD	字型 (无符号)	16	0 ~ ( $2^{16}-1$ )	0
INT	整型 (有符号)	16	$-2^{15} \sim (2^{15}-1)$	0
DWORD	双字型 (无符号)	32	0 ~ ( $2^{32}-1$ )	0

数据类型	说明	长度 (位)	取值范围	缺省值
DINT	双字整型 (有符号)	32	$-2^{31} \sim (2^{31}-1)$	0
REAL	实型	32	采用 ANSI/IEEE754-1985 标准; 约: $1.18*10^{-38} \sim 3.40*10^{38}$ , $-3.40*10^{38} \sim -1.18*10^{-38}$	0.0
STRING	字符串	0~255 字节	数字字母下划线	-

### 实数的格式:

实数 (浮点数) 表示为 32 位单精度型数字, 格式按照 ANSI/IEEE 754-1985 标准描述, 32 位浮点数由符号位, 指数部分, 位数部分组成, 如下图所示:



浮点数精确到小数点后 6 位。

### 实数的精度:

当一个非常大的数和一个非常小的数 (数字相差为 10 的 6 次方) 进行运算时可能导致不精确的运算结果

如  $1000000 + 1 = 1000000$

字符串的格式:

字符串为字符的序列, 其中每个字节存储一个字符, 字符串的第一个字节存储整个字符串的长度, 第二个字节开始储存字符串, 整个字符串的最大长度为 255, 包含 254 个字符和一个长度字节。

表 3-04 字符串的格式

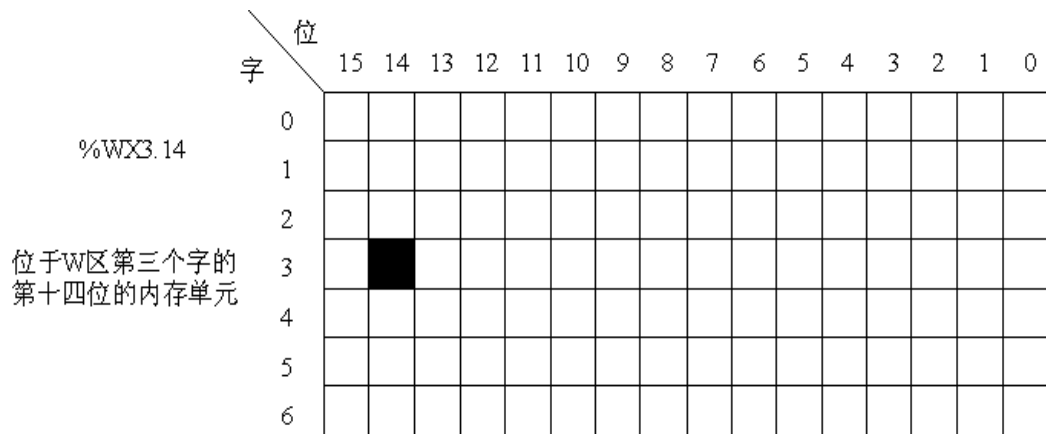
长度	字符 1	字符 2	字符...	字符 254
字节 0	字节 1	字节 2	字节...	字节 254

## 2.2 直接地址与内存单元之间的映射

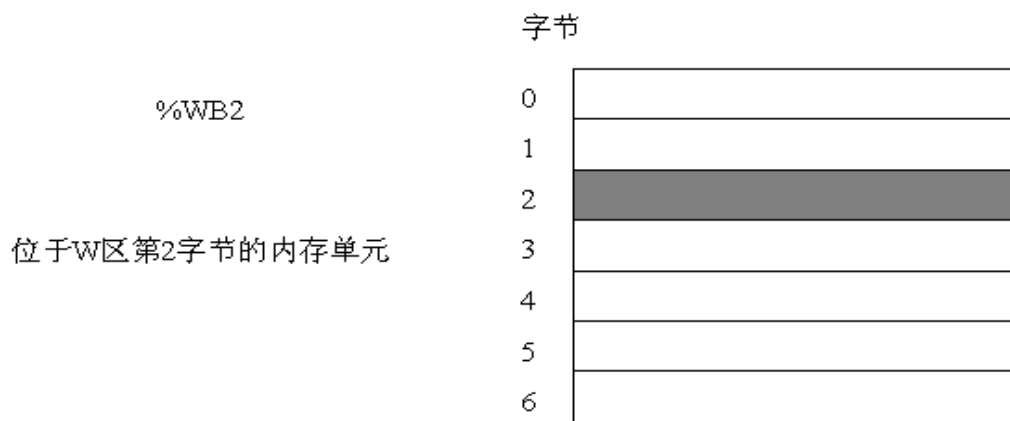
每一个合法的直接地址都对应于 CPU 中的一个内存单元。在程序中对直接地址的操作就是对其对应的内存单元进行操作。下面将以 W 区为例图解直接地址与内存单元之间的映射关系。



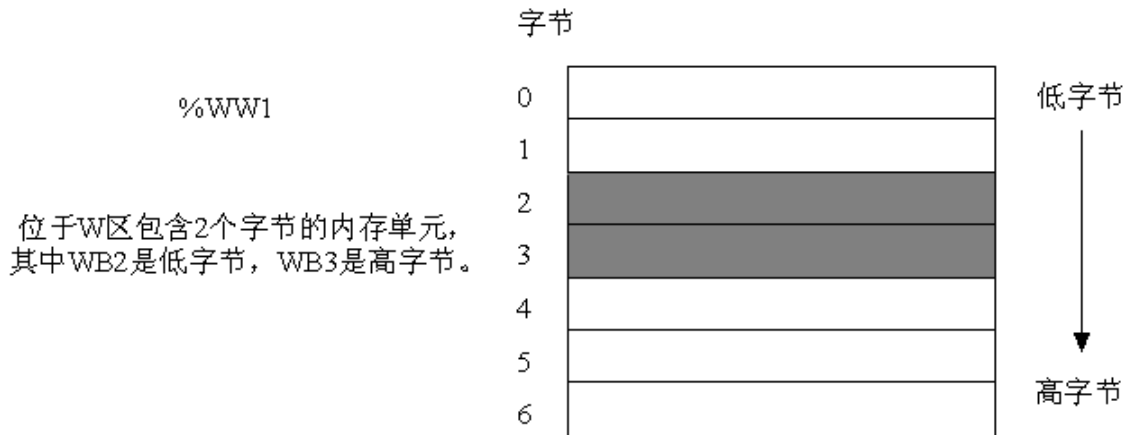
### 2.2.1 位地址:



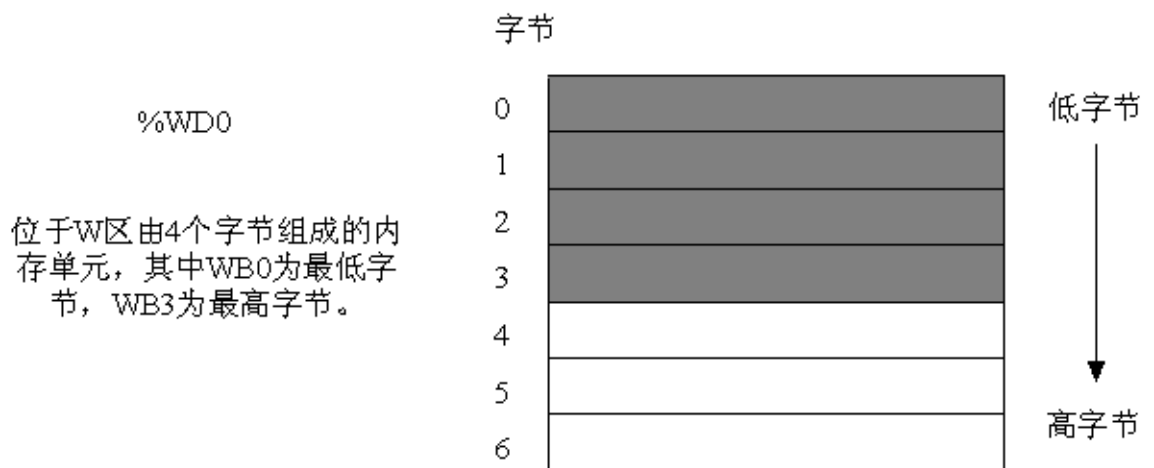
### 2.2.2 字节地址:



### 2.2.3 字地址:



### 2.2.4 双字地址:



## 2.3 数字直接量

有两类数字直接量：整数直接量和实数直接量。插在数字直接量数字间的单下划线字符（\_）是没有意义的。在数字直接量中，下划线字符不允许有其它用法。

十进制直接量应以传统的十进制符号表示。实数直接量以小数点的存在来区别。一个指数表示是十的整数幂乘以前面的数以获得所表示的值。十进制直接量及其指数可以包含一个前置符(+或-)。

整数直接量可以以 2、8 或 16 为基底表示。基底应为十进制符号。对于基底是 16 的整数直接量，应使用由字母 A-F 组成的一组扩展数字，它们分别表示传统意义的十进制 10-15。基底数不应包含前置符(+或-)。

布尔数据分别用 0 或 1 的整数直接量或关键字“FALSE”或“TRUE”来表示。

布尔或数字直接量的数据类型通过在直接量前添加类型前缀来规定，该类型前缀由基本数据类型名和符号“#”组成。

表 3-05 数字举例

序号	特性描述	举例
1	整数直接量	-12 , 12345, +32
2	实数直接量	-12.0 , 0.45, 3.1415926
3	带指数的实数直接量	-1.34E-12 或 -1.34 e-12 1.0E+6 或 1.0e+6 1.234E6 或 1.234e6
4	2 进制的直接量	2#1111_1111(十进制 255) 2#1110_0000 (十进制 240)
5	8 进制的直接量	8#377(十进制 255) 8#340 (十进制 240)
6	16 进制的直接量	16# FF 或 16# ff(十进制 255) 16#E0 或 16# e0(十进制 240)
7	布尔 0 和 1	0, 1, FALSE , TRUE

注：关键字 FALSE 和 TRUE 分别对应于布尔值 0 和 1。

# 第 3 章 编程指令

VPC1 系列 PLC 支持 IEC 61131-3 标准的基本指令及其大部分功能/功能块,编程风格符合 IEC 61131-3 标准要求,同时,根据实际的需要,对标准指令作了适当的扩充,可以满足不同用户、多应用领域工程的实际需要。

## 3.1 综述

本章对指令集中的所有指令进行了详细的说明,同时对大多数指令也提供了具体的使用实例。

在 LD 格式中,下文的描述没有具体提及能量流,能量流的含义是固定的,一个网络上各指令的状态控制着能量流在本网络上的流动。我们可以认为它是部分指令(无 EN 操作数的指令)的虚拟输入,并且其类型是 BOOL 型。为了便于描述,在下文中我们将能量流是否到达某点简称为在该点能量流的值为 1 或者 0。另外,在下文中对 EN 操作数和其数据类型也没有说明,因为它们均为 BOOL 型,且含义也是固定的:EN 的意思为“使能”,若某功能/功能块的 EN 值为 1,则该功能/功能块才被执行,能量流继续向前流动。

## 3.2 位指令

### 3.2.1 常开触点指令

功能说明

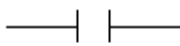
常开触点由 LD、AND、OR 指令提供。

LD 表示逻辑起始,读取指定接点的内容,用于从母线开始的第一个接点或电路块的第一个接点。

AND 用于串联接点,读取指定接点,输出与前面的输入条件之间的“逻辑与”结果,不能直接连在母线上或用于电路块开头。

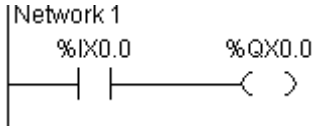
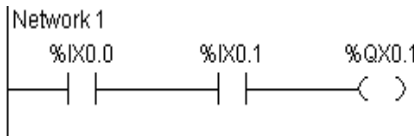
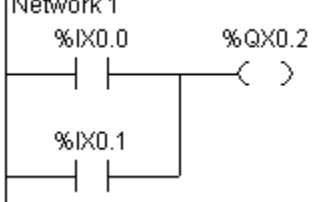
OR 用于并联接点,读取指定接点,输出与前面的输入条件之间的“逻辑或”结果,用于从(连接于母线或电路块的开头的)LD/LDN 指令开始,构成与到本指令之前为止的电路之间的 OR(逻辑或)的接点。

指令及其操作数说明:

	名称	指令格式
LD	常开触点	bit 
	LD	LD bit
IL	AND	AND bit
	OR	OR bit

操作数	输入/输出	数据类型	允许的内存区
bit	输入	BOOL	I、Q、W、S、TR、T、C

指令使用举例：

LD	IL																
	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>ST</td> <td>%QX0.0</td> </tr> </tbody> </table>	条	步	指令	操作数	1	1	LD	%IX0.0		2	ST	%QX0.0				
条	步	指令	操作数														
1	1	LD	%IX0.0														
	2	ST	%QX0.0														
	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>AND</td> <td>%IX0.1</td> </tr> <tr> <td></td> <td>3</td> <td>ST</td> <td>%QX0.1</td> </tr> </tbody> </table>	条	步	指令	操作数	1	1	LD	%IX0.0		2	AND	%IX0.1		3	ST	%QX0.1
条	步	指令	操作数														
1	1	LD	%IX0.0														
	2	AND	%IX0.1														
	3	ST	%QX0.1														
	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>OR</td> <td>%IX0.1</td> </tr> <tr> <td></td> <td>3</td> <td>ST</td> <td>%QX0.2</td> </tr> </tbody> </table>	条	步	指令	操作数	1	1	LD	%IX0.0		2	OR	%IX0.1		3	ST	%QX0.2
条	步	指令	操作数														
1	1	LD	%IX0.0														
	2	OR	%IX0.1														
	3	ST	%QX0.2														

### 3.2.2 常闭触点指令

功能说明

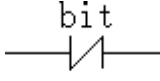
常闭触点由 LDN、ANDN、ORN 指令提供。

LDN 表示逻辑起始，将指定接点的内容取反后读入，用于从母线开始的第一个接点或电路块的第一个接点。

ANDN 用于串联接点，对指定接点内容取反，输出与前面的输入条件之间的“逻辑与”结果，不能直接连在母线上或用于电路块开头。

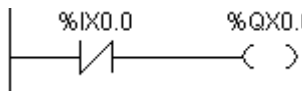
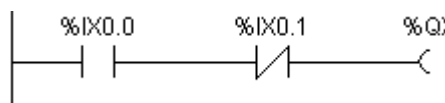
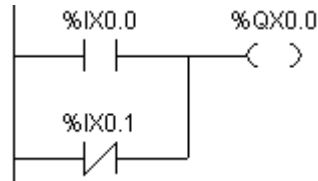
ORN 用于并联接点，对指定接点内容取反，输出与前面的输入条件之间的“逻辑或”结果，用于从（连接于母线或电路块的开头的）LD / LDN 指令开始，构成与到本指令之前为止的电路之间的 OR（逻辑或）的接点。

指令及其操作数说明：

LD	名称	指令格式
	常闭触点	$\text{bit}$ 
IL	LDN	LDN bit
	ANDN	ANDN bit
	ORN	ORN bit

操作数	输入/输出	数据类型	允许的内存区
bit	输入	BOOL	I、Q、W、S、TR、T、C

指令使用举例：

LD	IL																
	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LDN</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>ST</td> <td>%QX0.0</td> </tr> </tbody> </table>	条	步	指令	操作数	1	1	LDN	%IX0.0		2	ST	%QX0.0				
条	步	指令	操作数														
1	1	LDN	%IX0.0														
	2	ST	%QX0.0														
	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>ANDN</td> <td>%IX0.1</td> </tr> <tr> <td></td> <td>3</td> <td>ST</td> <td>%QX0.0</td> </tr> </tbody> </table>	条	步	指令	操作数	1	1	LD	%IX0.0		2	ANDN	%IX0.1		3	ST	%QX0.0
条	步	指令	操作数														
1	1	LD	%IX0.0														
	2	ANDN	%IX0.1														
	3	ST	%QX0.0														
	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>ORN</td> <td>%IX0.1</td> </tr> <tr> <td></td> <td>3</td> <td>ST</td> <td>%QX0.0</td> </tr> </tbody> </table>	条	步	指令	操作数	1	1	LD	%IX0.0		2	ORN	%IX0.1		3	ST	%QX0.0
条	步	指令	操作数														
1	1	LD	%IX0.0														
	2	ORN	%IX0.1														
	3	ST	%QX0.0														

### 3.2.3 普通线圈、复位线圈、置位线圈

功能说明

普通线圈：输出前面的能流值

复位线圈：复位 R 当输入信号为 ON 时，将指定接点置 OFF。

置位线圈：置位 S 当输入信号为 ON 时，将指定接点置 ON。

指令及其操作数说明：

	名称	指令格式
LD	普通线圈	bit —( )
	复位线圈	bit —(R)
	置位线圈	bit —(S)
IL	赋值	ST bit
	复位	R bit
	置位	S bit

操作数	输入/输出	数据类型	允许的内存区
bit	输入	BOOL	I、Q、W、TR

指令使用举例：

LD		IL																					
		<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>ST</td> <td>%QX0.0</td> </tr> <tr> <td></td> <td>3</td> <td>R</td> <td>%QX0.1</td> </tr> <tr> <td></td> <td>4</td> <td>S</td> <td>%QX0.2</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	ST	%QX0.0		3	R	%QX0.1		4	S	%QX0.2
条	步	指令	操作数																				
1	1	LD	%IX0.0																				
	2	ST	%QX0.0																				
	3	R	%QX0.1																				
	4	S	%QX0.2																				

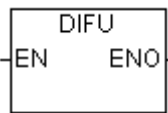
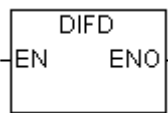
### 3.2.4 边沿（上升沿、下降沿）微分

功能说明

上升沿微分：输入信号的上升沿（OFF→ON）时，指定的接点 1 周期为 ON，1 周期后，本指令执行时为 OFF。

下降沿微分：输入信号的下降沿（ON→OFF）时，指定的接点 1 周期为 ON，1 周期后，本指令执行时为 OFF。

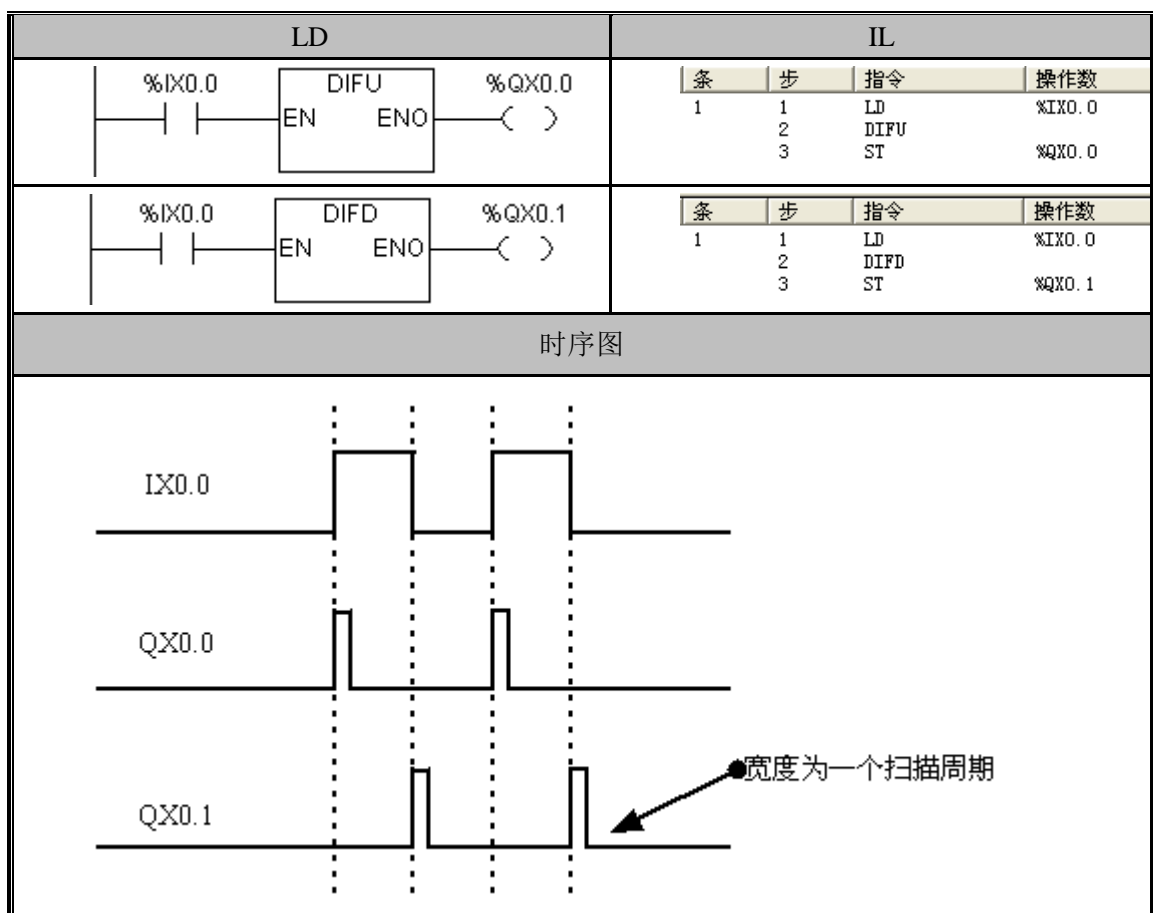
指令及其操作数说明：

	名称	指令格式
LD	上升沿微分	
	下降沿微分	
IL	上升沿微分	DIFU
	下降沿微分	DIFD

DIFU 用于检测 EN 输入的上升沿跳变：如果 EN 值产生了由 0 到 1 的跳变，则输出为 1 并保持一个扫描周期，然后 ENO 将输出为 0。

DIFD 用于检测 EN 输入的下降沿跳变：如果 EN 值产生了由 1 到 0 的跳变，则输出为 1 并保持一个扫描周期，然后 ENO 将输出为 0。

指令及其操作数说明：





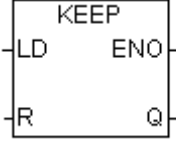
### 3.2.5 保持(KEEP)

#### 功能说明

进行保持继电器（自保持）的动作。

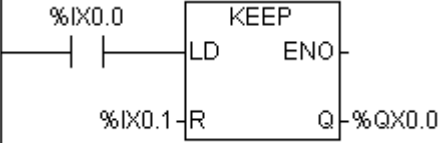
置位输入（输入条件）为 ON 时，保持 Q 所指定的继电器的 ON 状态。复位输入为 ON 时，进入 OFF 状态。置位输入（输入条件）和复位输入同时为 ON 时，复位输入优先。

指令及其操作数说明：

	名称	指令格式
LD	保持	
IL	保持	KEEP R,Q

操作数	输入/输出	数据类型	允许的内存区
LD	输入	BOOL	I、Q、W、TR
R	输入	BOOL	I、Q、W、TR
Q	输出	BOOL	I、Q、W、TR

指令使用举例：

LD		<p>当 IX0.0 为 1 时，保持 QX0.0 输出 1，当 IX0.1 为 1 时，QX0.0 输出 0。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>KEEP</td> <td>%IX0.1 %QX0.0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	KEEP
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	KEEP	%IX0.1 %QX0.0										

### 3.2.6 块与、块或

#### 功能说明

所谓电路块是指，从 LD/LDN 指令开始，到下一个 LD/LDN 指令之前的电路。

ANDLD 表示取电路块间的逻辑积，将本指令之前的电路块和电路块串联连接。串联 3 个以上的电路块时，可以采取顺次连接的形式，即先通过本指令串联 2 个电路块后，再通过本指令串联下一个电路块。另外，也可以在 3 个以上的电路块之后继续配置本指令，进行一次性串联。

ORLD 表示取电路块间的逻辑和，将本指令之前的电路块和电路块并联连接。

并联 3 个以上的电路块时，可以采取顺次连接的形式，即先通过本指令并联 2 个电路块后，再通过本指令并联下一个电路块。此外，也可以在 3 个以上的电路块之后继续配置本指令，进行一次性并联。

指令及其操作数说明:

	名称	指令格式
IL	块与	ANDLD
	块或	ORLD

电路块与、电路块或只有在 IL 中提供。在 IL 语言中只有简单的表达式，而不可能象在 LD、ST 等语言中那样采用复杂的表达式作为操作数，因此没有可视化的图形。

在 IL 程序中，执行“电路块与”之间的指令之前，先将当前值暂存，再执行“电路块与”之间的指令，执行完之后将结果与刚才暂存的值进行“与”运算，并将运算结果作为新的值。

类似地，执行“电路块或”之间的指令之前，首先将当前值暂存，再执行“电路块或”之间的指令，执行完之后将结果与刚才暂存的值进行“或”运算，并将运算结果作为新的值。

指令使用举例:

LD	IL																								
	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>LD</td> <td>%IO.1</td> </tr> <tr> <td></td> <td>3</td> <td>OR</td> <td>%IO.2</td> </tr> <tr> <td></td> <td>4</td> <td>ANDLD</td> <td></td> </tr> <tr> <td></td> <td>5</td> <td>ST</td> <td>%Q0.0</td> </tr> </tbody> </table>	条	步	指令	操作数	1	1	LD	%IX0.0		2	LD	%IO.1		3	OR	%IO.2		4	ANDLD			5	ST	%Q0.0
条	步	指令	操作数																						
1	1	LD	%IX0.0																						
	2	LD	%IO.1																						
	3	OR	%IO.2																						
	4	ANDLD																							
	5	ST	%Q0.0																						
	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>LD</td> <td>%IO.1</td> </tr> <tr> <td></td> <td>3</td> <td>AND</td> <td>%IO.2</td> </tr> <tr> <td></td> <td>4</td> <td>ORLD</td> <td></td> </tr> <tr> <td></td> <td>5</td> <td>ST</td> <td>%Q0.0</td> </tr> </tbody> </table>	条	步	指令	操作数	1	1	LD	%IX0.0		2	LD	%IO.1		3	AND	%IO.2		4	ORLD			5	ST	%Q0.0
条	步	指令	操作数																						
1	1	LD	%IX0.0																						
	2	LD	%IO.1																						
	3	AND	%IO.2																						
	4	ORLD																							
	5	ST	%Q0.0																						

### 3.3 移动指令

#### 3.3.1 MOV (字移动)

功能说明

将 S 的数据或常数传送到目的地 D。S 为常数时，可用于数据设定。

指令及其操作数说明:

	名称	指令格式
LD	MOV	
IL	MOV	MOV S, D

参数	输入/输出	数据类型	允许的内存区
S	输入	INT	I、Q、W、D、P、常量
D	输出	INT	I、Q、W、D、P

指令使用举例：

LD		<p>P_ON 固定为 1，因此 MOV 指令总是执行，常量 100 被赋给 DW100。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>MOV</td> <td>100 %DW100</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	MOV
条	步	指令	操作数										
1	1	LD	P_ON										
	2	MOV	100 %DW100										

### 3.3.2 MOVB（字节移动）

功能说明

将 S 的数据或常数传送到目的地 D。S 为常数时，可用于数据设定。

指令及其操作数说明：

	名称	指令格式
LD	MOVB	
IL	MOVB	MOVB S,D

参数	输入/输出	数据类型	允许的内存区
S	输入	SINT	I、Q、W、常量
D	输出	SINT	I、Q、W

指令使用举例：

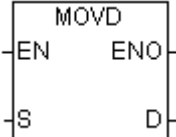
LD		<p>P_ON 固定为 1，因此 MOVB 指令总是执行，常量 100 被赋给 QB10。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>MOVB</td> <td>100 %QB10</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	MOVB
条	步	指令	操作数										
1	1	LD	P_ON										
	2	MOVB	100 %QB10										

### 3.3.3 MOVD (双字移动)

功能说明

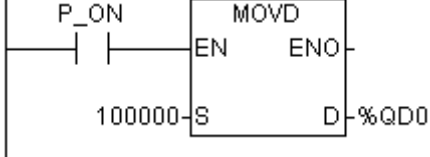
将 S 的数据或常数传送到目的地 D。S 为常数时，可用于数据设定。

指令及其操作数说明：

	名称	指令格式
LD	MOVD	
IL	MOVD	MOVD S,D

参数	输入/输出	数据类型	允许的内存区
S	输入	DINT	I、Q、W、D、常量
D	输出	DINT	I、Q、W、D

指令使用举例：

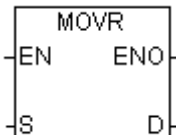
LD		P_ON 固定为 1，因此 MOVD 指令总是执行，常量 100000 被赋给 QD0。											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>MOVD</td> <td>100000 %QD0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	MOVD
条	步	指令	操作数										
1	1	LD	P_ON										
	2	MOVD	100000 %QD0										

### 3.3.4 MOVR (实数移动)

功能说明

将 S 的数据或常数传送到目的地 D。S 为常数时，可用于数据设定。

指令及其操作数说明：

	名称	指令格式
LD	MOVR	
IL	MOVR	MOVR S,D

参数	输入/输出	数据类型	允许的内存区
S	输入	REAL	I、Q、W、D、常量
D	输出	REAL	I、Q、W、D

指令使用举例：

LD		<p>P_ON 固定为 1，因此 MOVVR 指令总是执行，常量 100.2 被赋给 QD0。</p>											
IL	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">条</th> <th style="width: 10%;">步</th> <th style="width: 30%;">指令</th> <th style="width: 50%;">操作数</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">LD</td> <td style="text-align: center;">P_ON</td> </tr> <tr> <td></td> <td style="text-align: center;">2</td> <td style="text-align: center;">MOVVR</td> <td style="text-align: center;">100.2 %QD0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	MOVVR
条	步	指令	操作数										
1	1	LD	P_ON										
	2	MOVVR	100.2 %QD0										

### 3.3.5 BMW (字型块移动)

功能说明

将 S 开始的 N 个字数据传送到目的地 D。

指令及其操作数说明：

	名称	指令格式
LD	BMW	
IL	BMW	BMW S, N, D

参数	输入/输出	数据类型	允许的内存区
S	输入	WORD	I、Q、W
N	输入	BYTE	I、Q、W、常量
D	输出	WORD	I、Q、W

N 的范围为 1 到 255

如果 EN 为 1，则将输入变量 S 开始的 N 个字的内存值赋给输出变量 D 开始的内存。

如果 EN 为 0，则该指令不执行。

指令使用举例：

LD		<p>P_ON 固定为 1，因此 BMW 指令总是执行，WW0 开始的 100 个字被移动到 DW0 开始的内存中</p>											
IL	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">条</th> <th style="width: 10%;">步</th> <th style="width: 30%;">指令</th> <th style="width: 50%;">操作数</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">LD</td> <td style="text-align: center;">P_ON</td> </tr> <tr> <td></td> <td style="text-align: center;">2</td> <td style="text-align: center;">BMW</td> <td style="text-align: center;">%WW0 100 %DW0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	BMW
条	步	指令	操作数										
1	1	LD	P_ON										
	2	BMW	%WW0 100 %DW0										

### 3.3.6 BMB（字节型块移动）

功能说明

将 S 开始的 N 个字数据传送到目的地 D。

指令及其操作数说明：

	名称	指令格式
LD	暂不支持	暂不支持
IL	BMB	BMB S, N, D

参数	输入/输出	数据类型	允许的内存区
S	输入	BYTE	I、Q、W
N	输入	BYTE	I、Q、W、常量
D	输出	BYTE	I、Q、W

N 的范围为 1 到 255

如果 EN 为 1，则将输入变量 S 开始的 N 个字节的内存值赋给输出变量 D 开始的内存。

如果 EN 为 0，则该指令不执行。

指令使用举例：

LD					<p>P_ON 固定为 1，因此 BMB 指令总是执行，QB10 中存放的值为 N，WB0 开始的 N 个字节被移动到 QB1 开始的内存中</p>										
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>BMB</td> <td>%WB0 %QB10 %QB1</td> </tr> </tbody> </table>	条	步	指令		操作数	1	1	LD	P_ON		2	BMB	%WB0 %QB10 %QB1	
条	步	指令	操作数												
1	1	LD	P_ON												
	2	BMB	%WB0 %QB10 %QB1												

### 3.3.7 BMD（双字型块移动）

功能说明：将 S 开始的 N 个字数据传送到目的地址。

指令及其操作数说明：

	名称	指令格式
LD	BMD	
IL	BMD	BMD S, N, D

参数	输入/输出	数据类型	允许的内存区
S	输入	DWORD	I、Q、W
N	输入	BYTE	I、Q、W、常量
D	输出	DWORD	I、Q、W

N 的范围为 1 到 255

如果 EN 为 1，则将输入变量 S 开始的 N 个双字的内存值赋给输出变量 D 开始的内存。

如果 EN 为 0，则该指令不执行。

指令使用举例：

LD		<p>P_ON 固定为 1，因此 BMD 指令总是执行，将 DD0 开始的 200 个双字移动到 DD1000</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>BMD</td> <td>%DD0 200 %DD1000</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	BMD
条	步	指令	操作数										
1	1	LD	P_ON										
	2	BMD	%DD0 200 %DD1000										

### 3.3.8 SWAP（交换字节）

功能说明：将 IN 的字高字节和低字节交换

指令及其操作数说明：

	名称	指令格式
LD	SWAP	
IL	SWAP	SWAP IN

参数	输入/输出	数据类型	允许的内存区
IN	输入	WORD	I、Q、W、D、P

如果 EN 为 1，则该指令被执行：将 IN 的字高字节和低字节交换。如果 EN 为 0，则该指令不执行。

指令使用举例：

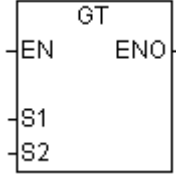
LD		<p>P_ON 恒为 1，因此 SWAP 指令总是执行：将 QW0 中的字节低位与高位交换。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>SWAP</td> <td>%QW0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	SWAP
条	步	指令	操作数										
1	1	LD	P_ON										
	2	SWAP	%QW0										

### 3.4. 比较指令

对 S1 和 S2 进行带符号的比较，比较结果返回到当前位置，作为一个触点值参与网络的计算。

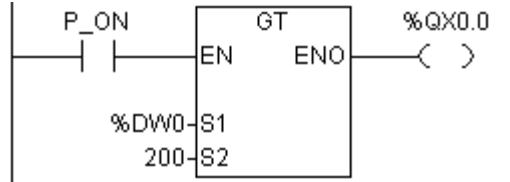
#### 3.4.1 GT (大于)

指令及其操作数说明：

	名称	指令格式
LD	GT	
IL	GT	GT S1,S2

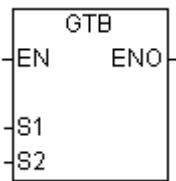
参数	输入/输出	数据类型	允许的内存区
S1	输入	INT	I、Q、W、D、P、常量
S2	输入	INT	I、Q、W、D、P、常量

指令使用举例：

LD		<p>P_ON 固定为 1，因此只要 DW0 的值大于 200，则 ANDLD 的值为 1，则 QX0.0 被置 1，否则 QX0.0 就置为 0。</p>																			
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>GT</td> <td>%DWO 200</td> </tr> <tr> <td></td> <td>3</td> <td>ANDLD</td> <td></td> </tr> <tr> <td></td> <td>4</td> <td>ST</td> <td>%QX0.0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	GT	%DWO 200		3	ANDLD			4	ST
条	步	指令	操作数																		
1	1	LD	P_ON																		
	2	GT	%DWO 200																		
	3	ANDLD																			
	4	ST	%QX0.0																		

#### 3.4.2 GTB (字节型大于)

指令及其操作数说明：

	名称	指令格式
LD	GTB	
IL	GTB	GTB S1,S2



参数	输入/输出	数据类型	允许的内存区
S1	输入	BYTE	I、Q、W、常量
S2	输入	BYTE	I、Q、W、常量

指令使用举例：

LD		<p>P_ON 固定为 1，因此只要 QB2 的值大于 100，则 ANDLD 的值为 1，则 QX0.0 被置 1，否则 QX0.0 就置为 0。</p>																			
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>GTB</td> <td>%QB2</td> </tr> <tr> <td></td> <td>3</td> <td>ANDLD</td> <td>100</td> </tr> <tr> <td></td> <td>4</td> <td>ST</td> <td>%QX0.0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	GTB	%QB2		3	ANDLD	100		4	ST
条	步	指令	操作数																		
1	1	LD	P_ON																		
	2	GTB	%QB2																		
	3	ANDLD	100																		
	4	ST	%QX0.0																		

### 3.4.3 GTD（双字型大于）

指令及其操作数说明：

	名称	指令格式
LD	GTD	
IL	GTD	GTD S1,S2

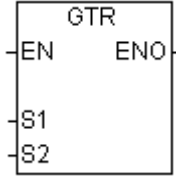
参数	输入/输出	数据类型	允许的内存区
S1	输入	DINT	I、Q、D、W、常量
S2	输入	DINT	I、Q、D、W、常量

指令使用举例：

LD		<p>P_ON 固定为 1，因此只要 QD0 的值大于 100，则 ANDLD 的值为 1，则 QX0.0 被置 1，否则 QX0.0 就置为 0。</p>																			
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>GTD</td> <td>%QD0</td> </tr> <tr> <td></td> <td>3</td> <td>ANDLD</td> <td>100</td> </tr> <tr> <td></td> <td>4</td> <td>ST</td> <td>%QX0.0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	GTD	%QD0		3	ANDLD	100		4	ST
条	步	指令	操作数																		
1	1	LD	P_ON																		
	2	GTD	%QD0																		
	3	ANDLD	100																		
	4	ST	%QX0.0																		

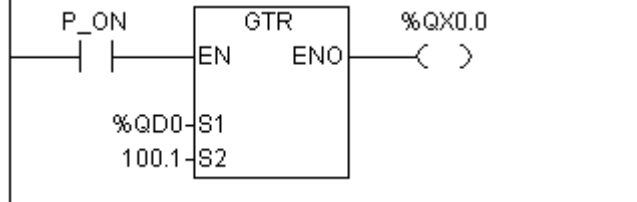
### 3.4.4 GTR (实型大于)

指令及其操作数说明:

	名称	指令格式
LD	GTR	
IL	GTR	GTR S1,S2

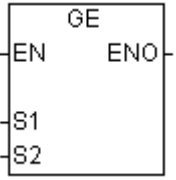
参数	输入/输出	数据类型	允许的内存区
S1	输入	REAL	I、Q、D、W、常量
S2	输入	REAL	I、Q、D、W、常量

指令使用举例:

LD		<p>P_ON 固定为 1, 因此只要 QD0 的值大于 100.1, 则 ANDLD 的值为 1, 则 QX0.0 被置 1, 否则 QX0.0 就置为 0。</p>																			
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>GTR</td> <td>%QD0 100.1</td> </tr> <tr> <td></td> <td>3</td> <td>ANDLD</td> <td></td> </tr> <tr> <td></td> <td>4</td> <td>ST</td> <td>%QX0.0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	GTR	%QD0 100.1		3	ANDLD			4	ST
条	步	指令	操作数																		
1	1	LD	P_ON																		
	2	GTR	%QD0 100.1																		
	3	ANDLD																			
	4	ST	%QX0.0																		

### 3.4.5 GE (大于等于)

指令及其操作数说明:

	名称	指令格式
LD	GE	
IL	GE	GE S1,S2

参数	输入/输出	数据类型	允许的内存区
S1	输入	INT	I、Q、W、D、P、常量
S2	输入	INT	I、Q、W、D、P、常量

指令使用举例：

LD		<p>P_ON 固定为 1，因此只要 DW0 的值大于等于 100，则 ANDLD 的值为 1，则 QX0.0 被置 1，否则 QX0.0 就置为 0。</p>																			
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>GE</td> <td>%DWO 100</td> </tr> <tr> <td></td> <td>3</td> <td>ANDLD</td> <td></td> </tr> <tr> <td></td> <td>4</td> <td>ST</td> <td>%QX0.0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	GE	%DWO 100		3	ANDLD			4	ST
条	步	指令	操作数																		
1	1	LD	P_ON																		
	2	GE	%DWO 100																		
	3	ANDLD																			
	4	ST	%QX0.0																		

### 3.4.6 GEB（字节型大于等于）

指令及其操作数说明：

	名称	指令格式
LD	GEB	
IL	GEB	GEB S1,S2

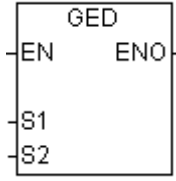
参数	输入/输出	数据类型	允许的内存区
S1	输入	BYTE	I、Q、W、常量
S2	输入	BYTE	I、Q、W、常量

指令使用举例：

LD		<p>P_ON 固定为 1，因此只要 QB2 的值大于等于 100，则 ANDLD 的值为 1，则 QX0.0 被置 1，否则 QX0.0 就置为 0。</p>																			
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>GEB</td> <td>%QB2 100</td> </tr> <tr> <td></td> <td>3</td> <td>ANDLD</td> <td></td> </tr> <tr> <td></td> <td>4</td> <td>ST</td> <td>%QX0.0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	GEB	%QB2 100		3	ANDLD			4	ST
条	步	指令	操作数																		
1	1	LD	P_ON																		
	2	GEB	%QB2 100																		
	3	ANDLD																			
	4	ST	%QX0.0																		

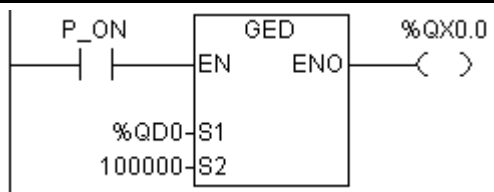
### 3.4.7 GED（双字型大于等于）

指令及其操作数说明：

	名称	指令格式
LD	GED	
IL	GED	GED S1,S2

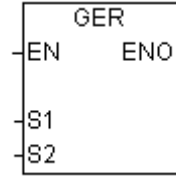
参数	输入/输出	数据类型	允许的内存区
S1	输入	DINT	I、Q、D、W、常量
S2	输入	DINT	I、Q、D、W、常量

指令使用举例：

LD		<p>P_ON 固定为 1，因此只要 QD0 的值大于等于 100000，则 ANDLD 的值为 1，则 QX0.0 被置 1，否则 QX0.0 就置为 0。</p>																			
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>GED</td> <td>%QD0 100000</td> </tr> <tr> <td></td> <td>3</td> <td>ANDLD</td> <td></td> </tr> <tr> <td></td> <td>4</td> <td>ST</td> <td>%QX0.0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	GED	%QD0 100000		3	ANDLD			4	ST
条	步	指令	操作数																		
1	1	LD	P_ON																		
	2	GED	%QD0 100000																		
	3	ANDLD																			
	4	ST	%QX0.0																		

### 3.4.8 GER（实型大于等于）

指令及其操作数说明：

	名称	指令格式
LD	GER	
IL	GER	GER S1,S2

参数	输入/输出	数据类型	允许的内存区
S1	输入	REAL	I、Q、D、W、常量
S2	输入	REAL	I、Q、D、W、常量

指令使用举例：

LD		<p>P_ON 固定为 1，因此只要 QD0 的值大于等于 10000.1，则 ANDLD 的值为 1，则 QX0.0 被置 1，否则 QX0.0 就置为 0。</p>																			
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>GER</td> <td>%QD0</td> </tr> <tr> <td></td> <td>3</td> <td>ANDLD</td> <td>10000.1</td> </tr> <tr> <td></td> <td>4</td> <td>ST</td> <td>%QX0.0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	GER	%QD0		3	ANDLD	10000.1		4	ST
条	步	指令	操作数																		
1	1	LD	P_ON																		
	2	GER	%QD0																		
	3	ANDLD	10000.1																		
	4	ST	%QX0.0																		

### 3.4.9 EQ（等于）

指令及其操作数说明：

	名称	指令格式
LD	EQ	
IL	EQ	EQ S1,S2

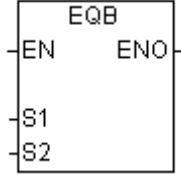
参数	输入/输出	数据类型	允许的内存区
S1	输入	INT	I、Q、W、D、P、常量
S2	输入	INT	I、Q、W、D、P、常量

指令使用举例：

LD		<p>P_ON 固定为 1，因此只要 DWO 的值等于 100，则 ANDLD 的值为 1，则 QX0.0 被置 1，否则 QX0.0 就置为 0。</p>																			
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>EQ</td> <td>%DWO</td> </tr> <tr> <td></td> <td>3</td> <td>ANDLD</td> <td>100</td> </tr> <tr> <td></td> <td>4</td> <td>ST</td> <td>%QX0.0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	EQ	%DWO		3	ANDLD	100		4	ST
条	步	指令	操作数																		
1	1	LD	P_ON																		
	2	EQ	%DWO																		
	3	ANDLD	100																		
	4	ST	%QX0.0																		

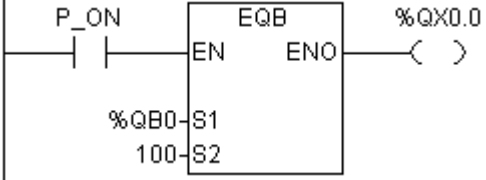
### 3.4.10 EQB（字节型等于）

指令及其操作数说明：

	名称	指令格式
LD	EQB	
IL	EQB	EQB S1,S2

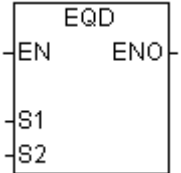
参数	输入/输出	数据类型	允许的内存区
S1	输入	BYTE	I、Q、W、常量
S2	输入	BYTE	I、Q、W、常量

指令使用举例：

LD		<p>P_ON 固定为 1，因此只要 QB0 的值等于 100，则 ANDLD 的值为 1，则 QX0.0 被置 1，否则 QX0.0 就置为 0。</p>																			
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>EQB</td> <td>%QB0 100</td> </tr> <tr> <td></td> <td>3</td> <td>ANDLD</td> <td></td> </tr> <tr> <td></td> <td>4</td> <td>ST</td> <td>%QX0.0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	EQB	%QB0 100		3	ANDLD			4	ST
条	步	指令	操作数																		
1	1	LD	P_ON																		
	2	EQB	%QB0 100																		
	3	ANDLD																			
	4	ST	%QX0.0																		

### 3.4.11 EQD（双字型等于）

指令及其操作数说明：

	名称	指令格式
LD	EQD	
IL	EQD	EQD S1,S2

参数	输入/输出	数据类型	允许的内存区
S1	输入	DINT	I、Q、D、W、常量
S2	输入	DINT	I、Q、D、W、常量

指令使用举例：

LD		<p>P_ON 固定为 1, 因此只要 QB0 的值等于 100000, 则 ANDLD 的值为 1, 则 QX0.0 被置 1, 否则 QX0.0 就置为 0。</p>																			
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>EQD</td> <td>%QD0 100000</td> </tr> <tr> <td></td> <td>3</td> <td>ANDLD</td> <td></td> </tr> <tr> <td></td> <td>4</td> <td>ST</td> <td>%QX0.0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	EQD	%QD0 100000		3	ANDLD			4	ST
条	步	指令	操作数																		
1	1	LD	P_ON																		
	2	EQD	%QD0 100000																		
	3	ANDLD																			
	4	ST	%QX0.0																		

### 3.4.12 EQR (实型大于)

指令及其操作数说明：

	名称	指令格式
LD	EQR	
IL	EQR	EQR S1,S2

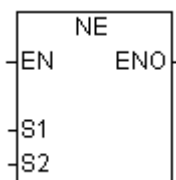
参数	输入/输出	数据类型	允许的内存区
S1	输入	REAL	I、Q、D、W、常量
S2	输入	REAL	I、Q、D、W、常量

指令使用举例：

LD		<p>P_ON 固定为 1, 因此只要 QB0 的值等于 100000.1, 则 ANDLD 的值为 1, 则 QX0.0 被置 1, 否则 QX0.0 就置为 0。</p>																			
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>EQR</td> <td>%QD0 100000.1</td> </tr> <tr> <td></td> <td>3</td> <td>ANDLD</td> <td></td> </tr> <tr> <td></td> <td>4</td> <td>ST</td> <td>%QX0.0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	EQR	%QD0 100000.1		3	ANDLD			4	ST
条	步	指令	操作数																		
1	1	LD	P_ON																		
	2	EQR	%QD0 100000.1																		
	3	ANDLD																			
	4	ST	%QX0.0																		

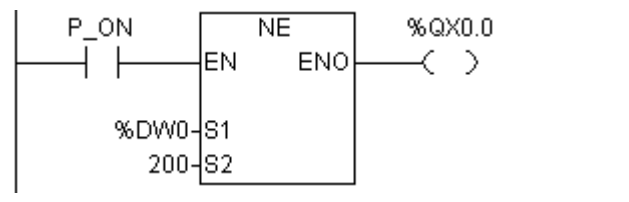
### 3.4.13 NE（不等于）

指令及其操作数说明：

	名称	指令格式
LD	NE	
IL	NE	NE S1,S2

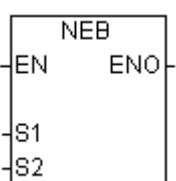
参数	输入/输出	数据类型	允许的内存区
S1	输入	INT	I、Q、W、D、P、常量
S2	输入	INT	I、Q、W、D、P、常量

指令使用举例：

LD		<p>P_ON 固定为 1，因此只要 DW0 的值不等于 200，则 ANDLD 的值为 1，则 QX0.0 被置 1，否则 QX0.0 就置为 0。</p>																			
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>NE</td> <td>%DW0 200</td> </tr> <tr> <td></td> <td>3</td> <td>ANDLD</td> <td></td> </tr> <tr> <td></td> <td>4</td> <td>ST</td> <td>%QX0.0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	NE	%DW0 200		3	ANDLD			4	ST
条	步	指令	操作数																		
1	1	LD	P_ON																		
	2	NE	%DW0 200																		
	3	ANDLD																			
	4	ST	%QX0.0																		

### 3.4.14 NEB（字节型不等于）

指令及其操作数说明：

	名称	指令格式
LD	NEB	
IL	NEB	NEB S1,S2

参数	输入/输出	数据类型	允许的内存区
S1	输入	BYTE	I、Q、W、常量
S2	输入	BYTE	I、Q、W、常量



指令使用举例：

LD		<p>P_ON 固定为 1，因此只要 QB0 的值不等于 100，则 ANDLD 的值为 1，则 QX0.0 被置 1，否则 QX0.0 就置为 0。</p>																			
IL	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">条</th> <th style="width: 10%;">步</th> <th style="width: 20%;">指令</th> <th style="width: 60%;">操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>NED</td> <td>%QB0 100</td> </tr> <tr> <td></td> <td>3</td> <td>ANDLD</td> <td></td> </tr> <tr> <td></td> <td>4</td> <td>ST</td> <td>%QX0.0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	NED	%QB0 100		3	ANDLD			4	ST
条	步	指令	操作数																		
1	1	LD	P_ON																		
	2	NED	%QB0 100																		
	3	ANDLD																			
	4	ST	%QX0.0																		

### 3.4.15 NED（双字型不等于）

指令及其操作数说明：

	名称	指令格式
LD	NED	
IL	NED	NED S1,S2

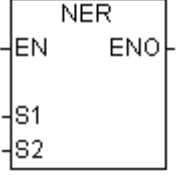
参数	输入/输出	数据类型	允许的内存区
S1	输入	DINT	I、Q、D、W、常量
S2	输入	DINT	I、Q、D、W、常量

指令使用举例：

LD		<p>P_ON 固定为 1，因此只要 QD1 的值不等于 100000，则 ANDLD 的值为 1，则 QX0.0 被置 1，否则 QX0.0 就置为 0。</p>																			
IL	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">条</th> <th style="width: 10%;">步</th> <th style="width: 20%;">指令</th> <th style="width: 60%;">操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>NED</td> <td>%QD1 100000</td> </tr> <tr> <td></td> <td>3</td> <td>ANDLD</td> <td></td> </tr> <tr> <td></td> <td>4</td> <td>ST</td> <td>%QX0.0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	NED	%QD1 100000		3	ANDLD			4	ST
条	步	指令	操作数																		
1	1	LD	P_ON																		
	2	NED	%QD1 100000																		
	3	ANDLD																			
	4	ST	%QX0.0																		

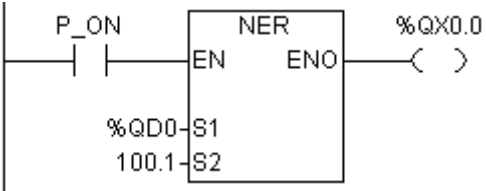
### 3.4.16 NER (实型不等于)

指令及其操作数说明:

