

SIEMENS

TI-S7 Converting Blocks 编程
Programming with TI-S7 Converting Blocks

Getting Started

Edition (2008 年 11 月)

摘要

本文档主要用于讨论与 TI-S7 Converting Blocks 编程相关的以下问题:

- ◇ 功能块的基本信息
- ◇ 基本使用讲解
- ◇ 简单应用例子

关键词

编程; Step7; TI-S7 Converting Blocks;

Key Words

Programming ; Step7; TI-S7 Converting Blocks;

目 录

TI-S7 CONVERTING BLOCKS编程	1
1. 前言	5
2. 功能块介绍	5
2.1. 功能划分	5
2.2. 功能描述	6
2.3. 位逻辑功能	8
2.3.1. 复位位范围: FC82	8
2.3.2. 复位立即输出范围: FC100	9
2.3.3. 置位位范围: FC83	11
2.3.4. 置位立即输出范围: FC101	12
2.4. 表功能	14
2.4.1. 添加到表格: FC84	14
2.4.2. 先进/先出取出表格数据: FC85	15
2.4.3. 查表: FC86	16
2.4.4. 后进/先出取出表格数据: FC87	18
2.4.5. 表格: FC88	19
2.4.6. 将表格移动到字: FC89	21
2.4.7. 字到表格: FC91	23
2.4.8. 关联数据表: FC103	25
2.4.9. 表到表操作: FC104	27
2.5. 移位功能	28
2.5.1. 字移位寄存器: FC90	28
2.5.2. 位移位寄存器: FC92	30
2.6. 移动功能及功能块	31
2.6.1. 间接块移动: FC81	31
2.6.2. 压缩数据: FB86	33
2.7. 定时器功能及功能块	36
2.7.1. 软件延时定时器 - 掉电保护: FC80	36
2.7.2. 离散控制报警定时器: FB81	37
2.7.3. 电机控制报警定时器: FB82	39
2.7.4. 可屏蔽事件Drum: FB85	42
2.8. 转换功能及功能块	47
2.8.1. 七段解码器: FC93	47
2.8.2. ASCII转换为十六进制: FC94	49

2.8.3.	十六进制转换为ASCII: FC95.....	50
2.8.4.	编码二进制位置: FC96.....	52
2.8.5.	解码二进制位置: FC97.....	53
2.8.6.	十进制补码: FC98.....	54
2.8.7.	位数求和: FC99.....	55
2.8.8.	标定值: FC105.....	56
2.8.9.	取消标定值: FC106.....	57
2.8.10.	超前/滞后算法: FB80.....	59
2.9.	浮点数数学运算功能	60
2.9.1.	标准偏差: FC102.....	60
2.10.	比较功能块	62
2.10.1.	索引矩阵比较: FB83.....	62
2.10.2.	扫描矩阵比较: FB84.....	64
3.	附录一推荐网址	66
3.1.	西门子自动化与驱动产品的在线技术支持	66

1. 前言

STEP7 在标准函数库中提供了 TI-S7 Converting blocks 目录，此目录中的函数为用户提供了
一些比较实用的功能，用户可以在程序中引用这些程序。用户可以在 LAD/STL/FBD 的编程界面下，
选择其中某个块后，通过按键盘上的 “F1” 键可以查看此块的帮助信息。鉴于目前国内对这些块
的使用还不普遍，本文将对这些程序块做简单的介绍。相关介绍将以 STEP7 的在线帮助文件为依
据，适当增加程序中难于理解的介绍及例子，减少程序中容易理解部分的介绍，希望本文的介绍
能够给用户有益的帮助。

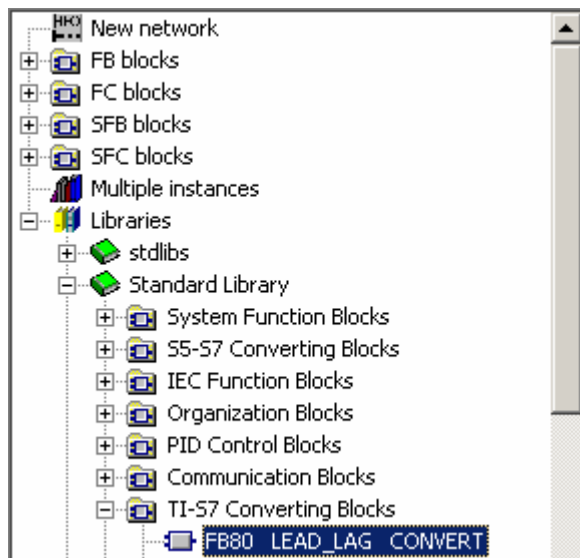


图 1-1: TI-S7 Converting Blocks

2. 功能块介绍

2.1. 功能划分

TI-S7 Converting Blocks 目录中的函数按照功能划分，可以分为与位逻辑、表功能、移位功
能、移动功能、定时器功能、块转换功能、浮点数数学运算功能、比较功能相关的 FC 或 FB。在
后续的章节中，本文将按照不同功能的划分，分别对这些功能块进行讲解。

相关功能	功能描述及编号
位逻辑功能	复位位范围: FC82
	复位立即输出范围: FC100

	置位位范围: FC83
	置位立即输出范围: FC101
表功能	添加到表格: FC84
	先进/先出取出表格数据: FC85
	查表: FC86
	后进/先出取出表格数据: FC87
	表格: FC88
	将表格移动到字: FC89
	字到表格: FC91
	关联数据表: FC103
	表到表操作: FC104
	移位功能
位移寄存器: FC92	
移动功能及功能块	间接块移动: FC81
	压缩数据: FB86
定时器功能及功能块	软件延时定时器 - 掉电保护: FC80
	离散控制报警定时器: FB81
	电机控制报警定时器: FB82
	可屏蔽事件 Drum: FB85
转换功能及功能块	七段解码器: FC93
	ASCII 转换为十六进制: FC94
	十六进制转换为 ASCII: FC95
	编码二进制位置: FC96
	解码二进制位置: FC97
	十进制补码: FC98
	位数求和: FC99
	标定值: FC105
	取消标定值: FC106
	超前/滞后算法: FB80
浮点数数学运算功能	标准偏差: FC102
比较功能块	索引矩阵比较: FB83
	扫描矩阵比较: FB84

表 2-1 TI-S7 Converting Blocks 分类

2.2. 功能描述

软件延时定时器 - 掉电保护: FC80

间接块移动: FC81

复位位范围: FC82

置位位范围: FC83

添加到表格: FC84

先进/先出取出表格数据: FC85

查表: FC86

后进/先出取出表格数据: FC87

表格: FC88

将表格移动到字: FC89

字移位寄存器: FC90

字到表格: FC91

位移寄存器: FC92

七段解码器: FC93

ASCII 转换为十六进制: FC94

十六进制转换为 ASCII: FC95

编码二进制位置: FC96

解码二进制位置: FC97

十进制补码: FC98

位数求和: FC99

复位立即输出范围: FC100

置位立即输出范围: FC101

标准偏差: FC102

关联数据表: FC103

表到表操作: FC104

标定值: FC105

取消标定值: FC106

超前/滞后算法: FB80

离散控制报警定时器: FB81

电机控制报警定时器: FB82

索引矩阵比较: FB83

扫描矩阵比较: FB84

可屏蔽事件 Drum: FB85

压缩数据: FB86

2.3. 位逻辑功能

2.3.1. 复位位范围：FC82

描述

如果 MCR 位为 1，RSET 功能将指定范围中每一位的信号状态复位为 0。如果 MCR 位为 0，该范围中每一位的信号状态保持不变。范围中要复位的位数由 N 指定，范围的起始位由 S_BIT 指示。

参数

参数	描述	数据类型	存储区	描述
EN	输入	BOOL	I、Q、M、D、L	使能输入端的信号状态为 1，激活该功能。
ENO	输出	BOOL	I、Q、M、D、L	如果该功能的执行无错误，使能输出端的信号状态为 1。
S_BIT	输入	*Pointer	I、Q、M、D	指向范围中的第一位。
N	输入	INT	I、Q、M、D、L、 P、常数	范围中要设置的位数。
* 用于跨区域寄存器间接寻址的双字指针格式				

表 2-2 FC82 参数说明

错误信息

如果 S_BIT 指针指向 I/O 外部输入和输出存储区(P 存储器)，则范围内每个位的信号状态保持不变，且 ENO 的信号状态设置为 0。

实例

如果输入 I0.0 的信号状态为 1 (激活)，并且 MCR 位为 1，则执行 RSET 功能。S_BIT 指向位于 M0.0 的第一个位。N 参数指定 10 个位要复位。执行该指令后，M0.0 至 M1.1 范围中 10 个位的信号状态都复位为 0。

如果该功能的执行未发生错误，ENO 和 Q0.0 的信号状态将设置为 1。

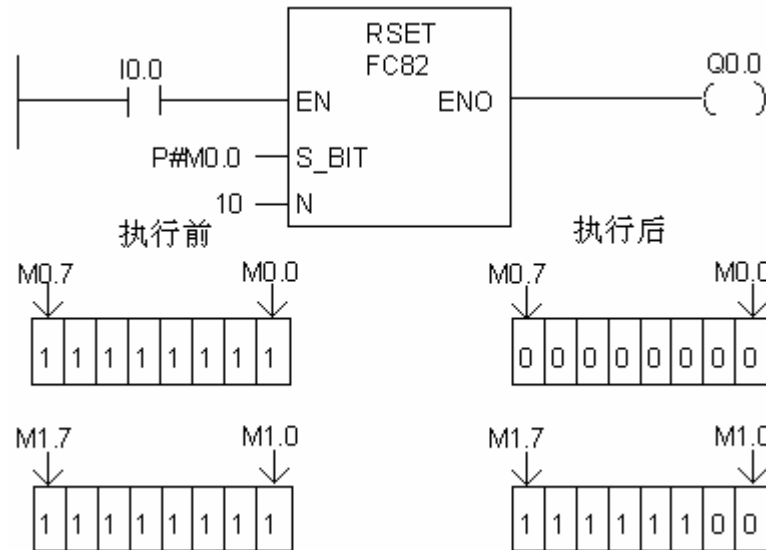


图 2-1: FC82 程序例子

特别说明:

此 FC 主要用于 MCR (主控继电器), 也可单独使用。

2.3.2. 复位立即输出范围: FC100

描述

如果 MCR 位为 1, RSETI 功能将一定范围内字节的信号状态复位为 0。如果 MCR 位为 0, 该范围中每个字节的信号状态保持不变。

S_BYTE 指向范围中第一个字节, N 指定范围的大小。范围的大小通过指定范围中的位数表示。例如, 要指定 2 个字节的范围, N 的值则输入 16 (16 位)。

注意: N 的值必须是八的倍数(例如, 8、16、24 等)。

S_BYTE 指针必须指向 I/O 外部输入与输出存储区 (P 存储器)。由于 P 存储器按照字节、字或双字被访问, S_BYTE 必须指向整字节地址, 即指针的位号必须为 0。

注意: 过程映像输出表(Q 存储器)中相应位的信号状态也复位为 0。

参数

参数	描述	数据类型	存储区	描述
EN	输入	BOOL	I、Q、M、D、L	使能输入端的信号状态为 1, 激活该功能。
ENO	输出	BOOL	I、Q、M、D、L	如果该功能的执行无错误, 使能输出端的信号状态为 1。
S_BIT	输入	*Pointer	I、Q、M、D	指向范围中的第一个字节

N	输入	INT	I、Q、M、D、L、 P、常数	设置为 1 的范围大小，以 8 的整数倍(例如，8、16、24 等)的位数表示。
* 用于跨区域寄存器间接寻址的双字指针格式				

表 2-3 FC100 参数说明

错误信息

如果发生以下情况之一，范围中每个位的信号状态保持不变，ENO 的信号状态也设置为 0:

- S_BYTE 指针引用了 I/O 外部输入和输出存储区 (P 存储器) 以外的其它存储区。
- S_BYTE 指针引用的地址不是整字节。
- N 的值不是八的倍数。

实例

如果输入 I0.0 的信号状态为 1 (激活)，并且 MCR 位为 1，则执行 RSETI 功能。在本例中，S_BYTE 指向 P2.0 处的第一个字节。N 参数指定要复位 16 个位(2 个字节)。执行该指令后，范围 P2.0 至 P3.7 中每个位的信号状态将复位为 0。

如果该功能的执行未发生错误，ENO 和 Q0.0 的信号状态将设置为 1。

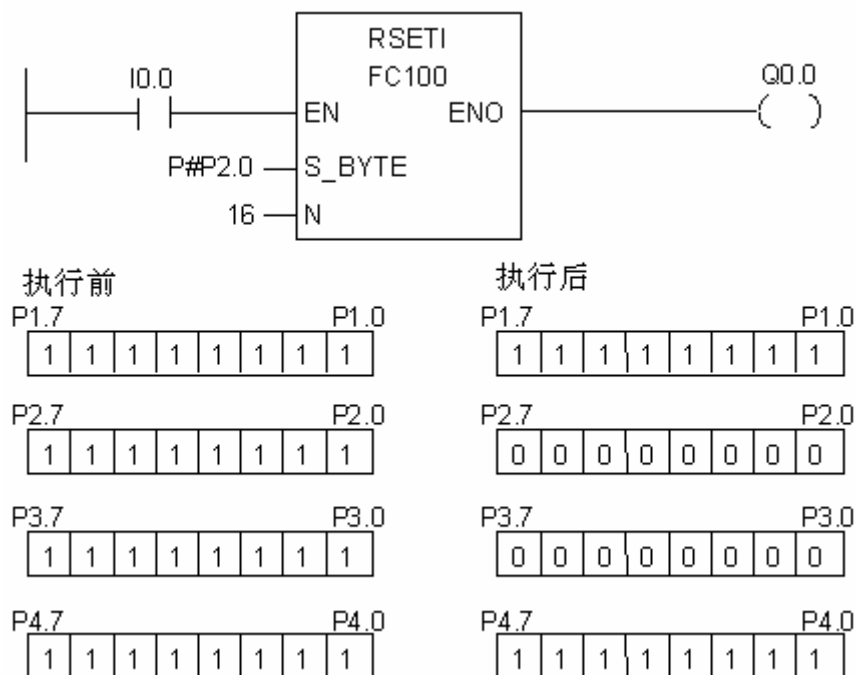


图 2-2: FC100 程序例子

特别说明:

此 FC 主要用于 MCR (主控继电器)，也可单独使用。

2.3.3. 置位范围：FC83

描述

如果 MCR 位为 1，SET 功能将指定范围中每个位的信号状态设置为 1。如果 MCR 位为 0，范围中每个位的信号状态保持不变。范围中要设置的位数由 N 指定，范围的起始位由 S_BIT 指示。

参数

参数	描述	数据类型	存储区	描述
EN	输入	BOOL	I、Q、M、D、L	使能输入端的信号状态为 1，激活该功能。
ENO	输出	BOOL	I、Q、M、D、L	如果该功能的执行无错误，使能输出端的信号状态为 1。
S_BIT	输入	*Pointer	I、Q、M、D	指向范围中的第一位。
N	输入	INT	I、Q、M、D、L、 P、常数	范围中要设置的位数。
* 用于跨区域寄存器间接寻址的双字指针格式				

表 2-4 FC83 参数说明

错误信息

如果 S_BIT 指针指向 I/O 外部输入和输出存储区(P)，则范围中每个位的信号状态保持不变，且 ENO 的信号状态设置为 0。

实例

如果输入 I0.0 的信号状态为 1 (激活)，并且 MCR 位为 1，则执行 SET 功能。在本例中，S_BIT 指向位于 M0.0 的第一位。参数 N 指定 10

个位要置位。执行该指令后，M0.0 至 M1.1 范围中 10 个位的信号状态都设置为 1。

如果该功能的执行未发生错误，ENO 和 Q0.0 的信号状态将设置为 1。

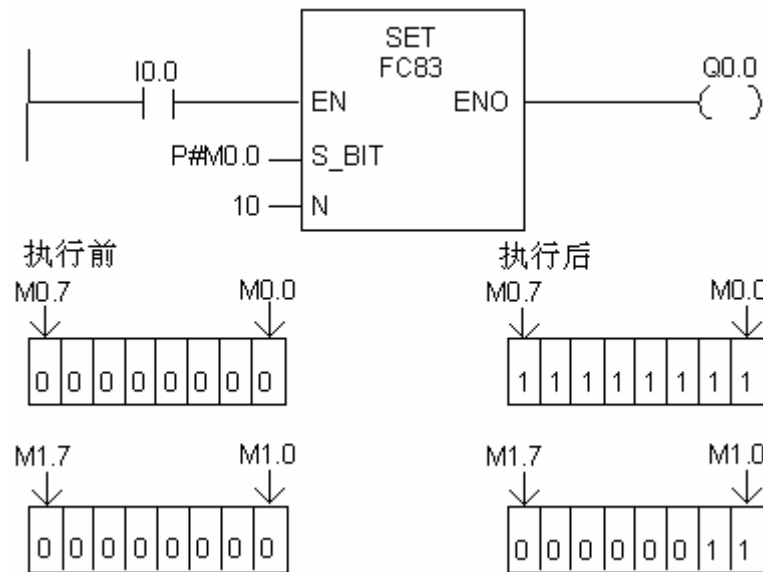


图 2-3: FC83 程序例子

特别说明:

此 FC 主要用于 MCR（主控继电器），也可单独使用。

2.3.4. 置位立即输出范围：FC101

描述

如果 MCR 位为 1，SETI 功能将一定范围内字节的信号状态设置为 1。如果 MCR 位为 0，该范围内每个字节的信号状态保持不变。

S_BYTE 指向范围中的第一个字节，N 指定范围的大小。范围的大小通过指定范围中的位数表示。

例如，要将范围指定为 2 个字节，则 N 值输入 16 (16 位)。

注意: N 值必须是八的倍数(例如，8、16、24 等)。

S_BYTE 指针必须引用外部输入与输出存储区 (P 存储器)。由于 P 存储器按照字节、字或双字被访问，S_BYTE 必须引用整字节地址，即指针的位号必须为 0。

注意: 过程映像输出表(Q 存储器)中相应位的信号状态也复位为 0。

参数

参数	描述	数据类型	存储区	描述
EN	输入	BOOL	I、Q、M、D、L	使能输入端的信号状态为 1，激活该功能。
ENO	输出	BOOL	I、Q、M、D、L	如果该功能的执行无错误，使能输出端的信号状态为 1。
S_BIT	输入	*Pointer	I、Q、M、D	指向范围中的第一位。
N	输入	INT	I、Q、M、D、L、P、常数	范围中要设置的位数。

* 用于跨区域寄存器间接寻址的双字指针格式

表 2-5 FC101 参数说明

错误信息

如果发生以下任何情况，范围中每个位的信号状态保持不变，ENO 的信号状态也设置为 0:

- S_BYTE 指针引用了 I/O 外部输入和输出存储区 (P 存储器) 以外的其它存储区。
- S_BYTE 指针引用了非整字节地址。
- N 的值不是八的倍数。

实例

如果输入 I0.0 的信号状态为 1 (激活)，并且 MCR 位为 1，则执行 SETI 功能。在本例中，S_BYTE 指向 P2.0 处的第一个字节。N 参数指定要置位 16 个位(2 个字节)。执行该指令后，范围 P2.0 至 P3.7 中每个位的信号状态都将设置为 1。

如果该功能的执行未发生错误，ENO 和 Q0.0 的信号状态将设置为 1。

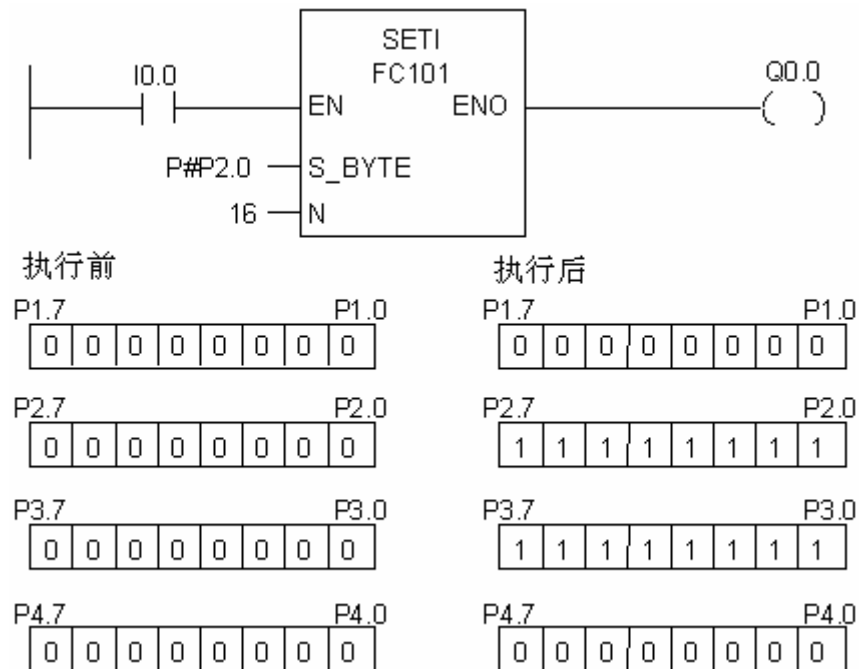


图 2-4: FC101 程序例子

特别说明:

