机器控制器MP900/MP2000系列

新梯形图编辑器

用户手册

客户服务热线（帮您解决技术问题）

电话 **400-821-3680**　传真 **021-5385-2008**

周一至周五（节假日除外）9:00～11:30，12:30～16:30　※24小时接收传真

销售

・安川電機（中国）有限公司
上海市黄浦区黄河路21号鸿祥大厦11-12楼
邮编: 200003
电话: 021-53852200
传真: 021-53853299

・安川電機（中国）有限公司　北京分公司
北京市东城区东长安街1号东方广场东方经贸城西三办公楼1011室
邮编: 100738
电话: 010-85184086
传真: 010-85184082

・安川電機（中国）有限公司　广州分公司
广州市天河区体育东路138号金利来数码网络大厦1108-10室
邮编: 510620
电话: 020-38780005
传真: 020-38780565

・安川電機（中国）有限公司　成都分公司
成都市总府路2号时代广场B座711室
邮编: 610016
电话: 028-86719370
传真: 028-86719371

总公司

・株式会社　安川電機
日本福岡県北九州市八幡西区城石2-1
邮编: 806-0064
电话: 0081-93-645-8800
传真: 0081-93-631-8837

YASKAWA

资料编号　SICP C880700 20A

YASKAWA　机器控制器MP900/MP2000　新梯形图编辑器　用户手册　程序命令篇

YASKAWA

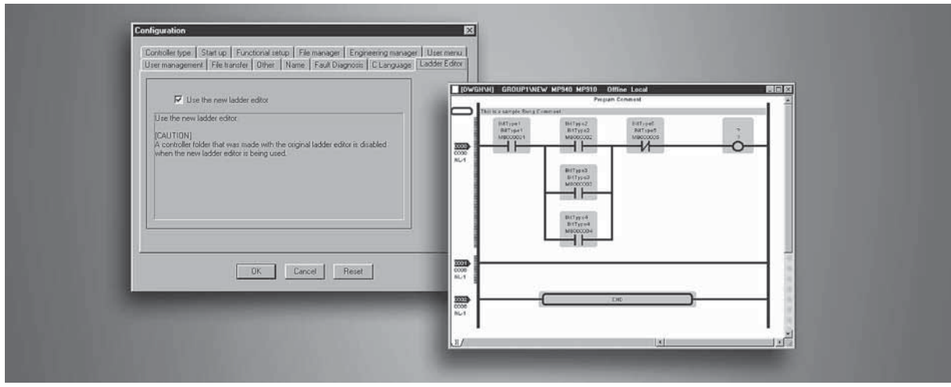
机器控制器MP900/MP2000系列

新梯形图编辑器

用户手册

程序命令篇

为了安全使用本产品，请务必阅读该使用说明书。
另外，请妥善保管该使用说明书，并将其交至最终用户手中。



资料编号　SICP C880700 20A

图标的说明

为使读者了解说明内容的区分，本书中设计了如下图标。并在必要的地方使用这些图标，以助读者理解。

表示需要熟记的重要事项。
同时也表示发生警报，但还不至于造成装置损坏的注意事项。

表示具体程序举例、操作实例。

表示补充事项或记住后会便于使用的功能。

表示对难于理解的用语进行解释，以及对事先没有说明而后出现的用语进行说明。

目录

图标的说明- iii
本手册的简介- viii
本手册的资料构成- viii
相关手册- ix
本手册的使用方法- xi
关于软件- xi

1章　梯形图指令

1.1　继电器电路指令 - 1-4
1.1.1　A触点指令(NOC) - 1-4
1.1.2　B触点指令(NCC) - 1-5
1.1.3　接通延时定时器指令(TON[10ms]) - - - - - - - - - - - - - - 1-5
1.1.4　断开延时定时器指令(TOFF[10ms]) - - - - - - - - - - - - - - 1-7
1.1.5　接通延时定时器指令(TON[1s]) - - - - - - - - - - - - - - - 1-8
1.1.6　断开延时定时器指令(TOFF[1s]) - - - - - - - - - - - - - - - 1-9
1.1.7　上升脉冲指令(ON-PLS) - - - - - - - - - - - - - - - - - - 1-10
1.1.8　下降脉冲指令(OFF-PLS) - - - - - - - - - - - - - - - - - - 1-12
1.1.9　线圈指令(COIL) - 1-13
1.1.10 置位线圈指令(S-COIL) - - - - - - - - - - - - - - - - - - 1-14
1.1.11 复位线圈指令(R-COIL) - - - - - - - - - - - - - - - - - - 1-15
1.2　数值运算指令 - 1-16
1.2.1　存储指令(STORE) - 1-16
1.2.2　加法指令(ADD) - 1-18
1.2.3　加法扩展指令(ADDX) - - - - - - - - - - - - - - - - - - - 1-20
1.2.4　减法指令(SUB) - 1-21
1.2.5　减法扩展指令(SUBX) - - - - - - - - - - - - - - - - - - - 1-23
1.2.6　乘法指令(MUL) - 1-24
1.2.7　除法指令(DIV) - 1-27
1.2.8　整型余数指令(MOD) - - - - - - - - - - - - - - - - - - - 1-29
1.2.9　实型余数指令(REM) - - - - - - - - - - - - - - - - - - - 1-30
1.2.10 增量指令(INC) - 1-31
1.2.11 减量指令(DEC) - 1-32
1.2.12 时间加法指令(TMADD) - - - - - - - - - - - - - - - - - - 1-33
1.2.13 时间减法指令(TMSUB) - - - - - - - - - - - - - - - - - - 1-35
1.2.14 时间经过指令(SPEND) - - - - - - - - - - - - - - - - - - 1-37
1.2.15 符号取反指令(INV) - - - - - - - - - - - - - - - - - - - 1-39
1.2.16 1的补码指令(COM) - - - - - - - - - - - - - - - - - - - 1-41
1.2.17 绝对值转换指令(ABS) - - - - - - - - - - - - - - - - - - 1-42
1.2.18 2进制转换指令(BIN) - - - - - - - - - - - - - - - - - - 1-44
1.2.19 BCD转换指令(BCD) - - - - - - - - - - - - - - - - - - - 1-45
1.2.20 校验转换指令(PARITY) - - - - - - - - - - - - - - - - - - 1-46
1.2.21 ASCII码转换1指令(ASCII) - - - - - - - - - - - - - - - - 1-47
1.2.22 ASCII码转换2指令(BINASC) - - - - - - - - - - - - - - - - 1-49
1.2.23 ASCII码转换3指令(ASCBIN) - - - - - - - - - - - - - - - - 1-50

1.3 逻辑运算 / 比较指令 - 1-51
1.3.1 逻辑与指令 (AND) - 1-51
1.3.2 逻辑或指令 (OR) - 1-52
1.3.3 逻辑异或指令 (XOR) - 1-53
1.3.4 比较指令 (＜) - 1-54
1.3.5 比较指令 (≦) - 1-55
1.3.6 比较指令 (＝) - 1-56
1.3.7 比较指令 (≠) - 1-57
1.3.8 比较指令 (≧) - 1-58
1.3.9 比较指令 (＞) - 1-59
1.3.10 范围检查指令 (RCHK) - 1-60
1.4 程序控制指令 - 1-63
1.4.1 图调用指令 (SEE) - 1-63
1.4.2 运动程序调用指令 (MSEE) - 1-64
1.4.3 函数调用指令 (FUNC) - 1-65
1.4.4 连续执行型直接输入指令 (INS) - - - - - - - - - - - - - - - - - - - 1-66
1.4.5 连续执行型直接输出指令 (OUTS) - - - - - - - - - - - - - - - - - - 1-69
1.4.6 扩展程序执行指令 (XCALL) - 1-71
1.4.7 WHILE 指令 (WHILE, END_WHILE) - - - - - - - - - - - - - - - - - - 1-72
1.4.8 IF 指令 (IF, END_IF) - 1-74
1.4.9 IF 指令 (IF, ELSE, END_IF) - 1-76
1.4.10 FOR 指令 (FOR, END_FOR) - 1-78
1.4.11 EXPRESSION 指令 (EXPRESSION) - - - - - - - - - - - - - - - - - - - 1-80
1.5 基本函数指令 - 1-81
1.5.1 平方根指令 (SQRT) - 1-81
1.5.2 正弦指令 (SIN) - 1-83
1.5.3 余弦指令 (COS) - 1-85
1.5.4 正切指令 (TAN) - 1-87
1.5.5 反正弦指令 (ASIN) - 1-88
1.5.6 反余弦指令 (ACOS) - 1-89
1.5.7 反正切指令 (ATAN) - 1-90
1.5.8 指数指令 (EXP) - 1-92
1.5.9 自然对数指令 (LN) - 1-93
1.5.10 常用对数指令 (LOG) - 1-94
1.6 数据操作指令 - 1-95
1.6.1 位循环左移指令 (ROTL) - 1-95
1.6.2 位循环右移指令 (ROTR) - 1-97
1.6.3 位传送指令 (MOVB) - 1-99
1.6.4 字传送指令 (MOVW) - 1-101
1.6.5 替换传送指令 (XCHG) - 1-103
1.6.6 表初始化指令 (SETW) - 1-105
1.6.7 字节→字展开指令 (BEXTD) - 1-107
1.6.8 字→字节压缩指令 (BPRESS) - 1-109
1.6.9 数据检索指令 (BSRCH) - 1-111
1.6.10 分类指令 (SORT) - 1-113
1.6.11 位左移指令 (SHFTL) - 1-114
1.6.12 位右移指令 (SHFTR) - 1-116
1.6.13 字复制指令 (COPYW) - 1-117
1.6.14 字节交换指令 (BSWAP) - 1-119

1.7 DDC 指令- -1-121
1.7.1 死区 A 指令 (DZA) - 1-121
1.7.2 死区 B 指令 (DZB) - 1-123
1.7.3 上下限值指令 (LIMIT) - 1-125
1.7.4 PI 控制指令 (PI) - 1-128
1.7.5 PD 控制指令 (PD) - 1-131
1.7.6 PID 控制指令 (PID) - 1-134
1.7.7 一阶延迟指令 (LAG) - 1-138
1.7.8 相位超前滞后指令 (LLAG)- 1-141
1.7.9 函数发生器指令 (FGN) - 1-144
1.7.10 反函数发生器指令 (IFGN)- 1-148
1.7.11 直线加减速器 1 指令 (LAU) - - - - - - - - - - - - - - - - - - - 1-152
1.7.12 直线加减速器 2 指令 (SLAU)- - - - - - - - - - - - - - - - - - - 1-156
1.7.13 脉宽调制指令 (PWM) - 1-163
1.8 表数据操作指令 -1-165
1.8.1 块调出指令 (TBLBR) - 1-165
1.8.2 块写入指令 (TBLBW) - 1-167
1.8.3 行检索指令：纵向 (TBLSRL)- 1-169
1.8.4 列检索指令：横向 (TBLSRC)- 1-171
1.8.5 块清除指令 (TBLCL) - 1-173
1.8.6 表间块传送指令 (TBLMV) - 1-175
1.8.7 Q 表调出指令 (QTBLR，QTBLRI) - - - - - - - - - - - - - - - - - 1-177
1.8.8 Q 表写入指令 (QTBLW，QTBLWI) - - - - - - - - - - - - - - - - - 1-179
1.8.9 Q 指针清除指令 (QTBLCL)- 1-181

2 章 系统标准函数指令

2.1 信息函数 -2-2
2.1.1 信息发送函数 (MSG-SND) - 2-2
2.1.2 信息接收函数 (MSG-RCV) - 2-15
2.2 示踪函数 - 2-24
2.2.1 示踪函数 (TRACE) - 2-24
2.2.2 数据示踪调出函数 (DTRC-RD) - - - - - - - - - - - - - - - - - - - 2-25
2.2.3 故障示踪调出函数 (FTRC-RD) - - - - - - - - - - - - - - - - - - - 2-28
2.2.4 变频器示踪调出函数 (ITRC-RD) - - - - - - - - - - - - - - - - - - 2-32
2.3 变频器函数 - 2-35
2.3.1 变频器常数写入函数 (ICNS-WR) - - - - - - - - - - - - - - - - - - 2-35
2.3.2 变频器常数调出函数 (ICNS-RD) - - - - - - - - - - - - - - - - - - 2-40
2.4 其他的函数 - 2-43
2.4.1 计数函数 (COUNTER) - 2-43
2.4.2 先进先出函数 (FINFOUT) - 2-45

附录 A　EXPRESSION

A.1　数式 －－－－－－－－－－－－－－－－－－－－－－－－－－－ A-2
A.1.1　运算符－－－－－－－－－－－－－－－－－－－－－－－－－－－－ A-2
A.1.2　运算对象－－－－－－－－－－－－－－－－－－－－－－－－－－－ A-4
A.1.3　函数－－－－－－－－－－－－－－－－－－－－－－－－－－－－－ A-4
A.2　可识别的表达式种类 －－－－－－－－－－－－－－－－－－－ A-5
A.2.1　算术运算符－－－－－－－－－－－－－－－－－－－－－－－－－－ A-5
A.2.2　比较运算符－－－－－－－－－－－－－－－－－－－－－－－－－－ A-5
A.2.3　逻辑运算符－－－－－－－－－－－－－－－－－－－－－－－－－－ A-5
A.2.4　赋值运算符－－－－－－－－－－－－－－－－－－－－－－－－－－ A-6
A.2.5　函数－－－－－－－－－－－－－－－－－－－－－－－－－－－－－ A-6
A.2.6　其他－－－－－－－－－－－－－－－－－－－－－－－－－－－－－ A-6
A.3　在梯形图程序中的应用 －－－－－－－－－－－－－－－－－ A-7
A.3.1　IF 指令句的条件表达式 －－－－－－－－－－－－－－－－－－－－ A-7
A.3.2　WHILE 指令句的条件表达式－－－－－－－－－－－－－－－－－－ A-7
A.3.3　EXPRESSION 指令句的运算表达式 －－－－－－－－－－－－－－ A-7

本手册的简介

- 本手册详细地说明了支持 MP900/MP2000 系列（以下称 MP 系列）的设计与维护的新梯形图编辑器、软件的操作方法。
- 本手册以充分理解 Microsoft Windows 95/98/2000/NT 的操作方法的人员为对象。
  关于 Windows 的打开 / 关闭及鼠标操作、Windows 应用程序的一般操作等，请参阅计算机所附的说明书。
- 为了能正确使用新梯形图编辑器，请仔细阅读本手册。请妥善保管本手册，以便需要时参阅。

本手册的资料构成

MP900 系列有 MP910、MP920、MP930、MP940 四种。

MP2000 系列有 MP2100 和 MP2300 两种。

资料构成根据上述产品构成修订。相关手册在下一页有介绍，敬请参阅。

相关手册

■ 机器控制器 MP900/MP2000 系列的相关手册包括下表所示的内容。请根据需要进行阅读。

用户手册名	资料编号	适用机型					
		MP910	MP920	MP930	MP940	MP2100	MP2300
机器控制器 MP930 用户手册 设计与维护篇	SI-C887-1.1			○			
机器控制器 MP900/MP2000 系列 用户手册 梯形图程序篇	SI-C887-1.2	○	○	○	○	○	○
机器控制器 MP900/MP2000 系列 用户手册 运动程序篇	SIZ-C887-1.3	○	○	○	○	○	○
机器控制器 MP900 系列 用户手册 示教操作器篇	SI-C887-1.6		○	○			
机器控制器 MP920 用户手册 设计与维护篇	SIZ-C887-2.1		○				
机器控制器 MP900 系列 编程装置用 软件 MPE720 用户手册 操作说明简易版	SIZ-C887-2.3	○	○	○	○		
机器控制器 MP920 用户手册 运动模块篇	SIZ-C887-2.5		○				
机器控制器 MP920 用户手册 通讯模块篇	SIZ-C887-2.6		○				
机器控制器 MP920 设置手册 EMC 指令用	SIBZ-C887-2.50		○				
机器控制器 MP910 用户手册 设计与维护篇	SIZ-C887-3.1	○					
机器控制器 MP940 用户手册 设计与维护篇	SIZ-C887-4.1				○		
机器控制器 MP940 设置手册 EMC 指令用	SIBZ-C887-4.50				○		
机器控制器 MP900 系列 用户手册 MECHATROLINK 篇	SIZ-C887-5.1	○	○	○	○		
机器控制器 MP900 系列 用户手册 260I/F DeviceNet 篇	SIZ-C887-5.2		○		○		

用户手册名	资料编号	适用机型					
		MP910	MP920	MP930	MP940	MP2100	MP2300
机器控制器 MP900 系列 MPPanel 用户手册	SIZ-C887-10.1		○	○	○		
机器控制器 MP900 系列 MPLogger 用户手册	SIZ-C887-11.1		○	○	○		
机器控制器 MP900 系列 MPLoader (Server) 用户手册	SIZ-C887-12.1		○	○	○		
机器控制器 MP900 系列 MPLoader (Client) 用户手册	SIZ-C887-12.2		○	○	○		
机器控制器 MP900/MP2000 系列 用户手册 新梯形图编辑器程序指令篇	SIZ-C887-13.1	○	○	○	○	○	○
机器控制器 MP900/MP2000 系列 用户手册 新梯形图编辑器操作篇	SIZ-C887-13.2	○	○	○	○	○	○
机器控制器 MP2300 用户手册 基本模块篇	SIJPC88070003A						○
机器控制器 MP2300 用户手册 通讯模块篇	SIJPC88070004A						○
机器控制器 MP900/MP2000 系列 编程装置用 用户手册标准版	SIJPC88070005A	○	○	○	○	○	○
机器控制器 MP2100 用户手册 设计与维护篇	SIJPC88070001A					○	

本手册的使用方法

■ 本手册的使用对象

本手册以下列人员为对象。

- 进行 MP900/MP2000 系列系统设计的人员
- 进行 MP900/MP2000 系列运动程序编程的人员
- 进行 MP900/MP2000 系列梯形图程序编程的人员

■ 缩略语及缩写符号

本手册使用如下所示的缩略语及缩写符号。

- PLC　：机器控制器的总称
- MPE720：编程装置用软件的总称

关于软件

■ 使用注意事项

- 本软件请在 1 台特定的电脑上使用。需要在其他电脑上使用时，请另行购买。
- 严禁复制本软件，并将其用于编程装置用之外的用途。
- 请妥善保管本软件的软盘。
- 严禁对本软件进行反编译、反汇编。
- 未经本公司许可，严禁将本软件的一部分或全部转让、交换、转借给第三者使用。

■ 注册商标

- Windows、Windows95/98/2000/NT 是美国 Microsoft 公司的注册商标。
- Pentium 是美国 Intel 公司的注册商标。
- 以太网（Ethernet）是美国 Xerox 公司的注册商标。

1 章

梯形图指令

本章对继电器电路指令、数值运算指令、逻辑运算 / 比较指令、程序控制指令、基本函数指令、数据操作指令、DDC 指令以及表数据操作指令进行了说明。

1 章　梯形图指令

1.1　继电器电路指令 - 1-4
1.1.1　A 触点指令 (NOC) - 1-4
1.1.2　B 触点指令 (NCC) - 1-5
1.1.3　接通延时定时器指令 (TON[10ms]) - - - - - - - - - - - - - - - - 1-5
1.1.4　断开延时定时器指令 (TOFF[10ms]) - - - - - - - - - - - - - - - - 1-7
1.1.5　接通延时定时器指令 (TON[1s]) - - - - - - - - - - - - - - - - - 1-8
1.1.6　断开延时定时器指令 (TOFF[1s]) - - - - - - - - - - - - - - - - - 1-9
1.1.7　上升脉冲指令 (ON-PLS) - 1-10
1.1.8　下降脉冲指令 (OFF-PLS) - 1-12
1.1.9　线圈指令 (COIL) - 1-13
1.1.10 置位线圈指令 (S-COIL) - 1-14
1.1.11 复位线圈指令 (R-COIL) - 1-15

1.2　数值运算指令 - 1-16
1.2.1　存储指令 (STORE) - 1-16
1.2.2　加法指令 (ADD) - 1-18
1.2.3　加法扩展指令 (ADDX) - 1-20
1.2.4　减法指令 (SUB) - 1-21
1.2.5　减法扩展指令 (SUBX) - 1-23
1.2.6　乘法指令 (MUL) - 1-24
1.2.7　除法指令 (DIV) - 1-27
1.2.8　整型余数指令 (MOD) - 1-29
1.2.9　实型余数指令 (REM) - 1-30
1.2.10 增量指令 (INC) - 1-31
1.2.11 减量指令 (DEC) - 1-32
1.2.12 时间加法指令 (TMADD) - 1-33
1.2.13 时间减法指令 (TMSUB) - 1-35
1.2.14 时间经过指令 (SPEND) - 1-37
1.2.15 符号取反指令 (INV) - 1-39

1.2.16 1的补码指令(COM) - 1-41
1.2.17 绝对值转换指令(ABS) - 1-42
1.2.18 2进制转换指令(BIN) - 1-44
1.2.19 BCD转换指令(BCD) - 1-45
1.2.20 校验转换指令(PARITY) - 1-46
1.2.21 ASCII码转换1指令(ASCII) - - - - - - - - - - - - - - - - - - - 1-47
1.2.22 ASCII码转换2指令(BINASC) - - - - - - - - - - - - - - - - - - - 1-49
1.2.23 ASCII码转换3指令(ASCBIN) - - - - - - - - - - - - - - - - - - - 1-50

1.3 逻辑运算/比较指令 - 1-51
1.3.1 逻辑与指令(AND) - 1-51
1.3.2 逻辑或指令(OR) - 1-52
1.3.3 逻辑异或指令(XOR) - 1-53
1.3.4 比较指令(＜) - 1-54
1.3.5 比较指令(≦) - 1-55
1.3.6 比较指令(＝) - 1-56
1.3.7 比较指令(≠) - 1-57
1.3.8 比较指令(≧) - 1-58
1.3.9 比较指令(＞) - 1-59
1.3.10 范围检查指令(RCHK) - 1-60

1.4 程序控制指令 - 1-63
1.4.1 图调用指令(SEE) - 1-63
1.4.2 运动程序调用指令(MSEE) - 1-64
1.4.3 函数调用指令(FUNC) - 1-65
1.4.4 连续执行型直接输入指令(INS) - - - - - - - - - - - - - - - - - 1-66
1.4.5 连续执行型直接输出指令(OUTS) - - - - - - - - - - - - - - - - - 1-69
1.4.6 扩展程序执行指令(XCALL) - - - - - - - - - - - - - - - - - - - 1-71
1.4.7 WHILE指令(WHILE，END_WHILE) - - - - - - - - - - - - - - - - - 1-72
1.4.8 IF指令(IF，END_IF) - 1-74
1.4.9 IF指令(IF，ELSE，END_IF) - - - - - - - - - - - - - - - - - - - 1-76
1.4.10 FOR指令(FOR，END_FOR) - 1-78
1.4.11 EXPRESSION指令(EXPRESSION) - - - - - - - - - - - - - - - - - 1-80

1.5 基本函数指令 - 1-81
1.5.1 平方根指令(SQRT) - 1-81
1.5.2 正弦指令(SIN) - 1-83
1.5.3 余弦指令(COS) - 1-85
1.5.4 正切指令(TAN) - 1-87
1.5.5 反正弦指令(ASIN) - 1-88
1.5.6 反余弦指令(ACOS) - 1-89
1.5.7 反正切指令(ATAN) - 1-90
1.5.8 指数指令(EXP) - 1-92
1.5.9 自然对数指令(LN) - 1-93
1.5.10 常用对数指令(LOG) - 1-94

1.6 数据操作指令 - 1-95
1.6.1 位循环左移指令(ROTL) - 1-95
1.6.2 位循环右移指令(ROTR) - 1-97
1.6.3 位传送指令(MOVB) - 1-99
1.6.4 字传送指令(MOVW) - 1-101
1.6.5 替换传送指令(XCHG) - 1-103
1.6.6 表初始化指令(SETW) - 1-105

1.6.7 字节→字展开指令（BEXTD）- - - - - - - - - - - - - - - - - - - 1-107
1.6.8 字→字节压缩指令（BPRESS） - - - - - - - - - - - - - - - - - - - 1-109
1.6.9 数据检索指令（BSRCH）- 1-111
1.6.10 分类指令（SORT） - 1-113
1.6.11 位左移指令（SHFTL）- 1-114
1.6.12 位右移指令（SHFTR）- 1-116
1.6.13 字复制指令（COPYW）- 1-117
1.6.14 字节交换指令（BSWAP）- 1-119
1.7 DDC 指令- 1-121
1.7.1 死区 A 指令（DZA）- 1-121
1.7.2 死区 B 指令（DZB）- 1-123
1.7.3 上下限值指令（LIMIT）- 1-125
1.7.4 PI 控制指令（PI） - 1-128
1.7.5 PD 控制指令（PD） - 1-131
1.7.6 PID 控制指令（PID） - 1-134
1.7.7 一阶延迟指令（LAG）- 1-138
1.7.8 相位超前滞后指令（LLAG） - 1-141
1.7.9 函数发生器指令（FGN）- 1-144
1.7.10 反函数发生器指令（IFGN） - - - - - - - - - - - - - - - - - - - 1-148
1.7.11 直线加减速器 1 指令（LAU）- - - - - - - - - - - - - - - - - - - 1-152
1.7.12 直线加减速器 2 指令（SLAU） - - - - - - - - - - - - - - - - - - 1-156
1.7.13 脉宽调制指令（PWM）- 1-163
1.8 表数据操作指令 - 1-165
1.8.1 块调出指令（TBLBR）- 1-165
1.8.2 块写入指令（TBLBW）- 1-167
1.8.3 行检索指令：纵向（TBLSRL） - - - - - - - - - - - - - - - - - - - 1-169
1.8.4 列检索指令：横向（TBLSRC） - - - - - - - - - - - - - - - - - - - 1-171
1.8.5 块清除指令（TBLCL）- 1-173
1.8.6 表间块传送指令（TBLMV）- 1-175
1.8.7 Q 表调出指令（QTBLR，QTBLRI） - - - - - - - - - - - - - - - - - 1-177
1.8.8 Q 表写入指令（QTBLW，QTBLWI） - - - - - - - - - - - - - - - - - 1-179
1.8.9 Q 指针清除指令（QTBLCL）- 1-181

1.1　继电器电路指令

1.1.1　A 触点指令 (NOC)

■ 概要

映像寄存器的值为 1(ON) 时，将位输出置为 ON。反之，映像寄存器的值为 0(OFF) 时，将位输出置为 OFF。

■ 格式

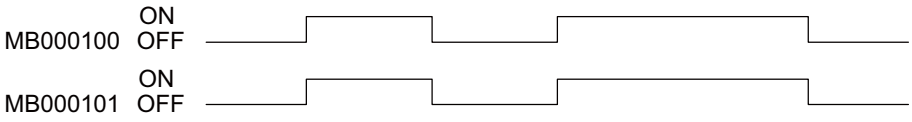
标记：NOC
全称：NO Contact
类别：RELAY
图标：

■ 参数

参数名称	设定
继电器编号	• 所有比特型寄存器 • 同上带下标字母

■ 程序举例

当 MB000100 ON 时，MB000101 为 ON。

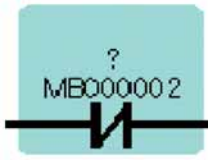


1.1.2 B 触点指令（NCC）

■ 概要

映像寄存器的值为 1(ON) 时，将位输出置为 OFF。反之，映像寄存器的值为 0(OFF) 时，将位输出置为 ON。

■ 格式



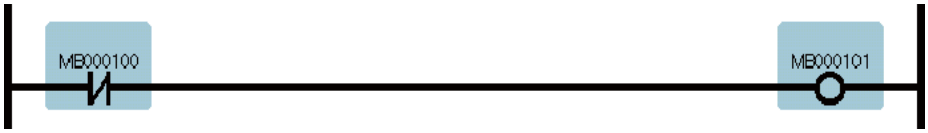
标记：NCC
全称：NC Contact
类别：RELAY
图标：

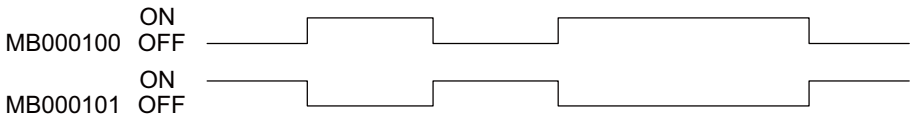
■ 参数

参数名称	设定
继电器编号	• 所有比特型寄存器 • 同上带下标字母

■ 程序举例

当 MB000100 ON 时，MB000101 为 OFF。





1.1.3 接通延时定时器指令（TON[10ms]）

■ 概要

位输入为 ON 时，进行计时。当 “计数值＝设定值”时，位输出为 ON。

计数中位输入为 OFF 时，停止计时。位输入再次为 ON 时，从 (0) 开始计数。同时，计数用的寄存器中存储实际计数时间（以 10ms 为单位）的值。

■ 格式

标记：TON[10ms]

全称：On-Delay Timer[10ms]

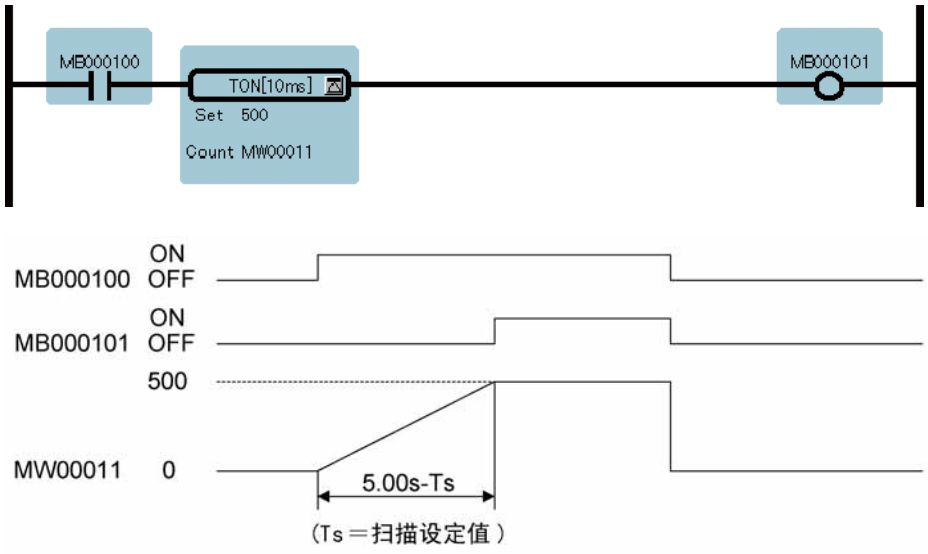
类别：RELAY

图标：

■ 参数

参数名称	设定
Set（设定值）	• 所有整型寄存器 • 同上带下标字母（0 ～ 65535(655.35s)：10ms 刻度） • 常数
Count（计数值）	• 整型寄存器（#、C 寄存器除外） • 同上带下标字母

■ 程序举例



重要

MW00011 作为计时器的计数用寄存器而工作。为避免重复，请务必设定未使用的寄存器。

1.1.4 断开延时定时器指令（TOFF[10ms]）

■ 概要

位输入为 OFF 时，进行计时。当“计数值＝设定值”时，位输出为 OFF。

计数中最近的位输入为 ON 时，停止计时。位输入再次为 OFF 时，从（0）开始计数。同时，计数用的寄存器中存储实际计数时间（以 10ms 为单位）的值。

■ 格式

标记：TOFF[10ms]

全称：Off-Delay Timer[10ms]

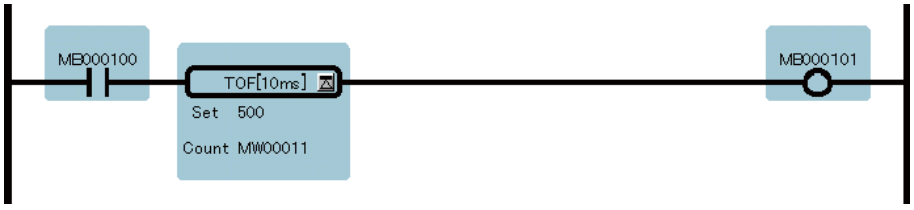
类别：RELAY

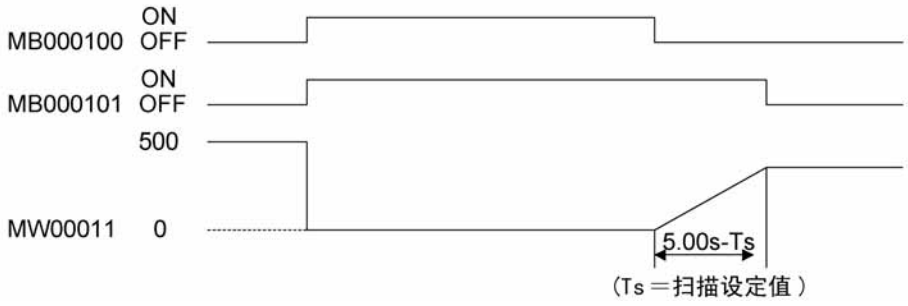
图标：

■ 参数

参数名称	设定
Set（设定值）	• 所有整型寄存器 • 同上带下标字母（0 ～ 65535(655.35s)：10ms 刻度） • 常数
Count（计数值）	• 整型寄存器（#、C 寄存器除外） • 同上带下标字母

■ 程序举例





重要

MW00011 作为计时器的计数用寄存器而工作。为避免重复，请务必设定未使用的寄存器。

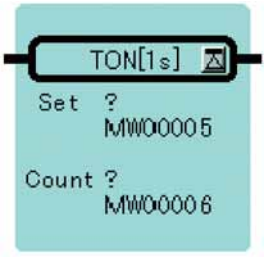
1.1.5　接通延时定时器指令（TON[1s]）

■ 概要

位输入为 ON 时，进行计时。当“计数值＝设定值”时，位输出为 ON。

计数中最近的位输入为 OFF 时，停止计时。位输入再次为 ON 时，从（0）开始计数。同时，计数用的寄存器中存储实际计数时间（以 1s 为单位）的值。

■ 格式

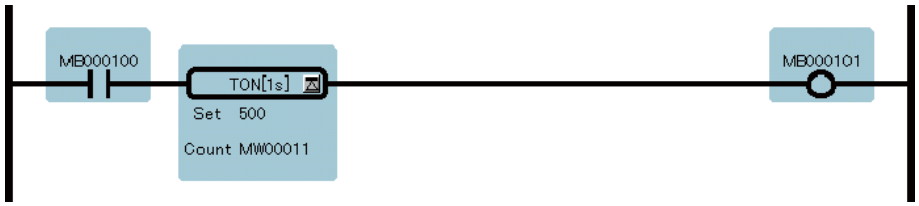


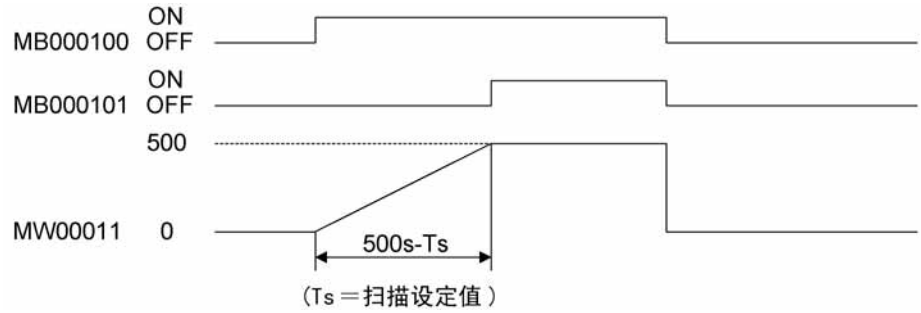
标记：TON[1s]
全称：On-Delay Timer[1s]
类别：RELAY
图标： 

■ 参数

参数名称	设定
Set（设定值）	• 所有整型寄存器 • 同上带下标字母（0 ～ 65535s：1s 刻度） • 常数
Count（计数值）	• 整型寄存器（#、C 寄存器除外） • 同上带下标字母

■ 程序举例





重要

MW00011 作为计时器的计数用寄存器而工作。为避免重复，请务必设定未使用的寄存器。

1.1.6 断开延时定时器指令（TOFF[1s]）

■ 概要

位输入为 OFF 时，进行计时。当“计数值＝设定值”时，位输出为 OFF。

计数中位输入为 OFF 时，停止计时。位输入再次为 ON 时，从（0）开始计数。同时，计数用的寄存器中存储实际计数时间（以 1s 为单位）的值。

■ 格式

标记：TOFF[1s]

全称：Off-Delay Timer[1s]

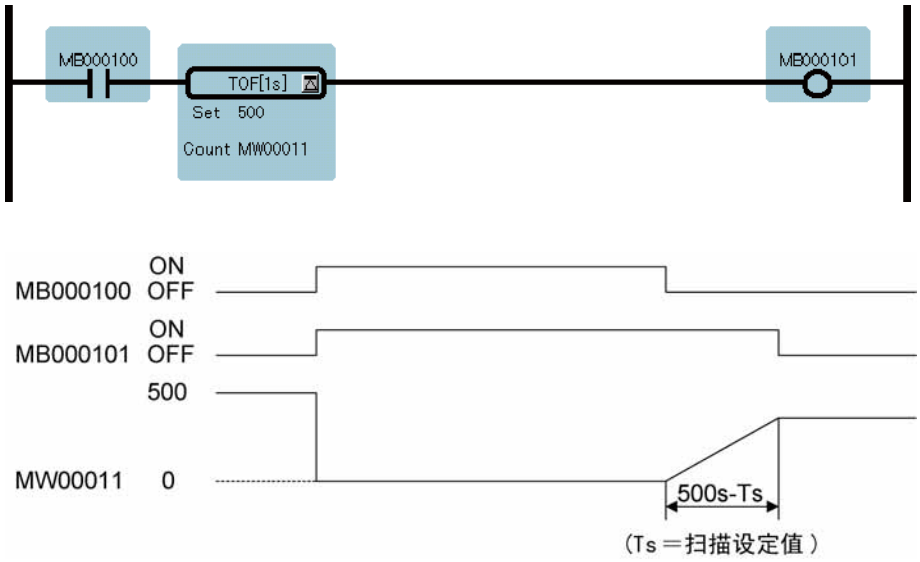
类别：RELAY

图标： 

■ 参数

参数名称	设定
Set（设定值）	• 所有整型寄存器 • 同上带下标字母（0 ～ 65535s：1s 刻度） • 常数
Count（计数值）	• 整型寄存器（#、C 寄存器除外） • 同上带下标字母

■ 程序举例



重要

MW00011 作为计时器的计数用寄存器而工作。为避免重复，请务必设定未使用的寄存器。

1.1.7　上升脉冲指令 (ON-PLS)

■ 概要

位输入状态从 OFF 变为 ON 时，位输出在一个扫描周期内为 ON。指定的寄存器用于保存位输出上次的值。

■ 格式

标记：ON-PLS
全称：Rise Pulse
类别：RELAY
图标： 

■ 参数

参数名称	设定
用于保存位输入上次值的寄存器编号	• 比特型寄存器 (#、C 寄存器除外) • 同上带下标字母

■ 程序举例

当 IB00001 从 OFF 变为 ON 时，MB000101 在一个扫描周期内为 ON。MB000100 用于保存 IB00001 上次的值。

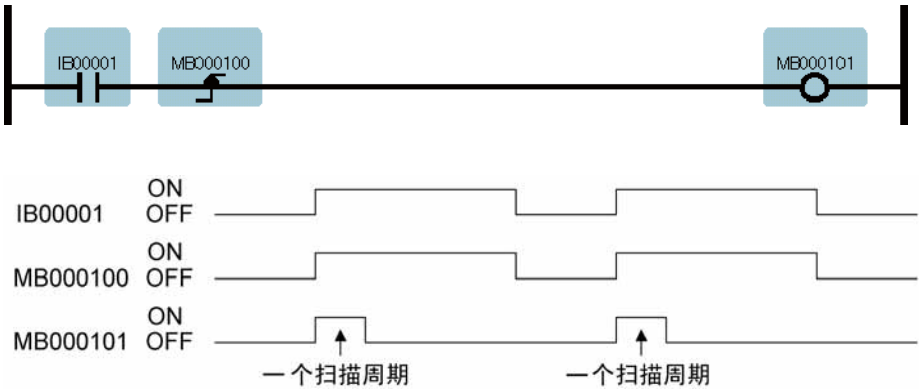


表 1.1 表示上升脉冲指令的寄存器状态。

表 1.1 上升脉冲指令的寄存器状态

输入		结果	
IB00001	MB000100 （IB00001 上次的值）	MB000100 （IB00001 的保存）	MB000100
OFF	OFF	OFF	OFF
OFF	ON	OFF	OFF
ON	OFF	ON	ON
ON	ON	ON	OFF

（注）在该程序举例中，检测出的并非 MB000100 的上升，而是 IB00001 的上升。MB000100 仅用于保存 IB00001 上次的值。

1.1.8　下降脉冲指令 (OFF-PLS)

■ 概要

位输入从 ON 变为 OFF 时，位输出在一个扫描周期内为 ON。指定的寄存器用于保存位输出上次的值。

■ 格式

标记：OFF-PLS
全称：Fall Pulse
类别：RELAY
图标： 

■ 参数

参数名称	设定
用于保存位输入上次值的寄存器编号	• 比特型寄存器 (#、C 寄存器除外) • 同上带下标字母

■ 程序举例

当 IB00001 从 ON 变为 OFF 时，MB000101 在一个扫描周期内为 ON。MB000100 用于保存 IB00001 上次的值。

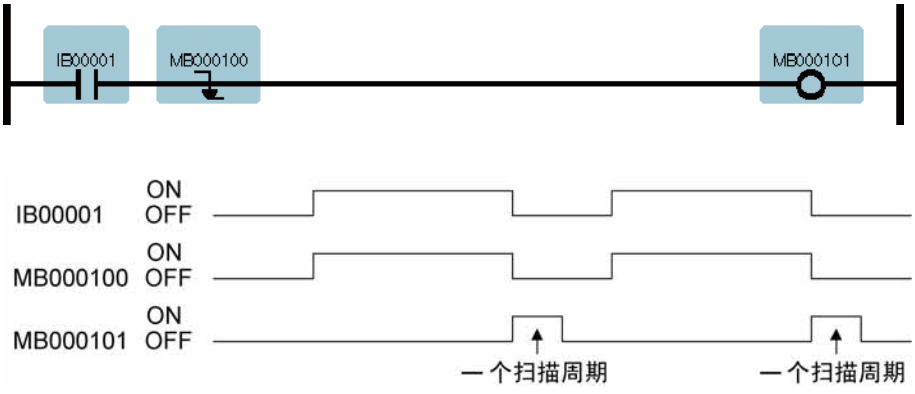


表 1.2 表示下降脉冲指令的寄存器状态。

表 1.2 下降脉冲指令的寄存器状态

输入		结果	
IB00001	MB000100 （IB00001 上次的值）	MB000100 （IB00001 的保存）	MB000101
OFF	OFF	OFF	OFF
OFF	ON	OFF	ON
ON	OFF	ON	OFF
ON	ON	ON	OFF

（注）在该程序举例中，检测出的并非 MB000100 的下降，而是 IB00001 的下降。MB000100 仅用于保存 IB00001 上次的值。

1.1.9 线圈指令（COIL）

■ 概要

位输入 ON 时，将映像寄存器的值置为 1(ON)。位输入 OFF 时，置为 0(OFF)。

■ 格式

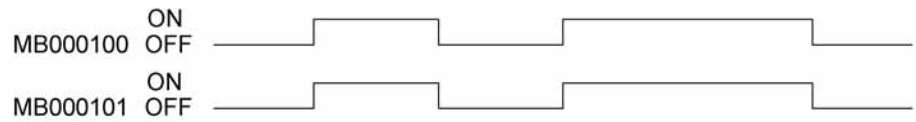
标记：COIL
全称：Coil
类别：RELAY
图标： 

■ 参数

参数名称	设定
线圈编号	• 比特型寄存器（#、C 寄存器除外） • 同上带下标字母

■ 程序举例

MB000100 为 ON 时，MB000101 为 ON。

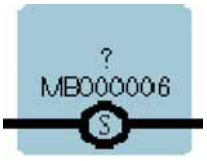


1.1.10　置位线圈指令 (S-COIL)

■ 概要

执行条件成立时，将输出置为 ON，并保持 ON 的状态。

■ 格式



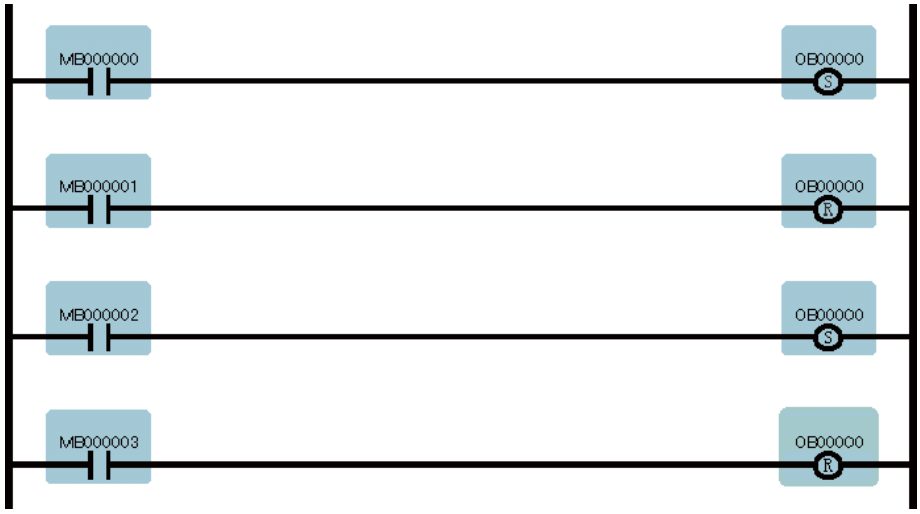
标记：S-Coil
全称：Set Coil
类别：RELAY
图标：

■ 参数

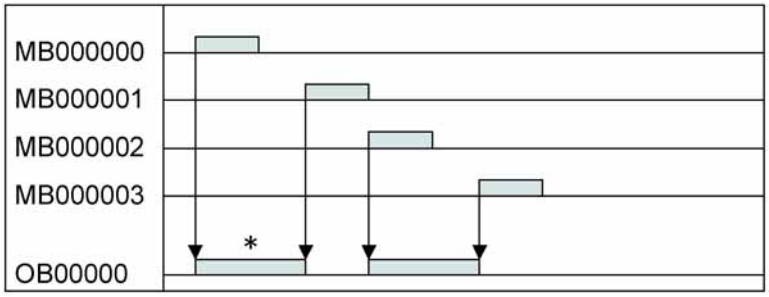
参数名称	设定
线圈编号	• 比特型寄存器 (#、C 寄存器除外) • 同上带下标字母