	名称	指令格式
LD	NER	
IL	NER	NER S1,S2

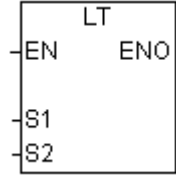
参数	输入/输出	数据类型	允许的内存区
S1	输入	REAL	I、Q、D、W、常量
S2	输入	REAL	I、Q、D、W、常量

指令使用举例:

LD		<p>P_ON 固定为 1, 因此只要 QD0 的值不等于 100.1, 则 ANDLD 的值为 1, 则 QX0.0 被置 1, 否则 QX0.0 就置为 0。</p>																			
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>NER</td> <td>%QD0 100.1</td> </tr> <tr> <td></td> <td>3</td> <td>ANDLD</td> <td></td> </tr> <tr> <td></td> <td>4</td> <td>ST</td> <td>%QX0.0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	NER	%QD0 100.1		3	ANDLD			4	ST
条	步	指令	操作数																		
1	1	LD	P_ON																		
	2	NER	%QD0 100.1																		
	3	ANDLD																			
	4	ST	%QX0.0																		

### 3.4.17 LT (小于)

指令及其操作数说明:

	名称	指令格式
LD	LT	
IL	LT	LT S1,S2

参数	输入/输出	数据类型	允许的内存区
S1	输入	INT	I、Q、W、D、P、常量
S2	输入	INT	I、Q、W、D、P、常量

指令使用举例：

LD		<p>P_ON 固定为 1，因此只要 DW0 的值小于 100，则 ANDLD 的值为 1，则 QX0.0 被置 1，否则 QX0.0 就置为 0。</p>																			
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>LT</td> <td>%DWO 100</td> </tr> <tr> <td></td> <td>3</td> <td>ANDLD</td> <td></td> </tr> <tr> <td></td> <td>4</td> <td>ST</td> <td>%QX0.0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	LT	%DWO 100		3	ANDLD			4	ST
条	步	指令	操作数																		
1	1	LD	P_ON																		
	2	LT	%DWO 100																		
	3	ANDLD																			
	4	ST	%QX0.0																		

### 3.4.18 LTB（字节型小于）

指令及其操作数说明：

	名称	指令格式
LD	LTB	
IL	LTB	LTB S1,S2

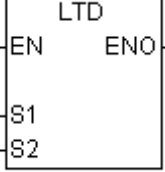
参数	输入/输出	数据类型	允许的内存区
S1	输入	BYTE	I、Q、W、常量
S2	输入	BYTE	I、Q、W、常量

指令使用举例：

LD		<p>P_ON 固定为 1，因此只要 QB0 的值小于 100，则 ANDLD 的值为 1，则 QX0.0 被置 1，否则 QX0.0 就置为 0。</p>																			
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>LTB</td> <td>%QB0 100</td> </tr> <tr> <td></td> <td>3</td> <td>ANDLD</td> <td></td> </tr> <tr> <td></td> <td>4</td> <td>ST</td> <td>%QX0.0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	LTB	%QB0 100		3	ANDLD			4	ST
条	步	指令	操作数																		
1	1	LD	P_ON																		
	2	LTB	%QB0 100																		
	3	ANDLD																			
	4	ST	%QX0.0																		

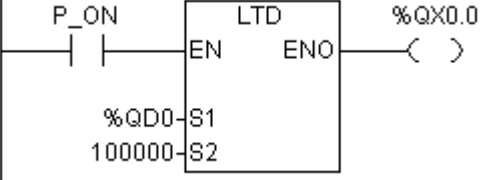
### 3.4.19 LTD (双字型小于)

指令及其操作数说明:

	名称	指令格式
LD	LTD	
IL	LTD	LTD S1,S2

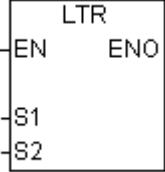
参数	输入/输出	数据类型	允许的内存区
S1	输入	DINT	I、Q、D、W、常量
S2	输入	DINT	I、Q、D、W、常量

指令使用举例:

LD		<p>P_ON 固定为 1，因此只要 QD0 的值小于 100000，则 ANDLD 的值为 1，则 QX0.0 被置 1，否则 QX0.0 就置为 0。</p>																			
IL	<table border="1" data-bbox="459 1218 1066 1391"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>LTD</td> <td>%QD0 100000</td> </tr> <tr> <td></td> <td>3</td> <td>ANDLD</td> <td></td> </tr> <tr> <td></td> <td>4</td> <td>ST</td> <td>%QX0.0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	LTD	%QD0 100000		3	ANDLD			4	ST
条	步	指令	操作数																		
1	1	LD	P_ON																		
	2	LTD	%QD0 100000																		
	3	ANDLD																			
	4	ST	%QX0.0																		

### 3.4.20 LTR (实型小于)

指令及其操作数说明:

	名称	指令格式
LD	LTR	
IL	LTR	LTR S1,S2

参数	输入/输出	数据类型	允许的内存区
S1	输入	REAL	I、Q、D、W、常量
S2	输入	REAL	I、Q、D、W、常量

指令使用举例：

LD		<p>P_ON 固定为 1，因此只要 QD0 的值小于 100000.1，则 ANDLD 的值为 1，则 QX0.0 被置 1，否则 QX0.0 就置为 0。</p>																			
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>LTR</td> <td>%QD0</td> </tr> <tr> <td></td> <td>3</td> <td>ANDLD</td> <td>100000.1</td> </tr> <tr> <td></td> <td>4</td> <td>ST</td> <td>%QX0.0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	LTR	%QD0		3	ANDLD	100000.1		4	ST
条	步	指令	操作数																		
1	1	LD	P_ON																		
	2	LTR	%QD0																		
	3	ANDLD	100000.1																		
	4	ST	%QX0.0																		

### 3.4.21 LE（小于等于）

指令及其操作数说明：

	名称	指令格式
LD	LE	
IL	LE	LE S1,S2

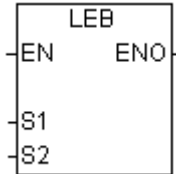
参数	输入/输出	数据类型	允许的内存区
S1	输入	INT	I、Q、W、D、P、常量
S2	输入	INT	I、Q、W、D、P、常量

指令使用举例：

LD		<p>P_ON 固定为 1，因此只要 DW0 的值小于等于 100，则 ANDLD 的值为 1，则 QX0.0 被置 1，否则 QX0.0 就置为 0。</p>																			
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>LE</td> <td>%DWO</td> </tr> <tr> <td></td> <td>3</td> <td>ANDLD</td> <td>100</td> </tr> <tr> <td></td> <td>4</td> <td>ST</td> <td>%QX0.0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	LE	%DWO		3	ANDLD	100		4	ST
条	步	指令	操作数																		
1	1	LD	P_ON																		
	2	LE	%DWO																		
	3	ANDLD	100																		
	4	ST	%QX0.0																		

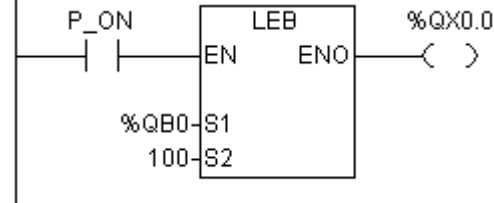
### 3.4.22 LEB（字节型大于等于）

指令及其操作数说明：

	名称	指令格式
LD	LEB	
IL	LEB	LEB S1,S2

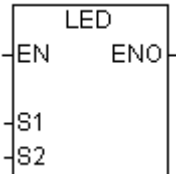
参数	输入/输出	数据类型	允许的内存区
S1	输入	BYTE	I、Q、W、常量
S2	输入	BYTE	I、Q、W、常量

指令使用举例：

LD		<p>P_ON 固定为 1，因此只要 QB0 的值小于等于 100，则 ANDLD 的值为 1，则 QX0.0 被置 1，否则 QX0.0 就置为 0。</p>																			
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>LEB</td> <td>%QB0 100</td> </tr> <tr> <td></td> <td>3</td> <td>ANDLD</td> <td></td> </tr> <tr> <td></td> <td>4</td> <td>ST</td> <td>%QX0.0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	LEB	%QB0 100		3	ANDLD			4	ST
条	步	指令	操作数																		
1	1	LD	P_ON																		
	2	LEB	%QB0 100																		
	3	ANDLD																			
	4	ST	%QX0.0																		

### 3.4.23 LED（双字型小于等于）

指令及其操作数说明：

	名称	指令格式
LD	LED	
IL	LED	LED S1,S2

参数	输入/输出	数据类型	允许的内存区
S1	输入	DINT	I、Q、D、W、常量
S2	输入	DINT	I、Q、D、W、常量

指令使用举例：

LD		<p>P_ON 固定为 1，因此只要 QD0 的值小于等于 100000，则 ANDLD 的值为 1，则 QX0.0 被置 1，否则 QX0.0 就置为 0。</p>																			
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>LED</td> <td>%QD0</td> </tr> <tr> <td></td> <td>3</td> <td>ANDLD</td> <td>100000</td> </tr> <tr> <td></td> <td>4</td> <td>ST</td> <td>%QX0.0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	LED	%QD0		3	ANDLD	100000		4	ST
条	步	指令	操作数																		
1	1	LD	P_ON																		
	2	LED	%QD0																		
	3	ANDLD	100000																		
	4	ST	%QX0.0																		

### 3.4.24 LER（实型小于等于）

指令及其操作数说明：

	名称	指令格式
LD	LER	
IL	LER	LER S1,S2

参数	输入/输出	数据类型	允许的内存区
S1	输入	REAL	I、Q、D、W、常量
S2	输入	REAL	I、Q、D、W、常量

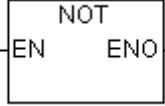
指令使用举例：

LD		<p>P_ON 固定为 1，因此只要 QD0 的值小于等于 100000.1，则 ANDLD 的值为 1，则 QX0.0 被置 1，否则 QX0.0 就置为 0。</p>																			
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>LER</td> <td>%QD0</td> </tr> <tr> <td></td> <td>3</td> <td>ANDLD</td> <td>100000.1</td> </tr> <tr> <td></td> <td>4</td> <td>ST</td> <td>%QX0.0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	LER	%QD0		3	ANDLD	100000.1		4	ST
条	步	指令	操作数																		
1	1	LD	P_ON																		
	2	LER	%QD0																		
	3	ANDLD	100000.1																		
	4	ST	%QX0.0																		

### 3.5 逻辑运算

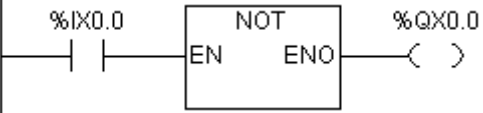
#### NOT（二进制数取反）

功能说明  
 将输入条件取反连接到下一段程序。  
 指令及其操作数说明：

	名称	指令格式
LD	NOT	
IL	NOT	NOT

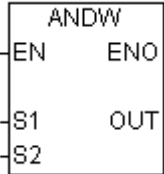
参数	输入/输出	数据类型	允许的内存区
IN	输入	BOOL	I、Q、W、TR

指令使用举例：

LD		将 IX0.0 的值取反，并将结果赋给 QX0.0。															
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>NOT</td> <td></td> </tr> <tr> <td></td> <td>3</td> <td>ST</td> <td>%QX0.0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	NOT			3	ST
条	步	指令	操作数														
1	1	LD	%IX0.0														
	2	NOT															
	3	ST	%QX0.0														

#### ANDW（字与）

功能说明  
 将 S1、S2 的值按位与进行操作，结果输出到 OUT。  
 指令及其操作数说明：

	名称	指令格式
LD	ANDW	
IL	ANDW	ANDW S1,S2,OUT



参数	输入/输出	数据类型	允许的内存区
S1	输入	WORD	I、Q、W、D、P、常量
S2	输入	WORD	I、Q、W、D、P、常量
OUT	输出	WORD	I、Q、W、D、P

指令使用举例：

LD		<p>若 IX0.0 为 0：不执行 ANDW 指令。</p> <p>若 IX0.0 为 1：将 DW0和DW2的值按位进行“与”运算，并将结果赋给 DW4。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>ANDW</td> <td>%DW0 %DW2 %DW4</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	ANDW
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	ANDW	%DW0 %DW2 %DW4										

### ANDB (字节与)

功能说明

将 S1、S2 的值按位与进行操作，结果输出到 OUT。

指令及其操作数说明：

	名称	指令格式
LD	ANDB	
IL	ANDB	ANDB S1,S2,OUT

参数	输入/输出	数据类型	允许的内存区
S1	输入	BYTE	I、Q、W、常量
S2	输入	BYTE	I、Q、W、常量
OUT	输出	BYTE	I、Q、W

指令使用举例：

LD		<p>若 IX0.0 为 0: 不执行 ANDB 指令。</p> <p>若 IX0.0 为 1: 将 QB0 和 QB2 的值按位进行“与”运算，并将结果赋给 QB4。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>ANDB</td> <td>%QB0 %QB2 %QB4</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	ANDB
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	ANDB	%QB0 %QB2 %QB4										

### ANDD(双字与)

功能说明

将 S1、S2 的值按位与进行操作，结果输出到 OUT。

指令及其操作数说明：

	名称	指令格式
LD	ANDD	
IL	ANDD	ANDD S1,S2,D

参数	输入/输出	数据类型	允许的内存区
S1	输入	DWORD	I、Q、W、D、常量
S2	输入	DWORD	I、Q、W、D、常量
OUT	输出	DWORD	I、Q、W、D

指令使用举例：

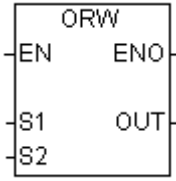
LD		<p>若 IX0.0 为 0: 不执行 ANDD 指令。</p> <p>若 IX0.0 为 1: 将 QD0 和 QD2 的值按位进行“与”运算，并将结果赋给 QD4。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>ANDD</td> <td>%QD0 %QD2 %QD4</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	ANDD
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	ANDD	%QD0 %QD2 %QD4										

### ORW(字或)

功能说明

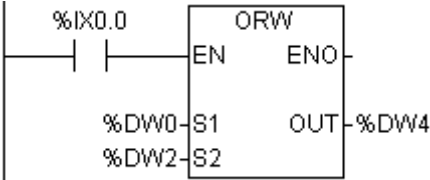
将 S1、S2 的值按位或进行操作，结果输出到 OUT。

指令及其操作数说明：

	名称	指令格式
LD	ORW	
IL	ORW	ORW S1,S2,D

参数	输入/输出	数据类型	允许的内存区
S1	输入	WORD	I、Q、W、D、P、常量
S2	输入	WORD	I、Q、W、D、P、常量
OUT	输出	WORD	I、Q、W、D、P

指令使用举例：

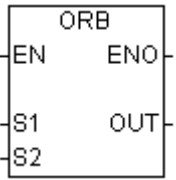
LD		<p>若 IX0.0 为 0：不执行 ORW 指令。 若 IX0.0 为 1：将 DW0 和 DW2 的值按位进行“或”运算，并将结果赋给 DW4。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>ORW</td> <td>%DW0 %DW2 %DW4</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	ORW
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	ORW	%DW0 %DW2 %DW4										

### ORB（字节或）

功能说明

将 S1、S2 的值按位或进行操作，结果输出到 OUT。

指令及其操作数说明：

	名称	指令格式
LD	ORB	
IL	ORB	ORB S1,S2,D

参数	输入/输出	数据类型	允许的内存区
S1	输入	BYTE	I、Q、W、常量
S2	输入	BYTE	I、Q、W、常量
OUT	输出	BYTE	I、Q、W

指令使用举例：

LD		<p>若 IX0.0 为 0: 不执行 ORB 指令。 若 IX0.0 为 1: 将 QB0 和 QB2 的值按位进行“或”运算，并将结果赋给 QB4。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>ORB</td> <td>%QB0 %QB2 %QB4</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	ORB
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	ORB	%QB0 %QB2 %QB4										

### ORD（双字或）

功能说明

将 S1、S2 的值进行按位或操作，结果输出到 OUT。

指令及其操作数说明：

	名称	指令格式
LD	ORD	
IL	ORD	ORD S1,S2,D

参数	输入/输出	数据类型	允许的内存区
S1	输入	DWORD	I、Q、W、D、常量
S2	输入	DWORD	I、Q、W、D、常量
OUT	输出	DWORD	I、Q、W、D

指令使用举例：

LD		<p>若 IX0.0 为 0: 不执行 ORD 指令。</p> <p>若 IX0.0 为 1: 将 QD0 和 QD2 的值按位进行“或”运算, 并将结果赋给 QD4。</p>												
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>ORD</td> <td>%QD0 %QD2 %QD4</td> </tr> </tbody> </table>	条	步	指令	操作数	1	1	LD	%IX0.0		2	ORD	%QD0 %QD2 %QD4	
条	步	指令	操作数											
1	1	LD	%IX0.0											
	2	ORD	%QD0 %QD2 %QD4											

## XOR (异或)

功能说明

对输入的 IN1、IN2 进行异或操作, 将结果输出到 OUT。

指令及其操作数说明:

	名称	指令格式
LD	XOR	
IL	XOR	LD IN1 XOR IN2,OUT

参数	输入/输出	数据类型	允许的内存区
IN1	输入	BOOL	I、Q、W、S、TR、T、C
IN2	输入	BOOL	I、Q、W、S、TR、T、C
OUT	输出	BOOL	I、Q、W、TR

指令使用举例:

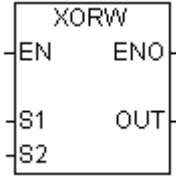
LD		<p>将 WX0.0 和 WX0.2 的值进行“异或”运算, 并将结果赋给 WX0.4。</p>												
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%WX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>XOR</td> <td>%WX0.2 %WX0.4</td> </tr> </tbody> </table>	条	步	指令	操作数	1	1	LD	%WX0.0		2	XOR	%WX0.2 %WX0.4	
条	步	指令	操作数											
1	1	LD	%WX0.0											
	2	XOR	%WX0.2 %WX0.4											

## XORW（字异或）

功能说明

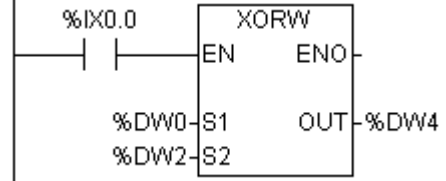
将 S1、S2 的值按位进行异或运算，结果输出到 OUT。

指令及其操作数说明：

	名称	指令格式
LD	XORW	
IL	XORW	XORW S1,S2, OUT

参数	输入/输出	数据类型	允许的内存区
S1	输入	WORD	I、Q、W、D、P、常量
S2	输入	WORD	I、Q、W、D、P、常量
OUT	输出	WORD	I、Q、W、D、P

指令使用举例：

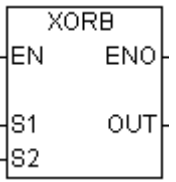
LD		<p>若 IX0.0 为 0：不执行 XORW 指令。 若 IX0.0 为 1：将 DW0 和 DW2 的值按位进行“异或”运算，并将结果赋给 DW4。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>XORW</td> <td>%DW0 %DW2 %DW4</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	XORW
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	XORW	%DW0 %DW2 %DW4										

## XORB（字节异或）

功能说明

将 S1、S2 的值按位进行异或运算，结果输出到 OUT。

指令及其操作数说明：

	名称	指令格式
LD	XORB	
IL	XORB	XORB S1,S2, OUT

参数	输入/输出	数据类型	允许的内存区
S1	输入	BYTE	I、Q、W、常量
S2	输入	BYTE	I、Q、W、常量
OUT	输出	BYTE	I、Q、W

指令使用举例：

LD		<p>若 IX0.0 为 0：不执行 XORB 指令。 若 IX0.0 为 1：将 QB0 和 QB2 的值按位进行“异或”运算，并将结果赋给 QB4。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>XORB</td> <td>%QB0 %QB2 %QB4</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	XORB
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	XORB	%QB0 %QB2 %QB4										

### XORD（双字异或）

功能说明

将 S1、S2 的值按位进行异或运算，结果输出到 OUT。

指令及其操作数说明：

	名称	指令格式
LD	XORD	
IL	XORD	XORD S1,S2,OUT

参数	输入/输出	数据类型	允许的内存区
S1	输入	DWORD	I、Q、W、D、常量
S2	输入	DWORD	I、Q、W、D、常量
OUT	输出	DWORD	I、Q、W、D

指令使用举例：

LD		<p>若 IX0.0 为 0：不执行 XORD 指令。</p> <p>若 IX0.0 为 1：将 QD0 和 QD2 的值按位进行“异或”运算，并将结果赋给 QD4。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>XORD</td> <td>%QD0 %QD2 %QD4</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	XORD
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	XORD	%QD0 %QD2 %QD4										

### INW（反转字）

功能说明

取输入字 IN 的反码，输出到 OUT 指定的字中。

指令及其操作数说明：

	名称	指令格式
LD	INW	
IL	INW	INW IN, OUT

参数	输入/输出	数据类型	允许的内存区
IN	输入	WORD	I、Q、W、D、P、常量
OUT	输出	WORD	I、Q、W、D、P

指令使用举例：

LD		<p>若 IX0.0 为 0：不执行 INW 指令。</p> <p>若 IX0.0 为 1：将常数 43690 取反码得 21845，放在 DW10 中。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>INW</td> <td>43690 %DW10</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	INW
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	INW	43690 %DW10										

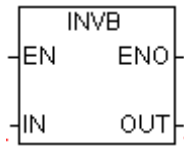
### INB（反转字节）

功能说明

取输入字节 IN 的反码，输出到 OUT 指定的字中。

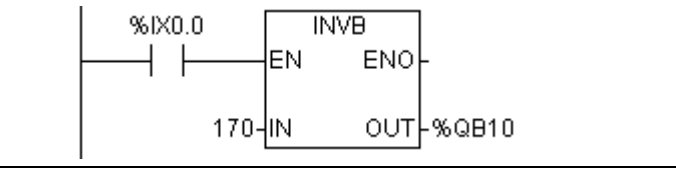
指令及其操作数说明：



	名称	指令格式
LD	INVB	
IL	INVB	INVB IN, OUT

参数	输入/输出	数据类型	允许的内存区
IN	输入	BYTE	I、Q、W、D、P、常量
OUT	输出	BYTE	I、Q、W、D、P

指令使用举例：

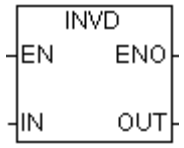
LD		<p>若 IX0.0 为 0: 不执行 INVB 指令。 若 IX0.0 为 1: 将常数 170 取反码得 85, 放在 QB10 中。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>INVB</td> <td>170 %QB10</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	INVB
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	INVB	170 %QB10										

### INVD (反转双字)

功能说明

取输入双字 IN 的反码，输出到 OUT 指定的字中。

指令及其操作数说明：

	名称	指令格式
LD	INVD	
IL	INVD	INVD IN, OUT

参数	输入/输出	数据类型	允许的内存区
IN	输入	DWORD	I、Q、W、常量
OUT	输出	DWORD	I、Q、W

指令使用举例：

LD					若 IX0.0 为 0：不执行 INVW 指令。 若 IX0.0 为 1：将常数 2863312554 取反码得 1431654741，放在 QD1 中。										
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>INVW</td> <td>2863312554 %QD1</td> </tr> </tbody> </table>	条	步	指令		操作数	1	1	LD	%IX0.0		2	INVW	2863312554 %QD1	
条	步	指令	操作数												
1	1	LD	%IX0.0												
	2	INVW	2863312554 %QD1												

### 3.6 移位指令

#### SHL（左移）

功能说明

将 D 的二进制位全部向左移动 C 指定的位数，移出的高位被舍弃并且低位补 0。

指令及其操作数说明：

	名称	指令格式
LD	SHL	
IL	SHL	SHL D,C

参数	输入/输出	数据类型	允许的内存区
D	输入	WORD	I、Q、W、D、P
C	输入	BYTE	I、Q、W、D、P、常量

**注意：**该指令的操作数 C 不能超过 16（即 WORD 型数据的位数）。

指令使用举例：

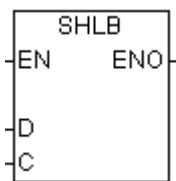
LD					若 IX0.0 为 0：不执行 SHL 指令。 若 IX0.0 为 1：将 QW0 值的全部二进制位左移 1 位，并将结果仍赋给 QW0。										
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>SHL</td> <td>%QW0 1</td> </tr> </tbody> </table>	条	步	指令		操作数	1	1	LD	%IX0.0		2	SHL	%QW0 1	
条	步	指令	操作数												
1	1	LD	%IX0.0												
	2	SHL	%QW0 1												

#### SHLB（字节左移）

功能说明

将 D 的二进制位全部向左移动 C 指定的位数，移出的高位被舍弃并且低位补 0。

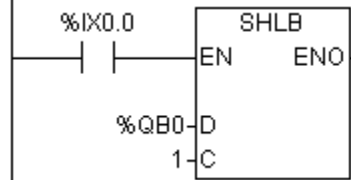
指令及其操作数说明:

	名称	指令格式
LD	SHLB	
IL	SHLB	SHLB D,C

参数	输入/输出	数据类型	允许的内存区
D	输入	BYTE	I、Q、W
C	输入	BYTE	I、Q、W、常量

**注意:** 该指令的操作数 C 不能超过 8 (即 BYTE 型数据的位数)。

指令使用举例:

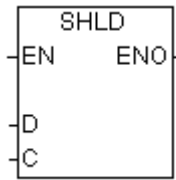
LD		<p>若 IX0.0 为 0: 不执行 SHLB 指令。</p> <p>若 IX0.0 为 1: 将 QB0 值的全部二进制位左移 1 位, 并将结果仍赋给 QB0。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>SHLB</td> <td>%QB0 1</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	SHLB
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	SHLB	%QB0 1										

### SHLD (双字左移)

功能说明

将 D 的二进制位全部向左移动 C 指定的位数, 移出的高位被舍弃并且低位补 0。

指令及其操作数说明:

	名称	指令格式
LD	SHLD	
IL	SHLD	SHLD D,C

参数	输入/输出	数据类型	允许的内存区
D	输入	DWORD	I、Q、W、D
C	输入	BYTE	I、Q、W、D、常量

**注意:** 该指令的操作数 C 不能超过 32 (即 DWORD 型数据的位数)。

指令使用举例:

LD		<p>若 IX0.0 为 0: 不执行 SHLD 指令。</p> <p>若 IX0.0 为 1: 将 QD0 值的全部二进制位左移 1 位, 并将结果仍赋给 QD0。</p>												
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>SHLD</td> <td>%QD0 1</td> </tr> </tbody> </table>	条	步	指令	操作数	1	1	LD	%IX0.0		2	SHLD	%QD0 1	
条	步	指令	操作数											
1	1	LD	%IX0.0											
	2	SHLD	%QD0 1											

### ROL (循环左移)

#### 功能说明

将 D 的二进制位全部向左移动 C 指定的位数, 移出的高位被依次移进低位位置, 最终的结果赋给 D。

指令及其操作数说明:

	名称	指令格式
LD	ROL	
IL	ROL	ROL D,C

参数	输入/输出	数据类型	允许的内存区
D	输入	WORD	I、Q、W、D、P
C	输入	BYTE	I、Q、W、D、P、常量

**注意:** 该指令的操作数 C 不能超过 16 (即 WORD 型数据的位数)。

指令使用举例:

LD		<p>若 IX0.0 为 0: 不执行 ROL 指令。</p> <p>若 IX0.0 为 1: 将 QW0 值的全部二进制位左移 1 位, 移出的高位补到低位位置, 并将结果仍赋给 QW0。</p>												
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>ROL</td> <td>%QW0 1</td> </tr> </tbody> </table>	条	步	指令	操作数	1	1	LD	%IX0.0		2	ROL	%QW0 1	
条	步	指令	操作数											
1	1	LD	%IX0.0											
	2	ROL	%QW0 1											

### ROLB (字节循环左移)

#### 功能说明

将 D 的二进制位全部向左移动 C 指定的位数, 移出的高位被依次移进低位位置, 最终

的结果赋给 D。

指令及其操作数说明：

	名称	指令格式
LD	ROLB	
IL	ROLB	ROLB D,C

参数	输入/输出	数据类型	允许的内存区
D	输入	BYTE	I、Q、W
C	输入	BYTE	I、Q、W、常量

**注意：**该指令的操作数 C 不能超过 8（即 BYTE 型数据的位数）。

指令使用举例：

LD		<p>若 IX0.0 为 0：不执行 ROLB 指令。</p> <p>若 IX0.0 为 1：将 QB0 值的全部二进制位左移 1 位，移出的高位补到低位位置，并将结果仍赋给 QB0。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>ROLB</td> <td>%QB0 1</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	ROLB
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	ROLB	%QB0 1										

### ROLD（双字循环左移）

功能说明

将 D 的二进制位全部向左移动 C 指定的位数，移出的高位被依次移进低位位置，最终的结果赋给 D。

指令及其操作数说明：

	名称	指令格式
LD	ROLD	
IL	ROLD	ROLD D,C

参数	输入/输出	数据类型	允许的内存区
D	输入	DWORD	I、Q、W、D
C	输入	BYTE	I、Q、W、D、常量

**注意：**该指令的操作数 C 不能超过 32（即 DWORD 型数据的位数）。

指令使用举例：

LD		<p>若 IX0.0 为 0：不执行 ROLD 指令。</p> <p>若 IX0.0 为 1：将 QD0 值的全部二进制位左移 1 位，移出的高位补到低位位置，并将结果仍赋给 QD0。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>ROL</td> <td>%QD0 1</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	ROL
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	ROL	%QD0 1										

### SHR（右移）

功能说明

将 D 的二进制位全部向右移动 C 指定的位数，移出的低位被舍弃并且高位补 0，最终的结果赋给 D。

指令及其操作数说明：

	名称	指令格式
LD	SHR	
IL	SHR	SHR D,C

参数	输入/输出	数据类型	允许的内存区
D	输入	WORD	I、Q、W、D、P
C	输入	BYTE	I、Q、W、D、P、常量

**注意：**该指令的操作数 C 不能超过 16（即 WORD 型数据的位数）。

指令使用举例：

LD		<p>若 IX0.0 为 0：不执行 SHR 指令。</p> <p>若 IX0.0 为 1：将 QW0 值的全部二进制位右移 1 位，并将结果仍赋给 QW0。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>SHR</td> <td>%QW0 1</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	SHR
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	SHR	%QW0 1										

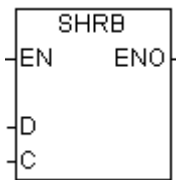
### SHRB（字节右移）

功能说明

将 D 的二进制位全部向右移动 C 指定的位数，移出的低位被舍弃并且高位补 0，最终

的结果赋给 D。

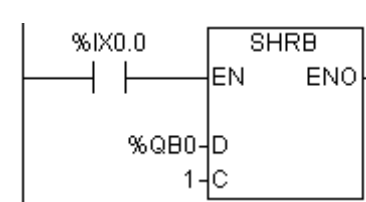
指令及其操作数说明：

	名称	指令格式
LD	SHRB	
IL	SHRB	SHRB D,C

参数	输入/输出	数据类型	允许的内存区
D	输入	BYTE	I、Q、W
C	输入	BYTE	I、Q、W、常量

**注意：**该指令的操作数 C 不能超过 8（即 BYTE 型数据的位数）。

指令使用举例：

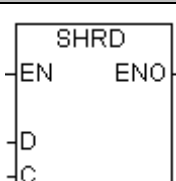
LD		<p>若 IX0.0 为 0：不执行 SHRB 指令。</p> <p>若 IX0.0 为 1：将 QB0 值的全部二进制位右移 1 位，并将结果仍赋给 QB0。</p>												
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>SHRB</td> <td>%QB0 1</td> </tr> </tbody> </table>	条	步	指令	操作数	1	1	LD	%IX0.0		2	SHRB	%QB0 1	
条	步	指令	操作数											
1	1	LD	%IX0.0											
	2	SHRB	%QB0 1											

### SHRD（双字右移）

功能说明

将 D 的二进制位全部向右移动 C 指定的位数，移出的低位被舍弃并且高位补 0，最终的结果赋给 D。

指令及其操作数说明：

	名称	指令格式
LD	SHRD	
IL	SHRD	SHRD D,C

参数	输入/输出	数据类型	允许的内存区
D	输入	DWORD	I、Q、W
C	输入	BYTE	I、Q、W、常量

**注意：**该指令的操作数 C 不能超过 32（即 DWORD 型数据的位数）。

指令使用举例：

LD		<p>若 IX0.0 为 0：不执行 SHR 指令。 若 IX0.0 为 1：将 QD0 值的全部二进制位右移 1 位，并将结果仍赋给 QD0。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>SHR</td> <td>%QD0 1</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	SHR
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	SHR	%QD0 1										

### ROR（循环右移）

功能说明

将 D 的二进制位全部向右移动 C 指定的位数，移出的低位被依次移进高位位置，最终的结果赋给 D。

指令及其操作数说明：

	名称	指令格式
LD	ROR	
IL	ROR	ROR D,C

参数	输入/输出	数据类型	允许的内存区
D	输入	WORD	I、Q、W、D、P
C	输入	BYTE	I、Q、W、D、P、常量

**注意：**该指令的操作数 C 不能超过 16（即 WORD 型数据的位数）。

指令使用举例：

LD		<p>若 IX0.0 为 0：不执行 ROR 指令。 若 IX0.0 为 1：将 QW0 值的全部二进制位右移 1 位，移出的低位补到高位位置，并将结果仍赋给 QW0。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>ROR</td> <td>%QW0 1</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	ROR
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	ROR	%QW0 1										

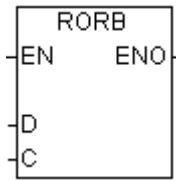


## RORB（字节循环右移）

### 功能说明

将 D 的二进制位全部向右移动 C 指定的位数，移出的低位被依次移进高位位置，最终的结果赋给 D。

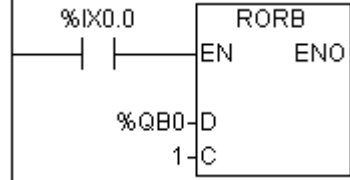
指令及其操作数说明：

	名称	指令格式
LD	RORB	
IL	RORB	RORB D,C

参数	输入/输出	数据类型	允许的内存区
D	输入	BYTE	I、Q、W、
C	输入	BYTE	I、Q、W、常量

**注意：**该指令的操作数 C 不能超过 8（即 BYTE 型数据的位数）。

指令使用举例：

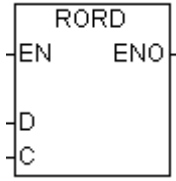
LD		<p>若 IX0.0 为 0：不执行 RORB 指令。</p> <p>若 IX0.0 为 1：将 QB0 值的全部二进制位右移 1 位，移出的低位补到高位位置，并将结果仍赋给 QB0。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>RORB</td> <td>%QB0 1</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	RORB
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	RORB	%QB0 1										

## RORD（循环右移）

### 功能说明

将 D 的二进制位全部向右移动 C 指定的位数，移出的低位被依次移进高位位置，最终的结果赋给 D。

指令及其操作数说明：

	名称	指令格式
LD	RORD	
IL	RORD	RORD D,C

参数	输入/输出	数据类型	允许的内存区
D	输入	DWORD	I、Q、W、D
C	输入	BYTE	I、Q、W、D、常量

**注意：**该指令的操作数 C 不能超过 32（即 DWORD 型数据的位数）。

指令使用举例：

LD		<p>若 IX0.0 为 0：不执行 RORD 指令。</p> <p>若 IX0.0 为 1：将 QD0 值的全部二进制位右移 1 位，移出的低位补到高位位置，并将结果仍赋给 QD0。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>RORD</td> <td>%QD0 1</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	RORD
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	RORD	%QD0 1										

### 3.7 类型转换

#### B2I（BCD 转整型）

功能说明

将 IN 的值由 BCD 数转换成整型数据并赋给 Q。BCD 数的范围是 0~9999，转换后 0~270F（Hex）。

指令及其操作数说明：

	名称	指令格式
LD	B2I	
IL	B2I	B2I IN,Q

参数	输入/输出	数据类型	允许的内存区
IN	输入	WORD	I、Q、W、D、P、常量
Q	输出	WORD	I、Q、W、D、P

指令使用举例：

LD				IX0.0 为 1 时 B2I 指令执行：将 BCD 数 WW1 转换成整数并赋给 DW0。										
IL	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">条</th> <th style="width: 10%;">步</th> <th style="width: 30%;">指令</th> <th style="width: 50%;">操作数</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">LD</td> <td style="text-align: center;">%IX0.0</td> </tr> <tr> <td></td> <td style="text-align: center;">2</td> <td style="text-align: center;">B2I</td> <td style="text-align: center;">%WW1 %DWO</td> </tr> </tbody> </table>	条	步		指令	操作数	1	1	LD	%IX0.0		2	B2I	%WW1 %DWO
条	步	指令	操作数											
1	1	LD	%IX0.0											
	2	B2I	%WW1 %DWO											
结果	<p style="text-align: center;">BCD码转换到整数</p>													

### I2B（整型转 BCD）

功能说明

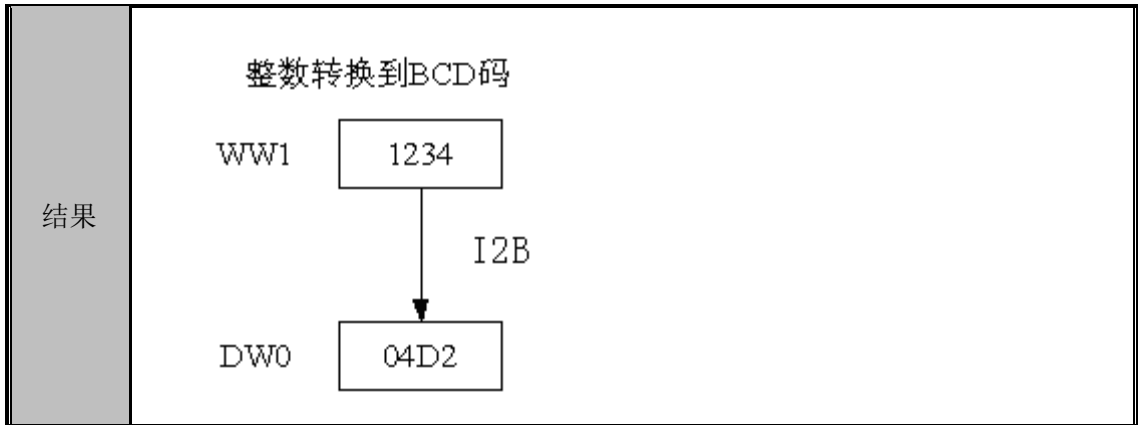
将 IN 的值由整型数据转换成 BCD 数并赋给 Q。整数的范围是 0~270F（Hex），转换后 0~9999。

指令及其操作数说明：

		名称	指令格式
LD		I2B	
IL		I2B	I2B IN,Q
参数	输入/输出	数据类型	允许的内存区
IN	输入	WORD	I、Q、W、D、P、常量
Q	输出	WORD	I、Q、W、D、P

指令使用举例：

LD				IX0.0 为 1 时 I2B 指令执行：将整数 WW1 转换成 BCD 数并赋给 DW0。										
IL	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">条</th> <th style="width: 10%;">步</th> <th style="width: 30%;">指令</th> <th style="width: 50%;">操作数</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">LD</td> <td style="text-align: center;">%IX0.0</td> </tr> <tr> <td></td> <td style="text-align: center;">2</td> <td style="text-align: center;">I2B</td> <td style="text-align: center;">%WW1 %DWO</td> </tr> </tbody> </table>	条	步		指令	操作数	1	1	LD	%IX0.0		2	I2B	%WW1 %DWO
条	步	指令	操作数											
1	1	LD	%IX0.0											
	2	I2B	%WW1 %DWO											

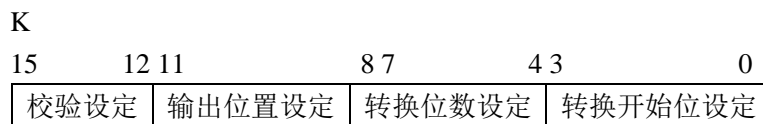
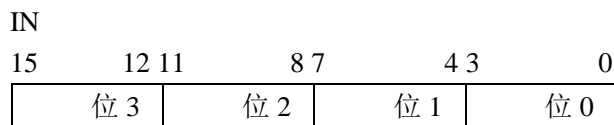


**ASC (ASCII 代码转换)**

功能说明  
 将整数 IN 的数值指定的位转换为 ASCII 字符  
 指令及其操作数说明：

	名称	指令格式
LD	ASC	
IL	ASC	ASC IN, K, Q

参数	输入/输出	数据类型	允许的内存区
IN	输入	WORD	I、Q、W、D、P
K	输入	WORD	I、Q、W、常量
Q	输出	WORD	I、Q、W



校验设定：

- 无校验
- 偶校验
- 奇校验

输出位置设定：

低字节开始输出

高字节开始输出

转换位数设定:

1 位

2 位

3 位

4 位

转换开始位设定:

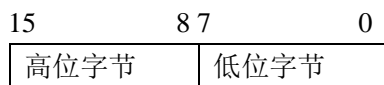
位 0 开始

位 1 开始

位 2 开始

位 3 开始:

Q



指令使用举例:

D		<p>P_ON 恒为 1，因此 ASC 指令总是执行：将 WW0 中的整数按设置转换成 ASCII 字符数组存放在 QW1 开头通道中。</p>											
L	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>ASC</td> <td>%WW0 289 %QW1</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	ASC
条	步	指令	操作数										
1	1	LD	P_ON										
	2	ASC	%WW0 289 %QW1										

### BTI (BYTE 转 INT)

功能说明:

将 IN 的数值，从 BYTE 型提升为 INT 型

指令及其操作数说明:

	名称	指令格式
LD	BTI	
IL	BTI	BTI IN,Q

参数	输入/输出	数据类型	允许的内存区
IN	输入	BYTE	I、Q、W、常量
Q	输出	INT	I、Q、W、D、P

指令使用举例：

LD		<p>P_ON 恒为 1，因此 BTI 指令总是执行：将 QB2 中的字节型数转换成整数并赋给 QW3。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>BTI</td> <td>%QB2 %QW3</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	BTI
条	步	指令	操作数										
1	1	LD	P_ON										
	2	BTI	%QB2 %QW3										

### ITA（整型转换为 ASCII）

功能说明

将整数 IN 的数值转换为 ASCII 字符数组，格式 FMT 指定小数点右边的转换精度，以及小数点以逗号还是句号显示，转换结果存放在以 Q 开头的 8 个连续字节中。

FMT 格式说明：

MSB				LSB			
7	6	5	4	3	2	1	0
0	0	0	0	c	n	n	n

其中：c 为 0 代表小数点为句号，c 为 1 代表小数点为逗号，nnn 为小数点右边的位数。

指令及其操作数说明：

	名称	指令格式
LD	ITA	
IL	ITA	ITA IN, FMT, Q

参数	输入/输出	数据类型	允许的内存区
IN	输入	INT	I、Q、W、D、P
FMT	输入	BYTE	I、Q、W、常量
Q	输出	BYTE	I、Q、W

指令使用举例：

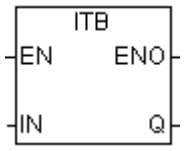
LD		<p>P_ON 恒为 1，因此 ITA 指令总是执行：将 WW0 中的整数转换成 ASCII 字符数组存放在 QB0 开头的 8 个字节中。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>ITA</td> <td>%WW0 3 %QB0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	ITA
条	步	指令	操作数										
1	1	LD	P_ON										
	2	ITA	%WW0 3 %QB0										

若 WW0 中为整数 12345，则 QB0 开头的 8 个连续字节中分别存放着

QB0	QB1	QB2	QB3	QB4	QB5	QB6	QB7
32	32	49	50	46	51	52	53

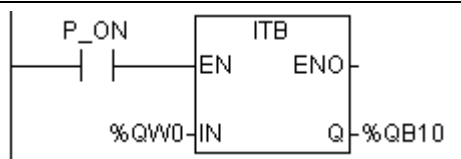
### ITB (INT 转 BYTE)

功能说明  
 将 IN 的数值，从 INT 型转换为 BYTE 型  
 指令及其操作数说明：

	名称	指令格式
LD	ITB	
IL	ITB	ITB IN,Q

参数	输入/输出	数据类型	允许的内存区
IN	输入	INT	I、Q、W、D、P、常量
Q	输出	BYTE	I、Q、W

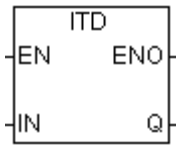
指令使用举例：

LD		P_ON 恒为 1，因此 ITB 指令总是执行：将 QW0 中的整型数转换成字节型并赋给 QB10。											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>ITB</td> <td>%QW0 %QB10</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	ITB
条	步	指令	操作数										
1	1	LD	P_ON										
	2	ITB	%QW0 %QB10										

**注意：**由整形转换为字节型，超出字节型表示范围的数值将被丢失。例如：整型 0x3EB 转换为字节型后将变为 0xEB。

### ITD (INT 转 DINT)

功能说明  
 将 IN 的数值，从 INT 型转换为 DINT 型  
 指令及其操作数说明：

	名称	指令格式
LD	ITD	
IL	ITD	ITD IN,Q

参数	输入/输出	数据类型	允许的内存区
IN	输入	INT	I、Q、W、D、P、常量
Q	输出	DINT	I、Q、W、D

指令使用举例：

LD				<p>P_ON 恒为 1，因此 ITD 指令总是执行：将 QW0 中的整型数转换成双整型数并赋给 QD10。</p>										
	IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>ITD</td> <td>%QW0 %QD10</td> </tr> </tbody> </table>	条		步	指令	操作数	1	1	LD	P_ON		2	ITD
条	步	指令	操作数											
1	1	LD	P_ON											
	2	ITD	%QW0 %QD10											

### DTI (DINT 转 INT)

功能说明

将 IN 的数值，从 DINT 型转换为 INT 型

指令及其操作数说明：

	名称	指令格式
LD	DTI	
IL	DTI	DTI IN,Q

参数	输入/输出	数据类型	允许的内存区
IN	输入	DINT	I、Q、W、D、常量
Q	输出	INT	I、Q、W、D

指令使用举例：

LD				<p>P_ON 恒为 1，因此 DTI 指令总是执行：将 QD0 中的双整型数转换成整型数并赋给 QW10。</p>										
	IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>DTI</td> <td>%QD0 %QW10</td> </tr> </tbody> </table>	条		步	指令	操作数	1	1	LD	P_ON		2	DTI
条	步	指令	操作数											
1	1	LD	P_ON											
	2	DTI	%QD0 %QW10											

注意：DINT 转换为 INT 可能会丢失数据，INT 型截取 DINT 型的低 16 位。

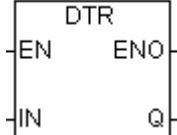


## DTR (DINT 转 REAL)

功能说明

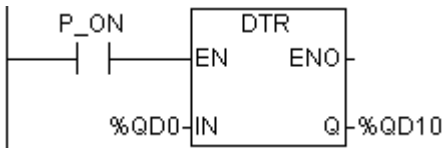
将 IN 的数值，从 DINT 型转换为 REAL 型

指令及其操作数说明：

	名称	指令格式
LD	DTR	
IL	DTR	DTR IN,Q

参数	输入/输出	数据类型	允许的内存区
IN	输入	DINT	I、Q、W、D、常量
Q	输出	REAL	I、Q、W、D

指令使用举例：

LD		<p>P_ON 恒为 1，因此 DTR 指令总是执行：将 QD0 中的双整型数转换成实型数并赋给 QD10。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>DTR</td> <td>%QD0 %QD10</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	DTR
条	步	指令	操作数										
1	1	LD	P_ON										
	2	DTR	%QD0 %QD10										

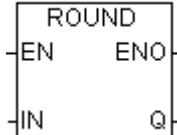
注意：DINT 转换为 REAL 可能会丢失数据精度，REAL 型只有 6 位有效数字。

## ROUND (实型转为双整型)

功能说明

将 IN 的数值，从 REAL 型转换为 DINT 型，小数部分四舍五入。

指令及其操作数说明：

	名称	指令格式
LD	ROUND	
IL	ROUND	ROUND IN,Q

参数	输入/输出	数据类型	允许的内存区
IN	输入	REAL	I、Q、W、D、常量
Q	输出	DINT	I、Q、W、D

指令使用举例：

LD		<p>P_ON 恒为 1, 因此 ROUND 指令总是执行：将#10.55 转换成 S 双正型数并赋给 QD10。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>ROUND</td> <td>10.55 %QD10</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	ROUND
条	步	指令	操作数										
1	1	LD	P_ON										
	2	ROUND	10.55 %QD10										

### TRUNC (实型转为双整型)

功能说明

将 IN 的数值，从 REAL 型转换为 DINT 型，小数丢弃。

指令及其操作数说明：

	名称	指令格式
LD	TRUNC	
IL	TRUNC	TRUNC IN,Q

参数	输入/输出	数据类型	允许的内存区
IN	输入	REAL	I、Q、W、D、常量
Q	输出	DINT	I、Q、W、D

指令使用举例：

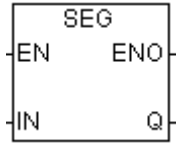
LD		<p>P_ON 恒为 1, 因此 TRUNC 指令总是执行：将#10.55 转换成 S 双正型数并赋给 QD10。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>TRUNC</td> <td>10.55 %QD10</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	TRUNC
条	步	指令	操作数										
1	1	LD	P_ON										
	2	TRUNC	10.55 %QD10										

### SEG (7 段数码管显示)

功能说明

将 IN 指定的字符转换为 7 段数码管显示位模式。

指令及其操作数说明：

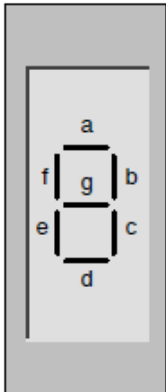
	名称	指令格式
LD	SEG	
IL	SEG	SEG IN,Q

参数	输入/输出	数据类型	允许的内存区
IN	输入	BYTE	I、Q、W、常量
Q	输出	BYTE	I、Q、W

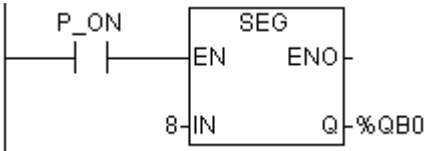
IN 的范围为 0 到 15

转换对应表

(IN) LSD	段 显示	(OUT) -gfe dcba	(IN) LSD	段 显示	(OUT) -gfe dcba
0		0011 1111	8		0111 1111
1		0000 0110	9		0110 0111
2		0101 1011	A		0111 0111
3		0100 1111	B		0111 1100
4		0110 0110	C		0011 1001
5		0110 1101	D		0101 1110
6		0111 1101	E		0111 1001
7		0000 0111	F		0111 0001



指令使用举例：

LD		<p>P_ON 恒为 1，因此 SEG 指令总是执行：将常数 8 转换为 7 段数码管显示位模式赋给 QB0。</p>											
IL	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td style="text-align: center;">2</td> <td>SEG</td> <td>8 %QB0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	SEG
条	步	指令	操作数										
1	1	LD	P_ON										
	2	SEG	8 %QB0										

### ATH (ASCII 字符串转换为十六进制数)

功能说明

将 IN 开始的指定长度的 ASCII 字符串转换为十六进制数，每个 ASCII 码转换后占半个字节。最大字符串长度为 255。

指令及其操作数说明：

	名称	指令格式
LD	ATH	
IL	ATH	ATH IN, LEN, D

参数	输入/输出	数据类型	允许的内存区
IN	输入	BYTE	I、Q、W、常量
LEN	输出	BYTE	I、Q、W、常量
D	输出	BYTE	I、Q、W

注意：LEN 的范围为 0 到 255。

指令使用举例：

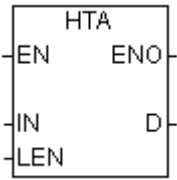
LD		<p>P_ON 固定为 1，因此 ATH 指令总是执行，从 WB0 开始的 3 个 ASCII 字符被转换为 16 进制数，存放在 QB0 开始的内存中。</p>													
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>ATH</td> <td>%WB0 3 %QB0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	ATH	%WB0 3 %QB0	
条	步	指令	操作数												
1	1	LD	P_ON												
	2	ATH	%WB0 3 %QB0												
结果	<table> <tr> <td>WB0</td> <td>WB1</td> <td>WB2</td> </tr> <tr> <td>51</td> <td>67</td> <td>65</td> </tr> <tr> <td>QB0</td> <td>QB1</td> <td></td> </tr> <tr> <td>0x3C</td> <td>0xAX</td> <td></td> </tr> </table> <p>X 表示内存值未变动</p>		WB0	WB1	WB2	51	67	65	QB0	QB1		0x3C	0xAX		
WB0	WB1	WB2													
51	67	65													
QB0	QB1														
0x3C	0xAX														