此 FC 主要用于 MCR (主控继电器)，也可单独使用。

2.4. 表功能

2.4.1. 添加到表格：FC84

描述

ATT 功能将 DATA 添加到表格的下一个条目，并使条目数加一。表格由字组成。ATT 功能允许向表格添加条目，以供 FIFO 和 LIFO 功能使用。

- FIFO 或 LIFO 表格中的第一个条目含有表格的最大条目数（表格长度）。
- 表格的第二个条目含有已输入的条目数。
- 表格的第三个条目含有数据的第一个字。

注意：创建表格时必须初始化前两个条目，也就是说，表格必须已经被构建完毕，表格最大长度（最大容纳数据个数）及已输入条目数（实际有效数据个数）应当被正确设置，否则，将影响本程序的正确执行。

参数

参数	描述	数据类型	存储区	描述
EN	输入	BOOL	I、Q、M、D、L	使能输入端的信号状态为 1，激活该功能。
ENO	输出	BOOL	I、Q、M、D、L	如果该功能的执行无错误，使能输出端的信号状态为 1。
DATA	输入	WORD	I、Q、M、D、L、P、常数	要添加到表格的数据。
TABLE	输入	*Pointer	I、Q、M、D	指向 FIFO 或 LIFO 表格的起始位置
* 用于跨区域寄存器间接寻址的双字指针格式				

表 2-6 FC84 参数说明

错误信息

如果条目数等于或大于表格长度，则不能向表格添加数据，并且 ENO 的信号状态设置为 0。

实例

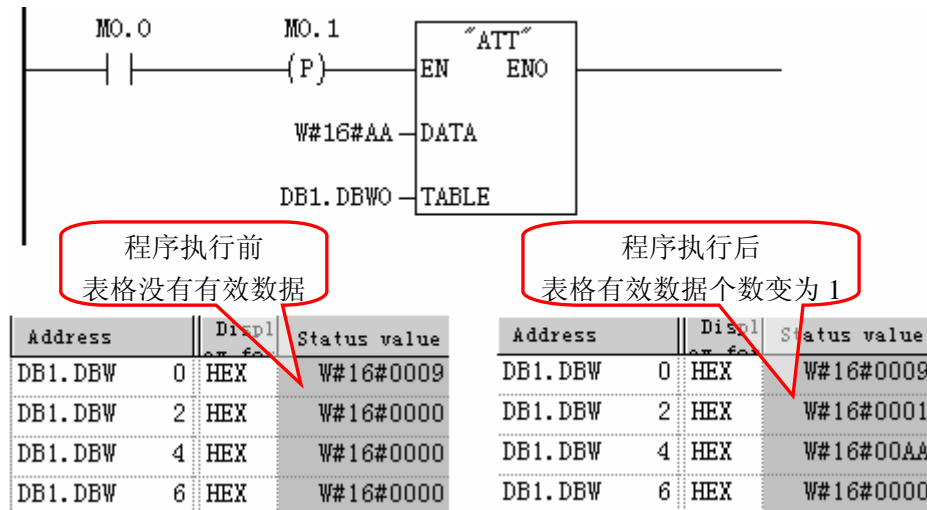


图 2-5: FC84 程序例子

当输入 M0.0 信号状态由 0 变为 1 时，此上升沿调用一次 FC84。将 DATA 作为第 1 个条目添加到表格中，同时条目数从 0 增加到 1。

注意：由于每次调用 ATT 功能，都将向表格增加一个数据，所以如果连续调用此功能，表格很快被填满。

2.4.2. 先进/先出取出表格数据：FC85

描述

FIFO 功能将返回 FIFO 表格最旧的条目作为功能值。条目的数量将减少一个，如果还有剩余条目，这些条目将在表格中下移。

FIFO 表格由字组成。可使用 ATT 功能向 FIFO 表格添加条目。

- 表格的第一个条目含有该表格的最大条目数(表格长度)。
- 表格的第二个条目含有已输入的条目数。
- 表格的第三个条目含有数据的第一个字。

注意：创建表格时必须初始化前两个条目，也就是说，表格必须已经被构建完毕，表格最大长度（最大容纳数据个数）及已输入条目数（实际有效数据个数）应当被正确设置，否则，将影响本程序的正确执行。

参数

参数	描述	数据类型	存储区	描述
EN	输入	BOOL	I、Q、M、D、L	使能输入端的信号状态为 1，激活该功能。
ENO	输出	BOOL	I、Q、M、D、L	如果该功能的执行无错误，使能输出端的信号状态为 1。

TABLE	输入	*Pointer	I、Q、M、D	指向 FIFO 表格的起始处
RET_VAL	输出	WORD	I、Q、M、D、L、P	FIFO 表格最旧的条目
* 用于跨区域寄存器间接寻址的双字指针格式				

表 2-7 FC85 参数说明

错误信息

如果 FIFO 表格是空的(条目数 = 0)，则不改变 RET_VAL，且 ENO 的信号状态设置为 0。

实例

当输入 M1.0 信号状态由 0 变为 1 时，此上升沿调用一次 FC85，执行 FIFO 功能。在本例中，表格中最旧的条目将作为功能值(MW2)返回。条目数从 3 递减为 2，其余条目在表中下移。

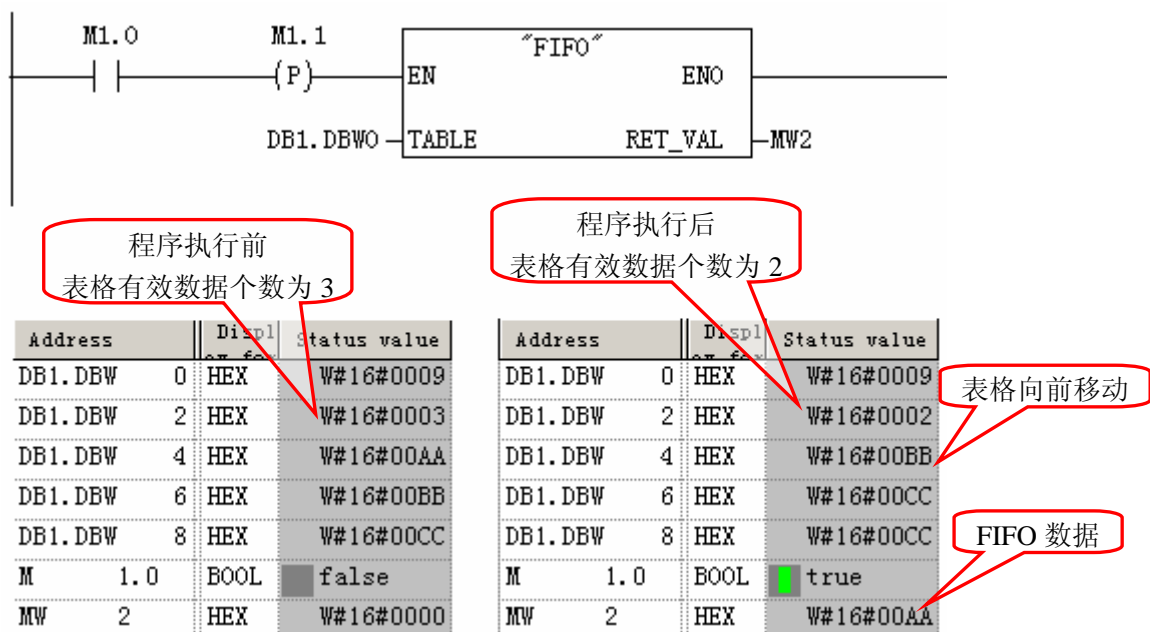


图 2-6: FC85 程序例子

注意：由于每次调用 FC85，都从表格取出一个数据，所以如果连续调用此功能，表格很快被取空。

2.4.3. 查表: FC86

描述

TBL_FIND 功能用于搜索存储器块中与众不同或不一致的模式。该功能在源模式(PATR)和源表格(SRC)条目之间执行指定的比较命令(CMD)。此功能会在表格中查找(INDX 索引条目之后) 满足比较命令的下个条目，并将其条目号置于 INDX 中。如果未找到匹配值，INDX 将指向表格末尾之后的位置，并关闭该功能的输出。

- 如果 CMD = 1，功能会搜索与 PATRN 值相等的第一个值。
- 如果 CMD = 2，该功能会搜索与 PATRN 值不等的第一个值。

- 表格的第一个条目含有该表格的最大条目数(表格长度)。
- 表格的第二个条目含有第一个表格值。

注意:

- 创建表格时必须初始化第 1 个条目 (实际有效数据个数), 也就是说, 表格必须已经被构建完毕。否则, 将影响本程序的正确执行。
- 注意此表格的条目数不同于 FIFO 及 LIFO 表格的结构, 此表格的第 2 个条目即为数据。

参数

参数	描述	数据类型	存储区	描述
EN	输入	BOOL	I、Q、M、D、L	使能输入端的信号状态为 1, 激活该功能。
ENO	输出	BOOL	I、Q、M、D、L	如果该功能的执行无错误, 使能输出端的信号状态为 1。
SRC	输入	*Pointer	I、Q、M、D	指向表格的起始处
PATRN	输入	*Pointer	I、Q、M、D	指向要搜索的模式
CMD	输入	BYTE	I、Q、M、D、L、P	指定命令类型: B#16#01 = 相等 B#16#02 = 不相等
E_TYPE	输入	BYTE	I、Q、M、D、L、P	指定表格条目的数据类型: B#16#02=BYTE, B#16#04= WORD B#16#05= INT, B#16#06 = DWORD B#16#07= DINT, B#16#08 = REAL
RET_VAL	输出	WORD	I、Q、M、D、L、P	如果该指令的执行没有错误, 将返回值 W#16#0000。对于 W#16#0000 以外的其它值, 参见"错误信息"。
INDX	输入/ 输出	WORD	I、Q、M、D、L	在表格中建立索引, 可提供: 输入: 搜索的起始条目号 输出: 匹配值的条目号
* 用于跨区域寄存器间接寻址的双字指针格式				

表 2-8 FC86 参数说明

错误信息

如果发生以下任何情况, 表格值不会变化。ENO 的信号状态设置为 0, 并相应地设置返回值:

RET_VAL 解释

W#16#0007 INDX 超出表格范围

W#16#0008 未找到匹配值。

W#16#0009 E_TYPE 和/或 CMD 无效。

实例

如果输入 M1.2 的信号状态为 1 (激活), 则执行 TBL_FIND 指令。在本例中, 由于 E_TYPE = 4,

表格中的数据从 SRC 指向的条目位置开始以字为单位存储。这些字与存储在 PATRN 所指位置的
模式值 AAAA 进行比较。由于 CMD 值为 1, 所以将寻找 SRC 中与 PATRN 值相等的第一个表格值。
INDX 值指向搜索开始的条目。执行该指令后, INDX 值将给出表格中满足比较命令的条目号。
如果该功能的执行没有错误, RET_VAL 设置为 W#16#0000。每次执行 MOVE 指令时都将初始化
MW4 的值。

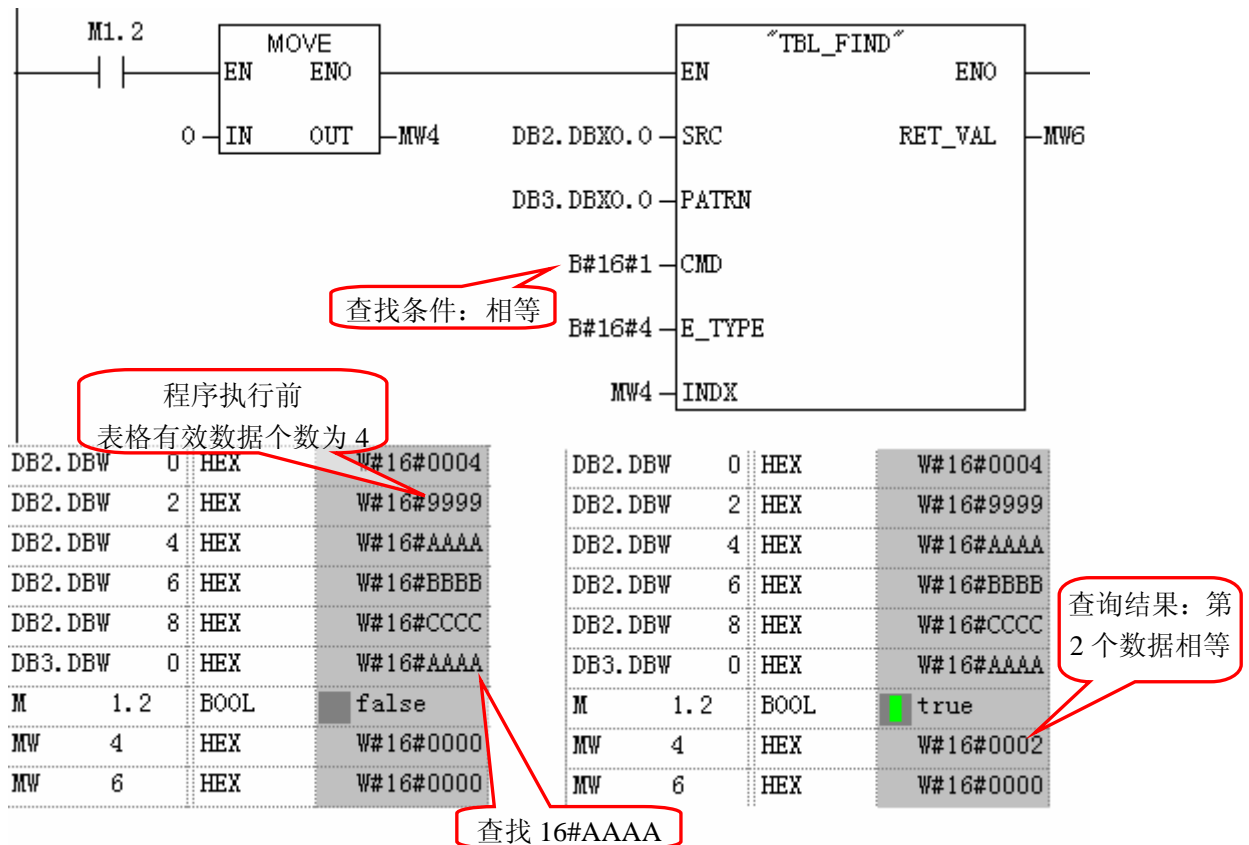


图 2-7: FC85 程序例子

2.4.4. 后进/先出取出表格数据: FC87

描述

LIFO 功能将返回 LIFO 表格的最新 (最近) 条目作为功能值, 并将条目数减一。LIFO 表格由字组成。可以使用 ATT 功能将条目添加到 LIFO 表格中。

- 表格的第一个条目含有该表格的最大条目数(表格长度)。
- 表格的第二个条目含有已输入的条目数。
- 表格的第三个条目含有数据的第一个字。

注意: 创建表格时必须初始化前两个条目, 也就是说, 表格必须已经被构建完毕, 表格最大长度 (最大容纳数据个数) 及已输入条目数 (实际有效数据个数) 应当被正确设置, 否则, 将影响本

程序的正确执行。

参数

参数	描述	数据类型	存储区	描述
EN	输入	BOOL	I、Q、M、D、L	使能输入端的信号状态为 1，激活该功能。
ENO	输出	BOOL	I、Q、M、D、L	如果该功能的执行无错误，使能输出端的信号状态为 1。
TABLE	输入	*Pointer	I、Q、M、D	指向 LIFO 表格的起始处
RET_VAL	输出	WORD	I、Q、M、D、L、P	LIFO 表格的最新条目

* 用于跨区域寄存器间接寻址的双字指针格式

表 2-9 FC87 参数说明

错误信息

如果 LIFO 表格是空的(条目数 = 0)，则 RET_VAL 不会改变，且 ENO 的信号状态设置为 0。

实例

当输入 M1.3 信号状态由 0 变为 1 时，此上升沿调用一次 FC87，执行 LIFO 功能。在本例中，表格中最新的条目将作为功能值(MW8)返回。条目数从 4 递减为 3，其余条目在表中不变。

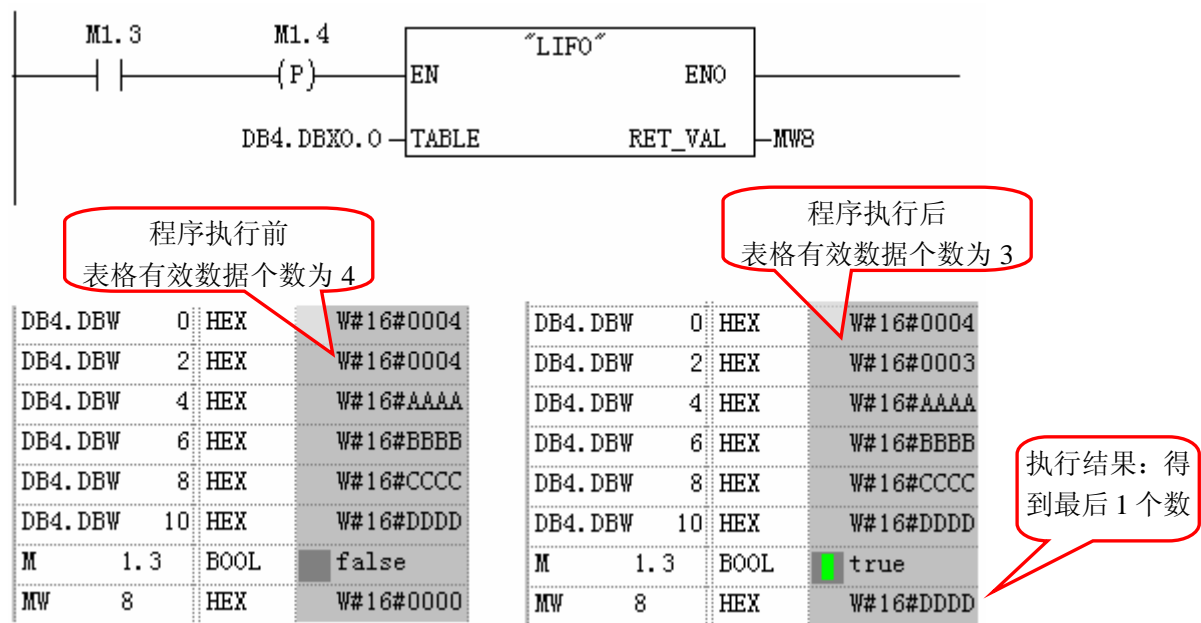


图 2-8: FC87 程序例子

注意：由于每次调用 FC87，都从表格取出一个数据，所以如果连续调用此功能，表格很快被取空。

2.4.5. 表格：FC88

描述

TBL 功能对源表格执行指定的操作 (由 CMD 指定) , 并将结果写入同一表格条目。

- 表格的第一个条目含有该表格的最大条目数 (表格长度)。
- 表格的第二个条目含有第一个表格值。
- 如果 E_TYPE 设置为实型, 则补码的 CMD 值无效。

注意: 创建表格时, 必须初始化第一个条目。

参数

参数	描述	数据类型	存储区	描述
EN	输入	BOOL	I、Q、M、D、L	使能输入端的信号状态为 1, 激活该功能。
ENO	输出	BOOL	I、Q、M、D、L	如果该功能的执行无错误, 使能输出端的信号状态为 1。
SRC	输入	*Pointer	I、Q、M、D	指向表格的起始处
CMD	输入	BYTE	I、Q、M、D、L、P	指示要执行的命令类型。有效命令及其值: B#16#03 = 补码 B#16#04 = 清除 B#16#05 = 取反 B#16#06 = 平方根
E_TYPE	输入	BYTE	I、Q、M、D、L、P	指示表格条目的数据类型。有效数据类型: B#16#04=WORD B#16#05=INT B#16#06=DWORD B#16#07= DINT B#16#08 = REAL
RET_VAL	输出	WORD	I、Q、M、D、L、P	如果该指令的执行没有错误, 将返回数值 W#16#0000。对于 W#16#0000 以外的其它值, 参见"错误信息"。
* 用于跨区域寄存器间接寻址的双字指针格式				