■ 程序举例

多次指定同一输出地点时



上例将进行如下动作。



* OB00000 为 OFF 时，在置位线圈指令下 OB00000 为 ON 状态。

1.1.11 复位线圈指令（R-COIL）

■ 概要

执行条件成立时，将输出置为 OFF，并保持 OFF 的状态。

■ 格式

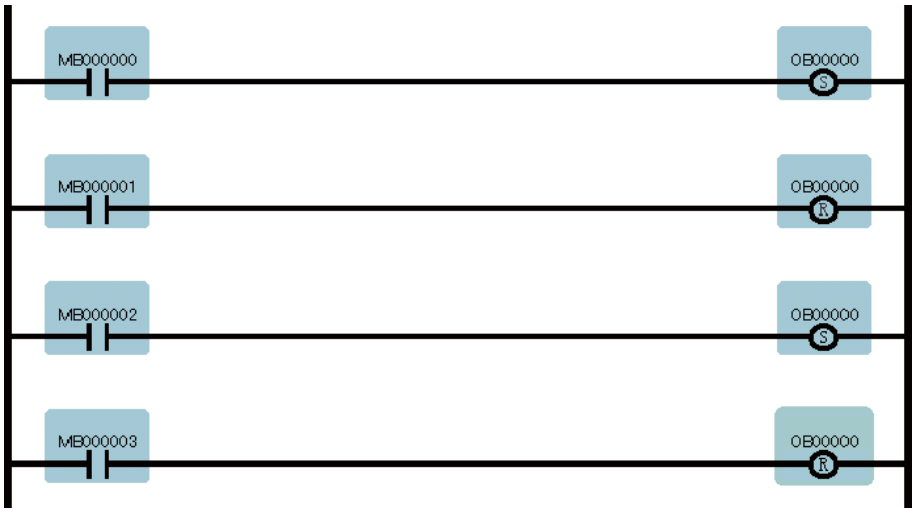
标记：R-Coil
全称：Reset Coil
类别：RELAY
图标：

■ 参数

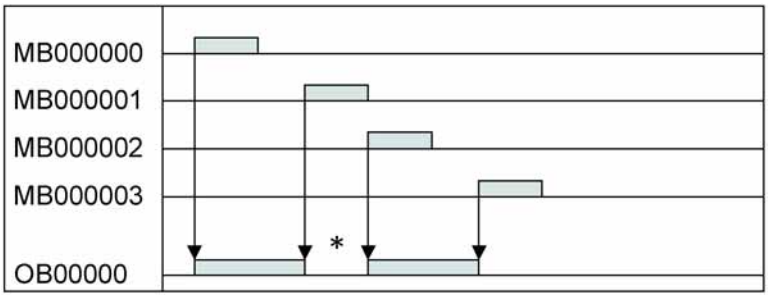
参数名称	设定
线圈编号	• 比特型寄存器（#、C 寄存器除外） • 同上带下标字母

■ 程序举例

多次指定同一输出地点时



上例将进行如下动作。



* OB00000 为 ON 时，在复位线圈指令下 OB00000 为 OFF 状态。

1.2　数值运算指令

1.2.1　存储指令（STORE）

■ 概要

将 Source 存储在 Dest 中。

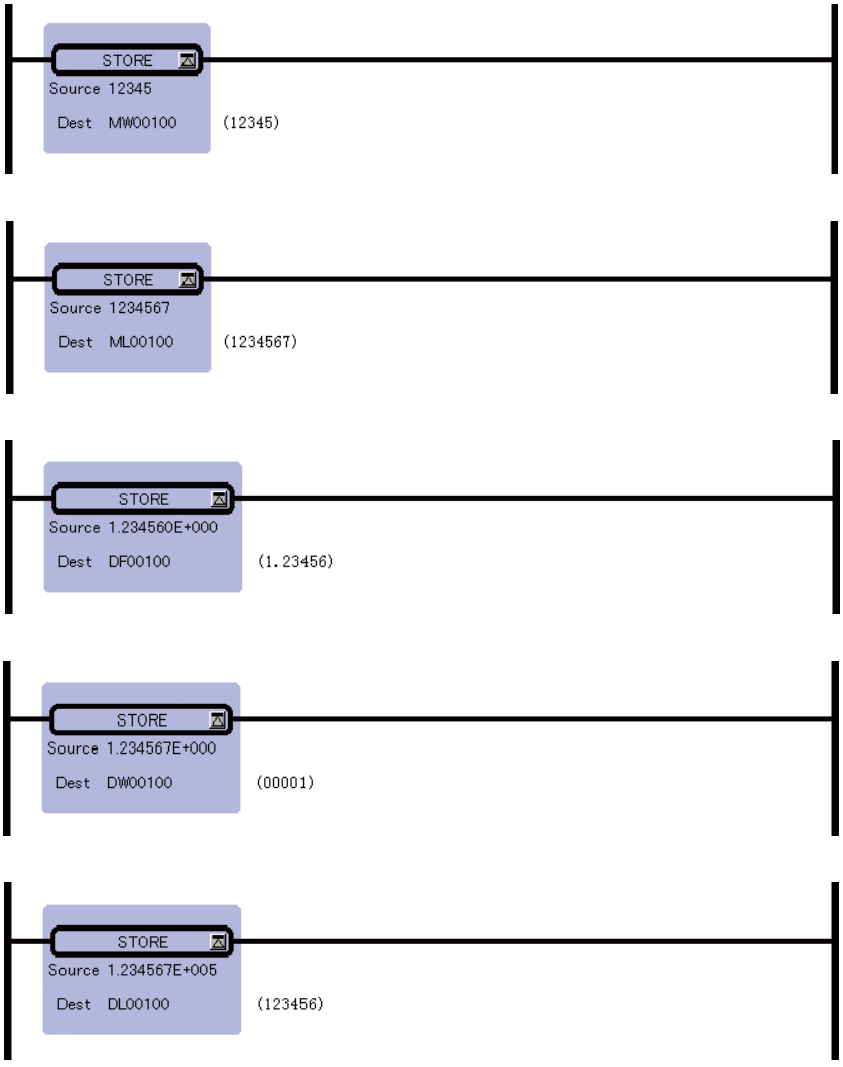
■ 格式

标记：STORE
全称：Store
类别：MATH
图标： 

■ 参数

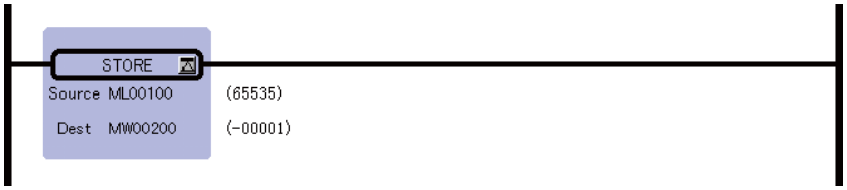
参数名称	设定
Source	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Dest	• 整型、长整型、实型寄存器（#、C 寄存器除外） • 同上带下标字母 • 下标寄存器

■ 程序



补充

将长整型的数据存储到整型寄存器中时，直接保留低16位。存储的数据即使超出整数范围（-32768～32767），也不会发生运算错误，敬请注意。

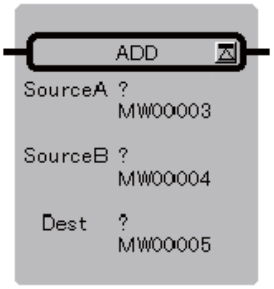


1.2.2　加法指令（ADD）

■ 概要

进行整型、长整型、实型数的加法运算。给 Source A 加上 Source B，将其结果存储在 Dest 中。
整型的运算结果大于 32767 时，发生数据上溢的运算错误。长整型的运算结果大于 2147483647 时，发生数据上溢的运算错误。

■ 格式



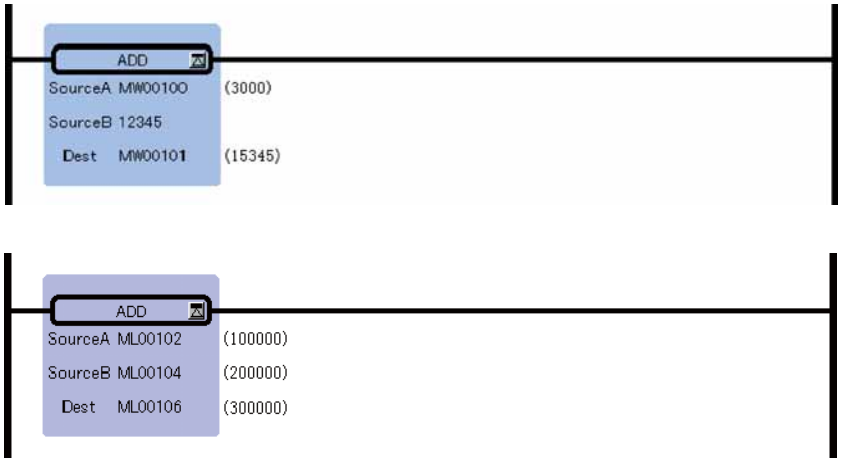
标记：ADD
全称：Add
类别：MATH
图标：

■ 参数

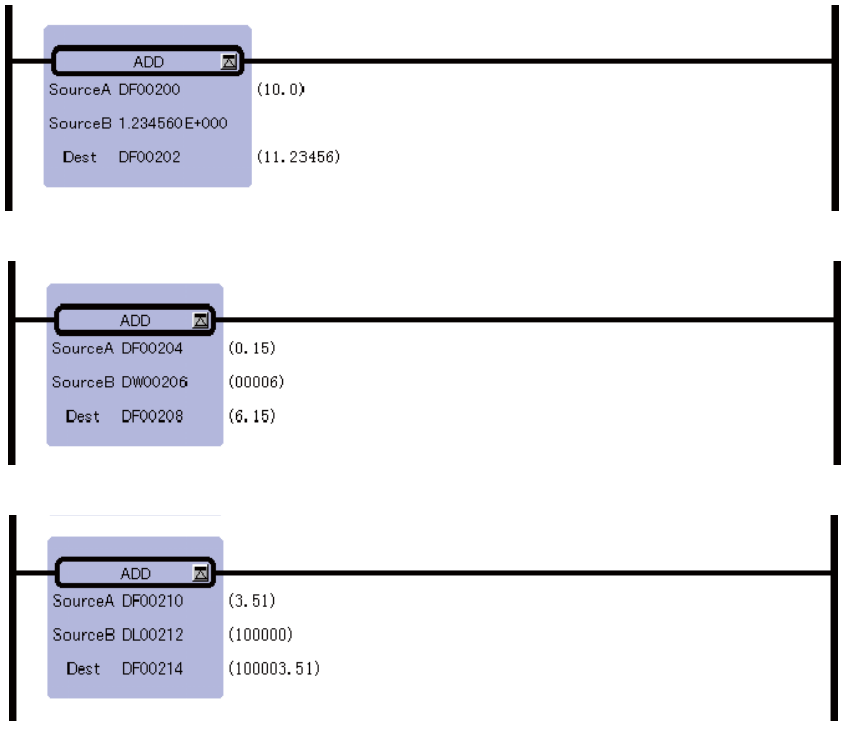
参数名称	设定
Source A	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Source B	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Dest	• 整型、长整型、实型寄存器（#、C 寄存器除外） • 同上带下标字母 • 下标寄存器

■ 程序举例

整型数的加法运算



实型数的加法运算



补充

长整型的加减法指令（＋，－，＋＋，－－）通常为 32 位。但在余数补偿运算表达式（前一指令为乘法指令（×），后一指令为除法指令（÷））中使用时为 64 位运算。

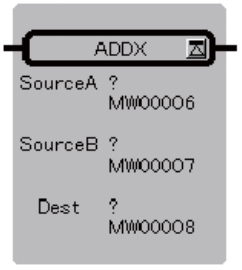
1.2.3　加法扩展指令（ADDX）

■ 概要

进行整型、长整型数的加法运算。给 Source A 加上 Source B，将其结果存储在 Dest 中。

即使运算结果数据上溢，也不会发生运算错误。

■ 格式



标记：ADDX

全称：Extended Add

类别：MATH

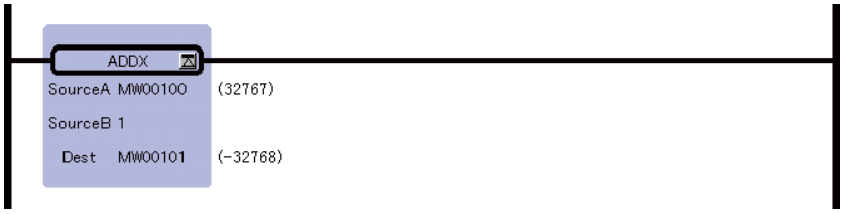
图标： 

■ 参数

参数名称	设定
Source A	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Source B	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Dest	• 整型、长整型、实型寄存器（#、C 寄存器除外） • 同上带下标字母 • 下标寄存器

■ 程序举例

在整型数的加法运算中为避免发生运算错误而使用。



长整型的加减法指令（＋，－，＋＋，－－）通常为 32 位。但在余数补偿运算表达式（前一指令为乘法指令（×），后一指令为除法指令（÷））中使用时为 64 位运算。

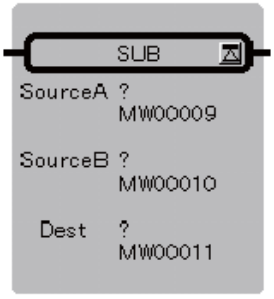
1.2.4 减法指令 (SUB)

■ 概要

进行整型、长整型、实型数的减法运算。从 Source A 减去 Source B，将其结果存储在 Dest 中。

整型的减法运算结果小于 -32768 时，发生数据下溢的运算错误。长整型的减法运算结果小于 -2147483648 时，发生数据下溢的运算错误。

■ 格式



标记：SUB

全称：Subtract

类别：MATH

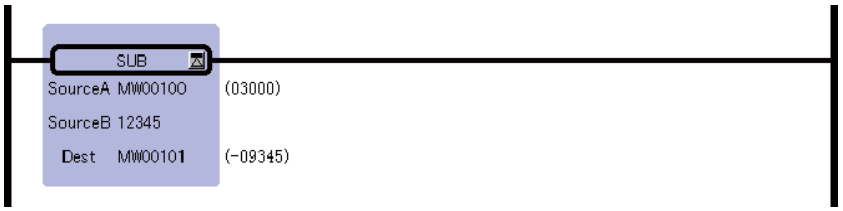
图标：

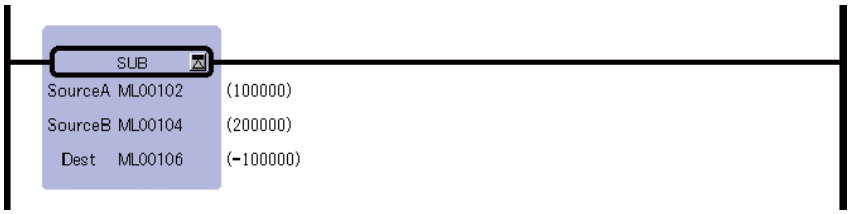
■ 参数

参数名称	设定
Source A	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Source B	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Dest	• 整型、长整型、实型寄存器 (#、C 寄存器除外) • 同上带下标字母 • 下标寄存器

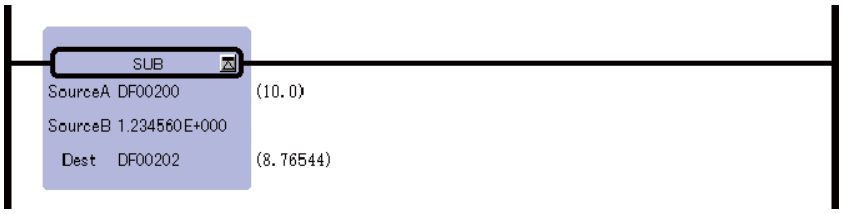
■ 程序举例

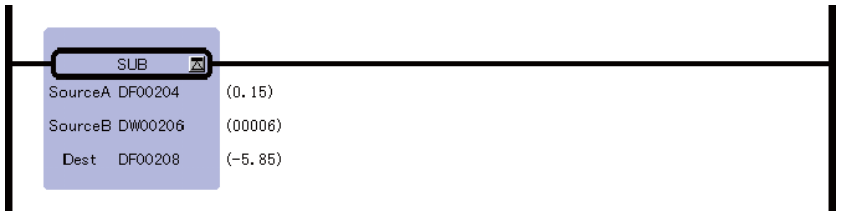
整型数的减法运算

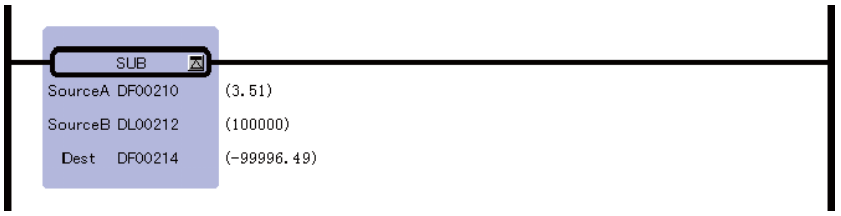




实型数的减法运算







补充

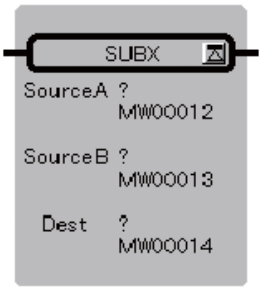
长整型的加减法指令（＋，－，＋＋，－－）通常为 32 位。但在余数补偿运算表达式（前一指令为乘法指令（×），后一指令为除法指令（÷））中使用时为 64 位运算。

1.2.5 减法扩展指令（SUBX）

■ 概要

进行整型、长整型数的减法运算。即使运算结果数据下溢，也不会发生运算错误。

■ 格式



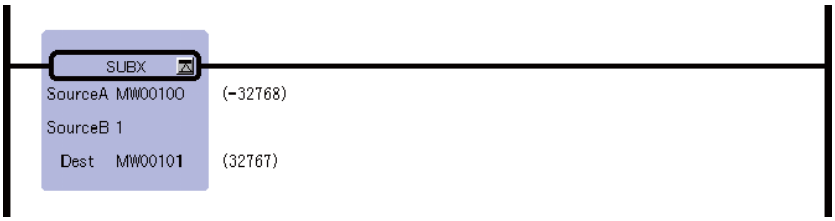
标记：SUB
全称：Subtract
类别：MATH
图标：

■ 参数

参数名称	设定
Source A	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Source B	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Dest	• 整型、长整型、实型寄存器（#、C 寄存器除外） • 同上带下标字母 • 下标寄存器

■ 程序举例

在整型数的减法运算中为避免发生运算错误而使用。



补充

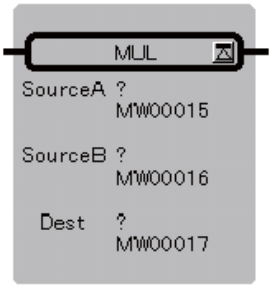
长整型的加减法指令（＋，－，＋＋，－－）通常为 32 位。但在余数补偿运算表达式（前一指令为乘法指令（×），后一指令为除法指令（÷））中使用时为 64 位运算。

1.2.6　乘法指令（MUL）

■ 概要

进行整型、长整型、实型数的乘法运算。Source A 乘以 Source B，将其结果存储在 Dest 中。

■ 格式



标记：MUL

全称：Multiply

类别：MATH

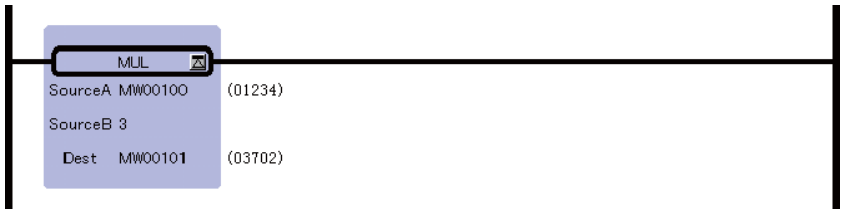
图标： 

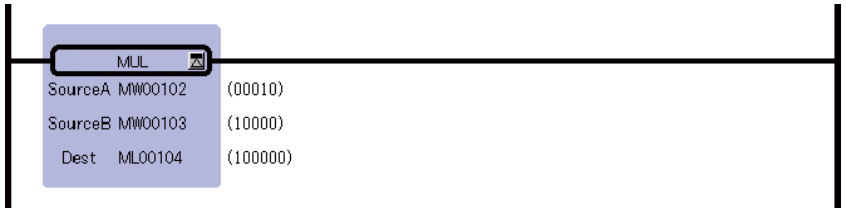
■ 参数

参数名称	设定
Source A	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Source B	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Dest	• 整型、长整型、实型寄存器（#、C 寄存器除外） • 同上带下标字母 • 下标寄存器

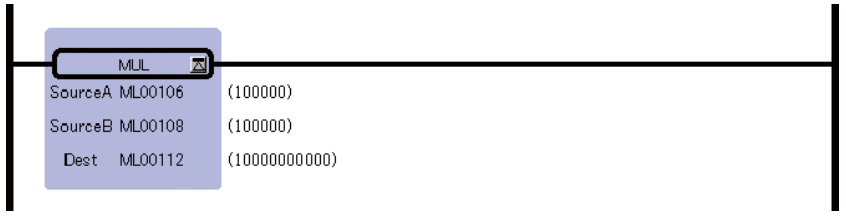
■ 程序举例

整型数的乘法运算

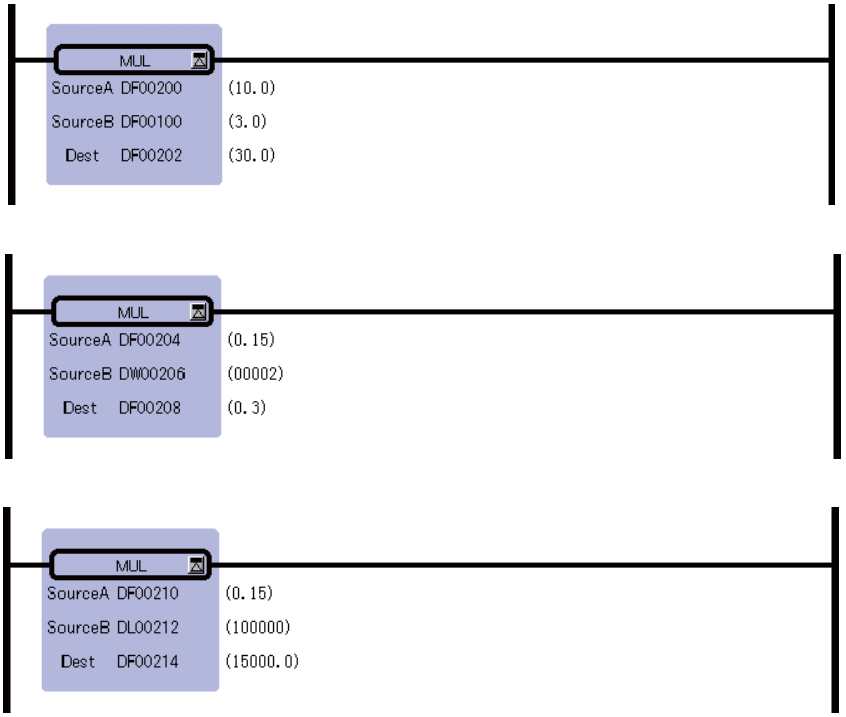




长整型数的乘法运算



实型数的乘法运算



补充

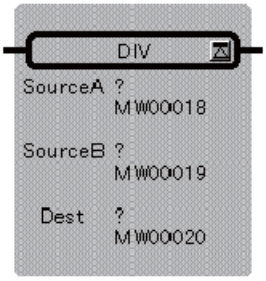
长整型的加减法指令（＋，－，＋＋，－－）通常为 32 位。但在余数补偿运算表达式（前一指令为乘法指令（×），后一指令为除法指令（÷））中使用时为 64 位运算。

1.2.7 除法指令 (DIV)

■ 概要

进行整型、长整型、实型数的除法运算。Source A 除以 Source B，将其结果存储在 Dest 中。

■ 格式



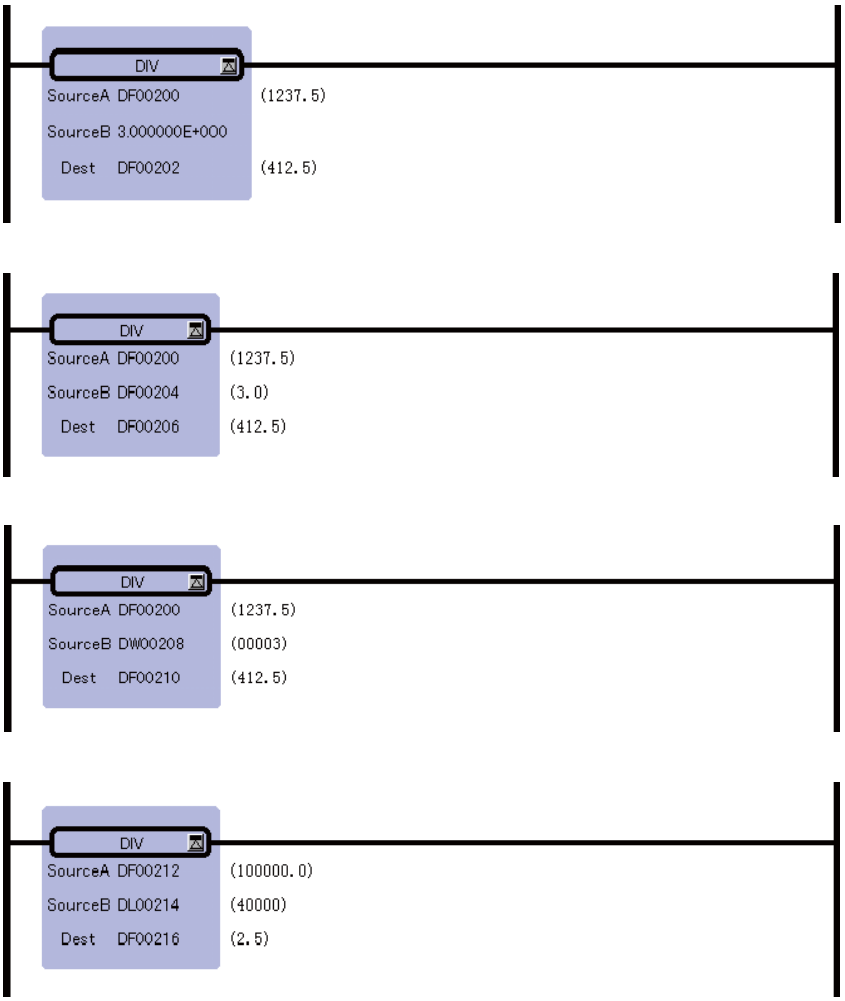
标记：DIV
全称：Divide
类别：MATH
图标：

■ 参数

参数名称	设定
Source A	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Source B	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Dest	• 整型、长整型、实型寄存器 (#、C 寄存器除外) • 同上带下标字母 • 下标寄存器

■ 程序举例

实型数的除法运算



1.2.8 整型余数指令（MOD）

■ 概要

将整型、长整型除法运算的余数存储在 Dest 中。请在除法指令后立即执行。若不在除法指令后立即执行，其后到下次数值运算指令出现之间的运算结果将无法保证。

■ 格式

标记：MOD

全称：Integer Remainder

类别：MATH

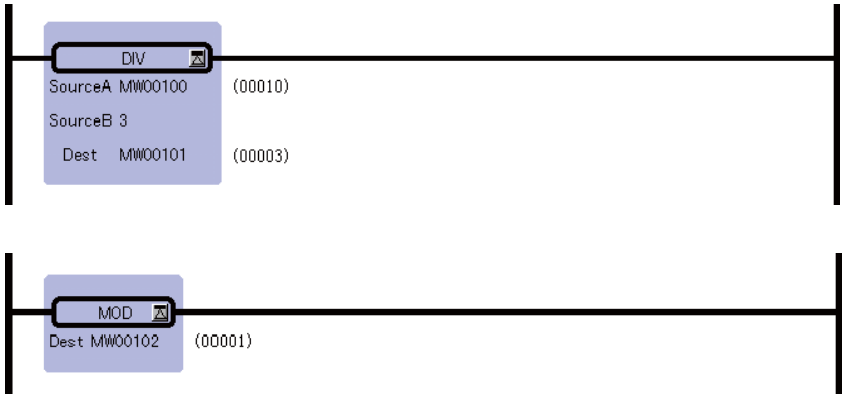
图标： 

■ 参数

参数名称	设定
Dest	• 整型、长整型、实型寄存器（#、C 寄存器除外） • 同上带下标字母 • 下标寄存器

■ 程序举例

将整型除法运算的商存储在 MW00101 中，余数存储在 MW00102 中。



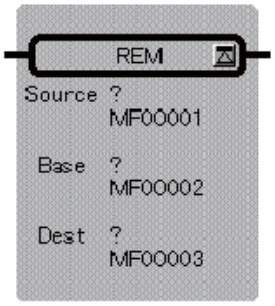
1.2.9　实型余数指令（REM）

■ 概要

将实型除法运算的余数存储在 Dest 中。此时所谓的余数，是指从 Source 连续减去 Base 所得的剩余。即，假设连续减去的次数为 n，则得到如下结果。

Dest ＝ Source － (Base × n)(0 ≦ Dest ＜ Base)

■ 格式



标记：REM
全称：Real Remainder
类别：MATH
图标：

■ 参数

参数名称	设定
Source	• 所有实型寄存器 • 同上带下标字母 • 常数
Base	• 所有实型寄存器 • 同上带下标字母 • 常数
Dest	• 实型寄存器（#、C 寄存器除外） • 同上带下标字母

■ 程序举例

求出 MF00200 除以常数 1.5 的余数，将其结果存储在 DF00202 中。

1.2.10 增量指令（INC）

■ 概要

给整型、长整型寄存器加 1。整型时，加法运算结果即使大于 32767，也不会发生数据上溢的运算错误。同样，长整型也不会发生数据上溢的运算错误。

■ 格式

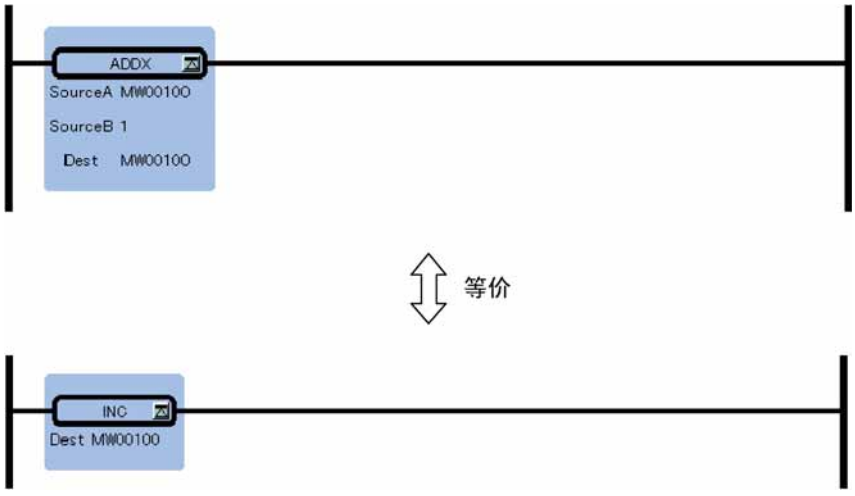
标记：INC
全称：Increment
类别：MATH
图标： 

■ 参数

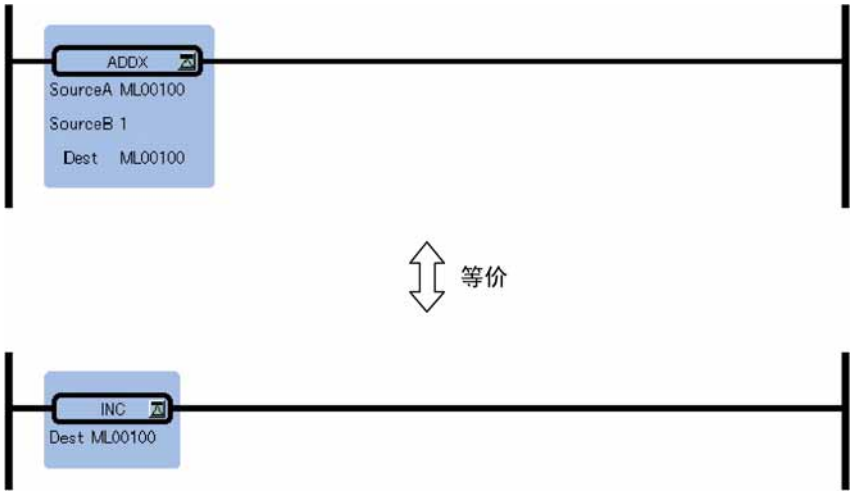
参数名称	设定
Dest	• 整型、长整型寄存器（#、C 寄存器除外） • 同上带下标字母 • 下标寄存器

■ 程序举例

整型



长整型



1.2.11　减量指令 (DEC)

■ 概要

从整型、长整型寄存器中减 1。整型时，减法运算结果即使小于 -32768，也不会发生数据下溢的运算错误。同样，长整型也不会发生数据下溢的运算错误。

■ 格式

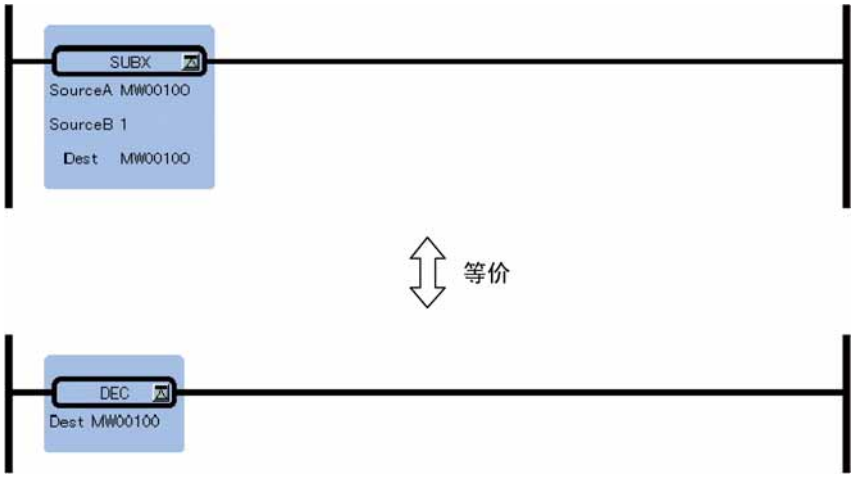
标记：DEC
全称：Decrement
类别：MATH
图标： 

■ 参数

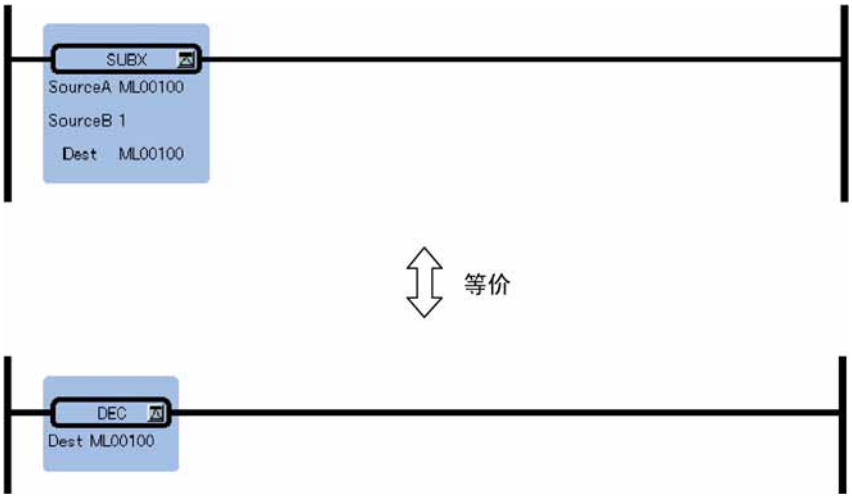
参数名称	设定
Dest	• 整型、长整型寄存器 (#、C 寄存器除外) • 同上带下标字母 • 下标寄存器

■ 程序举例

整型



长整型



1.2.12 时间加法指令（TMADD）

■ 概要

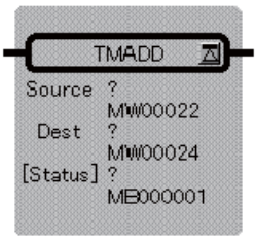
进行两个时间数据（时 / 分 / 秒）的加法运算。
给 Dest 加上 Source（加时间），将其结果存储在 Dest 中。Source 和 Dest 的数据格式如下。

表 1.3 数据格式

寄存器偏置	数据内容	数据范围（BCD）
0	时 / 分	高位字节（时）：0 ～ 23 低位字节（分）：0 ～ 59
1	秒	0000 ～ 0059

Source、Dest 及运算结果的内容在上述数据范围内时，正常执行运算，执行后的 Status 为 OFF。Source 及 Dest 超出数据范围时，不执行运算，Dest 的秒寄存器内存储 9999(H)，Status 为 ON。运算结果超出数据范围时，其值被直接存储在 Dest 中，Status 为 ON。

■ 格式



标记：TMADD
全称：Time Add
类别：MATH
图标：

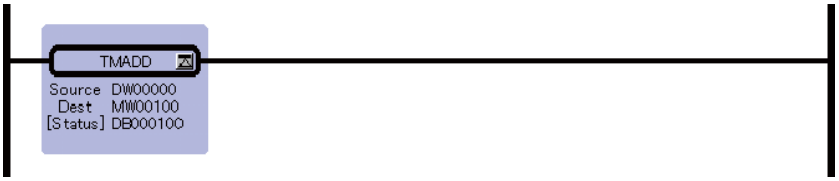
■ 参数

参数名称	设定
Source	• 所有整型寄存器 • 同上带下标字母
Dest	• 整型寄存器（#、C 寄存器除外） • 同上带下标字母
[Status] *	• 比特型寄存器（#、C 寄存器除外） • 同上带下标字母

*可省略 [Status]。

■ 程序举例

在 MW00100 ～ MW00101 的时间数据上加 DW00000 ～ DW00001 的时间数据。



8 时 40 分　32 秒　＋　1 时 22 分　16 秒　＝ 10 时 2 分　48 秒
(MW00100)　(MW00101)　(DW00000)　(DW00001)　(MW00100)　(MW00101)

时间数据	执行前	执行后
MW00100	0840H	1002H
MW00101	0032H	0048H
DW00000	0122H	0122H
DW00001	0016H	0016H

1.2.13 时间减法指令（TMSUB）

■ 概要

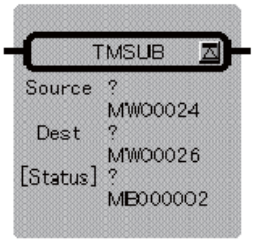
进行两个时间数据（时 / 分 / 秒）的减法运算。
从 Dest（被减时间）中减去 Source（减时间），将其结果存储在 Dest 中。Source 和 Dest 的数据格式如下。

表 1.4 数据格式

寄存器偏置	数据内容	数据范围（BCD）
0	时 / 分	高位字节（时）：0 ～ 23 低位字节（分）：0 ～ 59
1	秒	0000 ～ 0059

Source、Dest 及运算结果的内容在上述数据范围内时，正常执行运算，执行后的 Status 为 OFF。Source 及 Dest 超出数据范围时，不执行运算，Dest 的秒寄存器内存储 9999(H)，Status 为 ON。运算结果超出数据范围时，其值被直接存储在 Dest 中，Status 为 ON。

■ 格式



标记：TMSUB
全称：Time Sub
类别：MATH
图标：

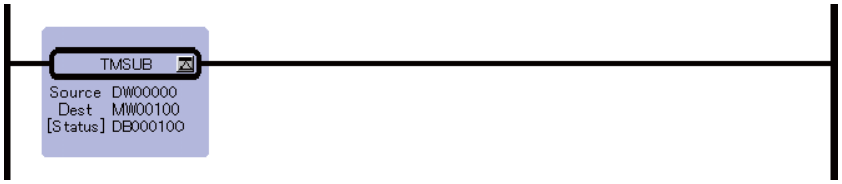
■ 参数

参数名称	设定
Source	• 所有整型寄存器 • 同上带下标字母
Dest	• 整型寄存器（#、C 寄存器除外） • 同上带下标字母
[Status] *	• 比特型寄存器（#、C 寄存器除外） • 同上带下标字母

*可省略 [Status]。

■ 程序举例

从 MW00100 ～ MW00101 的时间数据中减 DW00000 ～ DW00001 的时间数据。



8 时 40 分　32 秒　－　1 时 22 分　16 秒　＝　7 时 18 分　16 秒
（MW00100）（MW00101）　（DW00000）（DW00001）　（MW00100）（MW00101）

时间数据	执行前	执行后
MW00100	0840H	0718H
MW00101	0032H	0016H
DW00000	0122H	0122H
DW00001	0016H	0016H

1.2.14 时间经过指令（SPEND）

■ 概要

进行两个数据（年 / 月 / 日 / 时 / 分 / 秒）的减法运算，计算经过的时间。
从 Dest 中减去 Source，将其结果存储在 Dest 中。Source 和 Dest 的格式如下。

表 1.5 Source 的格式

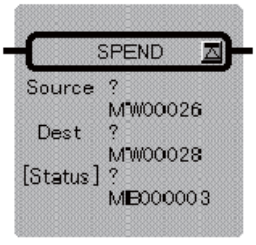
寄存器偏置	数据内容	数据范围（BCD）	输入输出
0	年（BCD）	0000 ～ 0099	IN
1	月 / 日（BCD）	高位字节（月）：1 ～ 12 低位字节（日）：1 ～ 31	IN
2	时 / 分（BCD）	高位字节（时）：0 ～ 23 低位字节（分）：0 ～ 59	IN
3	秒（BCD）	0000 ～ 0059	IN

表 1.6 Dest 的格式

寄存器偏置	数据内容	数据范围（BCD）	输入输出
0	年（BCD）	0000 ～ 0099	IN/OUT
1	月 / 日（BCD）	高位字节（月）：1 ～ 12 低位字节（日）：1 ～ 31	IN/OUT
2	时 / 分（BCD）	高位字节（时）：0 ～ 23 低位字节（分）：0 ～ 59	IN/OUT
3	秒（BCD）	0000 ～ 0059	IN/OUT
4	总计秒数	将运算结果的年 / 月 / 日 / 时 / 分换算成秒数的结果（长整数）	IN/OUT
5			

Source、Dest 及运算结果的内容在上述数据范围内时，正常执行运算，执行后的 Status 为 OFF。Source 及 Dest 超出数据范围时，不执行运算，Dest 的秒寄存器内存储 9999(H)，Status 为 ON。运算结果超出数据范围时，其值被直接存储在 Dest 中，Status 为 ON。

■ 格式



标记：SPEND
全称：Time Spend
类别：MATH
图标：

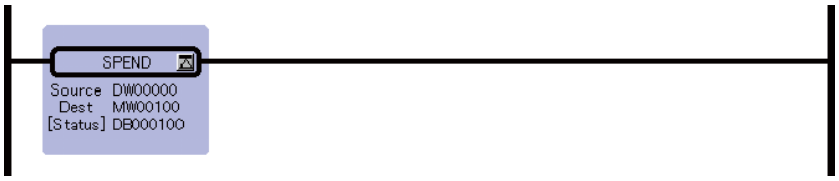
■ 参数

参数名称	设定
Source	• 所有整型寄存器 • 同上带下标字母
Dest	• 整型寄存器（#、C 寄存器除外） • 同上带下标字母
[Status]*	• 比特型寄存器（#、C 寄存器除外） • 同上带下标字母

*可省略 [Status] 的设定。

■ 程序举例

将 MW00100 ～ MW00103 的时间数据与 DW00000 ～ DW00003 的时间数据之间经过的时间存储在 MW00100 ～ MW00105 中。



98 年　　5 月 11 日　15 时 04 分　47 秒
(MW00100)　(MW00101)　(MW00102)　(MW00103)

－ 98 年　　4 月 2 日　8 时 13 分　08 秒
(DW00000)　(DW00101)　(DW00102)　(DW00103)

＝　0 年　　39 日　6 时 51 分　39 秒
(MW00100)　(MW00101)　(MW00102)　(MW00103)

时间数据	执行前	执行后
MW00100	H0098	H0000
MW00101	H0511	H0039
MW00102	H1504	H0651
MW00103	H0047	H0039
MW00104	—	3394299(10进制数)
MW00105	—	
DW00000	H0098	H0098
DW00001	H0402	H0402
DW00002	H0813	H0813
DW00003	H0008	H0008

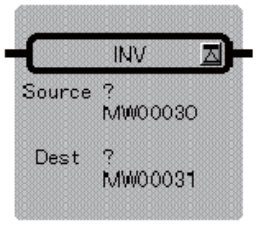
补充　运算结果中，年按 365 天计算，不考虑闰年。而且运算结果中不计算月数，只计算天数。

1.2.15 符号取反指令（INV）

■ 概要

将 Source 的符号反转，将其结果存储在 Dest 中。

■ 格式



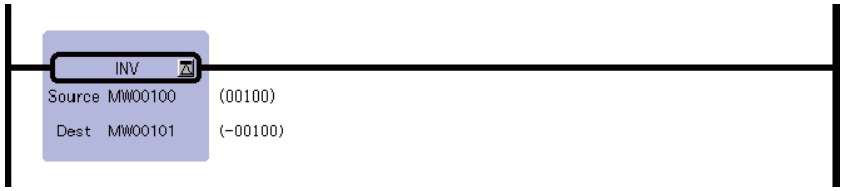
标记：INV
全称：Inverse
类别：MATH
图标：

■ 参数

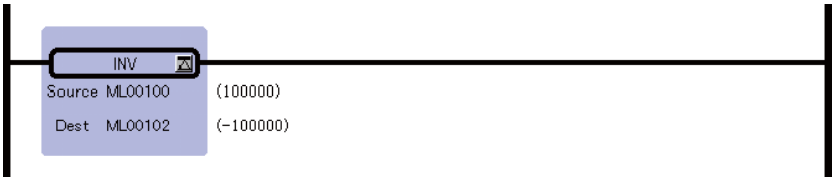
参数名称	设定
Source	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器
Dest	• 整型、长整型、实型寄存器（#、C 寄存器除外） • 同上带下标字母 • 下标寄存器