### HTA（十六进制数转换为 ASCII 字符串）

#### 功能说明

将 IN 开始的指定长度的十六进制数转换为 ASCII 字符串，每个十六进制数转换后占 1 字节。转换成最大字符串长度为 255。

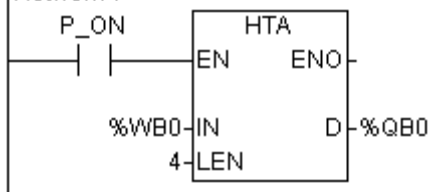
指令及其操作数说明：

	名称	指令格式
LD	HTA	
IL	HTA	HTA IN, LEN, D

参数	输入/输出	数据类型	允许的内存区
IN	输入	BYTE	I、Q、W、常量
LEN	输出	BYTE	I、Q、W、常量
D	输出	BYTE	I、Q、W

**注意：**LEN 的范围为 0 到 255。

指令使用举例：

LD		<p>P_ON 固定为 1，因此 HTA 指令总是执行，从 WB0 开始的 2 个 16 进制数被转换为 4 个 ASCII 字符，存放在 QB0 开始的内存中。</p>																	
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>HTA</td> <td>%WB0 4 %QB0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	HTA	%WB0 4 %QB0					
条	步	指令	操作数																
1	1	LD	P_ON																
	2	HTA	%WB0 4 %QB0																
结果	<table> <tr> <td>WB0</td> <td colspan="3">WB1</td> </tr> <tr> <td>0x3C</td> <td colspan="3">0xA0</td> </tr> <tr> <td>QB0</td> <td>QB1</td> <td>QB2</td> <td>QB3</td> </tr> <tr> <td>51</td> <td>67</td> <td>65</td> <td>48</td> </tr> </table>			WB0	WB1			0x3C	0xA0			QB0	QB1	QB2	QB3	51	67	65	48
WB0	WB1																		
0x3C	0xA0																		
QB0	QB1	QB2	QB3																
51	67	65	48																

### DTA（双字型整数转换为 ASCII 字符串）

功能说明

将 IN 指定的双整型十进制数转换为 ASCII 字符串。转换成的字符串存放在以 D 开头的连续 12 个字节里面

指令及其操作数说明:

	名称	指令格式
LD	DTA	
IL	DTA	DTA IN, FMT, D

参数	输入/输出	数据类型	允许的内存区
IN	输入	DINT	I、Q、W、常量
FMT	输出	BYTE	I、Q、W、常量
D	输出	BYTE	I、Q、W

FMT 格式说明:

MSB				LSB			
7	6	5	4	3	2	1	0
0	0	0	0	c	n	n	n

其中: c 为 0 代表小数点为句号, c 为 1 代表小数点为逗号, nnn 为小数点右边的位数。

指令使用举例:

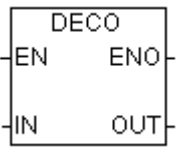
LD		<p>P_ON 固定为 1, 因此 DTA 指令总是执行, WD0 存放的双整型数被转换为 ASCII 字符串存放在 QB0 开始的 12 个字节中。</p>																								
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>DTA</td> <td>%WD0 12 %QB0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	DTA	%WD0 12 %QB0												
条	步	指令	操作数																							
1	1	LD	P_ON																							
	2	DTA	%WD0 12 %QB0																							
	<p>WD0</p> <p>-1234567</p> <table border="1"> <thead> <tr> <th>QB0</th> <th>QB1</th> <th>QB2</th> <th>QB3</th> <th>QB4</th> <th>QB5</th> </tr> </thead> <tbody> <tr> <td>32</td> <td>32</td> <td>32</td> <td>45</td> <td>49</td> <td>50</td> </tr> <tr> <th>QB6</th> <th>QB7</th> <th>QB8</th> <th>QB9</th> <th>QB10</th> <th>QB11</th> </tr> <tr> <td>51</td> <td>44</td> <td>52</td> <td>53</td> <td>54</td> <td>55</td> </tr> </tbody> </table>		QB0	QB1	QB2	QB3	QB4	QB5	32	32	32	45	49	50	QB6	QB7	QB8	QB9	QB10	QB11	51	44	52	53	54	55
QB0	QB1	QB2	QB3	QB4	QB5																					
32	32	32	45	49	50																					
QB6	QB7	QB8	QB9	QB10	QB11																					
51	44	52	53	54	55																					

## DECO (解码)

功能说明

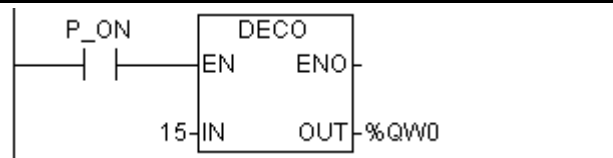
将 IN 指定的字节中低 4 位的数值解码, 将 OUT 的字中的指定位置一, 其他位置零。

指令及其操作数说明:

	名称	指令格式
LD	DECO	
IL	DECO	DECO IN, OUT

参数	输入/输出	数据类型	允许的内存区
IN	输出	BYTE	I、Q、W、常量
OUT	输出	WORD	I、Q、W、D、P

指令使用举例：

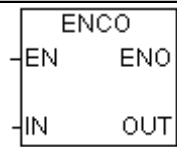
LD		<p>P_ON 固定为 1，因此 DECO 指令总是执行，#15 被解码为将 QW0 的第十五位置一，所以执行后，QW0 的值为 32768。</p>											
IL	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>DECO</td> <td>15 %QW0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	DECO
条	步	指令	操作数										
1	1	LD	P_ON										
	2	DECO	15 %QW0										

## ENCO (编码)

功能说明

将 IN 指定的字中最低一个 1 的位号写到输入字节 OUT 中。

指令及其操作数说明：

	名称	指令格式
LD	ENCO	
IL	ENCO	ENCO IN, OUT

参数	输入/输出	数据类型	允许的内存区
IN	输出	WORD	I、Q、W、常量
OUT	输出	BYTE	I、Q、W、D、P

指令使用举例：

LD			<p>P_ON 固定为 1，因此 ENCO 指令总是执行，#5 最低的 1 为第 0 位，所以 QB0 的值为 0。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>ENCO</td> <td>5 %QB0</td> </tr> </tbody> </table>	条		步	指令	操作数	1	1	LD	P_ON		2	ENCO	5 %QB0
条	步	指令	操作数											
1	1	LD	P_ON											
	2	ENCO	5 %QB0											

### 3.8 数学运算

#### ADD\_I (加法)

功能说明

将 IN1 所指定的数据和 IN2 所指定的数据进行有符号数加法运算，将结果输出到 OUT，不带进位。

指令及其操作数说明：

	名称	指令格式
LD	ADD_I	
IL	ADD_I	ADD_I IN1,IN2,OUT

参数	输入/输出	数据类型	允许的内存区
IN1	输入	INT	I、Q、W、D、P、常量
IN2	输入	INT	I、Q、W、D、P、常量
OUT	输出	INT	I、Q、W、D、P

指令使用举例：

LD			<p>若 IX0.0 为 0：不执行 ADD_I 指令。 若 IX0.0 为 1：DW4 = DW0 + 345。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>ADD_I</td> <td>%DW0 345 %DW4</td> </tr> </tbody> </table>	条		步	指令	操作数	1	1	LD	%IX0.0		2	ADD_I	%DW0 345 %DW4
条	步	指令	操作数											
1	1	LD	%IX0.0											
	2	ADD_I	%DW0 345 %DW4											

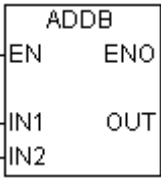


### ADDB (字节型加法)

功能说明

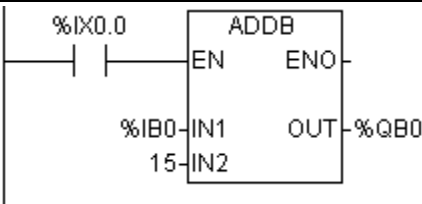
将 IN1 所指定的数据和 IN2 所指定的数据进行有符号数的加法运算，将结果输出到 OUT，不带进位。

指令及其操作数说明：

	名称	指令格式
LD	ADDB	
IL	ADDB	ADDB IN1,IN2,OUT

参数	输入/输出	数据类型	允许的内存区
IN1	输入	SINT	I、Q、W、常量
IN2	输入	SINT	I、Q、W、常量
OUT	输出	SINT	I、Q、W

指令使用举例：

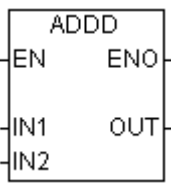
LD		<p>若 IX0.0 为 0：不执行 ADDB 指令。 若 IX0.0 为 1：QB0 = IB0 + 15。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>ADDB</td> <td>%IB0 15 %QB0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	ADDB
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	ADDB	%IB0 15 %QB0										

### ADDD (双字型加法)

功能说明

将 IN1 所指定的数据和 IN2 所指定的数据进行有符号数加法运算，将结果输出到 OUT，不带进位。

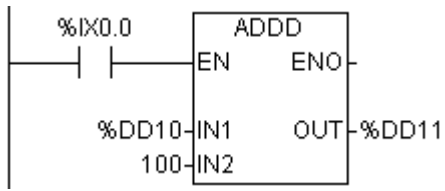
指令及其操作数说明：

	名称	指令格式
LD	ADDD	
IL	ADDD	ADDD IN1,IN2,OUT

参数	输入/输出	数据类型	允许的内存区
IN1	输入	DINT	I、Q、W、D、常量
IN2	输入	DINT	I、Q、W、D、常量
OUT	输出	DINT	I、Q、W、D

若 EN 值为 1，则 ADDD 指令的功能是： $OUT = IN1 + IN2$ 。

指令使用举例：

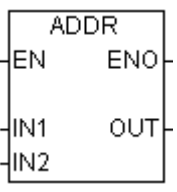
LD		<p>若 IX0.0 为 0：不执行 ADDD 指令。 若 IX0.0 为 1：DD11 = 100 + DD10。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>ADDD</td> <td>%DD10 100 %DD11</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	ADDD
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	ADDD	%DD10 100 %DD11										

### ADDR（实型加法）

功能说明

将 IN1 所指定的数据和 IN2 所指定的数据进行有符号数加法运算，将结果输出到 OUT，不带进位。

指令及其操作数说明：

	名称	指令格式
LD	ADDR	
IL	ADDR	ADDR IN1,IN2,OUT

参数	输入/输出	数据类型	允许的内存区
IN1	输入	REAL	I、Q、W、D、常量
IN2	输入	REAL	I、Q、W、D、常量
OUT	输出	REAL	I、Q、W、D

指令使用举例：

LD		<p>若 IX0.0 为 0：不执行 ADDR 指令。 若 IX0.0 为 1：DD11 = DD10 + 10.25。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>ADDR</td> <td>%DD10 10.25 %DD11</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	ADDR
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	ADDR	%DD10 10.25 %DD11										

### SUB\_I (减法)

功能说明

将 IN1 所指定的数据和 IN2 所指定的数据进行有符号数的减法运算，将结果输出到 OUT，不带借位。

指令及其操作数说明：

	名称	指令格式
LD	SUB_I	
IL	SUB_I	SUB_I IN1,IN2,OUT

参数	输入/输出	数据类型	允许的内存区
IN1	输入	INT	I、Q、W、D、P、常量
IN2	输入	INT	I、Q、W、D、P、常量
OUT	输出	INT	I、Q、W、D、P

指令使用举例：

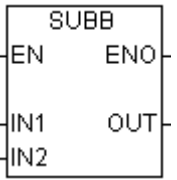
LD		<p>若 IX0.0 为 0：不执行 SUB_I 指令。 若 IX0.0 为 1：DW4 = DW0 - 345。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>SUB_I</td> <td>%DW0 345 %DW4</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	SUB_I
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	SUB_I	%DW0 345 %DW4										

### SUBB (字节型减法)

#### 功能说明

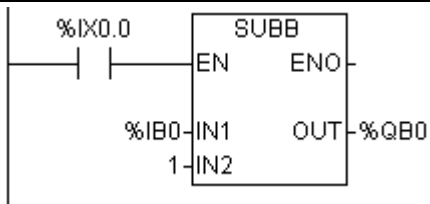
将 IN1 所指定的数据和 IN2 所指定的数据进行有符号数的减法运算，将结果输出到 OUT，不带借位。

指令及其操作数说明：

	名称	指令格式
LD	SUBB	
IL	SUBB	SUBB IN1,IN2,OUT

参数	输入/输出	数据类型	允许的内存区
IN1	输入	SINT	I、Q、W、常量
IN2	输入	SINT	I、Q、W、D、P、常量
OUT	输出	SINT	I、Q、W

指令使用举例：

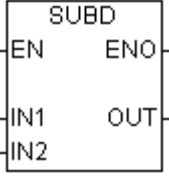
LD		<p>若 IX0.0 为 0：不执行 SUBB 指令。 若 IX0.0 为 1：QB0 = IB0 - 1。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>SUBB</td> <td>%IB0 1 %QB0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	SUBB
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	SUBB	%IB0 1 %QB0										

### SUBD (双字型减法)

#### 功能说明

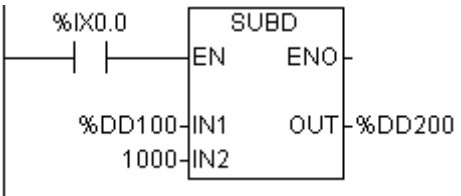
将 IN1 所指定的数据和 IN2 所指定的数据进行有符号数的减法运算，将结果输出到 OUT，不带借位。

指令及其操作数说明：

	名称	指令格式
LD	SUBD	
IL	SUBD	SUBD IN1,IN2,OUT

参数	输入/输出	数据类型	允许的内存区
IN1	输入	DINT	I、Q、W、常量
IN2	输入	DINT	I、Q、W、P、常量
OUT	输出	DINT	I、Q、W

指令使用举例：

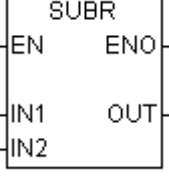
LD		<p>若 IX0.0 为 0：不执行 SUBD 指令。 若 IX0.0 为 1：DD200 = DD100 - 1000。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>SUBD</td> <td>%DD100 1000 %DD200</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	SUBD
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	SUBD	%DD100 1000 %DD200										

### SUBR（实型减法）

功能说明

将 IN1 所指定的数据和 IN2 所指定的数据进行有符号数的减法运算，将结果输出到 OUT，不带借位。

指令及其操作数说明：

	名称	指令格式
LD	SUBR	
IL	SUBR	SUBR IN1,IN2,OUT

参数	输入/输出	数据类型	允许的内存区
IN1	输入	REAL	I、Q、W、常量
IN2	输入	REAL	I、Q、W、P、常量
OUT	输出	REAL	I、Q、W

指令使用举例：

LD		<p>若 IX0.0 为 0：不执行 SUBR 指令。 若 IX0.0 为 1：DD200 = DD100 - 10.25。</p>											
IL	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">条</th> <th style="width: 10%;">步</th> <th style="width: 20%;">指令</th> <th style="width: 60%;">操作数</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">LD</td> <td style="text-align: center;">%IX0.0</td> </tr> <tr> <td></td> <td style="text-align: center;">2</td> <td style="text-align: center;">SUBR</td> <td style="text-align: center;">%DD100 10.25 %DD200</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	SUBR
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	SUBR	%DD100 10.25 %DD200										

### MUL\_I (乘法)

功能说明

将 IN1 所指定的数据和 IN2 所指定的数据进行有符号的乘法运算，将结果输出到 OUT、OUT+1 两个字中。OUT 存放结果的低 16 位，OUT+1 存放结果的高 16 位。

指令及其操作数说明：

	名称	指令格式
LD	MUL_I	
IL	MUL_I	MUL_I IN1,IN2,OUT

参数	输入/输出	数据类型	允许的内存区
IN1	输入	INT	I、Q、W、D、P、常量
IN2	输入	INT	I、Q、W、D、P、常量
OUT	输出	INT	I、Q、W、D、P

指令使用举例：

LD		<p>若 IX0.0 为 0：不执行 MUL_I 指令。 若 IX0.0 为 1：将 DW0 与 DW2 相乘，并将结果赋给 DW4 和 DW5。</p>											
IL	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">条</th> <th style="width: 10%;">步</th> <th style="width: 20%;">指令</th> <th style="width: 60%;">操作数</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">LD</td> <td style="text-align: center;">%IX0.0</td> </tr> <tr> <td></td> <td style="text-align: center;">2</td> <td style="text-align: center;">MUL_I</td> <td style="text-align: center;">%DW0 %DW2 %DW4</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	MUL_I
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	MUL_I	%DW0 %DW2 %DW4										

## MULB (字节型乘法)

### 功能说明

将 IN1 所指定的数据和 IN2 所指定的数据进行有符号的乘法运算，将结果输出到 OUT+1、OUT 两个字节中。OUT 存放结果的低 8 位，OUT+1 存放结果的高 8 位。

指令及其操作数说明：

	名称	指令格式
LD	MULB	
IL	MULB	MULB IN1,IN2,OUT

参数	输入/输出	数据类型	允许的内存区
IN1	输入	INT	I、Q、W、常量
IN2	输入	INT	I、Q、W、常量
OUT	输出	INT	I、Q、W

指令使用举例：

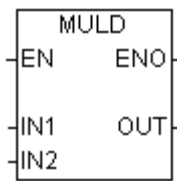
LD		<p>若 IX0.0 为 0：不执行 MULB 指令。 若 IX0.0 为 1：将 IB0 与 15 相乘，并将结果赋给 QB0。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>MULB</td> <td>%IB0 15 %QB0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	MULB
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	MULB	%IB0 15 %QB0										

## MULD (双字型乘法)

### 功能说明

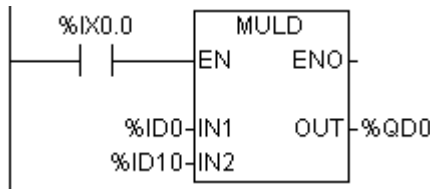
将 IN1 所指定的数据和 IN2 所指定的数据进行有符号的乘法运算，将结果输出到 OUT。

指令及其操作数说明：

	名称	指令格式
LD	MULD	
IL	MULD	MULD IN1,IN2,OUT

参数	输入/输出	数据类型	允许的内存区
IN1	输入	DWORD	I、Q、W、D、常量
IN2	输入	DWORD	I、Q、W、D、常量
OUT	输出	DWORD	I、Q、W、D

指令使用举例：

LD		<p>若 IX0.0 为 0：不执行 MULD 指令。 若 IX0.0 为 1：将 ID0 与 ID10 相乘，并将结果赋给 QD0。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>MULD</td> <td>%ID0 %ID10 %QD0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	MULD
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	MULD	%ID0 %ID10 %QD0										

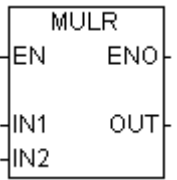
**注意：**结果有可能溢出，超出 32 位的部分将被舍弃。

### MULR（实型乘法）

功能说明

将 IN1 所指定的数据和 IN2 所指定的数据进行有符号的乘法运算，将结果输出到 OUT。

指令及其操作数说明：

	名称	指令格式
LD	MULR	
IL	MULR	MULR IN1,IN2,OUT

参数	输入/输出	数据类型	允许的内存区
IN1	输入	DWORD	I、Q、W、D、常量
IN2	输入	DWORD	I、Q、W、D、常量
OUT	输出	DWORD	I、Q、W、D



指令使用举例：

LD		<p>若 IX0.0 为 0：不执行 MULR 指令。 若 IX0.0 为 1：将 ID0 与 ID10 相乘，并将结果赋给 QD0。</p>											
IL	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">条</th> <th style="width: 10%;">步</th> <th style="width: 30%;">指令</th> <th style="width: 50%;">操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>MULR</td> <td>%ID0 %ID10 %QD0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	MULR
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	MULR	%ID0 %ID10 %QD0										

**注意：** 由于浮点数的精度是 6 位，所以结果超出精度的范围将会被舍弃。

### DIV\_I（除法）

功能说明

对 IN1 和 IN2 所指定的数据进行带符号除法运算，将商输出到 OUT，余数输出到 OUT+1。

指令及其操作数说明：

	名称	指令格式
LD	DIV_I	
IL	DIV_I	DIV_I IN1,IN2,OUT

参数	输入/输出	数据类型	允许的内存区
IN1	输入	INT	I、Q、W、D、P、常量
IN2	输入	INT	I、Q、W、D、P、常量
OUT	输出	INT	I、Q、W、D、P

指令使用举例：

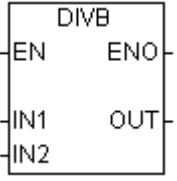
LD		<p>若 IX0.0 为 0：不执行 DIV_I 指令。 若 IX0.0 为 1：将 DW0 除以 DW2，并将结果赋给 DW4。</p>											
IL	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">条</th> <th style="width: 10%;">步</th> <th style="width: 30%;">指令</th> <th style="width: 50%;">操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>DIV_I</td> <td>%DW0 %DW2 %DW4</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	DIV_I
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	DIV_I	%DW0 %DW2 %DW4										

## DIVB（字节型除法）

功能说明

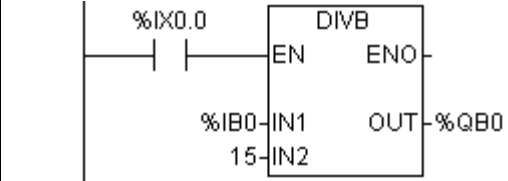
对 IN1 和 IN2 所指定的数据进行带符号除法运算，将商输出到 OUT，余数输出到 OUT+1。

指令及其操作数说明：

	名称	指令格式
LD	DIVB	
IL	DIVB	DIVB IN1,IN2,OUT

参数	输入/输出	数据类型	允许的内存区
IN1	输入	BYTE	I、Q、W、常量
IN2	输入	BYTE	I、Q、W、常量
OUT	输出	BYTE	I、Q、W

指令使用举例：

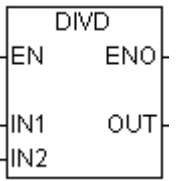
LD		<p>若 IX0.0 为 0：不执行 DIVB 指令。 若 IX0.0 为 1：将 IB0 除以 15，并将结果赋给 QB0。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>DIVB</td> <td>%IB0 15 %QB0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	DIVB
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	DIVB	%IB0 15 %QB0										

## DIVD（双字型除法）

功能说明

对 IN1 和 IN2 所指定的数据进行带符号除法运算，将商输出到 OUT，余数被舍弃。

指令及其操作数说明：

	名称	指令格式
LD	DIVD	
IL	DIVD	DIVD IN1,IN2,OUT

参数	输入/输出	数据类型	允许的内存区
IN1	输入	DWORD	I、Q、W、D、常量
IN2	输入	DWORD	I、Q、W、D、常量
OUT	输出	DWORD	I、Q、W、D

指令使用举例：

LD					<p>若 IX0.0 为 0：不执行 DIVD 指令。 若 IX0.0 为 1：将 ID0 除以 ID10，并将结果赋给 QD0。</p>									
	IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>DIVD</td> <td>%ID0 %ID10 %QD0</td> </tr> </tbody> </table>	条	步		指令	操作数	1	1	LD	%IX0.0		2	DIVD
条	步	指令	操作数											
1	1	LD	%IX0.0											
	2	DIVD	%ID0 %ID10 %QD0											

### DIVR（浮点型除法）

功能说明

对 IN1 和 IN2 所指定的数据进行带符号除法运算，将商输出到 OUT。

指令及其操作数说明：

	名称	指令格式
LD	DIVR	
IL	DIVR	DIVF IN1,IN2,OUT

参数	输入/输出	数据类型	允许的内存区
IN1	输入	DWORD	I、Q、W、D、常量
IN2	输入	DWORD	I、Q、W、D、常量
OUT	输出	DWORD	I、Q、W、D

指令使用举例：

LD		<p>若 IX0.0 为 0: 不执行 DIVR 指令。 若 IX0.0 为 1: 将 ID0 除以 ID10, 并将结果赋给 QD0。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>DIVR</td> <td>%ID0 %ID10 %QD0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	DIVR
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	DIVR	%ID0 %ID10 %QD0										

**注意:** 由于浮点数的精度是 6 位, 所以结果超出精度的范围将会被舍弃。

### INC\_I (自增)

#### 功能说明

对 D 所指定的数据进行自加操作, 在输入条件为 ON 的过程中 (直至 OFF), 每 PLC 扫描周期加 1。

指令及其操作数说明:

	名称	指令格式
LD	INC_I	
IL	INC_I	INC_I D

参数	输入/输出	数据类型	允许的内存区
EN	输入	BOOL	能量流
D	输出	INT	I、Q、W、D、P

指令使用举例:

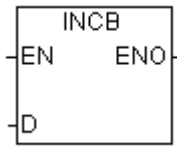
LD		<p>若 IX0.0 为 0: 不执行 INC_I 指令。 若 IX0.0 为 1: 将 DW100 加 1, 并将结果赋给 DW100。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>INC_I</td> <td>%DW100</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	INC_I
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	INC_I	%DW100										

### INCB (字节型自增)

#### 功能说明

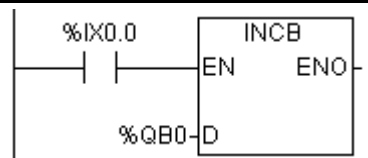
对 D 所指定的数据进行自加操作, 在输入条件为 ON 的过程中 (直至 OFF), 每 PLC 扫描周期加 1。

指令及其操作数说明：

	名称	指令格式
LD	INCB	
IL	INCB	INCB D

参数	输入/输出	数据类型	允许的内存区
EN	输入	BOOL	能量流
D	输出	BYTE	I、Q、W

指令使用举例：

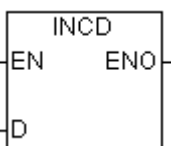
LD		若 IX0.0 为 0：不执行 INCB 指令。 若 IX0.0 为 1：将 QB0 加 1，并将结果赋给 QB0。											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>INCB</td> <td>%QB0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	INCB
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	INCB	%QB0										

### INCD（双字型自增）

功能说明

对 D 所指定的数据进行自加操作，在输入条件为 ON 的过程中（直至 OFF），每 PLC 扫描周期加 1。

指令及其操作数说明：

	名称	指令格式
LD	INCD	
IL	INCD	INCD D

参数	输入/输出	数据类型	允许的内存区
EN	输入	BOOL	能量流
D	输出	DINT	I、Q、W、D

指令使用举例：

LD					若 IX0.0 为 0: 不执行 INCD 指令。 若 IX0.0 为 1: 将 DD100 加 1, 并将结果赋给 DD100。										
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>INCD</td> <td>%DD100</td> </tr> </tbody> </table>	条	步	指令		操作数	1	1	LD	%IX0.0		2	INCD	%DD100	
条	步	指令	操作数												
1	1	LD	%IX0.0												
	2	INCD	%DD100												

### DEC\_I (自减)

#### 功能说明

对 D 所指定的数据进行自减操作, 在输入条件为 ON 的过程中 (直至 OFF), 每周期减 1。

指令及其操作数说明:

	名称	指令格式
LD	DEC_I	
IL	DEC_I	DEC_I D

参数	输入/输出	数据类型	允许的内存区
EN	输入	BOOL	能量流
D	输出	INT	I、Q、W、D、P

指令使用举例:

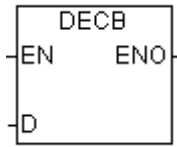
LD					若 IX0.0 为 0: 不执行 DEC_I 指令。 若 IX0.0 为 1: 将 DW100 减 1, 并将结果赋给 DW100。										
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>DEC_I</td> <td>%DW100</td> </tr> </tbody> </table>	条	步	指令		操作数	1	1	LD	%IX0.0		2	DEC_I	%DW100	
条	步	指令	操作数												
1	1	LD	%IX0.0												
	2	DEC_I	%DW100												

### DECB (字节型自减)

#### 功能说明

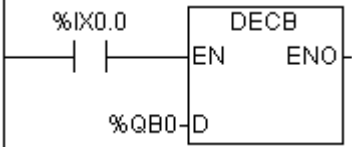
对 D 所指定的数据进行自减操作, 在输入条件为 ON 的过程中 (直至 OFF), 每周期减 1。

指令及其操作数说明:

	名称	指令格式
LD	DECB	
IL	DECB	DEC D

参数	输入/输出	数据类型	允许的内存区
EN	输入	BOOL	能量流
D	输出	BYTE	I、Q、W

指令使用举例：

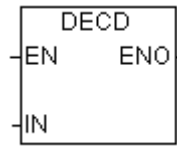
LD		<p>若 IX0.0 为 0：不执行 DECB 指令。 若 IX0.0 为 1：将 QB0 减 1，并将结果赋给 QB0。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>DECB</td> <td>%QB0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	DECB
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	DECB	%QB0										

### DECD（双字型自减）

功能说明

对 D 所指定的数据进行自减操作，在输入条件为 ON 的过程中（直至 OFF），每周期减 1。

指令及其操作数说明：

	名称	指令格式
LD	DECD	
IL	DECD	DECD D

参数	输入/输出	数据类型	允许的内存区
EN	输入	BOOL	能量流
D	输出	DINT	I、Q、W、D

指令使用举例：

LD		<p>若 IX0.0 为 0: 不执行 DECD 指令。 若 IX0.0 为 1: 将 DD100 减 1, 并将结果赋给 DD100。</p>											
IL	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">条</th> <th style="width: 10%;">步</th> <th style="width: 20%;">指令</th> <th style="width: 60%;">操作数</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">LD</td> <td style="text-align: center;">%IX0.0</td> </tr> <tr> <td></td> <td style="text-align: center;">2</td> <td style="text-align: center;">DECD</td> <td style="text-align: center;">%DD100</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	DECD
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	DECD	%DD100										

### SIN (正弦函数)

功能说明

将 IN 所指定的数据正弦运算,  $OUT = SIN(IN)$ , 结果存在 OUT 开始的 32 位通道中, IN 的值为角度。

指令及其操作数说明:

	名称	指令格式
LD	SIN	
IL	SIN	SIN IN, OUT

参数	输入/输出	数据类型	允许的内存区
IN	输入	REAL	I、Q、W、D、常量
OUT	输出	REAL	I、Q、W、D

指令使用举例:

LD		<p>若 IX0.0 为 0: 不执行 SIN 指令。 若 IX0.0 为 1: 将 ID0 取正弦值, 并将结果赋给 QD0。</p>											
IL	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">条</th> <th style="width: 10%;">步</th> <th style="width: 20%;">指令</th> <th style="width: 60%;">操作数</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">LD</td> <td style="text-align: center;">%IX0.0</td> </tr> <tr> <td></td> <td style="text-align: center;">2</td> <td style="text-align: center;">SIN</td> <td style="text-align: center;">%IDO %QD0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	SIN
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	SIN	%IDO %QD0										

### COS (余弦函数)

功能说明

将 IN 所指定的数据余弦运算,  $OUT = COS(IN)$ , 结果存在 OUT 开始的 32 位通道中, IN 的值为角度。

指令及其操作数说明:



	名称	指令格式
LD	COS	
IL	COS	COS IN, OUT

参数	输入/输出	数据类型	允许的内存区
IN	输入	REAL	I、Q、W、D、常量
OUT	输出	REAL	I、Q、W、D

指令使用举例：

LD		<p>若 IX0.0 为 0：不执行 COS 指令。 若 IX0.0 为 1：将 ID0 取余弦值，并将结果赋给 QD0。</p>											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>COS</td> <td>%IDO %QD0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	COS
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	COS	%IDO %QD0										

### TAN（正切函数）

功能说明

将 IN 所指定的数据正切运算， $OUT = \tan(IN)$ ，结果存在 OUT 开始的 32 位通道中。  
IN 的值为角度。

指令及其操作数说明：

	名称	指令格式
LD	TAN	
IL	TAN	TAN IN, OUT

参数	输入/输出	数据类型	允许的内存区
IN	输入	REAL	I、Q、W、D、常量
OUT	输出	REAL	I、Q、W、D

指令使用举例：

LD					若 IX0.0 为 0: 不执行 TAN 指令。 若 IX0.0 为 1: 将 ID0 取正切值, 并将结果赋给 QD0。										
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>TAN</td> <td>%IDO %QD0</td> </tr> </tbody> </table>	条	步	指令		操作数	1	1	LD	%IX0.0		2	TAN	%IDO %QD0	
条	步	指令	操作数												
1	1	LD	%IX0.0												
	2	TAN	%IDO %QD0												

### LN (工业对数函数)

功能说明

将 IN 所指定的数据工业对数运算, 将结果输出到 OUT。

指令及其操作数说明:

	名称	指令格式
LD	LN	
IL	LN	LN IN, OUT

参数	输入/输出	数据类型	允许的内存区
IN	输入	REAL	I、Q、W、D、常量
OUT	输出	REAL	I、Q、W、D

指令使用举例:

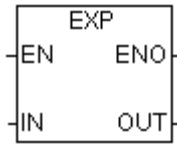
LD					若 IX0.0 为 0: 不执行 LN 指令。 若 IX0.0 为 1: 将 ID0 取对数值, 并将结果赋给 QD0。										
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>LN</td> <td>%IDO %QD0</td> </tr> </tbody> </table>	条	步	指令		操作数	1	1	LD	%IX0.0		2	LN	%IDO %QD0	
条	步	指令	操作数												
1	1	LD	%IX0.0												
	2	LN	%IDO %QD0												

### EXP (幂指数函数)

功能说明

将 IN 所指定的数据幂指数运算, 将结果输出到 OUT。

指令及其操作数说明:

	名称	指令格式
LD	EXP	
IL	EXP	EXP IN, OUT

参数	输入/输出	数据类型	允许的内存区
IN	输入	REAL	I、Q、W、D、常量
OUT	输出	REAL	I、Q、W、D

指令使用举例：

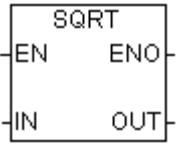
LD	LD %IX0.0 EXP %ID0, %QD0	<p>若 IX0.0 为 0：不执行 EXP 指令。 若 IX0.0 为 1：将 ID0 取幂指数，并将结果赋给 QD0。</p>
IL		

### SQRT（开平方函数）

功能说明

将 IN 所指定的数据开平方运算，将结果输出到 OUT，IN 的值为角度。

指令及其操作数说明：

	名称	指令格式
LD	SQRT	
IL	SQRT	SQRT IN, OUT

参数	输入/输出	数据类型	允许的内存区
IN	输入	REAL	I、Q、W、D、常量
OUT	输出	REAL	I、Q、W、D

指令使用举例：

LD		<p>若 IX0.0 为 0: 不执行 SQRT 指令。 若 IX0.0 为 1: 将 ID0 取平方根, 并将结果赋给 QD0。</p>												
IL	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">条</th> <th style="width: 10%;">步</th> <th style="width: 20%;">指令</th> <th style="width: 60%;">操作数</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">LD</td> <td style="text-align: center;">%IX0.0</td> </tr> <tr> <td></td> <td style="text-align: center;">2</td> <td style="text-align: center;">SQRT</td> <td style="text-align: center;">%ID0 %QD0</td> </tr> </tbody> </table>	条	步	指令	操作数	1	1	LD	%IX0.0		2	SQRT	%ID0 %QD0	
条	步	指令	操作数											
1	1	LD	%IX0.0											
	2	SQRT	%ID0 %QD0											

### 3.9 程序控制

子程序的建立

1、工程中在打开时默认有一个子程序, 如果想要添加更多的子程序可以右键单击程序管理器中的“程序块”项, 在弹出菜单中选择“插入子程序”, 如图 9.1 所示:



图 9.1

2、子程序在工程管理器的指令树最下方, 如图 9.2 所示:



图 9.2

3、在程序管理器中双击“程序块/子程序 0”项可以编辑子程序 0, 如图 9.3 所示:

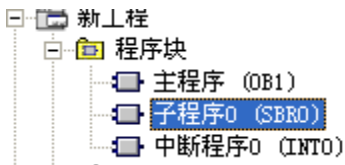


图 9.3

4、在主程序中选中要调用子程序的位置, 然后双击指令树窗口的子程序图标即可将子程序添加到主程序中进行调用。

5、子程序的调用, 翻译成 IL 语言将显示为 CAL 指令, 如图 9.4 所示:

条	步	指令	操作数
1	1	LD	P_ON
	2	SBR0	

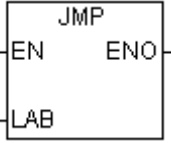
图 9.4

### JMP (条件跳转)

功能说明

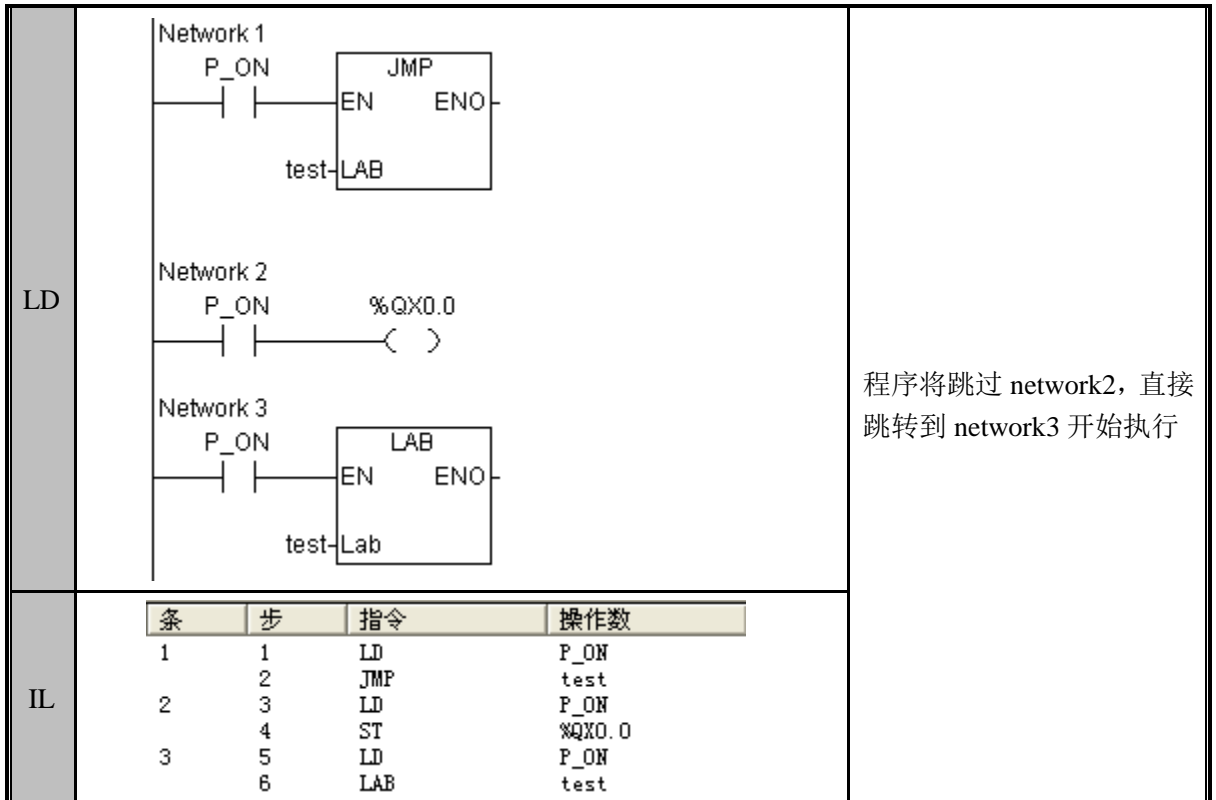
使程序跳转到 Lab 指定的标号处继续执行。

指令及其操作数说明：

	名称	指令格式
LD	JMP	
IL	JMP	JMP LAB

参数	输入/输出	数据类型	允许的内存区
EN	输入	BOOL	能量流
Lab	输入	字符串	字符串常量

指令使用举例：

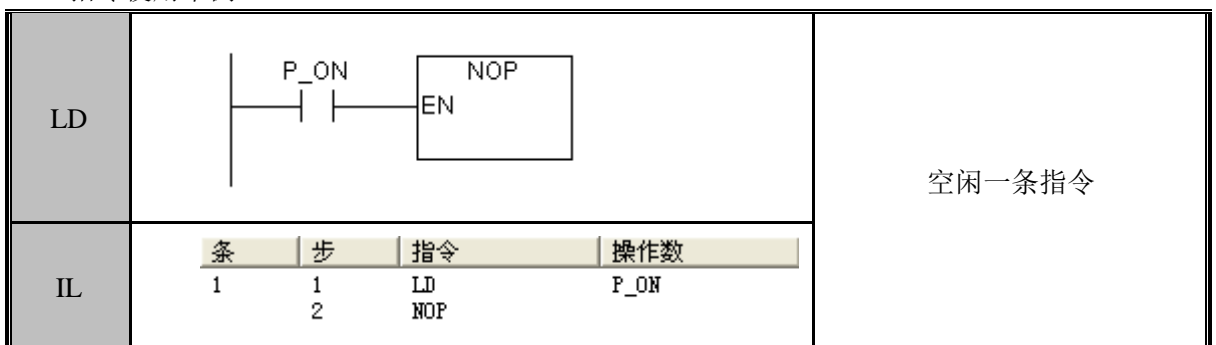


### NOP (空操作)

功能说明  
 空闲一条指令  
 指令及其操作数说明:

	名称	指令格式
LD	NOP	
IL	NOP	NOP

指令使用举例:

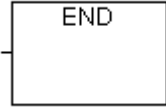


## END (程序结束)

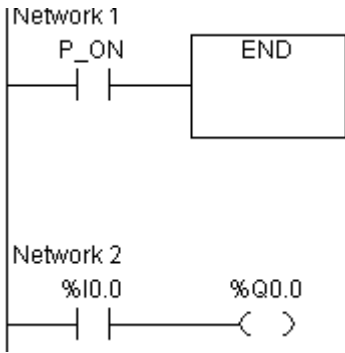
### 功能说明

结束当前程序，用于分段调试时不执行部分程序

指令及其操作数说明：

	名称	指令格式
LD	END	
IL	END	END

指令使用举例：

LD		<p>终止当前程序，END 以下的程序不会被执行。</p>																			
IL	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>END</td> <td></td> </tr> <tr> <td>2</td> <td>3</td> <td>LD</td> <td>%I0.0</td> </tr> <tr> <td></td> <td>4</td> <td>ST</td> <td>%Q0.0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	END		2	3	LD	%I0.0		4	ST
条	步	指令	操作数																		
1	1	LD	P_ON																		
	2	END																			
2	3	LD	%I0.0																		
	4	ST	%Q0.0																		


**注意：**END 指令前面的触点不影响 END 指令的执行，如上图所示，即使 END 前面的触点的值为 0，END 指令还是会执行，END 指令之下的程序不会执行。

## STOP (终止运行)

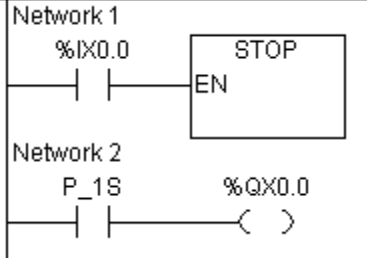
### 功能说明

终止 PLC 程序的运行，将 PLC 转换到编程模式，即当指令左侧能量流的值为 1，则终止 PLC 程序的运行，将 PLC 转换到编程模式，否则该指令不起作用，程序继续向下依次执行。

指令及其操作数说明：

	名称	指令格式
LD	STOP	
IL	STOP	STOP

指令使用举例：


LD		<p>在子程序 SBR_0 中： 若 IX0.0 为 0，则继续依次执行下面的指令。 若 IX0.0 为 1，则终止 PLC 程序的运行，将 PLC 转换到编程模式。</p>																			
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>STOP</td> <td></td> </tr> <tr> <td>2</td> <td>3</td> <td>LD</td> <td>P_1S</td> </tr> <tr> <td></td> <td>4</td> <td>ST</td> <td>%QX0.0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	STOP		2	3	LD	P_1S		4	ST
条	步	指令	操作数																		
1	1	LD	%IX0.0																		
	2	STOP																			
2	3	LD	P_1S																		
	4	ST	%QX0.0																		

### CRET（子程序有条件返回）

#### 功能说明

从子程序或中断服务程序中有条件返回，即：若当前值为 1，则中断当前程序并返回到该程序的调用点继续执行，否则该指令不起作用，程序继续向下依次执行。

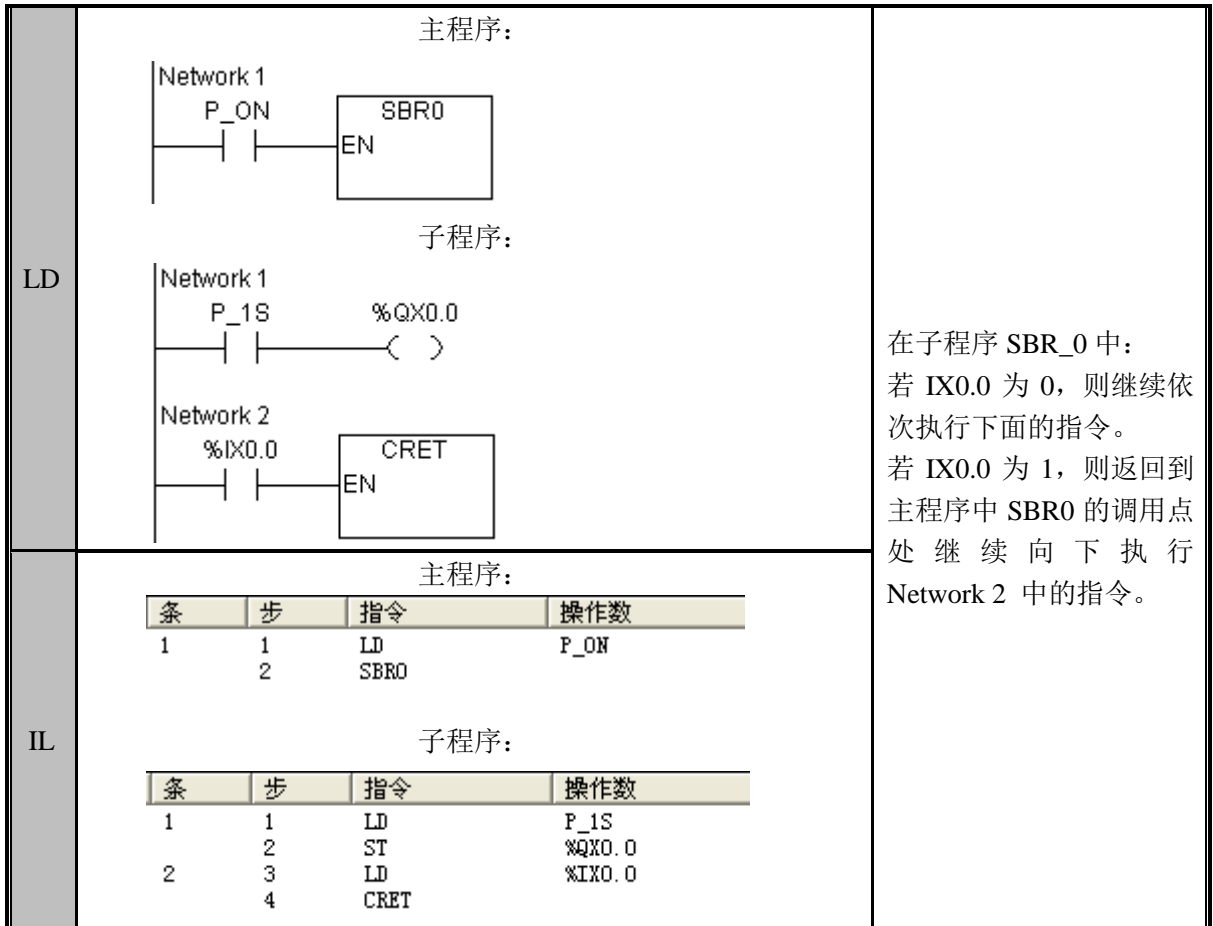
指令及其操作数说明：

	名称	指令格式
LD	CRET	
IL	CRET	CRET

**注意：**返回指令只能在子程序和中断服务程序中使用。

指令使用举例：



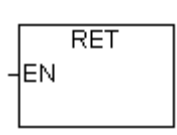


### RET（子程序无条件返回）

#### 功能说明

从子程序或中断服务程序中无条件返回，即：中断当前程序并返回到该程序的调用点继续执行。

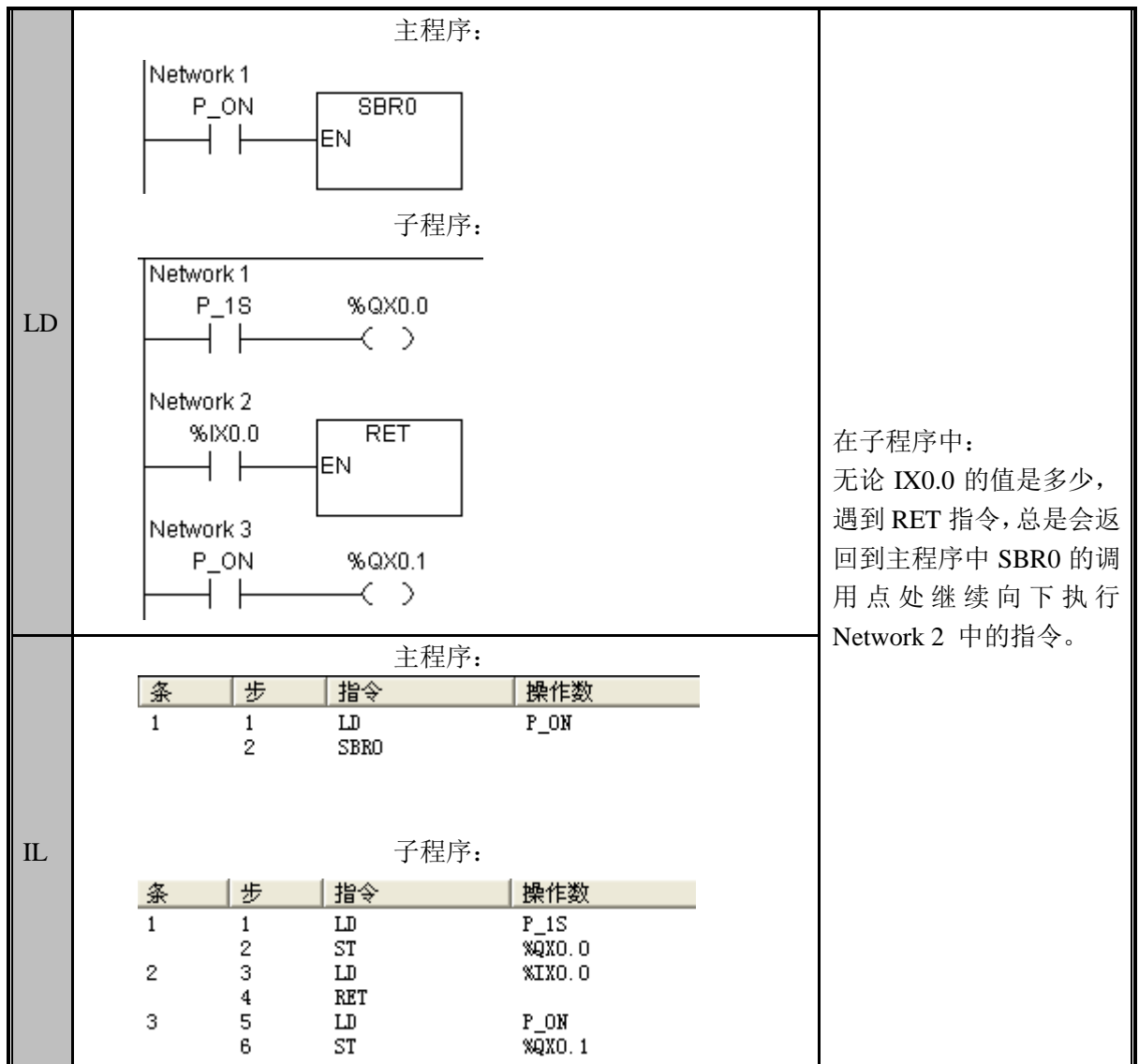
#### 指令及其操作数说明：

	名称	指令格式
LD	RET	
IL	RET	RET

参数	输入/输出	数据类型	允许的内存区
EN	输入	BOOL	能量流

**注意：**返回指令只能在子程序和中断服务程序中使用。


指令使用举例：



**注意：**RET 指令的执行不受前面输入条件的影响，例如，上图所示程序，即使 RET 之前的输入状态为 0，RET 指令依然会被执行。

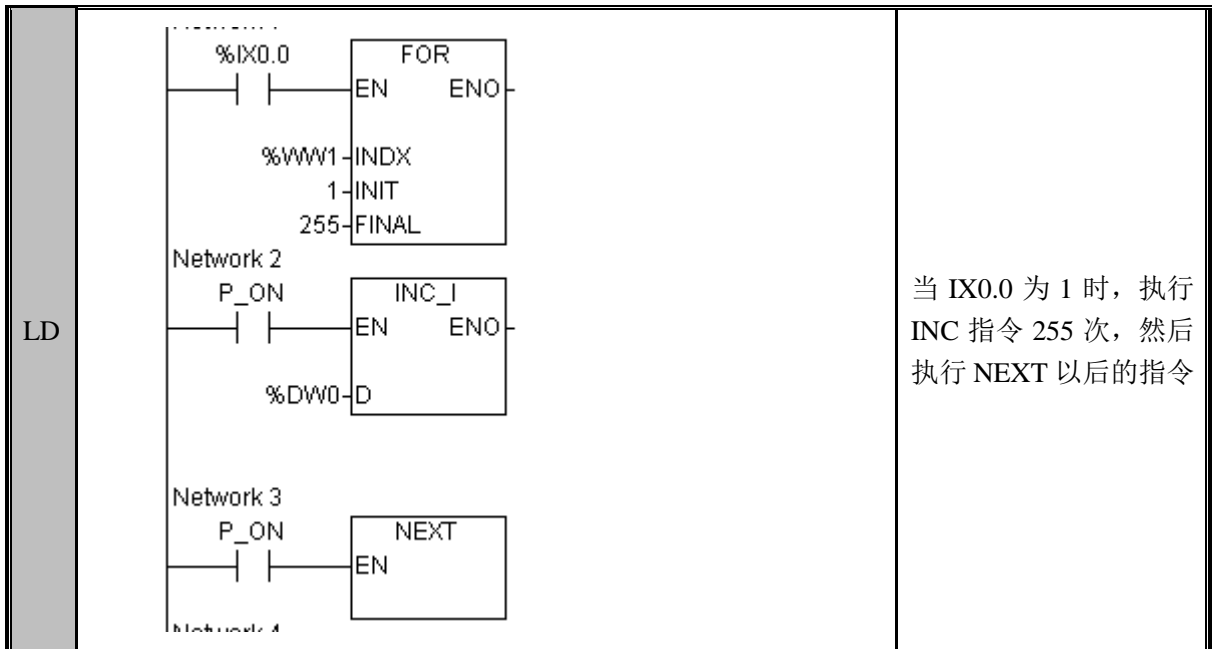
### FOR/NEXT (FOR/NEXT 循环)