表 2-10 FC88 参数说明

错误信息

如果 CMD 或 E_TYPE 无效, 或者 CMD 和 E_TYPE 互不一致, 或者以上两种情况同时存在, 表格值将不会改变。ENO 的信号状态将设置为 0, RET_VAL 将等于 W#16#0008。

实例

当输入 M1.5 信号状态由 0 变为 1 时, 此上升沿调用一次 FC88, 第一个表格条目中的表格长度值为 5, 因此将对下面 5 个表格条目取平方根, 并放回原位置。

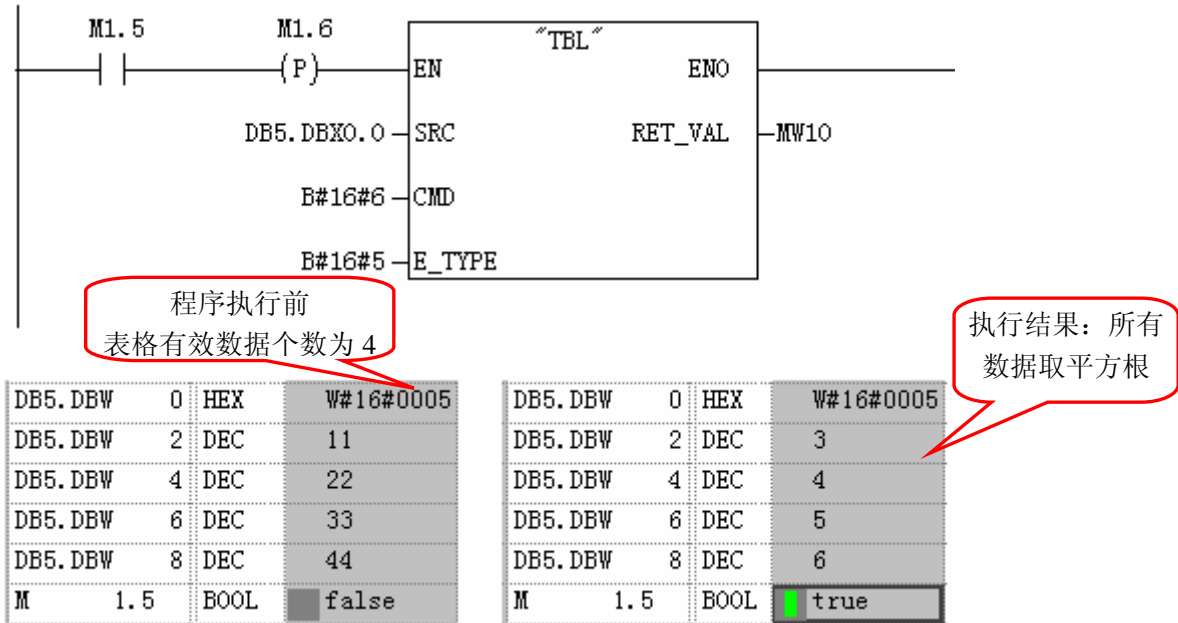


图 2-9: FC88 程序例子

注意: 由于每次调用 FC88，都将对完整表格进行一次操作，所以如果连续调用此功能，表格将出现问题，例如：对一个比较大的正整数连续取平方根操作，最后结果将为 1，如果 CMD=5，则数据将在正负数值之间不断变化。

2.4.6. 将表格移动到字: FC89

描述

TBL_WRD 功能可将 INDX 指示的条目从 SRC 表格复制到 DEST 指向的条目，然后增加 INDX (只要 INDX 小于表格第一个字 SRC [0] 给出的表格长度)。如果调用此指令时，INDX 的设置位于最后一个表格条目处，执行该指令后，Q 输出位将设置为 0。

- 表格的第一个条目含有该表格的最大条目数(表格长度)。
- 表格的第二个条目含有第一个表格值。

注意: 创建表格时，必须初始化第一个条目。

参数

参数	描述	数据类型	存储区	描述
EN	输入	BOOL	I、Q、M、D、L	使能输入端的信号状态为 1，激活该功能。
ENO	输出	BOOL	I、Q、M、D、L	如果该功能的执行无错误，使能输出端的信号状态为 1。
SRC	输入	*Pointer	I、Q、M、D	指向表格的起始处
DEST	输入	*Pointer	I、Q、M、D	指向目标的起始位置。

E_TYPE	输入	BYTE	I、Q、M、D、L、P	指示表格条目的数据类型。有效数据类型有： B#16#04=WORD B#16#05= INT B#16#06=DWORD B#16#07= DINT B#16#08 = REAL
RET_VAL	输出	WORD	I、Q、M、D、L、P	如果该指令的执行没有错误，将返回值 W#16#0000。对于 W#16#0000 以外的其它值，参见"错误信息"。
Q	输出	BOOL	Q、M、D、L	如果调用该功能时，INDX 变量含有表格的最后条目，则输出 0。
INDX	输入/ 输出	WORD	I、Q、M、L	要移动条目的条目号。
* 用于跨区域寄存器间接寻址的双字指针格式				

表 2-11 FC89 参数说明

错误信息

如果发生以下任何情况，则不执行该功能。ENO 的信号状态设置为 0，并相应地设置返回值：

RET_VAL 解释

W#16#0007 索引为 0。

W#16#0008 E_TYPE 无效。

W#16#0009 索引超出表格末尾。

实例

当输入 M20.0 信号状态由 0 变为 1 时，此上升沿调用一次 FC89，由于 E_TYPE 等于 4，存储在表格中从 SRC 指向的条目开始的字数据将复制到 DEST 指向的条目。INDX 值指向要移动的表格条目。成功执行该指令后，INDX 将从已移动条目自动向后增加一个条目。在本例中，调用该指令时，INDX 不会设置为表格的最后条目，因此指令执行后，Q 设置为 1。

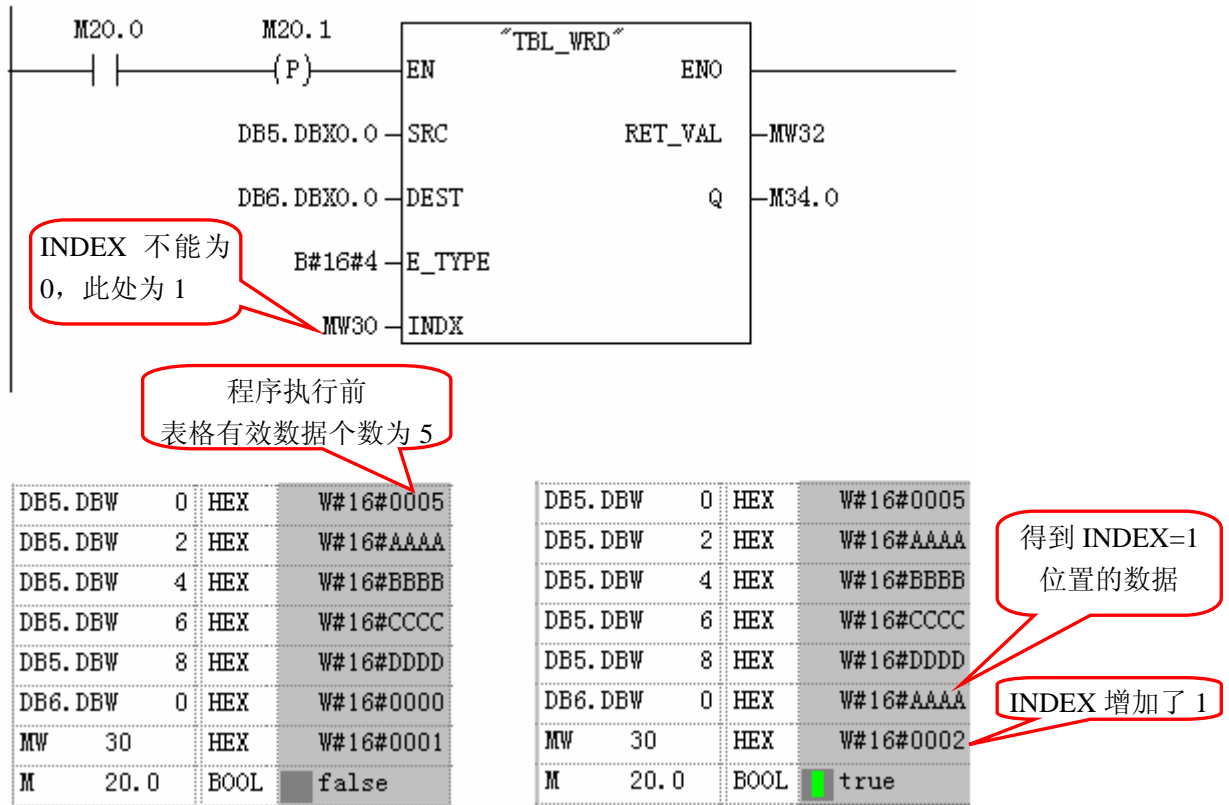


图 2-10: FC89 程序例子

注意: 由于每次调用 FC89, 都将取出当前 INDEX 位置的数据, 并将此数据复制到目标地址, INDEX 增加 1, 所以如果连续调用此功能, 程序将依次将所有数据复制到目的地址, 最后 INDEX 超出表格范围, RET_VAL=9 (超出表格范围) 的数据将超出表格位置。

2.4.7. 字到表格: FC91

描述

WRD_TBL 功能在源数据 (由 SRC 指定) 和由 INDX 指示的偏移量处的表格条目之间执行指定的命令 (CMD), 然后增加 INDX 的值 (只要 INDX 小于表格长度)。

- 表格的第一个条目含有该表格的最大条目数(表格长度)。
- 表格的第二个条目含有第一个表格值。
- 如果 E_TYPE 为 REAL, 则 CMD 只能为"移动"。

注意: 创建表格时, 必须初始化第一个条目。

参数

参数	描述	数据类型	存储区	描述
EN	输入	BOOL	I、Q、M、D、L	使能输入端的信号状态为 1, 激活该功能。

ENO	输出	BOOL	I、Q、M、D、L	如果该功能的执行无错误，使能输出端的信号状态为 1。
SRC	输入	*Pointer	I、Q、M、D	指向源数据
TABLE	输入	*Pointer	I、Q、M、D	指向表格的起始处
CMD	输入	BYTE	I、Q、M、D、L、P	指示要执行的命令类型。有效命令及其值： B#16#0E=移动 B#16#07=与运算 B#16#08=或运算 B#16#09=异或运算
E_TYPE	输入	BYTE	I、Q、M、D、L、P	指示表格条目的数据类型。有效数据类型： B#16#04=WORD B#16#05=INT B#16#06=DWORD B#16#07=DINT B#16#08=REAL
RET_VAL	输出	WORD	I、Q、M、D、L、P	如果该指令的执行没有错误，将返回值 W#16#0000。对于 W#16#0000 以外的其它值，参见"错误信息"。
Q	输出	BOOL	Q、M、D、L	如果 INDX 含有表格最后的条目号，则指示 0。
INDX	输入/ 输出	WORD	I、Q、M、D、L	要对其进行操作的条目的条目号。
* 用于跨区域寄存器间接寻址的双字指针格式				

表 2-12 FC91 参数说明

错误信息

如果发生以下任何情况，则不执行该功能。ENO 的信号状态设置为 0，并相应地设置返回值：

RET_VAL 解释

W#16#0007 索引为 0。

W#16#0008 CMD 或 E_TYPE 无效，或者 CMD 和 E_TYPE 不一致。

W#16#0009 索引超出表格末尾。

实例

FC91 的执行与 FC89 相反，将数据通过 4 种运算方式（移动、与、或、异或）与表格中的数据进行运算。如果 E_TYPE 的值为 6，双字数据将以 TABLE 指向的存储器位置为起点存储在表格中。第一个字中的表格长度指示表格含有 3 个双字。INDX 值指向要操作的表格条目。由于 CMD 值为 8，该指令对 INDX 指向的值执行或运算。由于 INDX 为 2，将对第二个双字 (66665544) 和 SRC 指向的值 (11111111) 执行或运算。执行该指令后，或运算命令的结果(77775555)将写回表格，INDX 也将自动增加一个条目。如果调用此指令时，INDX 设置为最后一个表格条目，则指令执行后，Q 输出位将设置为 0。在本例中，调用该指令时，INDX 未设置为最后一个表格条目，因此在执行后

Q 设置为 1。

如果该功能的执行没有错误，ENO 和 Q0.0 的信号状态将设置为 1，RET_VAL 将等于 W#16#0000。

2.4.8. 关联数据表：FC103

描述

CDT 功能将输入值 (IN) 与已有输入表 (IN_TBL) 中的值相比较，然后找出第一个大于或等于输入值的值。如果找到了该值，则通过该值的索引将输出表(OUT_TBL)中的相应值复制到输出值 (OUT)中。

- 输入表值应该按升序排列。即，最小的值位于第一个表格条目，最大的值位于最后一个表格条目。
- 输入值的大小、表格单元以及输出值根据 E_TYPE 确定。
- 表的第一个条目含有该表格的条目数(表格长度)。
- 表的第二个条目含有第一个表格值。
- 两个表中的单元数必须相等且大于零。

注意：创建每个表时，必须初始化第一个条目。

参数

参数	描述	数据类型	存储区	描述
EN	输入	BOOL	I、Q、M、D、L	使能输入端的信号状态为 1，激活该功能。
ENO	输出	BOOL	I、Q、M、D、L	如果该功能的执行无错误，使能输出端的信号状态为 1。
IN_TBL	输入	*Pointer	I、Q、M、D	指向输入表的起始位置
OUT_TBL	输入	*Pointer	I、Q、M、D	指向输出表的起始位置
IN	输入	*Pointer	I、Q、M、D	指向输入表
OUT	输入	*Pointer	I、Q、M、D	指向输出表
E_TYPE	输入	BYTE	I、Q、M、D、L、P	指示表格条目的数据类型。有效数据类型： B#16#05=INT B#16#07= DINT B#16#08=REAL
RET_VAL	输出	WORD	I、Q、M、D、L、P	如果该功能执行成功，将返回 W#16#0000。对于 W#16#0000 以外的其它值，参见"错误信息"。
* 用于跨区域寄存器间接寻址的双字指针格式				

表 2-13 FC103 参数说明

错误信息

如果发生以下任何情况，则不执行该功能。ENO 的信号状态设置为 0，并相应地设置返回值：

RET_VAL 解释

W#16#0001 为功能的参数指定的存储器类型无效。

W#16#0002 E_TYPE 无效。

W#16#0003 输入和输出表长度不匹配。

W#16#0004 表格长度为零。

W#16#0007 IN_TBL 中没有大于或等于输入值的值。

实例

如果 I0.0 的信号状态为 1 (激活), 则执行 CDT 功能。在本例中, 如每个表格中第一个字所示, IN_TBL 和 OUT_TBL 都含有 5 个表格条目。如 E_TYPE 如示, 表格值的数据类型为整型, IN 的值为 22。大于或等于 22 的 IN_TBL 值为 64, 索引为 5。OUT_TBL 中的相关值为 25。因此, 将值 25 写入 OUT。

如果该功能的执行没有错误, ENO 和 Q0.0 的信号状态将设置为 1, RET_VAL 等于 W#16#0000。

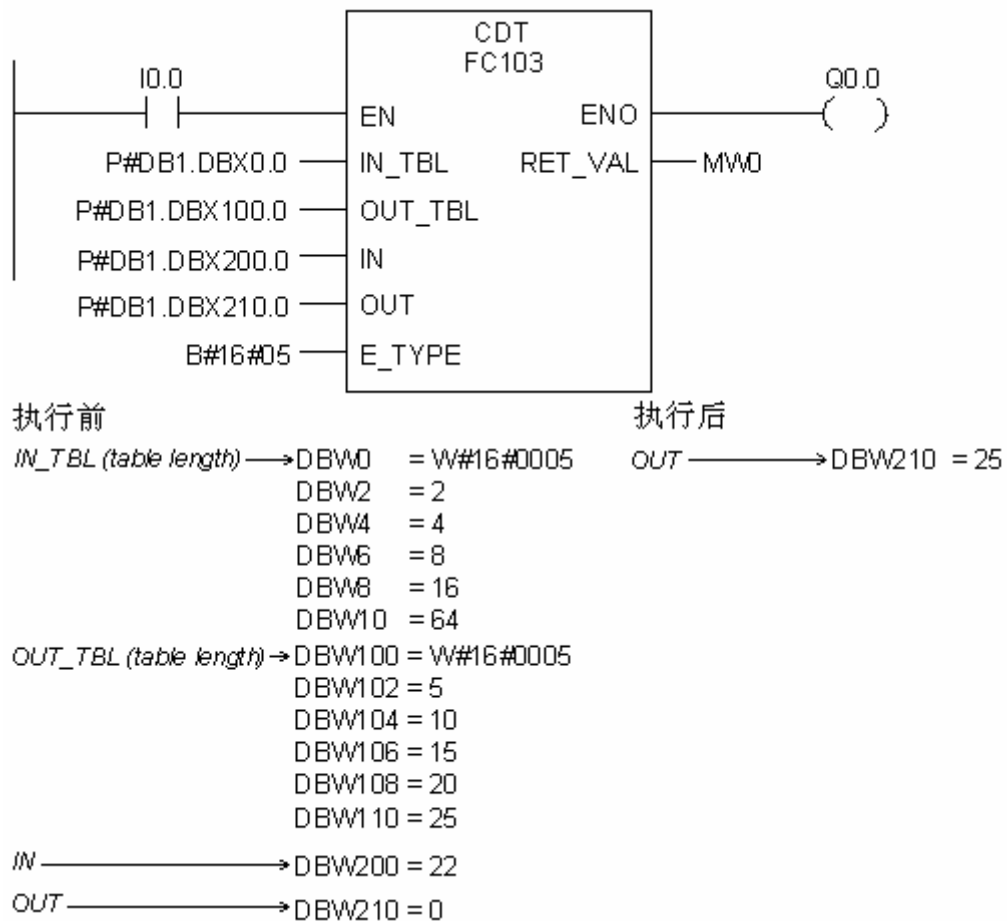


图 2-11: FC103 程序例子

2.4.9. 表到表操作：FC104

描述

TBL_TBL 功能在两个源表 (TBL1 和 TBL2) 的相应条目之间执行指定的命令 (CMD)，并将结果写入目标表 (DEST_TBL) 相应的条目中。

- INT、DINT 或 REAL 数据类型仅对数学运算有效。
- 表格的第一个条目含有该表的条目数(表长度)。
- 所有表中的条目数必须相等且大于零。

注意：创建每个表时，必须初始化第一个条目。

参数

参数	描述	数据类型	存储区	描述
EN	输入	BOOL	I、Q、M、D、L	使能输入端的信号状态为 1，激活该功能。
ENO	输出	BOOL	I、Q、M、D、L	如果该功能的执行无错误，使能输出端的信号状态为 1。
TBL1	输入	*Pointer	I、Q、M、D	指向第一个源表的起始处
TBL2	输入	*Pointer	I、Q、M、D	指向第二个源表的起始处
DEST_TBL	输入	*Pointer	I、Q、M、D	指向目标表的起始处
CMD	输入	BYTE	I、Q、M、D、L、P	指示要执行的命令类型。有效命令及其对应值： B#16#07=与运算 B#16#08=或运算 B#16#09=异或运算 B#16#0a=加 B#16#0b=减 B#16#0c=乘 B#16#0d=除
E_TYPE	输入	BYTE	I、Q、M、D、L、P	指示表格条目的数据类型。有效数据类型： B#16#04=WORD B#16#05=INT B#16#05=DWORD B#16#07=DINT B#16#08=REAL
RET_VAL	输出	WORD	I、Q、M、D、L、P	如果该功能执行成功，将返回值 W#16#0000。对于 W#16#0000 以外的其它值，参见"错误信息"。
* 用于跨区域寄存器间接寻址的双字指针格式				