■ 程序举例

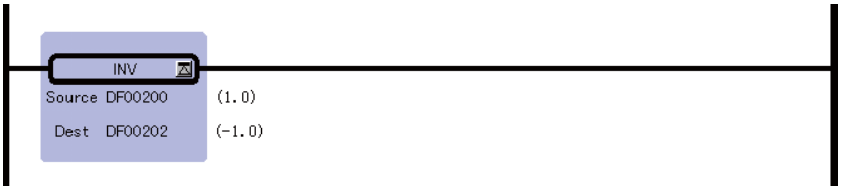
整型数据



长整型数据



实型数据

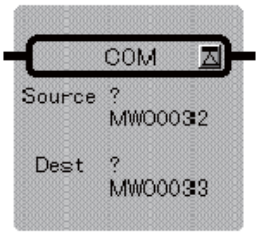


1.2.16 1 的补码指令（COM）

■ 概要

将 Source 的 1 的补码存储在 Dest 中。

■ 格式



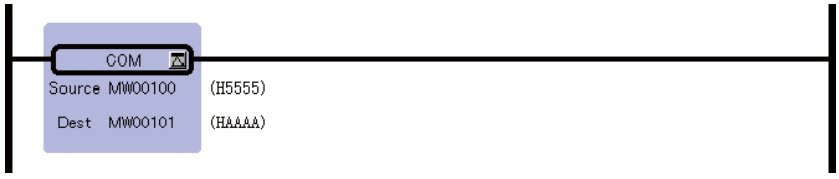
标记：COM
全称：Complement
类别：MATH
图标：

■ 参数

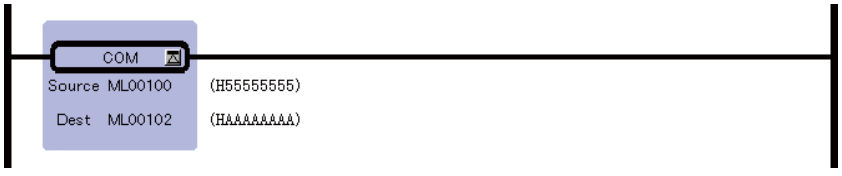
参数名称	设定
Source	• 所有整型、长整型寄存器 • 同上带下标字母 • 下标寄存器
Dest	• 整型、长整型寄存器（#、C 寄存器除外） • 同上带下标字母 • 下标寄存器

■ 程序举例

整型数据



长整型数据

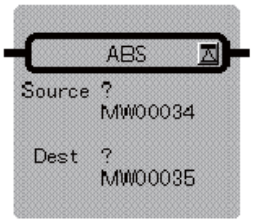


1.2.17　绝对值转换指令（ABS）

■ 概要

将 Source 的绝对值存储在 Dest 中。

■ 格式



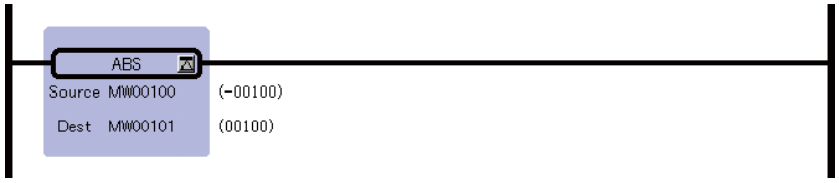
标记：ABS
全称：Absolute
类别：MATH
图标：

■ 参数

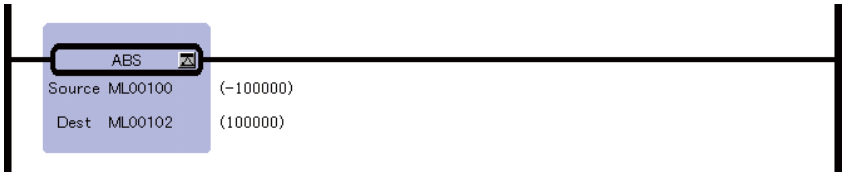
参数名称	设定
Source	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器
Dest	• 整型、长整型、实型寄存器（#、C 寄存器除外） • 同上带下标字母 • 下标寄存器

■ 程序举例

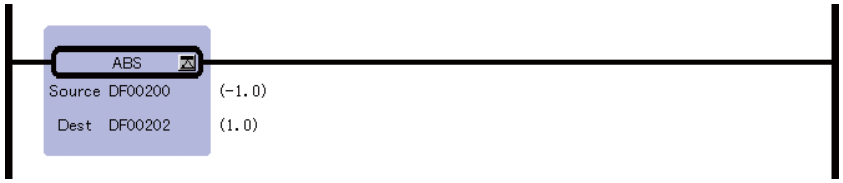
整型数据



长整型数据



实型数据



1.2.18　2进制转换指令（BIN）

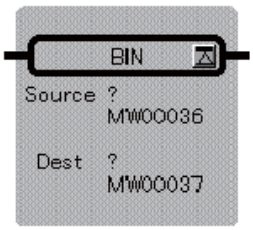
■ 概要

将Source的BCD格式的数值转换为2进制数值（BIN转换），将其结果存储在Dest中。
假设BCD格式的数值（4位）为abcd，用下列算式求出BIN指令的输出值Dest。

$$Dest = (a \times 1000) + (b \times 100) + (c \times 10) + d$$

当Source的数值不是BCD格式（123FH等）时，无法得出正确结果。

■ 格式



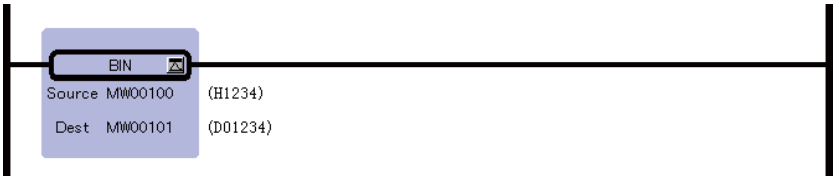
标记：BIN
全称：Convert to Binary
类别：MATH
图标：

■ 参数

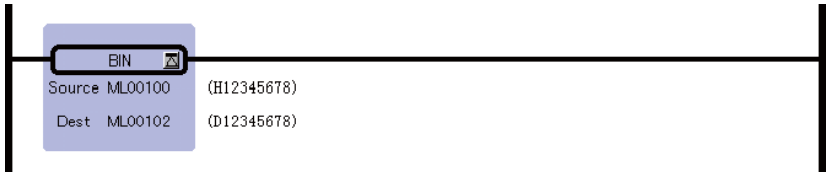
参数名称	设定
Source	• 所有整型、长整型寄存器 • 同上带下标字母 • 下标寄存器
Dest	• 整型、长整型寄存器（#、C寄存器除外） • 同上带下标字母 • 下标寄存器

■ 程序举例

整型数据



长整型数据



1.2.19 BCD 转换指令 (BCD)

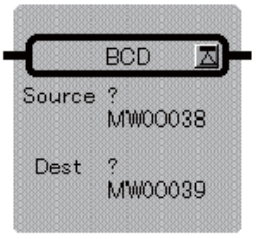
■ 概要

将 Source 的 2 进制格式的数值转换为 BCD 数值 (BCD 转换)，将其结果存储在 Dest 中。
假设 Source 的 10 进制数值 (4 位) 为 0abcd，用下列算式求出 BCD 指令的输出值 Dest。

$$Dest = (a \times 4096) + (b \times 256) + (c \times 16) + d$$

Source 的数值不是 BCD 格式 (大于 9999 的数及负数) 时，无法得出正确的结果。

■ 格式



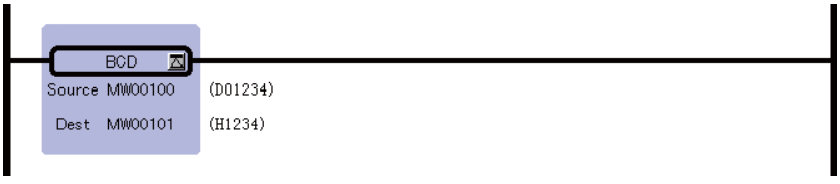
标记：BCD
全称：Convert to BCD
类别：MATH
图标：

■ 参数

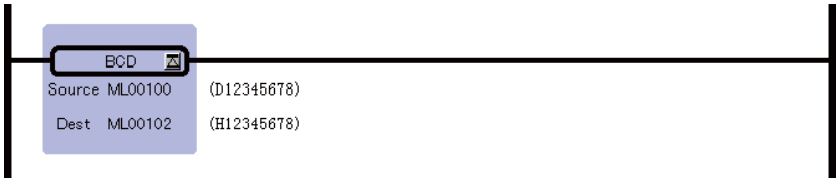
参数名称	设定
Source	• 所有整型、长整型寄存器 • 同上带下标字母 • 下标寄存器
Dest	• 整型、长整型寄存器 (#、C 寄存器除外) • 同上带下标字母 • 下标寄存器

■ 程序举例

整型数据



长整型数据



1.2.20　校验转换指令（PARITY）

■ 概要

计算 Source 的 2 进制格式位为 ON（＝1）的数，将其结果存储在 Dest 中。

■ 格式

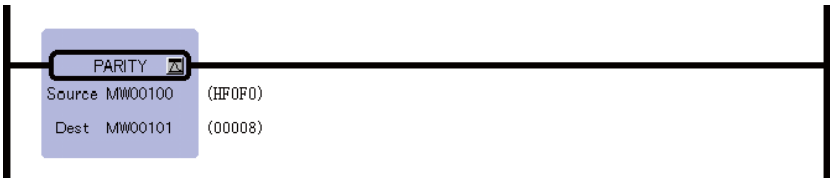
标记：PARITY
全称：Count ON Bit
类别：MATH
图标：

■ 参数

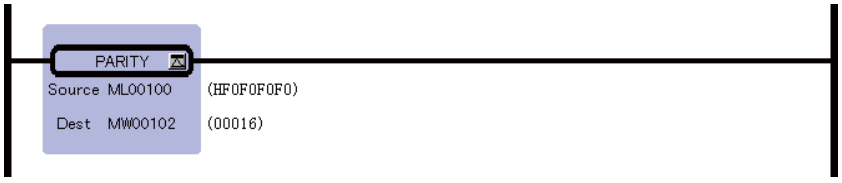
参数名称	设定
Source	• 所有整型、长整型寄存器 • 同上带下标字母 • 下标寄存器
Dest	• 整型、长整型寄存器（#、C 寄存器除外） • 同上带下标字母 • 下标寄存器

■ 程序举例

整型数据



长整型数据



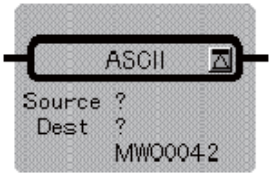
1.2.21 ASCII 码转换 1 指令（ASCII）

■ 概要

将 Source 中指定的字符（字符串）转换成 ASCII 码，将其结果存储在 Dest（整型寄存器）中。可区分大小写。

按照第 1 个字符：第 1 个字的低位字节、第 2 个字符：第 1 个字的高位字节这样的顺序存储。字符串的长度为奇数时，存储地点的最后的字的高位字节为 0。最多可以输入 32 个字符。

■ 格式



标记：ASCII

全称：Convert Character to ASCII

类别：MATH

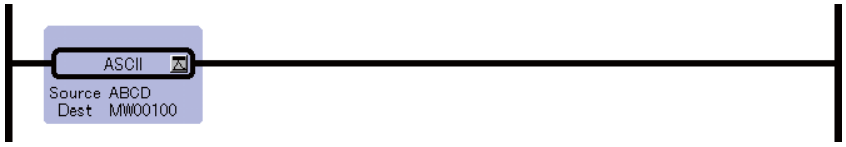
图标： 

■ 参数

参数名称	设定
Source	• ASCII 字符
Dest	• 整型寄存器（#、C 寄存器除外） • 同上带下标字母

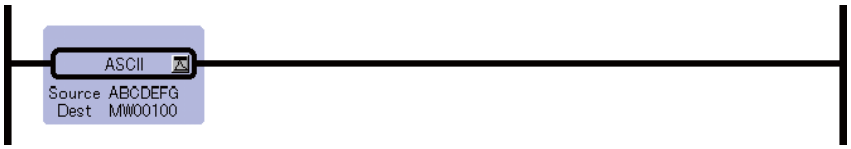
■ 程序举例

将字符串 “ABCD”存储在 MW00100 ～ MW00101 中。



	高位字节	低位字节	
MW00100	42H（‘B’）	41H（‘A’）	MW00100 = 4241H
MW00101	44H（‘D’）	43H（‘C’）	MW00101 = 4443H

将字符串 “ABCDEFG”存储在 MW00100 ～ MW00103 中。



	高位字节	低位字节	
MW00100	42H（‘B’）	41H（‘A’）	MW00100 ＝ 4241H
MW00101	44H（‘D’）	43H（‘C’）	MW00101 ＝ 4443H
MW00102	46H（‘F’）	45H（‘E’）	MW00100 ＝ 4645H
MW00103	00H	47H（‘G’）	MW00101 ＝ 0047H

↑空字节补 0。

1.2.22 ASCII 码转换 2 指令 (BINASC)

■ 概要

将 Source 中存储的 16 位 2 进制数据转换为 16 进制 4 位的 ASCII 码，将其结果存储在 Dest(2 字) 中。

■ 格式

标记：BINASC

全称：Convert Binary to ASCII

类别：MATH

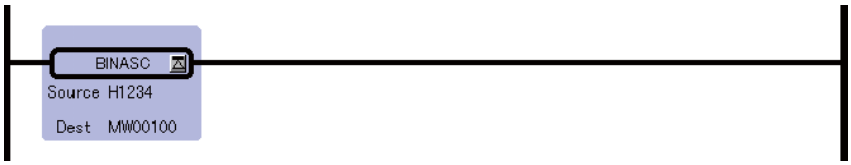
图标：

■ 参数

参数名称	设定
Source	• 所有整型寄存器 • 同上带下标字母 • 常数
Dest	• 整型寄存器 (#、C 寄存器除外) • 同上带下标字母

■ 程序举例

将 2 进制数据 1234H 转换成 ASCII 码，将其结果存储在 MW00100 ～ MW00101 中。



	高位字节	低位字节	
MW00100	32H(‘2’)	31H(‘1’)	MW00100 = 3231H
MW00101	34H(‘4’)	33H(‘3’)	MW00101 = 3433H

1.2.23　ASCII 码转换 3 指令（ASCBIN）

■ 概要

将 Source(2 字）中存储的 16 进制 4 位 ASCII 码表示的数据转换为 16 位的 2 进制数据，将其结果存储在 Dest(1 字）中。

■ 格式

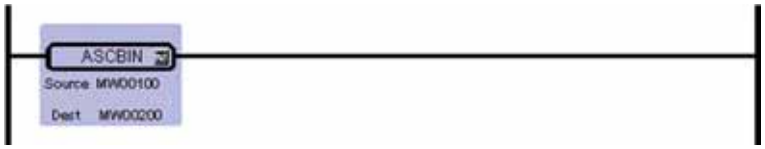
标记：ASCBIN
全称：Convert ASCII to Binary
类别：MATH
图标：

■ 参数

参数名称	设定
Source	• 所有整型寄存器 • 同上带下标字母
Dest	• 整型寄存器（#、C 寄存器除外） • 同上带下标字母

■ 程序举例

将 MW00100 ～ MW00101 中存储的 4 字节 ASCII 码转换为 2 进制 2 字节数据，将其结果存储在 MW00200 中。



	转换源数据			存储数据	
	高位字节	低位字节		高位数据	低位数据
MW00100	32H（‘2’）	31H（‘1’）	⇨ MW00200	12H	34H
MW00101	34H（‘4’）	33H（‘3’）			

1.3 逻辑运算 / 比较指令

1.3.1 逻辑与指令 (AND)

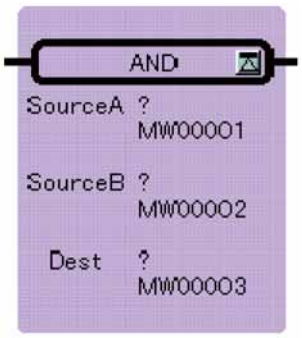
■ 概要

将 Source A 和 Source B 的逻辑与存储在 Dest 中。

表 1.7 位逻辑与真值表

Source A	Source B	Dest
0	0	0
0	1	0
1	0	0
1	1	1

■ 格式



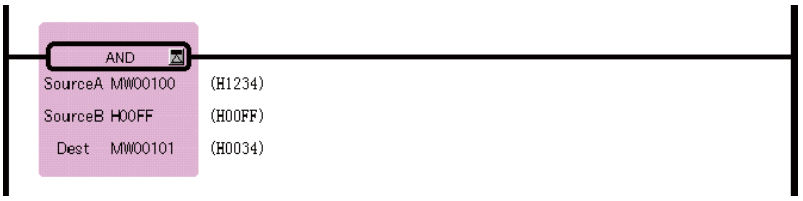
标记：AND
全称：AND
类别：LOGIC
图标：

■ 参数

参数名称	设定
Source A	• 所有整型、长整型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Source B	• 所有整型、长整型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Dest	• 整型、长整型寄存器 (#、C 寄存器除外) • 同上带下标字母 • 下标寄存器

■ 程序举例

将 MW00100 和常数的逻辑与存储在 MW00101 中。



1.3.2　逻辑或指令 (OR)

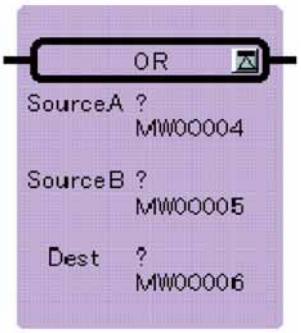
■ 概要

将 Source A 和 Source B 的逻辑或存储在 Dest 中。

表 1.8　位逻辑或真值表

Source A	Source B	Dest
0	0	0
0	1	1
1	0	1
1	1	1

■ 格式



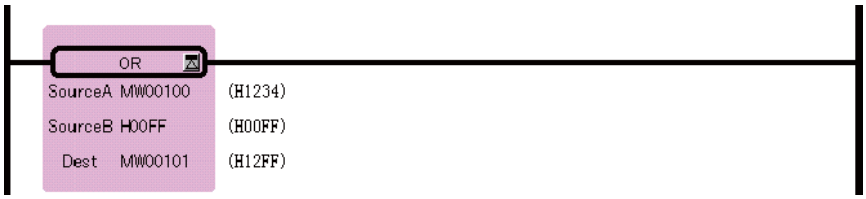
标记：OR
全称：Inclusive OR
类别：LOGIC
图标：

■ 参数

参数名称	设定
Source A	• 所有整型、长整型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Source B	• 所有整型、长整型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Dest	• 整型、长整型寄存器 (#、C 寄存器除外) • 同上带下标字母 • 下标寄存器

■ 程序举例

将 MW00100 和常数的逻辑或存储在 MW00101 中。



1.3.3 逻辑异或指令（XOR）

■ 概要

将 Source A 和 Source B 的逻辑异或存储在 Dest 中。

表 1.9 位逻辑异或真值表

Source A	Source B	Dest
0	0	0
0	1	1
1	0	1
1	1	0

■ 格式

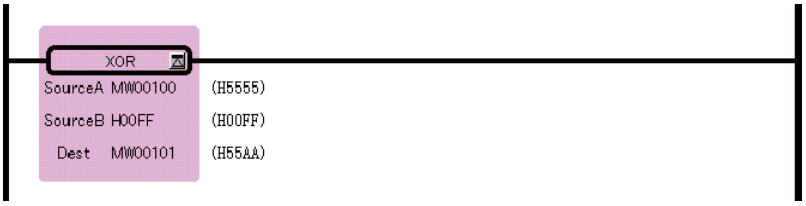
标记：XOR
全称：Exclusive OR
类别：LOGIC
图标：

■ 参数

参数名称	设定
Source A	• 所有整型、长整型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Source B	• 所有整型、长整型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Dest	• 整型、长整型寄存器（#、C 寄存器除外） • 同上带下标字母 • 下标寄存器

■ 程序举例

将 MW00100 和常数的逻辑异或存储在 MW00101 中。

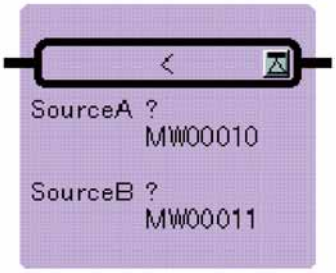


1.3.4　比较指令（＜）

■ 概要

进行 Source A 和 Source B 比较结果的位输出。（结果为真时为 ON。）

■ 格式



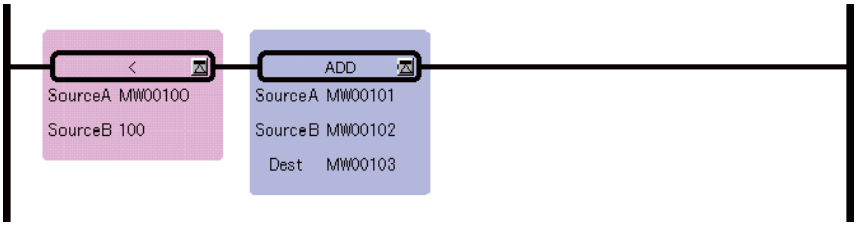
标记：＜
全称：Less Than(A＜B)
类别：LOGIC
图标：

■ 参数

参数名称	设定
Source A	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Source B	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数

■ 程序举例

MW00100 的值小于 100 时，执行其后的运算指令。

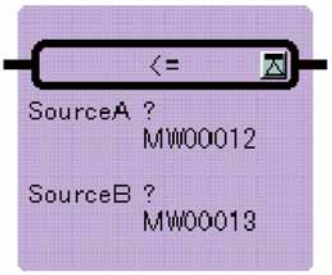


1.3.5 比较指令（≦）

■ 概要

进行 Source A 和 Source B 比较结果的位输出。（结果为真时为 ON。）

■ 格式



标记：≦

全称：Less Than or Equal(A ≦ B)

类别：LOGIC

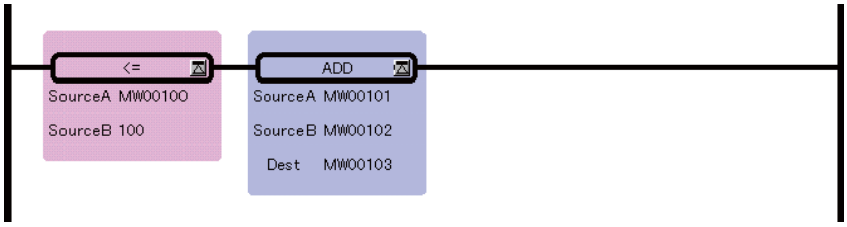
图标： 

■ 参数

参数名称	设定
Source A	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Source B	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数

■ 程序举例

MW00100 的值小于等于 100 时，执行其后的运算指令。

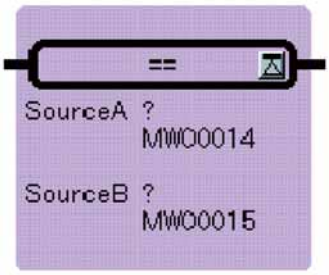


1.3.6　比较指令（＝）

■ 概要

进行 Source A 和 Source B 比较结果的位输出。（结果为真时为 ON。）

■ 格式



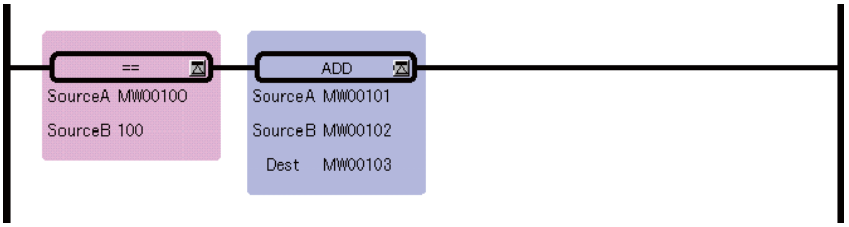
标记：＝
全称：Equal(A＝B)
类别：LOGIC
图标：

■ 参数

参数名称	设定
Source A	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Source B	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数

■ 程序举例

MW00100 的值等于 100 时，执行其后的运算指令。

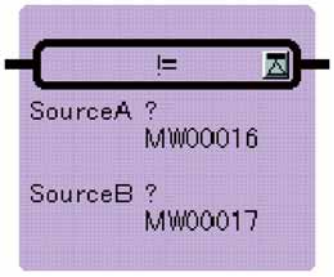


1.3.7 比较指令（≠）

■ 概要

进行 Source A 和 Source B 比较结果的位输出。（结果为真时为 ON。）

■ 格式



标记：≠

全称：Not Equal(A ≠ B)

类别：LOGIC

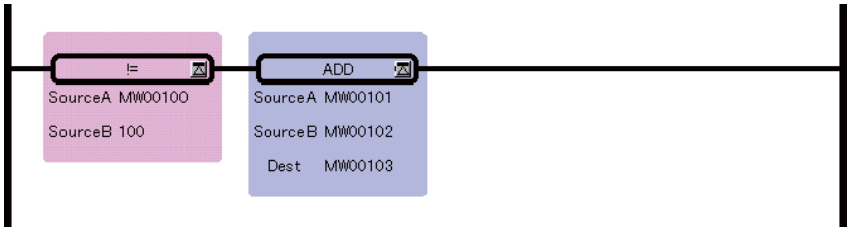
图标：

■ 参数

参数名称	设定
Source A	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Source B	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数

■ 程序举例

MW00100 的值不等于 100 时，执行其后的运算指令。

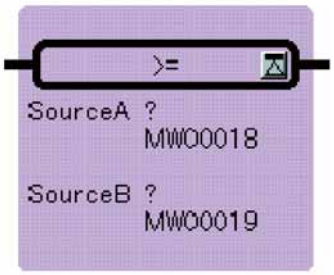


1.3.8　比较指令（≧）

■ 概要

进行 Source A 和 Source B 比较结果的位输出。（结果为真时为 ON。）

■ 格式



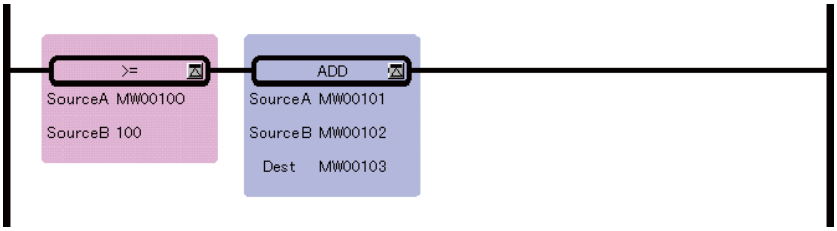
标记：≧
全称：Greater Than or Equal(A ≧ B)
类别：LOGIC
图标：

■ 参数

参数名称	设定
Source A	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Source B	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数

■ 程序举例

MW00100 的值大于等于 100 时，执行其后的运算指令。

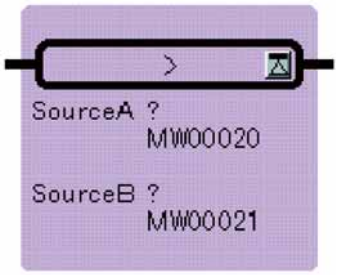


1.3.9 比较指令（＞）

■ 概要

进行 Source A 和 Source B 比较结果的位输出。（结果为真时为 ON。）

■ 格式



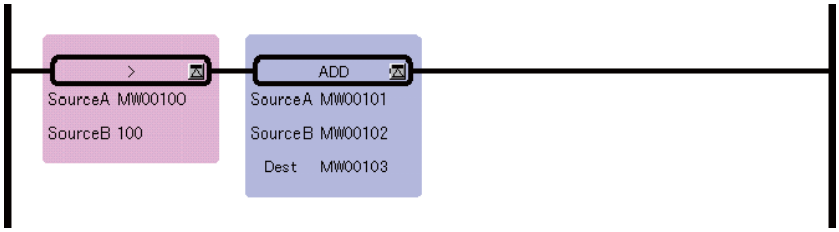
标记：＞
全称：Greater Than(A＞B)
类别：LOGIC
图标：

■ 参数

参数名称	设定
Source A	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Source B	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数

■ 程序举例

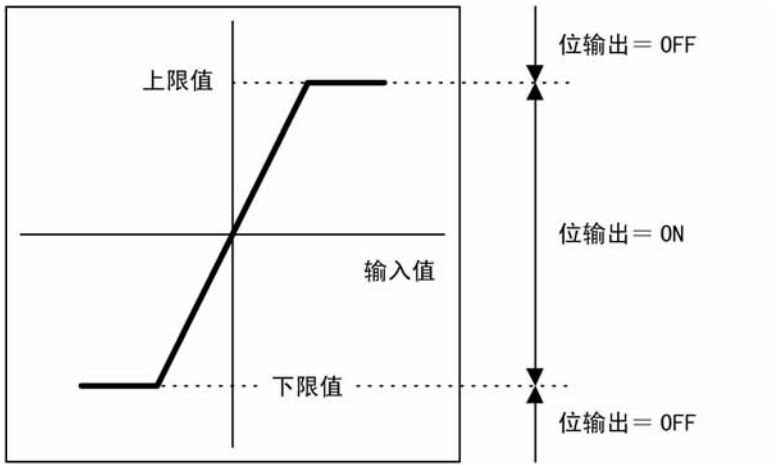
MW00100 的值大于 100 时，执行其后的运算指令。



1.3.10　范围检查指令（RCHK）

■ 概要

检查输入的 Input 内容是否在 Lower Limit 和 Upper Limit 的范围内，将其结果进行位输出。



- 下限值（Lower Limit）≦输入值（Input）≦上限值（Upper Limit）时，结果位输出＝ON
- 除上述以外时，结果位输出＝OFF

■ 格式

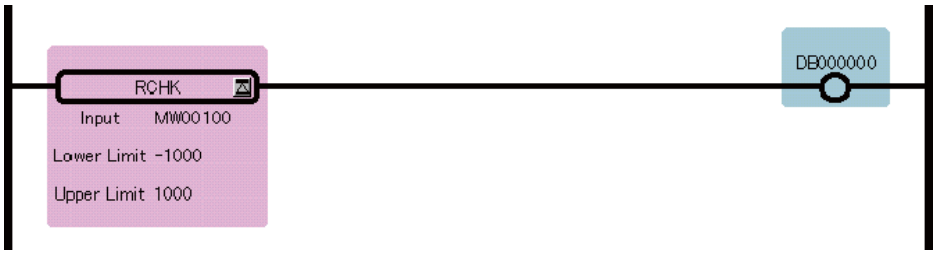
标记：RCHK
全称：Range Check
类别：LOGIC
图标：

■ 参数

参数名称	设定
Input （输入值）	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Lower Limit （下限值）	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Upper Limit （上限值）	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数

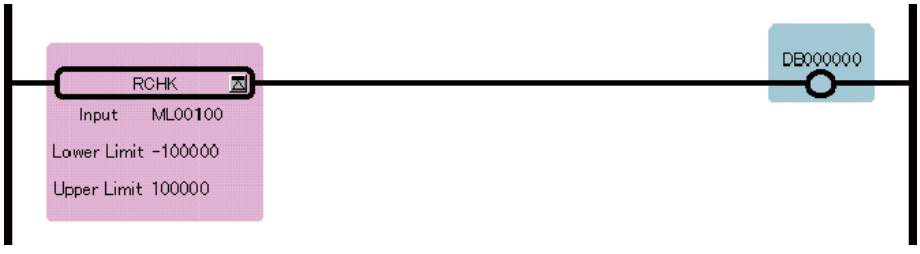
■ 程序举例

整型运算时



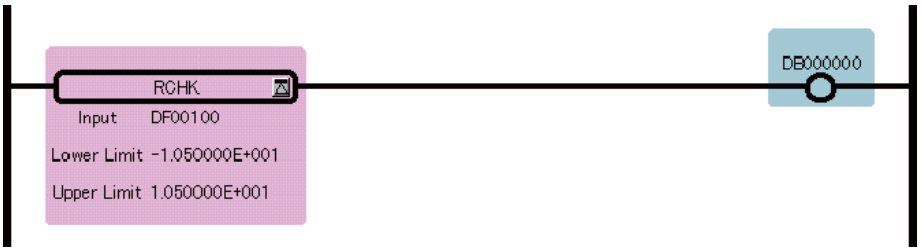
输入（MW00100）	输出（DB000000）
-1000 ＞ MW00100	OFF
-1000 ≦ MW00100 ≦ 1000	ON
MW00100 ＞ 1000	OFF

长整型运算时



输入（ML00100）	输出（DB000000）
-100000 ＞ ML00100	OFF
-100000 ≦ ML00100 ≦ 100000	ON
ML00100 ＞ 100000	OFF

实型运算时



输入（MF00100）	输出（DB000000）
-10.5 ＞ DF00100	OFF
-10.5 ≦ DF00100 ≦ 10.5	ON
DF00100 ＞ 10.5	OFF

1.4 程序控制指令

1.4.1 图调用指令 (SEE)

■ 概要

从主图调用子图，或从子图调用孙图时使用。但不同系列的图之间无法调用。例如，在 DWG.L 中无法表述 SEE.H01 的内容。

■ 格式

标记：SEE
全称：Call Program
类别：CONTROL
图标： 

■ 参数

参数名称	设定
Name	程序名

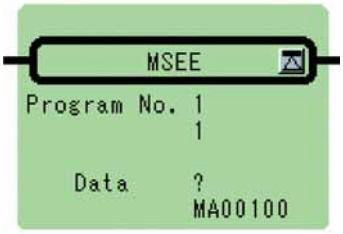
■ 程序举例

1.4.2　运动程序调用指令（MSEE）

■ 概要

调用指定的运动程序。

■ 格式

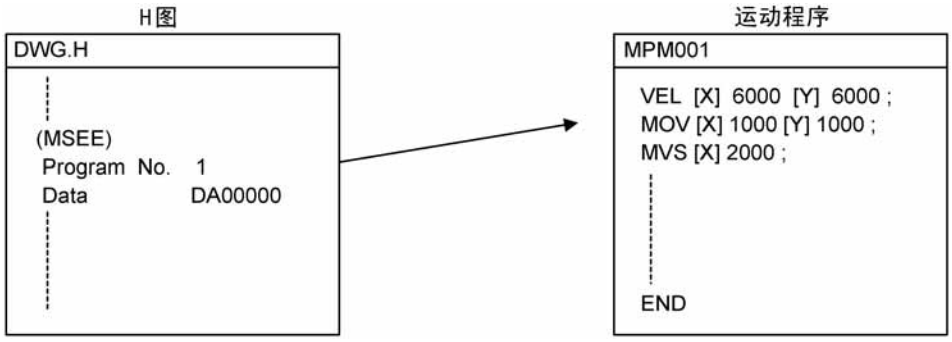


标记：MSEE
全称：Call Motion Program
类别：MOTION
图标：

■ 参数

参数名称	设定
Program No. （运动程序编号）	• 直接指定：001 ～ 256 的数值 • 间接指定：整型寄存器
Data （工作寄存器）	• 寄存器地址（#、 C 寄存器除外）

■ 程序举例

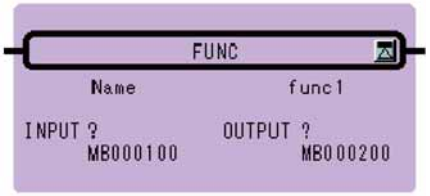


1.4.3 函数调用指令（FUNC）

■ 概要

从图，子图调用，以及从用户函数调用用户函数时使用。请事先定义需调用的用户函数。（ 系统已对系统函数做出了定义，因此没有必要再进行定义。）根据函数定义定义输入名称、输入输出数。

■ 格式



标记：FUNC
全称：User Function
类别：CONTROL
图标： 

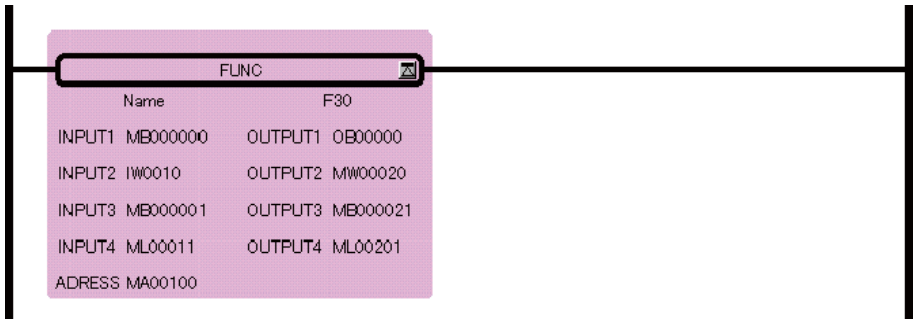
■ 参数

参数名称	设定
Name	函数程序名
INPUT（根据函数定义）	输入参数（类型取决于函数定义）
ADRESS（根据函数定义）	输入地址参数（地址型寄存器）
OUTPUT（根据函数定义）	输出参数（类型取决于函数定义）

由函数定义决定的输入输出参数的数据类型如下表所示。

输入数据类型	输入指定	说明
位输入	B-VAL	输入指定为比特型。 此比特型数据为函数的输入。
整型输入	I-VAL	输入指定为整型。 指定寄存器编号的内容（整型数据）为函数的输入。
	I-REG	输入指定为整型寄存器编号的内容。调用函数时，指定整型寄存器编号。 指定寄存器编号的内容（整型数据）为函数的输入。
长整型输入	L-VAL	输入指定为长整型。 调用函数时，指定寄存器编号的内容（长整型数据）为函数的输入。
	L-REG	输入指定为长整型寄存器编号的内容。 调用函数时，长整型寄存器编号的内容（长整型数据）为函数的输入。
实型输入	F-VAL	输入指定为实型。 指定寄存器编号的内容（实型数据）为函数的输入。
	F-REG	输入指定为实型寄存器编号的内容。调用函数时，指定实型寄存器编号。 指定寄存器编号的内容（实型数据）为函数的输入。
地址输入	—	将指定寄存器（任意整型寄存器）的地址传递给函数。用户函数时仅为一次输入。

■ 程序举例

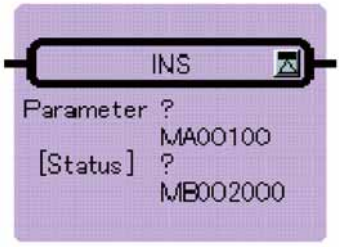


1.4.4　连续执行型直接输入指令（INS）

■ 概要

与系统输入输出（统一输入／统一输出）独立，在用户程序中执行输入输出。输入输出在直接输入输出指令执行时进行。在输入输出动作结束之前，不执行下一指令。

■ 格式



标记：INS
全称：Direct Input String
类别：CONTROL
图标：

■ 参数

参数名称	设定
Parameter （参数表首地址）	• 寄存器地址（#、C 寄存器除外） • 同上带下标字母
[Status]*	• 比特型寄存器（#、C 寄存器除外） • 同上带下标字母

*可省略 [Status]。

表 1.10 INS 指令参数 / 数据表

ADR	类型	符号	名称	规格	输入输出
0	W	RSSEL	单元指定 1	指定输入单元	IN
1	W	MDSEL	单元指定 2		IN
2	W	STS	状态	每个字的输入状态按相应的位输出	OUT
3	W	N	字数	指定连续输入字数	IN
4	W	ID1	输入数据 1	输出输入的数据 出错时存储 0 （MP930 时，仅 1 个数据）	OUT
• • •	• • •	• • •	• • •		• • •
N + 3	W	IDN	输入数据 N		OUT

RSSEL、MDSEL 的设定方法

1. RSSEL

指定对象模块的安装单元 / 插槽。

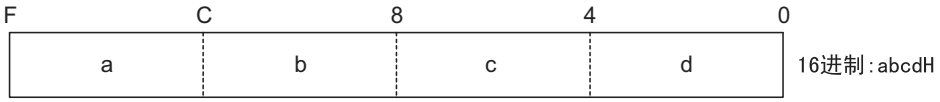
16 进制表示：xxyyH

xx ＝单元编号（1 ～ 4）

yy ＝插槽编号（1 ～ 9）

补充 在 MP930 中，单元编号＝ 1、插槽编号＝ 3 为固定。

2. MDSEL

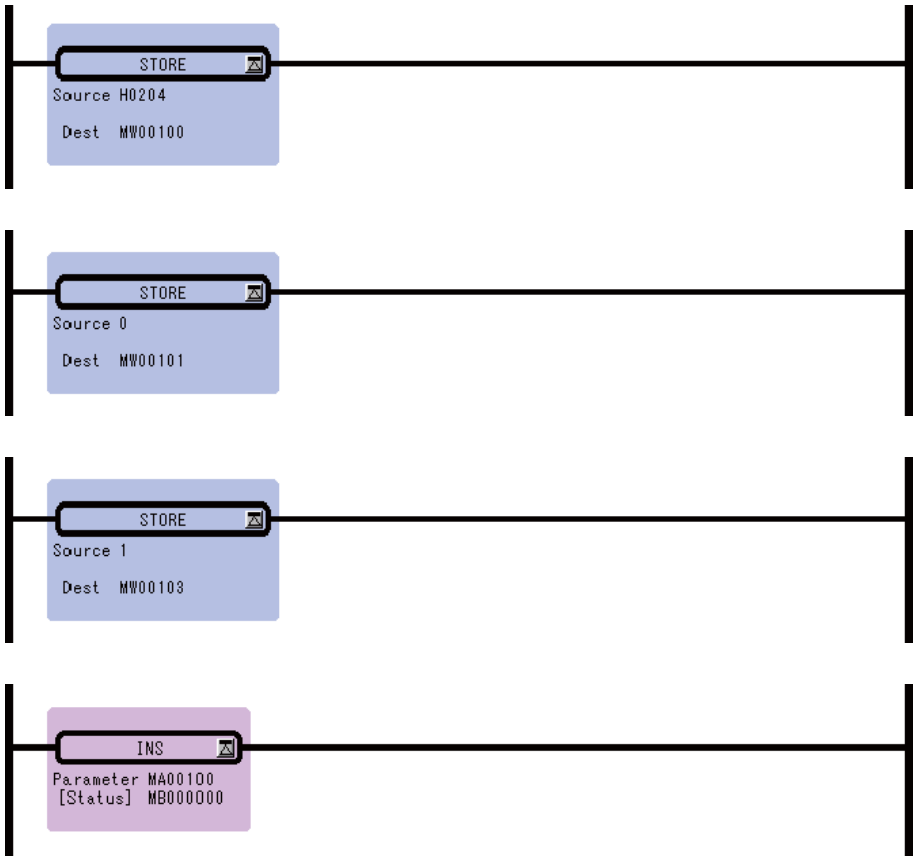


a：输入模块种类
0：分立输入模块
1：寄存器输入模块
b：单元编号（1 ～ 4）
c：插槽编号（1 ～ 9）
d：数据偏置（0 ～ 7）

补充 在 MP930 中，输入模块种类＝ 0、单元编号＝ 1、插槽编号＝ 3、数据偏置＝ 0 均被固定。

■ 程序举例

从安装在单元 2/ 插槽 4 上的 LIO 输入数据。



1.4.5 连续执行型直接输出指令（OUTS）

■ 概要

按照事先设定的参数 / 数据表的内容，对 1 模块执行连续直接输出。适用于直接输出的单元仅为 LIO。

■ 格式

标记：OUT
全称：Direct Output String
类别：CONTROL
图标：

■ 参数

参数名称	设定
Parameter （参数表首地址）	• 寄存器地址（#、C 寄存器除外） • 同上带下标字母
[Status]*	• 比特型寄存器（#、C 寄存器除外） • 同上带下标字母

* 可省略 [Status]。

表 1.11 OUTS 指令参数 / 数据表

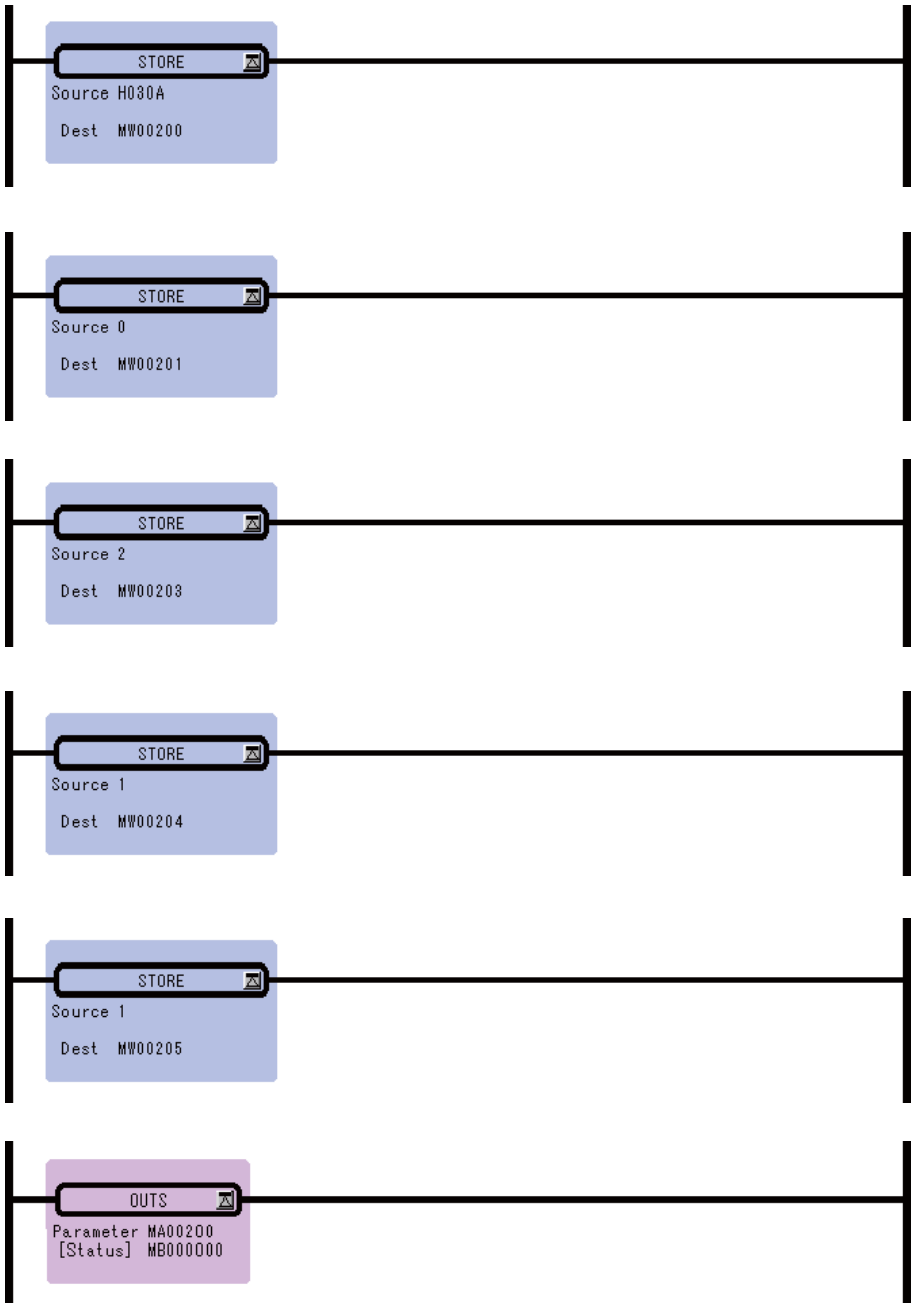
ADR	类型	符号	名称	规格	输入输出
0	W	RSSEL	单元指定 1	指定进行输出的模块	IN
1	W	MDSEL	单元指定 2		IN
2	W	STS	状态	每个字的输出状态按相应的位输出	OUT
3	W	N	字数	指定连续输出字	IN
4	W	OD1	输出数据 1	设定输出数据 （MP930 时，仅 1 个数据）	IN
• • •	• • •	• • •	• • •		• • •
N + 3	W	ODN	输出数据 N		IN