功能说明  
执行指定计数的循环。  
指令及其操作数说明：

	名称	指令格式
LD	FOR NEXT	
IL	FOR NEXT	FOR NEXT

参数	输入/输出	数据类型	允许的内存区
INDX	输入	WORD	I、Q、W、D、P
INIT	输入	WORD	I、Q、W、D、P、常量
FINAL	输入	WORD	I、Q、W、D、P、常量

FOR/NEXT 指令的作用是：通过 FOR 指定计数的循环。执行完毕后执行 NEXT 指令。  
指令使用举例：



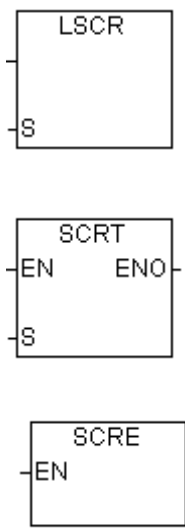
IL	1	1	LD	%IX0.0
		2	FOR	%W1 1 255
	2	3	LD	P_ON
		4	INC_I	%DWO
	3	5	LD	P_ON
		6	NEXT	

### SCR (工序控制)

功能说明

完成一系列重复的操作。

指令及其操作数说明:

	名称	指令格式
LD	LSCR SCRT SCRE	
IL	LSCR SCRT SCRE	LSCR S SCRT S SCRE

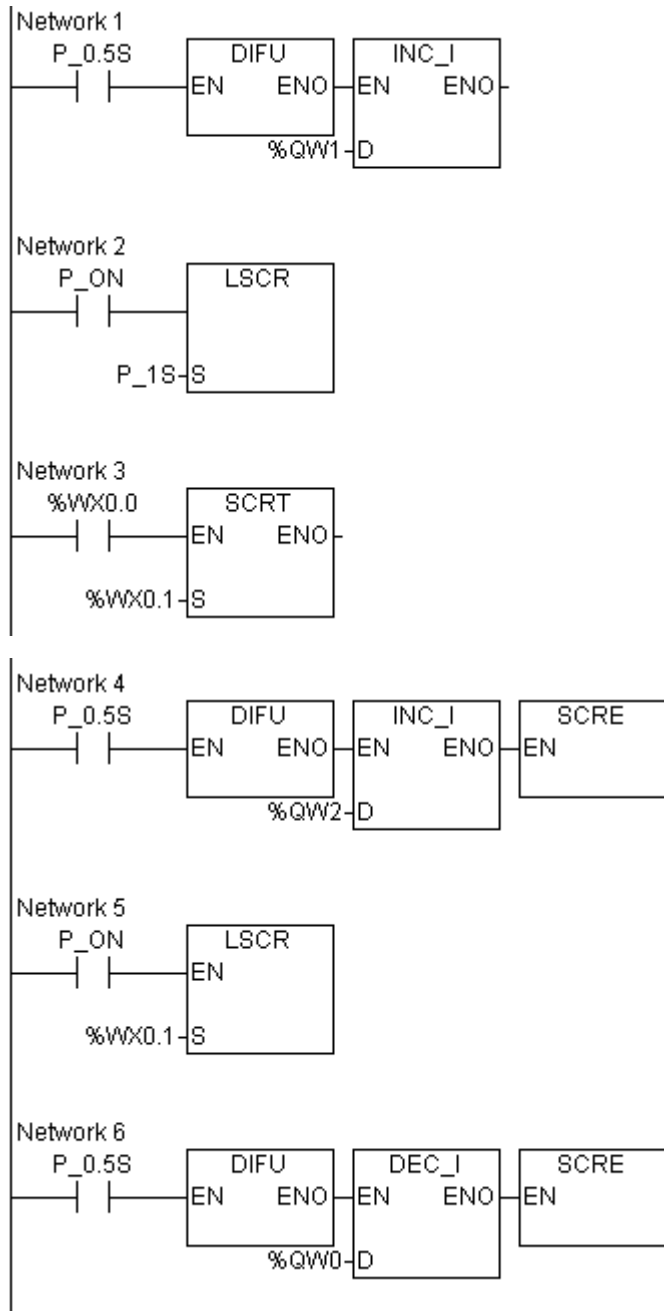
参数	输入/输出	数据类型	允许的内存区
S	输入	BIT	I、Q、W

LSCR 装载一个位，当这个位置 1 就开始当前工序

SCRT, 设置一个位，当这个位置一，结束当前工序，转移到下一个工序

SCRE, 结束当前工序

指令使用举例:



当 WX0.0 置位时，启动  
工序 1，执行自增操作，  
SCRT 激活 WX0.1,转到  
工序 2，执行自减操作

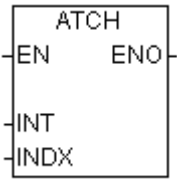
IL

条	步	指令	操作数
1	1	LD	P_0.5S
	2	DIFU	
	3	INC_I	%QW1
2	4	LD	P_ON
	5	LSCR	P_1S
3	6	LD	%WX0.0
	7	SCRT	%WX0.1
4	8	LD	P_0.5S
	9	DIFU	
	10	INC_I	%QW2
	11	SCRE	
5	12	LD	P_ON
	13	LSCR	%WX0.1
6	14	LD	P_0.5S
	15	DIFU	
	16	DEC_I	%QW0
	17	SCRE	

### 3.10 中断指令

#### ATCH (中断关联)

功能说明  
将中断事件和中断例程关联起来。  
指令及其操作数说明：

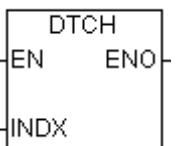
	名称	指令格式
LD	ATCH	
IL	ATCH	ATCH INT, INDX

参数	输入/输出	数据类型	允许的内存区
INT	输入	STRING	-
INDX	输入	常数	-

INT, 为中断子程序的标号, 字符串类型  
INDX, 为中断编号, 常量, 具体中断编号参见附录 F。

#### DTCH (中断关联解除)

功能说明  
将中断事件和中断例程解除关联。  
指令及其操作数说明：

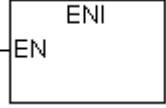
	名称	指令格式
LD	DTCH	
IL	DTCH	DTCH INDX

参数	输入/输出	数据类型	允许的内存区
INDX	输出	常数	-

INDX, 为中断编号, 常量, 具体中断编号参见附录 F。

### ENI（全局中断使能）

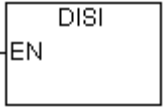
功能说明  
使能全局中断。  
指令及其操作数说明：

	名称	指令格式
LD	ENI	
IL	ENI	ENI

无操作数

### DISI（全局中断禁能）

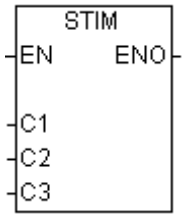
功能说明  
禁能全局中断。  
指令及其操作数说明：

	名称	指令格式
LD	DISI	
IL	DISI	DISI

无操作数

### STIM（时间间隔定时器）

功能说明  
用于控制时间间隔定时器完成四个基本功能：启动一个单次中断定时器；启动一个预定时间间隔定时器；读定时器的 PV 和停止定时器。设定 C1 的值来指定将要被执行的功能。  
指令及其操作数说明：

	名称	指令格式
LD	STIM	
IL	STIM	STIM C1, C2, C3

参数	输入/输出	数据类型	允许的内存区
C1	输入	常数	-
C2	输入	WORD	I、Q、W、D、P、常量
C3	输入	保留	默认为0

**注意：**C1 的值只能取 1、2、4、8。

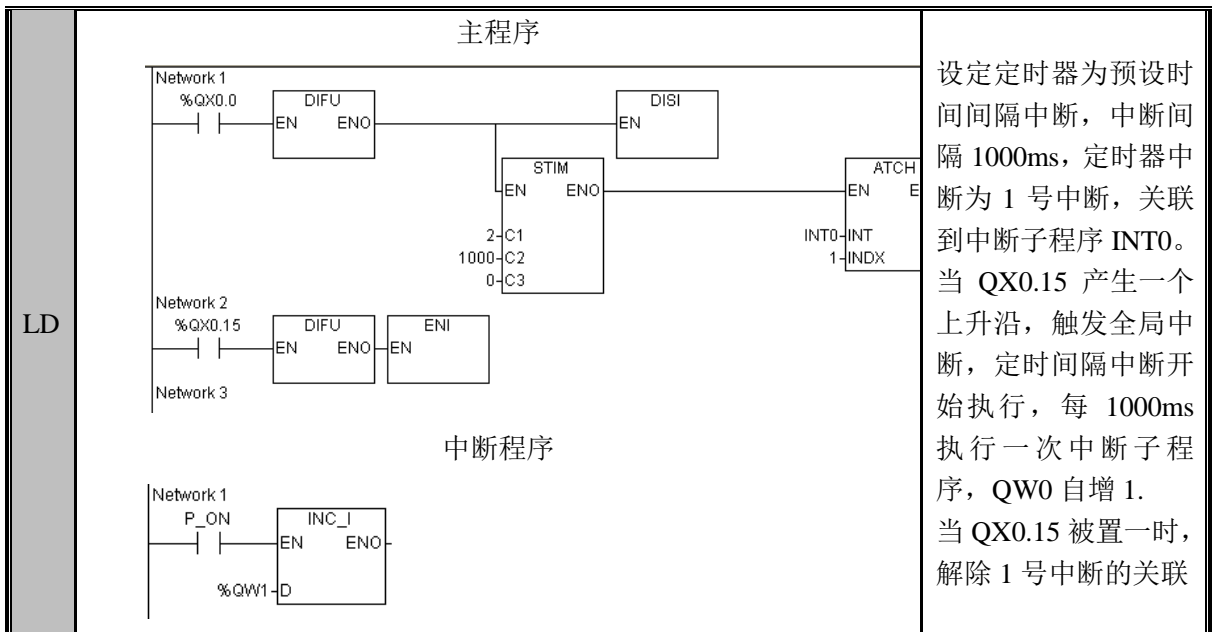
功能如下

C1 的值	功能
1	启动单次中断定时器
2	启动预设时间间隔中断定时器
4	读定时器当前值
8	停止定时器

当 C1 为 1 或 2 时，C2 指定中断间隔时间，单位为毫秒，当 C1 为 4 时，将定时器当前值存入 C2 指定的位置。

C3 保留，默认为 0。

中断指令使用举例：





		主程序			
条	步	指令	操作数		
IL	1	LD	%QX0.0		
		DIFU			
		DISI			
		STIM	2		
			1000		
			0		
		5	ATCH	INT0	
				1	
	2	LD	%QX0.15		
		DIFU			
		ENI			
		中断程序			
条	步	指令	操作数		
	1	LD	P_ON		
	2	INC_I	%QW1		

### 3.11 通讯指令

TXD（串口输出）、RXD（串口输入）

功能说明

用于自由口通信。在自由通讯方式下，只是按要求执行发送和接收，而不用考虑具体的数据意义，具体的通信协议由用户自己定义。用户可以使用自由通讯方式来编写各种自定义的通讯协议与第三方的通讯设备进行通讯。

指令及其操作数说明：

	名称	指令格式
LD	TXD	
	RXD	
IL	TXD	TXD S,C,N
	RXD	RXD S,C,N

参数	输入/输出	数据类型	允许的内存区
S	输入	WORD	I、Q、W、D、
C	输入	WORD	I、Q、W、D、常量
N	输入	WORD	常量

TXD 指令用于发送存放在数据缓冲区中的数据；参数 C 定义了所用的通讯口；参数 S 定义了数据缓冲区的起始地址，依次存放着待发送的数据字节；参数 N 由用户指定本次将要发送的字节数（1 到 255）。

RXD 指令用于接收数据并将接收到的数据存放在数据缓冲区中；参数 C 定义了所用的通讯口；参数 S 定义了数据缓冲区的起始地址，依次存放着接收到的有效数据字节；参数 N 由用户指定本次接收到的字节数。用户在调用 RXD 指令前，必须指定有效数据接收的起始条件和结束条件（见下面控制字的定义）。

**注意：**N 默认为 0，表示按缓冲区中实际接收到的有效字节数进行收取；

若 N 大于缓冲区中实际接收到的字节数，则按实际数目收取；

若 N 小于或等于缓冲区中实际接收到的字节数，则按 N 值进行收取；

若当前值为 1 则执行 TXD、RXD 指令，否则不执行。

执行条件

当通信自由口发送允许标志位置位时，可以执行发送指令。

对于串口 0 和串口 1，该标志位分别为 %SX41 和 %SX46。

执行接收指令需要先配置与接收相关的寄存器，指定消息的开始和结束条件。通讯部分依据指定的条件去分析处理接收。若消息结束，系统置接收完成标志。用户执行 RXD 指令前需先判断接收完成标志，接收完成标志置位才可正确地执行 RXD 指令。

对于串口 0 和串口 1，该标志位分别为 %SX40 和 %SX45。

缓冲区中接收到的消息应及时取走，否则可能被后面的数据冲掉。若接收检测到一个新的消息起始，则清零接收完成标志，开始新的接收过程。

自由通讯方式的状态字节及控制字节

除了 TXD、RXD 指令外，在 CPU 中同时还设置了多个标志位和控制字用于自由通讯。用户必须在程序中对这些控制字进行设置来控制通讯。另外，在通讯过程中 CPU 会自动对通讯状态进行检测，并将检测结果设置到指定的标志位，用户可以读取这些状态信息并在程序中进行相应的处理。下面将对这些标志位和控制字进行简要的汇总。

标志位	名称	描述
SX40	通信口 0 自由口接收完成	为 OFF 时表示没有接收到完整消息； 为 ON 时表示接收到完整消息
SX41	通信口 0 自由口发送允许	为 OFF 时表示不允许发送； 为 ON 时表示允许发送
SX45	通信口 1 自由口接收完成	为 OFF 时表示没有接收到完整消息； 为 ON 时表示接收到完整消息
SX46	通信口 1 自由口发送允许	为 OFF 时表示不允许发送； 为 ON 时表示允许发送

控制字		描述																																																
串口 0	串口 1																																																	
MW88	MW95	<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <tr> <td>MSB</td><td></td><td></td><td></td><td></td><td></td><td></td><td>LSB</td> </tr> <tr> <td>15</td><td></td><td></td><td></td><td></td><td></td><td></td><td>8</td> </tr> <tr> <td>x</td><td>x</td><td>x</td><td>x</td><td>d</td><td>c</td><td>b</td><td>a</td> </tr> </table> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <tr> <td>MSB</td><td></td><td></td><td></td><td></td><td></td><td></td><td>LSB</td> </tr> <tr> <td>7</td><td></td><td></td><td></td><td></td><td></td><td></td><td>0</td> </tr> <tr> <td>x</td><td>x</td><td>t</td><td>h</td><td>e</td><td>e</td><td>s</td><td>s</td> </tr> </table> <p>自由口主配置字。 其中： ss: 接收消息开始条件 00 = 起始字符 01 = 空闲行时间 10 = 任意字符 ee: 接收消息结束条件 00 = 结束字符 01 = 空闲行时间 10 = 接收字符数 11 = 消息定时器（此时不存在帧超时） h: 发送消息头设置 0 = 默认无变化 1 = 添加起始字符 t: 发送消息尾设置 0 = 默认无变化 1 = 添加结束字符 a: 起始字符设定方式 0 = 默认设定为 0x3A（ASCII 字符“:”） 1 = 由后面的寄存器设定 b: 结束字符设定方式 0 = 默认设定为 0x0D、0x0A（\CR\LF） 1 = 由后面的寄存器设定 c: 空闲行时间设定方式 0 = 当前波特率下 3.5 个字符时间 1 = 由后面的寄存器指定（要求不小于当前波特率下 16 个字符时间） d: 帧超时时间设定方式 0 = 不设帧超时 1 = 超时时间由后面的寄存器指定</p>	MSB							LSB	15							8	x	x	x	x	d	c	b	a	MSB							LSB	7							0	x	x	t	h	e	e	s	s
MSB							LSB																																											
15							8																																											
x	x	x	x	d	c	b	a																																											
MSB							LSB																																											
7							0																																											
x	x	t	h	e	e	s	s																																											

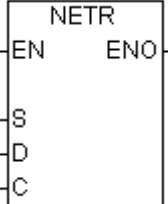
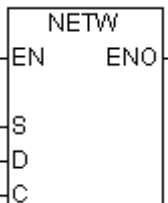
MW89	MW96	用于存放用户定义的接收起始字符。(00~FF Hex) 主配置字中接收消息开始条件设为起始字符时, 该配置字有效。此时, 当接收到该指定的字符时, 接收消息开始。
MW90	MW97	用于存放用户定义的接收结束字符。(00~FF Hex) 主配置字中接收消息结束条件设为结束字符时, 该配置字有效。此时, 当接收到该指定的字符时, 接收消息完成, 并将置位自由口接收完成标志位。
MW91	MW98	用于存放用户定义的接收字符数(0~256)。 00: 256 01-FF Hex: 1-255 高字节: 保留 主配置字中接收消息结束条件设为接收字符数时, 该配置字有效。此时, 若接收消息已开始, 当接收的字符达到该指定的个数时, 接收消息完成, 并将置位自由口接收完成标志位。
MW92	MW99	用于存放用户定义的空闲行时间(范围为1~1000, 单位ms)。 主配置字中接收消息开始或结束条件设为空闲行时间时, 该配置字有效。 若开始条件设为空闲行时间, 则当检测到线上有该指定时间的空闲时, 则接收消息开始。 若结束条件设为空闲行时间, 则当检测到线上有该指定时间的空闲时, 则接收消息完成, 并将置位自由口接收完成标志位。 注: 自定义空闲行时间最少为1ms, 并且应不小于当前波特率下16个字符时间
MW93	MW100	用于存放用户定义的消息定时器超时时间。 消息定时器超时时间最少为1ms, 典型数值约为在当前波特率下接收最长可能消息所需时间的1.5倍。 主配置字中接收消息结束条件设为消息定时器时, 该配置字有效。此时, 当接收消息开始后, 经过该时间后, 接收消息完成, 并将置位自由口接收完成标志位。
MW94	MW101	用于存放用户定义的接收帧超时值(范围为100~60000ms)。 进入有效接收状态后, 若在此超时时间内仍未收到结束条件, 系统就会结束接收状态, 并置接收超时错误。

## NETR (网络读)、NETW (网络写)

### 功能说明

这两条为网络指令, 用于PC-Net自组网中的通信。由PC-Net主站通过网络读(NETR)或网络写(NETW)的指令向网络中的从站发出数据访问请求, 等待从站的响应。这两条指令分别支持对网络中从设备特定区域的读取和写入。

指令及其操作数说明:

	名称	指令格式
LD	NETR	
	NETW	
IL	NETR	NETR S, D, C
	NETW	NETW S, D, C

参数	输入/输出	数据类型	允许的内存区
S	输入	WORD	I、Q、W、D、
D	输入	WORD	I、Q、W、D、
C	输入	WORD	I、Q、W、D、

NETR 指令用于从 PC-Net 网络中的其它 PLC 读取指定区域的指定数据，并保存到自身的指定位置。其操作数的意义如下：

S 表示源起始字，D 表示目标起始字，C 表示第一控制字。

指令控制字总共包含三个字的数据，定义如下：

字	位 00~07	位 08~15
C	串口号 (0 或 1)	从站设备地址 (1-31)
C+1	访问的数据字节数 (1-255)	
C+2	网络指令状态字	

NETW 指令用于将指定的数据写入到 PC-Net 网络中指定 PLC 的指定区域。其各操作数的意义如下：

S 表示源起始字，D 表示目标起始字，C 表示第一控制字。

指令控制字总共包含三个字的数据，定义如下：

字	位 00~07	位 08~15
C	串口号 (0 或 1)	从站设备地址 (1-31)
C+1	访问的数据字节数 (1-255)	
C+2	网络指令状态字	

网络指令状态字是网络指令控制字的一部分，用来记录网络指令的执行状况，便于用户对网络指令及其通讯过程进行掌控。其定义如下：

MSB						LSB							
5	4	3	2	1	0								
出错应答帧的错误代码 (Exception Code)						错误代码							

- D 完成 (请求已完成): 0 = 没有完成, 1 = 完成  
 A 激活 (请求已排队): 0 = 没有激活, 1 = 激活  
 E 出错 (指令执行出错): 0 = 没有出错, 1 = 出错

错误代码定义如下,

代码	定义
0	无错
1	串口工作方式错误: 不是网络通讯方式
2	不是网络通讯主站
3	网络指令请求队列溢出
4	参数错误: 非法的串口号或从设备地址
5	访问越界: 错误的地址或数据长度
6	超时出错: 远程站没有应答
7	出错应答帧: 错误代码见 Exception Code 部分
8-F	未用 (保留)

#### 执行条件

只有网络指令通信请求允许标志使能时, 才能执行网络指令。  
 串口 0 相关标志位为 %SX42; 串口 1 相关标志位为 %SX47。

#### MBAR, MBARX (添加非周期 Modbus 请求)

##### 功能说明

Modbus 主站指令, 用于 Modbus-RTU 主站的通信。添加非周期性的 Modbus-RTU 请求帧。

指令及其操作数说明:

	名称	指令格式
LD	MBAR	

IL	MBAR	MBAR N, F, D, T, S
----	------	--------------------

参数	输入/输出	数据类型	允许的内存区
N	输入	WORD	I、Q、W、D、常量
F	输入	WORD	I、Q、W、D
D	输入	WORD	I、Q、W、D、常量 0
T	输入	WORD	I、Q、W、D、常量
S	输出	WORD	I、Q、W、D、常量 0

	名称	指令格式
LD	MBARX	
IL	MBARX	MBARX N, F, D, T, S

参数	输入/输出	数据类型	允许的内存区
N	输入	WORD	I、Q、W、D、常量
F	输入	WORD	I、Q、W、D
D	输入	BIT	I、Q、W、D、常量 0
T	输入	WORD	I、Q、W、D、常量
S	输出	WORD	I、Q、W、D、常量 0

N: 串口号, 0 或 1,支持立即数和 W 型参数

F: 请求帧数据表头, 支持 W 型参数

D: 目标或源数据区, 当为读命令时, 表示读取数据的存放位置, 当为写命令时, 表示欲写入数据的地址; 支持 X(MBARX)、W(MBAR)型参数, 或立即数 0 (无需指定 data 区时)

T: 超时设置 (ms), 支持立即数和 W 型参数, 为 0 表示不修改之前的设定

S: 状态字, 支持 W 型参数, 或立即数 0 (不为该请求帧设状态字)

系统会根据请求帧的表头及后续数据, 自动分析需要计入的数据长度, 并在需要时添加由后面参数指定的数据域, 然后计算校验, 再组成发送帧。

若该指令的操作需要保存相关信息 (例如如果是读取数据时, 需指定保存数据的区域), 系统会将该信息保存到指定的区域。

用户可以设定从站响应的超时时间标准 (以 ms 为单位), 系统只有一个超时标准, 由最新的设定值决定。若该设定参数为 0, 表示不修改之前的设定。系统默认的超时标准为 1s。

主站操作状态字是用来记录 Modbus-RTU 主站发出命令的执行状况, 以便于用户对其通讯过程进行掌控。若用户不需为当前的通信请求设定状态字, 可以将该参数置 0。

状态字的定义如下:

MSB						LSB											
5	4	3	2	1	0												
异常应答帧的异常代码 (Exception Code)						错误代码											

D 完成 (请求已完成): 0 = 没有完成, 1 = 完成

A 激活 (请求已排队): 0 = 没有激活, 1 = 激活

E 出错 (指令执行出错): 0 = 没有出错, 1 = 出错

错误代码定义如下,

代码	定义
0	无错
1	串口工作方式错误: 不是 Modbus-RTU 主站方式
2	周期报文请求队列溢出
3	非周期报文请求队列溢出
4	参数错误: 非法的串口号或设定值
5	操作越界: 主站数据区越界
6	周期报文的轮询周期设定太短
7	超时出错: 从站没有应答
8	异常应答帧: 异常代码见 Exception Code 部分
9-F	未用 (保留)

Exception Code 定义遵循标准 Modbus 协议。

执行条件:

只有非周期报文请求允许使能时, 才能正确地执行非周期报文请求的指令。所以一般情况下, 用该标志位做添加非周期报文请求指令的执行条件。

串口 0 相关标志位为 %SX44; 串口 1 相关标志位为 %SX49。

若当前等待处理的队列已满, 系统会将该标志位清零。系统最大支持 8 个队列的非周期报文请求。当前队列中的所有请求报文被处理后, 系统自动将队列清零。

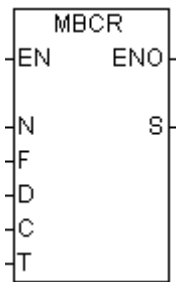
### MBCR, MBCRX (添加周期 Modbus 请求)

功能说明

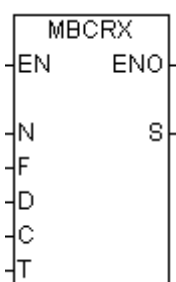
Modbus 主站指令, 用于 Modbus-RTU 主站的通信。添加周期性的 Modbus-RTU 请求帧。

指令及其操作数说明:



	名称	指令格式
LD	MBCR	
IL	MBCR	MBCR N, F, D, C, T, S

参数	输入/输出	数据类型	允许的内存区
N	输入	WORD	I、Q、W、D、常量
F	输入	WORD	I、Q、W、D
D	输入	WORD	I、Q、W、D、常量 0
C	输入	WORD	I、Q、W、D、常量
T	输入	WORD	I、Q、W、D、常量
S	输出	WORD	I、Q、W、D、常量 0

	名称	指令格式
LD	MBCRX	
IL	MBCRX	MBCRX N, F, D, C, T, S

参数	输入/输出	数据类型	允许的内存区
N	输入	WORD	I、Q、W、D、常量
F	输入	WORD	I、Q、W、D
D	输入	BIT	I、Q、W、D、常量 0
C	输入	WORD	I、Q、W、D、常量
T	输入	WORD	I、Q、W、D、常量
S	输出	WORD	I、Q、W、D、常量 0

N: 串口号, 0 或 1, 支持立即数和 W 型参数

F: 请求帧数据表头, 支持 W 型参数

D: 目标或源数据区, 当为读命令时, 表示读取数据的存放位置, 当为写命令时, 表示

欲写入的数据的地址；支持 X、W 型参数，或立即数 0（无需指定 data 区时）

T: 超时设置 (ms)，支持立即数和 W 型参数，为 0 表示不修改之前的设定

C: 循环扫描时间 (ms)，支持立即数和 W 型参数，为 0 表示不修改之前的设定  
otSet: 超时设置 (ms)，支持立即数和 W 型参数，为 0 表示不修改之前的设定

S: 状态字，支持 W 型参数，或立即数 0（不为该请求帧设状态字）

用户可以设定循环扫描的周期时间（以 ms 为单位），系统只有一个统一的扫描周期，由最新的设定值决定。若该设定参数为 0，表示不修改之前的设定。系统默认的扫描周期 1s。

与添加非周期报文的指令相比，除了多了设定周期扫描时间的参数外，其余都相同，在此不再赘述。

执行条件

只有周期报文请求允许使能时，才能正确地执行周期报文请求的指令。所以一般情况下，用该标志位做添加周期报文请求指令的执行条件。

串口 0 相关标志位为 %SX43；串口 1 相关标志位为 %SX48。

若当前周期报文的队列已满，系统会将该标志位清零，从而不允许继续添加周期性的报文。系统最大支持 16 个队列的周期报文请求。执行删除周期报文的指令后，系统将周期报文队列清空。

### MBDR（删除周期 Modbus 请求）

功能说明

Modbus 主站指令，用于 Modbus-RTU 主站的通信。删除周期性的 Modbus-RTU 请求帧。指令及其操作数说明：

	名称	指令格式
IL	MBDR	MBDR N

参数	输入/输出	数据类型	允许的内存区
N	输入	WORD	I、Q、W、D、常量

N: 串口号，0 或 1,支持立即数和 W 型参数

执行该指令后，将清除所有的周期报文，并将周期报文请求允许的标志位置 1。

## 3.12 表格指令

表格的定义：表格是由字型数据组成，表格的第一个字储存表格最大条目数，表格的第二个字为当前表格计数，指明表格中已存储的数据个数。

**注意：**每个表格最大存储的条目个数是 100。

### TBL（向表格中添加数据）

功能说明

向指定的表格中添加数据

指令及其操作数说明：

	名称	指令格式
LD	TBL	
IL	TBL	TBL DATA, TABLE

参数	输入/输出	数据类型	允许的内存区
DATA	输入	WORD	I、Q、W、D、P、常量
TABLE	输入	WORD	I、Q、W、D

DATA: 指定要添加到表格中的数据。

TABLE: 表格的首地址。

指令使用举例:


D		当IX10.0为1时，TBL 指令执行，将10添加到QW0开始的表格中																																																	
L	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>TBL</td> <td>10 %QW0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	TBL	10 %QW0																																					
条	步	指令	操作数																																																
1	1	LD	%IX0.0																																																
	2	TBL	10 %QW0																																																
	<table border="1"> <thead> <tr> <th>QW0</th> <th>QW1</th> <th>QW2</th> <th>QW3</th> <th>QW4</th> <th>QW5</th> <th>QW6</th> </tr> </thead> <tbody> <tr> <td>4</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>4</td> <td>1</td> <td>10</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>4</td> <td>2</td> <td>10</td> <td>10</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>4</td> <td>3</td> <td>10</td> <td>10</td> <td>10</td> <td>0</td> <td>0</td> </tr> <tr> <td>4</td> <td>4</td> <td>10</td> <td>10</td> <td>10</td> <td>10</td> <td>0</td> </tr> <tr> <td>4</td> <td>4</td> <td>10</td> <td>10</td> <td>10</td> <td>10</td> <td>0</td> </tr> </tbody> </table>	QW0	QW1	QW2	QW3	QW4	QW5	QW6	4	0	0	0	0	0	0	4	1	10	0	0	0	0	4	2	10	10	0	0	0	4	3	10	10	10	0	0	4	4	10	10	10	10	0	4	4	10	10	10	10	0	左边表格为 TBL 执行 5 次后的结果，QW0 的值是预先填入的
QW0	QW1	QW2	QW3	QW4	QW5	QW6																																													
4	0	0	0	0	0	0																																													
4	1	10	0	0	0	0																																													
4	2	10	10	0	0	0																																													
4	3	10	10	10	0	0																																													
4	4	10	10	10	10	0																																													
4	4	10	10	10	10	0																																													

### FIFO（先入先出）

#### 功能说明

将表格中的第一个数据移动到指定的地址中，表格中其他数据一次上移，表格计数减1。

指令及其操作数说明:

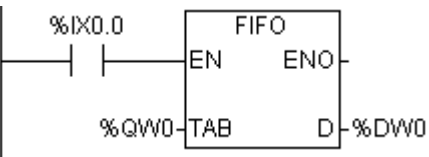
	名称	指令格式
LD	FIFO	
IL	FIFO	FIFO TAB, D

参数	输入/输出	数据类型	允许的内存区
TAB	输入	WORD	I、Q、W、D
D	输出	WORD	I、Q、W、D

TAB: 要移出数据的表格头地址。

D: 要存放数据的位置。

指令使用举例:

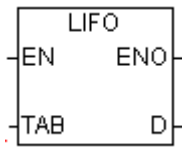
D		<p>当 IX10.0 为 1 时, FIFO 指令执行, 将 QW0 开始的表格中第一个数据移到 DW0 中。</p>																																								
L	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>FIFO</td> <td>%QW0 %DWO</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	FIFO	%QW0 %DWO																												
条	步	指令	操作数																																							
1	1	LD	%IX0.0																																							
	2	FIFO	%QW0 %DWO																																							
	<table border="1"> <thead> <tr> <th>QW0</th> <th>QW1</th> <th>QW2</th> <th>QW3</th> <th>QW4</th> <th>QW5</th> <th>QW6</th> <th>DW0</th> </tr> </thead> <tbody> <tr> <td>4</td> <td>4</td> <td>100</td> <td>10</td> <td>100</td> <td>10</td> <td>0</td> <td>0</td> </tr> <tr> <td>4</td> <td>3</td> <td>10</td> <td>100</td> <td>10</td> <td>10</td> <td>0</td> <td>100</td> </tr> <tr> <td>4</td> <td>2</td> <td>100</td> <td>10</td> <td>10</td> <td>10</td> <td>0</td> <td>10</td> </tr> <tr> <td>4</td> <td>1</td> <td>10</td> <td>10</td> <td>10</td> <td>10</td> <td>0</td> <td>100</td> </tr> </tbody> </table>	QW0	QW1	QW2	QW3	QW4	QW5	QW6	DW0	4	4	100	10	100	10	0	0	4	3	10	100	10	10	0	100	4	2	100	10	10	10	0	10	4	1	10	10	10	10	0	100	<p>左边表格为 FIFO 执行 3 次后的结果, QW0 的值是预先填入的。灰色部分是移动后留下的数据, 其数值保持原来的不变, 但因表格计数已经改变, 所以有可能会被其他表格指令覆盖掉。</p>
QW0	QW1	QW2	QW3	QW4	QW5	QW6	DW0																																			
4	4	100	10	100	10	0	0																																			
4	3	10	100	10	10	0	100																																			
4	2	100	10	10	10	0	10																																			
4	1	10	10	10	10	0	100																																			

### LIFO (后入先出)

#### 功能说明

将表格中的最后一个填入的数据移动到指定的地址中, 表格中其他数据保持不变, 表格计数减 1。

指令及其操作数说明:

	名称	指令格式
LD	LIFO	

IL	LIFO	LIFO TAB, D
----	------	-------------

参数	输入/输出	数据类型	允许的内存区
TAB	输入	WORD	I、Q、W、D
D	输出	WORD	I、Q、W、D

TAB: 要移出数据的表格头地址。

D: 要存放数据的位置。

指令使用举例:

D		<p>当 IX10.0 为 1 时, LIFO 指令执行, 将 QW0 开始的表格中第一个数据移到 DW0 中。</p>																																								
L	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>LIFO</td> <td>%QW0 %DW0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	LIFO	%QW0 %DW0																												
条	步	指令	操作数																																							
1	1	LD	%IX0.0																																							
	2	LIFO	%QW0 %DW0																																							
	<table border="1"> <thead> <tr> <th>QW0</th> <th>QW1</th> <th>QW2</th> <th>QW3</th> <th>QW4</th> <th>QW5</th> <th>QW6</th> <th>DW0</th> </tr> </thead> <tbody> <tr> <td>4</td> <td>4</td> <td>100</td> <td>10</td> <td>100</td> <td>10</td> <td>0</td> <td>0</td> </tr> <tr> <td>4</td> <td>3</td> <td>100</td> <td>10</td> <td>100</td> <td>10</td> <td>0</td> <td>10</td> </tr> <tr> <td>4</td> <td>3</td> <td>100</td> <td>10</td> <td>100</td> <td>10</td> <td>0</td> <td>100</td> </tr> <tr> <td>4</td> <td>2</td> <td>100</td> <td>10</td> <td>100</td> <td>10</td> <td>0</td> <td>10</td> </tr> </tbody> </table>	QW0	QW1	QW2	QW3	QW4	QW5	QW6	DW0	4	4	100	10	100	10	0	0	4	3	100	10	100	10	0	10	4	3	100	10	100	10	0	100	4	2	100	10	100	10	0	10	<p>左边表格为 LIFO 执行 3 次后的结果, QW0 的值是预先填入的。灰色部分是移动后留下的数据, 其数值保持原来的不变, 但因表格计数已经改变, 所以有可能会被其他表格指令覆盖掉。</p>
QW0	QW1	QW2	QW3	QW4	QW5	QW6	DW0																																			
4	4	100	10	100	10	0	0																																			
4	3	100	10	100	10	0	10																																			
4	3	100	10	100	10	0	100																																			
4	2	100	10	100	10	0	10																																			

### FILL (内存填充)

功能说明

将 N 个指定的数据填写到指定的内存中。

指令及其操作数说明:

	名称	指令格式
LD	FILL	
IL	FILL	FILL DATA, N, D

参数	输入/输出	数据类型	允许的内存区
DATA	输入	WORD	I、Q、W、D、P、常量
N	输入	WORD	I、Q、W、D、P、常量
D	输出	WORD	I、Q、W、D

DATA: 内存中要填充的数据。

D: 要存放数据的位置。

N: 要填充内存的数量, 范围为 1 到 255。

指令使用举例:

D		<p>当 IX10.0 为 1 时, FIFO 指令执行, 将常数 11 填充到 QW0 开始的 7 个内存地址中。</p>																					
L	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">条</th> <th style="width: 10%;">步</th> <th style="width: 30%;">指令</th> <th style="width: 50%;">操作数</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td style="text-align: center;">2</td> <td>FILL</td> <td>11 7 %QW0</td> </tr> </tbody> </table>	条	步	指令	操作数	1	1	LD	%IX0.0		2	FILL	11 7 %QW0										
条	步	指令	操作数																				
1	1	LD	%IX0.0																				
	2	FILL	11 7 %QW0																				
	<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th>QW0</th> <th>QW1</th> <th>QW2</th> <th>QW3</th> <th>QW4</th> <th>QW5</th> <th>QW6</th> </tr> </thead> <tbody> <tr> <td>4</td> <td>4</td> <td>100</td> <td>10</td> <td>100</td> <td>10</td> <td>0</td> </tr> <tr> <td>11</td> <td>11</td> <td>11</td> <td>11</td> <td>11</td> <td>11</td> <td>11</td> </tr> </tbody> </table>	QW0	QW1	QW2	QW3	QW4	QW5	QW6	4	4	100	10	100	10	0	11	11	11	11	11	11	11	<p>左边表格为 FILL 执行后的结果, QW0 开始的内存值是预先填入的。</p>
QW0	QW1	QW2	QW3	QW4	QW5	QW6																	
4	4	100	10	100	10	0																	
11	11	11	11	11	11	11																	

### FND (表格查找)

功能说明

从指定的条目开始查找匹配指定模式的条目, 找到后将条目的值输出到指定地址, 没有找到则返回表格最大条目数。

指令及其操作数说明:

	名称	指令格式
LD	FND	
IL	FND	FND TAB,PNT,IDX,CMD

参数	输入/输出	数据类型	允许的内存区
TAB	输入	WORD	I、Q、W、D
PNT	输入	WORD	I、Q、W、D、P、常量
IDX	输入/输出	WORD	I、Q、W、D
CMD	输入	BYTE	I、Q、W

TAB: 要查找的表格, 这里指定的不是表头, 而是表的第二个字——条目计数。

PNT: 要匹配的数据。

IDX: 从第几个条目开始查找, 找到后将条目值返回到该地址

CMD: 匹配模式, 取值范围只能是 1 到 4

CMD 的值	匹配模式
1	等于
2	不等于
3	小于
4	大于

指令使用举例：

D		<p>当 IX10.0 为 1 时，FND 指令执行，将 QW1 开始的表格中 WW1 指定的条目开始，等于 3 的第一个条目的值返回到 WW1 中。</p>																																
L	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>FND</td> <td>%QW1 3 %WW1 1</td> </tr> </tbody> </table>	条	步	指令	操作数	1	1	LD	%IX0.0		2	FND	%QW1 3 %WW1 1																					
条	步	指令	操作数																															
1	1	LD	%IX0.0																															
	2	FND	%QW1 3 %WW1 1																															
	<table border="1"> <tbody> <tr> <td>QW0</td><td>QW1</td><td>QW2</td><td>QW3</td><td>QW4</td><td>QW5</td><td>QW6</td><td>WW1</td> </tr> <tr> <td>10</td><td>10</td><td>8</td><td>1</td><td>2</td><td>3</td><td>4</td><td>0</td> </tr> <tr> <td>QW7</td><td>QW8</td><td>QW9</td><td>QW10</td><td>QW11</td><td>QX20.0</td><td>QW7</td><td>WW1</td> </tr> <tr> <td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>1</td><td>5</td><td>3</td> </tr> </tbody> </table>	QW0	QW1	QW2	QW3	QW4	QW5	QW6	WW1	10	10	8	1	2	3	4	0	QW7	QW8	QW9	QW10	QW11	QX20.0	QW7	WW1	5	6	7	8	9	1	5	3	<p>左边表格为 FND 执行后的结果，QW0 开始的表格是预先填好的。</p>
QW0	QW1	QW2	QW3	QW4	QW5	QW6	WW1																											
10	10	8	1	2	3	4	0																											
QW7	QW8	QW9	QW10	QW11	QX20.0	QW7	WW1																											
5	6	7	8	9	1	5	3																											

### SCL（比例缩放）

功能说明

将 S 所指定的无符号 BIN 数值根据 C 里面所指定的参数（缩放前 A，B 点的值和缩放后 A，B 点的值）所决定的一次函数进行运算，转换成无符号 BCD 数，结果存放在 D 所指定的通道中。

S：转换对象通道编号

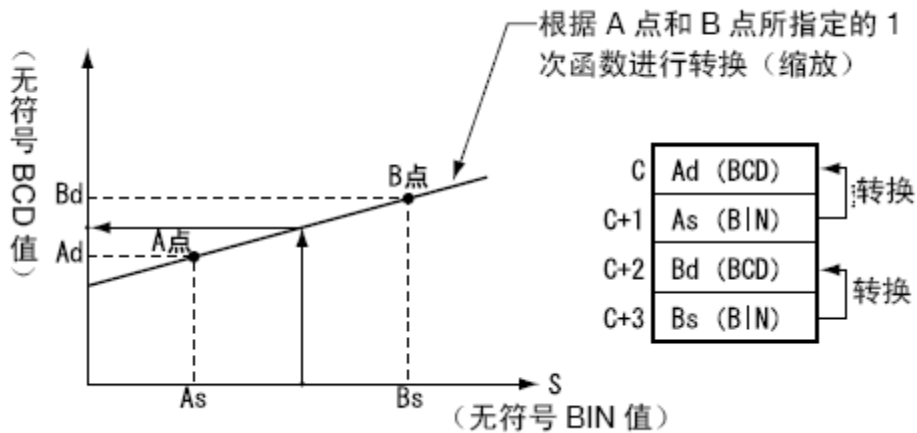
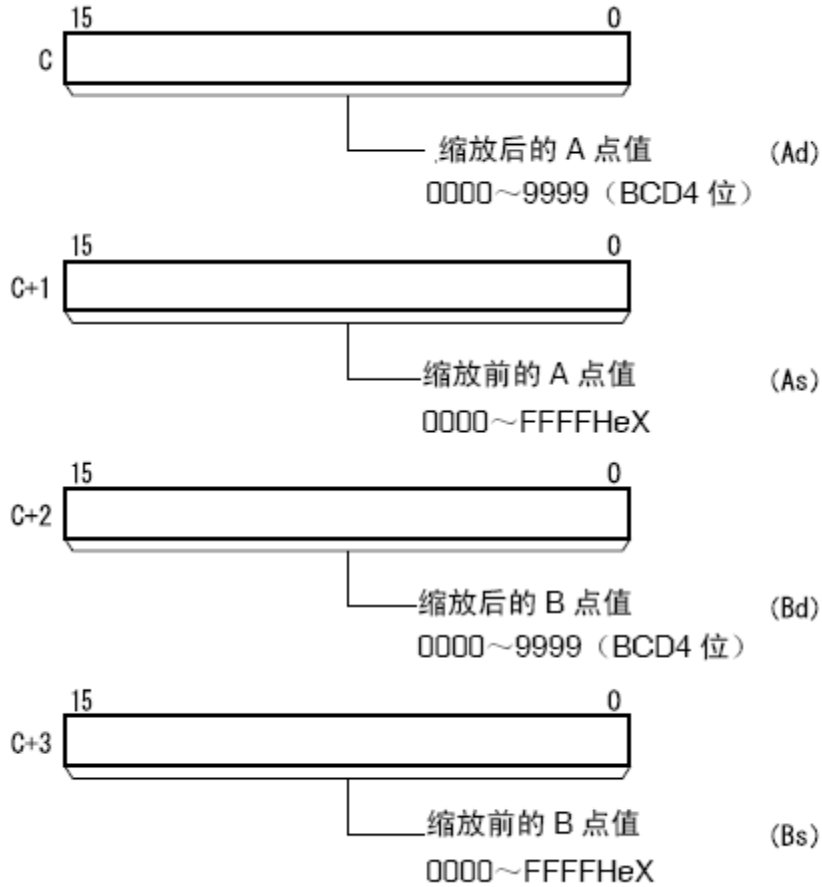
C：参数保存低位通道编号

D：转换结果保存通道编号

指令及其操作数说明：

	名称	指令格式
LD	SCL	
IL	SCL	SCL S,C,D

参数	输入/输出	数据类型	允许的内存区
EN	输入	BOOL	能量流
S	输入	WORD	I、Q、W、D、P、常量
C	输入	WORD	I、Q、W、D、P
D	输出	WORD	I、Q、W、D、P



转换公式:



$$D = Bd - \frac{(Bd - Ad)}{(Bs - As) \text{的BCD转换值}} \times (Bs - S)$$

转换结果小数点后四舍五入，结果小于 0 输出 0，大于 9999，输出 9999。如果当前 EN 值为 1，则该指令被执行，将 S 的值由无符号 BIN 数转换成无符号 BCD 数据并赋给 D；如果当前 EN 值为 0，则该指令不执行。

指令使用举例：

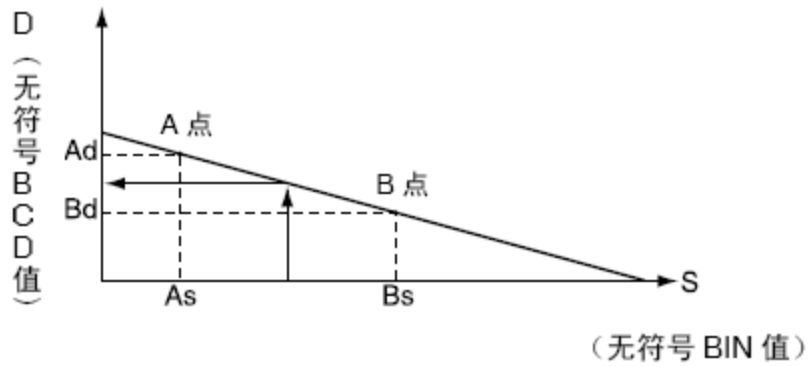
对于模拟信号 1~5V，0000~0FA0 Hex 的值被存储在 DW0 中时，将该值转换（缩放）为 0000~0300 的 BCD 值。

LD		IX0.0 为 ON 时，将来自模拟输入单元的 DW0 值根据 A 点（0000Hex→0000（BCD））和 B 点（0FA0Hex→0300（BCD））所决定的 1 次函数进行缩放，存储到 DW200。												
IL	<table border="1" style="margin: auto; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">条</th> <th style="width: 10%;">步</th> <th style="width: 20%;">指令</th> <th style="width: 60%;">操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>SCL</td> <td>%DW0 %DW100 %DW200</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX0.0		2	SCL	%DW0 %DW100 %DW200
条	步	指令	操作数											
1	1	LD	%IX0.0											
	2	SCL	%DW0 %DW100 %DW200											
示意图	<div style="display: flex; align-items: center;"> <div style="flex: 1;"> <p>D (DW0) 的内容</p> </div> <div style="flex: 1; margin-left: 20px;"> <table border="1" style="border-collapse: collapse;"> <tr> <td>C : DW100</td> <td style="text-align: center;">0 0 0 0</td> <td>Ad (BCD)</td> </tr> <tr> <td>C+1 : DW101</td> <td style="text-align: center;">0 0 0 0</td> <td>As (BIN)</td> </tr> <tr> <td>C+2 : DW102</td> <td style="text-align: center;">0 3 0 0</td> <td>Bd (BCD)</td> </tr> <tr> <td>C+3 : DW103</td> <td style="text-align: center;">0 F A 0</td> <td>Bs (BIN)</td> </tr> </table> </div> </div>		C : DW100	0 0 0 0	Ad (BCD)	C+1 : DW101	0 0 0 0	As (BIN)	C+2 : DW102	0 3 0 0	Bd (BCD)	C+3 : DW103	0 F A 0	Bs (BIN)
C : DW100	0 0 0 0	Ad (BCD)												
C+1 : DW101	0 0 0 0	As (BIN)												
C+2 : DW102	0 3 0 0	Bd (BCD)												
C+3 : DW103	0 F A 0	Bs (BIN)												

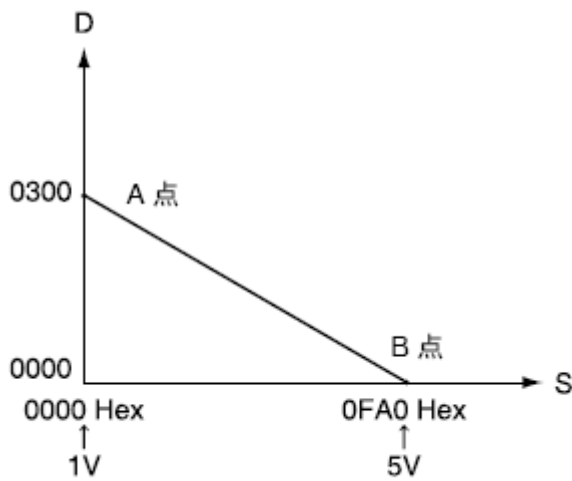
来自实际的模拟输入单元的变更值，对于 0.8V~5.2V 存储 FF38~1068Hex 的值。由于 SCL 指令将 S 的值作为 0000~FFFF Hex 的无符号 BIN 值处理，因此对于 1V(0000 Hex) 未满足的带符号 BIN 值 FF38~FFFFHex 无法进行正确缩放。所以，将来自实际模拟输入单元的转换值用 SCL 指令进行缩放时，为使 FF38Hex 转成 0000Hex，全体加 00C8 后执行 SCL 指令。