表 2-14 FC104 参数说明

错误信息

如果发生以下任何情况，则不执行该功能。ENO 的信号状态设置为 0，并相应地设置返回值：

RET_VAL 解释

W#16#0001 为功能的参数指定的存储器类型无效。

W#16#0002 E_TYPE 无效。

W#16#0003 输入和输出表长度不匹配。

W#16#0004 表格长度为零。

W#16#0005 E_TYPE 和 CMD 不一致。

W#16#0006 CMD 无效。

实例

如果 IO.0 的信号状态为 1 (激活), 则执行 TBL_TBL 功能。在本例中, 如每个表第一个字所示, 所有表都含有 3 个表格条目。如

E_TYPE 所示, 表中值的数据类型为字。如 CMD 所示, 要对 TBL1 和 TBL2 执行的命令是与运算。

如果该功能的执行没有错误, ENO 和 Q0.0 的信号状态将设置为 1, RET_VAL 等于 W#16#0000。

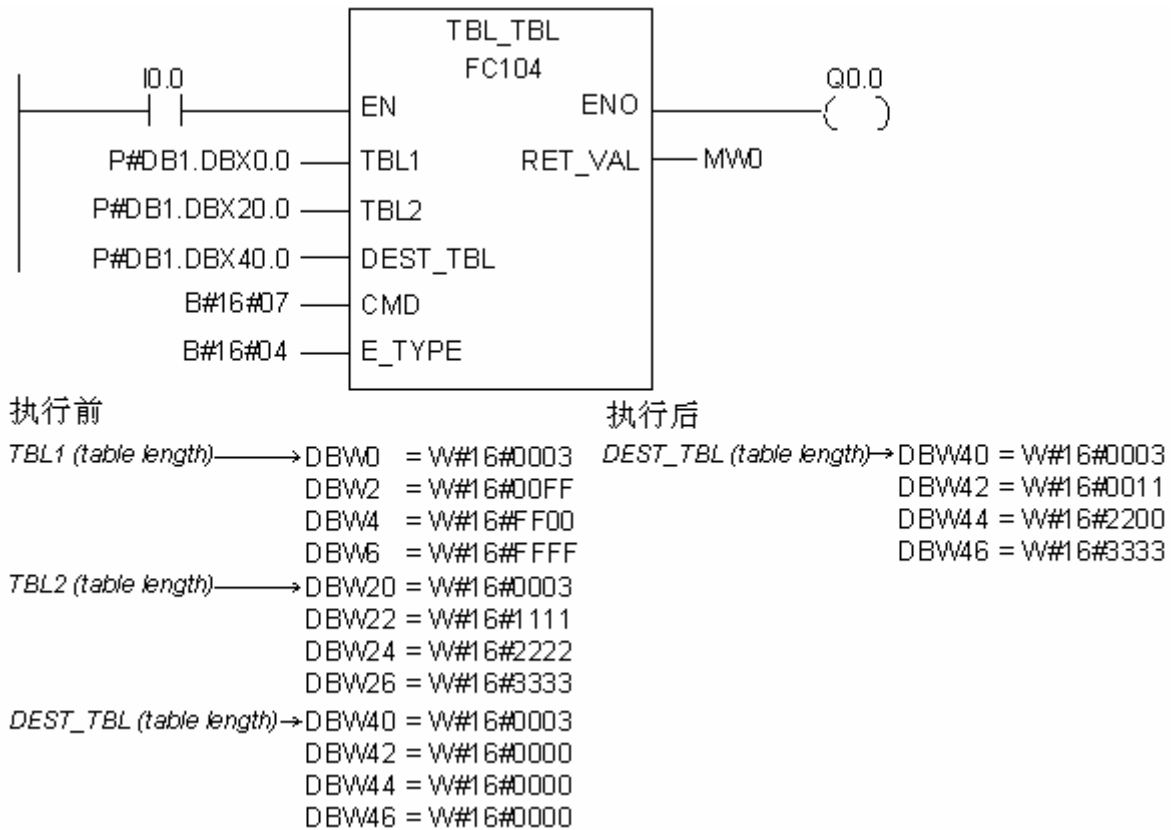


图 2-12: FC104 程序例子

2.5. 移位功能

2.5.1. 字移位寄存器: FC90

描述

WSR 功能将数据从指定的源位置移位到移位寄存器。每个值都将移动到下个位置。LENGTH 指定要移位的位置数。移位后，移位寄存器最后一个位置中的数据将会丢失。每次执行该指令时，都会从源位置(S_DATA)读取新数据；当 RESET 输入设置为 0 时，此数据将移入移位寄存器的起始位置(START)。如果 RESET 输入设置为 1，执行该指令时，寄存器位置将设置为零。移位寄存器为空或清零(即，复位或全部移入零)时，将使能输出 Q。

参数

参数	描述	数据类型	存储区	描述
EN	输入	BOOL	I、Q、M、D、L	使能输入端的信号状态为 1，激活该功能。
ENO	输出	BOOL	I、Q、M、D、L	如果该功能的执行无错误，使能输出端的信号状态为 1。
RESET	输入	BOOL	I、Q、M、D、L	设置为 1 时，复位移位寄存器
S_DATA	输入	*Pointer	I、Q、M、D	指向要插入表格的源数据元素
START	输入	*Pointer	I、Q、M、D	指向表格的起始处
LENGTH	输入	WORD	I、Q、M、D、L、P	要移位的条目数
E_TYPE	输入	BYTE	I、Q、M、D、L、P	指示表格条目的数据类型。有效数据类型为： B#16#04=WORD B#16#05= INT B#16#06=DWORD B#16#07= INT B#16#08=REAL
Q	输出	BOOL	Q、M、D、L	如果 RESET 为激活状态 (1) 或者所有移入的位都为零，则指示 0。
* 用于跨区域寄存器间接寻址的双字指针格式				

表 2-15 FC90 参数说明

错误信息

如果使用的 E_TYPE 无效，则不执行该功能，并且 ENO 的信号状态设置为 0。

实例

当输入 M20.2 信号状态由 0 变为 1 时，此上升沿调用一次 FC90，由于 E_TYPE 等于 5，程序认为目的地址区数据类型为 INT，LENGTH 参数设置为 4，表示将从 START 指针的第一个字开始，移位 4 个字位置。当表格中的第一个值移到下一位置后，第一个位置将由 S_DATA 指针指向的数据填充。表格的最后一个值将会丢失。每当 RESET 输入设置为 1 时，表格中的位置都将设置为零，而不发生移位。

如果该功能的执行未发生错误，ENO 和 Q0.0 的信号状态将设置为 1。

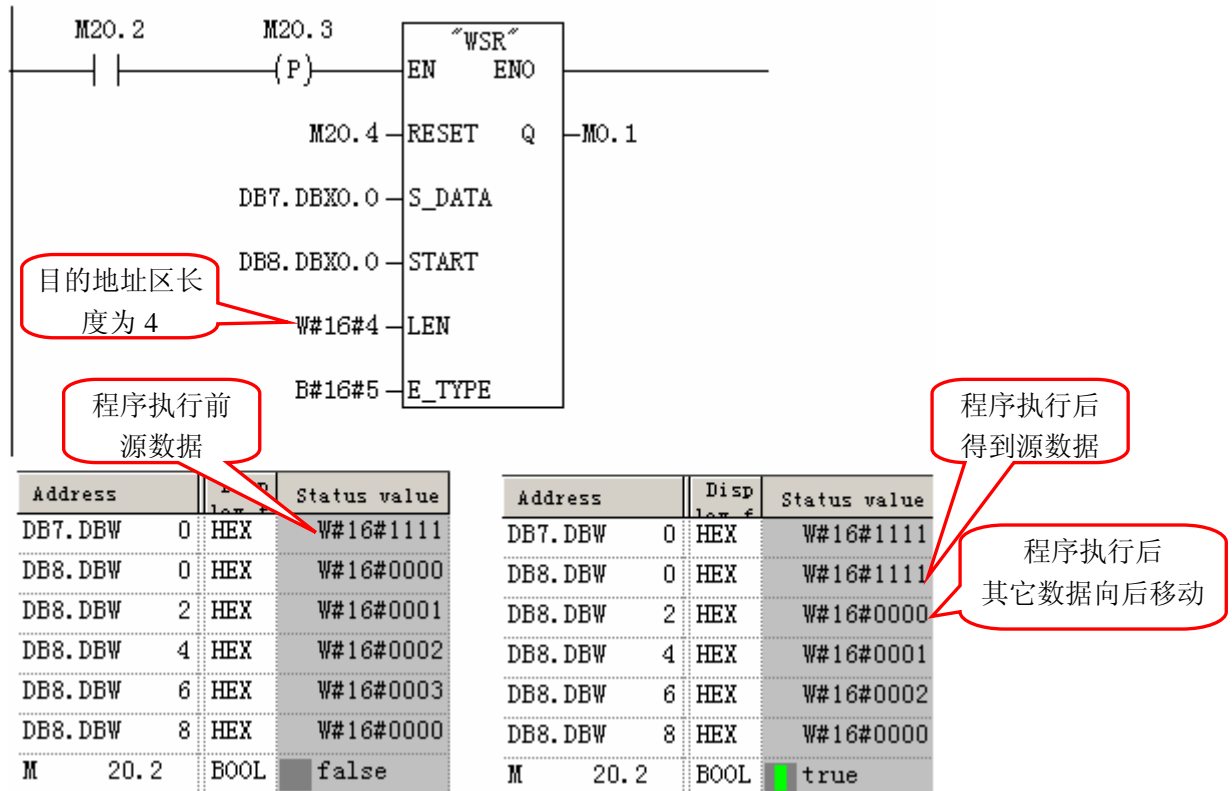


图 2-13: FC90 程序例子

注意: FC90 使用的目的地址区不同于 FIFO,LIFO 表格的结构, 用户需要注意这一点。如果连续调用此功能, 程序将重复把源地址区的数据移位到目的地址区, 如果源地址区数据保持不变, 例如: 16#1111, 目的地址区已存在数据将全部被移出, 并被全部填充为 16#1111。

2.5.2. 位移寄存器: FC92

描述

SHRB 功能将一位数据从 DATA 中指示的源位置移入移位寄存器。每次执行该指令时, 都会从源位置(DATA)读取新数据, 并在 RESET 输入的信号状态为 0 时, 将此数据移入移位寄存器的起始位置 (S_BIT)。所有后续位将移动一位。移位后, 最后位置(S_BIT+N) 中的位将会丢失。RESET 输入设置为 1 时, 表格中的位置将设置为 0, 而不移位。

参数

参数	描述	数据类型	存储区	描述
EN	输入	BOOL	I、Q、M、D、L	使能输入端的信号状态为 1, 激活该功能。
ENO	输出	BOOL	I、Q、M、D、L	如果该功能的执行无错误, 使能输出端的信号状态为 1。
DATA	输入	BOOL	I、Q、M、D、L	源数据位

RESET	输入	BOOL	I、Q、M、D、L	设置为 1 时，将复位移位寄存器
S_BIT	输入	*Pointer	I、Q、M、D	指向移位寄存器的起始位
N	输入	WORD	I、Q、M、D、L、P	移位寄存器的长度 (要移位的位数)
* 用于跨区域寄存器间接寻址的双字指针格式				

表 2-16 FC92 参数说明

错误信息

此功能不检测任何错误情况。

实例

当输入 M20.2 信号状态由 0 变为 1 时，此上升沿调用一次 FC92，在本例中，域 N 设置为 14 (十六进制表示为"E")，表示从 S_BIT 指针位置的第一位开始，移位 14 位。移位后，第一个位置将填充输入 DATA 指示的数据。最后一位的值将会丢失。

如果该功能的执行未发生错误，ENO 和 Q0.0 的信号状态将设置为 1。

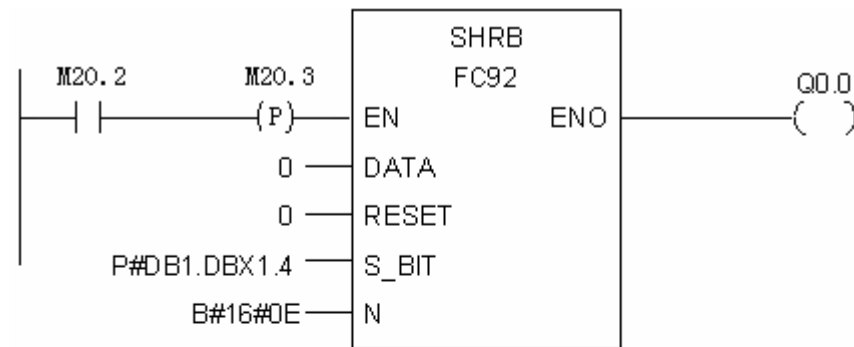


图 2-14: FC92 程序例子

注意：FC92 的使用注意事项可以参考 FC90 的注意事项。

2.6. 移动功能及功能块

2.6.1. 间接块移动：FC81

描述

IBLKMOV 功能可用于将一个由字节、字、整型、双字或长整型组成的数据块从源块移动到目标块。要移动的单元数由 LENGTH 确定，单元长度由 E_TYPE 确定。S_DATA 和 D_DATA 指针代表指向源和目标数据起始位置的指针位置。由于该功能使用间接手段来确定要移动的数据，所以此功能称为间接移动。

参数

参数	描述	数据类型	存储区	描述
EN	输入	BOOL	I、Q、M、D、L	使能输入端的信号状态为 1，激活该功能。
ENO	输出	BOOL	I、Q、M、D、L	如果该功能的执行无错误，使能输出端的信号状态为 1。
S_DATA	输入	*Pointer	I、Q、M、D	指向标识源数据起始位置的指针
LENGTH	输入	*Pointer	I、Q、M、D	指向要移动块的长度
D_DATA	输入	*Pointer	I、Q、M、D	指向标识目标数据起始位置的指针
E_TYPE	输入	BYTE	I、Q、M、D、L	指示数据类型。有效的 E_TYPE 值如下： B#16#02=BYTE B#16#04=WORD B#16#05=INT B#16#06=DWORD B#16#07=DINT B#16#08=REAL
* 用于跨区域寄存器间接寻址的双字指针格式				

表 2-17 FC81 参数说明

错误信息

如果输入了无效的 E_TYPE，则该功能不能执行，而且 ENO 的信号状态将设置为 0。

实例

如果输入 I0.0 的信号状态为 1 (激活)，则执行 IBLKMOV 功能。S_DATA 指向 DB1.DBX0.0，DB1.DBX0.0 中包含指针 DB1.DBX50.0 (源数据的起始位置)。D_DATA 指向 DB1DBX20.0，DB1DBX20.0 中含有指针 DB2.DBX10.0 (目标数据的起始位置)。执行该功能后，将移动一个双字块。

如果该功能的执行未发生错误，ENO 和 Q1.0 的信号状态将设置为 1。

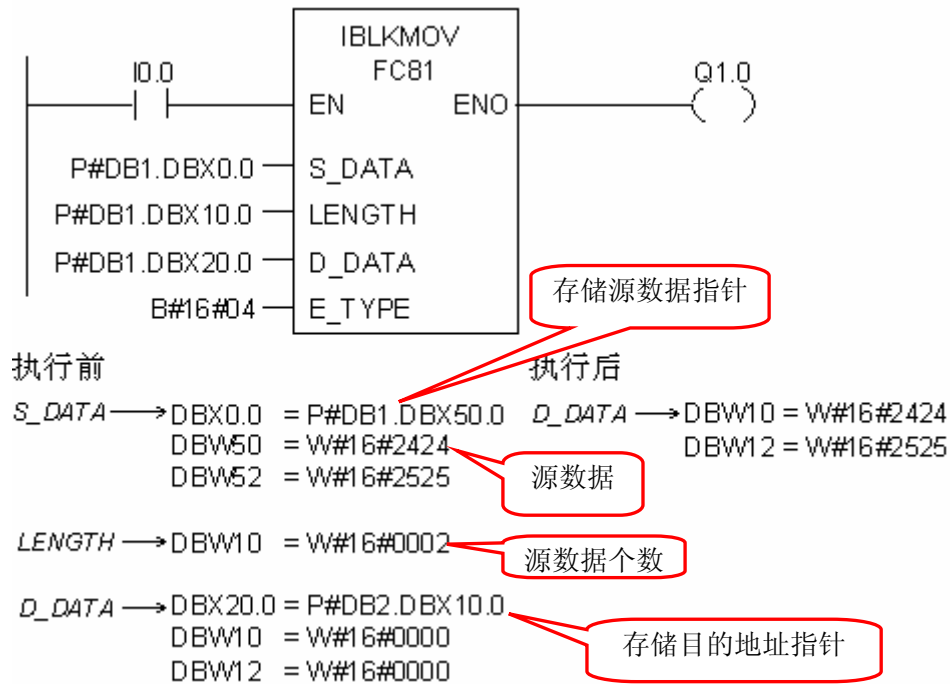


图 2-15: FC81 程序例子

注意：FC81 的源数据地址，目的数据地址，都以指针的形式存储于输入参数中，所以为被称为间接块移动。

2.6.2. 压缩数据：FB86

描述

PACK 功能块在随机位置和表之间交叉移动数据。传送方向由 DIR 指定。每个 PACK 指令最多处理五个数据 (P_DATA1 至 P_DATA5)。如果 DIR 指示"到"，PACK 将数据从这些位置移动到指定的表。同样，如果 DIR 指示"从"，则将数据从表分散到这些位置。

压缩"到"表格的规则如下：

- 单个位 (布尔) 压缩至表中的下一个可用位。
- 8 位数据类型压缩至表中的下一可用字节。将一个字节写入表中后，前一个字中未使用的位以零填充。
- 16 和 32 位数据类型压缩至表中的下一可用字。将一个字写入表中后，前一个字中未使用的位以零填充。

"从"表格压缩的规则如下：

- 不能跳过表的区段。
- 所有指定的 BOOL 点均从表格压缩。
- 8 位数据类型的压缩从表格中的第一个可用字节开始。即，从表格压缩的字节不包括表中

前一字节中的未用位。

- 16 和 32 位数据类型压缩自表格的第一个可用字。即，从表格压缩的字不包括表格中前一字中的未用位。

PACK 功能块支持的 ANY 指针的有效数据类型包括：

- BOOL
- WORD
- INT
- BYTE
- DINT
- REAL
- CHAR
- DWORD

参数

参数	描述	数据类型	存储区	描述
EN	输入	BOOL	I、Q、M、D、L	使能输入端的信号状态为 1，激活该功能。
ENO	输出	BOOL	I、Q、M、D、L	如果该功能的执行无错误，使能输出端的信号状态为 1。
TABLE	输入	*Pointer	I、Q、M、D	指向表的起始处
P_DATA1	输入	Any	I、Q、M、D	指向要压缩的数据块的起始处
P_DATA2	输入	Any	I、Q、M、D	指向要压缩的数据块的起始处
P_DATA3	输入	Any	I、Q、M、D	指向要压缩的数据块的起始处
P_DATA4	输入	Any	I、Q、M、D	指向要压缩的数据块的起始处
P_DATA5	输入	Any	I、Q、M、D	指向要压缩的数据块的起始处
ERR_CODE	输出	WORD	I、Q、M、D、L、P	如果该指令的执行没有错误，将返回 W#16#0000。对于 W#16#0000 以外的其它值，参见"错误信息"。
DIR	静态	BOOL	I、Q、M、D、L	压缩的方向。信号状态 0 = 到，信号状态 1 = 从。
* 用于跨区域寄存器间接寻址的双字指针格式				

表 2-18 FB86 参数说明

错误信息

如果发生以下任何情况，则不执行该功能块。ENO 的信号状态设置为 0，并相应地设置 ERR_CODE:

ERR_CODE 解释

W#16#0001 为功能参数指定的存储器类型无效。

W#16#0002 E_TYPE 无效。

实例

如果输入 I0.0 的信号状态为 1 (激活), 则执行 PACK 功能块。在本例中, 共有四个数据块压缩"到"表。

如果该功能块的执行未发生错误, ENO 和 Q0.0 的信号状态将设置为 1, 并且 ERR_CODE 等于 W#16#0000。

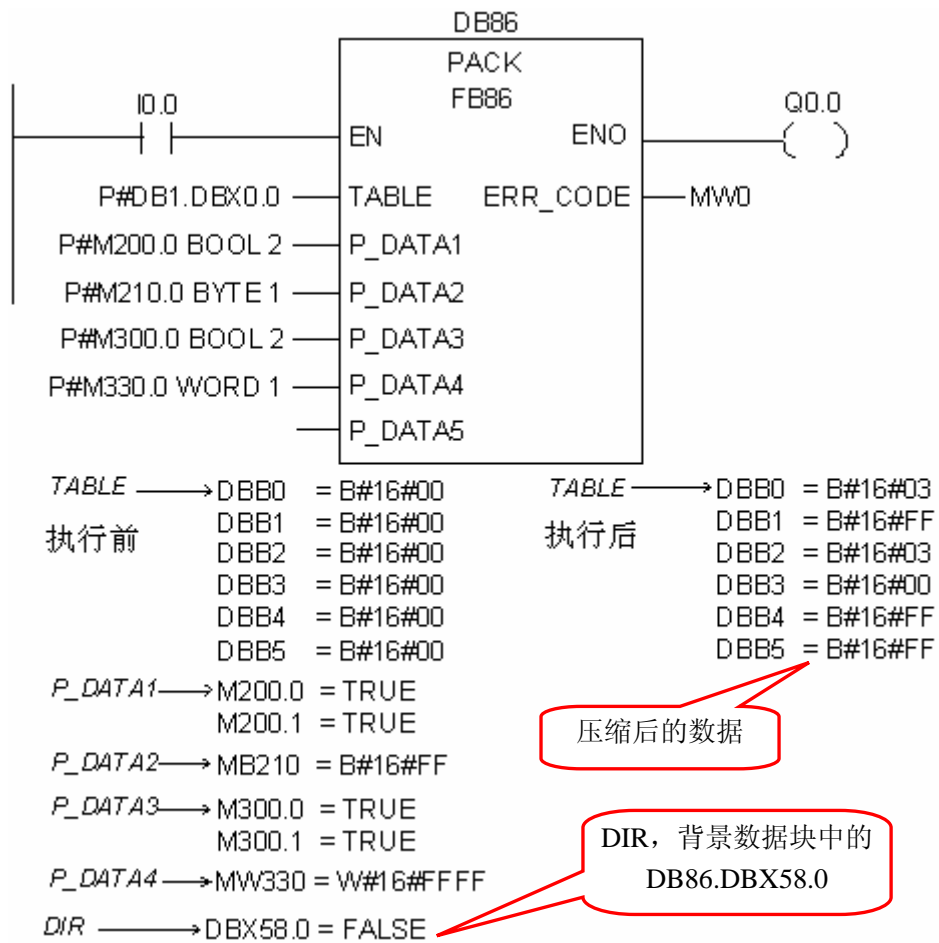


图 2-16: FB86 程序例子

注意: 可以使用"数据块编辑器"完成静态参数的初始化。如果 DIR 指示"到", PACK 将数据从这些位置移动到指定的表, 可以将分散的数据压缩并紧凑处理 (例如: 有利于减少 PLC 与上位机通信所使用的标签数量)。如果 DIR 指示"从", 则将数据从表分散到这些位置。可以应用在实际工程的多组配方处理当中 (同时也可以应用于上位机向 PLC 传递参数)。

2.7. 定时器功能及功能块

2.7.1. 软件延时定时器 – 掉电保护：FC80

描述

TONR 功能持续累计计时，直到当前所累计的时间（ET）值等于或超过预设值（PV）。由于该定时器使用 OB 的最后一个周期的执行时间进行累积计时，所以该定时器功能只能适用于具有重复特性的 OB(例如 OB1 和循环 OB)。

注意：必须将 OB 扫描时间从“启动信息”(参见“程序设计手册”中相应 OB 的变量说明表)移动至全局变量 DELTA_T。

只要 RESET 的信号状态是 0，TMR_EN 的信号状态是 1，并且 ET 小于 PV，该功能就将 DELTA_T 加到 ET 中。如果 TMR_EN 输入的信号状态不是 1，则不将时间加入 ET。当 ET 达到或超过 PV 时，输出 Q 的信号状态即设置为 1。Q 打开后，将保持打开状态，并且 ET 保持最后的值直到复位。RESET 信号状态为 1 时，该功能将 ET 复位为零，并关闭输出 Q。

参数

参数	描述	数据类型	存储区	描述
EN	输入	BOOL	I、Q、M、D、L	使能输入端的信号状态为 1，激活该功能。
ENO	输出	BOOL	I、Q、M、D、L	如果该功能的执行无错误，使能输出端的信号状态为 1。
TMR_EN	输入	BOOL	I、Q、M、D、L	定时器累计时间使能端
RESET	输入	BOOL	I、Q、M、D、L	如果 RESET = 1，则定时器复位为 0
PV	输入	DINT	P、I、Q、M、D、L、 常数	预设值
DELTA_T	输入	INT	I、Q、M、D、L、常 数	上个周期的 OB 扫描时间
Q	输出	BOOL	Q、M、D、L	如果 ET 等于或超过 PV，则 Q 的信号状态设置为 1
ET	输入/ 输出	DINT	I、Q、M、D、L	到目前为止所经过的时间
* 用于跨区域寄存器间接寻址的双字指针格式				

表 2-19 FC80 参数说明

错误信息

此功能不检测任何错误情况。

实例

如果输入 I0.0 的信号状态为 1 (激活), 则执行 TONR 功能。如果 I0.1 的信号状态为 1, I0.2 的信号状态为 0, 并且 ET 小于 PV, 则将 DELTA_T 加入 ET。如果 ET 小于 PV, Q1.1 的信号状态将保持 0。

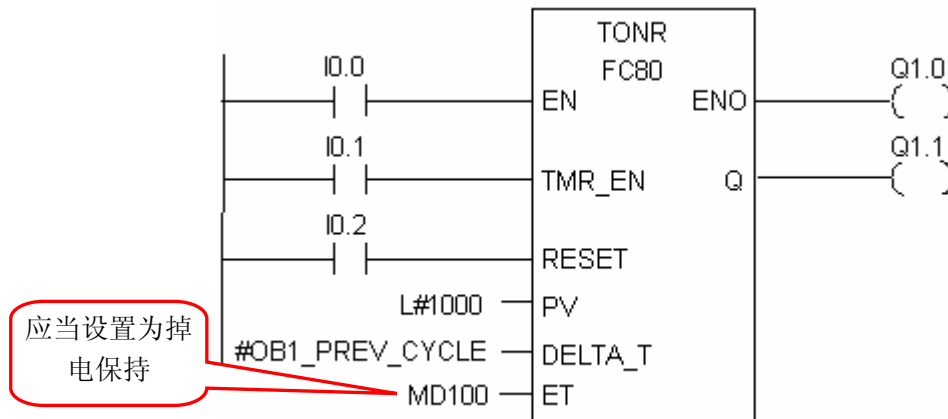


图 2-17: FC80 程序例子

2.7.2. 离散控制报警定时器: FB81

描述

命令(CMD)输入端转变为打开(或关闭)时, DCAT 功能块开始累计计时, 直到超过预设时间(PT), 或反馈输入(O_FB 或 C_FB)指示设备在预定时间内已打开(或关闭)。如果在收到反馈前就超过了预设时间, 则开启相应的警报。如果输入命令在预设时间前切换状态, 则重新计时。

- 当 CMD 输入的信号状态从 0 跳转到 1 时, Q 的信号状态将设置为 0, ET 设置为 0, 两个警报输出(OA、CA)的信号状态都设置为 0, 同时 CMD_HIS 的信号状态设置为 1。
- 当 CMD 输入的信号状态从 1 跳转到 0 时, Q 的信号状态设置为 0, ET 设置为 0, 两个警报输出(OA、CA)的信号状态设置为 0, 同时 CMD_HIS 的信号状态设置为 0。
- 当 CMD 和 CMD_HIS 的信号状态为 1, 且 O_FB 的信号状态为 0 时, 该功能块自上次执行后的时间(毫秒)增量将加入 ET。如果 ET
- 超过 PT, OA 的信号状态将设置为 1, 否则设置为 0。CMD_HIS 的信号状态将设置为与 CMD 相等。
- 当 CMD 和 CMD_HIS 的信号状态都为 1, O_FB 的信号状态为 1, 且 C_FB 的信号状态为 0 时, OA 的信号状态将设置为 0。ET 将设置为与 PT 相等, 这样当 O_FB 的信号状态以后设置为 0 时, 下次执行该功能块时将把警报设置为打开。CMD_HIS 的信号状态设置为与 CMD 相等。
- 当 CMD 和 CMD_HIS 的信号状态都为 0, 且 C_FB 的信号状态为 0 时, 该功能块自上次执

行后的时间(毫秒)增量将加入 ET。如果 ET 超过 PT，CA 的信号状态将设置为 0，否则设置为 1。CMD_HIS 的信号状态将设置为与 CMD 相等。

- 当 CMD 和 CMD_HIS 的信号状态都为 0，O_FB 的信号状态为 0，且 C_FB 的信号状态为 1 时，CA 的信号状态将设置为 0。ET 将设置为与 PT 相等，这样当 C_FB 的信号状态以后设置为 0 时，下次执行该功能块时将把警报设置为打开。CMD_HIS 的信号状态将设置为与 CMD 相等。
- 如果 O_FB 和 C_FB 的信号状态同时为 1，此情况将被视为错误，并且两个警报输出的信号状态均将设置为 1。

参数

参数	描述	数据类型	存储区	描述
EN	输入	BOOL	I、Q、M、D、L	使能输入端的信号状态为 1，激活该功能。
ENO	输出	BOOL	I、Q、M、D、L	如果该功能的执行无错误，使能输出端的信号状态为 1。
CMD	输入	BOOL	I、Q、M、D、L	信号状态 0 表示关闭命令。信号状态 1 表示打开命令。
O_FB	输入	BOOL	I、Q、M、D、L	打开反馈输入
C_FB	输入	BOOL	I、Q、M、D、L	关闭反馈输入
Q	输出	BOOL	I、Q、M、D、L	取决于 CMD 输入
OA	输出	BOOL	I、Q、M、D、L	打开警报输出
CA	输出	BOOL	I、Q、M、D、L	关闭警报输出
ET	静态	DINT	I、Q、M、D、L	历经时间的当前计数值，其中 1 次计数 = 1 毫秒
PT	静态	DINT	I、Q、M、D、L	定时器预置计数，其中 1 次计数 = 1 毫秒
PREV_TIME	静态	DWORD	I、Q、M、D、L	前一系统时间
CMD_HIS	静态	BOOL	I、Q、M、D、L	CMD 历史位

表 2-20 FB81 参数说明

错误信息

此功能块不检测任何错误情况。

实例

如果输入 I0.0 的信号状态为 1 (激活)，则执行 DCAT 功能块。在本例中，如 CMD_HIS 和 CMD 所示，CMD 输入从信号状态 0 跳转到 1。基于以上条件，Q 和 CMD_HIS 将获得信号状态 1，ET 将设置为 0，警报输出 OA 和 CA 将获得信号状态 0。

如果该功能块的执行没有错误，ENO 和 Q0.0 的信号状态将设置为 1。

注意：可以使用“数据块编辑器”完成静态参数的初始化。

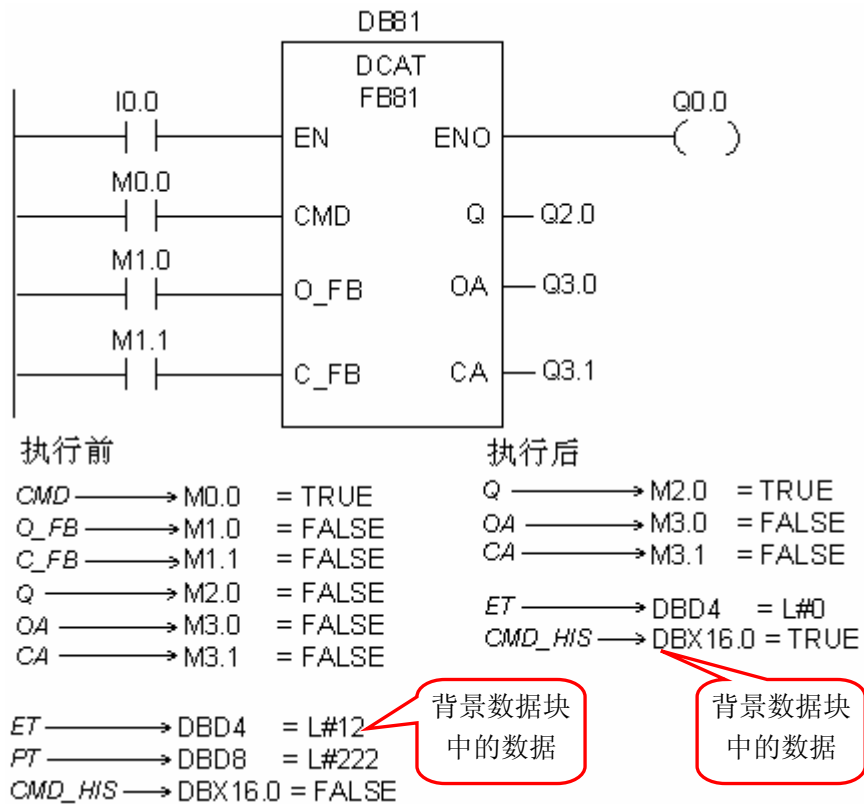


图 2-18: FB81 程序例子

2.7.3. 电机控制报警定时器: FB82

描述

MCAT 功能块从任何一个命令输入端（打开或关闭）的“开启”跳变开始累计计时，直到超过预设时间，或相应的反馈输入指示设备已在预定时间内完成指定的操作。如果收到反馈前就超过了预设时间，则开启相应警报。

MCAT 对各种输入条件的反应如下表所示。

MCAT 真值表

INPUTS								OUTPUTS								
ET	O_HIS	C_HIS	O_CMD	C_CMD	S_CMD	O_FB	C_FB	OO	CO	OA	CA	ET	O_HIS	C_HIS	Q	STATE
X	1	1	X	X	X	X	X	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	X	1	1	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	1	X	X	0	0	0	0	X	0	0	1	stop
X	X	X	1	1	X	X	X	0	0	0	0	X	0	0	1	stop
X	0	X	1	0	0	X	X	1	0	0	0	0	1	0	1	Begin open
<PT	1	0	X	0	0	0	X	1	0	0	0	INC	1	0	1	Opening
X	1	0	X	0	0	1	0	0	0	0	0	PT	1	0	1	Opened
>=PT	1	0	X	0	0	0	X	0	0	1	0	PT	1	0	0	Open alarm
X	X	0	0	1	0	X	X	0	1	0	0	0	0	1	1	Begin close
<PT	0	1	0	X	0	X	0	0	1	0	0	INC	0	1	1	Closing
X	0	1	0	X	0	0	1	0	0	0	0	PT	0	1	1	Closed
>=PT	0	1	0	X	0	X	0	0	0	0	1	PT	0	1	0	Close alarm
X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	1	stopped

表 2-21 MCAT 真值表

此处： INC 含义： 将自功能块上次执行以来经过的时间(毫秒)增量加入 ET。

PT 含义： 将 PT 设置为与 ET 相等。

X 含义： 不适用。

<PT 含义： ET<PT

>=PT 含义： ET>=PT

参数

参数	描述	数据类型	存储区	描述
EN	输入	BOOL	I、Q、M、D、L	使能输入端的信号状态为 1，激活该功能。
ENO	输出	BOOL	I、Q、M、D、L	如果该功能的执行无错误，使能输出端的信号状态为 1。
O_CMD	输入	BOOL	I、Q、M、D、L	打开命令输入
C_CMD	输入	BOOL	I、Q、M、D、L	关闭命令输入
S_CMD	输入	BOOL	I、Q、M、D、L	停止命令输入
O_FB	输入	BOOL	I、Q、M、D、L	打开反馈输入
C_FB	输入	BOOL	I、Q、M、D、L	关闭反馈输入
OO	输出	BOOL	I、Q、M、D、L	打开输出
CO	输出	BOOL	I、Q、M、D、L	关闭输出
OA	输出	BOOL	I、Q、M、D、L	打开警报输出
CA	输出	BOOL	I、Q、M、D、L	关闭警报输出
Q	输出	BOOL	I、Q、M、D、L	信号状态 1 表示警报状态

ET	静态	DINT	I、Q、M、D、L	历经时间的当前计数值,其中1次计数 = 1 毫秒
PT	静态	DINT	I、Q、M、D、L	定时器预置计数,其中1次计数 = 1 毫秒
PREV_TIME	静态	DWORD	I、Q、M、D、L	前一系统时间
O_HIS	静态	BOOL	I、Q、M、D、L	打开历史位
C_HIS	静态	BOOL	I、Q、M、D、L	关闭历史位

表 2-22 F82 参数说明

错误信息

此功能块不检测任何错误情况。

实例

如果输入 I0.0 的信号状态为 1 (激活), 则执行 MCAT 功能块。在本例中, 根据输入的状态, MCAT 处于 OPENING 状态, 并相应设置输出。

如果该功能块的执行没有错误, ENO 和 Q0.0 的信号状态将设置为 1。

注意: 可以使用"数据块编辑器"完成静态参数的初始化。

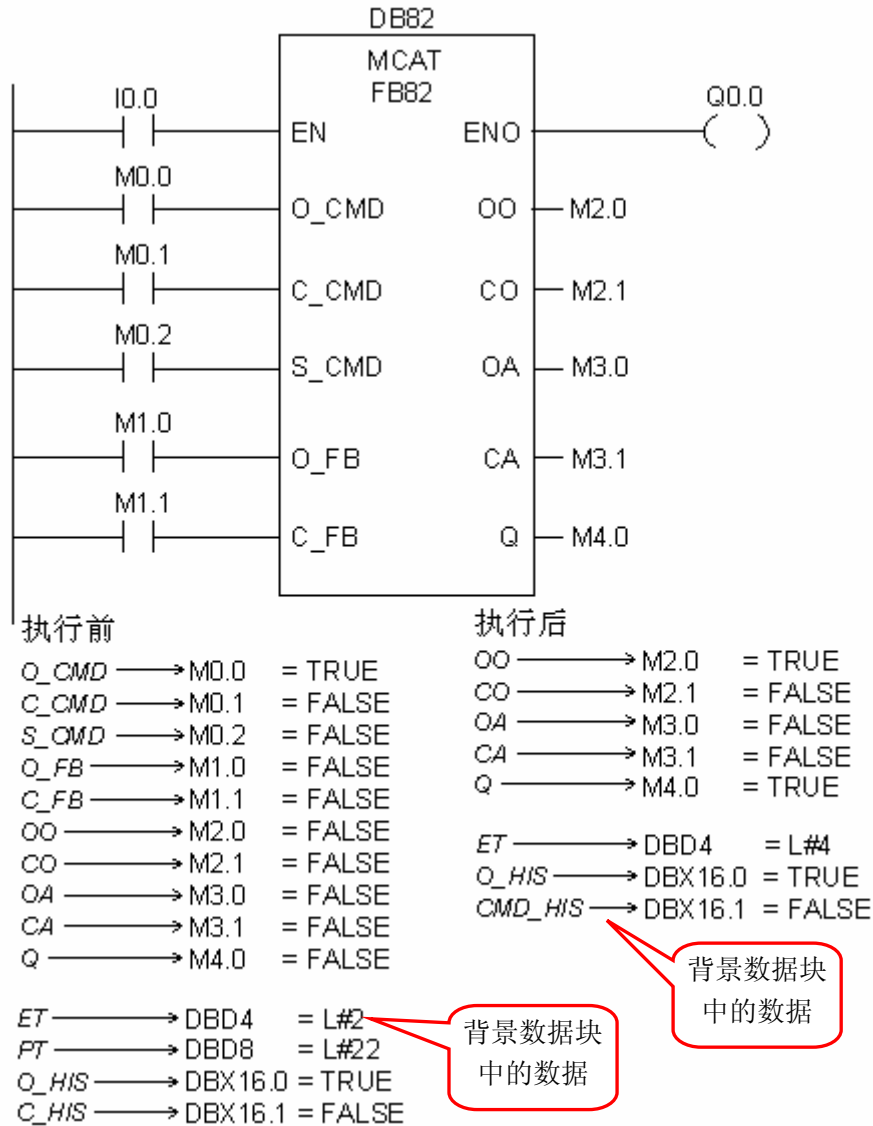


图 2-19: FB82 程序例子

2.7.4. 可屏蔽事件 Drum: FB85

描述

DRUM 功能块使用适当步骤的已编程值 (OUT_VAL) 驱动已编程输出位 (OUT1 至 OUT16) 和字输出 (OUT_WORD)。该适当步骤由 DRUM 位于该步骤时该步骤启用的掩码 (S_MASK)所决定。当该步的事件为真，且已超过当前步骤的程序设定时间时；或者点动输入从 0 跳转到 1 时，DRUM 进入下一步。当输入端 RESET 的信号状态为 1 时，复位 DRUM，从而把当前步骤设置为预设步骤 (DSP)。