RSSEL、MDSEL 的设定方法

RSSEL、MDSEL 的设定方法与 INS 指令相同。

■ 程序举例

向单元 3/ 插槽 10 上安装的 LIO-01 输出 2 个字。



补充

MP930 时因本地 I/O 被分配为默认值，通过使用 OUTS 指令，进行每个扫描周期两次输出。

1.4.6 扩展程序执行指令（XCALL）

■ 概要

在执行扩展程序时使用。

所谓扩展程序，是指表格式程序。在 1 个图中能使用多个 XCALL 指令，但不能 2 次以上调用同样的扩展程序。

■ 格式

标记：XCALL

全称：Call Extended Program

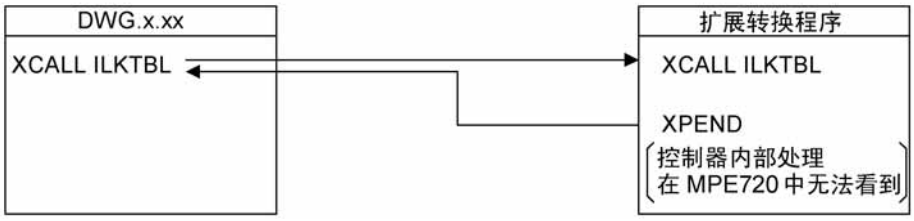
类别：CONTROL

图标：

■ 参数

参数名称	设定
Name （扩展程序种类）	MCTBL：常数表（M 寄存器） IOTBL：输入输出转换表 ILKTBL：联锁表 ASMTBL：部件总成表

■ 程序举例



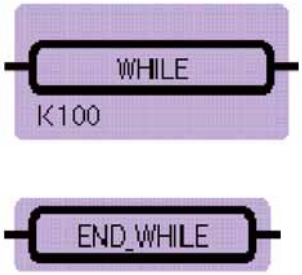
1.4.7 WHILE 指令 (WHILE, END_WHILE)

■ 概要

满足 WHILE 指令的条件表达式时，执行 WHILE…END_WHILE 间的循环体（指令体）。条件不满足时，不执行指令体，而进入 END_WHILE 下面的指令。

■ 格式

• 指令表达式 ON 时



标记：WHILE，END_WHILE
全称：While Do，End of While
类别：CONTROL
图标： 

• 指令表达式 OFF 时

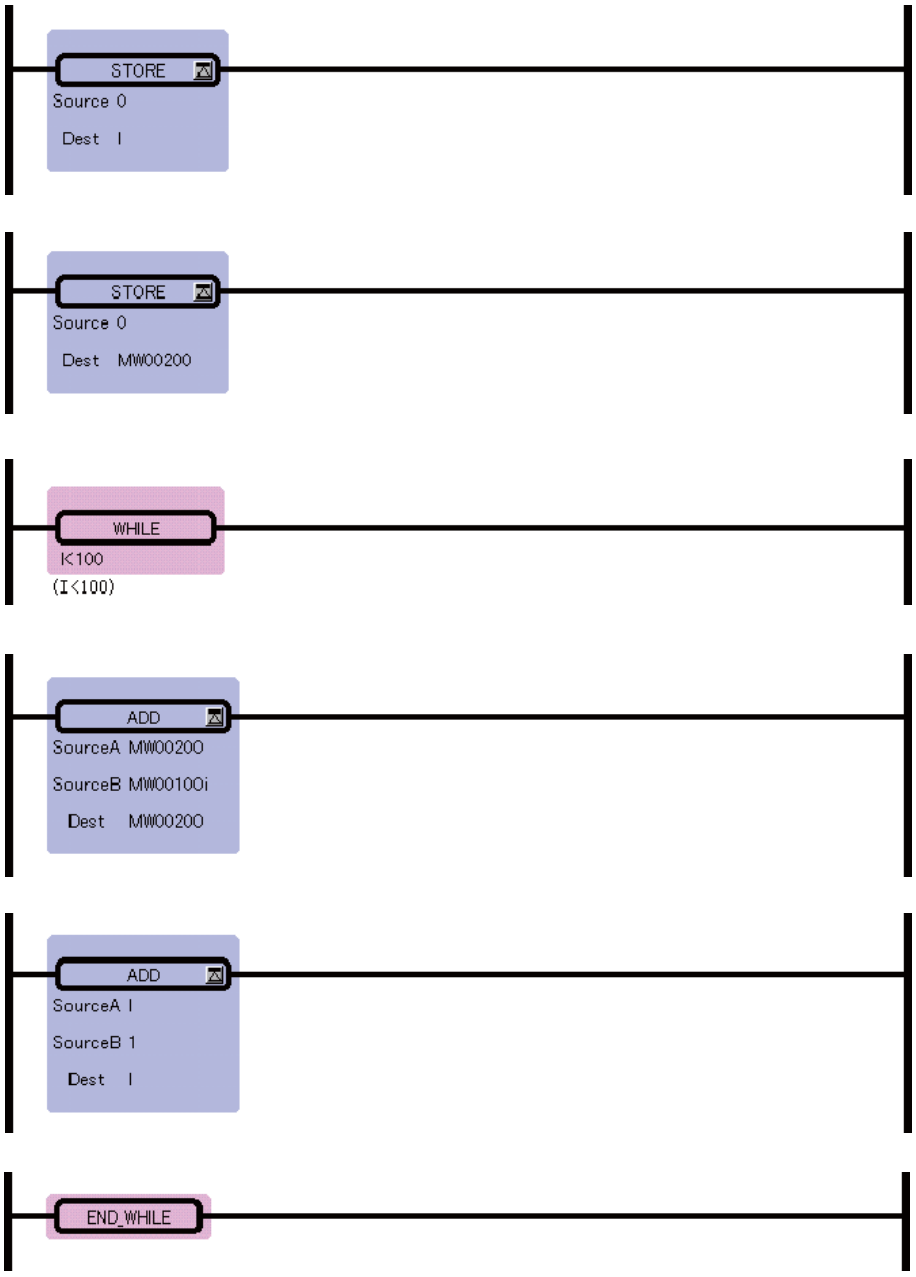
标记：WHILE-END_WHILE
全称：While Do and End of While
类别：CONTROL
图标： 

■ 参数

参数名称	设定
条件表达式	条件表达式用 Expression 表述。

■ 程序举例

对 MW00100 到 MW00199 的 100 个寄存器进行合计，将其结果存储在 MW00200 中。



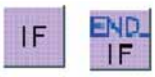
1.4.8　IF指令（IF，END_IF）

■ 概要

IF指令的条件表达式成立时，执行IF和END_IF之间的指令体，不成立时不执行。

■ 格式

• 指令表达式ON时

标记：IF，END_IF
全称：If Then，End of If
类别：CONTROL
图标： 

• 指令表达式OFF时

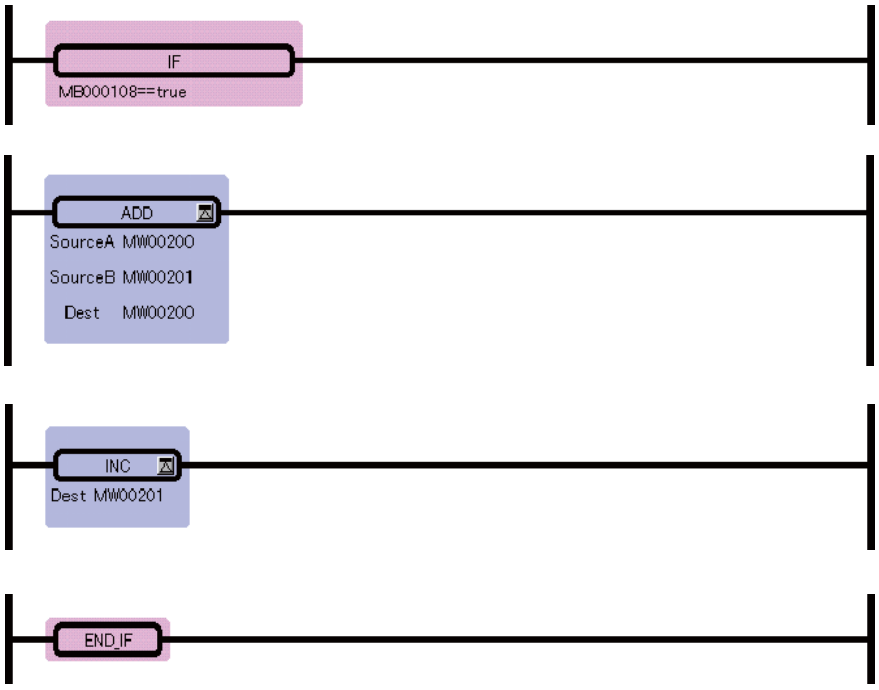
标记：IF-END_IF
全称：If Then and End of If
类别：CONTROL
图标： 

■ 参数

参数名称	设定
条件表达式	条件表达式用Expression表述。

■ 程序举例

MW00108 为 ON 时，给 MW00200 上加 MW00201，同时实现 MW00201 的自增。



1.4.9　IF 指令（IF，ELSE，END_IF）

■ 概要

IF 指令的条件表达式成立时，仅执行指令体 -1，不执行指令体 -2。

IF 指令的条件表达式不成立时，仅执行指令体 -2，不执行指令体 -1。

■ 格式

• 指令表达式 ON 时

标记：IF，ELSE，END_IF
全称：If Then，Else，End of If
类别：CONTROL
图标：

• 指令表达式 OFF 时

标记：IF-ELSE-END_IF
全称：If Then and Else and End of If
类别：CONTROL
图标：

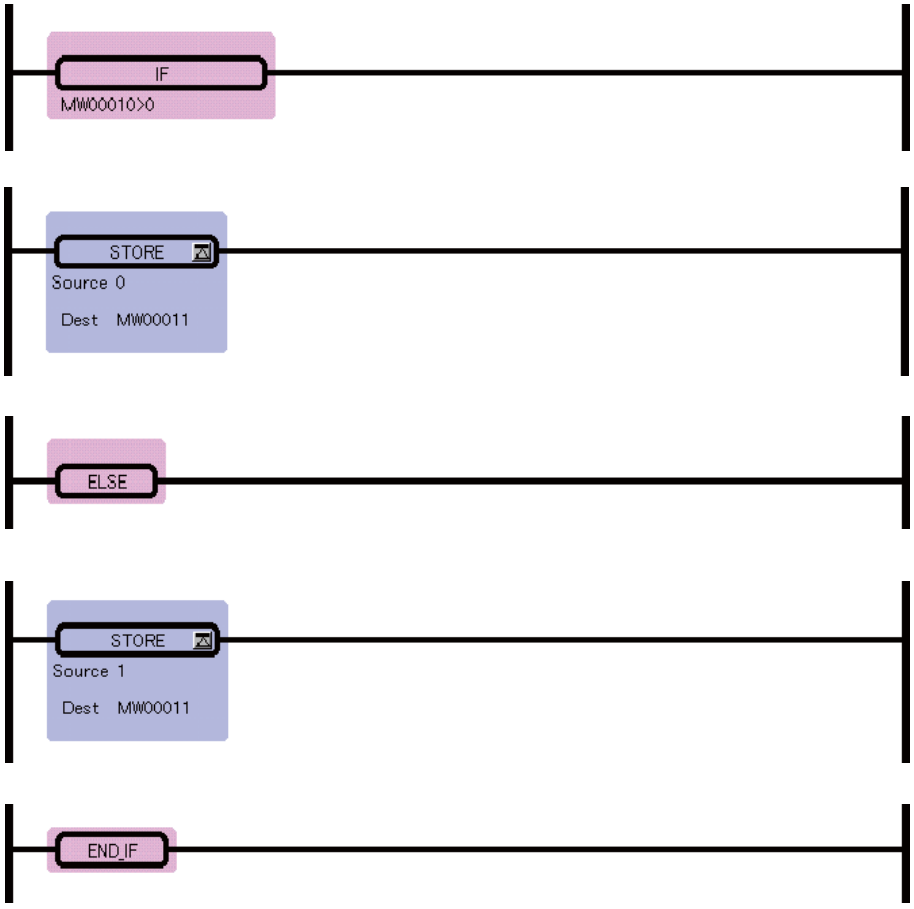
■ 参数

参数名称	设定
条件表达式	条件表达式用 Expression 表述。

■ 程序举例

MW00010 的内容大于 0 时，将 MW00011 的内容设为 0。

除此以外，将 MW00011 的内容设为 1。



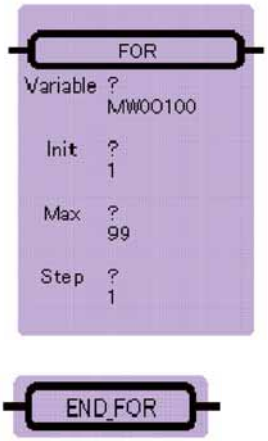
1.4.10　FOR 指令 (FOR，END_FOR)

■ 概要

FOR 指令和相应的 END_FOR 指令之间的指令体，仅循环执行指定的次数
N = (B − A + 1)/C。
变量 Variable 从初始值 Init 开始每循环执行一次就自加 Step 一次，当最终 Variable ＞ Max 时结束循环。

■ 格式

• 指令表达式 ON 时



标记：FOR，END_FOR
全称：For，End of For
类别：CONTROL
图标： 

• 指令表达式 OFF 时

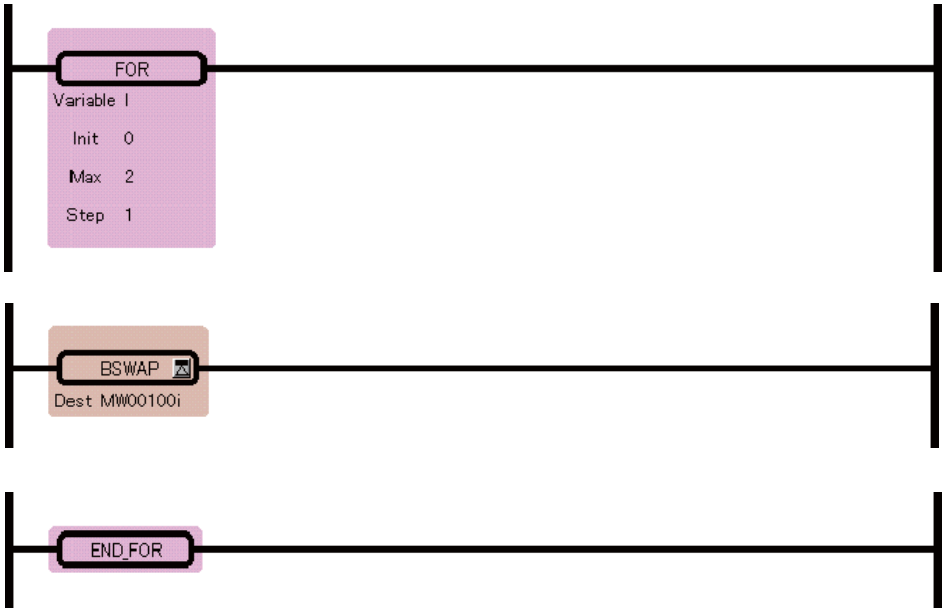
标记：FOR-END_FOR
全称：For and End of For
类别：CONTROL
图标： 

■ 参数

参数名称	设定
Variable（变量）	• 所有整型寄存器 • 同上带下标字母 • 下标寄存器（I、J）
Init（初始值）	• 所有整型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Max（最大值）	• 所有整型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Step（增加部分）	• 所有整型寄存器 • 同上带下标字母 • 下标寄存器 • 常数

■ 程序举例

交换 MW00100 ～ MW00102 的高位字节和低位字节。



1.4.11　EXPRESSION 指令 (EXPRESSION)

■ 概要

EXPRESSION 指令由一个块构成。

与线圈型元件考虑方法相同，输入行有启用 / 禁用指令的意思，因此左右两侧均可。

块内设置有运算表达式表述用的 Expression 盒，最多可进行 100 行运算表达式的表述。

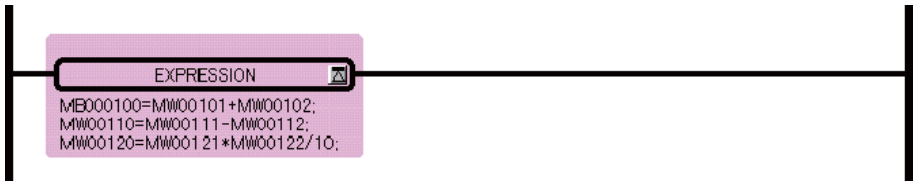
■ 格式

标记：EXPRESSION
全称：Expression
类别：CONTROL
图标：

■ 参数

参数名称	设定
运算表达式	表述运算表达式。

■ 程序举例



1.5 基本函数指令

1.5.1 平方根指令 (SQRT)

■ 概要

对整型及实型数据进行平方根运算。整型及实型的输入单位、输出结果不同。不能使用长整型数据。

整型数据

将输入 (Source) 的平方根存储在输出 (Dest) 中。
不同于数学概念中的平方根。在 MP 梯形图编辑器中使用的平方根按以下算式表示。

平方根＝ 32768 × sign(A) × SQRT(|A|/32768)
sign(A)：输入的符号
|A|：输入的绝对值

即，为数学平方根乘以 128$\sqrt{2}$(约 181.02) 得到的值。另，输入值为负数时，则计算绝对值的平方根，输出负数的运算结果。

输出值的最大误差为± 2。

实型数据

将输入 (Source) 的平方根存储在输出 (Dest) 中。另，输入值为负数时，则计算绝对值的平方根，输出负数的运算结果。

■ 格式

标记：SQRT
全称：Square Root
类别：FUNCTION
图标：

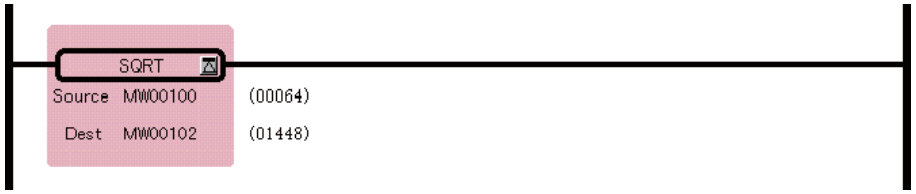
■ 参数

参数名称	设定
Source（输入）	• 所有整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Dest（输出）	• 整型、实型寄存器（#、C 寄存器除外） • 同上带下标字母 • 下标寄存器

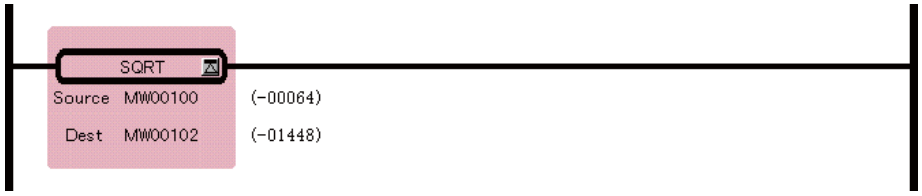
■ 程序举例

整型数据

• 输入值为正数时

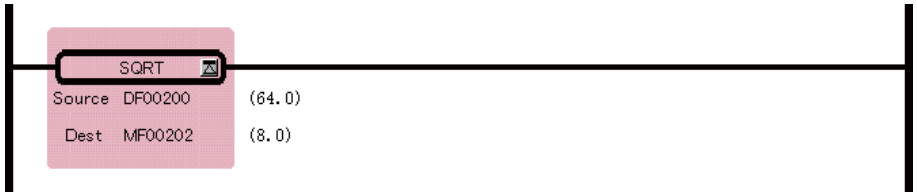


• 输入值为负数时

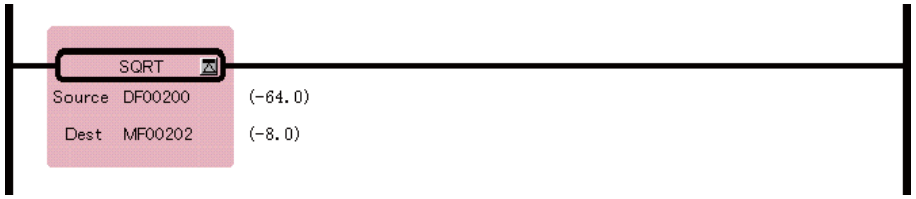


实型数据

• 输入值为正数时



• 输入值为负数时



1.5.2 正弦指令 (SIN)

■ 概要

对整型及实型数据进行正弦运算。整型及实型的输入单位、输出结果不同。不能使用长整型数据。

整型数据

可在 -327.68 ～ 327.67 度范围内使用。将 Source 作为输入 (1 ＝ 0.01 度) 使用，并将运算结果存储在 Dest 中。运算结果按 10000 倍的值输出。

如果误输入 -327.68 ～ 327.67 度以外的值，将不能得到正确的结果。例如，当输入 360.00 时，会输出 -295.36 度的结果。

实型数据

将 Source 作为输入 (单位＝度) 使用，并将其正弦值存储在 Dest 中。

■ 格式

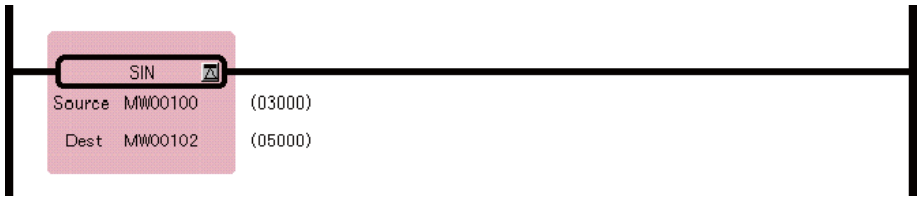
标记：SIN
全称：Sine
类别：FUNCTION
图标：

■ 参数

参数名称	设定
Source(输入)	• 所有整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Dest(输出)	• 整型、实型寄存器 (#、C 寄存器除外) • 同上带下标字母 • 下标寄存器

■ 程序举例

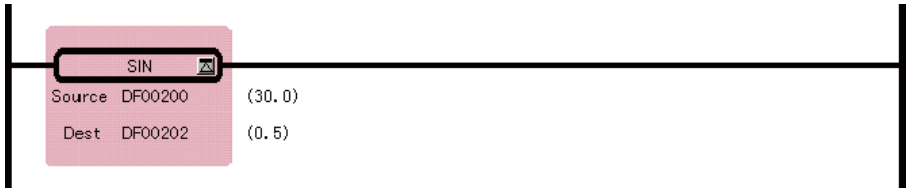
整型数据



输入　θ = 30 度 (MW00100 = 30 × 100 = 3000)

输出　SIN(θ) = 0.50(MW00102 = 0.50 × 10000 = 5000)

实型数据



1.5.3 余弦指令（COS）

■ 概要

对整型及实型数据进行余弦运算。整型及实型的输入单位、输出结果不同。不能使用长整型数据。

整型数据

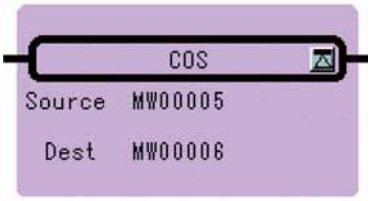
可在 -327.68 ～ 327.67 度范围内使用。将 Source 作为输入（1 ＝ 0.01 度）使用，并将运算结果存储在 Dest 中。运算结果按 10000 倍的值输出。

如果误输入 -327.68 ～ 327.67 度以外的值，将不能得到正确的结果。例如，当输入 360.00 时，会输出 -295.36 度的结果。

实型数据

将 Source 作为输入（单位＝度）使用，并将其余弦值存储在 Dest 中。

■ 格式



标记：COS

全称：Cosine

类别：FUNCTION

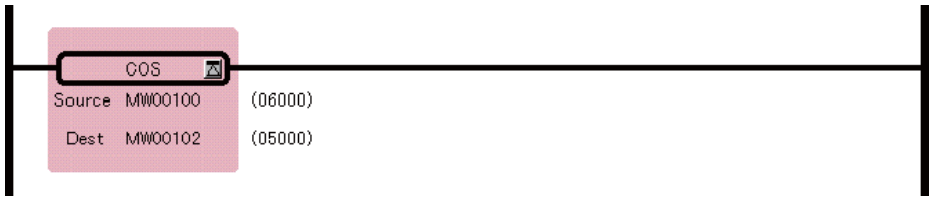
图标：

■ 参数

参数名称	设定
Source（输入）	• 所有整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Dest（输出）	• 整型、实型寄存器（#、C 寄存器除外） • 同上带下标字母 • 下标寄存器

■ 程序举例

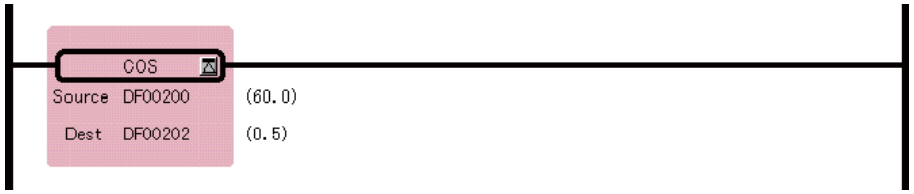
整型数据



输入　θ = 60 度（MW00100 = 60 × 100 = 6000）

输出　COS（θ）= 0.50（MW00102 = 0.50 × 10000 = 5000）

实型数据

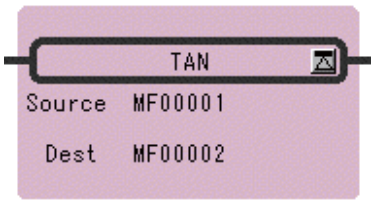


1.5.4 正切指令（TAN）

■ 概要

将 Source 作为输入（单位＝度）使用，并将其正切值存储在 Dest 中。仅实型运算。

■ 格式



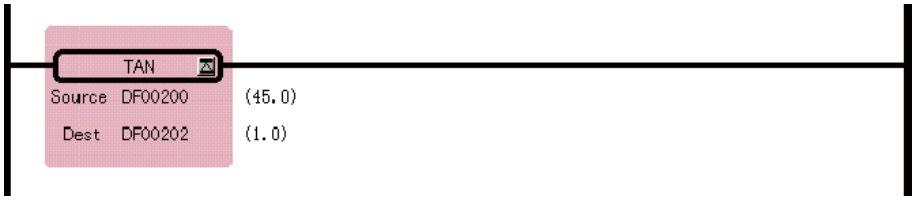
标记：TAN
全称：Tangent
类别：FUNCTION
图标：

■ 参数

参数名称	设定
Source（输入）	• 所有实型寄存器 • 同上带下标字母 • 常数
Dest（输出）	• 实型寄存器（#、C 寄存器除外） • 同上带下标字母

■ 程序举例

计算输入值（θ＝45.0 度）的正弦 TAN（θ）＝1.0。



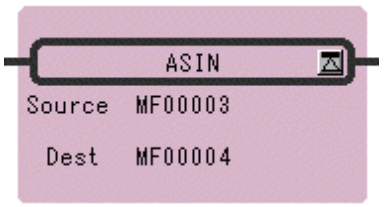
TAN 指令不能使用整型、长整型数据。

1.5.5　反正弦指令（ASIN）

■ 概要

将 Source 作为输入使用，并将其反正弦值（单位＝度）存储在 Dest 中。仅实型运算。

■ 格式



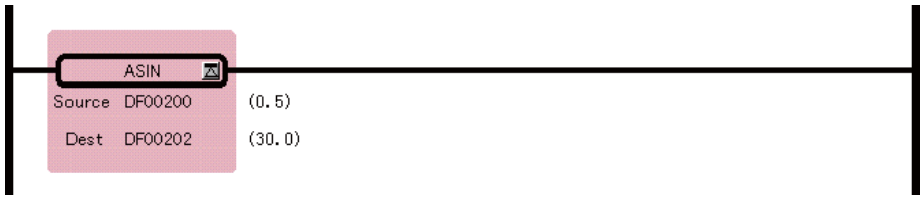
标记：ASIN
全称：Arc Sine
类别：FUNCTION
图标：

■ 参数

参数名称	设定
Source（输入）	• 所有实型寄存器 • 同上带下标字母 • 常数
Dest（输出）	• 实型寄存器（#、C 寄存器除外） • 同上带下标字母

■ 程序举例

计算输入值（0.5）的反正弦值 30.0 度＝ θ ＝ ASIN(0.5)。



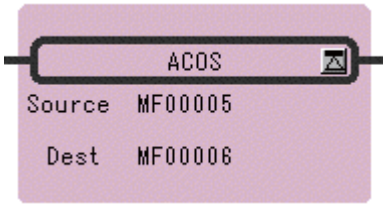
ASIN 指令不能使用整型、长整型数据。

1.5.6 反余弦指令（ACOS）

■ 概要

将 Source 作为输入使用，并将其反余弦值（单位＝度）存储在 Dest 中。仅实型运算。

■ 格式



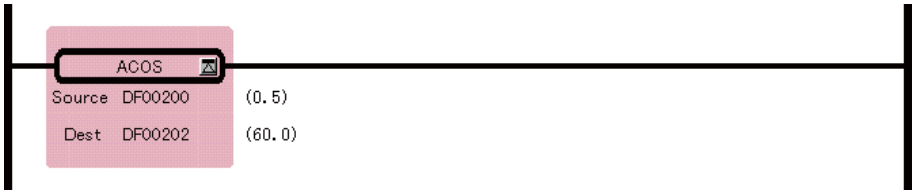
标记：ACOS
全称：Arc Cosine
类别：FUNCTION
图标：

■ 参数

参数名称	设定
Source（输入）	• 所有实型寄存器 • 同上带下标字母 • 常数
Dest（输出）	• 实型寄存器（#、C 寄存器除外） • 同上带下标字母

■ 程序举例

计算输入值（0.5）的反余弦值 60.0 度＝ θ ＝ ACOS(0.5)。



ACOS 指令不能使用整型、长整型数据。

1.5.7　反正切指令（ATAN）

■ 概要

对整型及实型数据进行反正切运算。整型及实型的输入单位、输出结果不同。不能使用长整型数据。

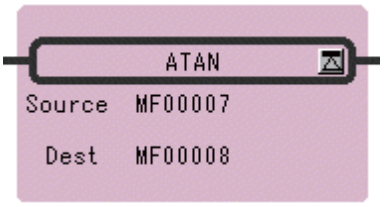
整型数据

可在 -327.68 ～ 327.67 度范围内使用。将 Source 作为输入（1 ＝ 0.01）使用，并将运算结果存储在 Dest 中。运算结果按 100 倍值输出。

实型数据

将 Source 作为输入使用，并将其反正切值（ 单位＝度 ）存储在 Dest 中。仅实型运算。不能使用整型、长整型数据。

■ 格式



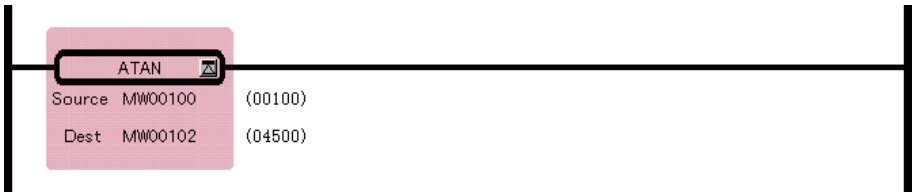
标记：ATAN
全称：Arc Tangent
类别：FUNCTION
图标：

■ 参数

参数名称	设定
Source（输入）	• 所有整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Dest（输出）	• 整型、实型寄存器（#、C 寄存器除外） • 同上带下标字母 • 下标寄存器

■ 程序举例

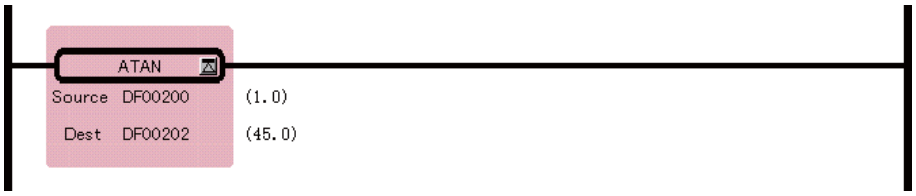
整型数据



输入　X = 1.00(MW00100 = 1.00 × 100 = 100)

输出　θ = 45 度 (MW00102 = 45 × 100 = 4500)

实型数据



1.5.8　指数指令（EXP）

■ 概要

将 Source 作为输入（x）使用，并将其自然对数的底（e）的输入次幂（e^x）作为运算结果存储在 Dest 中。仅实型运算。

■ 格式

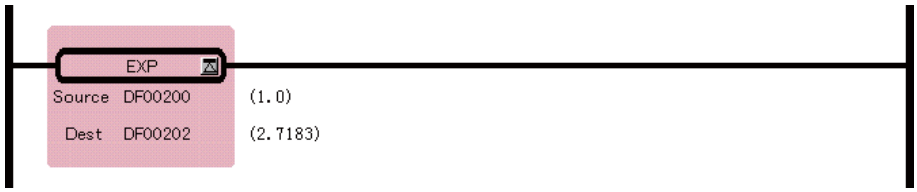
标记：EXP
全称：Exponential
类别：FUNCTION
图标：

■ 参数

参数名称	设定
Source（输入）	• 所有实型寄存器 • 同上带下标字母 • 常数
Dest（输出）	• 实型寄存器（#、C 寄存器除外） • 同上带下标字母

■ 程序举例

对输入值（x ＝ 1.0），计算 e 的幂（e ＝ 2.7183）。



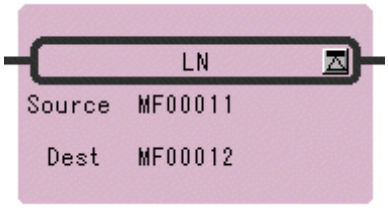
EXP 指令的运算结果上溢时，存储最大值（3.4…E ＋ 38），不会发生运算错误。

1.5.9 自然对数指令 (LN)

■ 概要

将 Source 作为输入 (x) 使用，并将其自然对数 (log e^x) 作为运算结果存储在 Dest 中。仅实型运算。

■ 格式



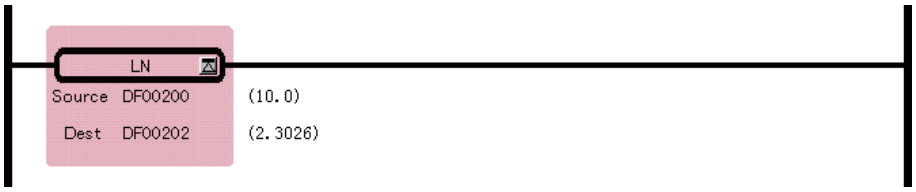
标记：LN
全称：Natural Logarithm
类别：FUNCTION
图标：

■ 参数

参数名称	设定
Source（输入）	• 所有实型寄存器 • 同上带下标字母 • 常数
Dest（输出）	• 实型寄存器（#、C 寄存器除外） • 同上带下标字母

■ 程序举例

对输入值 (x ＝ 10.0)，计算其自然对数 Loge(x) ＝ 2.3026。



补充

在 LN 指令中检查输入值 (x)，执行以下处理。

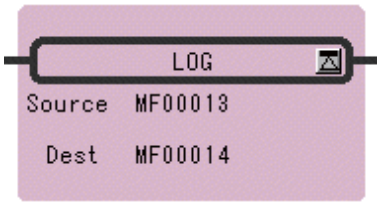
- 输入为负 LN(-1) 时，用绝对值计算。
- 输入为零 LN(0) 时，解为－∞。

1.5.10　常用对数指令（LOG）

■ 概要

将 Source 作为输入（x）使用，并将其常用对数（$\log_{10}x$）作为运算结果存储在 Dest 中。仅实型运算。

■ 格式



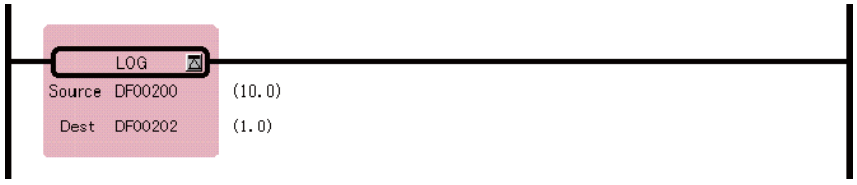
标记：LOG
全称：Logarithm Base 10
类别：FUNCTION
图标：

■ 参数

参数名称	设定
Source（输入）	• 所有实型寄存器 • 同上带下标字母 • 常数
Dest（输出）	• 实型寄存器（#、C 寄存器除外） • 同上带下标字母

■ 程序举例

对输入值（x ＝ 10.0），计算其自然对数 $Log_{10}(x)$ ＝ 1.0。



在 LOG 指令中检查输入值（x），执行以下处理。

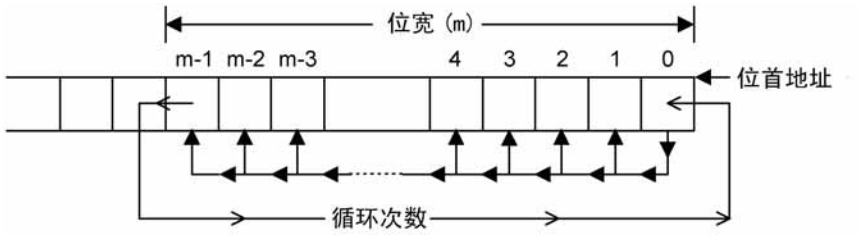
- 输入为负 LOG(-1) 时，用绝对值计算。
- 输入为零 LOG(0) 时，解为－∞。

1.6 数据操作指令

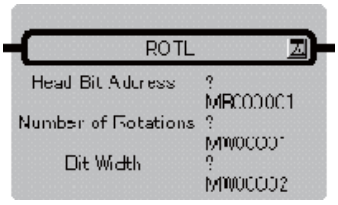
1.6.1 位循环左移指令（ROTL）

■ 概要

对由位首地址和位宽指定的位表按规定的循环次数，进行循环左移。



■ 格式



标记：ROTL

全称：Bit Rotate Left

类别：MOVE

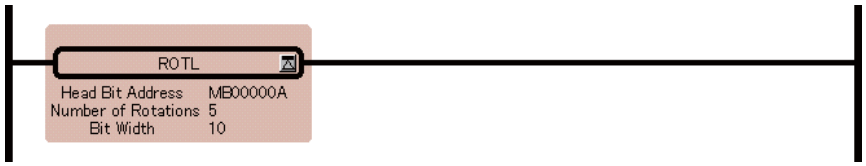
图标：

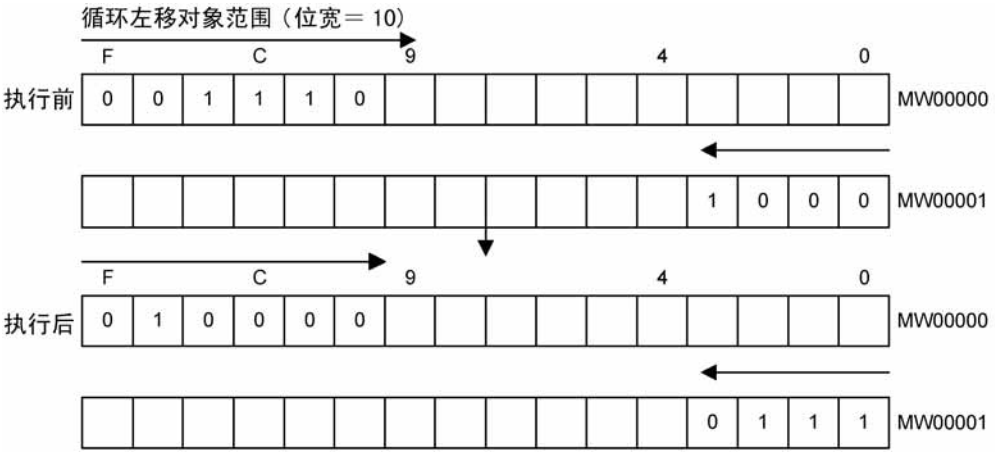
■ 参数

参数名称	设定
Head Bit Address （位首地址）	• 比特型地址（#、C 寄存器除外） • 同上带下标字母
Number of Rotations （循环次数）	• 所有整型寄存器 • 同上带下标字母 • 常数
Bit Width （位宽）	• 所有整型寄存器 • 同上带下标字母 • 常数

■ 程序举例

将从 MB00000A(MW00000 的位 A) 开始的位宽为 10 的数据循环左移 5 次。

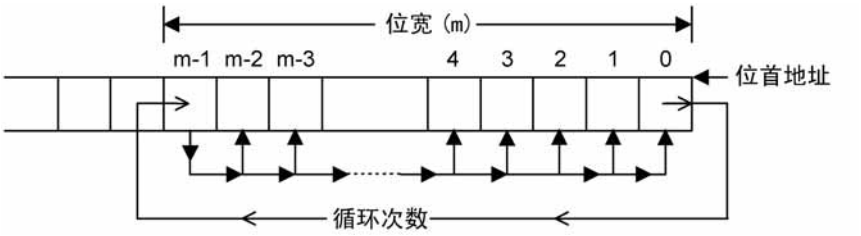




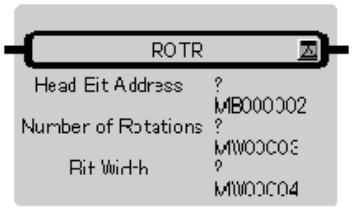
1.6.2 位循环右移指令 (ROTR)

■ 概要

对由位首地址和位宽指定的位表按规定的循环次数，进行循环右移。



■ 格式



标记：ROTR

全称：Bit Rotate Right

类别：MOVE

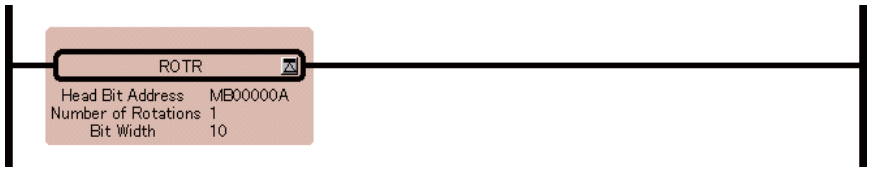
图标： 

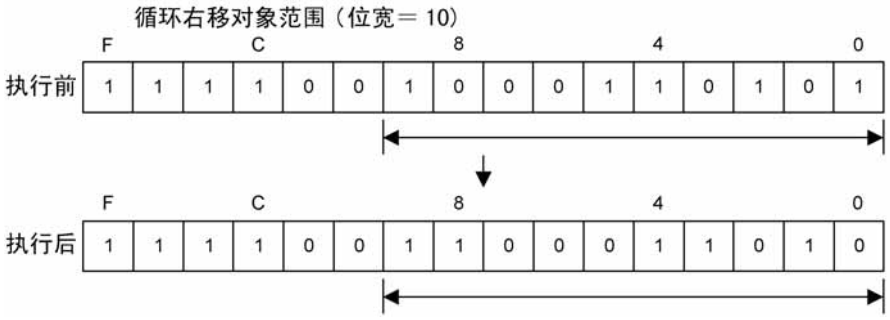
■ 参数

参数名称	设定
Head Bit Address （位首地址）	• 比特型地址 (#、C 寄存器除外） • 同上带下标字母
Number of Rotations （循环次数）	• 所有整型寄存器 • 同上带下标字母 • 常数
Bit Width （位宽）	• 所有整型寄存器 • 同上带下标字母 • 常数

■ 程序举例

将从 MB000000(MW00000 的位 0) 开始的位宽为 10 的数据循环右移 1 次。

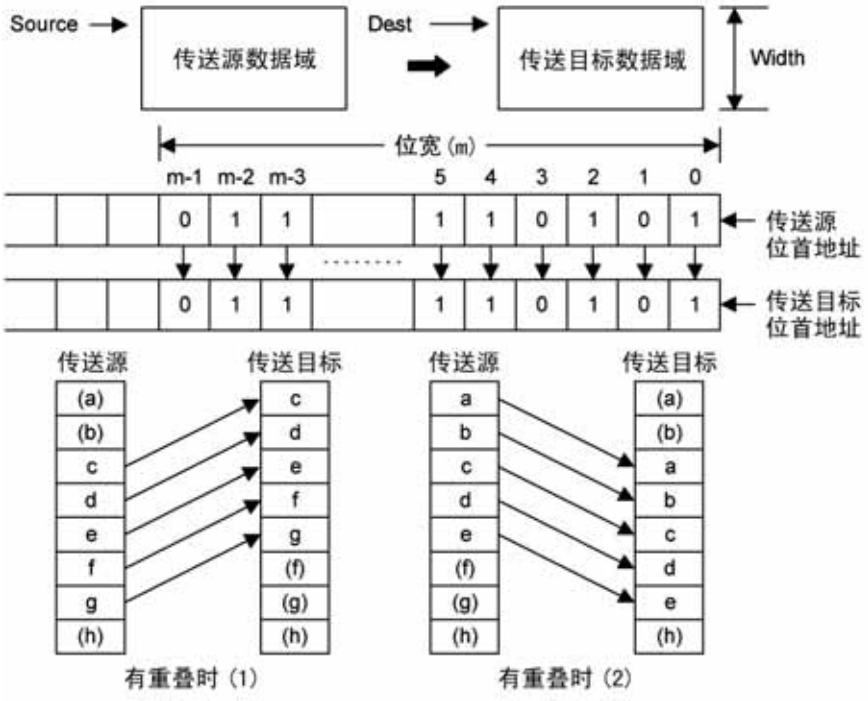




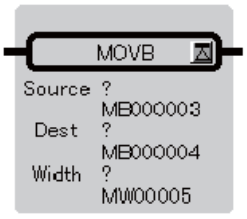
1.6.3 位传送指令（MOVB）

■ 概要

从传送源位首地址（Source）向传送目标的位首地址（Dest），传送相应宽度（Width）的位数据。传送是按继电器编号的增加方向逐位执行。
只要传送源位和传送目标位没有重叠，传送源位表将被保存。如果有重叠时，传送源位表有可能不被保存。