参考

关于逆缩放也可进行 As<Bs、Ad>Bd。



例如对于 1~5V (0000~0FAHex), 可进行向 0300~0000 的转换 (逆缩放)。



### SCL2 (比例缩放 2)

#### 功能说明

根据 C 中的偏移值参数, 将 S 中的有符号 BIN 值转换为有符号 BCD 值, 存放在 D 中。

S: 转换对象通道编号

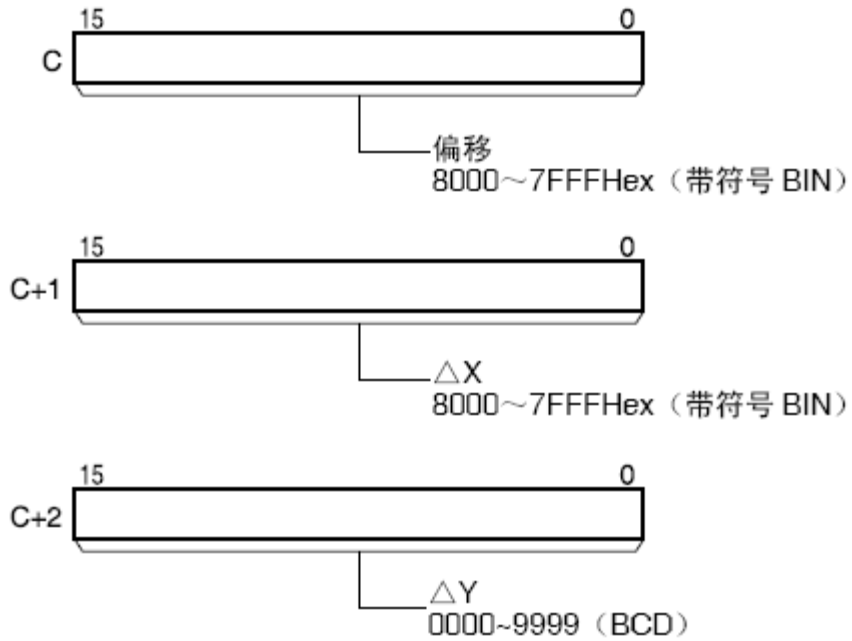
C: 参数保存低位通道编号

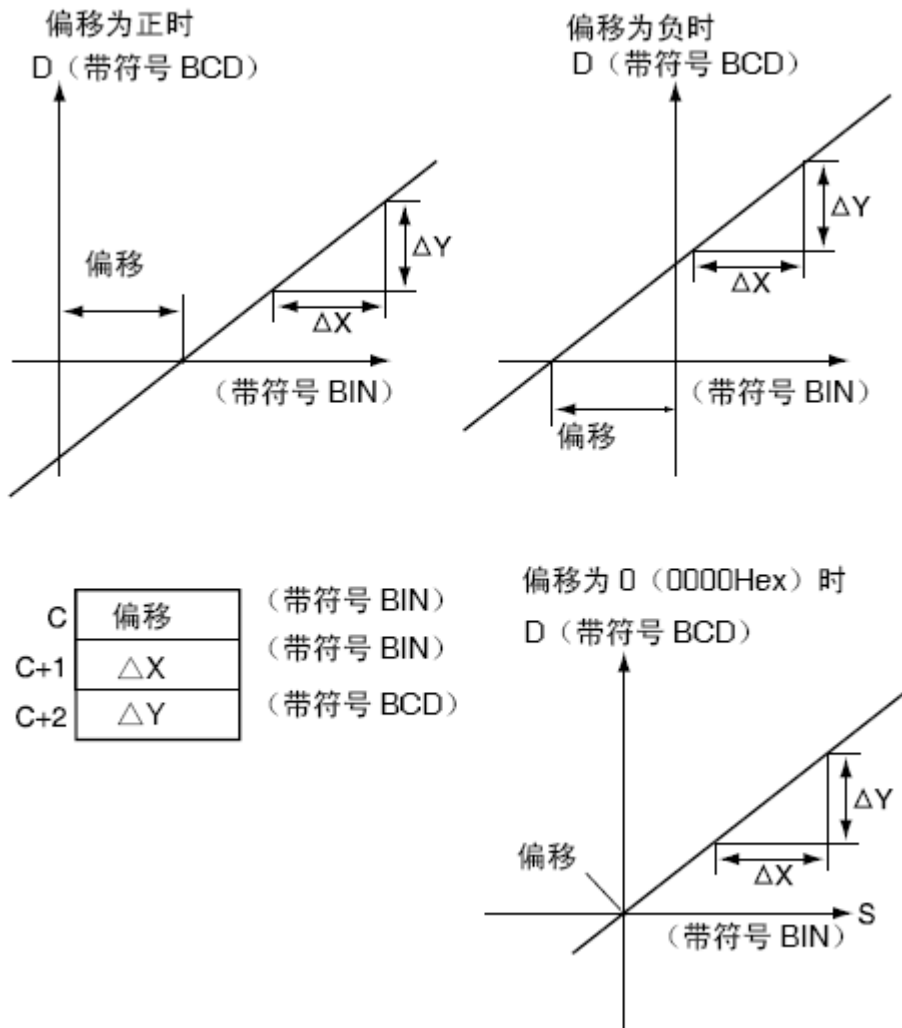
D: 转换结果保存通道编号

指令及其操作数说明:

	名称	指令格式
LD	SCL2	<pre> graph TD     subgraph SCL2         EN[EN]         S[S]         C[C]         D[D]     end </pre>
IL	SCL2	SCL2 S,C,D

参数	输入/输出	数据类型	允许的内存区
EN	输入	BOOL	能量流
S	输入	INT	I、Q、W、D、P、常量
C	输入	WORD	I、Q、W、D、P
D	输出	WORD	I、Q、W、D、P





转换公式:

$$D = \frac{\Delta Y}{\Delta X \text{的BCD转换值}} \times \{ (S \text{的BCD转换值}) - (\text{偏移的的BCD转换值}) \}$$

注:  $\frac{\Delta Y}{\Delta X}$  为斜率

偏移可以是正数、0、负数。

斜率可以是正数、0、负数。因此，可以进行逆缩放。

D (BCD 数据) 表示绝对值，进位标志 (CY) 表示正负。

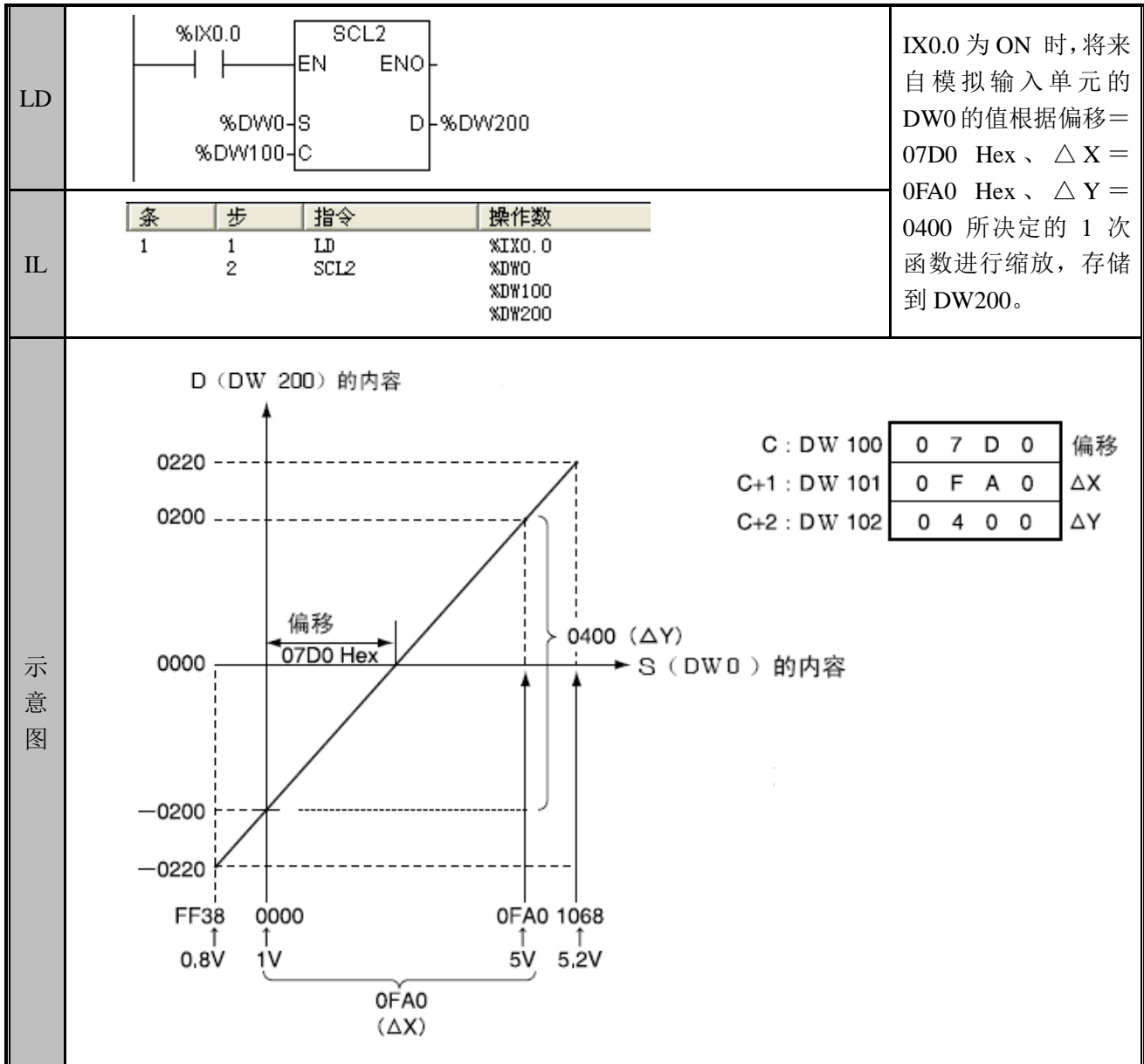
因此，转换结果在 -9999~9999 的范围内输出，转换结果超过上限 (9999) 时，输出 9999，超过下限时，输出 -9999。

如果当前 EN 值为 1，该指令被执行，将 S 的值由有符号 BIN 数转换成有符号 BCD 数并赋给 D；如果当前 EN 值为 0，该指令不执行。

指令使用举例:

对于模拟信号 1~5V，进行 -200~+200 的缩放时

对于模拟信号 1~5V，带符号 BIN 值 0000~0FA0 Hex 的值被存储在 DW0 中时，将该值转换 (缩放) 为 -0200~0200 的 BCD 值。



### SCL3 (比例缩放 3)

#### 功能说明

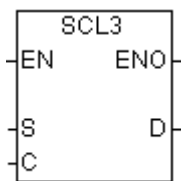
根据 C 中的偏移值参数, 将 S 中的有符号 BCD 值转换为有符号 BIN 值, 存放在 D 中。

S: 转换对象通道编号

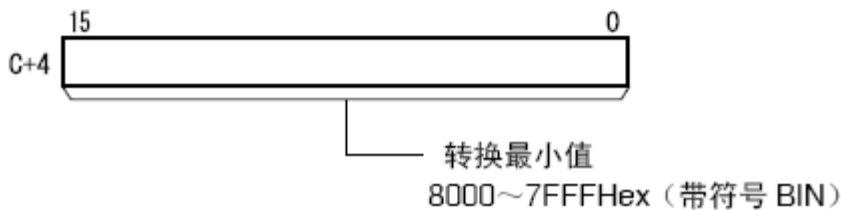
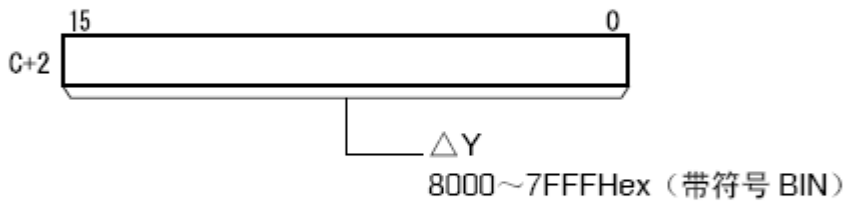
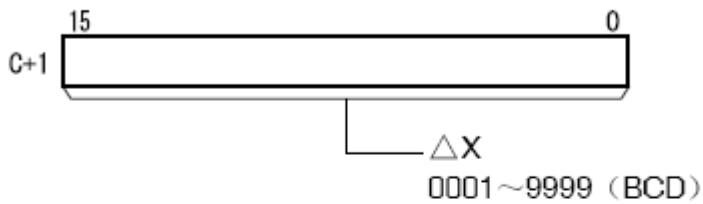
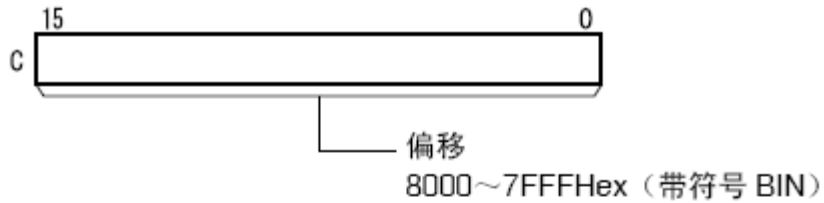
C: 参数保存低位通道编号

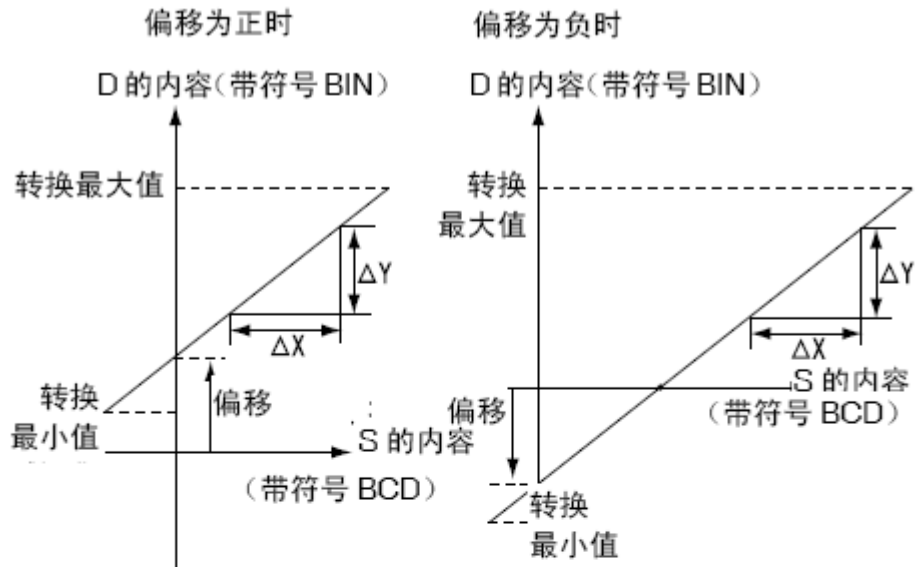
D: 转换结果保存通道编号

指令及其操作数说明:

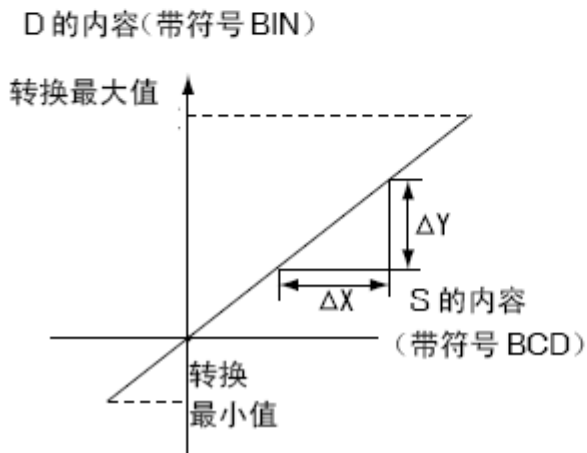
	名称	指令格式
LD	SCL3	
IL	SCL3	SCL3 S,C,D

参数	输入/输出	数据类型	允许的内存区
EN	输入	BOOL	能量流
S	输入	WORD	I、Q、W、D、P、常量
C	输入	WORD	I、Q、W、D、P
D	输出	INT	I、Q、W、D、P





偏移为 0 时



转换公式:

$$D = \frac{\Delta Y}{\Delta X \text{ 的 BIN 转换值}} \times (S \text{ 的 BIN 转换值}) + (\text{偏移})$$

注:  $\frac{\Delta Y}{\Delta X}$  为斜率

偏移可以为正数、0、负数。

斜率可以为正数、0、负数。因此，也可以进行逆缩放。

S 的 BCD 数据表示绝对值，用指令执行时的进位标志 (CY) 来区别正负。因此，转换对象数据为 -9999~9999 的范围。

转换结果的小数点之后数据四舍五入。

转换结果超过转换最大值 (C+3) 时，输出转换最大值，超过转换最小值 (C+4) 时，输出转换最小值。

如果当前 EN 值为 1，该指令被执行：将 S 的值由有符号 BCD 数转换成有符号 BIN 数据并赋给 D。如果当前 EN 值为 0，该指令不执行。

指令使用举例：

将 0~200 的值缩放为模拟信号（例如 1~5V）时，将带符号 BCD 值 0000~0200 的值转换（缩放）为模拟输出单元用数据带符号 BIN 值 0000~0FA0。

LD		<p>IX0.0 为 ON 时，将 DW0 的值根据偏移 = 0000Hex、<math>\Delta X = 0200</math>、<math>\Delta Y = 0FA0</math> Hex 所决定的 1 次函数进行缩放，存储到 DW200。</p>															
IL	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">条</th> <th style="width: 10%;">步</th> <th style="width: 20%;">指令</th> <th style="width: 60%;">操作数</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">LD</td> <td style="text-align: center;">%IX0.0</td> </tr> <tr> <td></td> <td style="text-align: center;">2</td> <td style="text-align: center;">SCL3</td> <td style="text-align: center;">%DW0 %DW100 %DW200</td> </tr> </tbody> </table>	条	步	指令	操作数	1	1	LD	%IX0.0		2	SCL3	%DW0 %DW100 %DW200				
条	步	指令	操作数														
1	1	LD	%IX0.0														
	2	SCL3	%DW0 %DW100 %DW200														
示意图		<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 20%;">C : DW 100</td> <td style="width: 20%; text-align: center;">0 0 0 0</td> <td style="width: 60%;">偏移</td> </tr> <tr> <td>C+1 : DW 101</td> <td style="text-align: center;">0 2 0 0</td> <td><math>\Delta X</math></td> </tr> <tr> <td>C+2 : DW 102</td> <td style="text-align: center;">0 F A 0</td> <td><math>\Delta Y</math></td> </tr> <tr> <td>C+3 : DW 103</td> <td style="text-align: center;">1 0 6 8</td> <td>转换最</td> </tr> <tr> <td>C+4 : DW 104</td> <td style="text-align: center;">F F 3 8</td> <td>转换最</td> </tr> </table>	C : DW 100	0 0 0 0	偏移	C+1 : DW 101	0 2 0 0	$\Delta X$	C+2 : DW 102	0 F A 0	$\Delta Y$	C+3 : DW 103	1 0 6 8	转换最	C+4 : DW 104	F F 3 8	转换最
C : DW 100	0 0 0 0	偏移															
C+1 : DW 101	0 2 0 0	$\Delta X$															
C+2 : DW 102	0 F A 0	$\Delta Y$															
C+3 : DW 103	1 0 6 8	转换最															
C+4 : DW 104	F F 3 8	转换最															

### 3.13 定时器

TON（打开延迟定时器）

指令及其操作数说明：

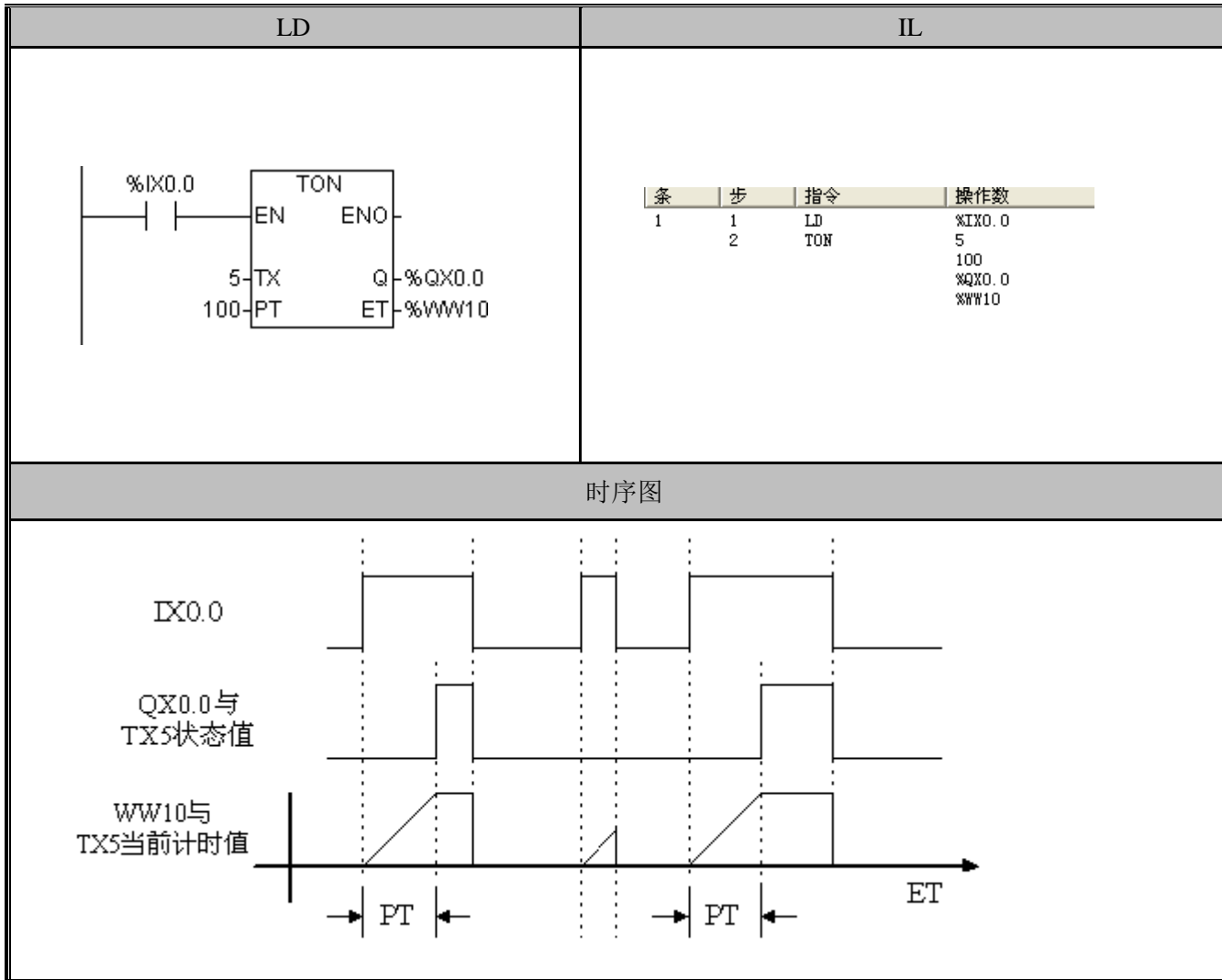
	名称	指令格式
LD	TON	
IL	TON	TON TX,PT,Q,ET



参数	输入/输出	数据类型	允许的内存区
TX	输入	常数	---
PT	输入	WORD	I、Q、W、D、P、常量
Q	输入	BOOL	I、Q、W、S、TR
ET	输出	WORD	I、Q、W、D、P

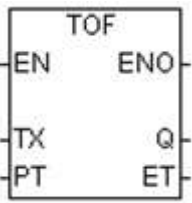
若检测到当前值的上升沿，则 TX 开始启动定时，若计时值大于等于预设值 PT 时，TX 停止，其状态值被置为 1；若当前值变为 0，则 TX 被复位，其状态值及计时值均被清零。每次扫描 TON 后，当前值均被设置为 TX 的状态值。

指令使用举例：



### TOF（关闭延迟定时器）

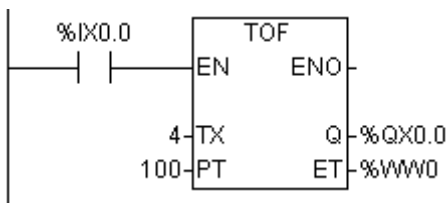
指令及其操作数说明：

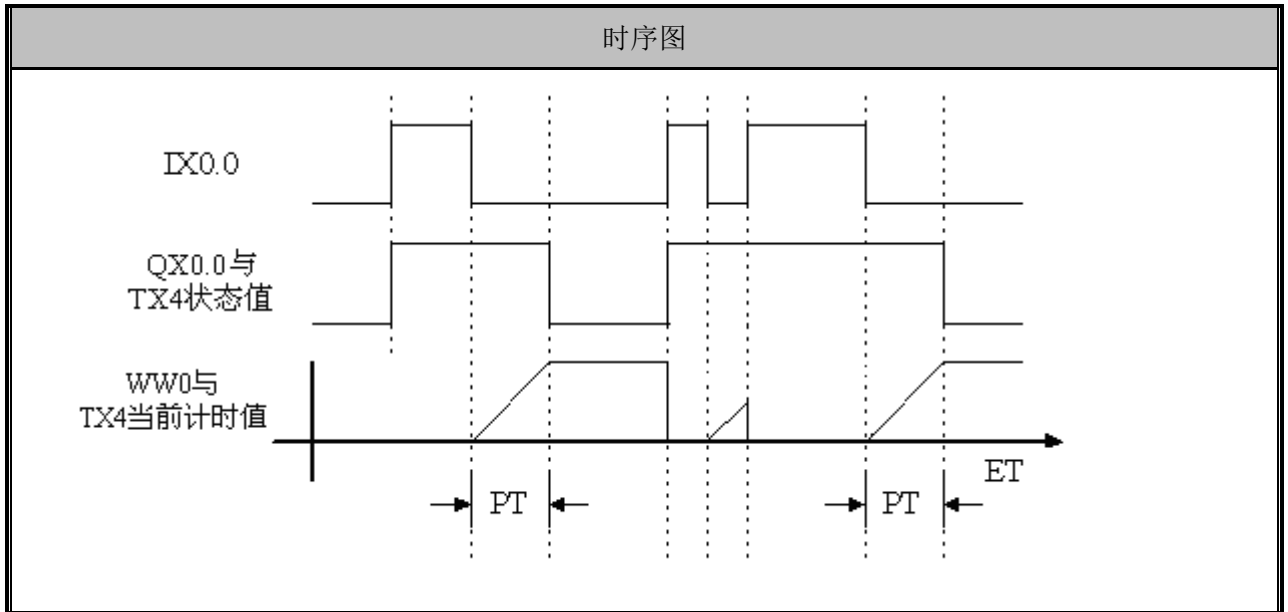
	名称	指令格式
LD	TOF	
IL	TOF	TOF TX,PT,Q,ET

参数	输入/输出	数据类型	允许的内存区
TX	输入	常数	---
PT	输入	WORD	I、Q、W、D、P、常量
Q	输入	BOOL	I、Q、W、S、TR
ET	输出	WORD	I、Q、W、D、P

若检测到当前值的下降沿，则 TX 开始启动定时，若计时值大于等于预设值 PT 时，TX 停止，其状态值被置为 0；若当前值变为 1，则 TX 被复位，其状态值被置为 1，且计时值被清零。每次扫描 TOF 后，当前值均被设置为 TX 的状态值。

指令使用举例：

LD	IL												
	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX0.0</td> </tr> <tr> <td></td> <td>2</td> <td>TOF</td> <td>4 100 %QX0.0 %WVO</td> </tr> </tbody> </table>	条	步	指令	操作数	1	1	LD	%IX0.0		2	TOF	4 100 %QX0.0 %WVO
条	步	指令	操作数										
1	1	LD	%IX0.0										
	2	TOF	4 100 %QX0.0 %WVO										



### TP（脉冲延迟定时器）

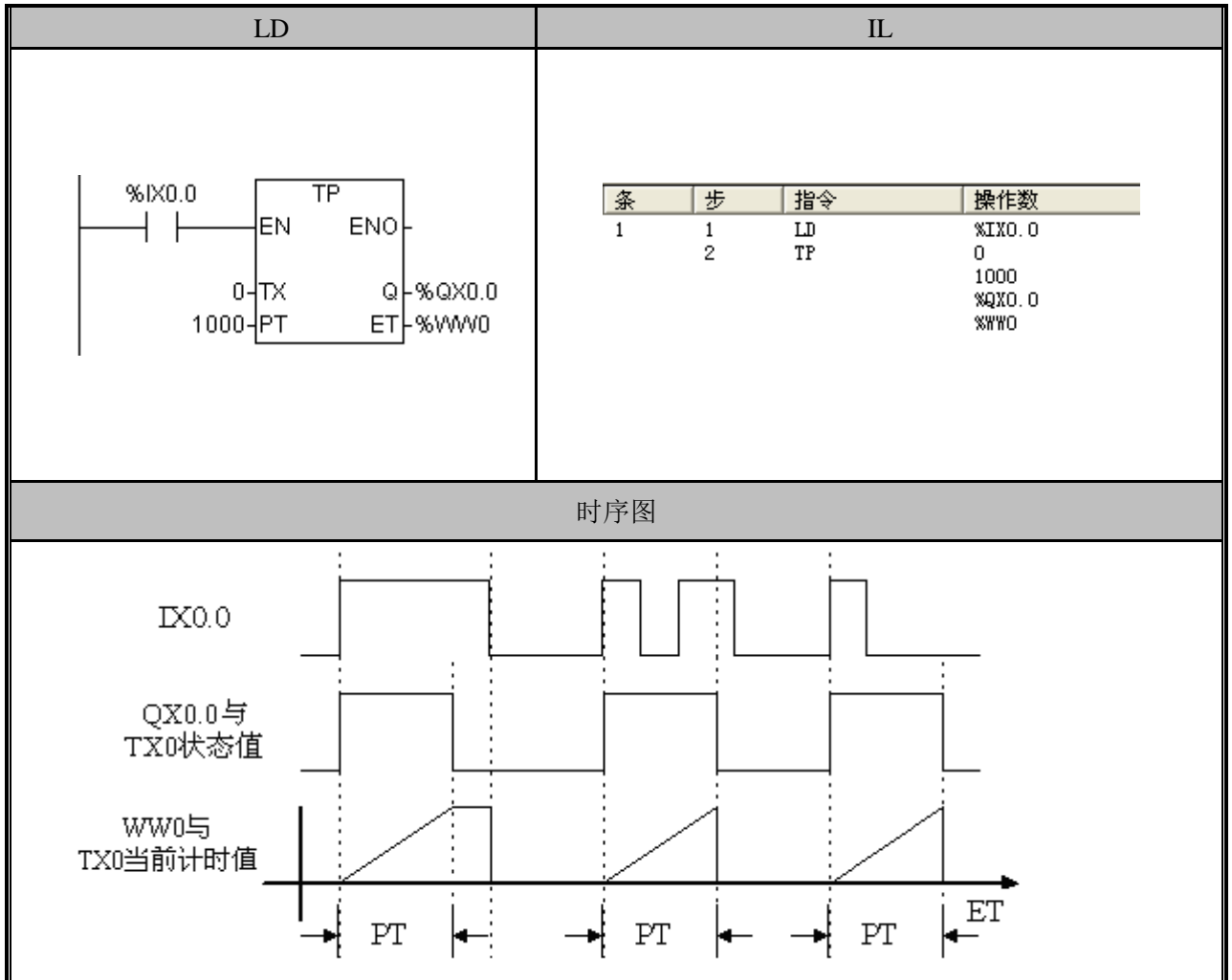
指令及其操作数说明：

	名称	指令格式
LD	TP	
IL	TP	TP TX, PT, Q, ET

参数	输入/输出	数据类型	允许的内存区
TX	输入	常数	---
PT	输入	WORD	I、Q、W、D、P、常量
Q	输入	BOOL	I、Q、W、S、TR
ET	输出	WORD	I、Q、W、D、P

若检测到当前值的上升沿，则 TX 开始启动定时，它的状态值输出一个恒定宽度的脉冲，脉宽值为预设时间 PT。每次扫描 TP 后，当前值均被设置为 TX 的状态值。

指令使用举例：



### 3.14 计数器

计数器是 IEC61131-3 标准中定义的功能块之一，共有 CTU、CTD 和 CTUD 三种。  
 CX 范围：编号 0—63，要求计数频率为 50Hz 以下（周期为 20ms 以上）。  
 CTU、CTD、CTUD 等 3 种指令不能使用同样的定时器号码，例如：  
 使用了 CTU CX0，就不能使用 CTD CX0；CTD CX31 和 CTUD CX31 不能同时使用。

#### CTU（增计数器）、CTD（减计数器）

##### 功能说明

增计数器“增计数”指令（CTU）在“增计数（CU）”输入的上升沿时从当前值增计数到预设值（PV）。当“当前值（CV）”大于等于预设值时，计数器输出位（Q）接通，增计数器停止计数；当重设输入（R）启用时，计数器重设。

减计数器“减计数”指令（CTD）在“减计数（CD）”输入的上升沿从预设值（PV）减计数。当“当前值（CV）”等于零时，计数器输出位（Q）接通，减计数器停止计数；当载入输入（LD）启用时，计数器重设并用预设值载入当前值。

指令及其操作数说明：

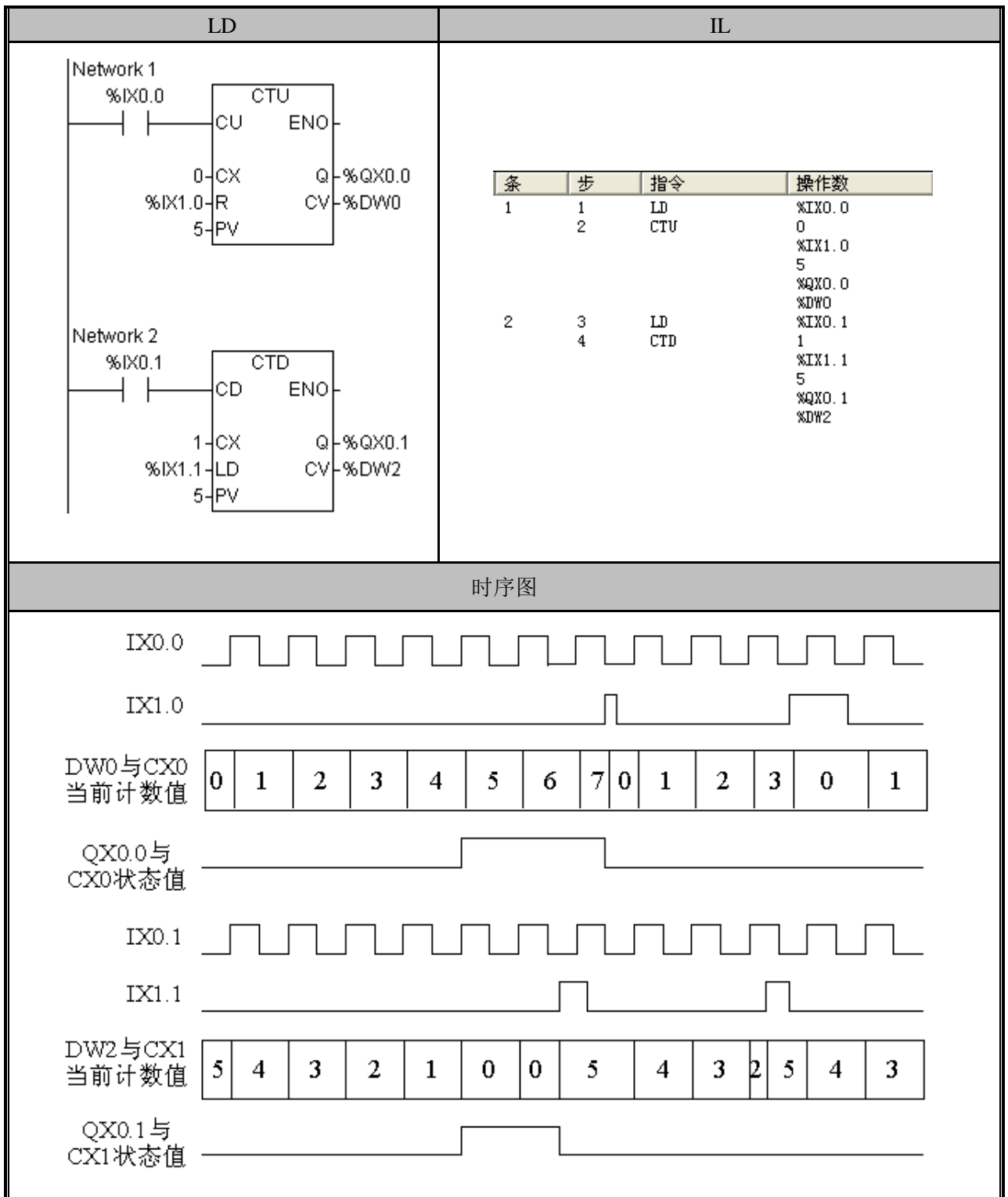
	名称	指令格式
LD	CTU	
	CTD	
IL	CTU	CTU CX,R,PV,Q,CV
	CTD	CTD CX,LD,PV,Q,CV

参数	输入/输出	数据类型	允许的内存区
CX	输入	常数	---
CU	输入	BOOL	I、Q、W、S、TR
CD	输入	BOOL	I、Q、W、S、TR
R	输入	BOOL	I、Q、W、S、TR
LD	输入	BOOL	I、Q、W、S、TR
PV	输入	WORD	I、Q、W、D、P、常量
Q	输出	BOOL	I、Q、W
CV	输出	WORD	I、Q、W、D、P

CTU 用于对当前值的上升沿进行递增（每次递增 1）计数，当 CX 的计数值 CV 大于等于预设值 PV 时，其输出 Q 及其状态值均被置为 1。若 R 输入端为 1，则 CX 被复位，其输出 Q 及状态值均被置为 0，同时 CV 也被清零。

CTD 用于对当前值的上升沿从预设值 PV 开始进行递减（每次递减 1）计数，当 CX 的计数值 CV 等于 0 时，其输出 Q 及其状态值均被置为 1，并且 CX 停止计数。若 LD 输入端为 1，则 CX 被复位，其输出 Q 及状态值均被置为 0，同时把预设值 PV 重新装入当前值 CV。

指令使用举例：



### CTUD (增/减计数器)

功能说明:

增/减计数器“增/减计数”指令 (CTUD) 在增计数 (CU) 或减计数 (CD) 输入的上升沿从当前值 (CV) 增或减计数。当前值等于预置时, 增输出 (QU) 接通; 当前值等于零时, 向下输出 (QD) 接通。当载入 (LD) 输入启用时, 计数器用预设值 (PV) 载入当前

值。相似地，当重设（R）启用时，计数器重设并用 0 载入当前值。当计数器达到预置或 0 时，停止计数。

指令及其操作数说明：

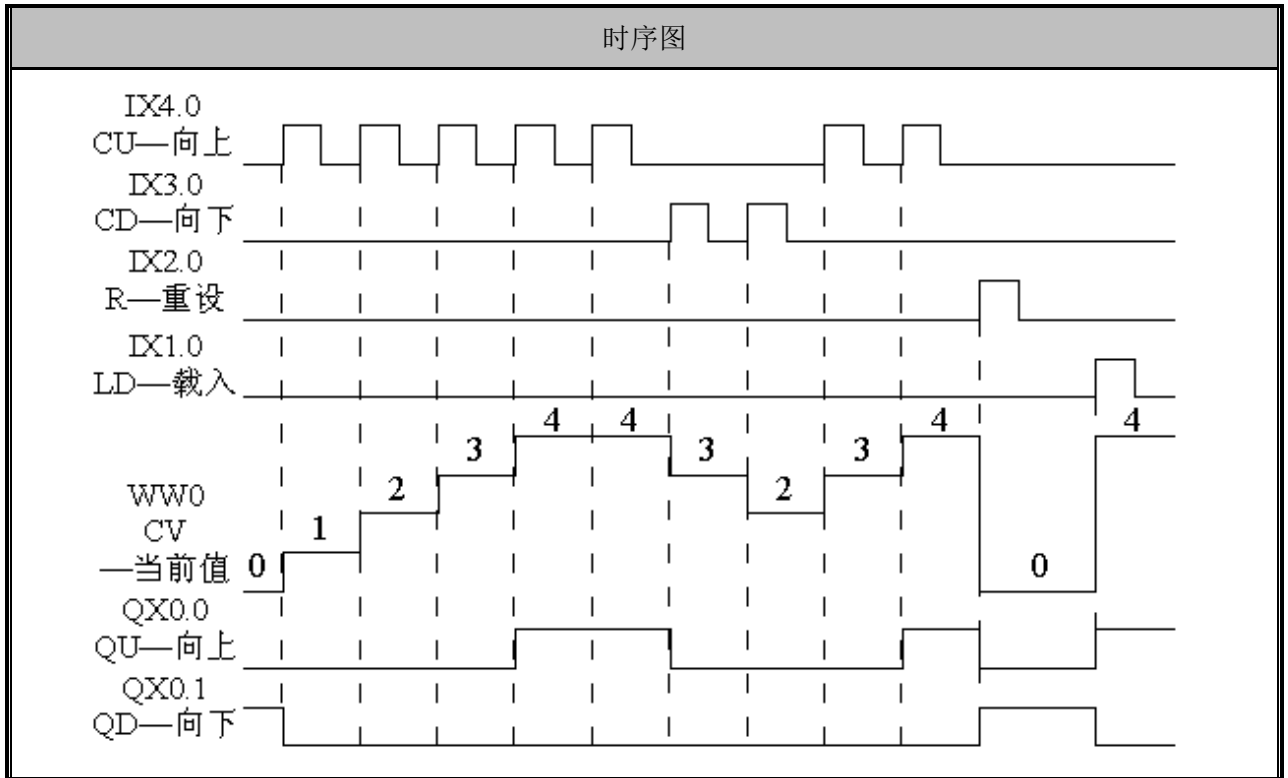
	名称	指令格式
LD	CTUD	
IL	CTUD	CTUD CU,CD,CX,R,LD,PV,QU,QD,CV

参数	输入/输出	数据类型	允许的内存区
CX	输入	常数	---
CU	输入	BOOL	I、Q、W、S、TR
CD	输入	BOOL	I、Q、W、S、TR
R	输入	BOOL	I、Q、W、S、TR
LD	输入	BOOL	I、Q、W、S、TR
PV	输入	WORD	I、Q、W、D、P、常量
QU	输出	BOOL	I、Q、W
QD	输出	BOOL	I、Q、W
CV	输出	WORD	I、Q、W、D、P

CTUD 在增计数（CU）或减计数（CD）输入的上升沿从当前值（CV）增或减计数。当前值等于预置时，增输出（QU）接通；当前值等于零时，向下输出（QD）接通。当载入（LD）输入启用时，计数器用预设值（PV）载入当前值。相似地，当重设（R）启用时，计数器重设并用 0 载入当前值。当计数器达到预置或 0 时，停止计数。

指令使用举例：

LD	IL												
	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX4.0</td> </tr> <tr> <td></td> <td>2</td> <td>CTUD</td> <td>%IX3.0 31 %IX2.0 %IX1.0 4 %QX0.0 %QX0.1 %WV0</td> </tr> </tbody> </table>	条	步	指令	操作数	1	1	LD	%IX4.0		2	CTUD	%IX3.0 31 %IX2.0 %IX1.0 4 %QX0.0 %QX0.1 %WV0
条	步	指令	操作数										
1	1	LD	%IX4.0										
	2	CTUD	%IX3.0 31 %IX2.0 %IX1.0 4 %QX0.0 %QX0.1 %WV0										



### 3.15 字符串指令

字符串格式定义：字符串由字节组成，第一个字节定义字符串的长度，字符串长度为0到254。

#### SLEN（字符串长度）

功能说明

返回指定字符串的长度到指定地址。

指令及其操作数说明：

	名称	指令格式
LD	SLEN	
IL	SLEN	SLEN IN,OUT

参数	输入/输出	数据类型	允许的内存区
IN	输入	BYTE	I、Q、W
OUT	输出	BYTE	I、Q、W

IN：字符串首地址。

OUT：字符串长度要输出到的地址。

指令使用举例：



D		当 IX10.0 为 1 时，SLEN 指令执行，将 QB1 开始的字符串长度输出到 WB0											
L	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX10.0</td> </tr> <tr> <td></td> <td>2</td> <td>SLEN</td> <td>%QB1 %WB0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX10.0		2	SLEN
条	步	指令	操作数										
1	1	LD	%IX10.0										
	2	SLEN	%QB1 %WB0										

### SCPY（字符串复制）

#### 功能说明

将指定的字符串复制到另一个字符串中。

指令及其操作数说明：

	名称	指令格式
LD	SCPY	
IL	SCPY	SCPY IN,OUT

参数	输入/输出	数据类型	允许的内存区
IN	输入	BYTE	I、Q、W
OUT	输出	BYTE	I、Q、W

IN：字符串首地址。

OUT：字符串要输出到的地址。

指令使用举例：

D		当 IX10.0 为 1 时，SCPY 指令执行，将 QB1 开始的字符串复制到 WB0 开始的字符串中											
L	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX10.0</td> </tr> <tr> <td></td> <td>2</td> <td>SCPY</td> <td>%QB1 %WB0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX10.0		2	SCPY
条	步	指令	操作数										
1	1	LD	%IX10.0										
	2	SCPY	%QB1 %WB0										

### SCAT（字符串连接）

#### 功能说明

将指定的字符串附加到另一个字符串的尾部。

指令及操作数说明

	名称	指令格式
LD	SCAT	
IL	SCAT	SCAT IN,OUT

参数	输入/输出	数据类型	允许的内存区
IN	输入	BYTE	I、Q、W
OUT	输出	BYTE	I、Q、W

IN: 字符串首地址。

OUT: 要附加到的字符串首地址。

使用举例

D		<p>当 IX10.0 为 1 时, SCAT 指令执行, 将 QB1 开始的字符串复制到 WB0 开始的字符串的尾部</p>											
L	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX10.0</td> </tr> <tr> <td></td> <td>2</td> <td>SCAT</td> <td>%QB1 %WB0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX10.0		2	SCAT
条	步	指令	操作数										
1	1	LD	%IX10.0										
	2	SCAT	%QB1 %WB0										

### SSCPY (从字符串复制子字符串)

功能说明

将一个字符串从指定位置开始的指定数目的字节复制到一个新字符串中。

指令及操作数说明

	名称	指令格式
LD	SSCPY	
IL	SSCPY	SSCPY IN,IDX,N,OUT

参数	输入/输出	数据类型	允许的内存区
IN	输入	BYTE	I、Q、W
IDX	输入	BYTE	I、Q、W、常量
N	输入	BYTE	I、Q、W、常量
OUT	输出	BYTE	I、Q、W

IN: 字符串首地址。

IDX: 字符串索引。

N: 要复制的字节数。

OUT: 要输出到的字符串的地址。

使用举例

D		<p>当 IX10.0 为 1 时, SSCPY 指令执行, 将 QB1 开始的字符串从第 3 个字节开始复制 3 个字节到 WB0 开始的字符串中</p>											
L	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">条</th> <th style="width: 10%;">步</th> <th style="width: 20%;">指令</th> <th style="width: 60%;">操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>%IX10.0</td> </tr> <tr> <td></td> <td>2</td> <td>SSCPY</td> <td>%QB1 3 3 %WB0</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX10.0		2	SSCPY
条	步	指令	操作数										
1	1	LD	%IX10.0										
	2	SSCPY	%QB1 3 3 %WB0										

### SFND (字符串查找)

功能说明

从第一个字符串的指定位置开始搜索第二个字符串出现的位置, 如果匹配到则将匹配到的字符串的第一个字节的位置返回到输出中, 匹配不到返回 0。

指令及操作数说明

	名称	指令格式
LD	SFND	
IL	SFND	SFND IN1, IN2, OUT

参数	输入/输出	数据类型	允许的内存区
IN1	输入	BYTE	I、Q、W
IN2	输入	BYTE	I、Q、W
OUT	输出	BYTE	I、Q、W

IN1: 字符串 1 首地址。

IN2: 字符串 2 首地址。

OUT: 要输出到的地址。

使用举例

D		<p>当 IX10.0 为 1 时，SFND 指令执行，将 QB0 开始的字符串从 QB5 指定的字节开始查找 WB0 相同的字符串，找到后将地址写到 QB5 中</p>											
L	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">条</th> <th style="width: 10%;">步</th> <th style="width: 20%;">指令</th> <th style="width: 60%;">操作数</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">LD</td> <td style="text-align: center;">%IX10.0</td> </tr> <tr> <td></td> <td style="text-align: center;">2</td> <td style="text-align: center;">SFND</td> <td style="text-align: center;">%QB0 %WB0 %QB5</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX10.0		2	SFND
条	步	指令	操作数										
1	1	LD	%IX10.0										
	2	SFND	%QB0 %WB0 %QB5										

### CFND（字符查找）

#### 功能说明

从第一个字符串的指定位置开始搜索第二个字符串中包含的字符集中任何一个字符第一次出现的位置，如果匹配到则将匹配到的字符的位置返回到输出中，匹配不到返回 0。

#### 指令及操作数说明

	名称	指令格式
LD	CFND	
IL	CFND	CFND IN1, IN2, OUT

参数	输入/输出	数据类型	允许的内存区
IN1	输入	BYTE	I、Q、W
IN2	输入	BYTE	I、Q、W
OUT	输出	BYTE	I、Q、W

IN1：字符串 1 首地址。

IN2：字符串 2 首地址。

OUT：要输出到的地址。

#### 使用举例

D		<p>当 IX10.0 为 1 时，CFND 指令执行，将 QB0 开始的字符串从 QB5 指定的字节开始查找 WB0 中字符串中包含的字符集中出现的任意字符第一次出现的位置，找到后将地址写到 QB5 中</p>											
L	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">条</th> <th style="width: 10%;">步</th> <th style="width: 20%;">指令</th> <th style="width: 60%;">操作数</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">LD</td> <td style="text-align: center;">%IX10.0</td> </tr> <tr> <td></td> <td style="text-align: center;">2</td> <td style="text-align: center;">CFND</td> <td style="text-align: center;">%QB0 %WB0 %QB5</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	%IX10.0		2	CFND
条	步	指令	操作数										
1	1	LD	%IX10.0										
	2	CFND	%QB0 %WB0 %QB5										

### 3.16 PID 指令

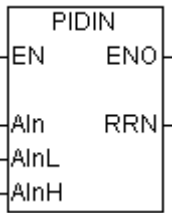
PID 指令由 PIDIN, PID 和 PIDOUT 三个指令块构成, 这三个指令都可以在中断程序中或者主程序中执行。

#### PIDIN (PIN 输入转换)

功能说明

将传感器采样值转换成实数, 再归一化为 0.0~1.0 之间的数。

指令及操作数说明

	名称	指令格式
LD	PIDIN	
IL	PIDIN	PIDIN AIn, AInL, AInH, RRN

参数	输入/输出	数据类型	允许的内存区
AIn	输入	INT	I、Q、W
AInL	输入	INT	I、Q、W
AInH	输入	INT	I、Q、W
RRN	输出	REAL	I、Q、W、D

AIn: 模拟量输入参数, 一般为一个 16 比特的有符号数输入。

AInL: 模拟量输入低值, 16 比特的有符号数。

AInH: 模拟量输入高值, 16 比特的有符号数。

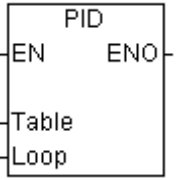
RRN: 模拟量输入标准化以后的输出, 0.0~1.0 的浮点数, 32 比特。