一个步骤所用的时间由预置 DRUM 时间基准(DTBP)与每步相应的预置计数/步骤值(S_PRESET)的

乘积确定。在新步骤开始时，将此计算值装载到 DCC。DCC 含有当前步骤的剩余时间。例如，如果 DTBP 等于 2，步骤 1 的预设值等于 100 (100 毫秒)，则 DCC 等于 200 (200 毫秒)。

步骤可以使用时间值、事件或上述二者编程。事件位的信号状态为 0 的步骤将在该事件位的信号状态变为 1 时，立即进入下一个步骤。只含有时间的步骤将立即开始计时。含有事件位且时间值大于 0 的步骤，将在该事件位的信号状态为 1 时开始计时。该事件位的信号状态被初始化为 1。

当步骤指针位于上一编程步骤(LST_STEP)，但是该步骤的时间已结束，则输出 Q 的信号状态设置为 1；否则设置为 0。Q 被置位后，DRUM 将保持在该步骤直到复位。

可组态掩码(S_MASK)允许选择输出字(OUT_WORD)中的各个位和由输出值(OUT_VAL)置位/复位的输出位(OUT1 至 OUT16)中的各个位。当可组态掩码的某个位的信号状态为 1 时，OUT_VAL 值将置位/复位相应的位。当可组态掩码的某个位的信号状态为 0 时，相应的位将保持不变。所有 16 个步骤的可组态掩码的每个位都在信号状态为 1 时初始化。

输出位 OUT1 对应输出字(OUT_WORD)的最低有效位。输出位 OUT16 对应输出字 (OUT_WORD) 的最高有效位。

参数

参数	描述	数据类型	存储区	描述
EN	输入	BOOL	I、Q、M、D、L	使能输入端的信号状态为 1，激活该功能。
ENO	输出	BOOL	I、Q、M、D、L	如果该功能的执行无错误，使能输出端的信号状态为 1。
RESET	输入	BOOL	I、Q、M、D、L	信号状态 1 表示复位状态
JOG	输入	BOOL	I、Q、M、D、L	信号状态从 0 跳变到 1，使 DRUM 进入下一步骤
DRUM_EN	输入	BOOL	I、Q、M、D、L	信号状态 1 允许 Drum 根据事件和计时条件继续前进
LST_STEP	输入	BYTE	I、Q、M、D、L、常数	最后编程的步骤的步骤号
EVENT1	输入	BOOL	I、Q、M、D、L	事件位 1；初始信号状态为 1
EVENT2	输入	BOOL	I、Q、M、D、L	事件位 2；初始信号状态为 1
...
EVENT16	输入	BOOL	I、Q、M、D、L	事件位 16；初始信号状态为 1

OUT1	输出	BOOL	I、Q、M、D、L	输出位 1
OUT2	输出	BOOL	I、Q、M、D、L	输出位 2
...
OUT16	输出	BOOL	I、Q、M、D、L	输出位 16
Q	输出	BOOL	I、Q、M、D、L	信号状态 1 表示最后一步超时
OUT_WORD	输出	WORD	I、Q、M、D、L、P	Drum 向其中写入输出值的字的位置
ERR_CODE	输出	WORD	I、Q、M、D、L、P	如果该指令的执行没有错误,将返回 W#16#0000。对于 W#16#0000 以外的其它值,参见"错误信息"
JOG_HIS	静态	BOOL	I、Q、M、D、L	点动历史位
EOD	静态	BOOL	I、Q、M、D、L	信号状态 1 表示最后一步超时
DSP	静态	BYTE	I、Q、M、D、L、P	预置 Drum 步骤
DSC	静态	BYTE	I、Q、M、D、L、P	当前 Drum 步骤
DCC	静态	DWORD	I、Q、M、D、L、P	当前 Drum 计数
DTBP	静态	WORD	I、Q、M、D、L、P	预置 Drum 时间基准
PREV_TIME	静态	DWORD	I、Q、M、D、L	前一系统时间
S_PRESET	静态	ARRAY of WORD	I、Q、M、D、L	每个步骤[1 至 16]的预置计数,其中 1 次计数 = 1 毫秒。
OUT_VAL	静态	ARRAY of BOOL	I、Q、M、D、L	每个步骤[1 至 16, 0 至 15]的输出值
S_MASK	静态	ARRAY of BOOL	I、Q、M、D、L	每步[1 至 16, 0 至 15]的可组态掩码。初始信号状态为 1

表 2-23 FB85 参数说明

错误信息

如果发生以下任何情况, DRUM 保持在当前步骤, ENO 的信号状态将设置为 0, 并相应设置 ERR_CODE:

ERR_CODE 解释

W#16#000B LST_STEP 值小于 1 或大于 16。

W#16#000C DSC 值小于 1 或大于 LST_STEP。

W#16#000D DSP 值小于 1 或大于 LST_STEP。

实例

如果 I0.0 的信号状态为 1 (激活), 则执行 DRUM 功能块。在本例中, Drum 从步骤 1 前进到步骤 2。根据步骤 2 的已组态掩码和步骤 2 的 OUT_VAL 位设置输出位 (OUT1、OUT2 等) 和 OUT_WORD。如果该功能块的执行未发生错误, ENO 和 Q0.0 的信号状态将设置为 1, 并且 ERR_CODE 等于

W#16#0000。

注意：可以使用"数据块编辑器"完成静态参数的初始化。

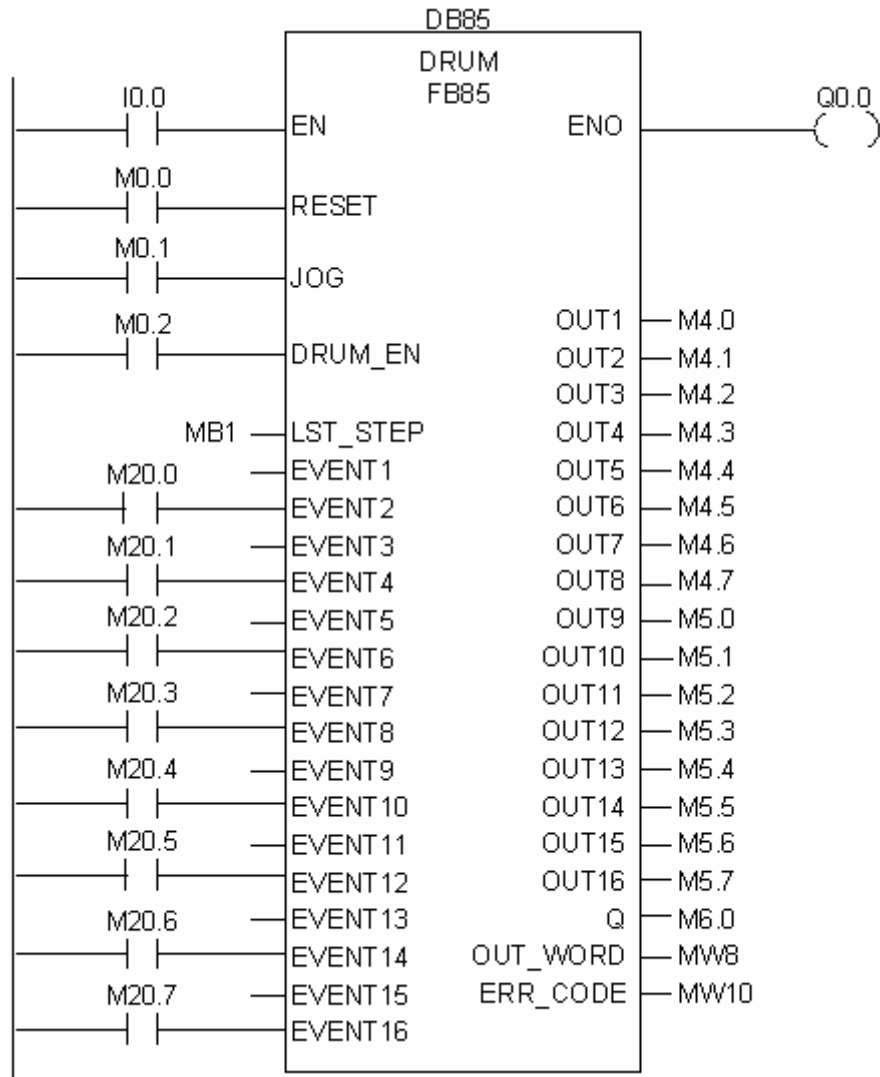


图 2-20: FB85 程序例子

执行前:

输入

RESET	M0.0	= FALSE
JOG	M0.1	= FALSE
DRUM_EN	M0.2	= TRUE
LST_STEP	MB1	= B#16#08
EVENT2	M20.0	= FALSE
EVENT4	M20.1	= FALSE
EVENT6	M20.2	= FALSE
EVENT8	M20.3	= FALSE
EVENT10	M20.4	= FALSE
EVENT12	M20.5	= FALSE
EVENT14	M20.6	= FALSE
EVENT16	M20.7	= FALSE

背景数据块 DB85

JOG_HIS DBX12.0 = FALSE
 EOD DBX12.1 = FALSE
 DSP DBB13 = W#16#0001
 DSC DBB14 = W#16#0001
 DCC DBD16 = DW#16#0000000A
 DTBP DBW20 = W#16#0001
 S_PRESET[1] DBW26 = W#16#0064
 S_PRESET[2] DBW28 = W#16#00C8
 OUT_VAL[1,0] DBX58.0 = TRUE
 OUT_VAL [1,1] DBX58.1 至 OUT_VAL [1,15] DBX59.7 都为 TRUE

 OUT_VAL[2,0] DBX60.0 = FALSE
 OUT_VAL [2,1] DBX60.1 至 OUT_VAL [2,15] DBX61.7 都为 FALSE

 S_MASK [2,0] DBX92.0 = FALSE
 S_MASK [2,1] DBX92.1 = TRUE
 S_MASK [2,2] DBX92.2 = TRUE
 S_MASK [2,3] DBX92.3 = TRUE
 S_MASK [2,4] DBX92.4 = TRUE
 S_MASK [2,5] DBX92.5 = FALSE
 S_MASK [2,6] DBX92.6 = TRUE
 S_MASK [2,7] DBX92.7 = TRUE
 S_MASK [2,8] DBX93.0 = FALSE
 S_MASK [2,9] DBX93.1 = FALSE
 S_MASK [2,10] DBX93.2 = TRUE
 S_MASK [2,11] DBX93.3 = TRUE
 S_MASK [2,12] DBX93.4 = TRUE
 S_MASK [2,13] DBX93.5 = TRUE
 S_MASK [2,14] DBX93.6 = FALSE
 S_MASK [2,15] DBX93.7 = TRUE

输出

Q M6.0 = FALSE
 OUTWORD MW8 = W#16#FFFF
 OUT1 M4.0 = TRUE
 OUT2 M4.1 至 OUT16 M5.7 都为 TRUE

执行后:

OUT1 M4.0 = TRUE
 OUT2 M4.1 = FALSE
 OUT3 M4.2 = FALSE
 OUT4 M4.3 = FALSE
 OUT5 M4.4 = FALSE
 OUT6 M4.5 = TRUE
 OUT7 M4.6 = FALSE

```

OUT8      M4.7    = FALSE
OUT9      M5.0    = TRUE
OUT10     M5.1    = TRUE
OUT11     M5.2    = FALSE
OUT12     M5.3    = FALSE
OUT13     M5.4    = FALSE
OUT14     M5.5    = FALSE
OUT15     M5.6    = TRUE
OUT16     M5.7    = FALSE
Q         M6.0    = FALSE
OUTWORD   MW8     = W#16#4321
ERR_CODE  MW10    = W#16#0000
背景数据块 DB85
JOG_HIS   DBX12.0 = FALSE
EOD       DBX12.1 = FALSE
DSC       DBB14   = W#16#0002
DCC       DBD16   = DW#16#000000C8
    
```

本例中的 OUT1-OUT16 的数值在执行前都为 TRUE, 即 OUT_VAL[1,0]至 OUT_VAL[1,16]的数值; 当程序执行后, OUT1-OUT16的数值等于 OUT_VAL[2,0] 至 OUT_VAL [2,15]的数值(由于 S_MASK [2,0]至 S_MASK [2,15]的部分数值为 FALSE,所以相关的 OUT 数值保持不变)。

注意: FB85 可以实现类似于顺序控制的功能, 用户可以把一系列的步骤动作对应为一系列的输出变化, 当用户控制当前步骤编号时, 输出就会发生相对应的变化; 用户在使用时, 可以通过组态掩码数组, 动态/静态的屏蔽某个步骤对应的输出变化, 以适应不同控制任务要求(例如, 假设 OUT2 在步骤 1, 3 时需要变化, 其它步骤中都不改变状态, 则 S_MASK [1,1], S_MASK [3,1],应当为 TRUE, S_MASK [2, 1], S_MASK [4, 1] 至 S_MASK [16, 1] 为 FALSE)。对于复杂的顺控程序, 用户应当通过使用 S7-GRAPH 编程语言来实现。

2.8. 转换功能及功能块

2.8.1. 七段解码器: FC93

描述

SEG 功能将指定源数据字 (IN) 中的四个十六进制数字的每个都转换为四个等价的 7 段显示代码, 并将其写入输出目标双字 (OUT) 。

输入十六进制数字与输出位模式的关系如下:

数字	gfedcba	显示
0000	00111111	0
0001	00000110	1
0010	01011011	2
0011	01001111	3
0100	01100110	4
0101	01101101	5
0110	01111101	6
0111	00000111	7
1000	01111111	8
1001	01100111	9
1010	01110111	A
1011	01111100	b
1100	00111001	C
1101	01011110	d
1110	01111001	E
1111	01110001	F

七段码显示

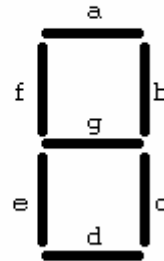


图 2-21: 输入十六进制数字与输出位模式的关系

参数

参数	描述	数据类型	存储区	描述
EN	输入	BOOL	I、Q、M、D、L	使能输入端的信号状态为 1, 激活该功能。
ENO	输出	BOOL	I、Q、M、D、L	如果该功能的执行无错误, 使能输出端的信号状态为 1。
IN	输入	WORD	I、M、D、P、常数	以四个十六进制数字表示的源数据字
OUT	输出	DWORD	Q、M、D、L、P	以四个字节表示的目标位模式

表 2-24 FC93 参数说明

错误信息

此功能不检测任何错误情况。

实例

如果输入 I0.0 的信号状态为 1 (激活), 则执行 SEG 功能。

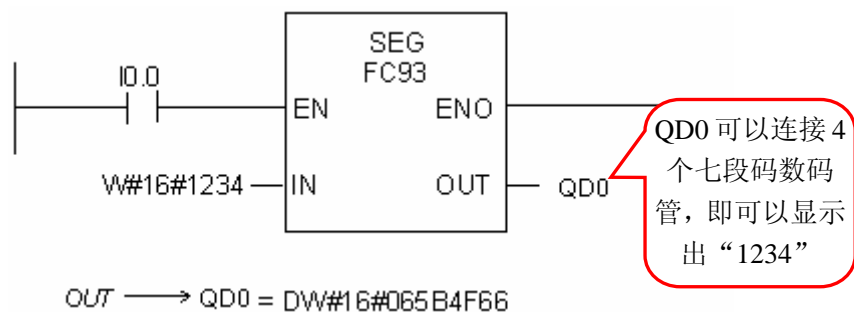


图 2-22: FC93 程序例子

2.8.2. ASCII 转换为十六进制: FC94

描述

ATH 功能将 IN 指向的 ASCII 字符串转换为压缩十六进制数字，然后将其存储在 OUT 指向的目标表格中。由于 ASCII 字符需要 8 位，而十六进制数字只需要 4 位，输出字长度仅是字长度的一半。按其读入顺序转换 ASCII 字符并将结果置于十六进制输出中。如果 ASCII 字符数为奇数，则最后转换的十六进制数字的最右侧半个字节以零填充。

参数

参数	描述	数据类型	存储区	描述
EN	输入	BOOL	I、Q、M、D、L	使能输入端的信号状态为 1，激活该功能。
ENO	输出	BOOL	I、Q、M、D、L	如果该功能的执行无错误，使能输出端的信号状态为 1。
IN	输入	*Pointer	I、Q、M、D、L	指向 ASCII 字符串的起始位置
N	输入	INT	I、Q、M、L、P	要转换的 ASCII 输入字符数
RET_VAL	输出	WORD	I、Q、M、D、L、P	如果该指令的执行没有错误，将返回值 W#16#0000。对于 W#16#0000 以外的其它值，参见"错误信息"
OUT	输出	*Pointer	Q、M、D、L	指向表格的起始位置
* 用于跨区域寄存器间接寻址的双字指针格式				

表 2-25 FC94 参数说明

错误信息

如果发现无效 ASCII 字符，则将其转换为 0。ENO 的信号状态将设置为 0，RET_VAL 将设置位 W#16#0007。

实例

如果输入 I0.0 的信号状态为 1 (激活)，则执行 ATH 指令。在本例中，输入参数 N 为 5 表示要转换五个 ASCII 字符。ASCII 字符以 IN 指针位置 DB1.DBX10.0 为起点，存储在数据块 1 中。输出字符串将以 OUT 指针位置 DB2.DBX0.0 (数据块 2) 为起点存储。由于 ASCII 输入字符数为奇数，最后一个十六进制位最右边的半个字节中的值全部为零，从而生成十六进制值 0xC0。(有关每个 ASCII 字符的等价十六进制值，请参考下表。)

如果该功能的执行没有错误，ENO 和 Q0.0 的信号状态将设置为 1，RET_VAL 等于 W#16#0000。

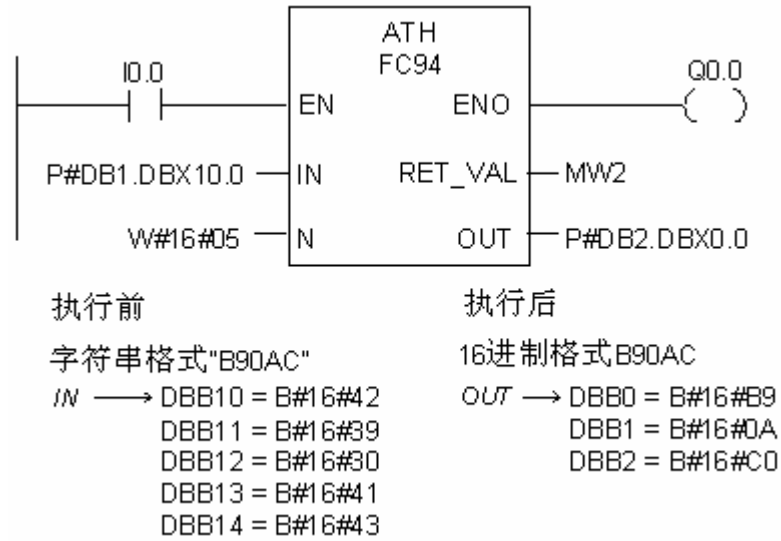


图 2-23: FC94 程序例子

ASCII 字符和等价的十六进制值

ASCII 字符格式	ASCII 对应 16 进制值	转换后的 16 进制值
"0"	30	0
"1"	31	1
"2"	32	2
"3"	33	3
"4"	34	4
"5"	35	5
"6"	36	6
"7"	37	7
"8"	38	8
"9"	39	9
"A"	41	A
"B"	42	B
"C"	43	C
"D"	44	D
"E"	45	E
"F"	46	F

表 2-26 ASCII 字符和等价的十六进制值

2.8.3. 十六进制转换为 ASCII: FC95

描述:

HTA 功能对 IN 指向的压缩十六进制数字进行转换, 并将其存储在 OUT 指向的目标字符串中。由于 ASCII 字符需要 8 位, 而十六进制位仅需要 4 位, 因此输出字长度将是输入字长度的 2 倍。十六进制数字的每个半字节将按其读入的顺序(首先转换十六进制数字最左边的半字节, 然后转换该

数位最右边的半字节) 转换为一个字符。

参数

参数	描述	数据类型	存储区	描述
EN	输入	BOOL	I、Q、M、D、L	使能输入端的信号状态为 1，激活该功能。
ENO	输出	BOOL	I、Q、M、D、L	如果该功能的执行无错误，使能输出端的信号状态为 1。
IN	输入	*Pointer	I、Q、M、D、	指向十六进制数位串的起始位置
N	输入	INT	I、Q、M、D、L、P	要转换的十六进制输入字节的数量
OUT	输出	*Pointer	Q、M、D、L	指向表格的起始位置
* 用于跨区域寄存器间接寻址的双字指针格式				