■ 格式



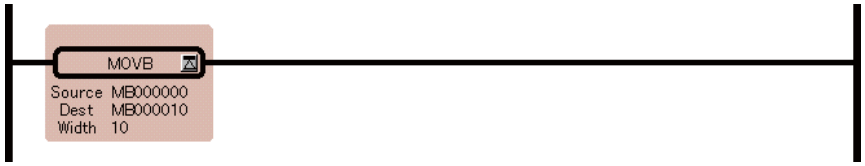
标记：MOVB
全称：Move Bit
类别：MOVE
图标：

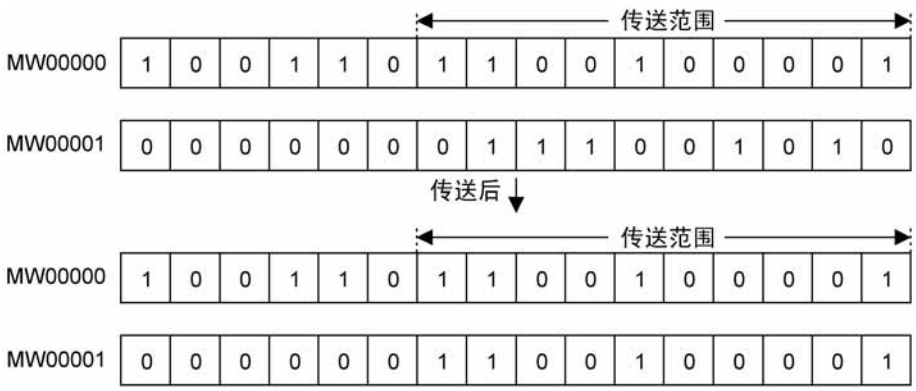
■ 参数

参数名称	设定
Source （传送源位地址）	• 所有比特型寄存器 • 同上带下标字母
Dest （传送目标位地址）	• 比特型寄存器 (#、C 寄存器除外） • 同上带下标字母
Width （位宽）	• 所有整型寄存器 • 同上带下标字母 • 常数

■ 程序举例

从 MB000000(MW00000 的位 0) 向 MB000010(MW00001 的位 0) 传送 10 位数据。

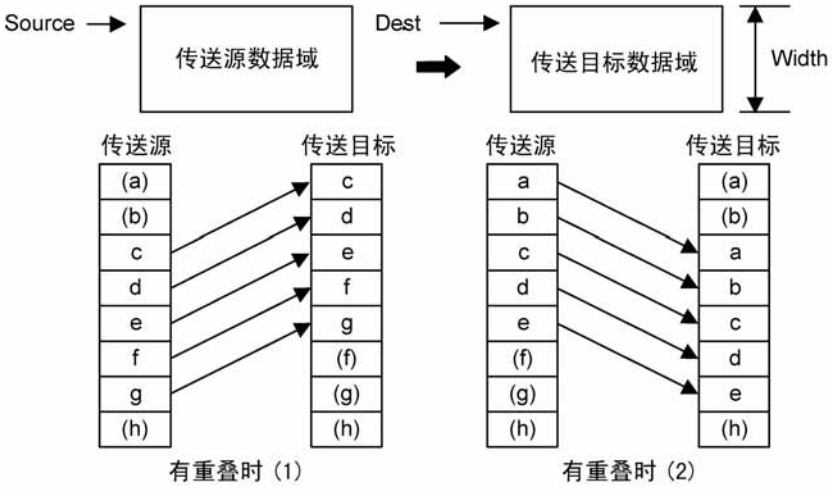




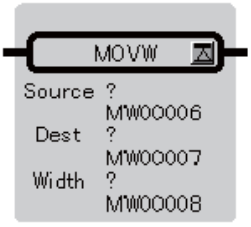
1.6.4 字传送指令（MOVW）

■ 概要

从传送源寄存器的首地址（Source）向传送目标寄存器的首地址（Dest）传送相应宽度（Width）的数据。传送作业按寄存器编号的增加方向逐个字执行。
只要传送源和传送目标没有重叠，传送源位表将被保存。如果有重叠时，传送源位表有可能不被保存。



■ 格式



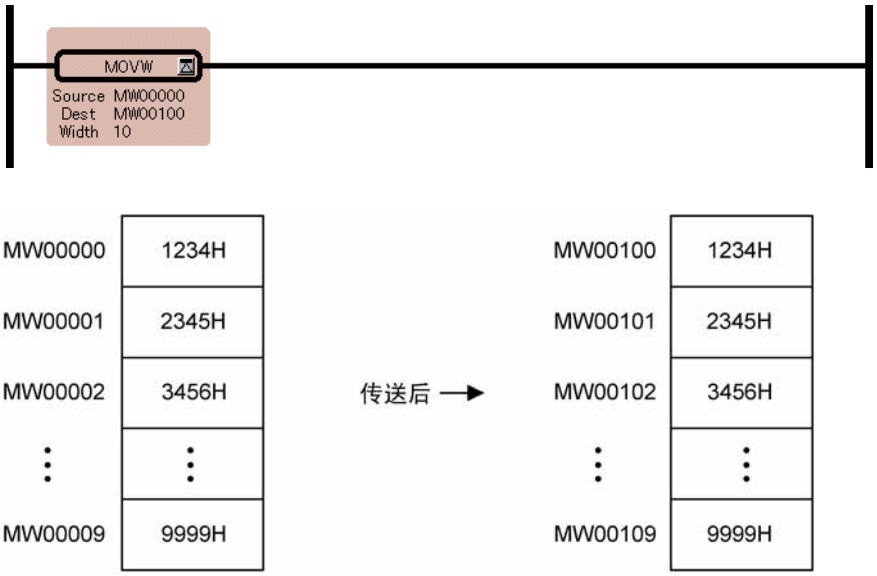
标记：MOVW
全称：Move Word
类别：MOVE
图标：

■ 参数

参数名称	设定
Source （传送源位地址）	• 所有整型寄存器 • 同上带下标字母
Dest （传送目标位地址）	• 整型寄存器（#、C 寄存器除外） • 同上带下标字母
Width （位宽）	• 所有整型寄存器 • 同上带下标字母 • 常数

■ 程序举例

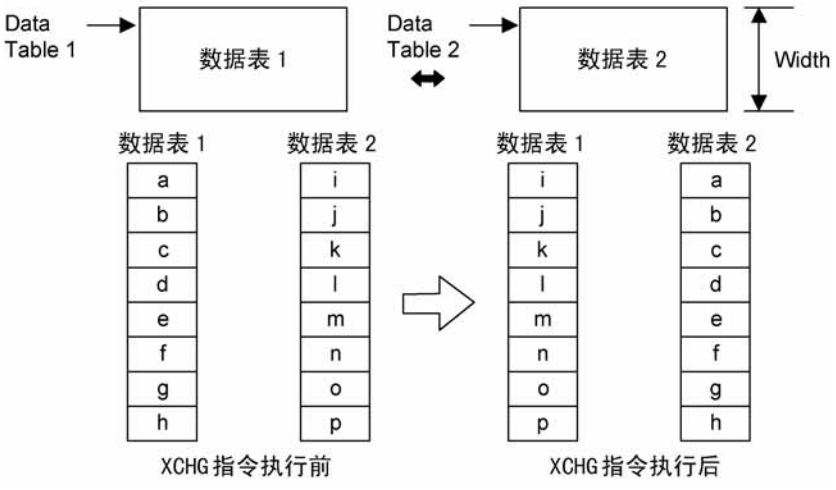
向 MW00100 ～ MW00109 传送 MW00000 ～ MW00009。



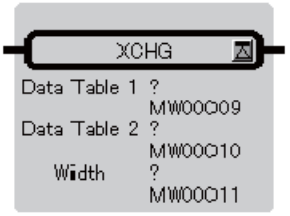
1.6.5 替换传送指令（XCHG）

■ 概要

数据表1(Data Table 1)和数据表2(Data Table 2)的数据按表宽(Width)交换内容。



■ 格式



标记：XCHG

全称：Exchange

类别：MOVE

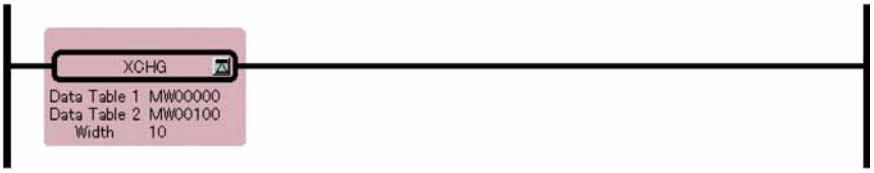
图标：

■ 参数

参数名称	设定
Data Table 1 （数据表 1）	• 整型寄存器（#、C 寄存器除外） • 同上带下标字母
Data Table 2 （数据表 2）	• 整型寄存器（#、C 寄存器除外） • 同上带下标字母
Width （表宽）	• 所有整型寄存器 • 同上带下标字母 • 常数

■ 程序举例

交换 MW00000 ～ MW00009 和 MW00100 ～ MW00109 的内容。

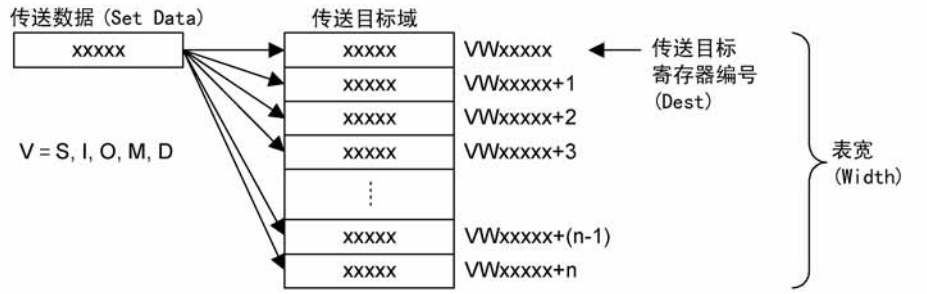


MW00000	1031H	MW00100	2050H		MW00000	2050H	MW00100	1031H
MW00001	1032H	MW00101	2051H		MW00001	2051H	MW00101	1032H
MW00002	1033H	MW00102	2052H		MW00002	2052H	MW00102	1033H
MW00003	1034H	MW00103	2053H		MW00003	2053H	MW00103	1034H
MW00004	1035H	MW00104	2054H	传送后→	MW00004	2054H	MW00104	1035H
MW00005	1036H	MW00105	2055H		MW00005	2055H	MW00105	1036H
MW00006	1037H	MW00106	2056H		MW00006	2056H	MW00106	1037H
MW00007	1038H	MW00107	2057H		MW00007	2057H	MW00107	1038H
MW00008	1039H	MW00108	2058H		MW00008	2058H	MW00108	1039H
MW00009	1030H	MW00109	2059H		MW00009	2059H	MW00109	1030H

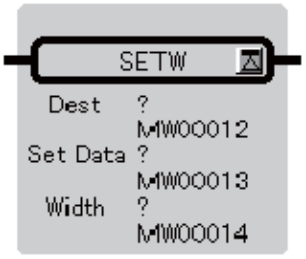
1.6.6 表初始化指令（SETW）

■ 概要

把由传送数据（Set Data）指定的数据存储在由传送目标寄存器编号（Dest）和表宽（Width）指定的所有寄存器中。存储作业按寄存器编号的增加方向逐个字进行。



■ 格式



标记：SETW

全称：Set Word

类别：MOVE

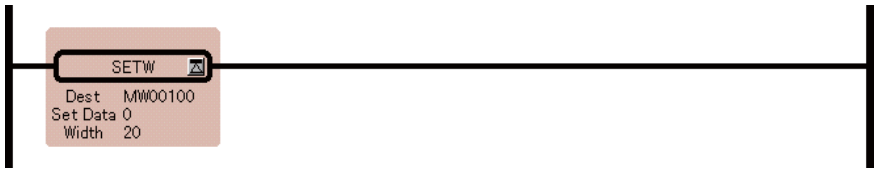
图标：

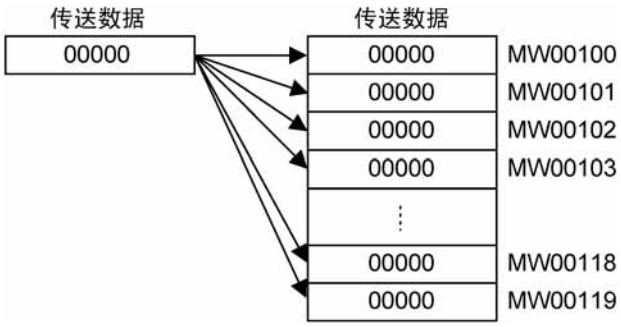
■ 参数

参数名称	设定
Dest （传送目标寄存器编号）	• 整型寄存器（#、C 寄存器除外） • 同上带下标字母
Set Data （传送数据）	• 整型寄存器（#、C 寄存器除外） • 同上带下标字母 • 常数
Width （表宽）	• 所有整型寄存器 • 同上带下标字母 • 常数

■ 程序举例

将 MW00100 ～ MW00119 的内容置为 0。

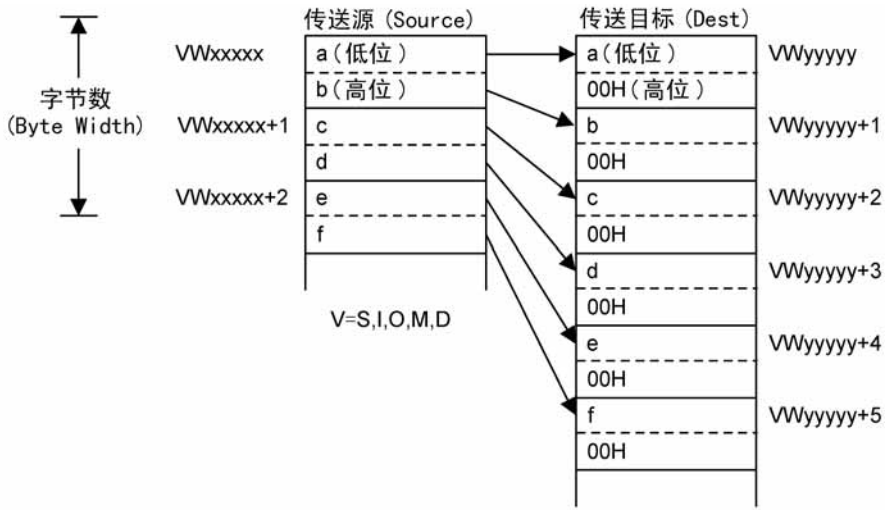




1.6.7 字节→字展开指令 (BEXTD)

■ 概要

将存储在传送源寄存器域 (Source) 内的字节列逐个字节地存储到传送目标寄存器 (Dest) 的字列中。传送目标寄存器的高位字节为 0。



■ 格式

标记：BEXTD

全称：Extend Byte to Word

类别：MOVE

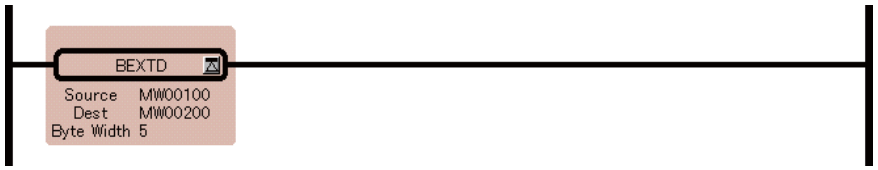
图标：

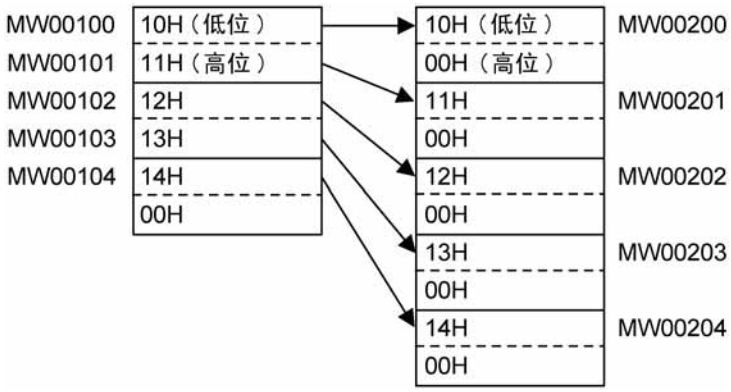
■ 参数

参数名称	设定
Source （传送源寄存器编号）	• 所有整型寄存器 • 同上带下标字母
Dest （传送目标寄存器编号）	• 整型寄存器 (#、C 寄存器除外) • 同上带下标字母
Byte Width （传送字节数）	• 所有整型寄存器 • 同上带下标字母 • 常数

■ 程序举例

将从 MW00100 开始的 5 字节展开成 MW00200 开始的 5 个字。

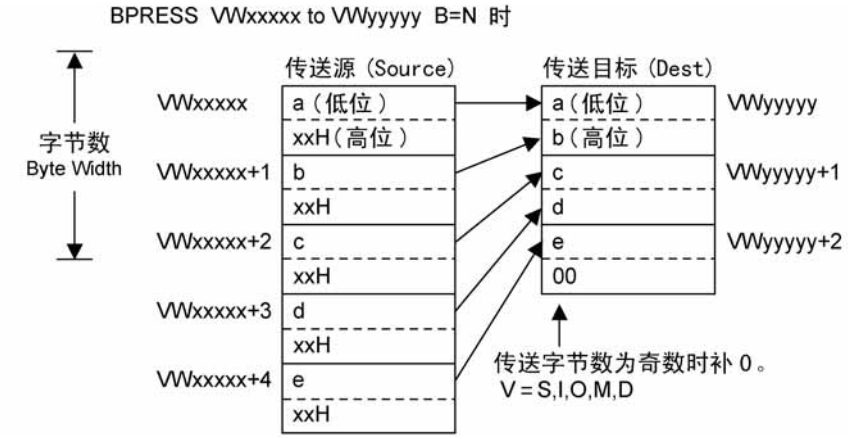




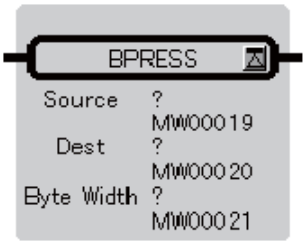
1.6.8 字→字节压缩指令（BPRESS）

■ 概要

将存储在传送源寄存器域（Source）内的字列的低位字节逐个字节地存储到传送目标寄存器（Dest）的字节列中。传送源寄存器的高位字节将被忽略。进行与 BEXTD 指令相反的动作。



■ 格式



标记：BPRESS

全称：Compress Word to Byte

类别：MOVE

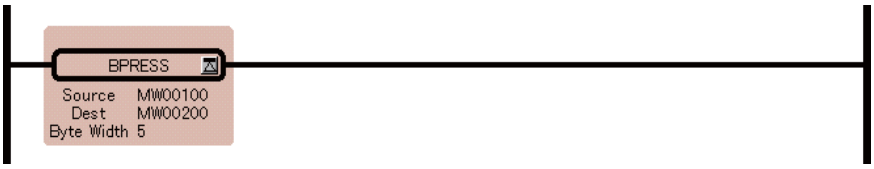
图标：

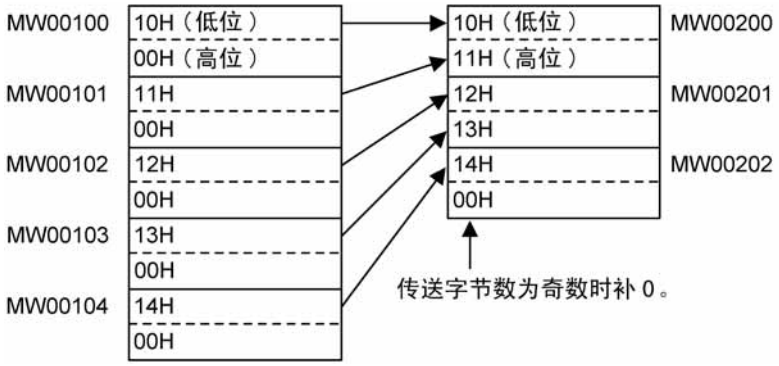
■ 参数

参数名称	设定
Source （传送源寄存器编号）	• 所有整型寄存器 • 同上带下标字母
Dest （传送目标寄存器编号）	• 整型寄存器（#、C 寄存器除外） • 同上带下标字母
Byte Width （传送字节数）	• 所有整型寄存器 • 同上带下标字母 • 常数

■ 程序举例

将从 MW00100 开始的 5 个字压缩成 MW00200 开始的 5 个字节。





1.6.9 数据检索指令 (BSRCH)

■ 概要

从指定的检索范围 (Source) 中用二进制检索方法检索指定数据 (Search Data)。将检索结果(从与其一致的数据检索范围首寄存器编号开始的偏置编号)存储到Result中。

■ 格式

标记：BSRCH

全称：Binary Data Search

类别：MOVE

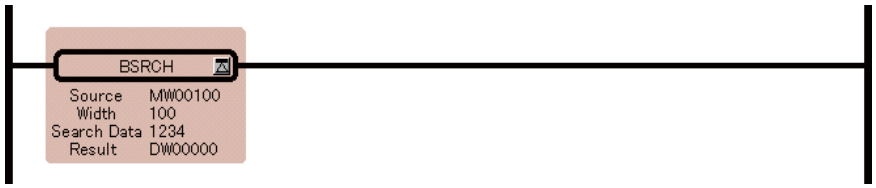
图标： 

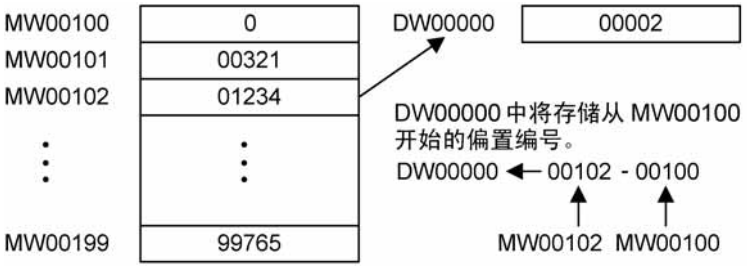
■ 参数

参数名称	设定
Source （检索范围首编号）	• 所有整型、长整型寄存器 • 同上带下标字母
Width （范围字符数）	• 所有整型、长整型寄存器 • 同上带下标字母
Search Data （检索数据）	• 所有整型、长整型寄存器 • 同上带下标字母 • 常数
Result （检索结果）	• 整型寄存器 (#、C 寄存器除外） • 同上带下标字母

■ 程序举例

以 MW00100 ～ MW00199 的寄存器为对象，检索与 01234 一致的数据，将其结果存储到 DW00000 中。



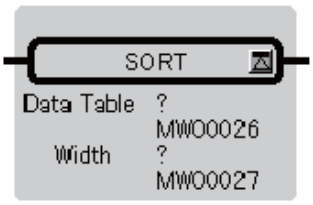


1.6.10 分类指令（SORT）

■ 概要

在指定的寄存器范围（Data Table，Width）内，按升序方式对数据进行分类。

■ 格式



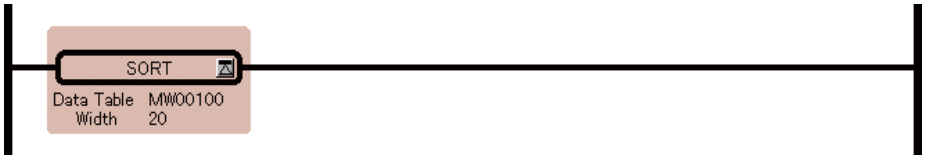
标记：SORT
全称：Sort
类别：MOVE
图标：

■ 参数

参数名称	设定
Data Table （分类范围首编号）	• 整型、长整型寄存器（#、C 寄存器除外） • 同上带下标字母
Width （范围寄存器数）	• 所有整型寄存器 • 同上带下标字母 • 常数

■ 程序举例

以 MW00100 ～ MW00119 的寄存器为对象，按升序方式对数据进行分类。

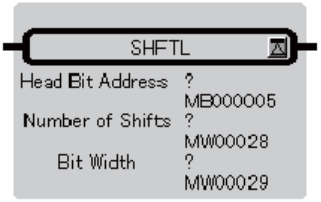


1.6.11　位左移指令（SHFTL）

■ 概要

对由位首地址（Head Bit Address）和位宽（Bit Width）指定的位列，按指定的移动次数（Number of Shifts）进行左移。
超出位宽的数据被舍去，不足的位为 0。

■ 格式



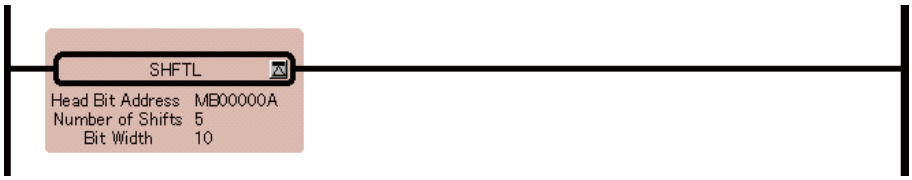
标记：SHFTL
全称：Bit Shift Left
类别：MOVE
图标： 

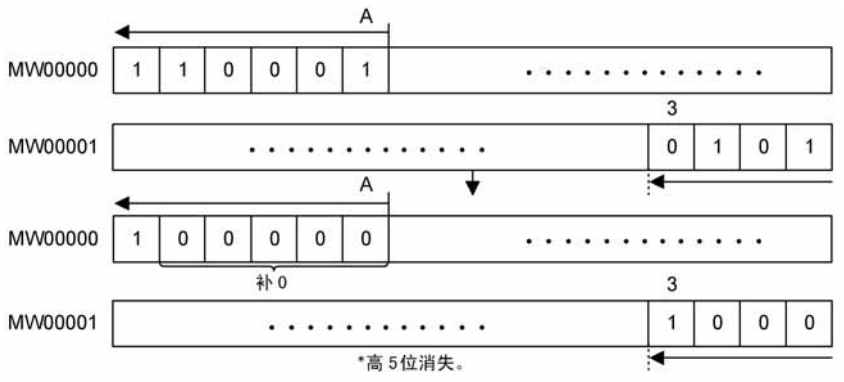
■ 参数

参数名称	设定
Head Bit Address（位首地址）	• 比特型寄存器（#、C 寄存器除外） • 同上带下标字母
Number of Shifts（移动次数）	• 所有整型寄存器 • 同上带下标字母 • 常数
Bit Width（位宽）	• 所有整型寄存器 • 同上带下标字母 • 常数

■ 程序举例

将从 SHFTL MB00000A(MW00000 的位 A) 开始的位宽为 10 的数据左移 5 位。



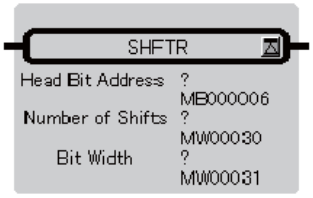


1.6.12　位右移指令（SHFTR）

■ 概要

对由位首地址（Head Bit Address）和位宽（Bit Width）指定的位列，按指定的移动次数（Number of Shifts）进行右移。
超出位宽的数据被舍去，不足的位为 0。

■ 格式



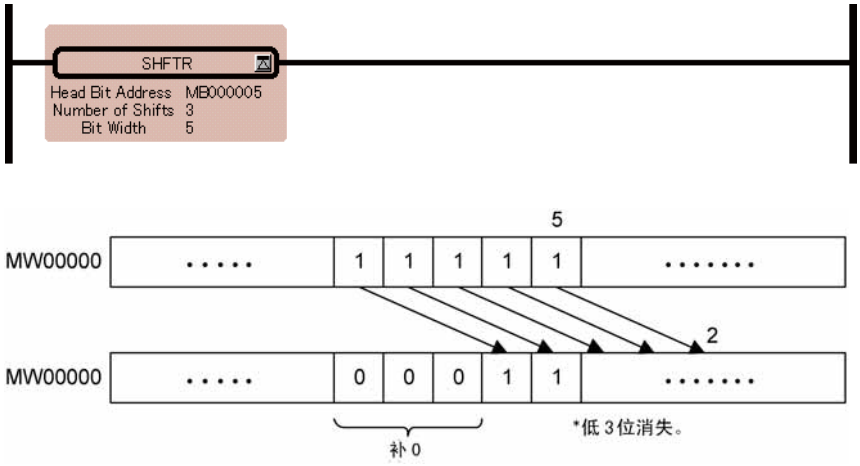
标记：SHFTR
全称：Bit Shift Right
类别：MOVE
图标： 

■ 参数

参数名称	设定
Head Bit Address（位首地址）	• 比特型寄存器（#、C 寄存器除外） • 同上带下标字母
Number of Shifts（移动次数）	• 所有整型寄存器 • 同上带下标字母 • 常数
Bit Width（位宽）	• 所有整型寄存器 • 同上带下标字母 • 常数

■ 程序举例

将从 SHFTR MB000005(MW00000 的位 5) 开始的位宽为 5 的数据右移 3 位。



1.6.13 字复制指令（COPYW）

■ 概要

从传送源寄存器首地址（Source）向传送目标寄存器首地址（Dest）传送相应字数（Width）的字数据。传送作业是从传送源向传送目标通过数据块复制的。传送源和传送目标即使重叠，传送的数据块也会被直接复制到传送目标中。

■ 格式

标记：COPYW

全称：Copy Word

类别：MOVE

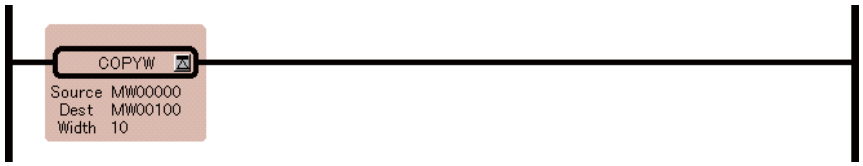
图标： 

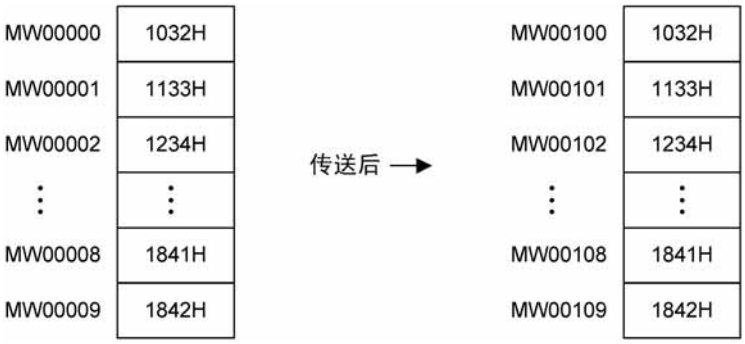
■ 参数

参数名称	设定
Source	• 所有整型寄存器 • 同上带下标字母
Dest	• 整型寄存器（#、C 寄存器除外） • 同上带下标字母
Width	• 所有整型寄存器 • 同上带下标字母 • 常数

■ 程序举例

向 MW00100 ～ MW00109 传送 MW00000 ～ MW00009。

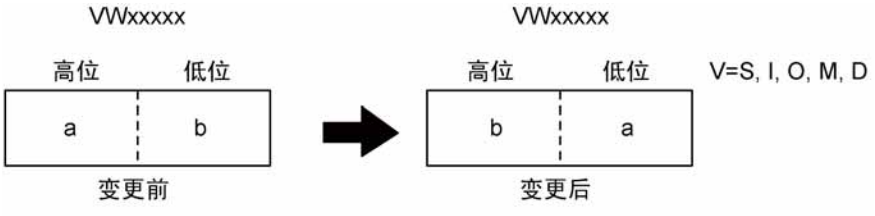




1.6.14 字节交换指令（BSWAP）

■ 概要

交换指定寄存器（Dest）的高位字节和低位字节。



■ 格式

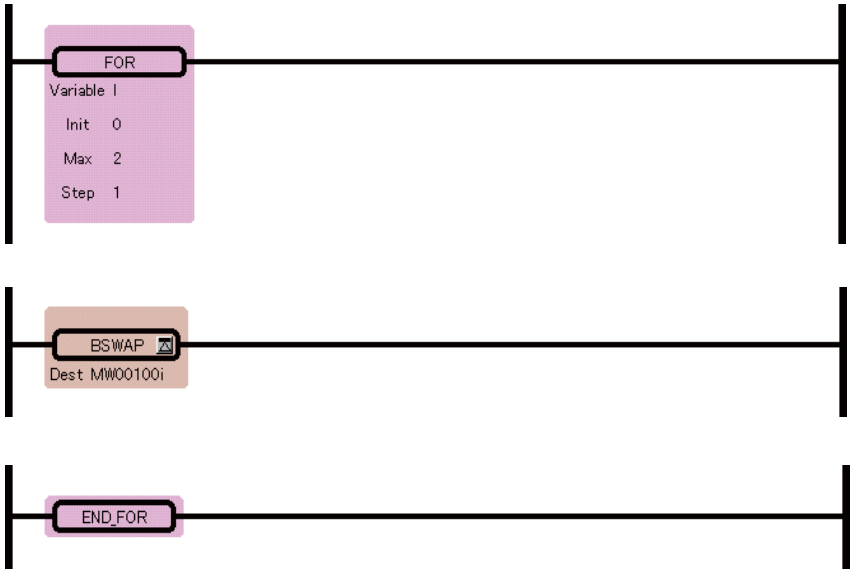
标记：BSWAP
全称：Byte Swap
类别：MOVE
图标：

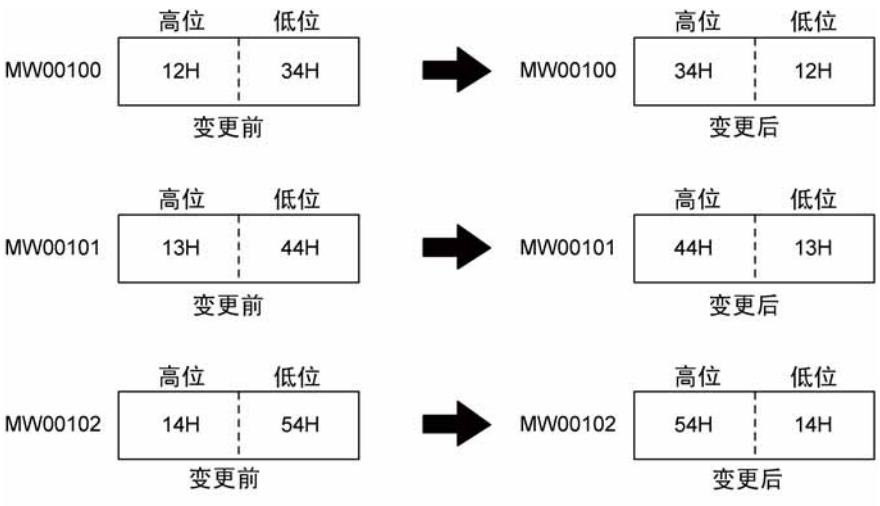
■ 参数

参数名称	设定
Dest （对象寄存器）	• 整型寄存器（#、C 寄存器除外） • 同上带下标字母

■ 程序举例

交换 MW00100 ～ MW00102 的高位字节和低位字节。




高位
低位
MW00100
12H
34H
变更前
MW00100
34H
12H
变更后
MW00101
13H
44H
变更前
MW00101
44H
13H
变更后
MW00102
14H
54H
变更前
MW00102
54H
14H
变更后

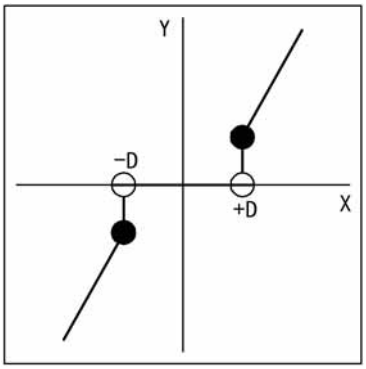
1.7 DDC 指令

1.7.1 死区 A 指令 (DZA)

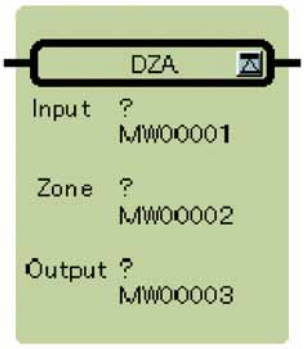
■ 概要

执行整型、长整型、实型的死区运算。以输入值为 Input、死区设定值为 Zone、输出值为 Output，则进行以下运算。

- Output ＝ Input （|Input| ≧ |Zone|）
- Output ＝ 0 （|Input| ＜ |Zone|）



■ 格式



标记：DZA
全称：Dead Zone A
类别：DDC
图标：

■ 参数

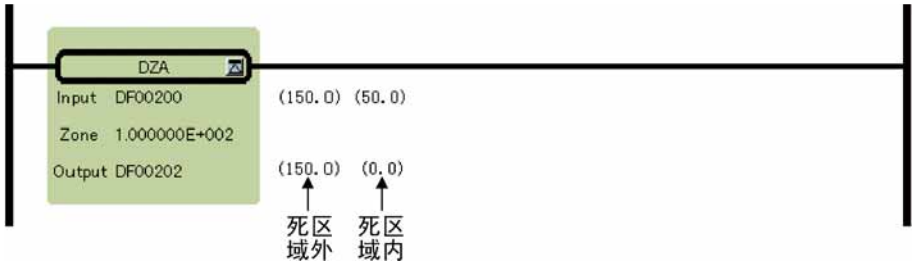
参数名称	设定
Input （输入值）	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Zone （死区设定值）	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Output （输出值）	• 所有整型、长整型、实型寄存器（#、C 寄存器除外） • 同上带下标字母 • 下标寄存器

■ 程序举例

整型运算时

长整型运算时

实型运算时

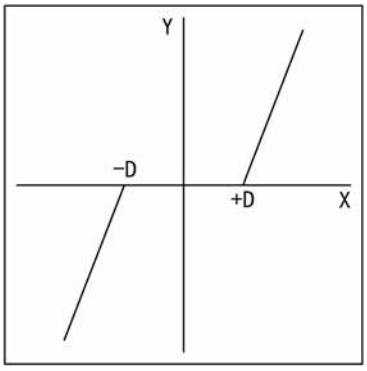


1.7.2 死区 B 指令 (DZB)

■ 概要

执行整型、长整型、实型的死区运算。以输入值为 Input、死区设定值为 Zone、输出值为 Output，则进行以下运算。

- Output ＝ Input － |Zone| (|Input| ≧ |Zone|, Input ≧ 0)
- Output ＝ Input ＋ |Zone| (|Input| ≧ |Zone|, Input ≦ 0)
- Output ＝ 0 (|Input| ＜ |Zone|)



■ 格式

标记：DZB
全称：Dead Zone B
类别：DDC
图标：

■ 参数

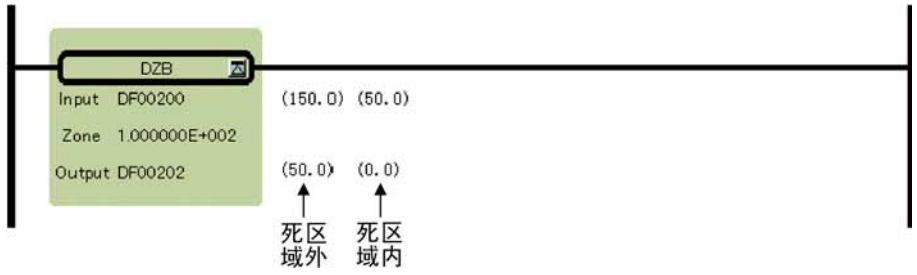
参数名称	设定
Input （输入值）	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Zone （死区设定值）	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Output （输出值）	• 所有整型、长整型、实型寄存器（#、C 寄存器除外） • 同上带下标字母 • 下标寄存器

■ 程序举例

整型运算时

长整型运算时

实型运算时

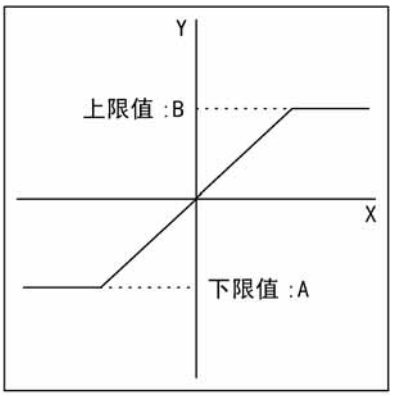


1.7.3 上下限值指令 (LIMIT)

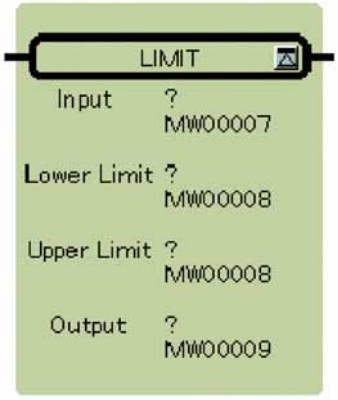
■ 概要

LIMIT 指令执行整型、长整型、实型的上下限值运算。以输入值为 Input、下限值为 Lower Limit、上限值为 Upper Limit、输出值为 Output，则执行以下运算。

- Output ＝ Lower Limit (Input ＜ Lower Limit)
- Output ＝ Input (Lower Limit ≦ Input ≦ Upper Limit)
- Output ＝ Upper Limit (Upper Limit ＜ Input)



■ 格式



标记：LIMIT

全称：Limit

类别：DDC

图标： 

■ 参数

参数名称	设定
Input （输入值）	所有整型、长整型、实型寄存器 同上带下标字母 下标寄存器 常数
Lower Limit （下限值）	所有整型、长整型、实型寄存器 同上带下标字母 下标寄存器 常数
Upper Limit （上限值）	所有整型、长整型、实型寄存器 同上带下标字母 下标寄存器 常数
Output （输出值）	整型、长整型寄存器（#、C 寄存器除外） 同上带下标字母 下标寄存器

■ 程序举例

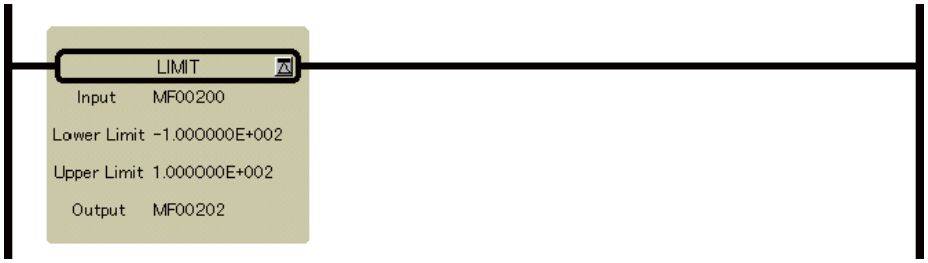
整型运算时

输入（MW00100）	输出（MW00102）
-100 ＞ MW00100	-00100（下限值超限）
-100 ≦ MW00100 ≦ 100	MW00100 的值（在上下限值内）
MW00100 ＞ 100	00100（上限值超限）

长整型运算时

输入（ML00100）	输出（ML00102）
-100000 ＞ ML00100	-100000（下限值超限）
-100000 ≦ ML00100 ≦ 100000	ML00100 的值（在上下限值内）
ML00100 ＞ 100000	100000（上限值超限）

实型运算时



输入（DF00200）	输出（DF00202）
-100.0 ＞ DF00200	-100.0（下限值超限）
-100.0 ≦ DF00200 ≦ 100.0	DF00200 的值（在上下限值内）
DF00200 ＞ 100.0	100.0（上限值超限）

1.7.4　PI 控制指令（PI）

■ 概要

PI 指令按照事先设定的参数表内容执行 PI 运算。PI 运算的输入（Input）可以使用整型或实型。不能使用长整型数据。整型和实型的参数表构成不同。PI 补偿值被输出到 Output 中。

表 1.12　整型 PI 指令参数表

ADR	类型	符号	名称	规格	输入输出
0	W	RLY	继电器输入输出	继电器输入、继电器输出*	IN/OUT
1	W	Kp	P 增益	P 补偿增益（增益 1 倍时为 100）	IN
2	W	Ki	积分调整增益	积分回路输入的增益（增益 1 倍时为 100）	IN
3	W	Ti	积分时间	积分时间（ms）	IN
4	W	IUL	积分上限 LIMIT	相对于 I 补偿值的上限值	IN
5	W	ILL	积分下限 LIMIT	相对于 I 补偿值的下限值	IN
6	W	UL	PI 上限 LIMIT	相对于 P ＋ I 补偿值的上限值	IN
7	W	LL	PI 下限 LIMIT	相对于 P ＋ I 补偿值的下限值	IN
8	W	DB	PI 输出死区	相对于 P ＋ I 补偿值的死区宽度	IN
9	W	Y	PI 输出	PI 补偿输出（向 Output 输出）	OUT
10	W	Yi	I 补偿值	I 补偿值保存	OUT
11	W	IREM	I 余数	I 余数保存	OUT

* 继电器输入输出位的分配如下。

BIT	符号	名称	规格	输入输出
0	IRST	积分复位	积分复位时输入 “闭”	IN
1 ～ 7		（备用）	输入用备用继电器	IN
8 ～ F		（备用）	输出用备用继电器	OUT

表 1.13　实型 PI 指令参数表

ADR	类型	符号	名称	规格	输入输出
0	W	RLY	继电器输入输出	继电器输入、继电器输出*	IN/OUT
1	W	—	（备用）	备用寄存器	—
2	F	Kp	P 增益	P 补偿增益	IN
4	F	Ki	积分调整增益	积分回路输入的增益	IN
6	F	Ti	积分时间	积分时间（s）	IN
8	F	IUL	积分上限 LIMIT	相对于 I 补偿值的上限值	IN
10	F	ILL	积分下限 LIMIT	相对于 I 补偿值的下限值	IN
12	F	UL	PI 上限 LIMIT	相对于 P ＋ I 补偿值的上限值	IN
14	F	LL	PI 下限 LIMIT	相对于 P ＋ I 补偿值的下限值	IN
16	F	DB	PI 输出死区	相对于 P ＋ I 补偿值的死区宽度	IN
18	F	Y	PI 输出	PI 补偿输出（向 Output 输出）	OUT
20	F	Yi	I 补偿值	I 补偿值保存	OUT