#### PID (比例积分微分)

功能说明

根据 PID 参数表的内容进行 PID 运算, 修改输出值, 同时也会修改积分值。

指令及操作数说明

	名称	指令格式
LD	PID	
IL	PID	PID Table, Loop

参数	输入/输出	数据类型	允许的内存区
Table	输入	INT	I、Q、W、D
Loop	输入	INT	I、Q、W、D、常量

Table: PID 参数所在的表格起始地址，可以是 I、Q、W、D 区。

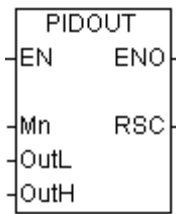
Loop: 常数，0~3，表示 0-3 个环路。

### PIDOUT (PID 输出转换)

功能说明

根据 PID 参数表的内容进行 PID 运算，修改输出值，同时也会修改积分值。

指令及操作数说明

	名称	指令格式
LD	PIDOUT	
IL	PIDOUT	PIDOUT Mn, OutL, OutH, RSC

参数	输入/输出	数据类型	允许的内存区
Mn	输入	REAL	I、Q、W、D、常量
OutL	输入	INT	I、Q、W、D、常量
OutH	输入	INT	I、Q、W、D、常量
RSC	输出	INT	I、Q、W、D

Mn: 0.0~1.0 之间的一个浮点数，32 比特。

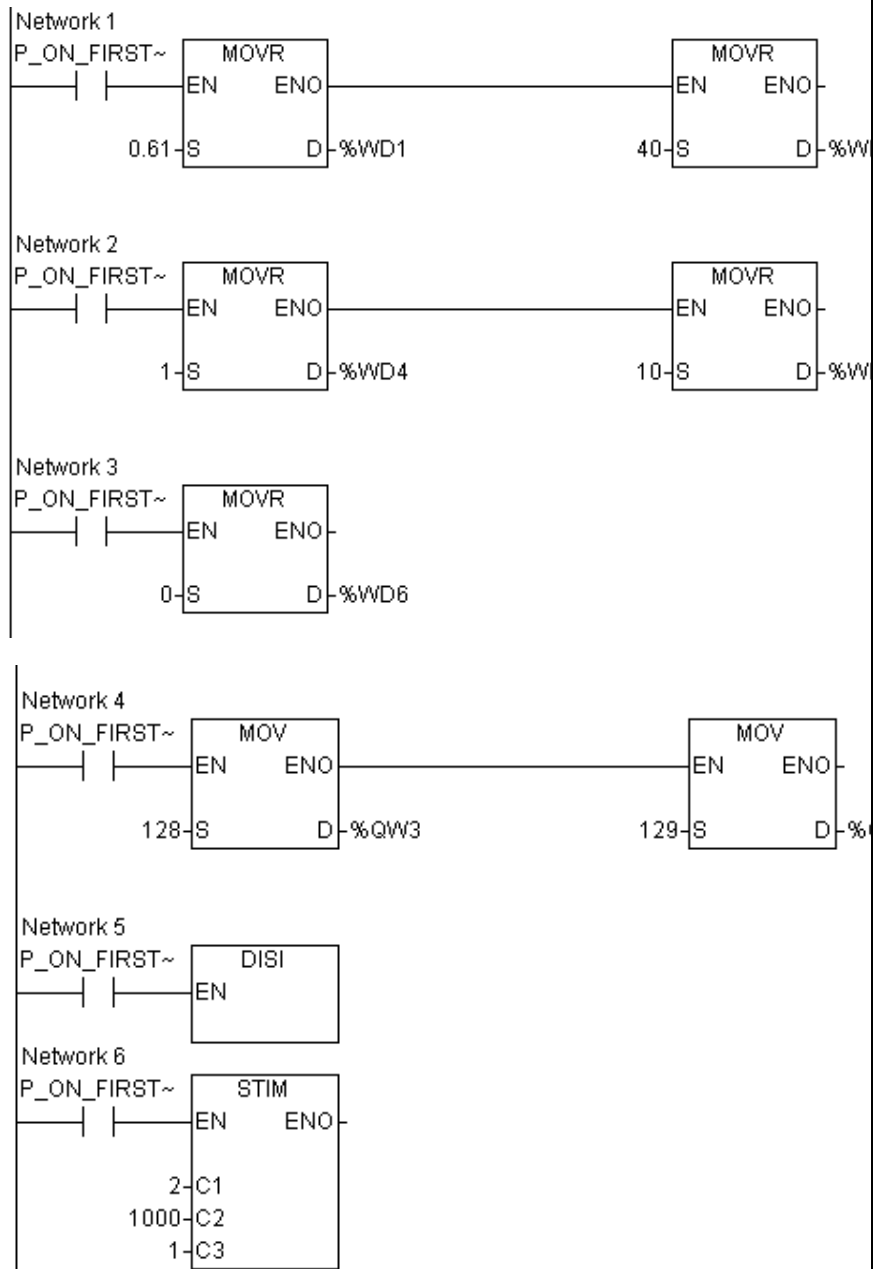
OutL: 模拟量输出低值，16 比特有符号数。

OutH: 模拟量输出高值，16 比特有符号数。

RSC: 16 比特的有符号数，回路输出的数值。

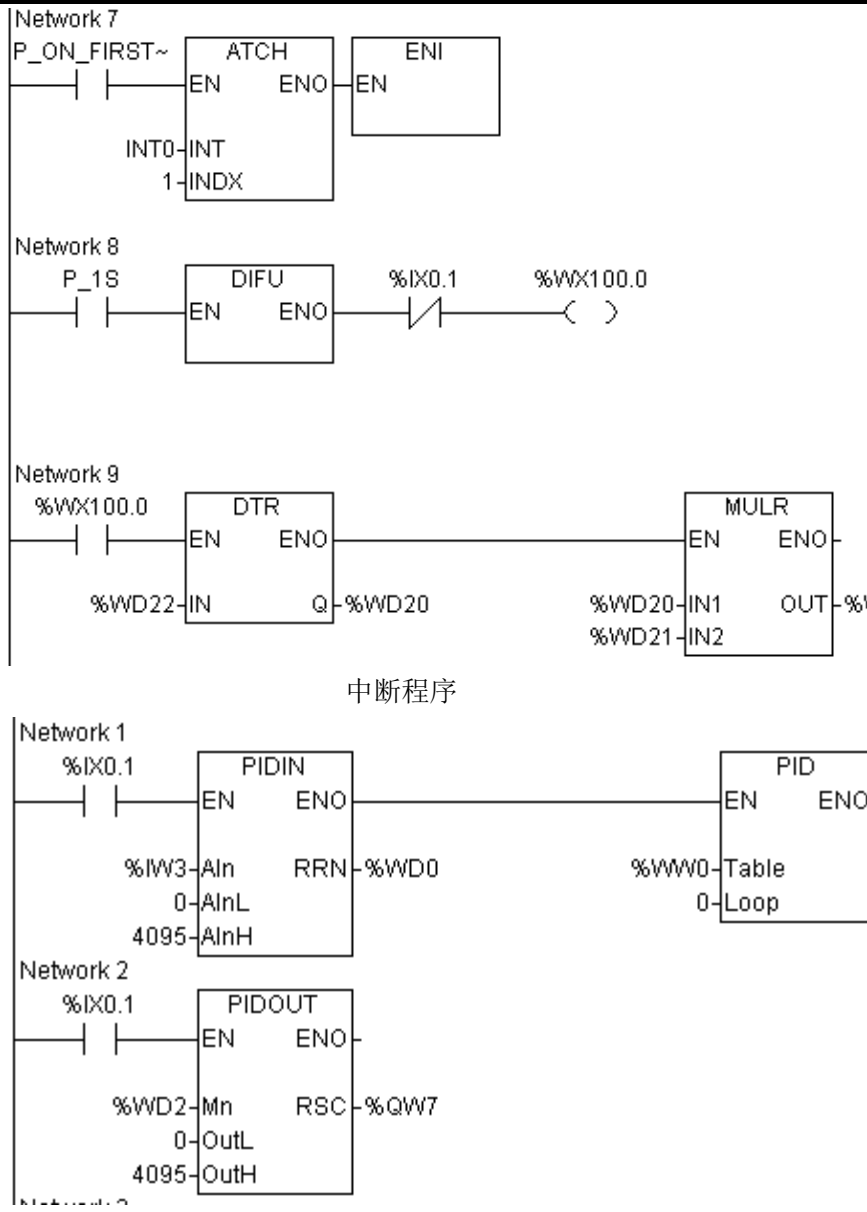
使用举例

主程序



给定值 0.61, P 为 40, I 为 10 分钟, D 为 0, 中断执行时间为 1 秒, 即采样时间为 1 秒, PID 指令每 1 秒被执行一次, PID 计算的输出也是每 1 秒被刷新一次。

D





主程序			
条	步	指令	操作数
1	1	LD	P_ON_FIRST_CYC
	2	MOVR	0.61
	3	MOVR	%WD1 40 %WD3
2	4	LD	P_ON_FIRST_CYC
	5	MOVR	1
	6	MOVR	%WD4 10 %WD5
3	7	LD	P_ON_FIRST_CYC
	8	MOVR	0 %WD6
4	9	LD	P_ON_FIRST_CYC
	10	MOV	128 %QW3
	11	MOV	129 %QW5
5	12	LD	P_ON_FIRST_CYC
	13	DISI	
6	14	LD	P_ON_FIRST_CYC
	15	STIM	2 1000 1
7	16	LD	P_ON_FIRST_CYC
	17	ATCH	INTO 1
8	18	ENI	
	19	LD	P_1S
	20	DIFU	
	21	ANDN	%IX0.1
	22	ST	%WX100.0
9	23	LD	%WX100.0
	24	DTR	%WD22 %WD20
	25	MULR	%WD20 %WD21 %WD2
中断程序			
条	步	指令	操作数
1	1	LD	%IX0.1
	2	PIDIN	%IW3 0 4095 %WD0
	3	PID	%WW0 0
2	4	LD	%IX0.1
	5	PIDOUT	%WD2 0 4095 %QW7

### 3.17 实时时钟指令


实时时钟的数据存储结构：实时时钟数据占4个字，第一个字的高字节为分钟值，低字节为秒值，第二个字的高字节为日期，低字节为小时，第三个字节的高字节为年，低字节为月，第四个字的高字节未定义，低字节为星期。

## TODR (读实时时钟)

功能说明

读实时时钟到指定地址，实时时钟的数据占4个字。

指令及操作数说明

	名称	指令格式
LD	TODR	
IL	TODR	TODR D

参数	输入/输出	数据类型	允许的内存区
D	输出	WORD	I、Q、W、D

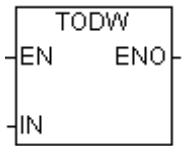
D: 实时时钟写入的地址。

## TODW (写实时时钟)

功能说明

将指定地址读取到的时钟数据设置到实时时钟，实时时钟的数据占4个字。

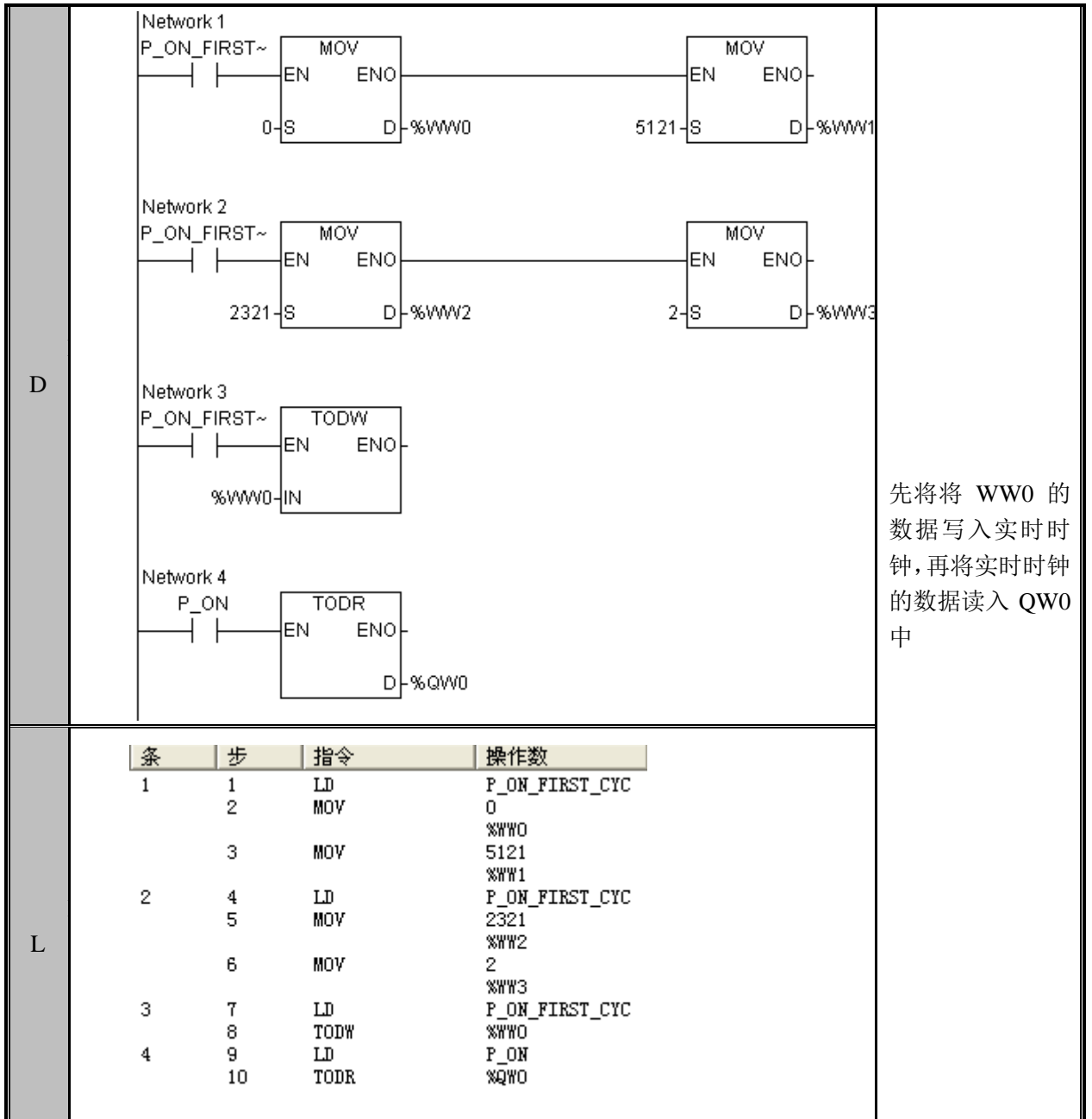
指令及操作数说明

	名称	指令格式
LD	TODW	
IL	TODW	TODW D

参数	输入/输出	数据类型	允许的内存区
IN	输入	WORD	I、Q、W、D

D: 实时时钟读入的地址。

使用举例



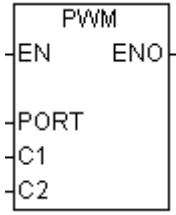
### 3.18 脉冲输出指令

#### PWM（脉宽调制输出）

##### 功能说明

从指定的端口输出频率和占空比可调的脉冲。

##### 指令及操作数说明

	名称	指令格式
LD	PWM	
IL	PWM	PWM PORT, C1, C2

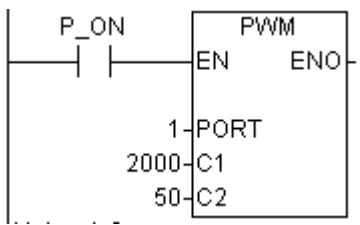
参数	输入/输出	数据类型	允许的内存区
PORT	输入	WORD	I、Q、W、D、常量
C1	输入	WORD	I、Q、W、D、常量
C2	输入	WORD	I、Q、W、D、常量

PORT: 指定 PWM 输出的端口, 取值范围是 1 到 6, 端口 1 到 6 分别对应%QX0.2 到%QX0.7。

C1: 指定输出的频率, 单位是 Hz, 取值范围从 1 到 25000。

C2: 指定输出的占空比, 范围 0 到 100, 0 就是全部低电平, 100 是全部高电平。

使用举例

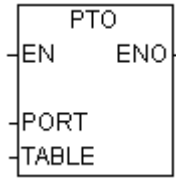
LD		从端口 1 输出 2k 赫兹, 占空比为 50 的脉冲											
IL	<table border="1"> <thead> <tr> <th>条</th> <th>步</th> <th>指令</th> <th>操作数</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>LD</td> <td>P_ON</td> </tr> <tr> <td></td> <td>2</td> <td>PWM</td> <td>1 2000 50</td> </tr> </tbody> </table>		条	步	指令	操作数	1	1	LD	P_ON		2	PWM
条	步	指令	操作数										
1	1	LD	P_ON										
	2	PWM	1 2000 50										

### PTO (脉冲连输出)

功能说明

从指定的端口输出指定个数频率可调的脉冲。

指令及操作数说明

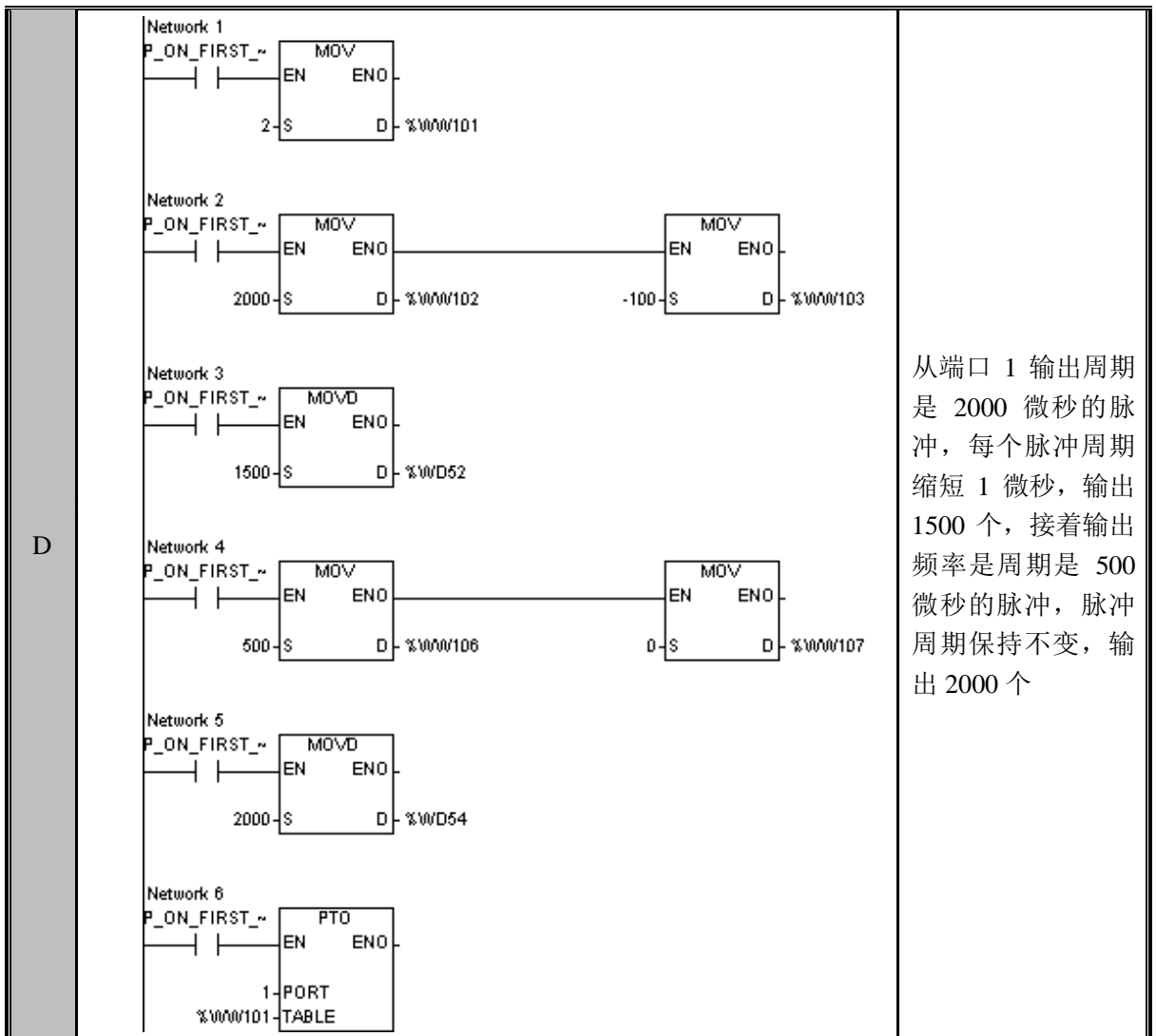
	名称	指令格式
LD	PTO	
IL	PTO	PTO PORT, TABLE

参数	输入/输出	数据类型	允许的内存区
PORT	输入	WORD	I、Q、W、D、常量
TABLE	输入	WORD	I、Q、W、D

**PORT:** 指定 PTO 输出的端口。取值范围是 1 到 6，端口 1 到 6 分别对应%QX0.2 到%QX0.7，其中端口 1、2、3、4 为各自独立输出；1、5、6 只能以相同的频率输出。

**TABLE:** 输出定义表。表格格式是：第一个字指定脉冲链个数，接下来每 4 个字为一个单位定义每个脉冲链的参数，第一个字指定脉冲起始周期，单位是微秒，第二个字节定义每个脉冲改变的脉冲周期，单位是 0.01 微秒，第三个双字定义要输出的脉冲个数，由于存在字节对齐问题，所以表格的头一个地址请使用一个奇数地址。

使用举例



	条	步	指令	操作数
L	1	1	LD	P_ON_FIRST_CYC
		2	MOV	2 %WW101
	2	3	LD	P_ON_FIRST_CYC
		4	MOV	2000 %WW102
		5	MOV	-100 %WW103
	3	6	LD	P_ON_FIRST_CYC
		7	MOVD	1500 %WD52
	4	8	LD	P_ON_FIRST_CYC
		9	MOV	500 %WW106
		10	MOV	0 %WW107
	5	11	LD	P_ON_FIRST_CYC
		12	MOVD	2000 %WD54
	6	13	LD	P_ON_FIRST_CYC
		14	PTO	1 %WW101


**注意：**当使用外部中断功能时 PTO 输出端口 2、3、4（对应%QX0.3，%QX0.4，%QX0.5）无法正常输出。

#### PWMSTOP (PWM 输出停止)

功能说明

停止指定端口的 PWM 脉冲输出。

指令及操作数说明

	名称	指令格式
LD	PWMSTOP	
IL	PWMSTOP	PWMSTOP PORT

参数	输入/输出	数据类型	允许的内存区
PORT	输入	WORD	常量 1 到 6

PORT: 指定 PWM 输出的端口，取值范围是 1 到 6，端口 1 到 6 分别对应%QX0.2 到%QX0.7。

**注意：**当 PORT 的参数范围不在 1 到 6 之间时，将关闭所有的 PWM 输出。

使用举例

D		当 IX0.0 的输入为 ON 时关闭端口 1 (QX0.2) 的 PWM 输出												
L	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">条</th> <th style="width: 10%;">步</th> <th style="width: 30%;">指令</th> <th style="width: 50%;">操作数</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">LD</td> <td style="text-align: center;">%IX0.0</td> </tr> <tr> <td></td> <td style="text-align: center;">2</td> <td style="text-align: center;">PWMSTOP</td> <td style="text-align: center;">1</td> </tr> </tbody> </table>	条	步	指令	操作数	1	1	LD	%IX0.0		2	PWMSTOP	1	
条	步	指令	操作数											
1	1	LD	%IX0.0											
	2	PWMSTOP	1											

### PLSSTOP (PTO 输出停止)

功能说明

停止指定端口的 PTO 脉冲输出。

指令及操作数说明

	名称	指令格式
LD	PLSSTOP	
IL	PLSSTOP	PLSSTOP PORT

参数	输入/输出	数据类型	允许的内存区
PORT	输入	WORD	常量 1 到 3

PORT: 指定 PTO 输出的端口, 取值范围是 1 到 3, 端口 1 到 3 分别对应%QX0.2 到%QX0.4。

**注意:** 当 PORT 的参数范围不在 1 到 3 之间时 PLC 将会提示运行错误。

使用举例

D		当 IX0.0 的输入为 ON 时关闭端口 1 (QX0.2) 的 PTO 输出												
L	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">条</th> <th style="width: 10%;">步</th> <th style="width: 30%;">指令</th> <th style="width: 50%;">操作数</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">LD</td> <td style="text-align: center;">%IX0.0</td> </tr> <tr> <td></td> <td style="text-align: center;">2</td> <td style="text-align: center;">PLSSTOP</td> <td style="text-align: center;">1</td> </tr> </tbody> </table>	条	步	指令	操作数	1	1	LD	%IX0.0		2	PLSSTOP	1	
条	步	指令	操作数											
1	1	LD	%IX0.0											
	2	PLSSTOP	1											

# 第 4 章 使用中断功能

## 4.1 中断程序中中断号的分配

中断号的分配使用：指令块 ATCH

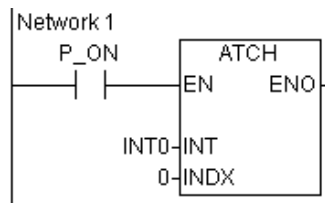


图 3.25 中断程序中中断号分配

如图 3.25，将中断号 0 分配给了中断程序 INT0。

## 4.2 中断程序中中断号的收回

中断号的收回使用：指令块 DTCH

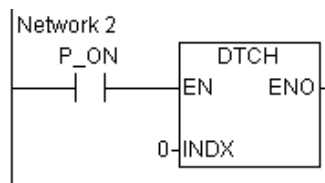


图 3.26 中断程序中中断号收回

如图 3.26，将中断号 0 收回。

## 4.3 中断的使能

中断的使能使用：指令块 ENI

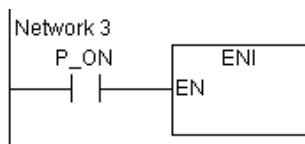


图 3.27 中断使能

## 4.4 中断的取消使能

中断的禁能使用：指令块 DISI



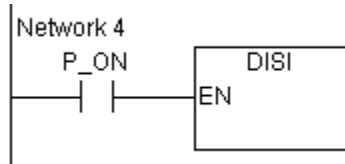


图 3.28 中断禁能

## 4.5 时钟中断的使用

请参见附录“指令块”中“STIM”指令块的使用。

## 4.6 中断事件与中断号分配表

中断事件	中断号	说明
定时器中断	1	定时器中断对应单次中断和周期性中断两种情况
ENT1 下降沿输入中断	2	IX0.3 作为输入
ENT1 上升沿输入中断	3	IX0.3 作为输入
ENT2 下降沿输入中断	4	IX0.4 作为输入
ENT2 上升沿输入中断	5	IX0.4 作为输入
ENT1 上升沿递增计数	6	IX0.3 作为输入
ENT1 下降沿递增计数	7	IX0.3 作为输入
ENT2 上升沿递增计数	8	IX0.4 作为输入
ENT2 下降沿递增计数	9	IX0.4 作为输入
ENT1 上升沿递减计数	10	IX0.3 作为输入
ENT1 下降沿递减计数	11	IX0.3 作为输入
ENT2 上升沿递减计数	12	IX0.4 作为输入
ENT2 下降沿递减计数	13	IX0.4 作为输入
CAP1 上升沿递增高速计数	14	IX0.2 作为输入
CAP1 脉冲加方向高速计数	15	IX0.2 作为脉冲, IX0.3 作为方向
CAP1、CAP2 增/减高速计数	16	IX0.2 作为递增, IX0.1 作为递减
IX0.5 下降沿输入中断	80	
IX0.5 上升沿输入中断	81	
IX0.0 下降沿输入中断	82	
IX0.0 上升沿输入中断	83	
IX0.1 下降沿输入中断	84	
IX0.1 上升沿输入中断	85	
IX0.2 下降沿输入中断	86	
IX0.2 上升沿输入中断	87	

# 第 5 章 使用 PLC 串行通信口

VPC1 系列 PLC (以下简称 PC) 的串行口通信方式分 5 种: 上位机链接通信 (编程口)、PC-Net 自组网通信、用户定义协议的自由口通信、Modbus-RTU 从站的通信、Modbus-RTU 主站的通讯。

PLC 有两个串行通信口: 串口 0 和串口 1。在物理上, 串口 0 为 RS-232C 口, 串口 1 为 RS-485 口, 除了物理接口不同, 两个串口支持的软件功能都相同, 对每种通信方式均支持, 见表 7-1:

表 7-1 通信方式概览

通信端口 通信方式	Port0 (RS232)	Port1 (RS485)
上位机链接	√	√
PC-Net	√	√
自由口	√	√
Modbus-RTU 从站	√	√
Modbus-RTU 主站	√	√

各串口当前通信方式由 PC 的工作模式 (编程模式还是运行模式) 及系统寄存器中的串口设置寄存器决定。当 PC 处于运行模式时, 串口按照寄存器的配置进行工作。当 PC 处于编程模式时, RS232 口将固定为上位机链接方式, 通信格式为: 115200bps, 8, N, 1。RS485 口的工作方式不受 PC 的模式的影响。

VPC1 系列 PLC 出厂时, RS232 口默认为上位机链接方式使用, 通信格式为: 115200bps, 8, N, 1。RS485 口默认为 Modbus-RTU 从站通信方式, 站地址为 4, 通信格式为: 19200bps, 8, N, 1。

在 PC 程序运行中, 每次都会判断串口工作配置字是否改变, 若有改变, 则重新配置其工作方式。

串口的配置具有断电保存的功能。上电时串口的工作方式将配置成和上次一样。

串口 0 的主配置字为系统寄存器 MW36; 串口 1 的主配置字为系统寄存器 MW37。在不同的通信方式下, 还可能会有进一步的配置要求。

## 5.1. 上位机链接通信

VPC1 可以通过 RS-232C/RS-485 口和上位机编程软件进行连接。在上位机链接通信模式下, 可对 VPC1 进行编程, 配置, 在线调试, 监控等操作。

推荐使用 RS-232C 口与上位机连接。

配置方式: 将寄存器 MW36 (或 MW37) 配置成上位机链接方式即可。

**注意:** 以下不作特殊说明时, 仅以串口 0 为例。串口 1 的情况类似。

特殊说明: 当 PC 处于编程模式时, 串口 0 将固定为上位机链接通信方式, 通信参数为: 115200bps, 8, N, 1。

## 5.2. PC-Net 自组网通信

### 5.2.1 PC-Net 自组网通信概述

PC-Net 自组网通信用于 PLC 之间构成的主从通信网络，自定义内部的通信协议。该网络支持两种形式的通信，一是 PLC 通过网络交换数据，以公共数据区的形式实现相互间的信息共享，实现这种功能，只需配置即可，称为 PC-Link 方式（简记为 PL）；另一种是对网络指令的支持（简记为 NI）。

网络中任何形式的通信都是由主站发起的，从站只是响应主站的请求，一个网络中只允许有一个主站。在进行网络通信前，应先配置各站点的主从模式，并配置各个从站的地址（一般通过用户程序进行配置，设置配置寄存器即可）。

#### PC-Link 方式描述

PC-Link 方式下，配置有一个公共的数据区，该数据区对网络中每个站点都是一样的。其中对应每一个站点（包括主站在内）都有一个属于自己的数据区域，每个 PLC 负责对自己的数据区的维护。整个公共数据区中除了自己的数据区以外，其余的部分则分别是网络中其它站点的数据区的映射区。用户程序中对其它站点的数据映射区，只能访问，不能修改。网络中各个单元的数据流关系如图所示。通过网络中周期性的数据交换实现公共数据区的一致。

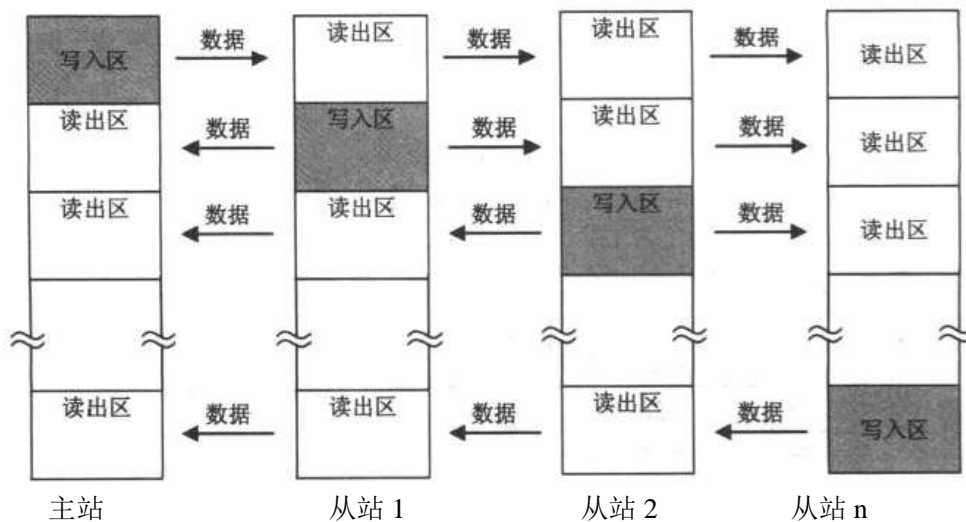


图 7.1 PC-Link 方式示意图

PC-Link 系统数据交换采用自动轮询的方式，它不需要编程，只对系统进行配置，数据交换便可以自动完成。数据的交换是由主站发起的。对数据区的配置由网络中的主站统一进行。整个公共数据区必须是连续的。

#### 网络指令方式描述

网络指令用于 PC Net 主站访问该网络中的从站，包括网络读（NETR）和网络写（NETW），由主站发起请求，等待从站的响应。该部分要配合指令一起完成。

为便于理解，可以把 PC-Link 方式看成周期性的网络数据交换，把网络指令方式看成非周期性的网络数据交换。

### 5.2.2 用户配置说明

配置寄存器：M 区第 50~87 字。

默认情况下，控制器是作为 PC Net 从站的。要构成 PC Net 网络时，需将其中的一个设备配置成主站（对于串口 0，在该设备的用户程序中修改其 PC 网络主配置字 %MW50）。如寄存器的说明所示，主站总是支持网络指令通信的，可以选择主站支持或不支持 PC-Link 通信。在网络中，只能存在 1 个主站。

若是配置成支持 PC-Link 通信的主站，则还应对整个 PC-Link 网络进行配置。这包括 PC-Link 从站数量、PC-Link 数据域起始地址、各 PC-Link 从站的地址及各站点 PC-Link 数据区大小。（对于串口 0，对应配置寄存器中从 %MW52 到 %MW68 的区域）

若是配置成仅支持网络指令的主站，则不需其它的配置。

若是配置成 PC Net 从站，则还需配置该从站的地址。（对于串口 0，对应寄存器 %MW51）

**注意：**PC NET 网络允许的最大节点数为 32（即最多 31 个从站），但其中 PC-Link 从站个数最多为 7 个。可以把 PC-Link 网络看成 PC NET 网络的子集。

通信口 0 为 RS-232 口，不支持直接的多机（大于 2）互连，若要利用该通信口组成多机网络，需进行电平转换，使之转换成 RS-485 口。

### 5.2.3 出错寄存器说明

出错寄存器：M 区第 42~49 字。

出现相关错误时，将被记录到上述对应的寄存器，并导致出错灯闪烁。若系统检测到出错消失，将清除相应的记录，停止出错灯闪烁。

### 5.2.4 PC-Link 说明

规定：PC-Link 通信只能支持对 PLC 的通用数据寄存器区的读写。

(1) PC-Link 通信时间估算：

本 PLC 允许的 PC-Link 网络最大节点数为 8，公共数据区最大为 32 个字节。下表给出了在最大配置下的仅仅传输字符的时间的估算。

网络节点数	公共数据区字节数	通信字符数	计入帧间隔后的 大约字符时间	典型波特率下的通信时间 (ms)			
				9600	19200	38400	115200
8	32	155	250 字符	300	150	75	25

由于系统的处理时间及延迟，实际一个 PC-Link 周期过程的时间还要稍长。

(2) PC-Link 级连

两个串口的工作是独立的，其网络关系也是各自独立的，可以分别地工作在 PC-Link 方式。利用这一点可以构成 PC-Link 级连网络，例如将一个串口设为 PC-Link 主站，而将另一个串口设为从站，所以该 PLC 在前一串口的网络中是作为主站，而在后一串口的网络中是作为从站，于是就构成了级连的网络。依此，可以构成更多极的网络，网络配置具有充分的自由度。

该级连网络实际上是属于 PC Net 的级连网络，PC-Link 工作方式只是其中的一部分，所以这种级连关系对于网络指令也是适用的。

**注意：**若两个串口都用做 PC-Link 模式时，要考虑到配置的数据区域的不冲突。

### 5.2.5 网络指令说明

网络指令用于与 PC 网络中的其它单元进行通信。由 PC-Net 主站通过网络读 (NETR) 或网络写 (NETW) 的指令向网络中的从站发出数据访问请求, 等待从站的响应。这两条指令分别支持对网络中从设备特定区域的读取和写入。

指令的说明及使用参考指令说明手册。

只有网络指令通信请求允许标志使能时, 才能执行网络指令。

串口 0 相关标志位为 %SX42; 串口 1 相关标志位为 %SX47。

#### (1) 网络读 (NETR)

网络读的指令表格式为: NETR S, D, C

其中: S 表示源起始字; D 表示目标起始字; C 表示第一控制字。

指令控制字总共包含三个字的数据, 定义如下

字	位 00~07	位 08~15
C	串口号 (0 或 1)	从站设备地址 (1-31)
C+1	访问的数据字节数 (1-255)	
C+2	网络指令状态字	

网络指令状态字是网络指令控制字的一部分, 用来记录网络指令的执行状况, 便于用户对网络指令及其通讯过程进行掌控。其定义如下:

MSB						LSB														
5	4	3	2	1	0															
出错应答帧的错误代码 (Exception Code)						错误代码														

D 完成 (请求已完成): 0 = 没有完成, 1 = 完成

A 激活 (请求已排队): 0 = 没有激活, 1 = 激活

E 出错 (指令执行出错): 0 = 没有出错, 1 = 出错

错误代码定义如下:

代码	定义
0	无错
1	串口工作方式错误: 不是网络通讯方式
2	不是网络通讯主站
3	网络指令请求队列溢出
4	参数错误: 非法的串口号或从设备地址
5	访问越界: 错误的数据地址或数据长度
6	超时出错: 远程站没有应答
7	出错应答帧: 错误代码见 Exception Code 部分
8-F	未用 (保留)

#### (2) 网络写 (NETW)

网络写的指令格式为 NETW S, D, C

其中: S 表示源起始字; D 表示目标起始字; C 表示第一控制字。

指令控制字总共包含三个字的数据, 定义如下:

字	位 00~07	位 08~15
C	串口号 (0 或 1)	从站设备地址 (1-31)
C+1	访问的数据字节数 (1-255)	
C+2	网络指令状态字	

网络指令状态字的定义与 NETR 相同。

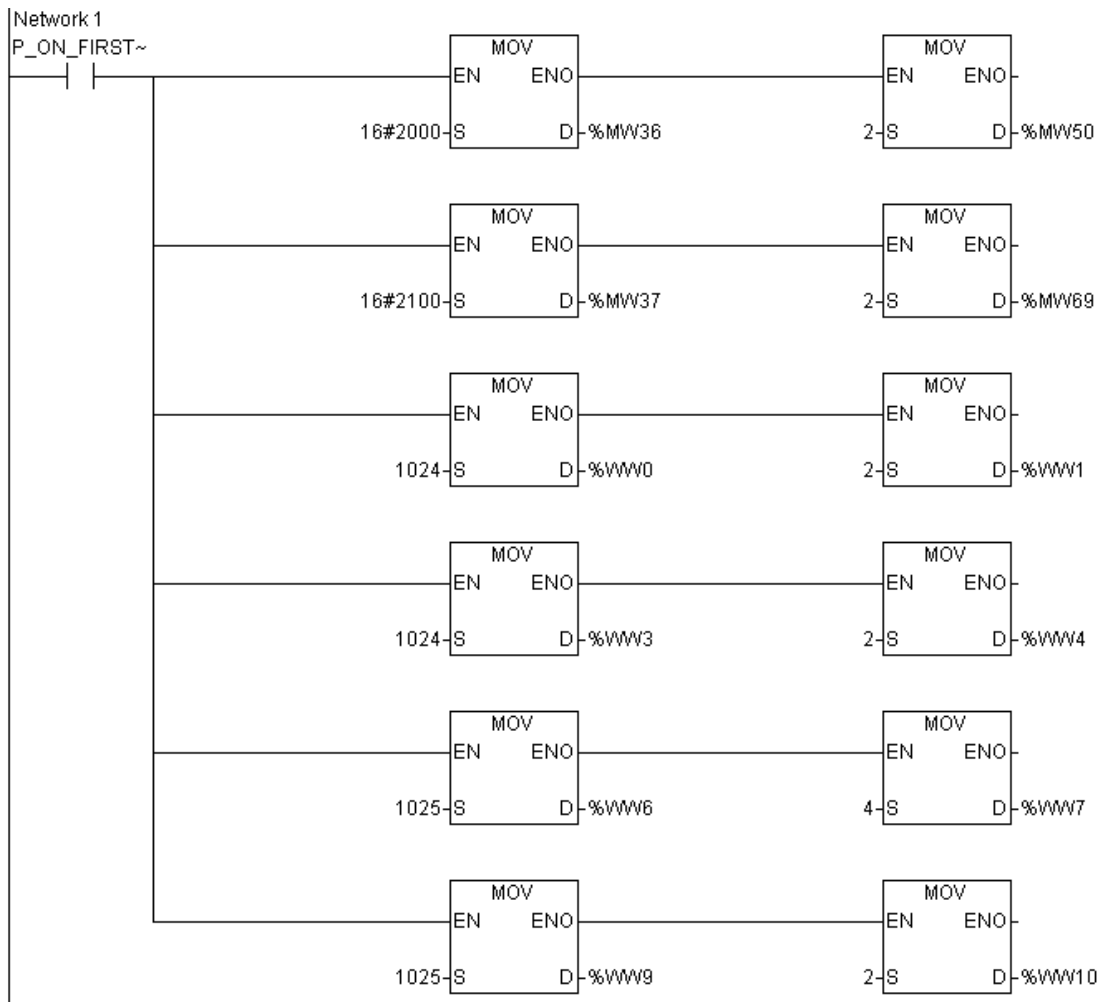
### 5.2.6 实例说明

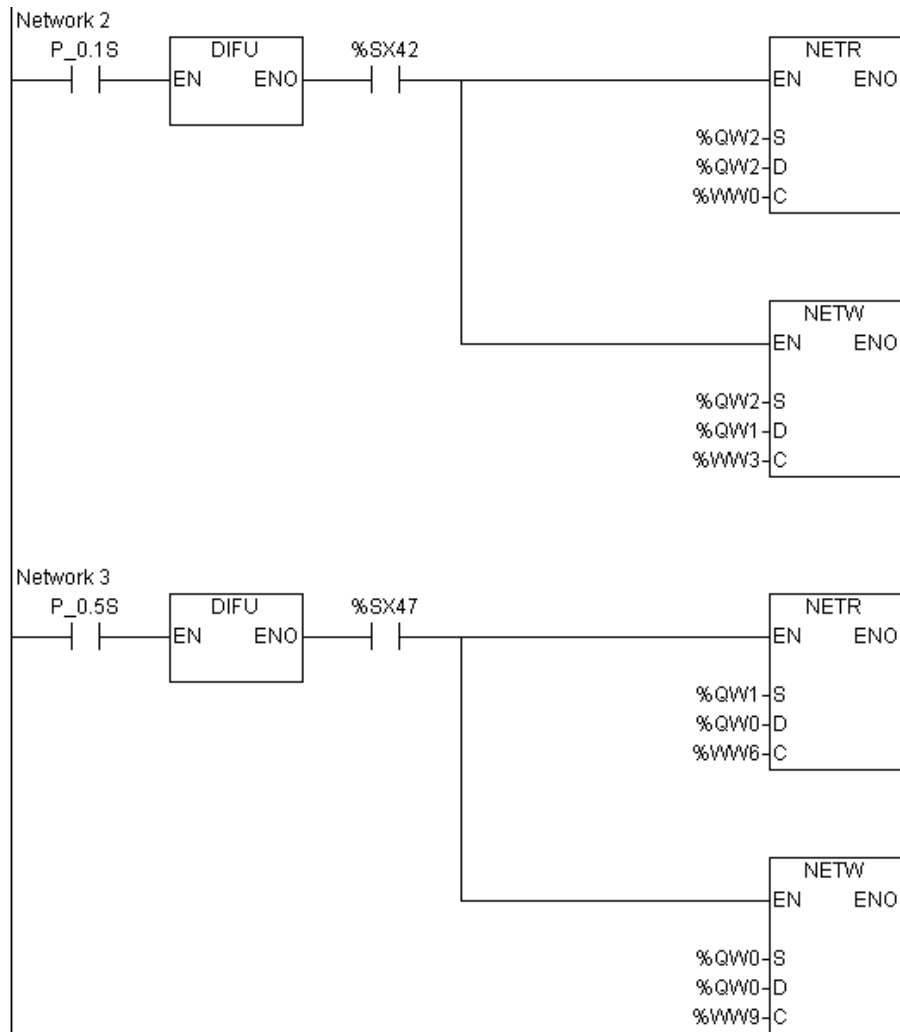
#### (1) 网络指令应用实例

实现的目的：从机的 QW2 执行自加的指令，主机利用两个串口的网络指令读和写，实现主站、从站的 QW0、QW1、QW2 与从站的 QW2 同步，通过 LED 显示出来。

i、主机的程序：

梯形图程序如下：





对应的指令表程序如下：

Network1

```

LD P_ON_FISRT_CYC //在上电第一次扫描时：
MOV #8192,%MW36 //初始化串口 0 工作方式：
//串口 0 工作在 PC Net 网络通信方式，
//波特率 115200，8 个数据位，无奇偶校验。
MOV #2,%MW50 //配置串口 0 的 PN 方式：为仅支持网络指令的主站
MOV #8448,%MW37 //初始化串口 1 工作方式：
//串口 0 工作在 PC Net 网络通信方式，
//波特率 57600，8 个数据位，无奇偶校验。
MOV #2,%MW69 //配置串口 1 的 PN 方式：为仅支持网络指令的主站
//初始化网络指令控制字，为后面使用网络指令做准备：
MOV #1024,%WW0 //选择串口号 0，欲访问的从站地址为 4（见网络指令说明）
MOV #2,%WW1 //设定访问的字节数：2 个字节

MOV #1024,%WW3 //选择串口号 0，欲访问的从站地址为 4
MOV #2,%WW4 //设定访问的字节数：2 个字节

MOV #1025,%WW6 //选择串口号 1，欲访问的从站地址为 4

```

```

MOV #4,%WW7 //设定访问的字节数：4 个字节

MOV #1025,%WW9 //选择串口号 1，欲访问的从站地址为 4
MOV #2,%WW10 //设定访问的字节数：2 个字节

Network2 //执行网络指令的通信
LD P_0.1S
DIFU //在 0.1s 时钟的上升沿，即每 0.2s 时：
AND %SX42 //若通信口 0 允许添加网络指令：
NETR %QW2, %QW2, %WW0 //发出网络读的请求，控制字为由 WW0 开始的 3 个字
//即用通信口 0 将 4 号从机的以 QW2 开始的 2 个字节
//的内容读到本机的以 QW2 开始的区域

NETW %QW2, %QW1, %WW3 //发出网络写的请求，控制字区域起始地址为 WW3
//用通信口 0 将本机的以 QW2 开始的 2 个字节的内容
//写入到 4 号从机的以 QW1 开始的区域

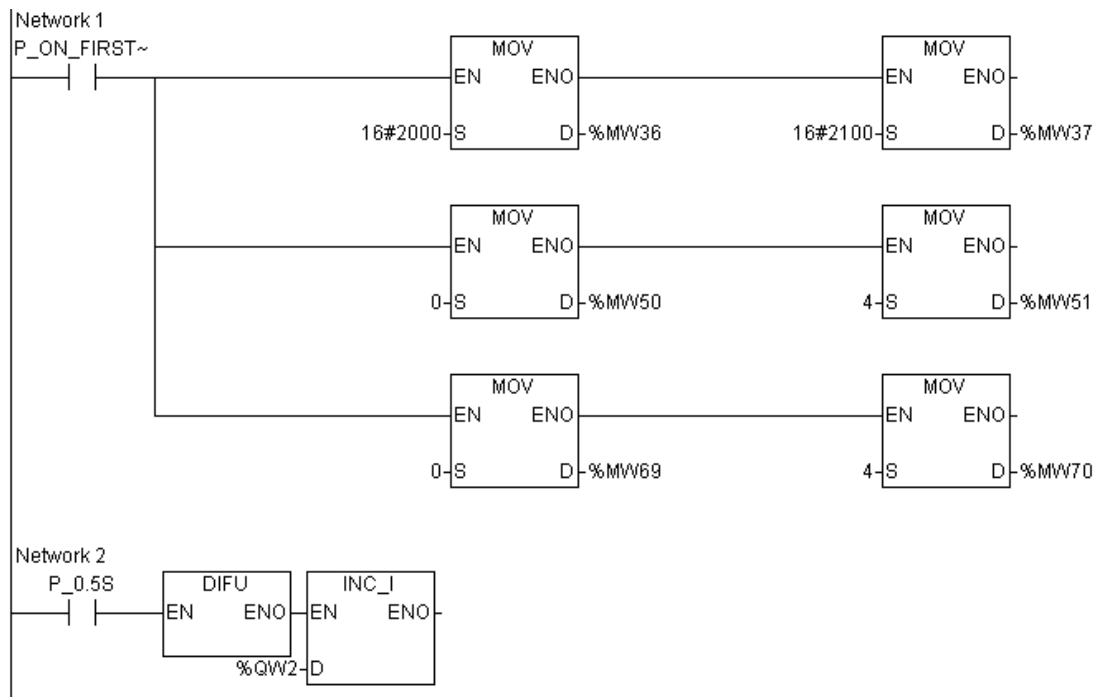
LD P_0.5S
DIFU //在 0.5s 时钟的上升沿，即每 1s 时：
AND %SX47 //若通信口 1 允许添加网络指令：
NETR %QW1, %QW0, %WW6 //发出网络读的请求，控制字区域起始地址为 WW6
//即用通信口 1 将 4 号从机的以 QW1 开始的 4 个字节
//的内容读到本机的以 QW0 开始的区域

NETW %QW0, %QW0, %WW9 //发出网络读的请求，控制字区域起始地址为 WW9
//即用通信口 1 将本机的以 QW0 开始的 2 个字节的内
//容写入到 4 号从机的以 QW0 开始的区域

```

ii、从机的程序  
梯形图程序如下：





对应的指令表程序如下：

```

Network1
LD P_ON_FISRT_CYC //在上电第一次扫描时：
MOV 16#2000,%MW36 //初始化串口 0 工作方式：
//串口 0 工作在 PC Net 网络通信方式，
//波特率 115200，8 个数据位，无奇偶校验。

MOV 16#2100,%MW37 //初始化串口 1 工作方式：
//串口 0 工作在 PC Net 网络通信方式，
//波特率 57600，8 个数据位，无奇偶校验。

MOV #0,%MW50 //配置串口 0 的 PN 方式：为 PN 从站
MOV #4,%MW51 //配置串口 0 的 PN 从站地址为 4

MOV #0,%MW69 //配置串口 1 的 PN 方式：为 PN 从站
MOV #4,%MW70 //配置串口 1 的 PN 从站地址为 4

Network2 //本机的 QW2 自加程序
LD P_0.5S
DIFU //在 0.5s 时钟的上升沿，即每 1s 时：
INC_I %QW2 //QW2 自加 1