表 2-27 FC95 参数说明

错误信息

此功能不检测任何错误情况。

实例

如果输入 I0.0 的信号状态为 1 (激活)，则执行 HTA 指令。在本例中，输入参数 N 为 3 表示要转换三个十六进制字符。十六进制字节以 IN 指针位置 DB1.DBX0.0 为起点存储在数据块 1 中。输出字符串将位于以 DB2.DBX0.0 (数据块 2)为起点的 OUT 指针位置。(有关每个十六进制值的 ASCII 等价值，请参考下表。)

如果该功能的执行未发生错误，ENO 和 Q0.0 的信号状态将设置为 1。

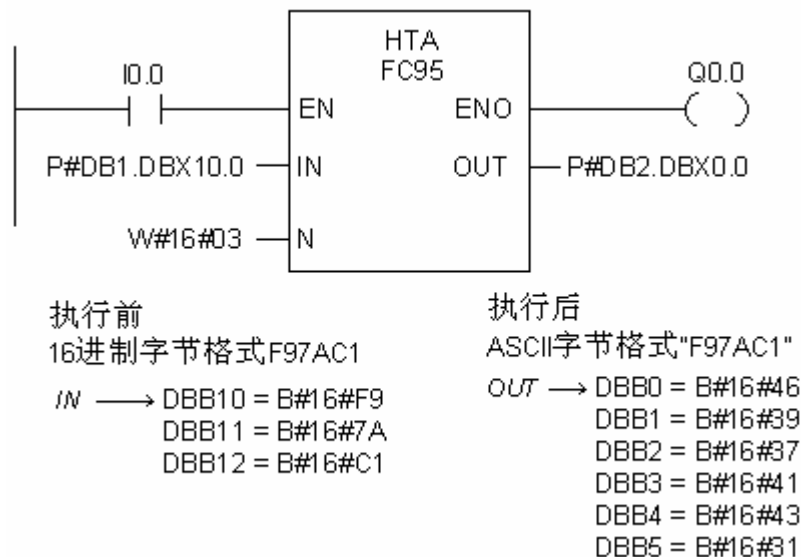


图 2-24: FC95 程序例子

十六进制数字和等价的 ASCII 十六进制值

16 进制值	对应 ASCII 码的 16 进	转换后的 ASCII 字符
--------	------------------	---------------

	制值	格式
0	30	“0”
1	31	“1”
2	32	“2”
3	33	“3”
4	34	“4”
5	35	“5”
6	36	“6”
7	37	“7”
8	38	“8”
9	39	“9”
A	41	“A”
B	42	“B”
C	43	“C”
D	44	“D”
E	45	“E”
F	46	“F”

表 2-28 十六进制数字和等价的 ASCII 十六进制值

2.8.4. 编码二进制位置：FC96

描述

ENCO 功能将检测 IN 的内容，查找其从右侧开始，按照二进制格式，首个被置为 1 的位置，并将结果作为功能值返回。如果 IN 是 DW#16#00000001 或 DW#16#00000000，则返回 0 值。

参数

参数	描述	数据类型	存储区	描述
EN	输入	BOOL	I、Q、M、D、L	使能输入端的信号状态为 1，激活该功能。
ENO	输出	BOOL	I、Q、M、D、L	如果该功能的执行无错误，使能输出端的信号状态为 1。
IN	输入	DWORD	I、M、D、L、P、常数	要编码的变量
RET_VAL	输出	INT	Q、M、D、L、P	返回值(含有 5 位二进制数)

表 2-29 FC96 参数说明

错误信息

此功能不检测任何错误情况。

实例

如果该功能的执行未发生错误，ENO 和 Q0.0 的信号状态将设置为 1。

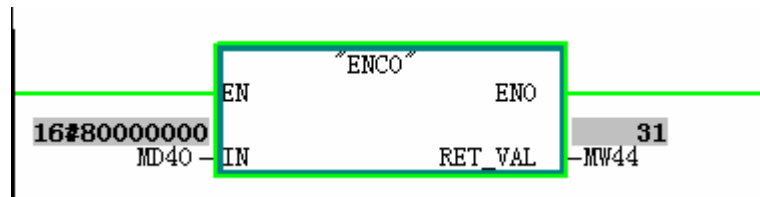


图 2-25: FC96 程序例子

注意： 此处列举输入/输出的关系如下：

输入=16#0/1	输出=0
输入=16#2	输出=1
输入=16#4	输出=2
输入=16#8	输出=3
输入=16#80000000	输出=16#31

2.8.5. 解码二进制位置：FC97

描述

DECO 功能通过设置该功能返回值中相应位的位置，将输入 IN 中的 5 位二进制数 (0-31) 转换为一个值。如果 IN 大于 31，将执行以 32 为模的模运算以获得一个 5 位二进制数。

参数

参数	描述	数据类型	存储区	描述
EN	输入	BOOL	I、Q、M、D、L	使能输入端的信号状态为 1，激活该功能。
ENO	输出	BOOL	I、Q、M、D、L	如果该功能的执行无错误，使能输出端的信号状态为 1。
IN	输入	DWORD	I、M、D、L、P、常数	要解码的变量
RET_VAL	输出	INT	Q、M、D、L、P	返回值

表 2-30 FC97 参数说明

错误信息

此功能不检测任何错误情况。

实例

如果该功能的执行未发生错误，ENO 和 Q0.0 的信号状态将设置为 1。

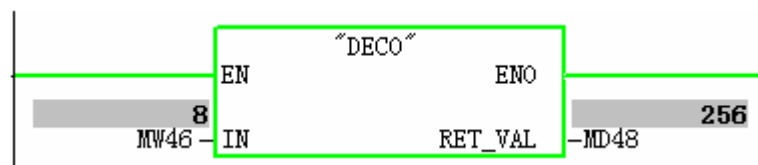


图 2-26: FC96 程序例子

注意： 此处列举输入/输出的关系如下：

输入=0 输出=1
 输入=1 输出=2
 输入=2 输出=4
 输入=3 输出=8
 输入=4 输出=16

2.8.6. 十进制补码：FC98

描述

BCDCPL 功能返回一个 7 位 BCD 数 IN 的十进制补码。此运算的数学公式如下：

$$10000000 \text{ (BCD 码)} \\ - \quad \quad \quad 7 \text{ 位BCD值}$$

十进制补码值(BCD 码)

参数

参数	描述	数据类型	存储区	描述
EN	输入	BOOL	I、Q、M、D、L	使能输入端的信号状态为 1，激活该功能。
ENO	输出	BOOL	I、Q、M、D、L	如果该功能的执行无错误，使能输出端的信号状态为 1。
IN	输入	DWORD	I、M、D、L、P、常数	7 位 BCD 数
RET_VAL	输出	INT	Q、M、D、L、P	返回值

表 2-31 FC98 参数说明

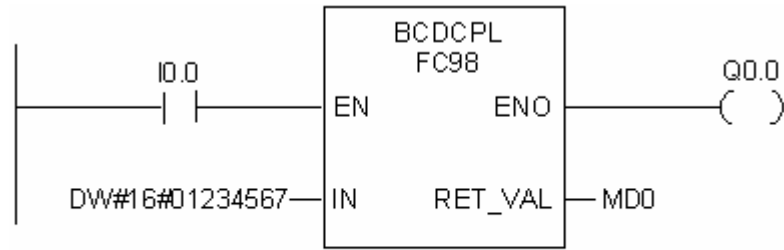
错误信息

此功能不检测任何错误情况。

实例

如果输入 I0.0 的信号状态为 1 (激活)，则执行 BCDCPL 功能。

如果该功能的执行未发生错误，ENO 和 Q0.0 的信号状态将设置为 1。



$RET_VAL \rightarrow M0.0 = DW\#16\#08765433$

图 2-27: FC98 程序例子

2.8.7. 位数求和: FC99

描述

BITSUM 功能计算输入 IN 中设置为 1 的位的数量，并将此值作为该功能的值返回。

参数

参数	描述	数据类型	存储区	描述
EN	输入	BOOL	I、Q、M、D、L	使能输入端的信号状态为 1，激活该功能。
ENO	输出	BOOL	I、Q、M、D、L	如果该功能的执行无错误，使能输出端的信号状态为 1。
IN	输入	DWORD	I、M、D、L、P、常数	要计算位数的变量
RET_VAL	输出	INT	Q、M、D、L、P	返回值

表 2-32 FC99 参数说明

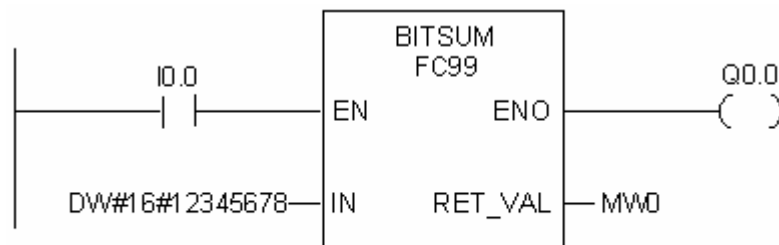
错误信息

此功能不检测任何错误情况。

实例

如果输入 I0.0 的信号状态为 1 (激活)，则执行 BITSUM 功能。在本例中，MWO 中的返回值为 13 (十六进制表示为"D")。该值是双字输入十六进制值 DW#16#12345678 中设置为 1 的位的总数。

如果该功能的执行未发生错误，ENO 和 Q0.0 的信号状态将设置为 1。



$RET_VAL \rightarrow MWO = W\#16\#000D$

图 2-28: FC99 程序例子

2.8.8. 标定值: FC105

描述

SCALE 功能接受一个整型值 (IN), 并将其转换为以工程单位表示的介于下限和上限 (LO_LIM 和 HI_LIM) 之间的实型值。将结果写入 OUT。SCALE 功能使用以下等式:

$$OUT = [((FLOAT(IN)-K1)/(K2-K1)) * (HI_LIM-LO_LIM)] + LO_LIM$$

常数 K1 和 K2 根据输入值是 BIPOLAR 还是 UNIPOLAR 设置。

- BIPOLAR: 假定输入整型值介于 -27648 与 27648 之间, 因此 K1 = -27648.0, K2 = +27648.0
- UNIPOLAR: 假定输入整型值介于 0 和 27648 之间, 因此 K1 = 0.0, K2 = +27648.0

如果输入整型值大于 K2, 输出(OUT)将钳位于 HI_LIM, 并返回一个错误。如果输入整型值小于 K1, 输出将钳位于 LO_LIM, 并返回一个错误。

通过设置 LO_LIM > HI_LIM 可获得反向标定。使用反向转换时, 输出值将随输入值的增加而减小。

参数

参数	描述	数据类型	存储区	描述
EN	输入	BOOL	I、Q、M、D、L	使能输入端的信号状态为 1, 激活该功能。
ENO	输出	BOOL	I、Q、M、D、L	如果该功能的执行无错误, 使能输出端的信号状态为 1。
IN	输入	INT	I、Q、M、D、L、P、 常数	欲转换为以工程单位表示的实型值的输入值
HI_LIM	输入	REAL	I、Q、M、D、L、P、 常数	以工程单位表示的上限值
LO_LIM	输入	REAL	I、Q、M、D、L、P、 常数	以工程单位表示的下限值
BIPOLAR	输入	BOOL	I、Q、M、D、L	信号状态为 1 表示输入值为双极性。 信号状态 0 表示输入值为单极性
OUT	输出	REAL	I、Q、M、D、L、P	转换的结果
RET_VAL	输出	WORD	I、Q、M、D、L、P	如果该指令的执行没有错误, 将返回值 W#16#0000。对于 W#16#0000 以外的其它值, 参见"错误信息"。

表 2-33 FC105 参数说明

错误信息

如果输入整型值大于 K2, 输出(OUT)将钳位于 HI_LIM, 并返回一个错误。如果输入整型值小于 K1, 输出将钳位于 LO_LIM, 并返回一个错误。ENO 的信号状态将设置为 0, RET_VAL 等于 W#16#0008。

实例

如果输入 I0.0 的信号状态为 1 (激活)，则执行 SCALE 功能。在本例中，整型值 22 将转换为介于 0.0 和 100.0 之间的实型值，并写入

OUT。如 I2.0 的信号状态所示，该输入值为 BIPOLAR。

如果该功能的执行没有错误，ENO 和 Q0.0 的信号状态将设置为 1，RET_VAL 等于 W#16#0000。

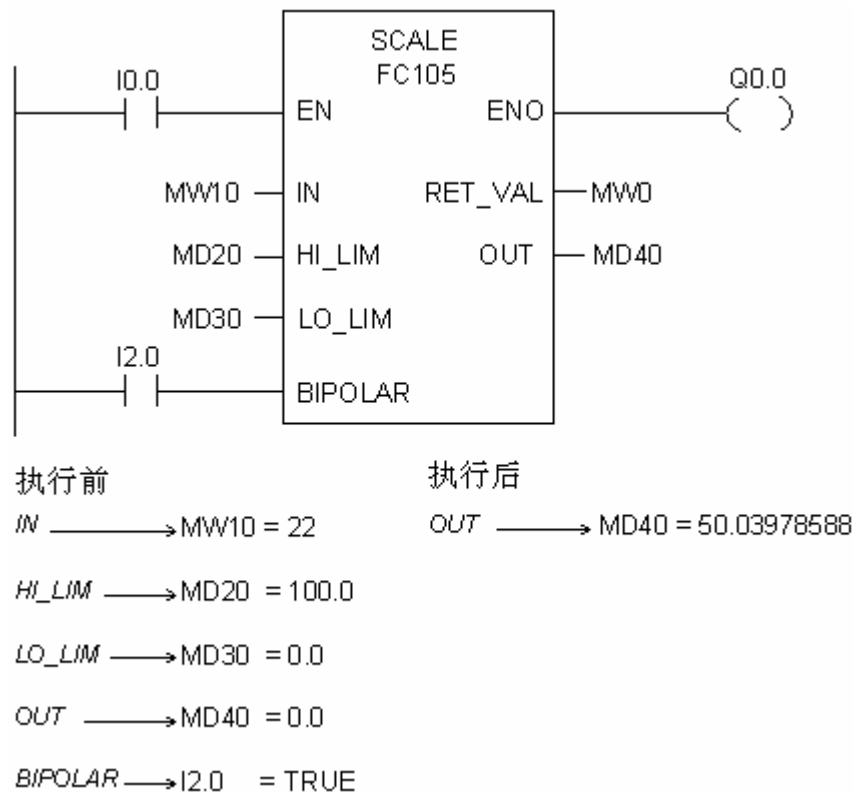


图 2-29: FC105 程序例子

2.8.9. 取消标定值: FC106

描述

UNSCALE 功能接收一个以工程单位表示、且标定于下限和上限 (LO_LIM 和 HI_LIM) 之间的实型输入值 (IN)，并将其转换为一个整型值。将结果写入 OUT。UNSCALE 功能使用以下等式：

$$OUT = [((IN-LO_LIM)/(HI_LIM-LO_LIM)) * (K2-K1)] + K1$$

根据输入值是 BIPOLAR 还是 UNIPOLAR 设置常数 K1 和 K2。

- BIPOLAR: 假定输出整型值介于 -27648 和 27648 之间，因此，K1 = -27648.0, K2 = +27648.0
- UNIPOLAR: 假定输出整型值介于 0 和 27648 之间，因此，K1 = 0.0, K2 = +27648.0

如果输入值超出 LO_LIM 和 HI_LIM 范围，输出(OUT)将钳位于距其类型(BIPOLAR 或 UNIPOLAR)的指定范围的下限或上限较近的一方，并返回一个错误。

参数

参数	描述	数据类型	存储区	描述
EN	输入	BOOL	I、Q、M、D、L	使能输入端的信号状态为 1，激活该功能。
ENO	输出	BOOL	I、Q、M、D、L	如果该功能的执行无错误，使能输出端的信号状态为 1。
IN	输入	REAL	I、Q、M、D、L、P、 常数	欲转换为整型值的输入值
HI_LIM	输入	REAL	I、Q、M、D、L、P、 常数	以工程单位表示的上限
LO_LIM	输入	REAL	I、Q、M、D、L、P、 常数	以工程单位表示的下限
BIPOLAR	输入	BOOL	I、Q、M、D、L	信号状态为 1 表示输入值为双极性。 信号状态 0 表示输入值为单极性
OUT	输出	INT	I、Q、M、D、L、P	转换结果
RET_VAL	输出	WORD	I、Q、M、D、L、P	如果该指令的执行没有错误，将返回值 W#16#0000。对于 W#16#0000 以外的其它值，参见"错误信息"

表 2-34 FC106 参数说明

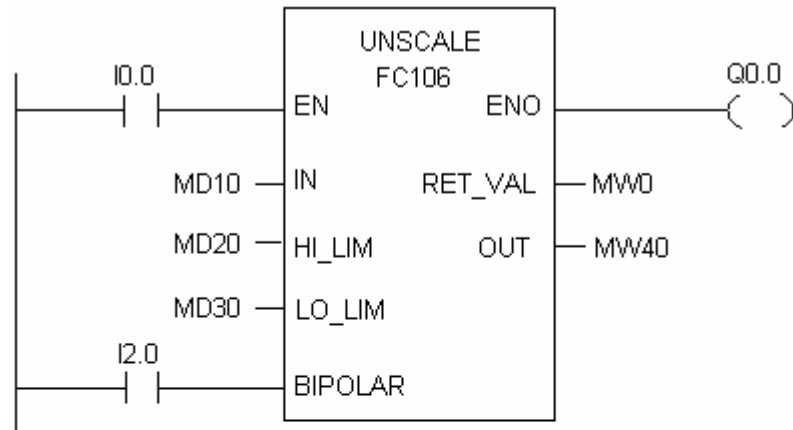
错误信息

如果输入值超出 LO_LIM 和 HI_LIM 范围，输出(OUT)将钳位于距其类型(BIPOLAR 或 UNIPOLAR)的指定范围的下限或上限较近的一方，并返回一个错误。ENO 的信号状态将设置为 0，RET_VAL 等于 W#16#0008。

实例

如果输入 I0.0 的信号状态为 1 (激活)，则执行 UNSCALE 功能。在本例中，标定于 0.0 和 100.0 之间的实型值 50.03978588，将转换为一个整型值，并写入 OUT。如 I2.0 的信号状态所示，该输入值为 BIPOLAR。

如果该功能的执行没有错误，ENO 和 Q0.0 的信号状态将设置为 1，RET_VAL 等于 W#16#0000。



执行前

$IN \longrightarrow MD10 = 50.03978588$

$HI_LIM \longrightarrow MD20 = 100.0$

$LO_LIM \longrightarrow MD30 = 0.0$

$OUT \longrightarrow MW40 = 0$

$BIPOLAR \longrightarrow I2.0 = TRUE$

执行后

$OUT \longrightarrow MW40 = 22$

图 2-30: FC106 程序例子

2.8.10. 超前/滞后算法: FB80

描述

LEAD_LAG 功能块可以对模拟变量进行信号处理。该功能根据输入(IN)和指定的增益(GAIN)、超前(LD_TIME)和滞后(LAG_TIME)

值计算输出(OUT)。增益值必须大于零。LEAD_LAG 算法使用以下等式:

$$OUT = \left[\frac{LG_TIME}{LG_TIME + SAMPLE_T} \right] PREV_OUT + GAIN \left[\frac{LD_TIME + SAMPLE_T}{LG_TIME + SAMPLE_T} \right] IN - GAIN \left[\frac{LD_TIME}{LG_TIME + SAMPLE_T} \right]$$

通常, LEAD_LAG 与回路一起用作动态前馈控制中的补偿器。LEAD_LAG 由两部分组成: 相位超前通过移位输出, 使输出超前输入, 而相位滞后通过移位输出, 使输出滞后输入。由于滞后操作相当于积分, 因此可用作噪声抑制器或低通滤波器。提前操作相当于微分, 因此是一个高通滤波器。LEAD_LAG 组合后可使低频输出相位滞后于输入, 高频率输出相位超前于输入, 因此可用作带通滤波器。

参数

参数	描述	数据类型	存储区	描述
EN	输入	BOOL	I、Q、M、D、L	使能输入端的信号状态为 1, 激活该功能。
ENO	输出	BOOL	I、Q、M、D、L	如果该功能的执行无错误, 使能输出