* 继电器输入输出位的分配如下。

BIT	符号	名称	规格	输入输出
0	IRST	积分复位	积分复位时输入 “闭”	IN
1 ～ 7	—	（备用）	输入用备用继电器	IN
8 ～ F	—	（备用）	输出用备用继电器	OUT

此处的 PI 运算如下所示。

$$\frac{Y}{X} = Kp + Ki \times \frac{1}{Ti \times S}$$

X：偏差输入值

Y：输出值

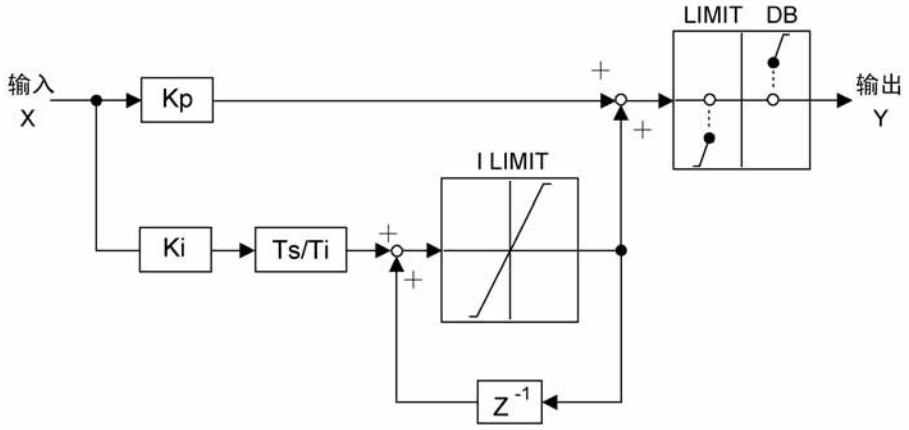
在 PI 指令内部按以下计算式进行运算。

$$Y = Kp \times X + \left\{ (Ki \times X + IREM) / \frac{Ti}{Ts} + Yi' \right\}$$

Yi’：上次 I 输出值

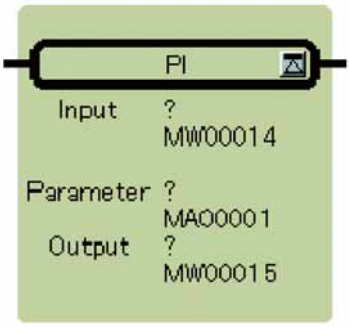
Ts ：扫描时间设定值

表 1.14 方块图



- PI 上下限值 (UL，LL) 运算
  PI 上下限值 (UL，LL) 是对 P 补偿量和 I 补偿量的合成值的限值，当 P 补偿量超过 PI 上下限值 (UL，LL) 时，P 补偿量为 PI 上下限值 (UL，LL)。此后，进行 PI 上下限值 (UL，LL) 检查运算。
- 在 P ＋ I 补偿量为 PI 上下限值 (UL，LL)，或属于 PI 死区 (DB) 时，当此次的 P 补偿量和 I 补偿量为同符号（发散方向）时，I 补偿量不进行更新，为上次的值。反之，当为异符号（向 0 收敛方向）时，I 补偿量更新为本次数值。
- 积分复位 (IRST) 为 “闭” 时
  输出 Ti ＝ 0、IREM ＝ 0。

■ 格式



标记：PI
全称：PI Control
类别：DDC
图标： 

■ 参数

参数名称	设定
Input （输入值）	• 所有整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Parameter （参数表首地址）	• 寄存器地址（#、C 寄存器除外） • 同上带下标字母
Output （输出值）	• 整型、实型寄存器（#、C 寄存器除外） • 同上带下标字母 • 下标寄存器

■ 程序举例

整型运算时

将 MW00100 ～ MW00111 作为参数表使用。

实型运算时

将 MF00200 ～ MF00220 作为参数表使用。

1.7.5 PD 控制指令 (PD)

■ 概要

PD 指令按照事先设定的参数表内容执行 PD 运算。PD 运算的输入 (Input) 可以使用整型或实型。不能使用长整型数据。整型和实型的参数表构造不同。PD 补偿值被输出到 Output 中。

表 1.15 整型 PD 指令参数表

ADR	类型	符号	名称	规格	输入输出
0	W	RLY	继电器输入输出	继电器输入、继电器输出 *	IN/OUT
1	W	Kp	P 增益	P 补偿增益 (增益 1 倍时为 100)	IN
2	W	Kd	D 增益	微分回路输入的增益 (增益 1 倍时为 100)	IN
3	W	Td1	发散侧微分时间	输入为发散方向时用的微分时间	IN
4	W	Td2	收敛侧微分时间	输入为收敛方向时用的微分时间	IN
5	W	UL	PD 上限 LIMIT	相对于 P ＋ D 补偿值的上限值	IN
6	W	LL	PD 下限 LIMIT	相对于 P ＋ D 补偿值的下限值	IN
7	W	DB	PD 输出死区	相对于 P ＋ D 补偿值的死区宽度	IN
8	W	Y	PD 输出	PD 补偿输出 (向 Output 输出)	OUT
9	W	X	输入值保存	本次偏差输入值保存	OUT

* 继电器输入输出位的分配如下。

BIT	符号	名称	规格	输入输出
1 ～ 7	—	(备用)	输入用备用继电器	IN
8 ～ F	—	(备用)	输出用备用继电器	OUT

表 1.16 实型 PD 指令参数表

ADR	类型	符号	名称	规格	输入输出
0	W	RLY	继电器输入输出	继电器输入、继电器输出 *	IN/OUT
1	W	—	(备用)	备用寄存器	—
2	F	Kp	P 增益	P 补偿增益	IN
4	F	Kd	D 增益	微分回路输入的增益 (增益 1 倍时为 100)	IN
6	F	Td1	发散侧微分时间	输入为发散方向时用的微分时间	IN
8	F	Td2	收敛侧微分时间	输入为收敛方向时用的微分时间	IN
10	F	UL	PD 上限 LIMIT	相对于 P ＋ D 补偿值的上限值	IN
12	F	LL	PD 下限 LIMIT	相对于 P ＋ D 补偿值的下限值	IN
14	F	DB	PD 输出死区	相对于 P ＋ D 补偿值的死区宽度	IN
16	F	Y	PD 输出	PD 补偿输出 (向 Output 输出)	OUT
18	F	X	输入值保存	本次偏差输入值保存	OUT

* 继电器输入输出位的分配如下。

BIT	符号	名称	规格	输入输出
0 ～ 7	—	(备用)	输入用备用继电器	IN
8 ～ F	—	(备用)	输出用备用继电器	OUT

此处的 PD 运算表示如下。

$$\frac{Y}{X} = Kp + Kd \times Td \times S$$

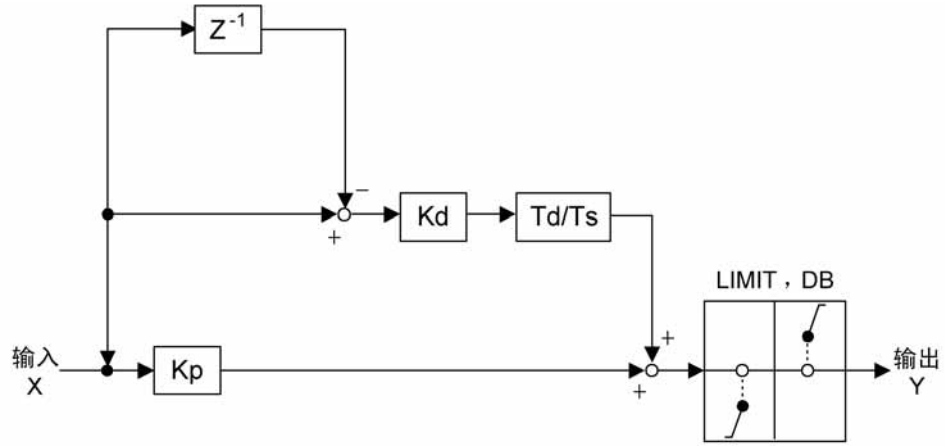
X：偏差输入值
Y：输出值

在 PD 指令内部按下式进行运算。

$$Y = Kp \times X + Kd \times (X - X') \times \frac{Td}{Ts}$$

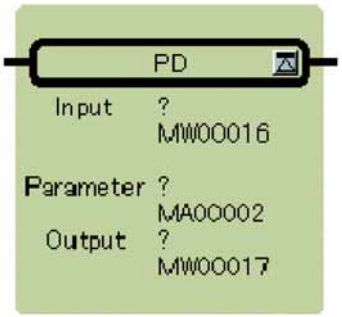
X'：上次输入值
Ts：扫描时间设定值

方块图



- PD 上下限值 (UL，LL) 运算
  PD 上下限值 (UL,LL) 是对 P 补偿量和 D 补偿量的合成量的限值，当 P 补偿量超过 PD 上下限值 (UL,LL) 时，P 补偿量为 PD 上下限值 (UL,LL)。此后，进行 PD 上下限值 (UL，LL) 检查运算。
- 微分 (D) 运算中，当偏差输入变化量 (X-X') 和上次偏差输入 (X') 同符号 (发散方向) 时将发散侧微分时间 (Td1) 作为微分时间使用。
- 微分 (D) 运算中，当偏差输入变化量 (X-X') 和上次偏差输入 (X') 异符号 (收敛方向) 时将收敛侧微分时间 (Td2) 作为微分时间使用。

■ 格式



标记：PD
全称：PD Control
类别：DDC
图标： 

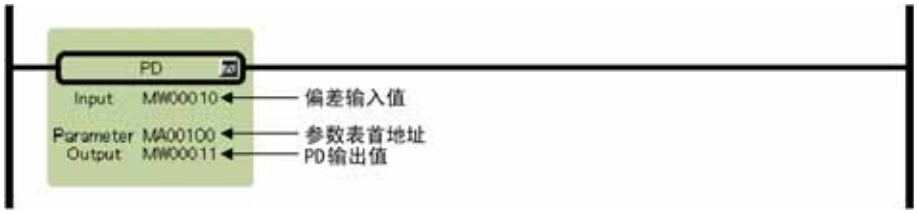
■ 参数

参数名称	设定
Input （输入值）	• 所有整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Parameter （参数表首地址）	• 寄存器地址（#、C 寄存器除外） • 同上带下标字母
Output （输出值）	• 整型、实型寄存器（#、C 寄存器除外） • 同上带下标字母 • 下标寄存器

■ 程序举例

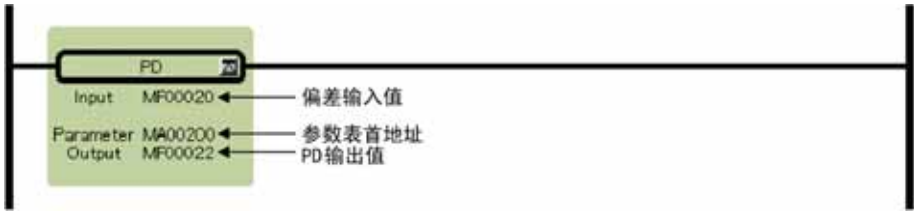
整型运算时

将 MW00100 ～ MW00109 作为参数表使用。



实型运算时

将 MF00200 ～ MF00218 作为参数表使用。



1.7.6　PID 控制指令 (PID)

■ 概要

PID 指令按照事先设定的参数表内容，执行 PID 运算。对 PID 运算的输入 (Input) 可以使用整型或实型。不能使用长整型数据。整型和实型参数表构造是不同的。向 Output 输出 PID 补偿值。

表 1.17　整型 PID 指令参数表

ADR	类型	符号	名称	规格	输入输出
0	W	RLY	继电器输入输出	继电器输入、继电器输出*	IN/OUT
1	W	Kp	P 增益	P 补偿增益 (增益 1 倍时为 100)	IN
2	W	Ki	I 增益	积分回路输入的增益 (增益 1 倍时为 100)	IN
3	W	Kd	D 增益	微分回路输入的增益 (增益 1 倍时为 100)	IN
4	W	Ti	积分时间	积分时间 (ms)	IN
5	W	Td1	发散侧微分时间	输入为发散方向时用的微分时间	IN
6	W	Td2	收敛侧微分时间	输入为收敛方向时用的微分时间	IN
7	W	IUL	积分上限 LIMIT	相对于 I 补偿值的上限值	IN
8	W	ILL	积分下限 LIMIT	相对于 I 补偿值的下限值	IN
9	W	UL	PID 上限 LIMIT	相对于 P ＋ I ＋ D 补偿值的上限值	IN
10	W	LL	PID 下限 LIMIT	相对于 P ＋ I ＋ D 补偿值的下限值	IN
11	W	DB	PID 输出死区	相对于 P ＋ I ＋ D 补偿值的死区宽度	IN
12	W	Y	PID 输出	PID 补偿输出 (向 Output 输出)	OUT
13	W	Yi	I 补偿值	I 补偿值保存	OUT
14	W	IREM	I 余数	I 余数保存	OUT
15	W	X	输入值保存	本次偏差输入值保存	OUT

* 继电器输入输出位的分配如下。

BIT	符号	名称	规格	输入输出
0	IRST	积分复位	积分复位时输入 “闭”	IN
1 ～ 7	—	(备用)	输入用备用继电器	IN
8 ～ F	—	(备用)	输出用备用继电器	OUT

表 1.18 实型 PID 指令参数表

ADR	类型	符号	名称	规格	输入输出
0	W	RLY	继电器输入输出	继电器输入、继电器输出*	IN/OUT
1	W	—	（备用）	备用寄存器	—
2	F	Kp	P 增益	P 补偿增益	IN
4	F	Ki	I 增益	积分回路输入的增益（增益 1 倍时为 100）	IN
6	F	Kd	D 增益	微分回路输入的增益（增益 1 倍时为 100）	IN
8	F	Ti	积分时间	积分时间（s）	IN
10	F	Td1	发散侧微分时间	输入为发散方向时用的微分时间	IN
12	F	Td2	收敛侧微分时间	输入为收敛方向时用的微分时间	IN
14	F	IUL	积分上限 LIMIT	相对于 I 补偿值的上限值	IN
16	F	ILL	积分下限 LIMIT	相对于 I 补偿值的下限值	IN
18	F	UL	PID 上限 LIMIT	相对于 P ＋ I ＋ D 补偿值的上限值	IN
20	F	LL	PID 下限 LIMIT	相对于 P ＋ I ＋ D 补偿值的下限值	IN
22	F	DB	PID 输出死区	相对于 P ＋ I ＋ D 补偿值的死区宽度	IN
24	F	Y	PID 输出	PID 补偿输出（向 Output 输出）	OUT
26	F	Yi	I 补偿值	I 补偿值保存	OUT
28	F	X	输入值保存	本次偏差输入值保存	OUT

* 继电器输入输出位的分配如下。

BIT	符号	名称	规格	输入输出
0	IRST	积分复位	积分复位时输入 “闭”	IN
1 ～ 7	—	（备用）	输入用备用继电器	IN
8 ～ F	—	（备用）	输出用备用继电器	OUT

此处的PID运算如下所示。

$$\frac{Y}{X} = Kp + Ki \times \frac{1}{Ti \times S} + Kd \times Td \times S$$

X：偏差输入值

Y：输出值

在PID指令内部按下式进行运算。

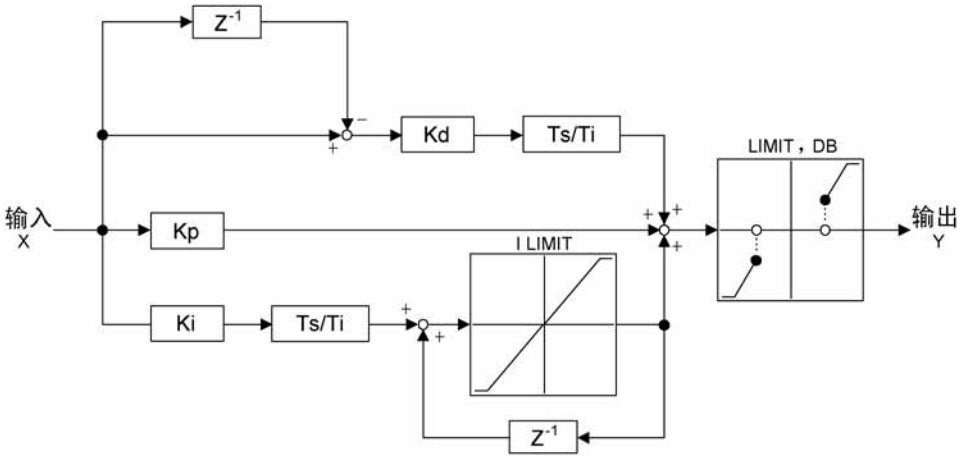
$$Y = Kp \times X + \left\{ (Ki \times X + IREM) / \frac{Ti}{Ts} + Yi' \right\} + Kd \times (X - X') \times \frac{Td}{Ts}$$

X’：上次输入值

Yi’：上次I输出值

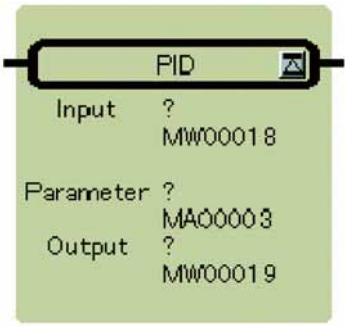
Ts：扫描时间设定值

方块图



- PID上下限值（UL，LL）运算
  PID上下限值（UL,LL）是对P补偿量和I补偿量的合成量的限值，当P补偿量超过PID上下限值（UL,LL）时，P补偿量为PID上下限值（UL,LL）。此后，进行PID上下限值（UL，LL）检查运算。
- 在P＋I＋D补偿量为PID上下限值（UL，LL），或属于PID死区（DB）时
  当此次的P补偿量和I补偿量为同符号（发散方向）时，I补偿量不进行更新，为上次值。反之，当为异符号（向0收敛方向）时，I补偿量更新为本次数值。
- 微分（D）运算中，当偏差输入变化量（X-X’）和上次偏差输入（X’）同符号（发散方向）时，将发散侧微分时间（Td1）作为微分时间使用。
- 微分（D）运算中，当偏差输入变化量（X-X’）和上次偏差输入（X’）异符号（收敛方向）时，将收敛侧微分时间（Td2）作为微分时间使用。
- 积分复位（IRST）为“闭”时
  输出Yi＝0、IREM＝0。

■ 格式



标记：PID

全称：PID Control

类别：DDC

图标： 

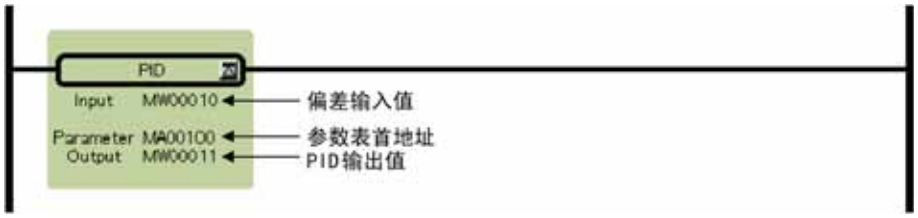
■ 参数

参数名称	设定
Input （输入值）	• 所有整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Parameter （参数表首地址）	• 寄存器地址（#、C 寄存器除外） • 同上带下标字母
Output （输出值）	• 整型、实型寄存器（#、C 寄存器除外） • 同上带下标字母 • 下标寄存器

■ 程序举例

整型运算时

将 MW00100 ～ MW00115 作为参数表使用。



实型运算时

将 MF00200 ～ MF00228 作为参数表使用。

1.7.7　一阶延迟指令（LAG）

■ 概要

LAG 指令按照事先设定的参数表内容执行一阶延迟运算。LAG 运算的输入（Input）可以使用整型或实型。不能使用长整型数据。整型和实型参数表构造不同。一阶延迟值被输出到 Output 中。

整型 LAG 指令参数表

ADR	类型	符号	名称	规格	输入输出
0	W	RLY	继电器输入输出	继电器输入、继电器输出*	IN/OUT
1	W	T	一阶延迟时间常数	一阶延迟时间常数（ms）	IN
2	W	Y	LAG 输出	LAG 输出（向 Output 输出）	OUT
3	W	REM	余数	余数保存	OUT

* 继电器输入输出位的分配如下。

BIT	符号	名称	规格	输入输出
0	IRST	LAG 复位	LAG 复位时输入“闭”	IN
1～7	—	（备用）	输入用备用继电器	IN
8～F	—	（备用）	输出用备用继电器	OUT

实型 LAG 指令参数表

ADR	类型	符号	名称	规格	输入输出
0	W	RLY	继电器输入输出	继电器输入、继电器输出*	IN/OUT
1	W	—	（备用）	备用寄存器	—
2	F	T	一阶延迟时间常数	一阶延迟时间常数（s）	IN
4	F	Y	LAG 输出	LAG 输出（向 Output 输出）	OUT

* 继电器输入输出位的分配如下。

BIT	符号	名称	规格	输入输出
0	IRST	LAG 复位	LAG 复位时输入“闭”	IN
1～7	—	（备用）	输入用备用继电器	IN
8～F	—	（备用）	输出用备用继电器	OUT

此处的 LAG 运算如下所示。

$$\frac{Y}{X} = \frac{1}{1+T\times S}$$

即

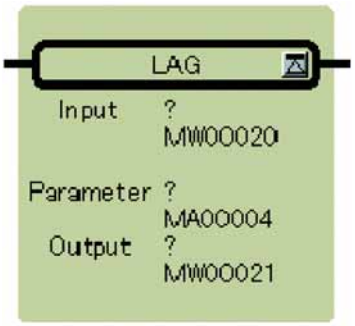
$$T\times\frac{dY}{dt}+Y = X$$

在 LAG 指令内部，假设 dt ＝ Ts、dY ＝ Y-Y'，进行以下运算。

$$Y = \frac{T\times Y'+Ts\times X+REM}{T+Ts}$$

X：输入值
Y：输出值
Y'：上次输出值
Ts：扫描时间设定值
LAG 复位（IRST）为 “闭”时，输出 Y ＝ 0、REM ＝ 0。

■ 格式



标记：LAG
全称：First Order Lag
类别：DDC
图标：

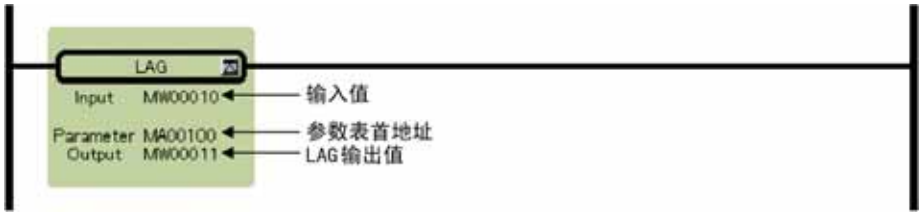
■ 参数

参数名称	设定
Input （输入值）	• 所有整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Parameter （参数表首地址）	• 寄存器地址（#、C 寄存器除外） • 同上带下标字母
Output （输出值）	• 整型、实型寄存器（#、C 寄存器除外） • 同上带下标字母 • 下标寄存器

■ 程序举例

整型运算时

将 MW00100 ～ MW00103 作为参数表使用。



实型运算时

将 MF00200 ～ MF00204 作为参数表使用。

1.7.8 相位超前滞后指令 (LLAG)

■ 概要

LLAG 指令按照事先设定的参数表内容执行相位超前滞后运算。LLAG 运算的输入(Input)可以使用整型或实型。不能使用长整型数据。整型和实型的参数表构造不同。相位超前滞后值被输出到 Output 中。

整型 LLAG 指令参数表

ADR	类型	符号	名称	规格	输入输出
0	W	RLY	继电器输入输出	继电器输入、继电器输出*	IN/OUT
1	W	T2	相位超前时间常数	相位超前时间常数 (ms)	IN
2	W	T1	相位滞后时间常数	相位滞后时间常数 (ms)	IN
3	W	Y	LLAG 输出	LLAG 输出 (向 Output 输出)	OUT
4	W	REM	余数	余数保存	OUT
5	W	X	输入值保存	输入值保存	OUT

* 继电器输入输出位的分配如下。

BIT	符号	名称	规格	输入输出
0	IRST	LLAG 复位	LLAG 复位时输入 “闭”	IN
1 ～ 7	—	(备用)	输入用备用继电器	IN
8 ～ F	—	(备用)	输出用备用继电器	OUT

实型 LLAG 指令参数表

ADR	类型	符号	名称	规格	输入输出
0	W	RLY	继电器输入输出	继电器输入、继电器输出*	IN/OUT
1	W	—	(备用)	备用寄存器	—
2	F	T2	相位超前时间常数	相位超前时间常数 (ms)	IN
4	F	T1	相位滞后时间常数	相位滞后时间常数 (ms)	IN
6	F	Y	LLAG 输出	LLAG 输出 (向 Output 输出)	OUT
8	F	X	输入值保存	输入值保存	OUT

* 继电器输入输出位的分配如下。

BIT	符号	名称	规格	输入输出
0	IRST	LLAG 复位	LLAG 复位时输入 “闭”	IN
1 ～ 7	—	(备用)	输入用备用继电器	IN
8 ～ F	—	(备用)	输出用备用继电器	OUT

此处的 LLAG 运算如下所示。

$$\frac{Y}{X} = \frac{1 + T2 \times S}{1 + T1 \times S}$$

即

$$T \times \frac{dY}{dt} + Y = T2 \times \frac{dX}{dt} + X$$

在 LLAG 指令内部，假设 dt ＝ Ts、dY ＝ Y-Y’、dX ＝ X-X’，进行以下运算。

$$Y = \frac{T1 \times Y' + (T2 + Ts) \times X - T2 \times X' + REM}{T1 + Ts}$$

X：输入值
Y：输出值
X’：上次输入值
Y’：上次输出值
Ts：扫描时间设定值
LLAG 复位（IRST）为 “闭”时，输出 Y ＝ 0、REM ＝ 0、X ＝ 0。

■ 格式

标记：LLAG
全称：Phase Lead Lag
类别：DDC
图标：

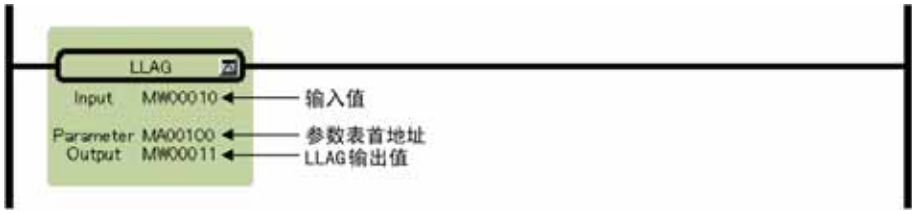
■ 参数

参数名称	设定
Input （输入值）	• 所有整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Parameter （参数表首地址）	• 寄存器地址（#、C 寄存器除外） • 同上带下标字母
Output （输出值）	• 整型、实型寄存器（#、C 寄存器除外） • 同上带下标字母 • 下标寄存器

■ 程序举例

整型运算时

将 MW00100 ～ MW00105 作为参数表使用。



实型运算时

将 MF00200 ～ MF00208 作为参数表使用。

1.7.9　函数发生器指令（FGN）

■ 概要

FGN 指令按照事先设定的参数表内容生成函数曲线。作为 FGN 指令的输入，可使用整型、长整型、实型，但其参数表构成不同。

表 1.19　整型 FGN 指令参数表

ADR	类型	符号	名称	规格	输入输出
0	W	N	数据数	X，Y 的组数	IN
1	W	X1	数据 1		IN
2	W	Y1	数据 1		IN
3	W	X2	数据 2		IN
4	W	Y2	数据 2		IN
・ ・ ・	・ ・ ・	・ ・ ・	・ ・ ・	・ ・ ・	・ ・ ・
2N-1	W	XN	数据 N		IN
2N	W	YN	数据 N		IN

表 1.20　长整型或实型 FGN 指令参数表

ADR	类型	符号	名称	规格	输入输出
0	W	N	数据数	X，Y 的组数	IN
1	W	—	（备用）	备用寄存器	IN
2	L/F	X1	数据 1		IN
4	L/F	Y1	数据 1		IN
6	L/F	X2	数据 2		IN
8	L/F	Y2	数据 2		IN
・ ・ ・	・ ・ ・	・ ・ ・	・ ・ ・	・ ・ ・	・ ・ ・
4N-2	L/F	XN	数据 N		IN
4N	L/F	YN	数据 N		IN

在 FGN 指令的参数表设定中，数据设为 X_n、Y_n 时，数据的设定要满足 $X_n \leqq X_{n+1}$。在执行 FGN 指令时，对于输入值 X，先检索参数表中满足 $X_n \leqq X \leqq X_{n+1}$ 的 X_n、Y_n 的组，然后通过下面的计算式计算输出值 Y。

$$Y = Y_n + \frac{Y_{n+1} - Y_n}{X_{n+1} - X_n} \times (X - X_n)(1 \leq n \leq N-1)$$

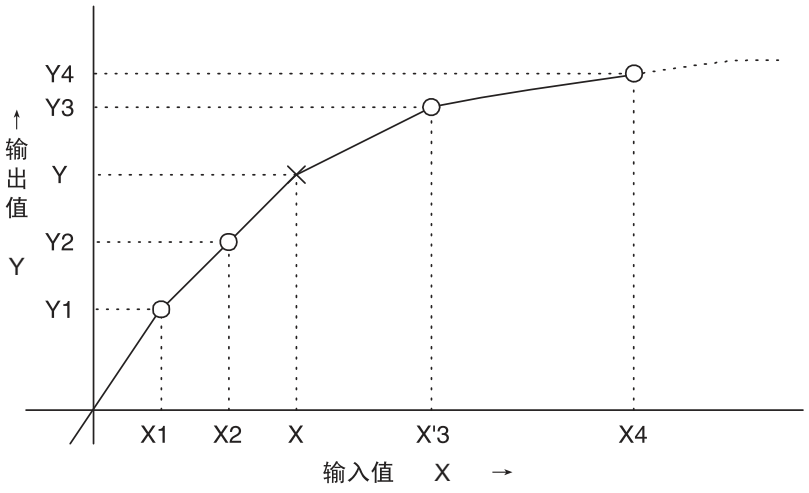
参数表设定数据与输入值 X、输出值 Y 的关系如下。对于输入值 X，当没有满足参数表中 $X_n \leqq X \leqq X_{n+1}$ 的 X_n、Y_n 的组时，按如下所示计算。

- $X < X_1$ 时

$$Y = Y_1 + \frac{Y_2 - Y_1}{X_2 - X_1}(X - X_1)$$

- $X > X_1$ 时

$$Y = Y_{n+1} + \frac{Y_n - Y_{n-1}}{X_n - X_{n-1}}(X - X_1)$$



■ 格式

标记：FGN
全称：Function Generator
类别：DDC
图标： 

■ 参数

参数名称	设定
Input （输入值）	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Parameter （参数表首地址）	• 寄存器地址 • 同上带下标字母
Output （输出值）	• 整型、长整型、实型寄存器 （#、C 寄存器除外） • 同上带下标字母 • 下标寄存器

■ 程序举例

整型运算时（数据数 N ＝ 20）

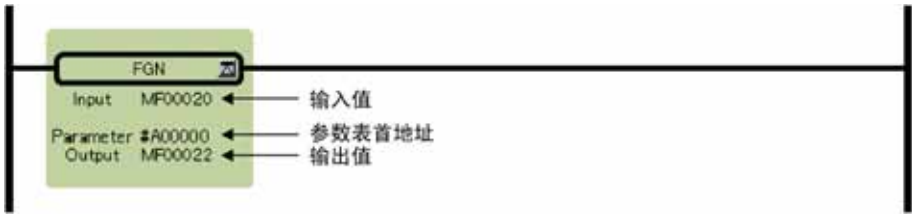
将 #W00000 ～ #W00040 作为参数表使用。

长整型运算时（数据数 N ＝ 20）

将 #L00000 ～ #L00080 作为参数表使用。

实型运算时

将 #F00000 ～ #F00008 作为参数表使用。



1.7.10　反函数发生器指令（IFGN）

■ 概要

IFGN 指令按照事先设定的参数表内容生成函数曲线。作为 IFGN 指令的输入，可使用整型、长整型、实型，但其参数表构成不同。参数表与 FGN 指令相同。

表 1.21　整型 IFGN 指令参数表

ADR	类型	符号	名称	规格	输入输出
0	W	N	数据数	X，Y 的组数	IN
1	W	X1	数据 1		IN
2	W	Y1	数据 1		IN
3	W	X2	数据 2		IN
4	W	Y2	数据 2		IN
• • •	• • •	• • •	• • •	• • •	• • •
2N-1	W	XN	数据 N		IN
2N	W	YN	数据 N		IN

表 1.22　长整型或实型 IFGN 指令参数表

ADR	类型	符号	名称	规格	输入输出
0	W	N	数据数	X，Y 的组数	IN
1	W	—	（备用）	备用寄存器	IN
2	L/F	X1	数据 1		IN
4	L/F	Y1	数据 1		IN
6	L/F	X2	数据 2		IN
8	L/F	Y2	数据 2		IN
• • •	• • •	• • •	• • •	• • •	• • •
4N-2	L/F	XN	数据 N		IN
4N	L/F	YN	数据 N		IN

在 IFGN 指令的参数表设定中，数据为 X_n、Y_n 时，数据要满足 $Y_n \leqq Y_{n+1}$。在执行 IFGN 指令时，对于输入值 Y，先检索参数表中满足 $Y_n \leqq Y \leqq Y_{n+1}$ 的 X_n、Y_n 的组，然后通过下面的计算式计算输出值 X。

$$X = X_n + \frac{X_{n+1} - X_n}{Y_{n+1} - Y_n} \times (Y - Y_n)(1 \leq n \leq N-1)$$

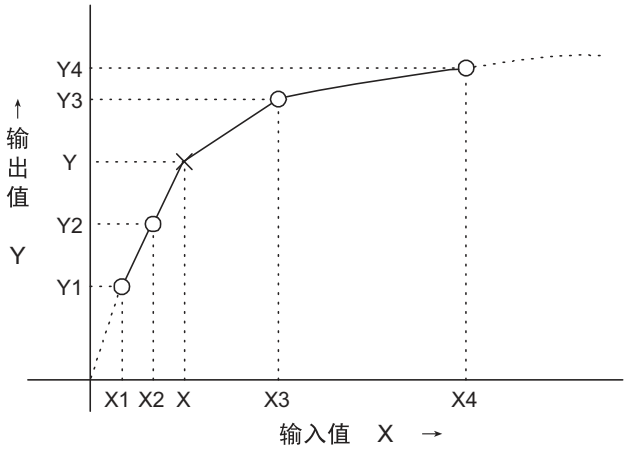
参数表设定数据与输入值 Y、输出值 X 的关系如下。对于输入值 Y，当没有满足参数表中 $Y_n \leqq Y \leqq Y_{n+1}$ 的 X_n、Y_n 的组时，按如下所示计算。

- $Y < Y_1$ 时

$$X = X_1 + \frac{X_2 - X_1}{Y_2 - Y_1}(Y - Y_1)$$

- $Y > Y_1$ 时

$$X = X_{n+1} + \frac{X_n - X_{n-1}}{Y_n - Y_{n-1}}(Y - Y_1)$$



■ 格式

标记：IFGN
全称：Inverse Function Generator
类别：DDC
图标：

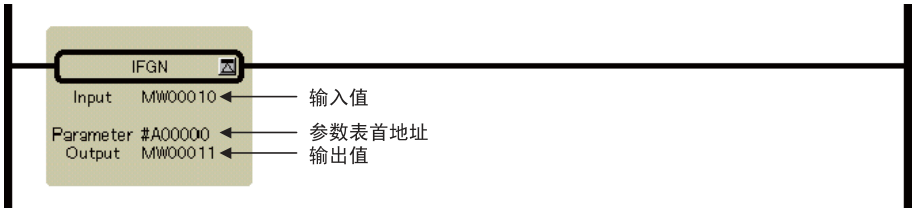
■ 参数

参数名称	设定
Input （输入值）	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Parameter （参数表首地址）	• 寄存器地址 • 同上带下标字母
Output （输出值）	• 整型、长整型、实型寄存器(#、C寄存器除外) • 同上带下标字母 • 下标寄存器

■ 程序举例

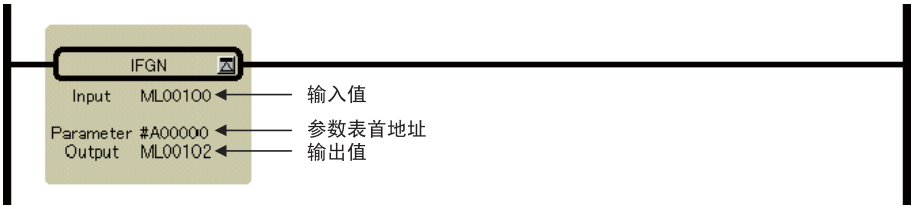
整型运算时（数据数 N ＝ 20）

将 #W00000 ～ #W00040 作为参数表使用。



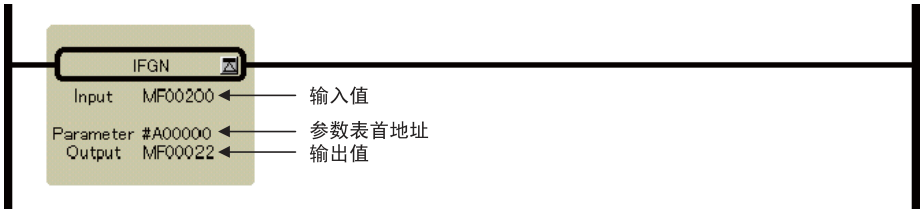
长整型运算时（数据数 N ＝ 20）

将 #L00000 ～ #L00080 作为参数表使用。



实型运算时（数据数 N ＝ 20）

将 #F00000 ～ #F00080 作为参数表使用。



1.7.11　直线加减速器 1 指令 (LAU)

■ 概要

LAU 指令对速度指令输入 (Input) 按一定的加减速率执行加速、减速。动作按照事先设定的参数表的内容进行。对 LAU 运算的输入可以使用整型或实型。不能使用长整型数据。整型和实型的参数表构成是不同的。速度值被输出到 Output 中。

表 1.23　整型 LAU 指令参数表

ADR	类型	符号	名称	规格	输入输出
0	W	RLY	继电器输入输出	继电器输入、继电器输出 *	IN/OUT
1	W	LV	输入 100% 电平	输入 100% 刻度	IN
2	W	AT	加速时间	0% ～ 100% 的加速时间 (0.1s)	IN
3	W	BT	减速时间	100% ～ 0% 的减速时间 (0.1s)	IN
4	W	QT	紧急停止时间	100% ～ 0% 的紧急停止时间 (0.1s)	IN
5	W	V	当前速度	LAU 输出 (向 Output 输出)	OUT
6	W	DVDT	当前加减速度	一般加速率以 5000 定标	OUT
7	W	—	(备用)	备用寄存器	—
8	W	VIM	上次速度指令	速度指令输入的上次值保存用	OUT
9	W	DVDTK	DVDT 系数	当前加速度 (DVDT) 的定标系数 (-32768 ～ 32767)	IN
10	L	REM	余数	加减速率的余数	OUT

* 继电器输入输出位的分配如下。

BIT	符号	名称	规格	输入输出
0	RN	在线运行中	在在线运行中输入 “闭”	IN
1	QS	紧急停止	紧急停止 输入 “开” *	IN
2	DVDTF	不执行 DVDT 运算	不执行 DVDT 运算 输入 “闭”	IN
3	DVDTS	DVDT 运算选择	DVDT 运算方式的选择	IN
4 ～ 7	—	(备用)	输入用备用继电器	IN
8	ARY	加速中	在加速中输出 “闭”	OUT
9	BRY	减速中	在减速中输出 “闭”	OUT
A	LSP	零速	零速 输出 “闭”	OUT
B	EQU	一致	输入值＝输出值 输出 “闭”	OUT
C ～ F	—	(备用)	输出用备用继电器	OUT

* 紧急停止 (QS) 为 “开” 时，作为加减速时间使用紧急停止时间 (QT)。

表 1.24 实型 LAU 指令参数表

ADR	类型	符号	名称	规格	输入输出
0	W	RLY	继电器输入输出	继电器输入、继电器输出*	IN/OUT
1	W	—	（备用）	备用寄存器	—
2	F	LV	输入 100% 的值	输入 100% 刻度	IN
4	F	AT	加速时间	0% ～ 100% 的加速时间 (0.1s)	IN
6	F	BT	减速时间	100% ～ 0% 的减速时间 (0.1s)	IN
8	F	QT	紧急停止时间	100% ～ 0% 的紧急停止时间 (0.1s)	IN
10	F	V	当前速度	LAU 输出（向 Output 输出）	OUT
12	F	DVDT	当前加减速度	一般加速率以 5000 定标	OUT

* 继电器输入输出位的分配如下。

BIT	符号	名称	规格	输入输出
0	RN	在线运行中	在在线运行中输入 “闭”	IN
1	QS	紧急停止	紧急停止 输入 “开” *	IN
2 ～ 7	—	（备用）	输入用备用继电器	IN
8	ARY	加速中	在加速中输出 “闭”	OUT
9	BRY	减速中	在减速中输出 “闭”	OUT
A	LSP	零速	零速 输出 “闭”	OUT
B	EQU	一致	输入值=输出值 输出 “闭”	OUT
C ～ F	—	（备用）	输出用备用继电器	OUT

* 紧急停止 (QS) 为 “开” 时，作为加减速时间使用紧急停止时间 (QT)。

在 LAU 指令内部按以下方式进行运算。

整型 LAU 指令

$$\text{加速率 (ADV)} = \frac{LV \times Ts(0.1ms) + REM}{AT(0.1s) \times 1000}$$

VI ＞ V’ (V’ ≧ 0) 时
V ＝ V’ ＋ ADV：加速中 (ARY)ON
VI ＜ V’ (V’ ≦ 0) 时
V ＝ V’ － ADV：加速中 (ARY)ON

$$\text{减速率 (BDV)} = \frac{LV \times Ts(0.1ms) + REM}{BT(0.1s) \times 1000}$$

VI ＞ V’ (V’ ＜ 0) 时
V ＝ V’ ＋ BDV：减速中 (BRY)ON
VI ＜ V’ (V’ ＞ 0) 时
V ＝ V’ － BDV：减速中 (BRY)ON

$$\text{紧急停止率 (QDV)} = \frac{LV \times Ts(0.1ms) + REM}{QT(0.1s) \times 1000}$$

QS ＝ ON(VI ＞ V’, V’ ＜ 0) 时
V ＝ V’ ＋ QDV：减速中 (BRY)ON
QS ＝ ON(VI ＜ V’, V’ ＞ 0) 时
V ＝ V’ － QDV：减速中 (BRY)ON

V’：上次速度输出值
VI：速度指令输入
Ts：扫描时间设定值

- DVDT 运算指令 (DVDTF) 为 ON 时，进行当前加减速度 (DVDT) 运算。
- DVDTF 为 OFF 时，输出 DVDT ＝ 0。
  DVDTF 为 ON 时，当前加减速度 (DVDT) 运算通过 DVDT 运算选择 (DVDTS)，进行以下任一运算后被输出。

DVDTS 为 ON 时:

$$DVDT = \frac{V - V'}{ADV} \times 5000$$

DVDTS 为 OFF 时：DVDT ＝ (V × DVDTK)-(V' × DVDTK);DVDTK:DVDT 系数 V ＝ 0 时，零速 (LSP) 为 ON，VI ＝ V 时，一致 (EQU) 为 ON。

- 在线运行中 (RN) 为 “开”时，输出 V ＝ 0、DVDT ＝ 0、REM ＝ 0。

实型 LAU 指令

加速率 (ADV) $= \frac{LV \times Ts(0.1ms)}{AT(0.1s) \times 10000}$

VI ＞ V'(V' ＞ 0) 时
V ＝ V' ＋ ADV：加速中 (ARY)ON
VI ＜ V'(V' ＜ 0) 时
V ＝ V' － ADV：加速中 (ARY)ON

减速率 (BDV) $= \frac{-LV \times Ts(0.1ms)}{BT(s) \times 10000}$

VI ＜ V'(V' ＞ 0) 时
V ＝ V' ＋ BDV：减速中 (BRY)ON
VI ＞ V'(V' ＜ 0) 时
V ＝ V' － BDV：减速中 (BRY)ON

紧急停止率 (QDV) $= \frac{-LV \times Ts(0.1ms)}{QT(s) \times 10000}$

QS ＝ ON (V' ＞ VI ≧ 0) 时
V ＝ V' ＋ QDV：减速中 (BRY)ON
QS ＝ ON (V' ＜ VI ≦ 0) 时
V ＝ V' － QDV：减速中 (BRY)ON

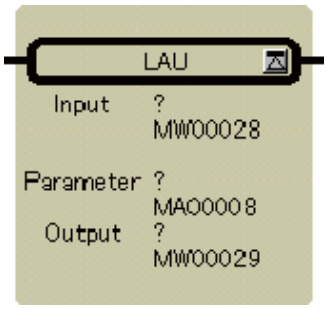
V'：上次速度输出值
VI：速度指令输入
Ts：扫描时间设定值 (ms)

进行以下运算后输出当前加减速度 (DVDT)。

$$DVDT = \frac{V - V'}{ADV} \times 5000$$

在线运行中 (RN) 为 “开”时，输出 V ＝ 0、DVDT ＝ 0。

■ 格式



标记：LAU
全称：Linear Accelerator
类别：DDC
图标：

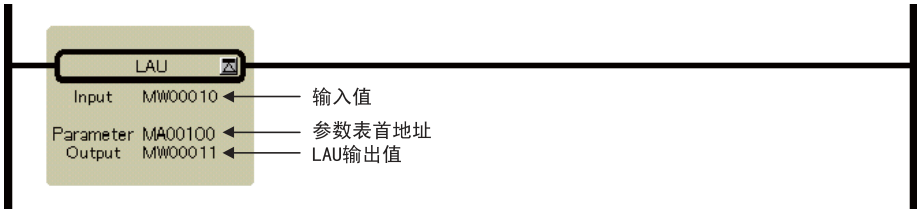
■ 参数

参数名称	设定
Input （输入值）	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Parameter （参数表首地址）	• 寄存器地址（#、C 寄存器除外） • 同上带下标字母
Output （输出值）	• 整型、长整型、实型寄存器（#、C 寄存器除外） • 同上带下标字母 • 下标寄存器