```

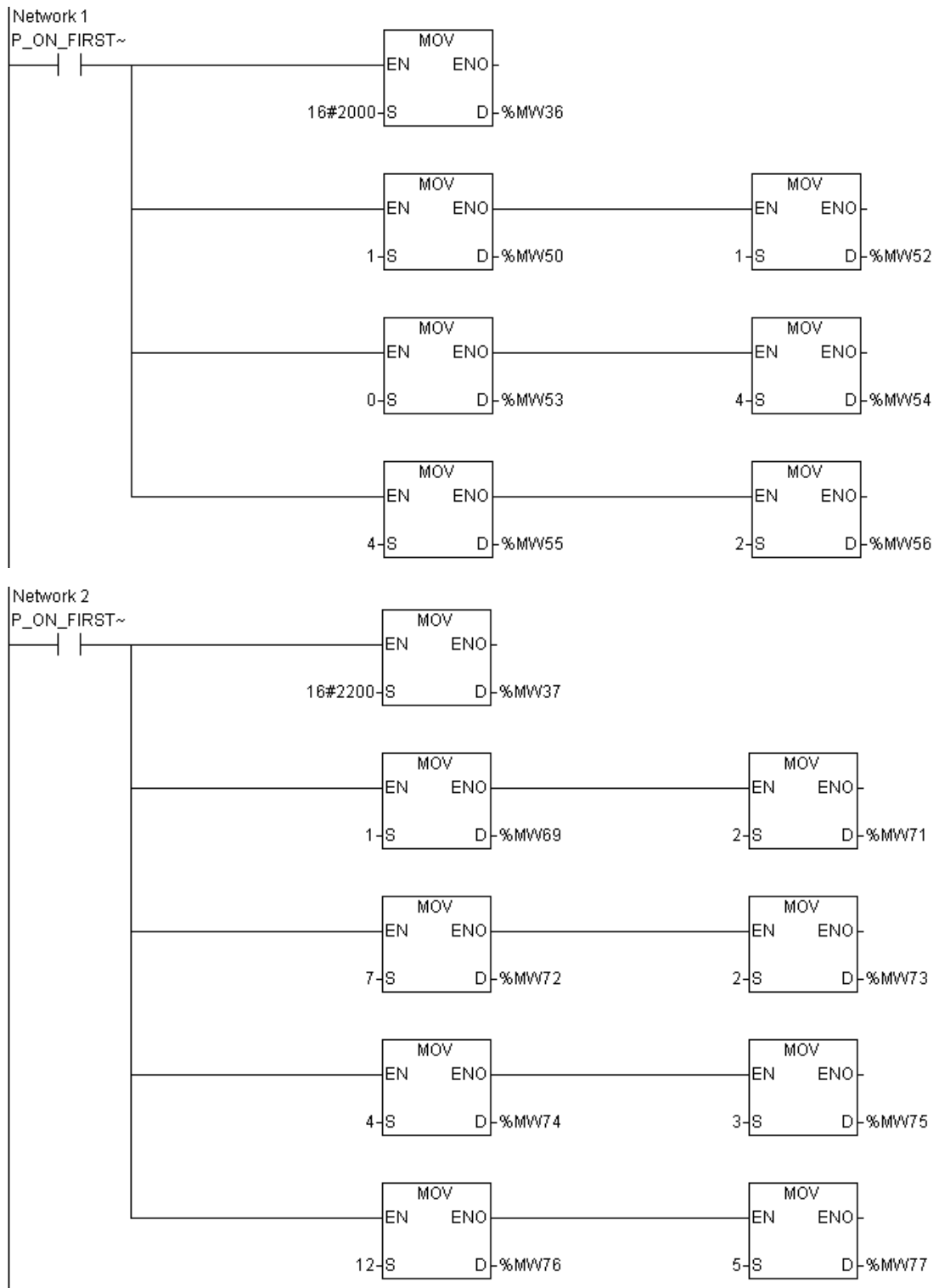
### 5.2.7 PC-Link 应用实例

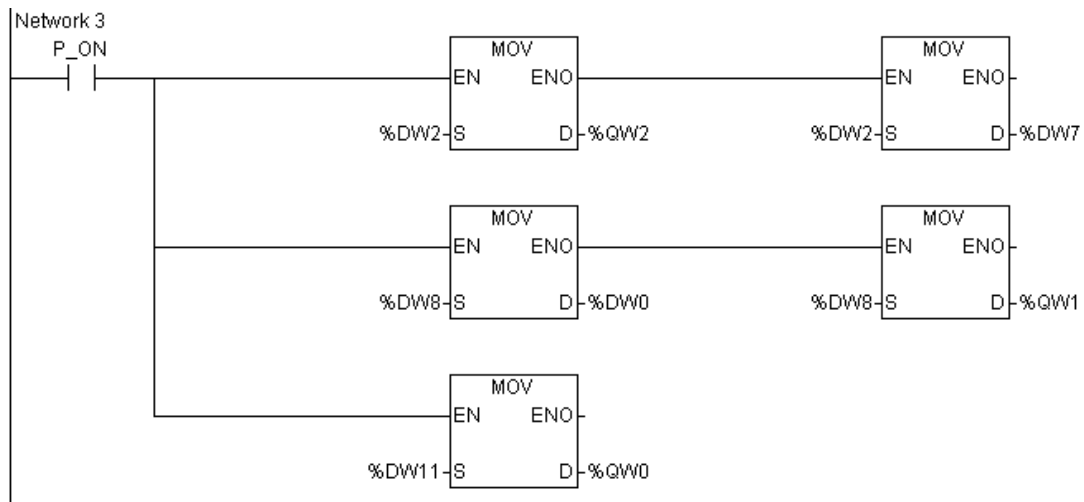
实现的目的：从站 1 的 QW3 执行自加的指令，主机利用两个串口分别组织两个 PL 网络（注意：通信口为 RS-232 口，只支持一对一直连，所以本例中该通信口的 PL 网络，即 PL0 网络只带有一个从站；另一个通信口为 RS-485 口，配置成带 2 个从站），利用 PL 的数

据交换及各自程序中的数据拷贝，实现主站和所有从站的 QW0、QW1、QW2 与从站 1 的 QW3 同步，通过 LED 显示出来。

### 5.2.7.1 主站的程序

梯形图程序如下：





对应的指令表程序如下：

Network1

```
LD P_ON_FISRT_CYC //在上电第一次扫描时：
MOV #8192,%MW36 //初始化串口 0 工作方式：
//串口 0 工作在 PC Net 网络通信方式，
//波特率 115200，8 个数据位，无奇偶校验。
MOV #1,%MW50 //配置串口 0 的 PN 方式：支持 PC-Link 和网络指令的主站
MOV #1,%MW52 //配置 PL 网络 0 的从站个数：1 个
MOV #0,%MW53 //配置 PL 数据区起始地址：DW0
MOV #4,%MW54 //配置 PL 数据区主站数据区大小：4 个字节
MOV #4,%MW55 //配置 PL 从站 1 地址：4
MOV #2,%MW56 //配置 PL 数据区从站 1 数据区大小：2 个字节
```

Network2

```
MOV #8704,%MW37 //初始化串口 1 工作方式：
//串口 1 工作在 PC Net 网络通信方式，
//波特率 38400，8 个数据位，无奇偶校验。
MOV #1,%MW69 //配置串口 1 的 PN 方式：支持 PC-Link 和网络指令的主站
MOV #2,%MW71 //配置 PL 网络 1 的从站个数：2 个
MOV #7,%MW72 //配置 PL 数据区起始地址：DW7
MOV #2,%MW73 //配置 PL 数据区主站数据区大小：2 个字节
MOV #4,%MW74 //配置 PL 从站 1 地址：4
MOV #3,%MW75 //配置 PL 数据区从站 1 数据区大小：3 个字节
MOV #12,%MW76 //配置 PL 从站 2 地址：4
MOV #5,%MW77 //配置 PL 数据区从站 2 数据区大小：5 个字节
```

Network3

```
LD P_ON //执行数据拷贝，将 PL 数据区中的数据拷到目标区
MOV %DW2,%QW2
MOV %DW2,%DW7
MOV %DW8,%DW0
```

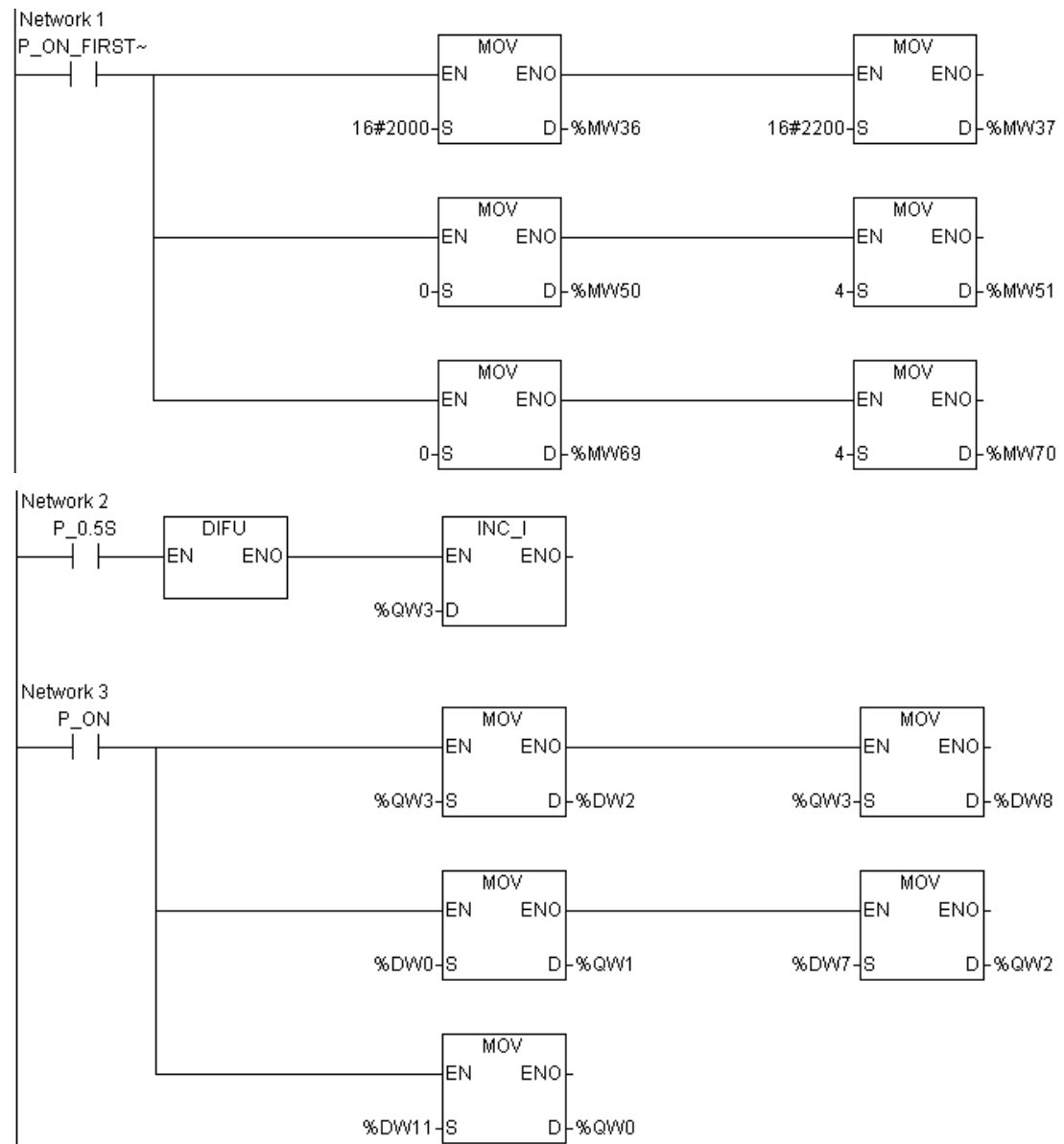
```

MOV %DW8,%QW1
MOV %DW11,%QW0

```

### 5.2.7.2 从站 1 的程序

梯形图程序如下：



对应的指令表程序如下：

```

Network1
LD P_ON_FISRT_CYC          //在上电第一次扫描时：
MOV #8192,%MW36           //初始化串口 0 工作方式：
                           //串口 0 工作在 PC Net 网络通信方式，
                           //波特率 115200，8 个数据位，无奇偶校验。

MOV #8704,%MW37          //初始化串口 1 工作方式：
                           //串口 0 工作在 PC Net 网络通信方式，

```

```

MOV #0,%MW50 //波特率 38400，8 个数据位，无奇偶校验。
MOV #4,%MW51 //配置串口 0 的 PN 方式：PN 从站
MOV #0,%MW69 //配置串口 0 的 PN 从站地址：4
MOV #4,%MW70 //配置串口 1 的 PN 方式：PN 从站
//配置串口 1 的 PN 从站地址：4

```

```

Network2 //本机的 QW3 自加程序
LD P_0.5S
DIFU //在 0.5s 时钟的上升沿，即每 1s 时：
INC_I %QW3 //QW3 自加 1

```

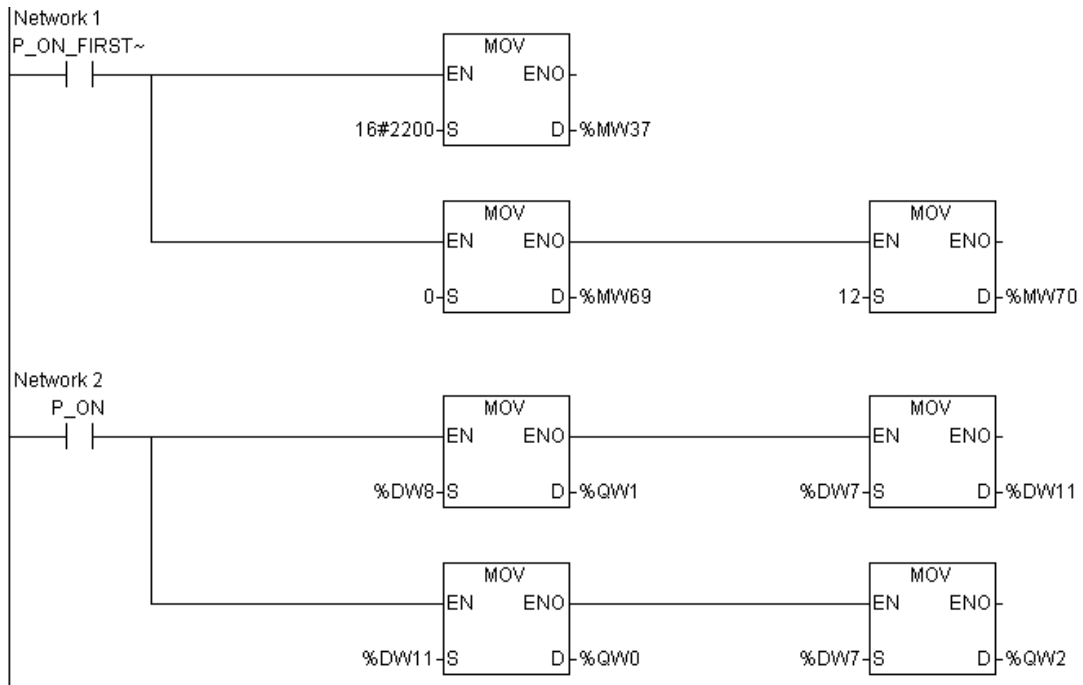
```

Network3 //执行数据拷贝，将各数据按需求拷到目标区
LD P_ON
MOV %QW3,%DW2
MOV %QW3,%DW8
MOV %DW0,%QW1
MOV %DW7,%QW2
MOV %DW11,%QW0

```

### 5.2.7.3 从站 2 的程序

梯形图程序如下：



对应的指令表程序如下：

```

Network1
LD P_ON_FISRT_CYC //在上电第一次扫描时：
MOV #8704,%MW37 //初始化串口 1 工作方式：

```

```

//串口 1 工作在 PC Net 网络通信方式，
//波特率 38400，8 个数据位，无奇偶校验。
MOV #0,%MW69 //配置串口 1 的 PN 方式：PN 从站
MOV #12,%MW70 //配置串口 1 的 PN 从站地址：12

Network2 //执行数据拷贝，将各数据按需求拷到目标区
LD P_ON
MOV %DW8,%QW1
MOV %DW7,%DW11
MOV %DW11,%QW0
MOV %DW7,%QW2

```

## 5.3 自由口通信

自由口通信是通过通信指令 TXD、RXD 来实现的，它的功能是按要求执行发送和接收，而不去管具体的数据意义，具体的通信协议由用户自己去定义，也称为无协议通信。前提是必须将串口设置为自由口方式，并将与自由口方式相关的寄存器配置好。该部分要与指令配合完成，指令的说明及使用参见指令说明手册。自由口通信每次数据长度最大为 256 字节。

### 5.3.1 用户配置说明

配置寄存器：M 区第 88~101 字

用户需配置好自由口消息接收的开始条件和结束条件，系统程序根据用户配置的开始和结束条件控制消息的接收。

用户可以配置发送消息的附加项，即可以为发送的消息定义头和尾，当执行发送指令时，系统程序会自动为消息加上预定义的头和尾部。系统默认是不为待发送消息添加头和尾的。

具体的配置方式在下面发送指令和接收指令中详细介绍。

### 5.3.2 出错寄存器说明

出错寄存器：对于串口 0 和串口 1，分别为 M 区第 40 字和 41 字。

### 5.3.3 发送指令 (TXD)

#### (1) 指令格式

该指令的指令表格式为：TXD S,C,N

其中：S 表示源起始字；C 表示控制字，这里表示串口号；N 表示发送的字节数。

#### (2) 执行条件

当通自由口发送允许标志位置位时，可以执行发送指令。

对于串口 0 和串口 1，该标志位分别为 %SX41 和 %SX46。

#### (3) 配置方式

系统程序可以为发送的消息添加起始字符和结束字符，具体配置方式如下（以串口 0

为例),

i、添加起始字符:

设置: ①M88.h = 1。(默认 M88.h = 0 表示不添加起始字符)

②起始字符设定方式: M88.a = 0, 默认起始字符 0x3A; M88.a = 1, 起始字符由用户自定义。

③自定义起始字符: M89, 可以定义为 00-FF Hex, 但必须是特征字符。

ii、添加结束字符:

设置: ①M88.t = 1。默认 M88.t = 0 表示不添加起始字符。

②字符设定方式: M88.b = 0, 默认结束字符为 0x0D、0x0A; M88.b = 1, 结束字符由用户自定义。

③自定义结束字符: M90, 可以定义为 00-FF Hex, 但必须是特征字符。

### 5.3.4 接收指令 (RXD)

(1) 指令格式

该指令的指令表格式为: RXD S, C, N

其中: S 表示源起始字; C 表示控制字, 这里表示串口号; N 表示接收的字节数。

**注意:** N 默认为 0, 表示按缓冲区中实际接收到的有效字节数进行收取;

若 N 大于缓冲区中实际接收到的字节数, 则按实际数目收取;

若 N 小于或等于缓冲区中实际接收到的字节数, 则按 N 值进行收取;

(2) 执行条件

执行接收指令需要先配置与接收相关的寄存器, 指定消息的开始和结束条件。通讯部分依据指定的条件去分析处理接收。若消息结束, 系统置接收完成标志。用户执行 RXD 指令前需先判断接收完成标志, 接收完成标志置位才可正确地执行 RXD 指令。

对于串口 0 和串口 1, 该标志位分别为 %SX40 和 %SX45。

缓冲区中接收到的消息应及时取走, 否则可能被后面的数据冲掉。若接收检测到一个新的消息起始, 则清零接收完成标志, 开始新的接收过程。

(3) 接收开始条件

接收指令支持多个开始条件, 以串口 0 为例, 由 M88.ss 的设定值决定。

i、起始字符: 起始字符是用作消息的第一个字符的任意字符。当接收到指定的起始字符时, 接收消息开始。

设置: ①M88.ss = 0;

②M88.a = 0, 默认起始字符 0x3A; M88.a = 1, 起始字符由用户自定义;

③M89: 自定义起始字符, 可以定义为 00-FF Hex, 但必须是特征字符。

ii、空闲行时间: 空闲行时间是指传输线上无数据的持续时间。当空闲行时间达到指定时间时, 启动新的接收, 之后接收到的数据是新的消息的起始。

设置: ①M88.ss = 1;

②M88.c 空闲行设定方式: 0-默认当前波特率下 3.5 个字符时间; 1-由用户自定义;

③M92: 自定义空闲行时间。支持的自定义空闲行时间要不小于 16 个字符时间。

iv、任意字符: 在还没有消息开始时, 接收到任意字符表示接收消息的开始。

设置: M88.ss = 2。

(3) 接收结束条件

接收指令支持多个结束条件, 以串口 0 为例, 由 M88.ee 的设定值决定。

i、结束字符：结束字符是用作指定消息结束的任何字符。当正处于接收中（接收开始后），若接收到指定的结束字符，则消息结束。

设置：①M88.ee = 0；

②M88.b = 0，默认结束字符 0x0D、0x0A；M88.b = 1，结束字符由用户自定义；

③M90：自定义结束字符，可以定义为 00-FF Hex，但必须是特征字符。

ii、空闲行时间：接收开始后，当遇到空闲行时间达到设定值，认为接收消息结束。

设置：①M88.ee = 1；

②M88.c 空闲行设定方式：0-默认当前波特率下 3.5 个字符时间；1-由用户自定义；

③M92：自定义空闲行时间。支持的自定义空闲行时间要不小于 16 个字符时间。

iv、接收字符数：接收到指定数量的字符时，认为接收消息结束。

设置：①M88.ee = 2。

②M91：自由口接收字符数设定。

iv、消息定时器：消息定时器在启动消息后指定的时间终止消息。一旦满足接收消息的启动条件，消息定时器就启动。当在 M93 中指定的毫秒数过去之后，消息定时器到时间，接收消息结束。一般地，当通讯设备不能保证字符之间有时间间隙，可以使用消息定时器来指定允许在消息启动后接收消息的最大时间。消息定时器所需的典型数值约为在选择的波特率下接收最长可能消息所需时间的 1.5 倍。

设置：①M88.ee = 3。

②M93：消息定时器超时设定。

**注意：**所有时间设置的精度是 1ms。

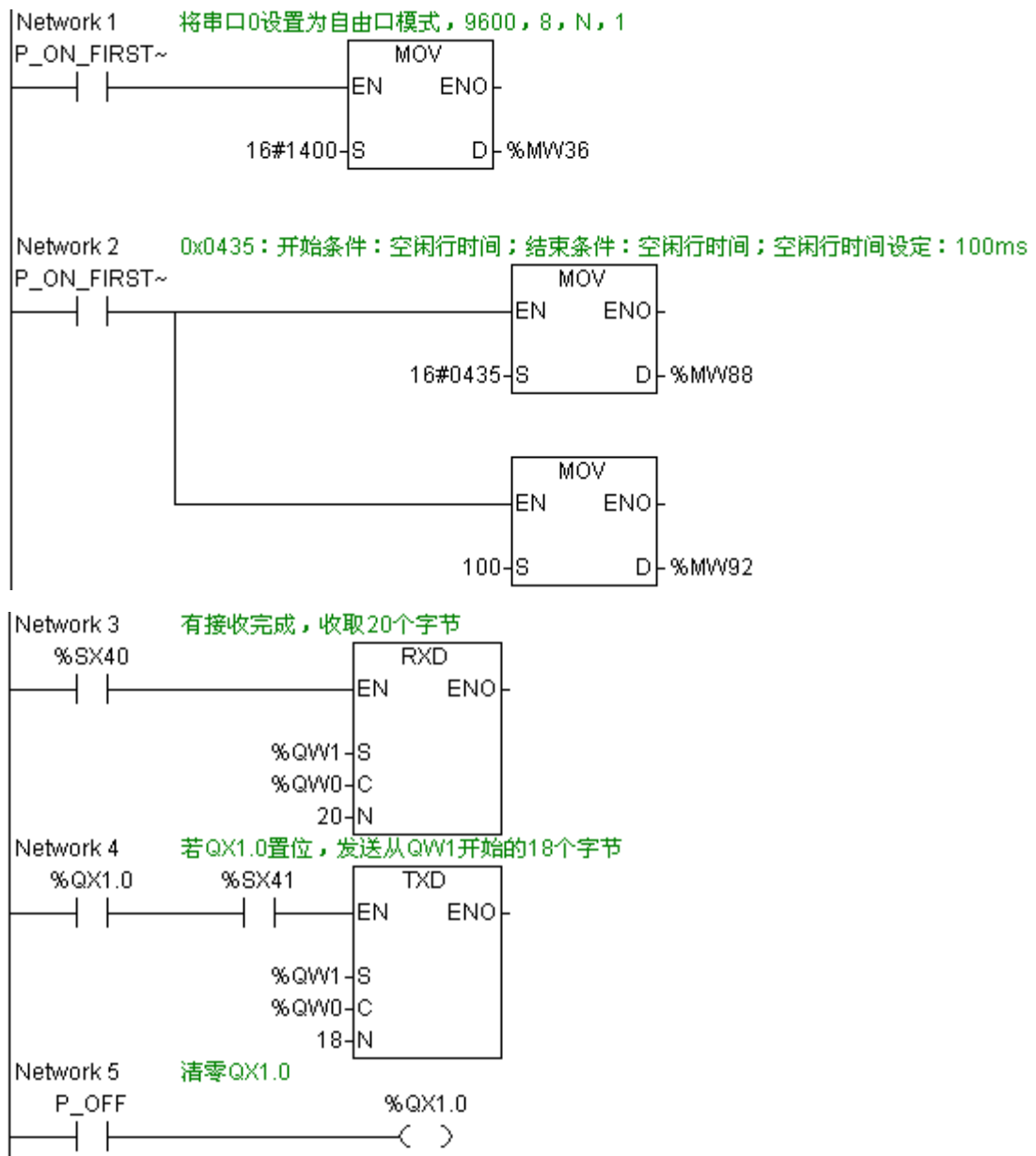
实例说明：

(1) 以空闲行时间作为接收消息的开始和结束条件。

实现的目的：串口以空闲行时间作为接收消息的开始和结束条件，用户指定空闲行时间为 100ms。当有接收事件完成时，指定收取 20 个字节，存放从 QW1 开始的区域。若收取到的首字节的最低位是 1 的话，则把以 QW1 开始的 18 个字节发送出去，且给发送的消息添加起始和结束字符，起始和结束字符分别采用默认的冒号和回车换行符。发送后，将 QW1 的最低位清零。

i、梯形图程序





ii、指令表程序

**Network1** //初始化串口工作方式  
LD P\_ON\_FISRT\_CYC //在上电第一次扫描时：  
MOV #5120,%MW36 //初始化串口0工作方式：  
//串口0工作在自由口方式，  
//波特率9600，8个数据位，无奇偶校验。

**Network2** //自由口模式配置  
LD P\_ON\_FISRT\_CYC //在上电第一次扫描时：  
MOV #1077,%MW88 //配置自由口接收消息的开始条件和结束条件为空闲行时

间

//空闲行时间由用户自己设定（在后面设定）  
//不设定帧超时

```

//给发送的消息自动添加起始和结束字符
//添加的起始和结束字符为默认的“:”和“回车换行符”
MOV #100,%MW92 //设定空闲行时间为 100ms

Network3 //检查接收事件并执行接收指令
LD %SX40 //当有接收事件完成时:
RXD %QW1,%QW0,#20 //执行接收指令, 收取 20 个字节存放到 QW1 开始的区域

Network4 //执行发送指令
LD %QX1.0 //当 QX1.0 (即接收到的首字节的最低位) 置位时:
AND %SX41 //若当前自由口允许发送,
TXD %QW1,%QW0,#18 //执行发送指令, 将以 QW1 开始的 18 个字节发送出去

Network5 //清零 QX1.0, 防止重复发送
LD P_OFF
ST %QX1.0

```

#### iv、运行情况说明

将程序下载到 PLC 中, 将 PLC 置于运行模式, 使串口 0 以自由口方式开始工作。

① 当向串口发送字符串“1234567890abcde”时, 串口发出字符串“:1234567890abcde\NUL\NUL\NUL\CR\LF”。

说明: 由于接收到的字符数小于指定收取的 20 个字节, 接收指令将按实际数目, 把收取的该字符串依次存放到以 QW1 开始的区域。因为收取的首字符为‘1’, 满足用户程序的发送条件, 将执行发送指令, 将以 QW1 开始的 18 个字节发送出去, 且发送前将添加消息的头和尾字符, 于是发出以上字符串。

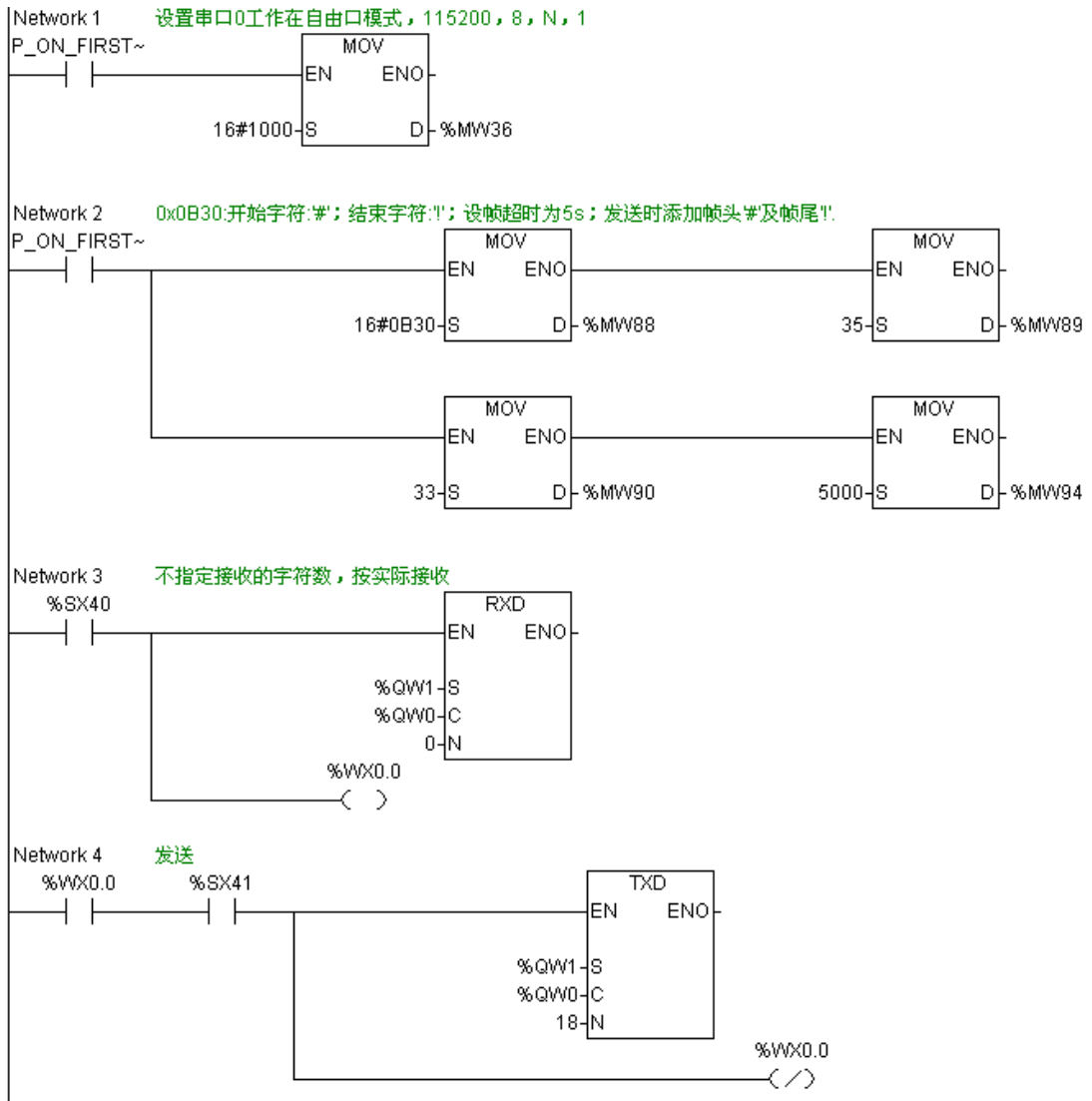
② 当向串口发送字符串“234567890abcdefghijkl”时, 串口不向外回应数据。

说明: 由于接收到的首字符是‘2’, 不满足回应条件, 所以串口没有数据回应。

(2) 以开始和结束字符作为接收消息的开始和结束条件

实现的目的: 串口以开始和结束字符作为接收消息的开始和结束条件, 用户指定起始字符为‘#’, 结束字符为‘!’; 设定接收帧超时时间为 5s。当有接收事件完成时, 将接收到的数据存放到从 QW1 开始的区域。再把以 QW1 开始的 18 个字节发送出去, 且给发送的消息添加起始和结束字符。

#### i、梯形图程序



ii、指令表程序

```

Network1 //初始化串口工作方式
LD P_ON_FISRT_CYC //在上电第一次扫描时：
MOV #4096,%MW36 //初始化串口 0 工作方式：
//串口 0 工作在自由口方式，
//波特率 115200，8 个数据位，无奇偶校验。

Network2 //自由口模式配置
LD P_ON_FISRT_CYC //在上电第一次扫描时：
MOV #2864,%MW88 //自由口接收消息的开始条件和结束条件为起始和结束字
符
//起始和结束字符由用户自己设定（在后面设定）
//设定接收帧超时时间
//给发送的消息自动添加起始和结束字符
MOV #35,%MW89 //设定起始字符为‘#’
MOV #33,%MW90 //设定结束字符为‘!’

```

```

MOV #5000,%MW94      //设定帧超时时间为 5s

Network3              //检查接收事件并执行接收指令
LD %SX40              //当有接收事件完成时:
RXD %QW1,%QW0,#0    //执行接收指令, 将接收到的字节存放到从 QW1 开始的区域
ST %WX0.0             //置标志位 WX0.0

Network4              //执行发送指令
LD %WX0.0             //当 WX0.0 置位时:
AND %SX41             //若当前自由口允许发送,
TXD %QW1,%QW0,#18   //执行发送指令, 将以 QW1 开始的 18 个字节发送出去
STN %WX0.0           //清零 WX0.0

```

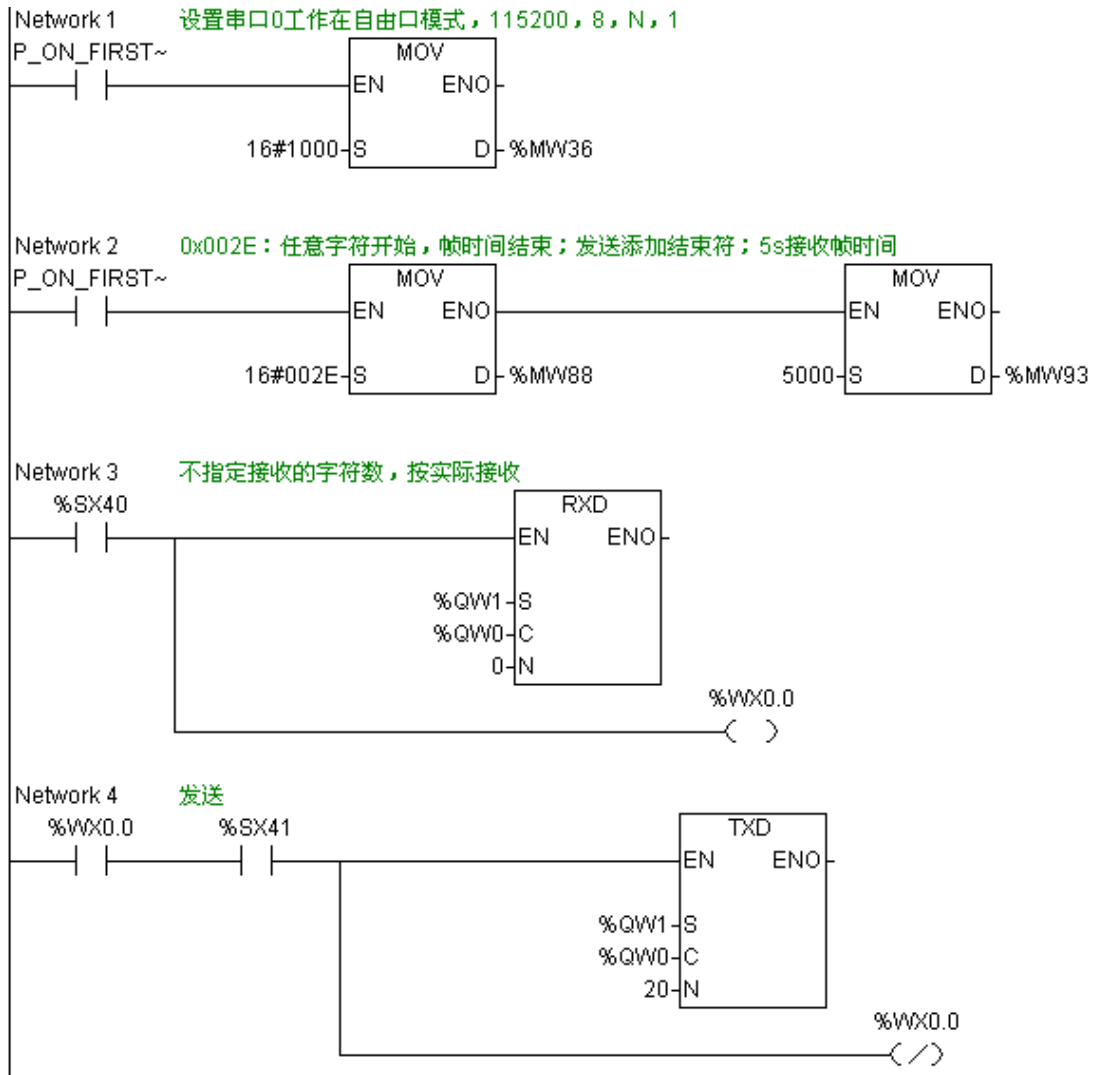
#### iv、运行情况

当向串口发送字符串“#0123456789abcdefghijklmn!”时，串口将回应字符串“##0123456789abcdefg!”。

(3) 以任意字符作为接收消息的开始条件，以帧时间作为结束条件

实现的目的：串口以任意字符作为接收消息的开始条件，以接收帧时间作为消息的结束条件，设定接收帧时间为 5s。当有接收事件完成时，将接收到的数据存放到从 QW1 开始的区域。再把以 QW1 开始的 20 个字节发送出去，且给发送的消息添加默认的开始字符。

#### i、梯形图程序



ii、指令表程序

```

Network1 //初始化串口工作方式
LD P_ON_FISRT_CYC //在上电第一次扫描时：
MOV #4096,%MW36 //初始化串口 0 工作方式：
//串口 0 工作在自由口方式，
//波特率 115200，8 个数据位，无奇偶校验。

Network2 //自由口模式配置
LD P_ON_FISRT_CYC //在上电第一次扫描时：
MOV #46,%MW88 //以任意字符为自由口接收消息的开始条件
//以接收时间为接收消息的结束条件
//给发送的消息自动添加结束字符
MOV #5000,%MW93 //设定接收时间为 5s

Network3 //检查接收事件并执行接收指令
LD %SX40 //当有接收事件完成时：
RXD %QW1,%QW0,#0 //执行接收指令，将接收到的字节存放到从 QW1 开始的区域

```

```

ST %WX0.0           //置标志位 WX0.0

Network4           //执行发送指令
LD %WX0.0         //当 WX0.0 置位时:
AND %SX41         //若当前自由口允许发送,
TXD %QW1,%QW0,#20 //执行发送指令, 将以 QW1 开始的 18 个字节发送出去
STN %WX0.0       //清零 WX0.0

```

## 5. 4Modbus-RTU 从站通信

Modbus-RTU 从站功能作为 PLC 的开放的扩展的功能，符合标准的 Modbus-RTU 协议。

按 Modbus-RTU 协议，数据帧是根据传输字符间的时间间隔来界定的，当接收字符后的停顿时间大于 3.5 个字符时间，则认为是一帧的结束。关于 Modbus-RTU 协议的具体说明，参见相关标准。

### 5. 4. 1 用户配置说明

该工作方式下，用户只需配置 Modbus 从机地址即可。默认的设备地址为 4。

配置寄存器：对于串口 0 和串口 1，分别为 M 区第 102 和第 103 字。

### 5. 4. 2 Modbus 编址

将 Modbus 地址写为 5 个字符数值（Modbus 数据地址用两个字节表示，最大编址范围是 0-65535），包括数据类型和偏移量。前一个字符确定数据类型，后四个字符选择数据类型范围内适当的值。Modbus 主设备把这些地址映射到正确的功能中。Modbus 从设备将地址映射到自身的实际寄存器区。

表 7-2 将 Modbus 地址映射到 PLC

数据区类型	支持的 Modbus 功能码	Modbus 地址	PLC 地址
离散输入量	2	00000	IX0.0
		00001	IX0.1
		...	...
		01599	IX99.15
离散输出量	1 5 15	10000	QX0.0
		10001	QX0.1
		...	...
		11599	QX99.15
输入寄存器	4	30000	IW0
		30001	IW1
		...	...
		30099	IW99

保持寄存器	3 6 16	40000	QW0
		40001	QW1
		...	...
		40099	QW99
		41000	WW0
		41001	WW1
		...	...
		41199	WW199
		42000	DW0
		42001	DW1
		...	...
		44047	DW2047
		45000	HW0
		45001	HW1
		...	...
		45039	HW39

关于 Modbus 协议的使用，见附录九“Modbus-RTU 协议简介”。

## 5.5. Modbus-RTU 主站通信

本 PLC 设备可以做 Modbus-RTU 主站使用，只要把串口工作方式设为 Modbus-RTU 主站即可。Modbus-RTU 主站功能是通过指令来实现的，即用户按照要求使用 Modbus 主站指令配置通信请求命令，执行该指令即是通知系统添加该请求报文，系统程序会自动按照配置生成相应的请求帧，然后添加到发送队列中等待发出。本 PLC 对非周期的通信请求和周期的通信请求都可支持，支持的最大非周期报文队列数为 8，最大的周期报文队列数为 16。指令的说明及使用参考指令说明手册。

用串口 1 (RS-485) 可直接连接多从机的网络，串口 0 (RS-232) 通过电平转换 (RS232/RS485 转换器) 也可以连接多从机的网络。两个串口的使用是独立的，即网络关系互不影响。

### 5.5.1 添加非周期报文指令 (MBAR)

#### (1) 相关标志位

只有非周期报文请求允许使能时，才能正确地执行非周期报文请求的指令。所以一般情况下，用该标志位做添加非周期报文请求指令的执行条件。

串口 0 相关标志位为 %SX44；串口 1 相关标志位为 %SX49。

若当前等待处理的队列已满，系统会将该标志位清零。系统最大支持 8 个队列的非周期报文请求。当前队列中的所有请求报文被处理后，系统自动将队列清零。

#### (2) 指令格式

该指令格式：MBAR comNum, frmHead, data, otSet, stateWord

comNum: 串口号，0 或 1, 支持立即数和 W 型参数。

frmHead: 请求帧数据表头，支持 W 型参数。

data: 目标或源数据区，当为读命令时，表示读取数据的存放位置，当为写命令时，表

示欲写入的数据的地址；支持 X、W 型参数，或立即数 0（无需指定 data 区时）。

otSet: 超时设置（ms），支持立即数和 W 型参数，为 0 表示不修改之前的设定。

stateWord: 状态字，支持 W 型参数，或立即数 0（不为该请求帧设状态字）。

系统会根据请求帧的表头及后续数据，自动分析需要计入的数据长度，并在需要时添加由后面参数指定的数据域，然后计算校验，再组成发送帧。

若该指令的操作需要保存相关信息（例如如果是读取数据时，需指定保存数据的区域），系统会将该信息保存到指定的区域。

用户可以设定从站响应的超时时间标准（以 ms 为单位），系统只有一个超时标准，由最新的设定值决定。若该设定参数为 0，表示不修改之前的设定。系统默认的超时标准为 1s。

主站操作状态字是用来记录 Modbus-RTU 主站发出命令的执行状况，便于用户对其通讯过程进行掌控。若用户不需要为当前的通信请求设定状态字，可以将该参数置 0。

状态字的定义如下：

MSB						LSB														
5	4	3	2	1	0															
异常应答帧的异常代码（Exception Code）										错误代码										

D 完成（请求已完成）： 0 = 没有完成，1 = 完成  
 A 激活（请求已排队）： 0 = 没有激活，1 = 激活  
 E 出错（指令执行出错）： 0 = 没有出错，1 = 出错  
 错误代码定义如下，

代码	定义
0	无错
1	串口工作方式错误：不是 Modbus-RTU 主站方式
2	周期报文请求队列溢出
3	非周期报文请求队列溢出
4	参数错误：非法的串口号或设定值
5	操作越界：主站数据区越界
6	周期报文的轮询周期设定太短
7	超时出错：从站没有应答
8	异常应答帧：异常代码见 Exception Code 部分
9-F	未用（保留）

Exception Code 定义遵循标准 Modbus 协议。

### （3）支持的 Modbus 功能码及使用说明

该功能支持的标准 Modbus 功能码有 1、2、3、4、5、6、15、16。以下举例说明。

#### • 0x01（Read Coil Status）

例如 MBAR #0, %WW0, %QW0, #0, #0

其中

寄存器	数据(Hex)
%WW0	04
%WW1	01
%WW2	27
%WW3	10
%WW4	00



%WW5	25
------	----

该指令表示通过串口 0, 读取 4 号从站, 以地址 0x2710 开始的连续 0x25 个离散输出位。并将读取的位数据依次保存到主站中的以%QX0.0 开始的区域。不改变应答超时标准。不为该过程设定状态字。

• 0x02 (Read Input Status)

例如 MBAR %DW0, %WW10, %WX16.2, %DW1, %DW2

其中,

寄存器	数据(Hex)
%DW0	01
%DW1	C8
%WW10	0C
%WW11	02
%WW12	00
%WW13	05
%WW14	00
%WW15	07

该指令表示通过串口 1, 读取 12 号从站, 以地址 0x0005 开始的连续 0x07 个离散输入位。并将读取的位数据依次保存到主站中的以%WX16.2 开始的区域。应答超时标准设定为%DW1 中的数值, 即 200ms。为该通信请求设定状态字%DW2。

• 0x03 (Read Holding Registers)

例如 MBAR #1, %WW18, %QW2, #500, #0

其中,

寄存器	数据(Hex)
%WW18	04
%WW19	03
%WW20	9C
%WW21	42
%WW22	00
%WW23	06

该指令表示通过串口 1, 读取 4 号从站, 以地址 0x9C42 开始的连续 6 个寄存器的值。并将读取的数据依次保存到主站中的以%QW2 开始的区域。应答超时标准设为 500ms。不为该过程设定状态字。

• 0x04 (Read Input Registers)

例如 MBAR #0, %WW0, %QW0, #0, #0

其中,

寄存器	数据(Hex)
%WW40	04
%WW41	04
%WW42	75
%WW43	30
%WW44	00
%WW45	01

该指令表示通过串口 0，读取 4 号从站，以地址 0x7530 开始的 1 个寄存器的值。并将读取的数据保存到主站中的%QW0 区域。不改变应答超时标准。不为该过程设定状态字。

• 0x05 (Force Single Ciol)

例如：MBAR #0, %DW2, #0, #0, #0

其中，

寄存器	数据(Hex)
%DW2	04
%DW3	05
%DW4	27
%DW5	10
%DW6	FF
%DW7	00

该指令表示通过串口 0，将 4 号从站的地址为 0x2710 的离散输出位置 1。不改变应答超时标准。不为该过程设定状态字。

• 0x06 (Preset Single Registers)

例如 MBAR #0, %WW50, %WW54, #0, #0

其中，

寄存器	数据(Hex)
%WW50	04
%WW51	06
%WW52	9C
%WW53	41
%WW54	1234

该指令表示通过串口 0，将 4 号从站的地址为 0x9C41 的保持寄存器的值设为主站中寄存器%WW54 的值，即 0x1234。不改变应答超时标准。不为该过程设定状态字。

• 0x0F (Force Multiple Ciol)

例如 MBAR #0, %DW60, %QX0.2, #0, #0

其中，

寄存器	数据(Hex)
%DW60	04
%DW61	0F
%DW62	27
%DW63	12
%DW64	00
%DW65	15
%DW66	03
%QW0	30
%DW1	82
%DW2	D6

该指令表示通过串口 0, 将主站中从%QX0.2 开始的连续 0x15 个位数据写入到 4 号从站的以地址 0x2712 开始的离散输出点。不改变应答超时标准。不为该过程设定状态字。

**注意：**因为操作的位偏移不为 0, 所以组织的发送帧的数据域中的数据要由指定的%QW0 区开始的数据经过移位操作得到。

本例中, 组织成的发送帧为 (Hex) 04 0F 27 12 00 15 03 8C A0 15 CRC。

#### • 0x10 (Preset Multiple Registers)

例如 MBAR #0, %WW70, %QW0, #0, #0

其中,

寄存器	数据(Hex)
%WW70	04
%WW71	10
%WW72	9C
%WW73	40
%WW74	00
%WW75	03
%WW76	06
%QW0	1234
%QW1	5678
%QW2	9ABC

该指令表示通过串口 0, 将主站中从%QW0 开始的 3 个寄存器的值依次写入到 4 号从站的以地址为 0x9C40 开始的保持寄存器中。不改变应答超时标准。不为该过程设定状态字。

组织成的发送帧为 (Hex) 04 10 9C 40 00 03 06 12 34 56 78 9A BC CRC。

### 5.5.2 添加周期报文指令 (MBCR)

#### (1) 相关标志位

只有周期报文请求允许使能时, 才能正确地执行周期报文请求的指令。所以一般情况下, 用该标志位做添加周期报文请求指令的执行条件。

串口 0 相关标志位为%SX43; 串口 1 相关标志位为%SX48。

若当前周期报文的队列已满，系统会将该标志位清零，从而不允许继续添加周期性的报文。系统最大支持 16 个队列的周期报文请求。执行删除周期报文的指令后，系统将周期报文队列清空。

#### (2) 指令格式

该指令格式：MBCR comNum, frmHead, data, cycleT, otSet, stateWord

comNum: 串口号，0 或 1,支持立即数和 W 型参数。

frmHead: 请求帧数据表头，支持 W 型参数。

data: 目标或源数据区，当为读命令时，表示读取数据的存放位置，当为写命令时，表示欲写入数据的地址；支持 X、W 型参数，或立即数 0（无需指定 data 区时）。

cycleT: 循环扫描时间（ms），支持立即数和 W 型参数，为 0 表示不修改之前的设定

otSet: 超时设置（ms），支持立即数和 W 型参数，为 0 表示不修改之前的设定。

stateWord: 状态字，支持 W 型参数，或立即数 0（不为该请求帧设状态字）。

用户可以设定循环扫描的周期时间（以 ms 为单位），系统只有一个统一的扫描周期，由最新的设定值决定。若该设定参数为 0，表示不修改之前的设定。系统默认的扫描周期 1s。

与添加非周期报文的指令相比，除了多了个设定周期扫描时间的参数外，其余都一样。在此不再赘述。

### 5.5.3 删除周期报文指令（MBDR）

该指令格式：MBDR comNum

comNum: 串口号，0 或 1,支持立即数和 W 型参数。

执行该指令后，将清除所有周期报文，并将周期报文请求允许的标志位置 1。

### 5.5.4 出错寄存器说明

出错寄存器：对于串口 0，M 区第 45、43 字；对于串口 1，M 区第 49、47 字。出现相关错误时，将被记录到上述对应的寄存器，并导致错误等闪烁，若系统检测到错误消失，将清除相应的记录，也不再导致错误灯闪烁。

# 第 6 章 Modbus-RTU 协议简介

## 6.1. Modbus-RTU 帧结构

起始应有不小于 3.5 个字符的报文间隔	目标站号	功能码	数据	CRC 校验码
	1 字节	1 字节	N 字节	2 字节

## 6.2. Modbus-RTU 命令介绍

**注意：**下面对于各请求命令的“应答格式”的描述是指命令被正确执行时的应答格式。若 CPU 接收到错误的命令或者命令被执行错误，则返回的应答帧中“功能码”部分变为：功能码的最高位置 1 后得到的数据。例如功能码为 01，若响应错误，则返回的功能码为 0x81。

### 功能码 01：读线圈（开关量输出）

功能说明：读取目标站中开关量输出的值（ON/OFF）。

请求格式：

目标站号	功能码	起始地址高字节	起始地址低字节	读取个数高字节	读取个数低字节	CRC 校验码
1 字节	01	1 字节	1 字节	1 字节	1 字节	2 字节

应答格式：

目标站号	功能码	返回数据字节数	返回数据字节 1	返回数据字节 2	...	CRC 校验码
1 字节	01	1 字节	1 字节	1 字节	1 字节	2 字节

举例：从 17 号目标站读取地址为 10019~10055 的线圈的值。

请求	
域名	数据（十六进制）
目标站号	11
功能码	01
起始地址高字节	27
起始地址低字节	23
读取个数高字节	00
读取个数低字节	25
CRC 校验码	04
	3F