				端的信号状态为 1。
IN	输入	REAL	I、Q、M、D、L、P、 常数	要处理的当前采样期间的输入值
SAMPLE_T	输入	INT	I、Q、M、D、L、P、 常数	采样时间
OUT	输出	REAL	I、Q、M、D、L、P、 常数	LEAD_LAG 操作的结果
ERR_CODE	输出	WORD	I、Q、M、D、L、P	如果该指令的执行没有错误，将返回值 W#16#0000。对于 W#16#0000 以外的其它值，参见"错误信息"
LD_TIME	静态	REAL	I、Q、M、D、L、P、 常数	与采样时间相同单位的超前时间
LG_TIME	静态	REAL	I、Q、M、D、L、P、 常数	与采样时间相同单位的滞后时间
GAIN	静态	REAL	I、Q、M、D、L、P、 常数	%/%的增益(稳态下输出变化与输入变化的比率)
PREV_IN	静态	REAL	I、Q、M、D、L、P、 常数	前一输入
PREV_OUT	静态	REAL	I、Q、M、D、L、P、 常数	前一输出

表 2-35 FB80 参数说明

错误信息

如果 GAIN 小于或等于 0，则不执行该功能块。ENO 的信号状态将设置为 0，ERR_CODE 等于 W#16#0009。

实例

如果输入 I0.0 的信号状态为 1 (激活)，则执行 LEAD_LAG 功能块。在本例中，使用 LEAD_LAG 算法处理输入值(IN) 2.0，以生成输出(OUT)。

如果该功能块的执行未发生错误，ENO 和 Q0.0 的信号状态将设置为 1，ERR_CODE 将等于 W#16#0000。

注意：可以使用"数据块编辑器"完成静态参数的初始化。

2.9. 浮点数数学运算功能

2.9.1. 标准偏差：FC102

描述

DEV 功能可计算存储在表格 (TBL) 中的一组值的标准差，并将结果存储在 OUT 中。根据以下公

式计算标准偏差:

$$\text{标准偏差} = \sqrt{\frac{(N * \text{SqSum}) - \text{Sum}^2}{N * (N - 1)}}$$

其中:

Sum = TBL 中值的总和

N = TBL 中值的数量

SqSum = TBL 中所有值的平方和

所有计算均使用 IEEE 浮点数值，所需的类型转换由该功能自动完成。

- 表格的第一个条目含有该表格的条目数(表格长度)。
- 表格的第二个条目含有第一个表格值。
- 表格条目的大小和计算值(OUT)由 E_TYPE 确定。

参数

参数	描述	数据类型	存储区	描述
EN	输入	BOOL	I、Q、M、D、L	使能输入端的信号状态为 1，激活该功能。
ENO	输出	BOOL	I、Q、M、D、L	如果该功能的执行无错误，使能输出端的信号状态为 1。
TBL	输入	*Pointer	I、Q、M、D	**指向数值表的起始位置
OUT	输入	*Pointer	I、Q、M、D	**指向计算所得标准偏差值的位置
E_TYPE	输入	BYTE	I、Q、M、D、L、P	指示表格条目的数据类型。有效数据类型：B#16#05 = INTB#16#07 = DINTB#16#08 = REAL
RET_VAL	输出	WORD	I、Q、M、D、L、P	如果该指令的执行没有错误，将返回值 W#16#0000。对于 W#16#0000 以外的其它值，参见"错误信息"
* 用于跨区域寄存器间接寻址的双字指针格式				
** TBL 和 OUT 必须指向同一数据块				

表 2-36 FC102 参数说明

错误信息

如果发生以下任何情况，则不执行该功能。ENO 的信号状态设置为 0，并相应地设置返回值：

RET_VAL 解释

W#16#0001 为功能的参数指定的存储器类型无效。

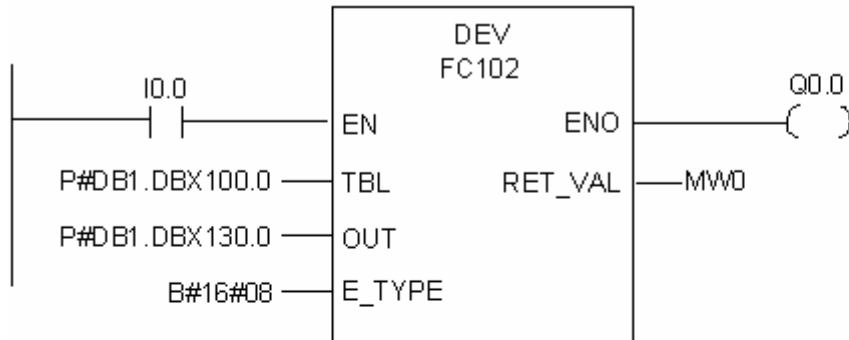
W#16#0002 E_TYPE 无效。

W#16#0004 表格长度为零。

实例

如果 I0.0 的信号状态为 1 (激活)，则执行 DEV 功能。在本例中，表格中第一个字表明，共有五个表格值。E_TYPE 表明表格值的数据类型是实型。

如果该功能的执行没有错误，ENO 和 Q0.0 的信号状态将设置为 1，RET_VAL 等于 W#16#0000。



执行前

TBL (table length) → DBW100 = W#16#0005
 DBD102 = 2.0
 DBD106 = 4.0
 DBD110 = 8.0
 DBD114 = 16.0
 DBD118 = 32.0

OUT → DBD130 = 0.0

执行后

OUT → DBD130 = 12.19836055

图 2-31: FC102 程序例子

2.10. 比较功能块

2.10.1. 索引矩阵比较: FB83

描述

IMC 功能块将最多 16 个已编程输入位 (IN_BIT0 至 IN_BIT15) 的信号状态与一个比较掩码的相应位进行比较。最多可编程 16 个带掩码的步骤。IN_BIT0 与 CMP_VAL [x,0] 比较，其中 x 是步骤号；IN_BIT1 与 CMP_VAL [x,1] 比较，等等。要比较的掩码步骤号由 CMP_STEP 值指示。未编程的输入位或未编程的掩码位的缺省信号状态为 FALSE。如果找到了匹配值，输出 OUT 的信号状态将设置为 1；否则设置为 0。

参数

参数	描述	数据类型	存储区	描述
EN	输入	BOOL	I、Q、M、D、L	使能输入端的信号状态为 1，激活该功能。

ENO	输出	BOOL	I、Q、M、D、L	如果该功能的执行无错误,使能输出端的信号状态为 1。
IN_BIT0	输入	BOOL	I、Q、M、D、L	要与掩码位 0 比较的输入位 0。
IN_BIT1	输入	BOOL	I、Q、M、D、L	要与掩码位 1 比较的输入位 1。
...
IN_BIT15	输入	BOOL	I、Q、M、D、L	要与掩码位 15 比较的输入位 15。
CMP_STEP	输入	BYTE	I、Q、M、D、L、P	要比较的掩码步骤号。
OUT	输出	BOOL	I、Q、M、D、L	信号状态 1 表示已找到一个匹配值。 信号状态 0 表示未找到匹配值。
ERR_CODE	输出	WORD	I、Q、M、D、L、P	如果该指令的执行没有错误,将返回 W#16#0000。对于 W#16#0000 以外的其它值,参见"错误信息"
CMP_VAL	静态	ARRAY OF BOOL	I、Q、M、D、L	比较掩码[0 至 15,0 至 15],其中索引的第一个数表示步骤号,第二个数表示掩码的位号

表 2-37 FB83 参数说明

错误信息

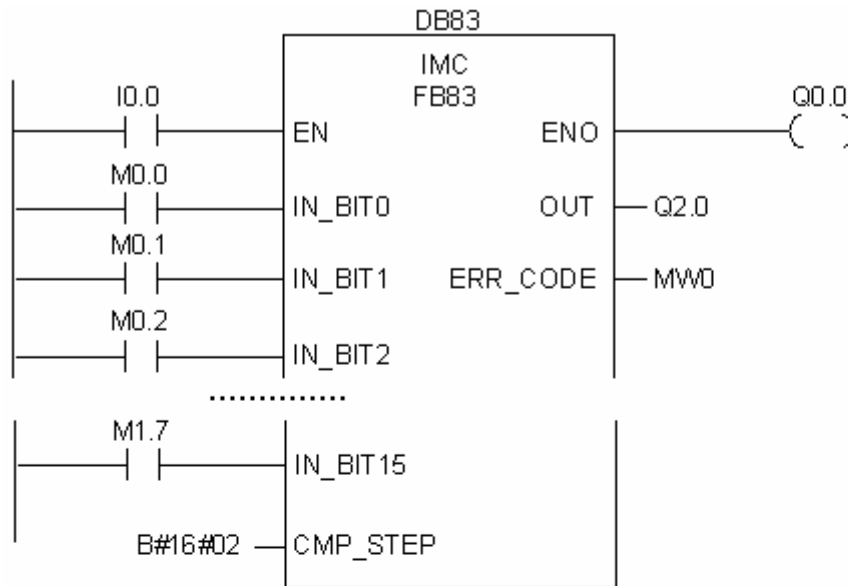
如果 CMP_STEP 的值大于 15,则不执行该功能块。ENO 的信号状态将设置为 0,ERR_CODE 等于 W#16#000A。

实例

如果输入 I0.0 的信号状态为 1(激活),则执行 IMC 功能块。在本例中,所有 16 个输入位都与步骤 2 的掩码进行比较(如 CMP_STEP 所示)。由于输入位与步骤 2 的掩码匹配,OUT 的信号状态设置为 TRUE。

如果该功能块的执行未发生错误,ENO 和 Q0.0 的信号状态将设置为 1,且 ERR_CODE 等于 W#16#0000。

注意: 可以使用"数据块编辑器"完成静态参数的初始化。



执行前

<i>IN_BIT0</i> → M0.0 = TRUE	<c ompare to>	<i>CMP_VAL</i> [2,0] → DBX12.0 = TRUE
<i>IN_BIT1</i> → M0.1 = TRUE	<c ompare to>	<i>CMP_VAL</i> [2,1] → DBX12.1 = TRUE
<i>IN_BIT2</i> → M0.2 = FALSE	<c ompare to>	<i>CMP_VAL</i> [2,2] → DBX12.2 = FALSE
<i>IN_BIT3</i> → M0.3 = TRUE	<c ompare to>	<i>CMP_VAL</i> [2,3] → DBX12.3 = TRUE
<i>IN_BIT4</i> → M0.4 = TRUE	<c ompare to>	<i>CMP_VAL</i> [2,4] → DBX12.4 = TRUE
<i>IN_BIT5</i> → M0.5 = FALSE	<c ompare to>	<i>CMP_VAL</i> [2,5] → DBX12.5 = FALSE
<i>IN_BIT6</i> → M0.6 = TRUE	<c ompare to>	<i>CMP_VAL</i> [2,6] → DBX12.6 = TRUE
<i>IN_BIT7</i> → M0.7 = TRUE	<c ompare to>	<i>CMP_VAL</i> [2,7] → DBX12.7 = TRUE
<i>IN_BIT8</i> → M1.0 = FALSE	<c ompare to>	<i>CMP_VAL</i> [2,8] → DBX13.0 = FALSE
<i>IN_BIT9</i> → M1.1 = TRUE	<c ompare to>	<i>CMP_VAL</i> [2,9] → DBX13.1 = TRUE
<i>IN_BIT10</i> → M1.2 = TRUE	<c ompare to>	<i>CMP_VAL</i> [2,10] → DBX13.2 = TRUE
<i>IN_BIT11</i> → M1.3 = FALSE	<c ompare to>	<i>CMP_VAL</i> [2,11] → DBX13.3 = FALSE
<i>IN_BIT12</i> → M1.4 = TRUE	<c ompare to>	<i>CMP_VAL</i> [2,12] → DBX13.4 = TRUE
<i>IN_BIT13</i> → M1.5 = TRUE	<c ompare to>	<i>CMP_VAL</i> [2,13] → DBX13.5 = TRUE
<i>IN_BIT14</i> → M1.6 = FALSE	<c ompare to>	<i>CMP_VAL</i> [2,14] → DBX13.6 = FALSE
<i>IN_BIT15</i> → M1.7 = TRUE	<c ompare to>	<i>CMP_VAL</i> [2,15] → DBX13.7 = TRUE
<i>OUT</i> → Q2.0 = FALSE		

执行后

OUT → Q2.0 = TRUE

图 2-32: FB83 程序例子

2.10.2. 扫描矩阵比较: FB84

描述

SMC 功能块将最多 16 个已编程输入位 (*IN_BIT0* 至 *IN_BIT15*) 的信号状态与每个步骤的比较掩码的相应位进行比较。比较从步骤 1 开始, 一直到最后一个编程步骤(LAST), 或直到找到匹配值。*IN_BIT0* 与 *CMP_VAL* [x,0]比较, 其中 x 是步骤号; *IN_BIT1* 与 *CMP_VAL* [x, 1] 比较, 等等。找到匹配值时, 输出 *OUT* 的信号状态设置为 1, 并将匹配掩码的步骤号写入 *OUT_STEP*。未编程的输入位或未编程的掩码位的缺省信号状态为 FALSE。如果多个步骤有匹配掩码, 则 *OUT_STEP* 仅

指示找到的第一个匹配掩码。如果没找到匹配掩码，输出 OUT 的信号状态设置为 0，且 OUT_STEP 值比 LAST 大一。

参数

参数	描述	数据类型	存储区	描述
EN	输入	BOOL	I、Q、M、D、L	使能输入端的信号状态为 1，激活该功能。
ENO	输出	BOOL	I、Q、M、D、L	如果该功能的执行无错误，使能输出端的信号状态为 1。
IN_BIT0	输入	BOOL	I、Q、M、D、L	要与掩码位 0 比较的输入位 0。
IN_BIT1	输入	BOOL	I、Q、M、D、L	要与掩码位 1 比较的输入位 1。
...
IN_BIT15	输入	BOOL	I、Q、M、D、L	要与掩码位 15 比较的输入位 15。
OUT	输出	BOOL	I、Q、M、D、L	信号状态 1 表示已找到一个匹配值。 信号状态 0 表示未找到匹配值。
ERR_CODE	输出	WORD	I、Q、M、D、L、P	如果该指令的执行没有错误，将返回 W#16#0000。对于 W#16#0000 以外的其它值，参见"错误信息"
OUT_STEP	输出	BYTE	I、Q、M、D、L、P	含有匹配掩码的步骤号，如果未找到匹配掩码，则为比 LAST 大一的步骤号
LAST	静态	BYTE	I、Q、M、D、L、P	指定为获得匹配掩码而要扫描的最后步骤的步骤号
CMP_VAL	静态	ARRAY OF BOOL	I、Q、M、D、L	比较掩码[0 至 15,0 至 15]，其中索引的第一个数表示步骤号，第二个数表示掩码的位号

表 2-38 FB84 参数说明

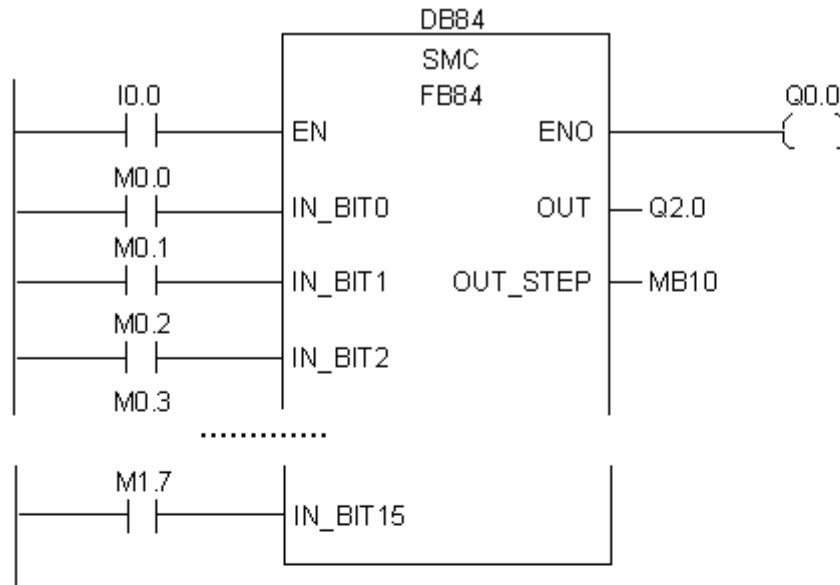
错误信息

如果 LAST 的值大于 15，则不执行该功能块。ENO 的信号状态将设置为 0，ERR_CODE 等于 W#16#000E。

实例

如果输入 I0.0 的信号状态为 1 (激活)，则执行 SMC 功能块。在本例中，所有 16 个输入位都与步骤 0 到步骤 5 (如 LAST 所示)的掩码进行比较，或者直到找到匹配值。由于步骤 2 的掩码与输入位匹配，因此只扫描步骤 0 至步骤 2 的掩码。

如果该功能块的执行未发生错误，ENO 和 Q0.0 的信号状态将设置为 1，ERR_CODE 等于 W#16#0000。



执行前

$IN_BIT0 \rightarrow M0.0 = TRUE$	<compare to>	$CMP_VAL [2,0] \rightarrow DBX12.0 = TRUE$
$IN_BIT1 \rightarrow M0.1 = TRUE$	<compare to>	$CMP_VAL [2,1] \rightarrow DBX12.1 = TRUE$
$IN_BIT2 \rightarrow M0.2 = FALSE$	<compare to>	$CMP_VAL [2,2] \rightarrow DBX12.2 = FALSE$
$IN_BIT3 \rightarrow M0.3 = TRUE$	<compare to>	$CMP_VAL [2,3] \rightarrow DBX12.3 = TRUE$
$IN_BIT4 \rightarrow M0.4 = TRUE$	<compare to>	$CMP_VAL [2,4] \rightarrow DBX12.4 = TRUE$
$IN_BIT5 \rightarrow M0.5 = FALSE$	<compare to>	$CMP_VAL [2,5] \rightarrow DBX12.5 = FALSE$
$IN_BIT6 \rightarrow M0.6 = TRUE$	<compare to>	$CMP_VAL [2,6] \rightarrow DBX12.6 = TRUE$
$IN_BIT7 \rightarrow M0.7 = TRUE$	<compare to>	$CMP_VAL [2,7] \rightarrow DBX12.7 = TRUE$
$IN_BIT8 \rightarrow M1.0 = FALSE$	<compare to>	$CMP_VAL [2,8] \rightarrow DBX13.0 = FALSE$
$IN_BIT9 \rightarrow M1.1 = TRUE$	<compare to>	$CMP_VAL [2,9] \rightarrow DBX13.1 = TRUE$
$IN_BIT10 \rightarrow M1.2 = TRUE$	<compare to>	$CMP_VAL [2,10] \rightarrow DBX13.2 = TRUE$
$IN_BIT11 \rightarrow M1.3 = FALSE$	<compare to>	$CMP_VAL [2,11] \rightarrow DBX13.3 = FALSE$
$IN_BIT12 \rightarrow M1.4 = TRUE$	<compare to>	$CMP_VAL [2,12] \rightarrow DBX13.4 = TRUE$
$IN_BIT13 \rightarrow M1.5 = TRUE$	<compare to>	$CMP_VAL [2,13] \rightarrow DBX13.5 = TRUE$
$IN_BIT14 \rightarrow M1.6 = FALSE$	<compare to>	$CMP_VAL [2,14] \rightarrow DBX13.6 = FALSE$
$IN_BIT15 \rightarrow M1.7 = TRUE$	<compare to>	$CMP_VAL [2,15] \rightarrow DBX13.7 = TRUE$

DB84

$OUT \rightarrow Q2.0 = FALSE$
 $OUT_STEP \rightarrow MB10 = B\#16\#00$

DB84

$LAST \rightarrow DB84 = B\#16\#05$

执行后

$OUT \rightarrow Q2.0 = TRUE$
 $OUT_STEP \rightarrow MB10 = B\#16\#02$

图 2-33: FB84 程序例子

注意：可以使用"数据块编辑器"完成静态参数的初始化。

3. 附录一推荐网址

3.1. 西门子自动化与驱动产品的在线技术支持

AS

西门子（中国）有限公司

工业自动化与驱动技术集团 客户服务与支持中心

网站首页: <http://www.ad.siemens.com.cn/Service/>

AS下载中心:

<http://www.ad.siemens.com.cn/download/DocList.aspx?TypeId=0&CatFirst=1&CatSecond=-1&CatThird=-1>

专家推荐精品文档: <http://www.ad.siemens.com.cn/Service/recommend.asp>

“找答案” AS版区: <http://www.ad.siemens.com.cn/service/answer/category.asp?cid=1027>

版权© 西门子（中国）有限公司 2001-2008 版权保留

复制、传播或者使用该文件或文件内容必须经过权利人书面明确同意。侵权者将承担权利人的全部损失。权利人保留一切权利，包括复制、发行，以及改编、汇编的权利。

西门子（中国）有限公司