■ 程序举例

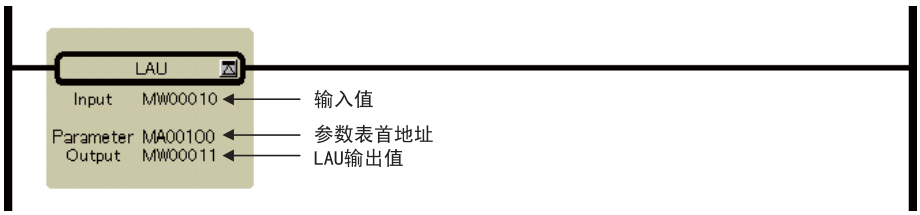
整型运算时

将 MW00100 ～ MW00106 作为参数表使用。



实型运算时

将 MF00200 ～ MF00212 作为参数表使用。



1.7.12　直线加减速器 2 指令 (SLAU)

■ 概要

SLAU 指令对于速度指令输入 (Input)，按可变的加减速率执行加速、减速。动作按照事先设定的参数表的内容进行。速度指令输入的输入值可正可负。另，请设定直线加减速时间 (AT/BT) ≧加减速 S 字时间 (AAT/BT)。

对 SLAU 运算的输入可以使用整型或实型。不能使用长整型数据。整型和实型的参数表构成不同。

表 1.25　整型 SLAU 指令参数表

ADR	类型	符号	名称	规格	输入输出
0	W	RLY	继电器输入输出	继电器输入、继电器输出 *	IN/OUT
1	W	LV	输入 100% 电平	输入 100% 刻度	IN
2	W	AT	加速时间	0% ～ 100% 的加速时间 (0.1s)	IN
3	W	BT	减速时间	100% ～ 0% 的减速时间 (0.1s)	IN
4	W	QT	紧急停止时间	100% ～ 0% 的紧急停止时间 (0.1s)	IN
5	W	AAT	加速 S 字时间	加速时的 S 字时间 (0.01 ～ 32.00s)	IN
6	W	BBT	减速 S 字时间	减速时的 S 字时间 (0.01 ～ 32.00s)	IN
7	W	V	当前速度	SLAU 输出 (向 Output 输出)	—
8	W	DVDT1	当前加减速度	一般加速率以 5000 定标	OUT
9	W	—	(备用)	备用寄存器	—
10	W	ABMD	保持时速度上升	发出保持指令后到稳定状态的速度变化量	OUT
11	W	REM1	余数	加减速率的余数	OUT
12	W	—	(备用)	预约寄存器	—
13	W	VIM	上次速度指令	速度指令输入的上次值保存用	OUT
14	L	DVDT2	当前加减速度 2	实际的加减速度放大 1000 倍	OUT
16	L	DVDT3	当前加减速度 3	当前加减速度 (＝ DVDT2/1000)	OUT
18	L	REM2	余数	S 字区间加减速率的余数	OUT
20	W	REM3	余数	当前速度的余数	OUT
21	W	DVDTK	DVDT1 系数	当前加减速度 1(DVDT1) 的定标系数 (-32768 ～ 32767)	IN

* 继电器输入输出位的分配如下。

BIT	符号	名称	规格	输入输出
0	RN	在线运行中	在在线运行中输入 “闭”	IN
1	QS	紧急停止	紧急停止 输入 “开” *	IN
2	DVDTF	不执行 DVDT1 运算	不执行 DVDT1 运算 输入 “闭”	IN
3	DVDTS	DVDT1 运算选择	DVDT1 运算方式的选择	IN
4 ～ 7	—	（备用）	输入用备用继电器	IN
8	ARY	加速中	在加速中输出 “闭”	OUT
9	BRY	减速中	在减速中输出 “闭”	OUT
A	LSP	零速	零速 输出 “闭”	OUT
B	EQU	一致	输入值＝输出值 输出 “闭”	OUT
C	EQU	（备用）	输出用备用继电器	OUT
D	CCF	工作继电器	系统内部工作继电器	OUT
E	BBF	工作继电器	系统内部工作继电器	OUT
F	AAF	工作继电器	系统内部工作继电器	OUT

* 紧急停止 (QS) 为 “开” 时，作为加减速时间使用紧急停止时间 (QT)。

表 1.26 实型 SLAU 指令参数表

ADR	类型	符号	名称	规格	输入输出
0	W	RLY	继电器输入输出	继电器输入、继电器输出 *	IN/OUT
1	W	—	（备用）	备用寄存器	—
2	F	LV	输入 100% 电平	输入 100% 刻度	IN
4	F	AT	加速时间	0% ～ 100% 的加速时间	IN
6	F	BT	减速时间	100% ～ 0% 的减速时间	IN
8	F	QT	紧急停止时间	100% ～ 0% 的紧急停止时间	IN
10	F	AAT	加速 S 字时间	加速时的 S 字时间 (0.01 ～ 32.00s)	IN
12	F	BBT	减速 S 字时间	减速时的 S 字时间 (0.01 ～ 32.00s)	IN
14	F	V	当前速度	SLAU 输出（向 Output 输出）	OUT
16	F	DVDT1	当前加减速度	输出实际的加减速度	OUT
18	F	ABMD	保持时速度上升	发出保持指令后到稳定状态的速度变化量	OUT

* 继电器输入输出位的分配如下。

BIT	符号	名称	规格	输入输出
0	RN	在线运行中	在在线运行中输入 “闭”	IN
1	QS	紧急停止	紧急停止 输入 “开” *	IN
2 ～ 7	—	（备用）	输入用备用继电器	IN
8	ARY	加速中	在加速中输出 “闭”	OUT
9	BRY	减速中	在减速中输出 “闭”	OUT
A	LSP	零速	零速时输出 “闭”	OUT
B	EQU	一致	输入值＝输出值 输出 “闭”	OUT
C ～ F	—	（备用）	输出用备用继电器	OUT

在 SLAU 指令内部按以下方式进行运算。

整型 SLAU 指令

$$\text{加速率 (ADV)} = \frac{LV \times Ts(0.1ms) + REM1}{AT(0.1s) \times 1000}$$

在 S 字区间外 (ADVS ＞ ADV)
VI ＞ V’(V’ ≧ 0) 时
V ＝ V’ ＋ ADV；加速中 (ARY)ON
VI ＜ V’(V’ ≦ 0) 时
V ＝ V’ － ADV；加速中 (ARY)ON

$$\text{减速率 (BDV)} = \frac{LV \times Ts(0.1ms) + REM1}{BT(0.1s) \times 1000}$$

在 S 字区间外 (BDVS ＞ BDV)
VI ＞ V’(V’ ＜ 0) 时
V ＝ V’ ＋ BDV；减速中 (BRY)ON
VI ＜ V’(V’ ＞ 0) 时
V ＝ V’ － BDV; 减速中 (BRY)ON

$$\text{紧急停止率 (QDV)} = \frac{LV \times Ts(0.1ms) + REM1}{QT(0.1s) \times 1000}$$

QS ＝ ON(VI ＞ V’，V’ ＜ 0) 时
V ＝ V’ ＋ QDV; 减速中 (BRY)ON
QS ＝ ON(VI ＜ V’，V’ ＞ 0) 时
V ＝ V’ － DV; 减速中 (BRY)ON

（注）紧急停止时不做 S 字动作，而进行直线动作（和 LAU 紧急停止时的动作相同）。

S 字区间加速率 (ADVS) ＝ ADVS’ ± AADVS

$$AADVS = \frac{ADV \times Ts(0.1ms) + REM2}{AAT(0.01s) \times 100}$$

在 S 字区间内 (BDVS ＞ BDV)
VI ＞ V’(V’ ≧ 0) 时
V ＝ V’ ＋ ADVS; 减速中 (ARY)ON
VI ＜ V’(V’ ≦ 0) 时
V ＝ V’-ADVS; 加速中 (ARY)ON

S 字区间减速率 (BDVS) ＝ BDVS’ ± BBDVS

$$BBDVS = \frac{BDV \times Ts(0.1ms) + REM2}{BBT(0.01s) \times 100}$$

在 S 字区间内 (BDVS ＞ BDV)
VI ＞ V’(V’ ＜ 0) 时
V ＝ V’ ＋ BDVS; 减速中 (BRY)ON
VI ＜ V’(V’ ＞ 0) 时
V ＝ V’-BDVS; 加速中 (BRY)ON

V’：上次速度输出值
VI：速度指令输入
Ts：扫描时间设定值

- DVDT1 运算指令 (DVDTF) 为 ON 时，进行当前加减速度 1(DVDT1) 运算。
- DVDTF 为 OFF 时，输出 DVDT1 ＝ 0。
  DVDTF 为 ON 时，通过 DVDT1 运算选择 (DVDTS)，在进行以下任一运算后输出当前加减速度 1(DVDT1)。

DVDTS 为 ON 时：

$$DVDT1 = \frac{(V - V')}{ADV} \times 5000$$

DVDTS 为 OFF 时：DVDT ＝ (V × DVDTK)-(V’ × DVDTK);DVDTK:DVDT 系数

- 当前加减速度 2(DVDT2) 按以下方法输出。

  加速中：S 字区间内：DVDT2 ＝± ADVS
  　　S 字区间外：DVDT2 ＝± ADV

  减速中：S 字区间内：DVDT2 ＝± BDVS
  　　S 字区间外：DVDT2 ＝± BDV

- 进行以下运算后输出保持时速度上升量 (ABMD)。

$$ABMD = \frac{DVDT2' \times DVDT2'}{2 \times AADVS(BBDVS)}$$

;DVDT2’ ＝当前加减速度 2(DVDT2) 上次值

- V ＝ 0 时零速 (LSP) 为 ON，VI ＝ V 时一致 (EQU) 为 ON。
- 在线运行中 (RN) 为 “开”时，输出 V ＝ 0、DVDT1 ＝ 0、DVDT2 ＝ 0、DVDT3 ＝ 0、ABMD ＝ 0、REM1 ＝ 0、REM2 ＝ 0、REM3 ＝ 0。

实型 SLAU 指令

$$加速率\ (ADV) = \frac{LV \times Ts(0.1ms)}{AT(s) \times 10000}$$

在 S 字区间外 (ADVS ＞ ADV)
VI ＞ V’(V’ ＞ 0) 时：
V ＝ V’ ＋ ADV

$$减速率\ (BDV) = \frac{-LV \times Ts(0.1ms)}{BT(s) \times 10000}$$

在 S 字区间外 (BDVS ＜ BDV)
VI ＜ V’(V’ ＞ 0) 时：
V ＝ V’ ＋ BDV

$$紧急停止率\ (QDV) = \frac{-LV \times Ts(0.1ms)}{QT(s) \times 10000}$$

QS ＝ ON(V’ ＞ VI) 时：
V ＝ V’ ＋ QDV

S 字区间加速率 (ADVS) ＝ ADVS’ ± AADVS

$$AADVS = \frac{ADV \times Ts(0.1ms)}{AAT(s) \times 10000}$$

:ADVS’ ＝ ADVS 的上次值
在 S 字区间内 (ADVS ＜ ADV)
VI ＞ V’(V’ ＞ 0) 时：
V ＝ V’ ＋ ADVS

S 字区间减速率 (BDVS) ＝ BDVS’ ± BBDVS

$$BBDVS = \frac{BDV \times Ts(0.1ms)}{BBT(s) \times 10000}$$

:BDVS’ ＝ BDVS 的上次值
在 S 字区间外 (BDVS ＞ BDV)
VI ＜ V’(V’ ＞ 0) 时
V ＝ V’ ＋ BDVS

V’: 上次速度输出值
VI: 速度指令输入
Ts: 扫描时间设定值

- 进行以下运算后输出当前加减速度 (DVDT)。

  加速中：S 字区间内：DVDT ＝ ADVS
  　　S 字区间外：DVDT ＝ ADV

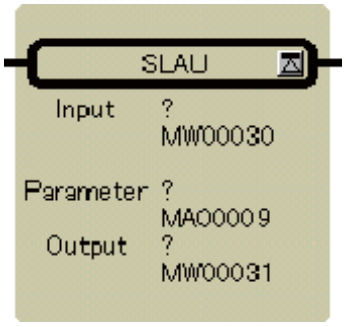
  减速中：S 字区间内：DVDT ＝ BDVS
  　　S 字区间外：DVDT ＝ BDV

• 进行以下运算后，输出保持时速度上升量 (ABMD)。

$$ABMD = \frac{DVDT \times DVDT}{2 \times AADVS(BBDVS)}$$

• 在线运行中 (RN) 为 “开” 时，输出 V ＝ 0、DVDT ＝ 0、ABMD ＝ 0。

■ 格式



标记：SLAU
全称：S-Curve Linear Accelerator
类别：DDC
图标： 

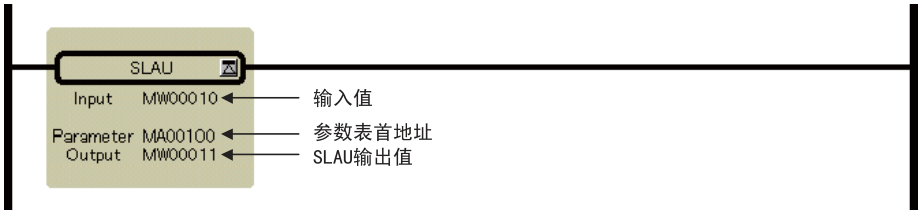
■ 参数

参数名称	设定
Input （输入值）	• 所有整型、长整型、实型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Parameter （参数表首地址）	• 寄存器地址 (#、C 寄存器除外） • 同上带下标字母
Output （输出值）	• 整型、长整型、实型寄存器 (#、C 寄存器除外） • 同上带下标字母 • 下标寄存器

■ 程序举例

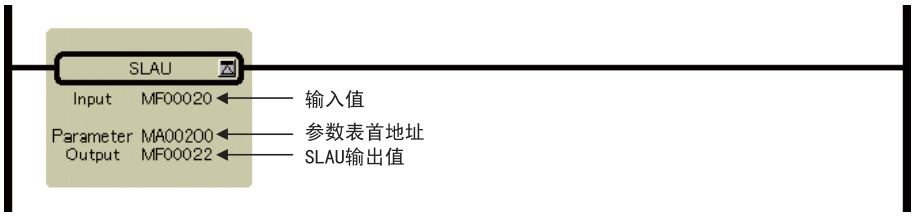
整型运算时

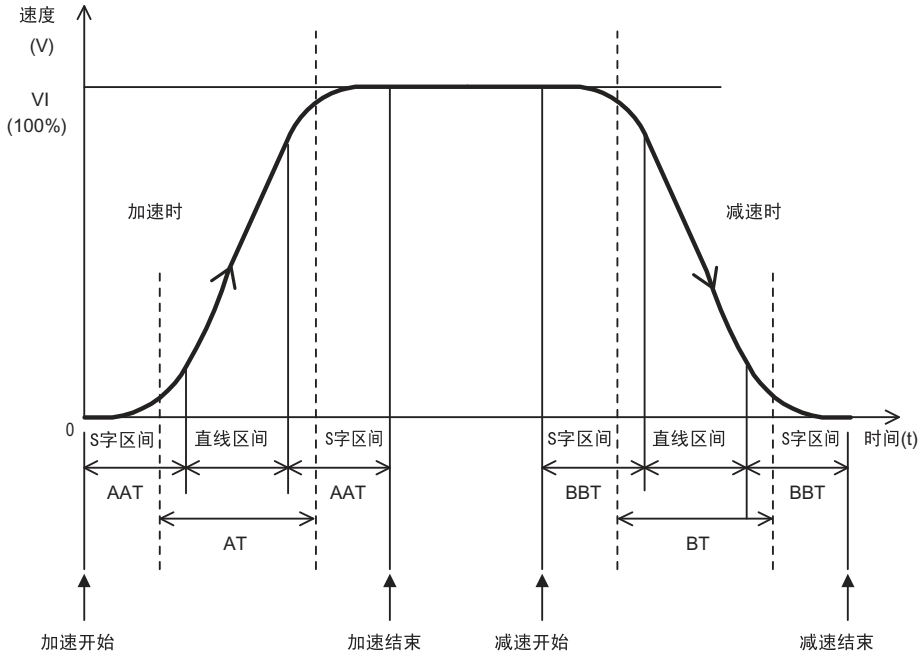
将 MW00100 ～ MW00111 作为参数表使用。



实型运算时

将 MF00200 ～ MF00218 作为参数表使用。



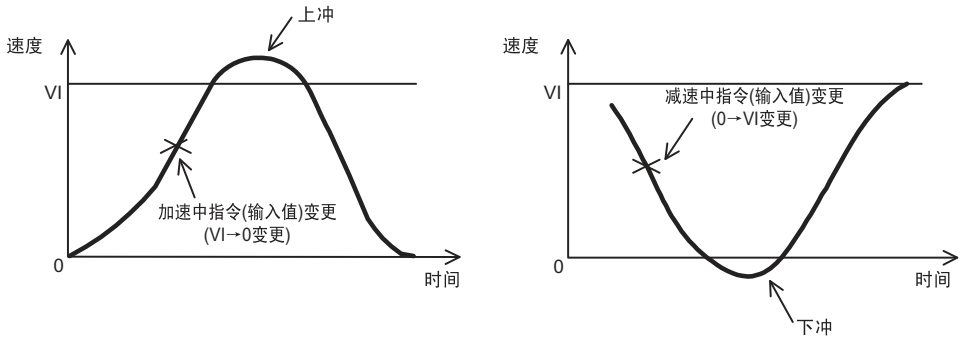


（注）使用整型 SLAU 指令时，请注意以下事项。
未达到输入值 (VI) 前（加减速中）时，请勿改变输入值 (VI)。
如果在加减速中改变输入值 (VI)，则有可能发生上冲 / 下冲。
（参照下图）
在加减速中改变输入值 (VI) 时，请使用以下任何一种编制应用程序。

*1. 请使用实型 SLAU 指令。

*2. 使用整型 SLAU 指令时，请同时使用 LIMIT 指令。即，将整型 SLAU 指令输出值作为 LIMIT 指令输入值，限制上冲 / 下冲。

考虑应用程序编制的难易度，建议*1 使用实型 SLAU 指令。



1.7.13 脉宽调制指令 (PWM)

■ 概要

PWM 指令将 Input 作为输入值 (-100.00 ～ 100.00%，单位：0.01%)，进行 PWM 转换，并将其结果输出到 Output 和参数表中。不能进行长整型运算和实型运算。PWM 的 ON 输出的时间和次数按如下所示计算。

$$\text{ON 输出时间} = \frac{\text{PWMT}(X + 10000)}{20000}$$

$$\text{ON 输出次数} = \frac{\text{PWMT}(X + 10000)}{20000}$$

X：输入值
Ts：扫描时间设定值
100.00% 输入时：全部 ON
0% 输入时：50% 的功 (50%ON)
-100.00% 输入时：全部 OFF
PWM 复位 (PWMRST) 为“闭”时，内部运算全部被复位，将此点作为开始点，执行 PWM 运算。电源 ON 后，将 PWMRST 置为 “闭”，清除内部运算后，使用 PWM 指令。

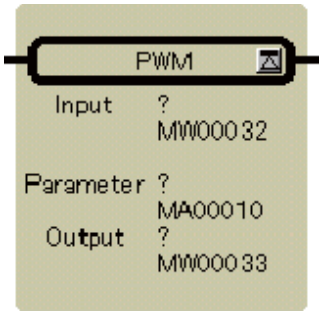
表 1.27 PWN 指令参数表

ADR	类型	符号	名称	规格	输入输出
0	W	RLY	继电器输入输出	继电器输入、继电器输出 *	IN/OUT
1	W	RWMT	PWM 周期	PWM 周期 (1MS)(1 ～ 327676ms)	IN
2	W	ONCNT	ON 输出设定定时器	ON 输出设定定时器 (1ms)	OUT
3	W	CVON	ON 输出计数定时器	ON 输出计数定时器 (1ms)	OUT
4	W	CVONREM	ON 输出计数定时器余数	ON 输出计数定时器余数 (0.1ms)	OUT
5	W	OFFCNT	OFF 输出设定定时器	OFF 输出设定定时器 (1ms)	OUT
6	W	CVOFF	OFF 输出计数定时器	OFF 输出计数定时器 (1ms)	OUT
7	W	CVOFFREM	OFF 输出计数定时器余数	OFF 输出计数定时器余数 (0.1ms)	OUT

* 继电器输入输出位的分配如下。

BIT	符号	名称	规格	输入输出
0	PWMRST	RWM 复位	PWM 复位时输入 “闭”	IN
2 ～ 7	—	(备用)	输入用备用继电器	IN
8	PWMOUT	PWM 输出	PWM 的输出 (2 值输出 ON = 1，OFF = 0)	OUT
9 ～ F	—	(备用)	输出用备用继电器	OUT

■ 格式



标记：PWM

全称：Pulse Width Modulation

类别：DDC

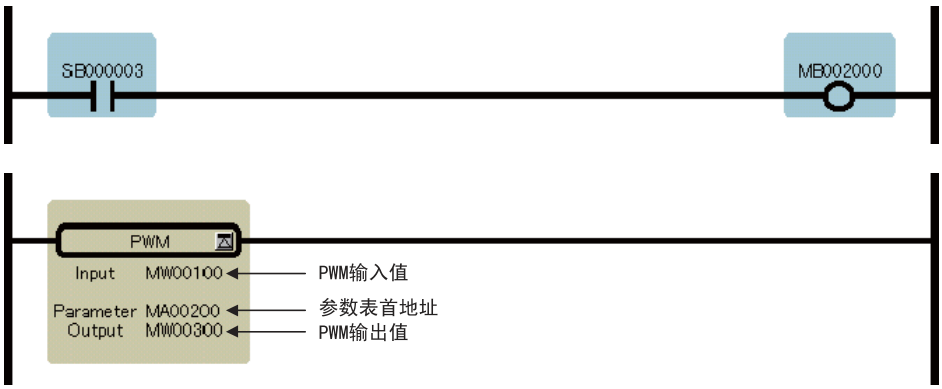
图标： 

■ 参数

参数名称	设定
Input （输入值）	• 整型寄存器 • 同上带下标字母 • 下标寄存器 • 常数
Parameter （参数表首地址）	• 寄存器地址（#、C 寄存器除外） • 同上带下标字母
Output （输出值）	• 整型寄存器（#、C 寄存器除外） • 同上带下标字母 • 下标寄存器

■ 程序举例

将 MW00100 作为 PWN 输入、MW00200 ～ MW000207 作为参数表使用。



在 DWG.L 的第一个扫描周期（在 DWG.H 使用时为 SB000001）时进行 PWM 复位。

1.8 表数据操作指令

1.8.1 块调出指令 (TBLBR)

■ 概要

以块的形式连续调出由表名称 (Table Name)、行编号、列编号来指定的文件寄存器表的要素，存储在从指定的寄存器 (Read Data) 开始的连续域中。调出的要素类型通过指定的表自动判别。

忽略存储目标寄存器的类型，不转换调出值的数据类型，按照表的要素类型存储。

在表调用过程中，发生表名称、行编号、列编号、存储位置、数据长度不足等错误时会提示出错，此时停止数据调出，传送目标寄存器中的内容保持不变。

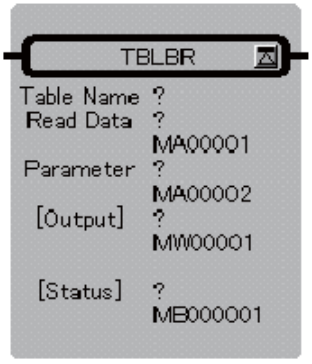
正常结束时，向 [Output] 设置传送字数，将 [Status] 置为 OFF。出错时，向 [Output] 设置错误代码，将 [Status] 置为 ON。

表 1.28 错误代码一览

错误代码	出错名称	内容
0001H	调用表未定义	对象表未定义。
0002H	行编号范围外	表要素的行编号不在对象表范围内。
0003H	列编号范围外	表要素的列编号不在对象表范围内。
0004H	要素个数不对	对象要素的个数不当。
0005H	存储目标域不足	存储域不足。
0006H	要素类型不足	指定的要素类型有异常。
0007H	Q 缓冲器出错	Q 缓冲器无数据时，执行调出指令； Q 缓冲器满数据时，以指针步进方式写入数据。
0008H	Q 表出错	指定的表不是 Q 型表。
0009H	系统出错	指令执行中，在系统内部被检测出未预见的错误。

ADR	类型	符号	名称	规格	输入输出
0	L	ROW1	表要素开始行编号	对象表要素的开始行编号 (1 ～ 65535)	IN
2	L	COL1	表要素开始列编号	对象表要素的开始列编号 (1 ～ 32767)	IN
4	W	RLEN	行要素个数	行要素个数 (1 ～ 32767)	IN
5	W	CLEN	列要素个数	列要素个数 (1 ～ 32767)	IN

■ 格式



标记：TBLBR
全称：Table Block Read
类别：TABLE
图标：

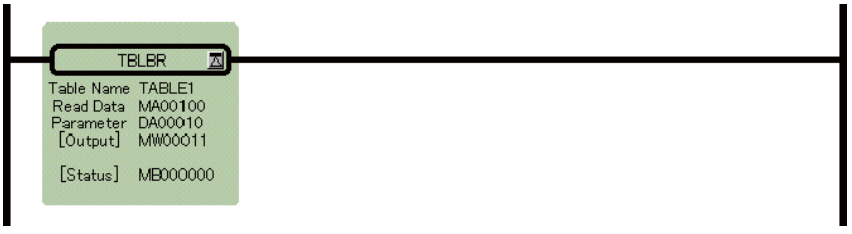
■ 参数

参数名称	设定
Table Name （传送源表名称）	• 表名称
Read Data （传送目标首地址）	• 寄存器地址（#、C 寄存器除外） • 同上带下标字母
Parameter （参数表首地址）	• 寄存器地址 • 同上带下标字母
[Output]* （传送字数）	• 整型寄存器（#、C 寄存器除外） • 同上带下标字母 • 下标寄存器
[Status]* （状态）	• 比特型寄存器（#、C 寄存器除外） • 同上带下标字母

* 可省略 [Output]、[Status]。

■ 程序举例

在定义的 TABLE1 表中，将 DW00010 ～ DW00013 作为参数表，以块的形状，将从表的要素开始位置到结束位置的数据（要素类型为整型）存储在从 MW00100 开始的域中。



1.8.2 块写入指令（TBLBW）

■ 概要

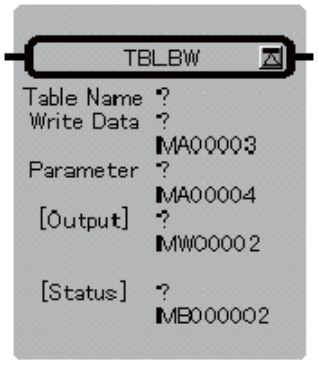
在由表名称（Table Name）、行编号、列编号来指定的文件寄存器表的要素内，以块的形式，存储从指定的寄存器（Write Data）开始的连续域。将存储目标的表的要素类型和传送源的寄存器的类型作为一致来处理。

在表调用过程中，表名称、行编号、列编号、存储位置、数据长度不足等错误情况发生时，会提示出错，停止数据写入，传送目标寄存器内容不变。

正常结束时，向［Output］设置传送字数，将［Status］置为 OFF。出错时，向［Output］设置错误代码，将［Status］置为 ON。

ADR	类型	符号	名称	规格	输入输出
0	L	ROW1	表要素开始行编号	对象表要素的开始行编号（1 ～ 65535）	IN
2	L	COL1	表要素开始列编号	对象表要素的开始列编号（1 ～ 32767）	IN
4	W	RLEN	行要素个数	行要素个数（1 ～ 32767）	IN
5	W	CLEN	列要素个数	列要素个数（1 ～ 32767）	IN

■ 格式



标记：TBLBW

全称：Table Block Write

类别：TABLE

图标： 

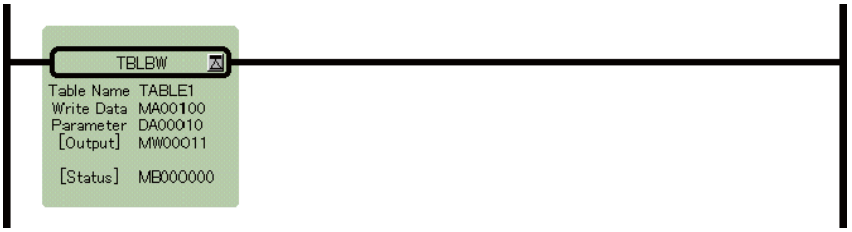
■ 参数

参数名称	设定
Table Name （传送目标表名称）	• 表名称
Write Data （传送源首地址）	• 寄存器地址（#、C 寄存器除外） • 同上带下标字母
Parameter （参数表首地址）	• 寄存器地址 • 同上带下标字母
[Output]* （传送字数）	• 整型寄存器（#、C 寄存器除外） • 同上带下标字母 • 下标寄存器
[Status]* （状态）	• 比特型寄存器（#、C 寄存器除外） • 同上带下标字母

* 可省略 [Output]、[Status]。

■ 程序举例

从定义为 TABLE1 的表中，将 DW00010 ～ DW00013 作为参数表，将从 MW00100 开始的数据以块的形状存储在从表的要素开始位置到结束位置的（要素类型为整型）域中。



1.8.3 行检索指令：纵向（TBLSRL）

■ 概要

检索由表名称（Table Name）、行编号、列编号指定的文件寄存器表的列要素，当与被指定的寄存器（Search Data）的数据一致时，报告该行的号码。通过指定的表，自动判别检索对象数据的类型。

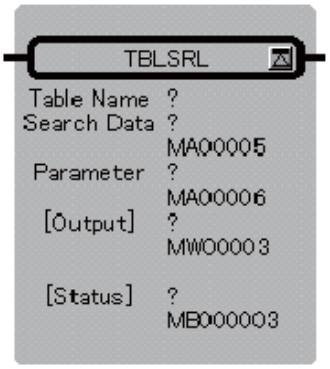
在表调用过程中，发生表名称、行编号、列编号、存储目标、数据长度不足等错误时，会提示出错。

正常结束时，如果发现一致的列要素，则参数的检索结果为 1、向 [Output] 置相应编号，将 [Status] 置为 OFF。如果没有发现，则置检索结果为 0。

发生错误时，向 [Output] 置错误代码，将 [Status] 置为 ON。

ADR	类型	符号	名称	规格	输入输出
0	L	ROW1	表要素首行编号	对象表要素的首行编号 (1 ～ 65535)	IN
2	L	ROW2	表要素末行编号	对象表要素的末行编号 (1 ～ 65535)	IN
4	L	COLUMN	表要素列编号	对象表要素的列编号 (1 ～ 32767)	IN
6	W	FIND	检索结果	检索结果 0：无相应行 1：有相应行	OUT

■ 格式



标记：TBLSRL

全称：Table Row Search

类别：TABLE

图标：

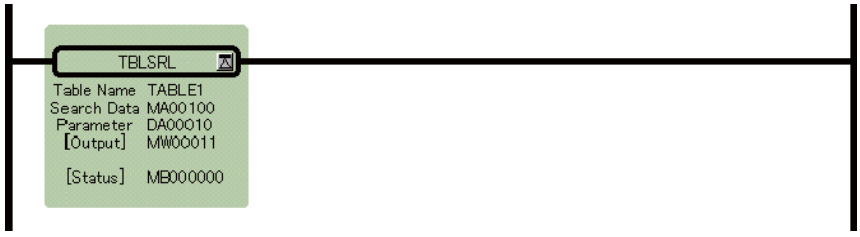
■ 参数

参数名称	设定
Table Name （检索表名称）	• 表名称
Search Data （检索数据首地址）	• 寄存器地址 • 同上带下标字母
Parameter （参数表首地址）	• 寄存器地址 • 同上带下标字母
[Output]* （传送字数）	• 整型寄存器（#、C 寄存器除外） • 同上带下标字母 • 下标寄存器
[Status]* （状态）	• 比特型寄存器（#、C 寄存器除外） • 同上带下标字母

* 可省略 [Output]、[Status]。

■ 程序举例

将 DW00010 ～ DW00013 作为参数表，检索被定义的表 TABLE1 中是否有与 MW00100（检索表的要素类型为整型时）一致的数据。



1.8.4 列检索指令：横向（TBLSRC）

■ 概要

检索由表名称（Table Name）、行编号、列编号指定的文件寄存器表的行要素，当与指定寄存器（Search Data）的数据一致时，报告该列编号。通过指定的表，自动判别检索对象数据的类型。

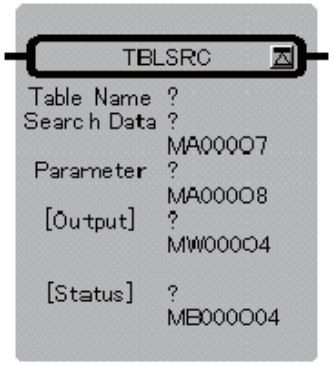
在表调用过程中，发生表名称、行编号、列编号、存储目标、数据长度不足等错误时，会提示出错。

正常结束时，如果发现一致的行要素，则参数的检索结果为 1、向［Output］置相应编号、将［Status］置为 OFF。如果没有发现，则置检索结果为 0。

发生错误时，向［Output］置错误代码，将［Status］置为 ON。

ADR	类型	符号	名称	规格	输入输出
0	L	ROW1	表要素行编号	对象表要素的行编号 （1 ～ 65535）	IN
2	L	COLUMN1	表要素首列编号	对象表要素的首列编号 （1 ～ 32767）	IN
4	L	COLUMN2	表要素末列编号	对象表要素的末列编号 （1 ～ 32767）	IN
6	W	FIND	检索结果	检索结果 0：无相应列 1：有相应列	OUT

■ 格式



标记：TBLSRC

全称：Table Column Search

类别：TABLE

图标：

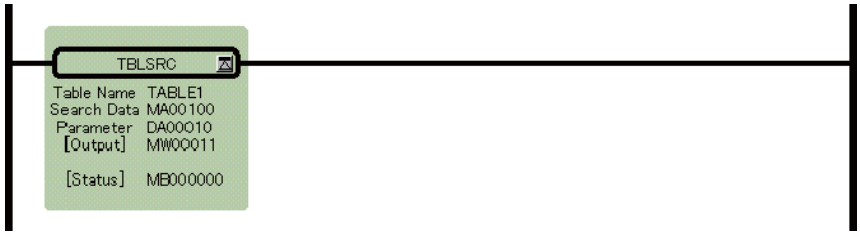
■ 参数

参数名称	设定
Table Name （检索表名称）	• 表名称
Search Data （检索数据首地址）	• 寄存器地址 • 同上带下标字母
Parameter （参数表首地址）	• 寄存器地址 • 同上带下标字母
[Output]* （传送字数）	• 整型寄存器（#、C 寄存器除外） • 同上带下标字母 • 下标寄存器
[Status]* （状态）	• 比特型寄存器（#、C 寄存器除外） • 同上带下标字母

* 可省略 [Output]、[Status]。

■ 程序举例

将 DW00010 ～ DW00013 作为参数表，检索被定义的表 TABLE1 中是否有与 MW00100（检索表的要素类型为整型时）一致的数据。



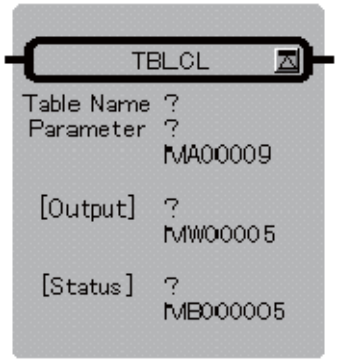
1.8.5 块清除指令（TBLCL）

■ 概要

清除由表名称（Table Name）、行编号、列编号指定的文件寄存器表的块要素的数据。
当要素的类型为字符串时写入空格，为数值时写入 0。
表要素首行编号和首列编号均为 0 时，清除整个表。
在表调用过程中，发生表名称、行编号、列编号、存储位置、数据长度不足等错误时，会提示出错，不进行数据写入。
正常结束时，向 [Output] 置清除字数，将 [Status] 置为 OFF。出错时，向 [Output] 置错误代码，将 [Status] 置为 ON。

ADR	类型	符号	名称	规格	输入输出
0	L	ROW	表要素首行编号	对象表要素的首行编号 (1 ～ 65535)	IN
2	L	COL	表要素首列编号	对象表要素的首列编号 (1 ～ 32767)	IN
4	W	RLEN	行要素个数	行要素个数 (1 ～ 32767)	IN
5	W	CLEN	列要素个数	列要素个数 (1 ～ 32767)	IN

■ 格式



标记：TBLCL
全称：Table Block Clear
类别：TABLE
图标：

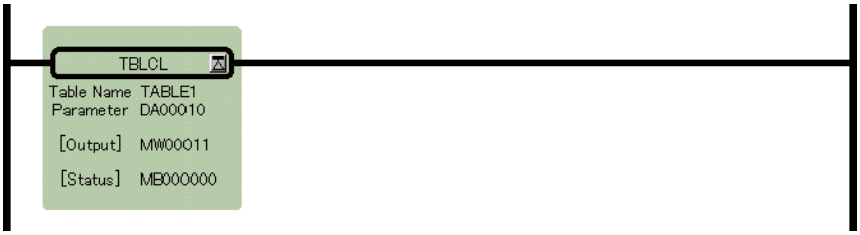
■ 参数

参数名称	设定
Table Name （表名称）	• 表名称
Parameter （参数表首地址）	• 寄存器地址 • 同上带下标字母
[Output]* （传送字数）	• 整型寄存器（#、C 寄存器除外） • 同上带下标字母 • 下标寄存器
[Status]* （状态）	• 比特型寄存器（#、C 寄存器除外） • 同上带下标字母

* 可省略 [Output]、[Status]。

■ 程序举例

将被定义为 TABLE1 的表中被指定的块 DW00010 ～ DW00013 作为参数表清除。



1.8.6 表间块传送指令（TBLMV）

■ 概要

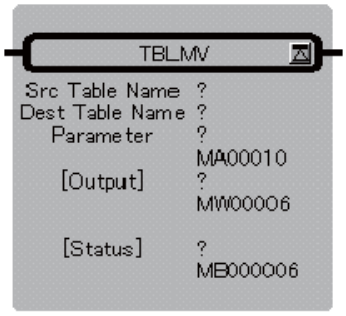
向表的其它块传送由表名称（Table Name）、行编号、列编号指定的文件寄存器表的块要素数据。同一表内的传送、不同表间的传送均可进行。传送源和传送目标的列要素的类型不同时提示出错，不进行数据的写入。

正常结束时，向［Output］置传送字数，将［Status］置为 OFF。出错时，向［Output］置错误代码，将［Status］置为 ON。

表 1.29 表间块传送指令参数表

ADR	类型	符号	名称	规格	输入输出
0	L	ROW1	表要素首行编号	传送源表要素的首行编号 （1 ～ 65535）	IN
2	L	COLUMN1	表要素首列编号	传送源表要素首列编号 （1 ～ 32767）	IN
4	W	RLEN	行要素个数	传送行要素个数（1 ～ 32767）	IN
5	W	CLEN	列要素个数	传送列要素个数（1 ～ 32767）	IN
6	L	ROW2	表要素首行编号	传送目标表要素首行编号 （1 ～ 65535）	IN
7	L	COLUMN2	表要素首列编号	传送目标表要素首列编号 （1 ～ 32767）	IN

■ 格式



标记：TBLMV

全称：Table Block Move

类别：TABLE

图标：

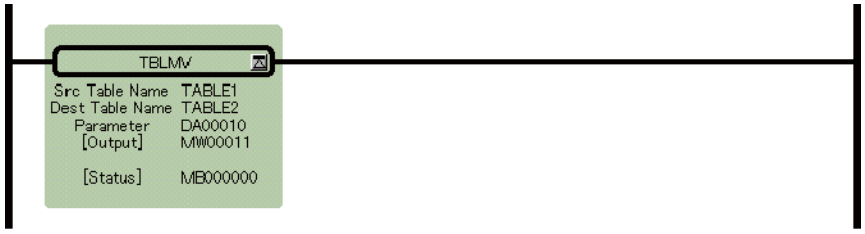
■ 参数

参数名称	设定
Src Table Name （传送源表名称）	• 表名称
Dest Table Name （传送目标表名称）	• 表名称
Parameter （参数表首地址）	• 寄存器地址 • 同上带下标字母
[Output]* （传送字数）	• 整型寄存器（#、C 寄存器除外） • 同上带下标字母 • 下标寄存器
[Status]* （状态）	• 比特型寄存器（#、C 寄存器除外） • 同上带下标字母

* 可省略 [Output]、[Status]。

■ 程序举例

有被定义为 TABLE1、TABLE2 的表，将 DW00010 ～ DW00015 作为参数表，将 TABLE1 中指定的块传送到 TABLE2 指定的块中。



1.8.7 Q 表调出指令（QTBLR，QTBLRI）

■ 概要

连续调出由表名称（Table Name）、行编号、列编号指定的文件寄存器表的列要素，存储在从指定的寄存器（Read Data）开始的连续域中。

通过指定的表自动判别调出的要素类型。忽略存储目标寄存器的类型，不转换调出值的数据类型，按照表的要素类型存储调出值。QTBLR 指令时，Q 表调出指针不发生变化。QTBLRI 指令时，Q 表调出指针步进 1 行。

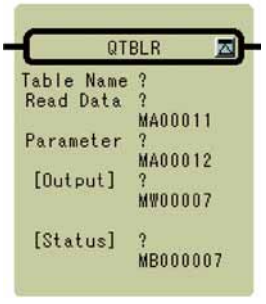
在表调用过程中，发生表名称、行编号、列编号、存储位置、数据长度不足等错误时，会提示出错，并停止数据读出，指示针也不步进。存储目标寄存器的内容保持不变。

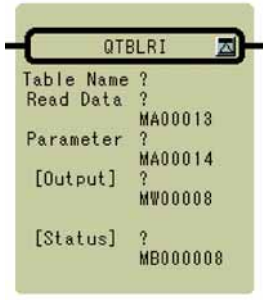
正常结束时，向 [Output] 置传送字数，将 [Status] 置为 OFF。出错时，向 [Output] 置错误代码，将 [Status] 置为 ON。指针值不变化。

表 1.30 Q 表调出指令参数表

ADR	类型	符号	名称	规格	输入输出
0	L	ROW	表要素相应行编号	对象表要素相应行编号 (1 ～ 65535)	IN
2	L	COLUMN	表要素首列编号	传送源表要素首列编号 (1 ～ 32767)	IN
4	W	CLEN	列要素个数	传送列要素个数 (1 ～ 32767)	IN
5	W	—	—	备用	—
6	L	RPTR	调出指针	执行后的 Q 调出指针	OUT
8	L	WPTR	写入指针	执行后的 Q 写入指针	OUT

■ 格式





标记：QTBLR，QTBLRI

全称：Queue Table Read，Queue Table Read

类别：TABLE

图标：

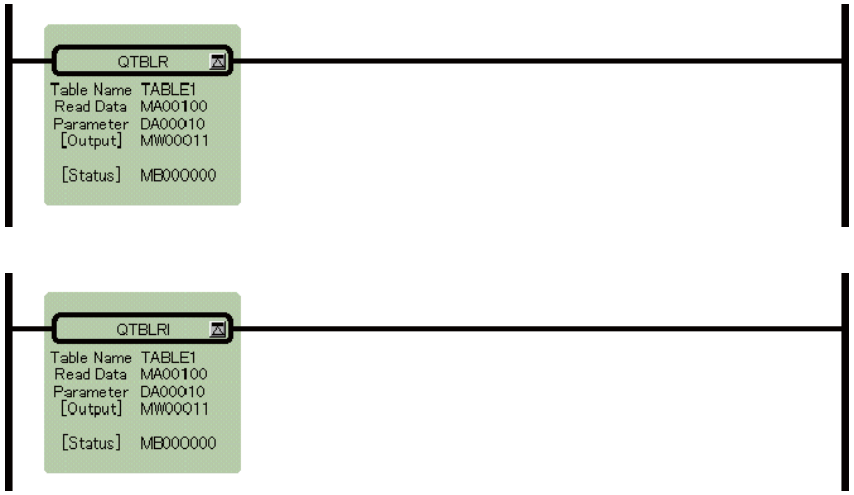
■ 参数

参数名称	设定
Table Name （传送源表名称）	• 表名称
Read Data （传送目标数据首地址）	• 寄存器地址（#、C寄存器除外） • 同上带下标字母
Parameter （参数表首地址）	• 寄存器地址 • 同上带下标字母
[Output]* （传送字数）	• 整型寄存器（#、C寄存器除外） • 同上带下标字母 • 下标寄存器
[Status]* （状态）	• 比特型寄存器（#、C寄存器除外） • 同上带下标字母

* 可省略[Output]、[Status]。

■ 程序举例

从被定义的表TABLE1中，以DW00010～DW00012为参数表，从MW00100开始按列要素个数存储表的列要素数据（要素类型为整型）。



1.8.8 Q 表写入指令（QTBLW，QTBLWI）

■ 概要

调出从指定的寄存器(Write Data)开始的连续域的数据，向由表名称(Table Name)、行编号、列编号指定的文件寄存器表的列要素写入。将存储目标的表要素类型和存储源寄存器的类型作为一致来处理。

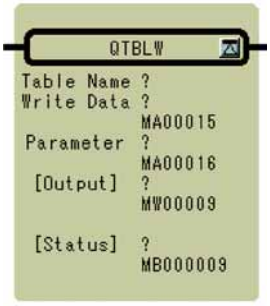
QTBLW 指令时，Q 表写入指针不发生变化。QTBLWI 指令时，Q 表写入指针步进 1 行。

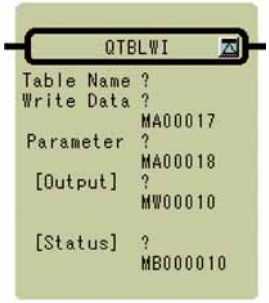
在表调用过程中，发生表名称、行编号、列编号、存储位置、数据长度不足等错误时，会提示出错，并停止数据写入，指针也不步进。

正常结束时，向 [Output] 置传送字数，将 [Status] 置为 OFF。出错时，向 [Output] 置错误代码，将 [Status] 置为 ON。指针值不变化。