应答	
域名	数据（十六进制）
目标站号	11
功能码	01
返回数据字节数	05
数据字节 (线圈 10026~10019)	CD
数据字节 (线圈 10034~10027)	6B
数据字节 (线圈 10042~10035)	B2
数据字节 (线圈 10050~10043)	0E
数据字节 (线圈 10055~10051)	1B
CRC 校验码	45
	E6

由起始地址开始，线圈状态作为二进制位，从地位向高位填充字节。线圈 10026~10019 的状态对应字节 0xCD，或者 11001101B。线圈 10026 是字节的 MSB，线圈 10019 是字节的 LSB。从左到右，线圈 10026 到 10019 的状态是：ON-ON-OFF-OFF-ON-ON-OFF-ON。下一个字节对应线圈 10034~10027，依次类推。如果读取个数不是 8 的整数倍，那么最后一个字节的剩余位用 0 填补。在最后一个字节中，线圈 10055~10051 的状态对应字节 0x1B，或者 00011011B。线圈 10055 到 10051 的状态是：ON-ON-OFF-ON-ON。

### 功能码 02：读输入状态（开关量输入）

功能说明：读取目标站中开关量输入的值（ON/OFF）。

请求格式：

目标站号	功能码	起始地址 高字节	起始地址 低字节	读取个数 高字节	读取个数 低字节	CRC 校验码
1 字节	02	1 字节	1 字节	1 字节	1 字节	2 字节

应答格式：

目标站号	功能码	返回数据 字节数	返回数据 字节 1	返回数据 字节 2	...	CRC 校验码
1 字节	02	1 字节	1 字节	1 字节	1 字节	2 字节

举例：从 17 号目标站读取地址为 196~217 的开关量输入的值。

请求	
域名	数据（十六进制）
目标站号	11
功能码	02
起始地址高字节	00
起始地址低字节	C4
读取个数高字节	00
读取个数低字节	16
CRC 校验码	BA
	A9

应答	
域名	数据（十六进制）
目标站号	11
功能码	02
返回数据字节数	03
数据（输入 203~196）	AC
数据（输入 211~204）	DB
数据（输入 217~212）	35
CRC 校验码	20
	18

### 功能码 03：读保持寄存器（模拟量输出）

功能说明：读取目标站中保持寄存器的内容。

请求格式：

目标站号	功能码	起始地址高字节	起始地址低字节	读取个数高字节	读取个数低字节	CRC 校验码
1 字节	03	1 字节	1 字节	1 字节	1 字节	2 字节

应答格式：

目标站号	功能码	返回数据字节数	返回数据字节 1	返回数据字节 2	...	CRC 校验码
1 字节	03	1 字节	1 字节	1 字节	...	2 字节

举例：从 17 号目标站读取地址为 41107~41109 的寄存器的值。

请求	
域名	数据（十六进制）
目标站号	11
功能码	03
起始地址高字节	A0
起始地址低字节	93
读取个数高字节	00
读取个数低字节	03
CRC 校验码	D5
	76

应答	
域名	数据（十六进制）
目标站号	11
功能码	03
返回数据字节数	06
数据高字节（寄存器 41107）	02
数据低字节（寄存器 41107）	2B
数据高字节（寄存器 41108）	00
数据低字节（寄存器 41108）	00
数据高字节（寄存器 41109）	00
数据低字节（寄存器 41109）	64
CRC 校验码	C8
	BA

每个寄存器由 2 个字节构成，第 1 个字节表示高位数据，第 2 个字节表示低位数据。寄存器 41107 的内容是 0x022B (555d)。寄存器 41108~41109 的内容是 0x0000 和 0x0064 (0d 和 100d)。

#### 功能码 04: 读输入寄存器（模拟量输入）

功能说明：读取目标站中输入寄存器的内容。

请求格式：

目标站号	功能码	起始地址 高字节	起始地址 低字节	读取个数 高字节	读取个数 低字节	CRC 校验码
1 字节	04	1 字节	1 字节	1 字节	1 字节	2 字节

应答格式：



目标站号	功能码	返回数据 字节数	返回数据 字节 1	返回数据 字节 2	...	CRC 校验码
1 字节	04	1 字节	1 字节	1 字节	...	2 字节

举例：从 17 号目标站读取地址为 30008 的寄存器的值。

请求	
域名	数据（十六进制）
目标站号	11
功能码	04
起始地址高字节	75
起始地址低字节	38
读取个数高字节	00
读取个数低字节	01
CRC 校验码	A8
	9B

应答	
域名	数据（十六进制）
目标站号	11
功能码	04
返回数据字节数	02
数据高字节（寄存器 30008）	00
数据低字节（寄存器 30008）	0A
CRC 校验码	F8
	F4

寄存器 30008 的内容是 0x000A（10d）。

### 功能码 05：写单个线圈（开关量输出）

功能说明：将单线圈强制为 ON 或 OFF。除非控制逻辑在以后的程序中处理线圈，该线圈的强制值一直保持有效。

请求格式：

目标站号	功能码	起始地址 高字节	起始地址 低字节	强制值 高字节	强制值 低字节	CRC 校验码
1 字节	05	1 字节	1 字节	1 字节	1 字节	2 字节

**注意：**强制值=0xFF00，则置线圈为 ON；强制值=0x0000，则置线圈为 OFF。

应答格式：若设置成功，原文返回

举例：将 17 号目标站地址为 10172 的线圈置 ON

请求	
域名	数据（十六进制）
目标站号	11
功能码	05
线圈地址高字节	27
线圈地址低字节	BC
强制数据高字节	FF
强制数据低字节	00
CRC 校验码	45
	FA

应答	
域名	数据（十六进制）
目标站号	11
功能码	05
线圈地址高字节	27
线圈地址低字节	BC
强制数据高字节	FF
强制数据低字节	00
CRC 校验码	45
	FA

### 功能码 06：写单个保持寄存器（模拟量输出）

功能说明：预设单保持寄存器的值。除非控制逻辑在以后的程序中处理寄存器内容，该寄存器将一直保持预设值。

请求格式：

目标站号	功能码	寄存器地址 高字节	寄存器地址 低字节	预设值 高字节	预设值 低字节	CRC 校验码
1 字节	06	1 字节	1 字节	1 字节	1 字节	2 字节

应答格式：

若设置成功，原文返回

举例：将 17 号目标站地址为 40001 的寄存器的值置为 0x0003。

请求	
域名	数据（十六进制）
目标站号	11
功能码	06
寄存器地址高字节	9C
寄存器地址低字节	41
预设数据高字节	00
预设数据低字节	03
CRC 校验码	B5
	1F

应答	
域名	数据（十六进制）
目标站号	11
功能码	06
寄存器地址高字节	9C
寄存器地址低字节	41
预设数据高字节	00
预设数据低字节	03
CRC 校验码	B5
	1F

### 功能码 15：写多个线圈（开关量输出）

功能说明：将一系列线圈强制为 ON 或 OFF。除非控制逻辑在以后的程序中处理线圈，这些线圈的强制值一直保持有效。

请求格式：

目标站号	功能码	起始地址高字节	起始地址低字节	数量高字节	数量低字节	强制值字节数	强制数据第 1 字节	...	CRC 校验码
1 字节	15	1 字节	1 字节	1 字节	1 字节	1 字节	1 字节	...	2 字节

应答格式：

目标站号	功能码	起始地址高字节	起始地址低字节	数量高字节	数量低字节	CRC 校验码
1 字节	15	1 字节	1 字节	1 字节	1 字节	2 字节

举例：将 17 号目标站的地址从 10019 开始的 10 个线圈的值进行强制。

请求	
域名	数据（十六进制）
目标站号	11
功能码	0F
起始地址高字节	27
起始地址低字节	23
数量高字节	00
数量低字节	0A
字节数	02
强制数据第 1 字节 (线圈 10026~10019)	CD
强制数据第 1 字节 (线圈 10028~10027)	01
CRC 校验码	ED
	F9

应答	
域名	数据（十六进制）
目标站号	11
功能码	0F
起始地址高字节	27
起始地址低字节	23
线圈数量高字节	00
线圈数量低字节	0A
CRC 校验码	2C
	22

位:	1	1	0	0	1	1	0	1
线圈:	10026	10025	10024	10023	10022	10021	10020	10019
位:	0	0	0	0	0	0	0	1
线圈:	-	-	-	-	-	-	10028	10027

第 1 个字节发送 0xCD 到线圈 10026~10019，最低有效位写到序列中地址最低的线圈 10019 中。

下一个字节发送 0x01 到线圈 10028~10027，最低有效位写到序列中地址最低的线圈 10027 中。最后一个字节中没有使用的位用 0 填充。

### 功能码 16：写多个保持寄存器（模拟量输出）

功能说明：预设一系列保持寄存器的值。除非控制逻辑在以后的程序中处理寄存器内容，寄存器将一直保持预设值。

请求格式：

目标站号	功能码	起始地址高字节	起始地址低字节	数量高字节	数量低字节	预设值字节数	预设值1高字节	预设值1低字节	...	CRC 校验码
1 字节	16	1 字节	1 字节	1 字节	1 字节	1 字节	1 字节	1 字节	...	2 字节

应答格式：

目标站号	功能码	起始地址高字节	起始地址低字节	数量高字节	数量低字节	CRC 校验码
1 字节	16	1 字节	1 字节	1 字节	1 字节	2 字节

举例：将 17 号目标站地址从 40001 开始的 2 个寄存器的值预设为 0x000A 和 0x0102。

请求	
域名	数据（十六进制）
目标站号	11
功能码	10
起始地址高字节	9C
起始地址低字节	41
寄存器数量高字节	00
寄存器数量低字节	02
字节数	04
数据 1 高字节	00
数据 1 低字节	0A
数据 2 高字节	01
数据 2 低字节	02
CRC 校验码	3B
	C6

应答	
域名	数据（十六进制）
目标站号	11
功能码	10
起始地址高字节	9C
起始地址低字节	41

寄存器数量高字节	00
寄存器数量低字节	02
CRC 校验码	3D
	1C

### 6.3 异常回应

错误码:

错误码	名称	说明
01	非法功能	从站不能执行请求帧中功能码所对应的操作。
02	非法数据地址	从站不能访问请求帧数据域中的数据地址。
03	非法数据值	从站不允许操作请求帧数据域中的某个值。
04	从站设备错误	当从站尝试执行请求操作时，发生一个不可恢复的错误。

错误回应格式:

目标站号	功能码	错误码	CRC 校验码
1 字节	1 字节	1 字节	2 字节

举例：主站发出请求，从站进行错误回应。

请求	
域名	数据（十六进制）
目标站号	0A
功能码	01
起始地址高字节	04
起始地址低字节	A1
读取个数高字节	00
读取个数低字节	01
CRC 校验码	AC
	63

应答	
域名	数据（十六进制）
目标站号	0A
功能码	81
错误码	02
CRC 校验码	B0
	53

在这个例子中，主站向 10 号从站发送请求。功能码 01 表示执行读取线圈状态操作，请

求读取地址为 1245 (0x04A1) 的线圈状态，读取线圈个数是 1。

如果线圈地址不存在于从站设备中，从站会在错误应答中将功能码高位置 1，返回错误码 02，表示非法的数据地址。

## 6.4 CRC 校验算法

在 Modbus RTU 协议中，使用 CRC 作为帧的校验方式。下面是用 C 编写的两种 CRC 算法：

### 6.4.1 直接计算 CRC

```
/* 参数: chData -- const BYTE*, 指向待校验数据存储区的首地址
        uNO -- 待校验数据的字节个数
返回值: WORD 型, 计算出的 CRC 值。 */
WORD CalcCrc(const BYTE* chData, WORD uNo)
{
    WORD crc=0xFFFF;
    WORD wCrc;
    UCHAR i,j;
    for (i=0; i<uNo; i++)
    {
        crc ^= chData[i];
        for (j=0; j<8; j++)
        {
            if (crc & 1)
            {
                crc >>= 1;
                crc ^= 0xA001;
            }
            else
                crc >>= 1;
        }
    }
    wCrc=( (WORD)LOBYTE(crc) )<<8;
    wCrc=wCrc| ( (WORD)HIBYTE(crc) );
    return (wCrc);
}
```

### 6.4.2 查表快速计算 CRC

```
/* 高字节 CRC 表 */
const UCHAR auchCRCHi[] =
```

```

{
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40
} ;

```

/\* 低字节 CRC 表 \*/

```

const UCHAR auchCRCLo[] =
{
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06,
0x07, 0xC7, 0x05, 0xC5, 0xC4, 0x04, 0xCC, 0x0C, 0x0D, 0xCD,
0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09,
0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A,
0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC, 0x14, 0xD4,
0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3,
0xF2, 0x32, 0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4,
0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A,
0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29,
0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF, 0x2D, 0xED,
0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,

```



```

0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60,
0x61, 0xA1, 0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67,
0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F,
0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68,
0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA, 0xBE, 0x7E,
0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71,
0x70, 0xB0, 0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92,
0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C,
0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B,
0x99, 0x59, 0x58, 0x98, 0x88, 0x48, 0x49, 0x89, 0x4B, 0x8B,
0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42,
0x43, 0x83, 0x41, 0x81, 0x80, 0x40
};

```

```

/* 参数: puchMsg  -- const BYTE*, 指向待校验数据存储区的首地址
          usDataLen  -- 待校验数据的字节个数
返回值: WORD 型, 计算出的 CRC 值。 */
WORD CKDNSerialCom::CalCrcFast(const BYTE* puchMsg, WORD usDataLen)
{
    BYTE uchCRCHi = 0xFF; /* CRC 高字节初始化 */
    BYTE uchCRCLo = 0xFF; /* CRC 低字节初始化 */
    WORD uIndex; /* CRC 查表的索引 */
    while (usDataLen--)
    {
        uIndex = uchCRCHi ^ *puchMsg++; /* 计算 CRC */

        uchCRCHi = uchCRCLo ^ auchCRCHi[uIndex];
        uchCRCLo = auchCRCLo[uIndex];
    }
    return (uchCRCHi << 8 | uchCRCLo);
}

```

## 第 7 章 系统寄存器一览表 (M 区)

地址	名称	说明
00		保留
01	设定的程序扫描时间	范围 0000—9999，单位：毫秒 通过上位机软件设定的程序扫描时间，固定时间扫描模式有效
02	用户模式选择	安全模式/运行模式选择码(低字节有效)：0-运行模式；1-安全模式
03	扫描次数	从 PLC 运行开始扫描的次数 (16 进制)
04		共 4 个字节，04 地址装高 2 位字节，03 地址装低位字节
05	扫描周期	由 PLC 运行用户程序得到的扫描周期
06	断电次数	从第一次上电运行以来，断电的次数
07	PLC 机型	0：未知机型 1：VPC1 系列 ...
08	内存清除设定	位 15-12: IOM 保持状态：0 清除 1 保持 位 11—0：保留 如果位 15-12 状态为强制状态，且该 4 位值为 1，则从断电转到上电状态时，PLC 存储器 D 区的 %DW0-%DW511，W 区的 %WW0-%WW199。
09	强制状态设定	位 15-12: IOM 保持状态：0 清除 1 保持 位 11—0：保留 如果位 15-12 状态为强制状态，且该 4 位值为 1，则从运行状态转到编程状态时，强制状态被保持。
10	严重错误码	见附录严重错误代码说明部分
11	非严重错误码	见附录非严重错误代码说明部分
12	PLC 系统时间	分：秒 BCD 格式
13	PLC 系统时间	日：时： BCD 格式
14	PLC 系统日期	年：月： BCD 格式
15	星期几	周几 BCD 格式
16	模拟电阻 0	0x0000-0x03FF
17	模拟电阻 1	0x0000-0x03FF
18	HSC0 控制字	参考 VPC1 系列 PLC 用户手册高速计数器部分
19	HSC1 控制字	参考 VPC1 系列 PLC 用户手册高速计数器部分
20	HSC0 中断初始 值	参考 VPC1 系列 PLC 用户手册高速计数器部分
21		

22	保留	保留为高速计数器使用																																																
23	HSC0 中断类型	8 位，高 8 位保留																																																
24	HSC1 中断初始 值	参考 VPC1 系列 PLC 用户手册高速计数器部分																																																
25																																																		
26	保留	保留为高速计数器使用																																																
27	HSC1 中断类型	8 位，高 8 位保留																																																
28	HSC0 错误类型 (1 表示出错， 0 表示正确)	Bit1: 配置寄存器设定错误 Bit2: HDEF 设定错误 Bit3: 初始值设定错误 Bit4: 中断值设定错误																																																
29	HSC1 错误类型 (该位为 1 表示 出错, 为 0 表示正 确)	Bit1: 配置寄存器设定错误 Bit2: HDEF 设定错误 Bit3: 初始值设定错误 Bit4: 中断值设定错误																																																
30	HSC0 当前值	24 位有符号数																																																
31																																																		
32	HSC1 当前值	24 位有符号数																																																
33																																																		
34	扩展通信模块输入区地址	DEVICE NET、Profibus、Ethernet/IP 扩展通信模块的通信是通过同 D 区进行数据交换的，扩展通信模块输入区长度固定为 10 个字，																																																
35	扩展通信模块输出区地址	扩展通信模块输出区长度固定为 10 个字																																																
通信相关参数																																																		
36	串口 0 设置寄存器	<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <tr> <td>MSB</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>LSB</td> </tr> <tr> <td>15</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>8</td> </tr> <tr> <td>x</td> <td>m</td> <td>m</td> <td>m</td> <td>x</td> <td>b</td> <td>b</td> <td>b</td> </tr> <tr> <td>MSB</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>LSB</td> </tr> <tr> <td>7</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>0</td> </tr> <tr> <td>x</td> <td>p</td> <td>p</td> <td>p</td> <td>d</td> <td>d</td> <td>s</td> <td>s</td> </tr> </table> <p>其中： mmm 为 000: 默认的同上位机编程调试模式 001: 自由口通信模式 010: PC 主从网络通信模式（可同时支持 PC-Link 通信和网络指令通信） 011: Modbus 从站模式 100: Modbus 主站模式</p>	MSB							LSB	15							8	x	m	m	m	x	b	b	b	MSB							LSB	7							0	x	p	p	p	d	d	s	s
MSB							LSB																																											
15							8																																											
x	m	m	m	x	b	b	b																																											
MSB							LSB																																											
7							0																																											
x	p	p	p	d	d	s	s																																											

		<p>其它值：保留（默认到同上位机编程调试模式）</p> <p>bbb: 波特率  000: 115200 波特    001: 57600 波特  010: 38400 波特    011: 19200 波特  100: 9600 波特      101: 4800 波特  110: 2400 波特      111: 1200 波特</p> <p>ss: 起始位/停止位  00: 1 位起始位/1 位停止位  01: 1 位起始位/2 位停止位</p> <p>dd:每个字符的数据位  00: 每个字符 8 位  01: 每个字符 7 位</p> <p>ppp:奇偶校验选择  000: 无奇偶校验  001: 奇校验                    010: 偶校验  011: 该位强制为 1    100: 该位强制为 0  其它值：默认到无奇偶校验</p>																																																
37	串口 1 设置寄存器	<table border="1" data-bbox="568 1133 1391 1263"> <tr><td>MSB</td><td></td><td></td><td></td><td></td><td></td><td></td><td>LSB</td></tr> <tr><td>15</td><td></td><td></td><td></td><td></td><td></td><td></td><td>8</td></tr> <tr><td>x</td><td>m</td><td>m</td><td>m</td><td>x</td><td>b</td><td>b</td><td>b</td></tr> </table> <table border="1" data-bbox="568 1308 1401 1438"> <tr><td>MSB</td><td></td><td></td><td></td><td></td><td></td><td></td><td>LSB</td></tr> <tr><td>7</td><td></td><td></td><td></td><td></td><td></td><td></td><td>0</td></tr> <tr><td>x</td><td>p</td><td>p</td><td>p</td><td>d</td><td>d</td><td>s</td><td>s</td></tr> </table> <p>其中：  mmm 为  000: 同上位机编程调试模式  001: 自由口通信模式  010: PC 主从网络通信模式（可同时支持 PC-Link 通信和网络指令通信）  011: Modbus 从站模式  100: Modbus 主站模式  其它值：保留（默认到 Modbus 从站模式）</p> <p>bbb: 波特率  000: 115200 波特    001: 57600 波特  010: 38400 波特    011: 19200 波特  100: 9600 波特      101: 4800 波特</p>	MSB							LSB	15							8	x	m	m	m	x	b	b	b	MSB							LSB	7							0	x	p	p	p	d	d	s	s
MSB							LSB																																											
15							8																																											
x	m	m	m	x	b	b	b																																											
MSB							LSB																																											
7							0																																											
x	p	p	p	d	d	s	s																																											

		<p>110: 2400 波特    111: 1200 波特</p> <p>ss: 起始位/停止位  00: 1 位起始位/1 位停止位  01: 1 位起始位/2 位停止位</p> <p>dd:每个字符的数据位  00: 每个字符 8 位  01: 每个字符 7 位</p> <p>ppp:奇偶校验选择  000: 无奇偶校验  001: 奇校验                    010: 偶校验  011: 该位强制为 1    100: 该位强制为 0  其它值: 默认到无奇偶校验</p>																																																
38-39	保留																																																	
40	串口 0 自由口通信出错寄存器	<table border="1"> <tr><td>MSB</td><td></td><td></td><td></td><td></td><td></td><td></td><td>LSB</td></tr> <tr><td>15</td><td></td><td></td><td></td><td></td><td></td><td></td><td>8</td></tr> <tr><td>X</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr> </table> <table border="1"> <tr><td>MSB</td><td></td><td></td><td></td><td></td><td></td><td></td><td>LSB</td></tr> <tr><td>7</td><td></td><td></td><td></td><td></td><td></td><td></td><td>0</td></tr> <tr><td>x</td><td>e6</td><td>e5</td><td>e4</td><td>e3</td><td>e2</td><td>e1</td><td>e0</td></tr> </table> <p>若某位置位，表示有相关错误。否则，表示无该错误。  各个位错误定义如下：  e0-串口工作方式错误（串口不是自由口方式）；  e1-配置错误；  e2-接收超时；  e3-接收数据溢出（超出缓冲区容量）；  e4-发送失败（不允许发送）  e5-发送口忙  e6-自由口读指令读取的数据量超过接收的数据量</p>	MSB							LSB	15							8	X	x	x	x	x	x	x	x	MSB							LSB	7							0	x	e6	e5	e4	e3	e2	e1	e0
MSB							LSB																																											
15							8																																											
X	x	x	x	x	x	x	x																																											
MSB							LSB																																											
7							0																																											
x	e6	e5	e4	e3	e2	e1	e0																																											
41	串口 1 自由口通信出错寄存器	<table border="1"> <tr><td>MSB</td><td></td><td></td><td></td><td></td><td></td><td></td><td>LSB</td></tr> <tr><td>15</td><td></td><td></td><td></td><td></td><td></td><td></td><td>8</td></tr> <tr><td>X</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr> </table> <table border="1"> <tr><td>MSB</td><td></td><td></td><td></td><td></td><td></td><td></td><td>LSB</td></tr> <tr><td>7</td><td></td><td></td><td></td><td></td><td></td><td></td><td>0</td></tr> <tr><td>x</td><td>e6</td><td>e5</td><td>e4</td><td>e3</td><td>e2</td><td>e1</td><td>e0</td></tr> </table> <p>若某位置位，表示有相关错误。否则，表示无该错误。</p>	MSB							LSB	15							8	X	x	x	x	x	x	x	x	MSB							LSB	7							0	x	e6	e5	e4	e3	e2	e1	e0
MSB							LSB																																											
15							8																																											
X	x	x	x	x	x	x	x																																											
MSB							LSB																																											
7							0																																											
x	e6	e5	e4	e3	e2	e1	e0																																											

		<p>各个位错误定义如下：  e0-串口工作方式错误（串口不是自由口方式）；  e1-配置错误；  e2-接收超时；  e3-接收数据溢出（超出缓冲区容量）；  e4-发送失败（不允许发送）  e5-发送口忙  e6-自由口读指令读取的数据量超过接收的数据量</p>																																																
42	串口 0 PN 网络通讯出错寄存器	<p>网络通信错误定义如下：</p> <table border="1" style="margin-bottom: 10px;"> <tr> <td>MSB</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>LSB</td> </tr> <tr> <td>15</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>8</td> </tr> <tr> <td>X</td> <td>x</td> <td>x</td> <td>x</td> <td>x</td> <td>x</td> <td>e9</td> <td>e8</td> </tr> </table> <table border="1" style="margin-bottom: 10px;"> <tr> <td>MSB</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>LSB</td> </tr> <tr> <td>7</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>0</td> </tr> <tr> <td>e7</td> <td>e6</td> <td>e5</td> <td>e4</td> <td>e3</td> <td>e2</td> <td>e1</td> <td>e0</td> </tr> </table> <p>低字节：网络通信错误类型。若某位置位，表示有相关错误，否则，表示无该错误。定义如下，  e0-配置错误；  e1-两个串口配置冲突；  e2-保留；  e3-网络指令通信请求队列溢出；  e4-PL 方式访问从站无应答；  e5-PL 方式访问从站异常（收到异常回应帧）  e6-NI 方式访问从站无应答；  e7-NI 方式访问从站异常（收到异常回应帧）  e8-主站访问错误；  e9-非主站模式下使用网络指令；  注：主站和从站的错误在一起定义，e0 对于设备做主站或从站都有用，e8、e9 仅对设备做从站时有用，其余错误仅对设备做主站时有用；  错误 e4、e5 互斥，将记录最新的错误，若是错误 e4 或 e5 时，故障情况将在 M43 中记录；  错误 e6、e7 互斥，将记录最新的错误，若是错误 e6 或 e7 时，故障情况将在 M44 中记录</p>	MSB							LSB	15							8	X	x	x	x	x	x	e9	e8	MSB							LSB	7							0	e7	e6	e5	e4	e3	e2	e1	e0
MSB							LSB																																											
15							8																																											
X	x	x	x	x	x	e9	e8																																											
MSB							LSB																																											
7							0																																											
e7	e6	e5	e4	e3	e2	e1	e0																																											
43	网络 0 访问从站故障记录 1	<p>低字节： 当前通讯故障时的功能码  bit8-bit12: 错误时的从机地址（1-31）  bit13-bit15: 收到出错回应帧时的错误代码(1-7)  <b>注意：</b> 这里只记录最新的错误情况</p>																																																
44	网络 0 访问从站故障记录 2	<p>低字节： 当前通讯故障时的功能码  bit8-bit12: 错误时的从机地址（1-31）  bit13-bit15: 收到出错回应帧时的错误代码(1-7)  <b>注意：</b> 这里只记录最新的错误情况</p>																																																

45	串口 0 Modbus-RTU 通讯出错寄 存器	MSB						LSB	
		15						8	
		X	x	x	x	x	x	e9	e8
		MSB						LSB	
		7						0	
		e7	e6	e5	e4	e3	e2	e1	e0
<p>Modbus-RTU 通信错误类型。若某位置位，表示有相关错误，否则，表示无该错误。定义如下，</p> <p>e0-配置错误；</p> <p>e1-Modbus 从站模式下使用主站指令；</p> <p>e2-主站访问错误；</p> <p>e3-周期报文请求失败；</p> <p>e4-非周期报文请求失败；</p> <p>e5-保留；</p> <p>e6-保留；</p> <p>e7-周期报文的轮询周期设定太短；</p> <p>e8-访问从站无应答；</p> <p>e9-访问从站异常（收到异常回应帧）</p> <p>注：主站和从站的错误在一起定义，e0、e1、e2 仅对设备做从站时有用，其余错误仅对设备做主站时有用；</p> <p>错误 e8、e9 互斥，将记录最新的错误，若是错误 e8 或 e9 时，故障情况将在 M43 中记录；</p>									
46	串口 1 PN 网 络通讯出错 寄存器	网络通信错误定义如下：							
		MSB						LSB	
		15						8	
		X	x	x	x	x	x	e9	e8
		MSB						LSB	
		7						0	
e7	e6	e5	e4	e3	e2	e1	e0		
<p>低字节：网络通信错误类型。若某位置位，表示有相关错误，否则，表示无该错误。定义如下，</p> <p>e0-配置错误；</p> <p>e1-两个串口配置冲突；</p> <p>e2-保留；</p> <p>e3-网络指令通信请求队列溢出；</p> <p>e4-PL 方式访问从站无应答；</p> <p>e5-PL 方式访问从站异常（收到异常回应帧）</p> <p>e6-NI 方式访问从站无应答；</p> <p>e7-NI 方式访问从站异常（收到异常回应帧）</p> <p>e8-主站访问错误；</p> <p>e9-非主站模式下使用网络指令；</p> <p>注：主站和从站的错误在一起定义，e0 对于设备做主站或从站都有用，</p>									

		e8、e9 仅对设备做从站时有用，其余错误仅对设备做主站时有用； 错误 e4、e5 互斥，将记录最新的错误，若是错误 e4 或 e5 时，故障情况将在 M43 中记录； 错误 e6、e7 互斥，将记录最新的错误，若是错误 e6 或 e7 时，故障情况将在 M44 中记录																																																
47	网络 1 访问从站故障记录 1	低字节：当前通讯故障时的功能码 bit8-bit12: 错误时的从机地址（1-31） bit13-bit15: 收到出错回应帧时的错误代码(1-7) <b>注意：</b> 这里只记录最新的错误情况																																																
48	网络 1 访问从站故障记录 2	低字节：当前通讯故障时的功能码 bit8-bit12: 错误时的从机地址（1-31） bit13-bit15: 收到出错回应帧时的错误代码(1-7) <b>注意：</b> 这里只记录最新的错误情况																																																
49	串口 1 Modbus-RTU 通讯出错寄存器	<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <tr> <td>MSB</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>LSB</td> </tr> <tr> <td>15</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>8</td> </tr> <tr> <td>X</td> <td>x</td> <td>x</td> <td>x</td> <td>x</td> <td>x</td> <td>e9</td> <td>e8</td> </tr> <tr> <td>MSB</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>LSB</td> </tr> <tr> <td>7</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>0</td> </tr> <tr> <td>e7</td> <td>e6</td> <td>e5</td> <td>e4</td> <td>e3</td> <td>e2</td> <td>e1</td> <td>e0</td> </tr> </table> <p>Modbus-RTU 通信错误类型。若某位置位，表示有相关错误，否则，表示无该错误。定义如下， e0-配置错误； e1-Modbus 从站模式下使用主站指令； e2-主站访问错误； e3-周期报文请求失败； e4-非周期报文请求失败； e5-保留； e6-保留； e7-周期报文的轮询周期设定太短； e8-访问从站无应答； e9-访问从站异常（收到异常回应帧） 注：主站和从站的错误在一起定义，e0、e1、e2 仅对设备做从站时有用，其余错误仅对设备做主站时有用； 错误 e8、e9 互斥，将记录最新的错误，若是错误 e8 或 e9 时，故障情况将在 M47 中记录；</p>	MSB							LSB	15							8	X	x	x	x	x	x	e9	e8	MSB							LSB	7							0	e7	e6	e5	e4	e3	e2	e1	e0
MSB							LSB																																											
15							8																																											
X	x	x	x	x	x	e9	e8																																											
MSB							LSB																																											
7							0																																											
e7	e6	e5	e4	e3	e2	e1	e0																																											
50	串口 0 PC 网络主配置字	低字节：0 为从站，1 为支持网络指令和 PC-Link 通信的主站，2 为仅支持网络指令的主站 高字节：保留 注：配置为从站时，51 有效，配置为支持 PC-Link 的主站时，52 以下有效。																																																
51	从站地址	输入范围：1-31，配置为从站时该项有效。																																																



52	PC-Link 从站数量	0-7，以下需要配置数据区的范围由 PC-Link 从站数量决定。 该值为 0 时表示只有网络指令的通信。
53	PC-Link 数据域起始地址	PC-Link 网络中公共数据域的起始地址：0-2047，表示 D0-D2047 注：公共数据域中数据分区按下面的顺序（主站、从站 1、从站 2、...）及配置的大小依次排列，且要保证整个数据域的总大小不超过 32 字节。
54	主站数据区大小	PC-Link 网络的公共数据域中主站的数据区的大小（字节数）：1-16
55	从站 1 地址	输入范围：1-31
56	从站 1 PC-Link 数据区大小	PC-Link 网络的公共数据域中从站 1 的数据区的大小（字节数）：1-16
57	从站 2 地址	输入范围：1-31
58	从站 2 PC-Link 数据区大小	PC-Link 网络的公共数据域中从站 2 的数据区的大小（字节数）：1-16
...	...	
...	...	
...	...	
67	从站 7 地址	输入范围：1-31
68	从站 7 PC-Link 数据区大小	PC-Link 网络的公共数据域中从站 7 的数据区的大小（字节数）：1-16
69	串口 1 PC 网络主配置字	低字节：0 为从站，1 为支持网络指令和 PC-Link 通信的主站，2 为仅支持网络指令的主站 高字节：保留 注：配置为从站时，70 有效，配置为支持 PC-Link 的主站时，71 以下有效。
70	从站地址	输入范围：1-31，配置为从站时该项有效。
71	PC-Link 从站数量	0-7，以下需要配置数据区的范围由 PC-Link 从站数量决定。 该值为 0 时表示仅支持网络指令。
72	PC-Link 数据域起始地址	PC-Link 网络中公共数据域的起始地址：0-2047，表示 D0-D2047 注：公共数据域中数据分区按下面的顺序（主站、从站 1、从站 2、...）及配置的大小依次排列，且要保证整个数据域的总大小不超过 32 字节。
73	主站数据区大小	PC-Link 网络的公共数据域中主站的数据区的大小（字节数）：1-16
74	从站 1 地址	输入范围：1-31
75	从站 1 PC-Link 数据区大小	PC-Link 网络的公共数据域中从站 1 的数据区的大小（字节数）：1-16
76	从站 2 地址	输入范围：1-31
77	从站 2 PC-Link 数据	PC-Link 网络的公共数据域中从站 2 的数据区的大小（字节数）：1-16

	区大小																																																	
...	...																																																	
...	...																																																	
...	...																																																	
...	...																																																	
86	从站 7 地址	输入范围： 1-31																																																
87	从站 7 PC-Link 数据 区大小	PC-Link 网络的公共数据域中从站 7 的数据区的大小（字节数）： 1-16																																																
88	串口 0 自由口 主配置字	<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <tr> <td>MSB</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>LSB</td> </tr> <tr> <td>15</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>8</td> </tr> <tr> <td>x</td> <td>x</td> <td>x</td> <td>x</td> <td>d</td> <td>c</td> <td>b</td> <td>a</td> </tr> </table> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <tr> <td>MSB</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>LSB</td> </tr> <tr> <td>7</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>0</td> </tr> <tr> <td>x</td> <td>x</td> <td>t</td> <td>h</td> <td>e</td> <td>e</td> <td>s</td> <td>s</td> </tr> </table> <p>其中：</p> <p>ss: 接收消息开始条件  00 = 起始字符  01 = 空闲行时间  10 = 任意字符</p> <p>ee: 接收消息结束条件  00 = 结束字符  01 = 空闲行时间  10 = 接收字符数  11 = 消息定时器（此时不存在帧超时）</p> <p>h: 发送消息头设置  0 = 默认无变化  1 = 添加起始字符</p> <p>t: 发送消息尾设置  0 = 默认无变化  1 = 添加结束字符</p> <p>a: 起始字符设定方式  0= 默认设定为 0x3A（ASCII 字符“:”）  1= 由后面的寄存器设定</p> <p>b: 结束字符设定方式  0= 默认设定为 0x0D、0x0A（\CR\LF）  1 = 由后面的寄存器设定</p> <p>c: 空闲行时间设定方式  0 = 当前波特率下 3.5 个字符时间</p>	MSB							LSB	15							8	x	x	x	x	d	c	b	a	MSB							LSB	7							0	x	x	t	h	e	e	s	s
MSB							LSB																																											
15							8																																											
x	x	x	x	d	c	b	a																																											
MSB							LSB																																											
7							0																																											
x	x	t	h	e	e	s	s																																											

		<p>1 = 由后面的寄存器指定（要求不小于当前波特率下 16 个字符时间）</p> <p>d: 帧超时时间设定方式</p> <p>0 = 不设帧超时</p> <p>1 = 超时时间由后面的寄存器指定</p>																																																
89	串口 0 自由口接收消息开始字符设定	<p>00-FF Hex</p> <p>主配置字中接收消息开始条件设为起始字符时，该配置字有效。此时，当接收到该指定的字符时，接收消息开始。</p>																																																
90	串口 0 自由口接收消息结束字符设定	<p>00-FF Hex</p> <p>主配置字中接收消息结束条件设为结束字符时，该配置字有效。此时，当接收到该指定的字符时，接收消息完成，并将置位自由口接收完成标志位。</p>																																																
91	串口 0 自由口接收字符数设定	<p>00: 256</p> <p>01-FF Hex: 1-255</p> <p>高字节: 保留</p> <p>主配置字中接收消息结束条件设为接收字符数时，该配置字有效。此时，若接收消息已开始，当接收的字符达到该指定的个数时，接收消息完成，并将置位自由口接收完成标志位。</p>																																																
92	串口 0 自由口空闲行时间设定	<p>1-1000（单位 1ms）</p> <p>主配置字中接收消息开始或结束条件设为空闲行时间时，该配置字有效。</p> <p>若开始条件设为空闲行时间，则当检测到线上有该指定时间的空闲时，则接收消息开始。</p> <p>若结束条件设为空闲行时间，则当检测到线上有该指定时间的空闲时，则接收消息完成，并将置位自由口接收完成标志位。</p> <p>注：自定义空闲行时间最少为 1ms，并且应不小于当前波特率下 16 个字符时间</p>																																																
93	串口 0 自由口消息定时器超时设定	<p>（单位 1ms）</p> <p>主配置字中接收消息结束条件设为消息定时器时，该配置字有效。此时，当接收消息开始后，经过该时间后，接收消息完成，并将置位自由口接收完成标志位。</p> <p>注：消息定时器超时时间最少为 1ms，典型数值约为在当前波特率下接收最长可能消息所需时间的 1.5 倍。</p>																																																
94	串口 0 自由口帧超时时间设定	<p>100-60000（单位 1ms）</p> <p>进入有效接收状态后，若在此超时时间内仍未收到结束条件，系统就会结束接收状态，并置接收超时错误。</p>																																																
95	串口 1 自由口主配置字	<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <tr> <td>MSB</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>LSB</td> </tr> <tr> <td>15</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>8</td> </tr> <tr> <td>x</td> <td>x</td> <td>x</td> <td>x</td> <td>d</td> <td>c</td> <td>b</td> <td>a</td> </tr> </table> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <tr> <td>MSB</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>LSB</td> </tr> <tr> <td>7</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>0</td> </tr> <tr> <td>x</td> <td>x</td> <td>t</td> <td>h</td> <td>e</td> <td>e</td> <td>s</td> <td>s</td> </tr> </table>	MSB							LSB	15							8	x	x	x	x	d	c	b	a	MSB							LSB	7							0	x	x	t	h	e	e	s	s
MSB							LSB																																											
15							8																																											
x	x	x	x	d	c	b	a																																											
MSB							LSB																																											
7							0																																											
x	x	t	h	e	e	s	s																																											

		<p>其中：</p> <p>ss: 接收消息开始条件  00 = 起始字符  01 = 空闲行时间  10 = 任意字符</p> <p>ee: 接收消息结束条件  00 = 结束字符  01 = 空闲行时间  10 = 接收字符数  11 = 消息定时器（此时不存在帧超时）</p> <p>h: 发送消息头设置  0 = 默认无变化  1 = 添加起始字符</p> <p>t: 发送消息尾设置  0 = 默认无变化  1 = 添加结束字符</p> <p>a: 起始字符设定方式  0= 默认设定为 0x3A（ASCII 字符“:”）  1= 由后面的寄存器设定</p> <p>b: 结束字符设定方式  0= 默认设定为 0x0D、0x0A（\CR\LF）  1 = 由后面的寄存器设定</p> <p>c: 空闲行时间设定方式  0 = 当前波特率下 3.5 个字符时间  1 = 由后面的寄存器指定（要求不小于当前波特率下 16 个字符时间）</p> <p>d: 帧超时时间设定方式  0 = 不设帧超时  1 = 超时时间由后面的寄存器指定</p>
96	串口 1 自由口 接收消息开始 字符设定	<b>00-FF Hex</b> 主配置字中接收消息开始条件设为起始字符时，该配置字有效。此时，当接收到该指定的字符时，接收消息开始。
97	串口 1 自由口 接收消息结束 字符设定	<b>00-FF Hex</b> 主配置字中接收消息结束条件设为结束字符时，该配置字有效。此时，当接收到该指定的字符时，接收消息完成，并将置位自由口接收完成标志位。
98	串口 1 自由口 接收字符数 设定	<b>00: 256</b> <b>01-FF Hex: 1-255</b> 高字节：保留 主配置字中接收消息结束条件设为接收字符数时，该配置字有效。此时，若接收消息已开始，当接收的字符达到该指定的个数时，接收消息完成，并将置位自由口接收完成标志位。

99	串口1自由口 空闲行时间 设定	<p>1-1000（单位 1ms）</p> <p>主配置字中接收消息开始或结束条件设为空闲行时间时，该配置字有效。</p> <p>若开始条件设为空闲行时间，则当检测到线上有该指定时间的空闲时，则接收消息开始。</p> <p>若结束条件设为空闲行时间，则当检测到线上有该指定时间的空闲时，则接收消息完成，并将置位自由口接收完成标志位。</p> <p>注：自定义空闲行时间最少为 1ms，并且应不小于当前波特率下 16 个字符时间</p>																																																
100	串口1自由口 消息定时器 超时设定	<p>（单位 1ms）</p> <p>主配置字中接收消息结束条件设为消息定时器时，该配置字有效。此时，当接收消息开始后，经过该时间后，接收消息完成，并将置位自由口接收完成标志位。</p> <p>注：消息定时器超时时间最少为 1ms，典型数值约为在当前波特率下接收最长可能消息所需时间的 1.5 倍。</p>																																																
101	串口1自由口 帧超时时间 设定	<p>100-60000（单位 1ms）</p> <p>进入有效接收状态后，若在此超时时间内仍未收到结束条件，系统就会结束接收状态，并置接收超时错误。</p>																																																
102	串口0 Modbus-RTU 从站模式配 置字	<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <tr> <td>MSB</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>LSB</td> </tr> <tr> <td>15</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>8</td> </tr> <tr> <td>x</td> <td>x</td> <td>x</td> <td>x</td> <td>x</td> <td>x</td> <td>x</td> <td>x</td> </tr> </table> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <tr> <td>MSB</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>LSB</td> </tr> <tr> <td>7</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>0</td> </tr> <tr> <td>x</td> <td>x</td> <td>x</td> <td>x</td> <td>x</td> <td>x</td> <td>x</td> <td>x</td> </tr> </table> <p>其中： 低字节：Modbus-RTU 从站地址。（1-255） 高字节：保留</p>	MSB							LSB	15							8	x	x	x	x	x	x	x	x	MSB							LSB	7							0	x	x	x	x	x	x	x	x
MSB							LSB																																											
15							8																																											
x	x	x	x	x	x	x	x																																											
MSB							LSB																																											
7							0																																											
x	x	x	x	x	x	x	x																																											
103	串口1 Modbus-RTU 从站模式配 置字	<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <tr> <td>MSB</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>LSB</td> </tr> <tr> <td>15</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>8</td> </tr> <tr> <td>x</td> <td>x</td> <td>x</td> <td>x</td> <td>x</td> <td>x</td> <td>x</td> <td>x</td> </tr> </table> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <tr> <td>MSB</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>LSB</td> </tr> <tr> <td>7</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>0</td> </tr> <tr> <td>x</td> <td>x</td> <td>x</td> <td>x</td> <td>x</td> <td>x</td> <td>x</td> <td>x</td> </tr> </table> <p>其中： 低字节：Modbus-RTU 从站地址。（1-255） 高字节：保留</p>	MSB							LSB	15							8	x	x	x	x	x	x	x	x	MSB							LSB	7							0	x	x	x	x	x	x	x	x
MSB							LSB																																											
15							8																																											
x	x	x	x	x	x	x	x																																											
MSB							LSB																																											
7							0																																											
x	x	x	x	x	x	x	x																																											

104	端口地址（与上位机编程软件通信）	低字节：串口 0 地址（1~254） 高字节：串口 1 地址（1~254）
120	密码级别	0：完全权限 1：部分权限 2：最小权限
扩展模块相关		
180	扩展模块的输入输出字节总长度	下 8 位：总输入字节长度；上 8 位：总输出字节长度
181	连接的扩展模块的数量	
182	扩展模块 0 标识寄存器	低字节有用，表示模块的类型，如下： 1: 16 点数字量输入模块 2: 16 点数字量晶体管输出模块 6: 4 通道模拟量输出模块 9: 12 点数字量输入和 8 点数字量输出模块 10: 4 通道模拟量输入模块 11: DeviceNet 从站模块 12: Profibus-DP 从站模块 13: Ethernet/IP 从站模块
183	扩展模块 0 出错寄存器	0: 不存在 1: 存在且正常 2: 出错
184	扩展模块 1 标识寄存器	格式同扩展模块 0 标识寄存器
185	扩展模块 1 出错寄存器	格式同扩展模块 0 出错寄存器
186	扩展模块 2 标识寄存器	格式同扩展模块 0 标识寄存器
187	扩展模块 2 出错寄存器	格式同扩展模块 0 出错寄存器
188	模块 3 标识寄存器	格式同扩展模块 0 标识寄存器
189	扩展模块 3 出错寄存器	格式同扩展模块 0 出错寄存器
190	扩展模块 4 标识寄存器	格式同扩展模块 0 标识寄存器
191	扩展模块 4 出错寄存器	格式同扩展模块 0 出错寄存器
192	扩展模块 5 标识寄存器	格式同扩展模块 0 标识寄存器
193	扩展模块 5 出错寄存器	格式同扩展模块 0 出错寄存器

	错寄存器	
194	扩展模块6标识寄存器	格式同扩展模块0标识寄存器
195	扩展模块6出错寄存器	格式同扩展模块0出错寄存器
196	扩展模块7标识寄存器	格式同扩展模块0标识寄存器
197	扩展模块7出错寄存器	格式同扩展模块0出错寄存器
198-199		保留
200		测试保留
201-255		





## 第 8 章 特殊数据寄存器一览表 (S 区)

以下特殊寄存器可以用在 BOOL 型触点或指令参数中

位地址	名称	说明
00		保留
01	常 ON	继电器常闭
02	常 OFF	继电器常开
03	扫描脉冲继电器	每次扫描交替开闭
04	运行初期 ON	脉冲继电器只在第一个扫描周期闭合, 从第二个扫描周期开始断开并保持
05	运行初期 OFF	脉冲继电器只在第一个扫描周期断开, 从第二个扫描周期开始闭合并保持
06	1 秒翻转继电器 1 秒 ON, 1 秒 OFF	交替开闭
07	0.5 秒翻转继电器 1 秒 ON, 0.5 秒 OFF	交替开闭
08	0.5 分钟 ON, 0.5 分钟 OFF	交替开闭, 提供 1 分钟时钟脉冲
09	0.1 秒翻转继电器 0.1 秒 ON, 0.1 秒 OFF	0.1 秒交替开闭(时间精度依赖用户程序的运行时间, 要求用户程序运行时间小于 80 毫秒, 否则可能该标志不准确)
10	看门狗复位标志	1:看门狗超时动作 0:未动作
11	10 秒翻转继电器 10 秒 ON, 10 秒 OFF	
12	0.02 秒翻转继电器 0.02 秒 ON, 0.1 秒 OFF	0.02 秒变换一次状态
13-15		保留
16	T1 使用标志	0: T1 给 485 通信使用 1: T1 用做其它应用 注: 当 T1 用做其它应用时, 在 UART1 通信时, 若对方设备响应时间极快(响应间隔<2ms), 可能造成通信不畅, 此时建议降低波特率。特别地, 当 PLC 用做 Modbus 主站, 若从站的响应间隔<1ms, 推荐使用不高于 57600 的波特率。
17	程序扫描时间选择	0: 默认无限循环 1: 定时循环, 时间参数参考 M01 程序扫描时间
18	程序扫描时间是否超时标志	在定时循环的情况: 0 正常 1 超时
19	错误的指令码	0: 正常

		1: 出错
31	程序文件过大(由上位机控制文件大小, 不设计)	0: 正常 1: 出错
32	扩展 I/O 模块通信出错	0: 正常 1: 出错
33	程序指令执行错	
34- 35		保留
36	UART0 通信出错	0: 正常 1: 出错
37	UART1 通信出错	0: 正常 1: 出错
40	通信口 0 自由口接收完成	为 OFF 时表示没有接收到完整消息; 为 ON 时表示接收到完整消息
41	通信口 0 自由口发送允许	为 OFF 时表示不允许发送; 为 ON 时表示允许发送
42	通信口 0 网络指令通信请求允许	为 OFF 时表示不允许网络指令通信请求; 为 ON 时表示允许网络指令通信请求
43	通信口 0 Modbus-RTU 周期报文请求允许	为 OFF 时表示不允许添加周期报文; 为 ON 时表示允许添加周期报文
44	通信口 0 Modbus-RTU 非周期报文请求允许	为 OFF 时表示不允许添加非周期报文; 为 ON 时表示允许添加非周期报文
45	通信口 1 自由口接收完成	为 OFF 时表示没有接收到完整消息; 为 ON 时表示接收到完整消息
46	通信口 1 自由口发送允许	为 OFF 时表示不允许发送; 为 ON 时表示允许发送
47	通信口 1 网络指令通信请求允许	为 OFF 时表示不允许网络指令通信请求; 为 ON 时表示允许网络指令通信请求
48	通信口 1 Modbus-RTU 周期报文请求允许	为 OFF 时表示不允许添加周期报文; 为 ON 时表示允许添加周期报文
49	通信口 1 Modbus-RTU 非周期报文请求允许	为 OFF 时表示不允许添加非周期报文; 为 ON 时表示允许添加非周期报文
50	断电上电以后 HR 区数据未被保持	0: 正常 1: 未被保持, 且 HR 区数据已经被清 0
51	内存硬件故障标志	0: 正常 1: 出错
52	电池电压低标志	0: 正常 1: 电压低
53	FLASH 存储器故障标志	0: 正常 1: 出错
54	程序校验和错误	0: 正常 1: 出错
63	进位标志 Carry (CY) Flag	指令执行过程中 0: 无进位 1: 有进位

64	>GR 大于标志	比较指令执行结果 0: 其它 1: 大于
65	=EQ 等于标志	比较指令执行结果 0: 其它 1: 等于
66	<LT 小于标志	比较指令执行结果 0: 其它 1: 小于
67	>=GE 大于等于标志	比较指令执行结果 0: 其它 1: 大于等于
68	<=LE 小于等于标志	比较指令执行结果 0: 其它 1: 小于等于
69	<>NE 不等于标志	比较指令执行结果 0: 其它 1: 不等于
70	N 负数标志	指令执行结果 0: 非负数 1: 负数
71	指令执行错误标志	
72	溢出标志	
73~79	系统保留	-
80	PTO 脉冲串输出结束标志	0: 未结束 1: 结束

# 第 9 章 错误代码

## 9.1 严重错误代码

M 区第 10 字存放严重错误代码，意义如下

代码	意义
0000	没有严重错误显示；无错
0001	SRAM 内存故障
0002	FLASH 存储器故障出错
0004	I/O 通信错误
0008	保留
0010	保留
0020	保留
0040	保留
0080	用户程序检验和出错
0100	保留
0200	保留
0400	保留
0800	保留
1000	保留
2000	保留
4000	保留
8000	保留

## 9.2 非严重错误代码

M 区第 11 字存放严重错误代码，意义如下

代码	意义
0000	没有严重错误显示；无错
0001	用户程序扫描时间超出预定时间
0002	保留
0004	保留
0008	保留
0010	串口 0 错误
0020	串口 1 错误
0040	锂电池电压低警告
0080	指令执行错误
0100	HR 区掉电以后数据未能保持住
0200	保留
0400	保留
0800	保留
8000	看门狗溢出错误

## Shanghai Electrical Apparatus Research Institute (Group) Co.,Ltd

---

上海电科集团总线技术研发中心  
地址：上海市普陀区武宁路 505 号  
电话：021-62574990\*476  
传真：021-62166010  
邮编：200063  
网址：<http://www.seari.com.cn>

VPC1-PRO-DOC(0910)

内容如有变动  
恕不另行通知