ADR	类型	符号	名称	规格	输入输出
0	L	ROW	表要素相应行编号	对象表要素相应行编号 (1 ～ 65535)	IN
2	L	COLUMN	表要素首列编号	传送源表要素首列编号 (1 ～ 32767)	IN
4	W	CLEN	列要素个数	连续写入的列要素个数 (1 ～ 32767)	IN
5	W	—	—	备用	—
6	L	RPTR	调出指针	执行后的 Q 调出指针	OUT
8	L	WPTR	写入指针	执行后的 Q 写入指针	OUT

■ 格式





标记：QTBLW，QTBLWI

全称：Queue Table Write，Queue Table Pointer Clear

类别：TABLE

图标：

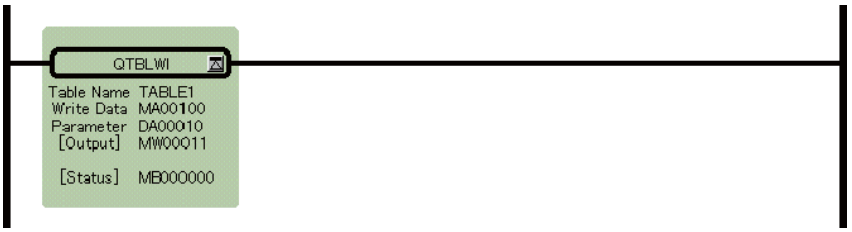
■ 参数

参数名称	设定
Table Name （传送目标表名称）	• 表名称
Write Data （传送源数据地址）	• 寄存器地址（#、C寄存器除外） • 同上带下标字母
Parameter （参数表首地址）	• 寄存器地址 • 同上带下标字母
[Output]* （传送字数）	• 整型寄存器（#、C寄存器除外） • 同上带下标字母 • 下标寄存器
[Status]* （状态）	• 比特型寄存器（#、C寄存器除外） • 同上带下标字母

* 可省略[Output]、[Status]。

■ 程序举例

在被定义为TABLE1的表中，将DW00010～DW00013作为参数表使用，将从MW00100开始的相应列要素个数的整型连续数据写入表的列要素数据内。



1.8.9 Q 指针清除指令（QTBLCL）

■ 概要

将由表名称（Table Name）指定的文件寄存器表的 Q 调出指针和 Q 写入指针返回到初始状态（第 1 行）。
正常结束时，向 [Output] 置 0，将 [Status] 置为 OFF。
发生错误时，向 [Output] 置错误代码，将 [Status] 置为 ON。

■ 格式

标记：QTBLCL
全称：Queue Table Pointer Clear
类别：TABLE
图标：

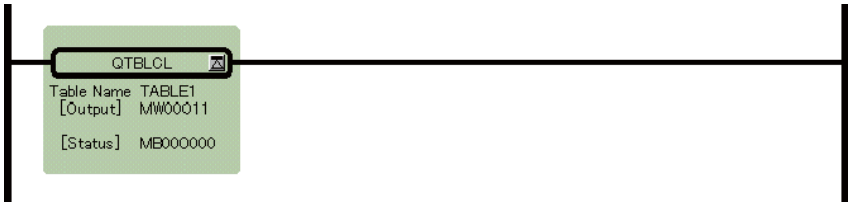
■ 参数

参数名称	设定
Table Name	• 表名称
[Output]* （传送字数）	• 整型寄存器（#、C 寄存器除外） • 同上带下标字母 • 下标寄存器
[Status]* （状态）	• 比特型寄存器（#、C 寄存器除外） • 同上带下标字母

* 可省略 [Output]、[Status]。

■ 程序举例

将 TABLE1 的 Q 调出指针和 Q 写入指针返回到初始状态。



2 章

系统标准函数指令

2.1 信息函数 - 2-2
2.1.1 信息发送函数 (MSG-SND) - 2-2
2.1.2 信息接收函数 (MSG-RCV) - 2-15
2.2 示踪函数 - 2-24
2.2.1 示踪函数 (TRACE) - 2-24
2.2.2 数据示踪调出函数 (DTRC-RD) - - - - - - - - - - - - - - - - - - - 2-25
2.2.3 故障示踪调出函数 (FTRC-RD) - - - - - - - - - - - - - - - - - - - 2-28
2.2.4 变频器示踪调出函数 (ITRC-RD) - - - - - - - - - - - - - - - - - - 2-32
2.3 变频器函数 - 2-35
2.3.1 变频器常数写入函数 (ICNS-WR) - - - - - - - - - - - - - - - - - - 2-35
2.3.2 变频器常数调出函数 (ICNS-RD) - - - - - - - - - - - - - - - - - - 2-40
2.4 其他的函数 - 2-43
2.4.1 计数函数 (COUNTER) - 2-43
2.4.2 先进先出函数 (FINFOUT) - 2-45

2.1　信息函数

2.1.1　信息发送函数 (MSG-SND)

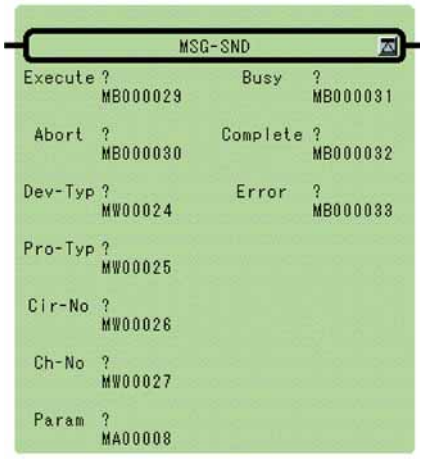
■ 概要

向用传送器件型指定的线路上的对方局发送信息。支持多个协议型。

[传送器件] CPU 模块、215IF、217IF、218IF、SVB-01

[协议] MEMBUS 通信，无序

■ 格式



标记：MSG-SND

全称：Massage Send

类别：SYSTEM

图标： 

■ 参数

输入输出	参数名称	输入输出指定	设定
输入	Execute	B-VAL	信息发送指令
	Abort	B-VAL	信息发送强制中止指令
	Dev-Typ	I-REG	传送器件类别 CPU 模块＝8　215IF ＝1 217IF ＝5　218IF ＝6 SVB-01 ＝11
	Pto-Typ	I-REG	传送协议 存储总线＝1 无序＝2
	Cir-No	I-REG	线路编号 CPU模块＝1，2　215IF＝1～8　217IF＝1～24 218IF ＝1 ～ 8，SVB-01 ＝1 ～ 16
	Ch-No	I-REG	传送缓冲通道编号 CPU 模块＝1，2　215IF ＝1 ～ 13　217IF ＝1 218IF ＝1 ～ 10，SVB-01 ＝1 ～ 8
	Param	地址输入	设定数据首地址 (MW，DW，#W)
输出	Busy	B-VAL	信息发送中
	Complete	B-VAL	信息发送完毕
	Error	B-VAL	发生错误

■ 参数详情

按照参数 No. 将内容、功能等进行了归纳。
下表列出了参数表。

参数 No.	IN/OUT	内容	
		存储总线	无序
PARAM　00	OUT	处理结果	处理结果
PARAM　01	OUT	状态	状态
PARAM　02	IN	对方站 #	对方站 #
PARAM　03	SYS	系统预约	系统预约
PARAM　04	IN	功能代码	
PARAM　05	IN	数据地址	数据地址
PARAM　06	IN	数据大小	数据大小
PARAM　07	IN	对方 CPU#	对方 CPU#
PARAM　08	IN	线圈偏置	
PARAM　09	IN	输入继电器偏置	
PARAM　10	IN	输入寄存器偏置	
PARAM　11	IN	保持寄存器偏置	寄存器偏置
PARAM　12	SYS	系统用	系统用
PARAM　13	SYS	系统预约	系统预约
PARAM　14	SYS	系统预约	系统预约
PARAM　15	SYS	系统预约	系统预约
PARAM　16	SYS	系统预约	系统预约

处理结果 (PARAM00)

向高位字节输出处理结果。低位字节为系统解析用。

- 00xx：处理中 (BUSY)
- 10xx：处理完毕 (COMPLETE)
- 8xxx：发生错误 (ERROR)

  错误分类如下所示。

  - 81xx：功能代码错误

    欲发送未使用的功能代码。或进行了接收。

  - 82xx：地址设定错误

    数据地址、线圈偏置、输入继电器偏置、输入寄存器偏置、保持寄存器偏置均在范围以外。

  - 83xx：数据大小错误

    发送或接收的数据大小的设定在范围以外。

  - 84xx：线路编号设定错误

    线路编号的设定在范围以外。

  - 85xx：通道编号设定错误

    通道编号的设定在范围以外。

  - 86xx：站地址错误

    站编号的设定在范围以外。

  - 88xx：传送部错误

    从传送部返回了错误响应。

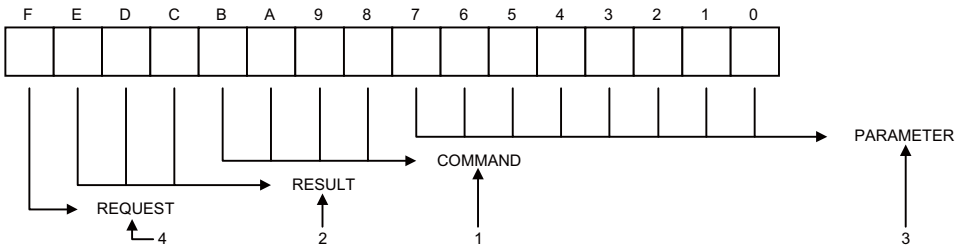
  - 89xx：器件选择错误

    选择了不能使用的器件。

状态 (PARAM01)

输出传送部的状态。

位的分配如下图所示。



1. COMMAND

下表给出了 COMMAND 的一览表。

代码	缩略符号	意义
1	U_SEND	通用信息发送。
2	U_REC	通用信息接收。
3	ABORT	强制中止。
8	M_SEND	发送存储总线指令…接收到响应后完毕。
9	M_REC	接收存储总线指令…同时发送响应。
C	MR_SEND	发送存储总线响应。

2. RESULT

下表给出了 RESULT 的缩略符号和意义。

代码	缩略符号	意义
0	—	执行中。
1	SEND_OK	正常发送完毕。
2	REC_OK	正常接收完毕。
3	ABORT_OK	强制中止完毕。
4	FMT_NG	参数格式错误。
5	SEQ_NG, 或 INIT_NG	指令顺序错误。 尚未接收到令牌。未与传送系统连接。
6	RESET_NG, 或 O_RING_NG	复位状态。 环外。即使超过令牌监视时间也未接收到令牌。
7	REC_NG	数据接收错误（在低位程序中检测到了错误）。

3. PARAMETER

当 RESULT ＝ 4(FMT_NG) 时，显示下表的错误代码。除此以外的时候，则显示对方的站地址。

代码	错误内容
00	无错误
01	站地址范围外
02	存储总线响应接收监视错误
03	再发送次数设定错误
04	循环领域设定错误
05	信息信号 CPU 编号错误
06	信息信号寄存器编号错误
07	信息信号字数错误

4. REQUEST

1 ＝请求

2 ＝受理完毕报告

对方站 #(PARAM02)／串行

1 ～ 254：向指定了器件地址的站发送信息。

功能代码 (PARAM04)

设定发送的存储总线功能代码。
下表列出了功能代码。

功能代码		设定
00H	未使用	×
01H	调出线圈的状态	○
02H	调出输入继电器的状态	○
03H	调出保持寄存器的内容	○
04H	调出输入寄存器的内容	○
05H	单线圈的状态变更	○
06H	向单保持寄存器写入	○
07H	未使用	×
08H	回送测试	○
09H	调出保持寄存器的内容（扩展）	○
0AH	调出输入寄存器的内容（扩展）	○
0BH	向保持寄存器写入（扩展）	○
0CH	未使用	×
0DH	不连续地调出保持寄存器的内容（扩展）	○
0EH	不连续地向保持寄存器写入（扩展）	○
0FH	多个线圈的状态变更	○
10H	向多个保持寄存器写入	○
11H ～ 20H	未使用	×
21H ～ 3FH	系统预约	×
40H ～ 4FH	系统预约	×
50H ～	未使用	×

（注）1. ○：可设定　×：不可设定
2. 主控制器动作时的发送、接收寄存器仅为 MW(MB)。控制器动作时，线圈、保持寄存器、输入继电器、输入寄存器分别以 MB、MW、IB、IW 为对象。

数据地址

根据功能代码，地址的设定范围如下表所示。

功能代码		设定
00H	未使用	无效
01H	调出线圈的状态	0 ～ 65535(0 ～ FFFFH)*
02H	调出输入继电器的状态	0 ～ 65535(0 ～ FFFFH)*
03H	调出保持寄存器的内容	0 ～ 32767(0 ～ 7FFFH)
04H	调出输入寄存器的内容	0 ～ 32767(0 ～ 7FFFH)
05H	单线圈的状态变更	0 ～ 65535(0 ～ FFFFH)*
06H	向单保持寄存器写入	0 ～ 32767(0 ～ 7FFFH)
07H	未使用	无效
08H	回送测试	无效
09H	调出保持寄存器的内容（扩展）	0 ～ 32767(0 ～ 7FFFH)
0AH	调出输入寄存器的内容（扩展）	0 ～ 32767(0 ～ 7FFFH)
0BH	向保持寄存器写入（扩展）	0 ～ 32767(0 ～ 7FFFH)
0CH	未使用	无效
0DH	不连续地调出保持寄存器的内容（扩展）	0 ～ 32767(0 ～ 7FFFH)
0EH	不连续地向保持寄存器写入（扩展）	0 ～ 32767(0 ～ 7FFFH)
0FH	多个线圈的状态变更	0 ～ 65535(0 ～ FFFFH)*
10H	向多个保持寄存器写入	0 ～ 32767(0 ～ 7FFFH)

* 线圈、继电器的调出 / 写入要求：设定数据的首位地址。
寄存器的连续调出 / 写入要求：设定数据的首字地址。
寄存器的不连续调出 / 写入要求：设定地址表的首字地址。

■ 数据大小 (PARAM06)

设定调出或写入请求的数据大小（位数或字数）。设定范围根据功能代码而异。
下表给出了串行数据的大小设定范围。

功能代码		数据大小设定范围	
		215IF/218IF	CPU 模块 / 217IF/SVB-01
00H	未使用	无效	
01H	调出线圈的状态	1 ～ 2000(1 ～ 07D0H) 位	
02H	调出输入继电器的状态	1 ～ 2000(1 ～ 07D0H) 位	
03H	调出保持寄存器的内容	1 ～ 125(1 ～ 007DH) 字	
04H	调出输入寄存器的内容	1 ～ 125(1 ～ 007DH) 字	
05H	单线圈的状态变更	无效	
06H	向单保持寄存器写入	无效	
07H1	未使用	无效	
08H	回送测试	无效	
09H	调出保持寄存器的内容（扩展）	1 ～ 508 (1 ～ 01FCH) 字	1 ～ 252 (1 ～ 00FCH) / 字
0AH	调出输入寄存器的内容（扩展）	1 ～ 508 (1 ～ 01FCH) 字	1 ～ 252 (1 ～ 00FCH) / 字
0CH	未使用	无效	
0DH	不连续地调出保持寄存器的内容（扩展）	1 ～ 508 (1 ～ 01FCH) 字	1 ～ 252 (1 ～ 00FCH) / 字
0EH	不连续地向保持寄存器写入（扩展）	1 ～ 254 (1 ～ 01FEH) 字	1 ～ 126 (1 ～ 007EH) / 字
0FH	多个线圈的状态变更	1 ～ 800(1 ～ 0320H) 位	
10H	向多个保持寄存器写入	1 ～ 100(1 ～ 0064H) 字	

对方 CPU#(PARAM07)

设定对方 CPU 编号的 0。

线圈偏置 (PARAM08)

设定线圈的偏置字地址。当功能代码为 01H、05H、0FH 时有效。

输入继电器偏置 (PARAM09)

设定输入继电器的偏置字地址。当功能代码为 02H 时有效。

输入寄存器偏置 (PARAM10)

设定输入寄存器的偏置字地址。当功能代码为 04H、0AH 时有效。

保持寄存器偏置 (PARAM11)

设定保持寄存器的偏置字地址。功能代码为 03H、06H、09H、0BH、0DH、0EH、10H 时有效。

系统用 (PARAM12)

保持使用中的通道编号。在接通电源时的最初的扫描中，请务必通过用户程序将设定值置为 0000H。其后由于系统要使用该值，请在用户程序中不要变更该设定值。

数据地址、大小、偏置的关系

数据地址、大小以及偏置的关系如下图所示。

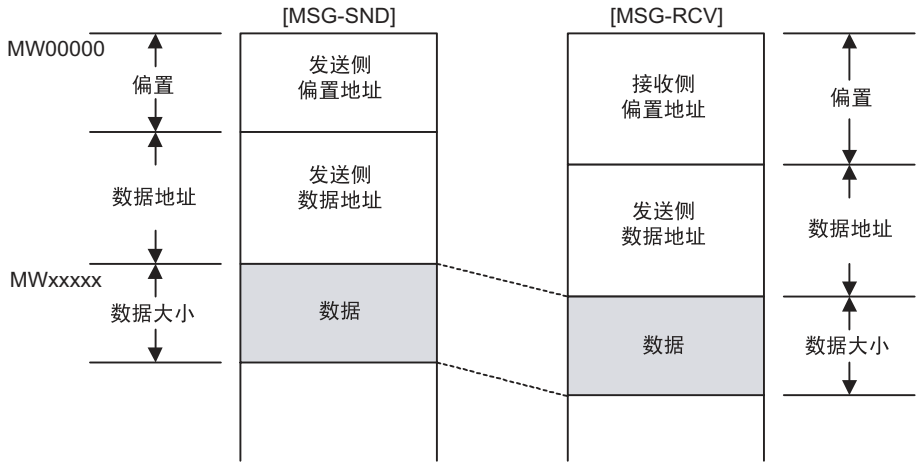


图 2.1 数据地址、大小以及偏置的关系

传送协议＝无序时

不必进行 PARAM04、PARAM09、PARAM10 的设定。可发送的寄存器仅为 MW。PARAM11 为 MW 的偏置字地址。

■ 输入

EXECUTE（发送执行指令）

指令为 “ON”时，进行信息发送。

ABORT（发送强制中止指令）

强制性地中止发送。优先于 EXECUTE（发送执行指令）。

DEV-TYP（传送设备类别）

指定传送器件的类别。

PRO-TYP（传送协议）

指定传送协议。当为无序时，不接收来自对方的响应。
存储总线：设定＝1
无序　：设定＝2

CIR-NO（线路编号）

指定线路编号。

CPU 模块＝1、2，215IF ＝1～8，217IF ＝1～24，218IF ＝1～8，SVB-01 ＝1～16

CH-NO（通道编号）

指定传送部的通道编号。但在指定时应注意对于同一线路，通道编号不得重复。

CPU 模块＝1，215IF ＝1～13，217IF ＝1，218IF ＝1～10，SVB-01 ＝1～8

PARAM（设定数据首地址）

指定设定数据的首地址。

BUSY（处理中）

显示正在处理中。请使 EXECUTE 保持 “ON”。

COMPLETE（处理完毕）

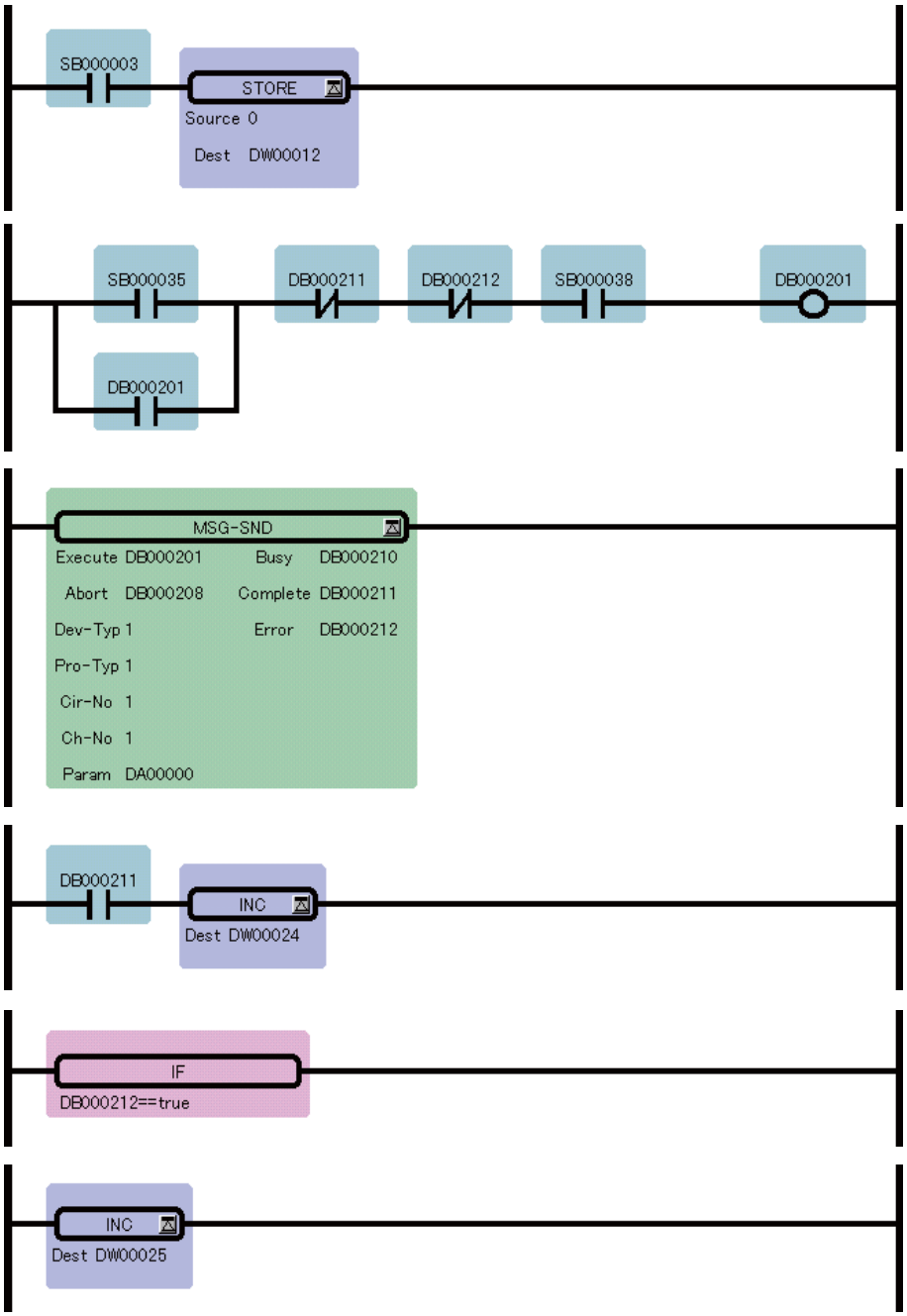
正常结束后，仅 1 个扫描为 “ON”。

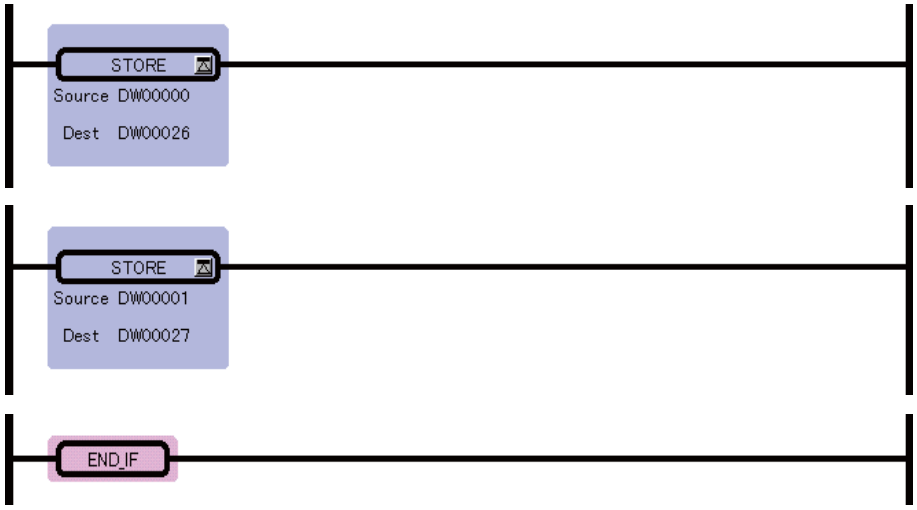
ERROR（发生错误）

发生错误后，仅 1 个扫描为 “ON”。关于原因，请参照 PARAM00 和 PARAM01。

■ 程序举例

下图给出了程序实例。




STORE
Source DW00000
Dest DW00026
STORE
Source DW00001
Dest DW00027
END_IF

2.1.2 信息接收函数（MSG-RCV）

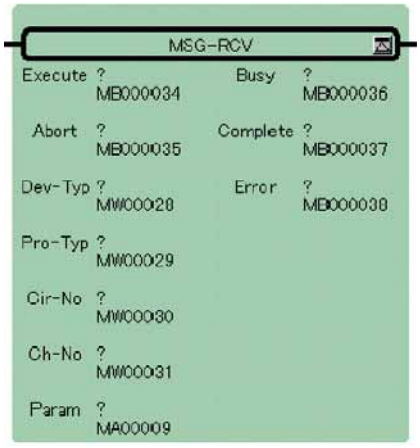
■ 概要

从通过传送器件型指定的线路上的对方局接收信息。支持多个协议型。执行指令(Excute)请在 Complete 或 Error 变为 “ON”以前保持原状。

[传送器件]CPU 模块、215IF、217IF、218IF、SVB-01

[协议] 存储总线，无序

■ 格式



标记：MSG-RCV

全称：Massage Receive

类别：SYSTEM

图标：

■ 参数

输入输出	参数名称	输入输出指定	设定
输入	Execute	B-VAL	信息接收指令
	Abort	B-VAL	信息接收强制中止指令
	Dev-Typ	I-REG	传送器件类别 CPU 模块＝8　215IF ＝1 217IF ＝5　218IF ＝6　SVB-01 ＝11
	Pro-Typ	I-REG	传送协议 (RTU、ASCII 的设定通过模块构成定义来进行) 存储总线＝1　无序＝2
	Cir-No	I-REG	线路编号 CPU 模块＝1　215IF ＝1 ～ 8 217IF ＝1 ～ 24 218IF ＝1 ～ 8，SVB-01 ＝1 ～ 16
	Ch-No	I-REG	传送缓冲通道编号 CPU 模块＝1　215IF ＝1 ～ 13 217IF ＝1　218IF ＝1 ～ 10 SVB-01 ＝1 ～ 8
	Param	地址输入	设定数据首地址 (MW，DW，#W)
输出	Busy	B-VAL	信息发送中
	Complete	B-VAL	信息发送完毕
	Error	B-VAL	发生错误

■ 参数详情

按照参数 No. 将内容、功能等进行了归纳。
下表中列出了参数表。

参数 No.	IN/OUT	内容	
		存储总线	无序
PARAM 00	OUT	处理结果	处理结果
PARAM 01	OUT	状态	状态
PARAM 02	OUT	对方站 #	对方站 #
PARAM 03	SYS	系统预约	系统预约
PARAM 04	OUT	功能代码	
PARAM 05	OUT	数据地址	数据地址
PARAM 06	OUT	数据大小	数据大小
PARAM 07	OUT	对方 CPU#	对方 CPU#
PARAM 08	IN	线圈偏置	
PARAM 09	IN	输入继电器偏置	
PARAM 10	IN	输入寄存器偏置	
PARAM 11	IN	保持寄存器偏置	寄存器偏置
PARAM 12	IN	写入范围 LO	寄存器偏置
PARAM 13	IN	写入范围 HI	寄存器偏置
PARAM 14	SYS	系统用	系统用
PARAM 15	SYS	系统预约	系统预约
PARAM 16	SYS	系统预约	系统预约

处理结果 (PARAM00)

向高位字节输出处理结果。低位字节为系统解析用。

- 00xx：处理中 (BUSY)
- 10xx：处理完毕 (COMPLETE)
- 8xxx：发生错误 (ERROR)

  错误分类如下所示。

  - 81xx：功能代码错误

    接收了未使用的功能代码。

  - 82xx：地址设定错误

    数据地址、线圈偏置、输入继电器偏置、输入寄存器偏置、保持寄存器偏置的设定均在范围以外。

  - 83xx：数据大小错误

    发送或接收的数据大小在范围以外。

  - 84xx：线路编号设定错误

    线路编号的设定在范围以外。

  - 85xx：通道编号设定错误

    通道编号的设定在范围以外。

  - 86xx：站地址错误

    站编号的设定在范围以外。

  - 88xx：传送部错误

    从传送部返回了错误响应。

  - 89xx：器件选择错误

    选择了不能使用的器件。

状态 (PARAM01)

输出传送部的状态。详情请参照 “2.1.1 信息发送函数 (MSG-SND)” 中的参数的状态 (PARAM01)。

对方站 #(PARAM02)

输出发送源的站编号。

功能代码（PARAM04）

输出接收的存储总线功能代码。
下表列出了功能代码。

功能代码		设定
00H	未使用	×
01H	调出线圈的状态	○
02H	调出输入继电器的状态	○
03H	调出保持寄存器的内容	○
04H	调出输入寄存器的内容	○
05H	单线圈的状态变更	○
06H	向单保持寄存器写入	○
07H	未使用	×
08H	回送测试	○
09H	调出保持寄存器的内容（扩展）	○
0AH	调出输入寄存器的内容（扩展）	○
0BH	向保持寄存器写入（扩展）	○
0CH	未使用	×
0DH	不连续地调出保持寄存器的内容（扩展）	○
0EH	不连续地向保持寄存器写入（扩展）	○
0FH	多个线圈的状态变更	○
10H	向多个保持寄存器写入	○
11H ～ 20H	未使用	×
21H ～ 3FH	系统预约	×
40H ～ 4FH	系统预约	×
50H ～	未使用	×

（注）从控制器动作时，线圈、保持寄存器、输入继电器、输入寄存器分别以 MB、MW、IB、IW 为对象。

数据地址（PARAM05）

发送侧所请求的数据地址被输出。

数据大小（PARAM06）

调出或写入请求的数据大小（位数或字数）被输出。

对方 CPU#(PARAM07)

对方 CPU 编号的 0（固定）被输出。

线圈偏置 (PARAM08)

设定线圈的偏置字地址。
当功能代码为 01H、05H、0FH 时有效。

输入继电器偏置 (PARAM09)

设定输入继电器的偏置字地址。
当功能代码为 02H 时有效。

输入寄存器偏置 (PARAM10)

设定输入寄存器的偏置字地址。
当功能代码为 04H、0AH 时有效。

保持寄存器偏置 (PARAM11)

设定保持寄存器的偏置字地址。
当功能代码为 03H、06H、09H、0BH、0DH、0EH、10H 时有效。

写入范围 LO(PARAM12)、写入范围 HI(PARAM13)

设定对于写入请求的写入许可范围。不在该范围内的请求即为错误。当功能代码为 0BH、0EH、0FH、10H 时有效。

0 ≦写入范围 LO ≦写入范围 HI ≦ MW 地址的最大值

系统用 (PARAM14)

保持使用中的通道编号。在接通电源时的最初的扫描中，请务必通过用户程序将设定值置为 0000H。其后由于系统要使用该值，请在用户程序中不要变更该设定值。

传送协议＝无序时

无需进行 PARAM 04、PARAM 08、PARAM 09、PARAM 10、PARAM 11 的设定。
PARAM 12 与写入目标中的 MW 的偏置字地址兼用。

■ 输入

EXECUTE（接收执行指令）

指令为 “ON”时，进行信息发送。
在 COMPLETE（处理完毕）或 ERROR（发生错误）变为 “ON”以前需要保持。

ABORT（接收强制中止指令）

强制性地中止发送。优先于 EXECUTE（接收执行指令）。

DEV-TYP（传送器件类别）

指定传送器件的类别。
CPU 模块＝ 8，215IF ＝ 1，217IF ＝ 5，218IF ＝ 6，SVB-01 ＝ 11

PRO-TYP（传送协议）

指定传送协议。当为无序时，不向对方局发送响应。

存储总线：设定＝ 1
无序　：设定＝ 2

CIR-NO（线路编号）

指定线路编号。

CPU 模块＝ 1、2，215IF ＝ 1 ～ 8，217IF ＝ 1 ～ 24，218IF ＝ 1 ～ 8，SVB-01 ＝ 1 ～ 16

CH-NO（通道编号）

指定传送部的通道编号。但在指定时应注意对于同一线路，通道编号不得重复。

CPU 模块＝ 1，215IF ＝ 1 ～ 13，217IF ＝ 1，218IF ＝ 1 ～ 10，SVB-01 ＝ 1 ～ 8

PARAM（设定数据首地址）

指定设定数据的首地址。设定数据的详情请参照 “2.1.2 信息接收函数 (MSG-RCV)” 中的 “■ 参数详情”。

■ 输出

BUSY（处理中）

表示正在处理中。请使 EXECUTE 保持 “ON”。

COMPLETE（处理完毕）

正常结束后，仅 1 个扫描为 “ON”。

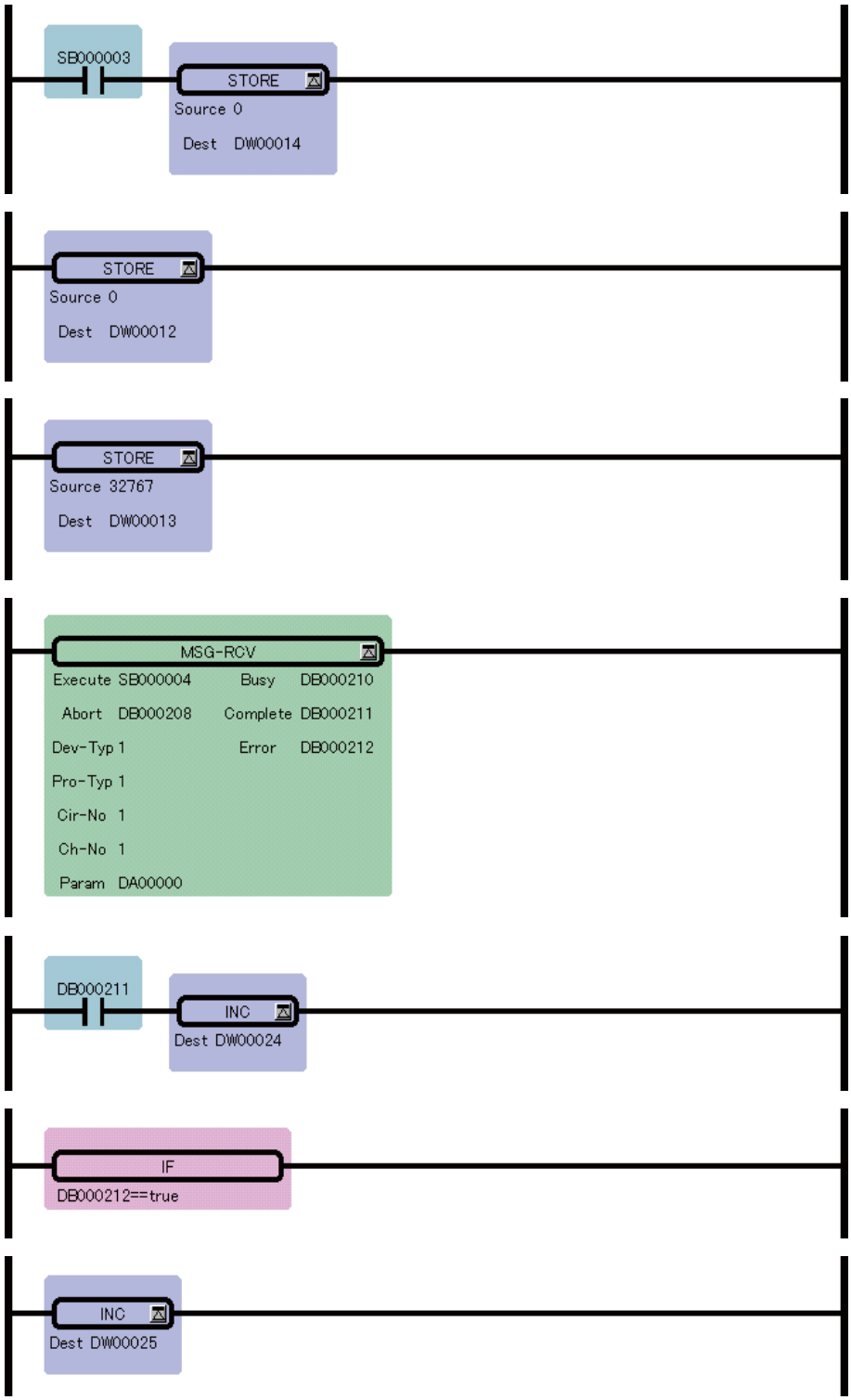
ERROR（发生错误）

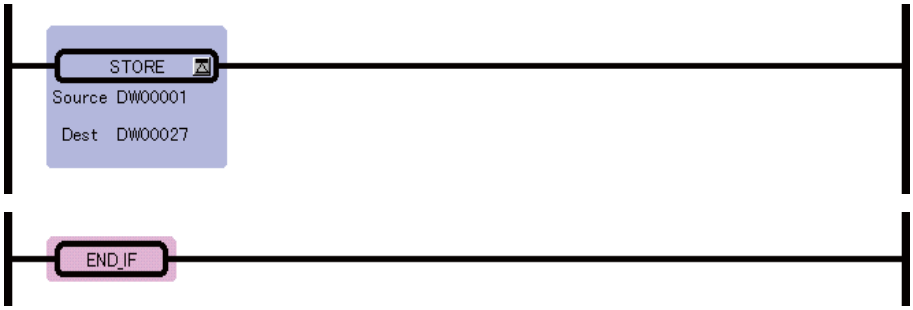
发生错误后，仅 1 个扫描为 “ON”。

关于原因，请参照上述 “■ 参数详情”中的 PARAM00 和 PARAM01。

■ 程序举例

下图给出了程序实例。




STORE
Source DW00001
Dest DW00027
END_IF

2.2　示踪函数

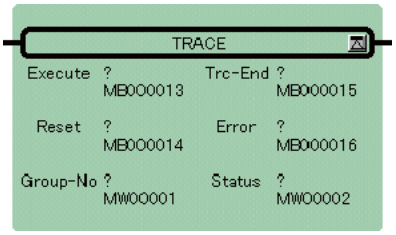
2.2.1　示踪函数（TRACE）

■ 概要

对用示踪组 No. 指定的示踪数据进行示踪执行控制。示踪定义通过“数据示踪定义”来执行。

- 示踪执行指令（EXECUTE）为 ON 时，执行示踪。
- 示踪复位指令（RESET）为 ON 时，示踪次数计数器被复位。此时示踪结束，（TRC-END）也被复位。
- 当示踪执行次数达到设定的次数时（用示踪定义设定），示踪结束，（TRC-END）变为 ON。

■ 格式



标记：TRACE
全称：Trace
类别：SYSTEM
图标： 

■ 参数

输入输出	参数名称	输入输出指定	设定
输入	Execute	B-VAL	示踪执行指令
	Reset	B-VAL	示踪复位指令
	Group-No	I-REG	示踪组 No. 的指定（1 ～ 4）
输出	Trc-End	B-VAL	示踪结束
	Error	B-VAL	发生错误
	Status	I-REG	示踪执行状态

下表给出了示踪执行状态（STATUS）的构成

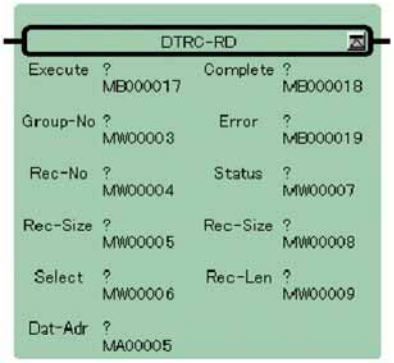
名称	Bit 名称	备考
示踪数据满	Bit 0	将指定组的数据示踪内存内旋转一圈后变为 ON。
系统预约	Bit 1 ～ Bit 7	
无示踪定义	Bit8	函数未被执行。
指定组 No. 错误	Bit9	函数未被执行。
系统预约	Bit 10 ～ Bit 12	
执行时间错误	Bit13	函数未被执行。
系统预约	Bit14	
系统预约	Bit15	

2.2.2 数据示踪调出函数 (DTRC-RD)

■ 概要

调出控制器本身的示踪数据，将其存储在用户寄存器中。指定记录编号、记录数后，可调出示踪内存内的数据。仅指定记录内的必要项目也可调出。

■ 格式



标记：DTRC-RD
全称：Data-Trace Read
类别：SYSTEM
图标： 

■ 参数

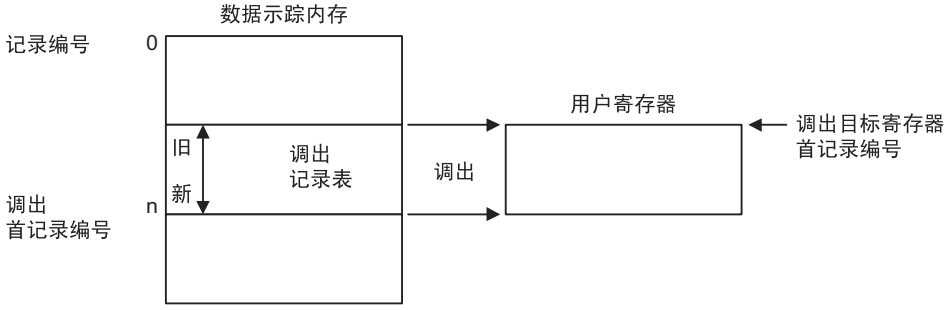
输入输出	参数名称	输入输出指定	设定
输入	Execute	B-VAL	数据示踪读取执行指定
	Group-No	I-REG	数据示踪组 No. 的指定 (1 ～ 4)
	Rec-No	I-REG	调出首记录 No. 的指定 (0 ～最大记录数－ 1)
	Rec-Size	I-REG	调出请求记录 No. 的指定 (0 ～最大记录数－ 1)
	Select	I-REG	调出项目 (0001H ～ FFFFH) 位0～F与示踪定义的数据指定1～16相对应
	Dat-Adr	地址输入	调出首寄存器 No. 的指定 (MW、DW 的地址)
输出	Complete	B-VAL	示踪读取结束
	Error	B-VAL	发生错误
	Status	I-REG	数据示踪读取执行状态
	Rec-Size	I-REG	调出记录数
	Rec-Len	I-REG	调出 1 个记录长度 (字)

表 2.1　数据示踪读取执行状态 (STATUS) 的构成

名称	位编号	备考
系统预约	bit0 ～ bit7	
无示踪定义	bit8	函数未被执行。
组 No. 错误	bit9	函数未被执行。
指定记录 No. 错误	bit10	
指定记录数错误	bit11	函数未被执行。
数据存储错误	bit12	函数未被执行。
系统预约	bit13	
系统预约	bit14	
地址输入错误	bit15	函数未被执行。

■ 数据的调出

如下图所示进行数据的调出。

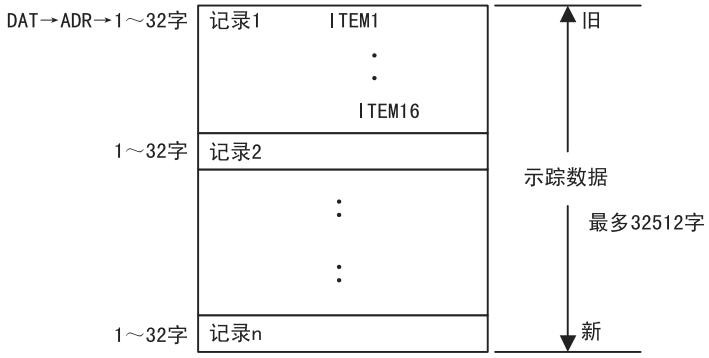


示踪组的最新记录编号如下表所示，分别被存储在 SW00100 ～ SE00103 中。

系统寄存器编号	数据示踪定义
SW00100	组 1 用
SW00101	组 2 用
SW00102	组 3 用
SW00103	组 4 用
SW00104	—
SW00105	—
SW00106	—
SW00107	—

■ 调出数据的构成

调出数据的构成如下图所示。



记录长度

记录由所选项目的数据构成。

1 记录的字长度＝ Bn × 1 字＋ Wn × 1 字＋ Ln × 2 字＋ Fn × 2 字
Bn：比特型寄存器的选择个数
Wn：字型寄存器的选择个数
Ln：长整型寄存器的选择个数
Fn：实型寄存器的选择个数

记录长度最大＝ 32 字（长整型寄存器或实型寄存器为 16 个时）

记录长度最小＝ 1 字（比特型或整型寄存器为 1 个时）

记录数

记录数如下表所示。

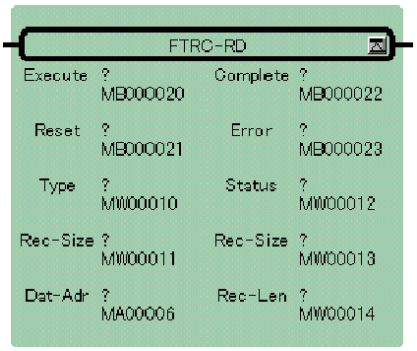
最大记录数	32512/ 记录数
记录长度最大时的记录数	0 ～ 1015
记录长度最小时的记录数	0 ～ 32511

2.2.3　故障示踪调出函数（FTRC-RD）

■ 概要

调出故障示踪数据，将其存储在用户寄存器中。可从示踪缓冲中指定并调出所需的记录数。指定并调出故障发生数据或修复数据中的任一个。可对故障示踪缓冲进行复位（初始化）。

■ 格式



标记：FTRC-RD
全称：Failure-Trace Read
类别：SYSTEM
图标： 

■ 参数

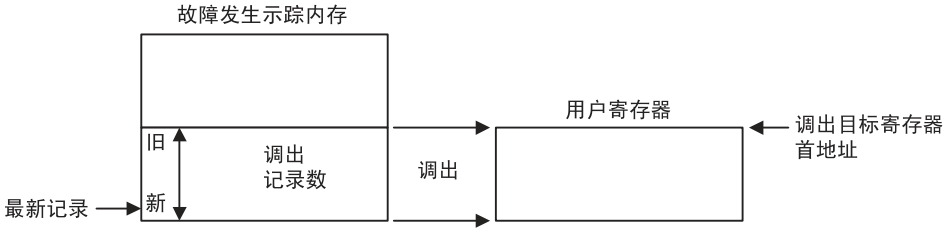
输入输出	参数名称	输入输出指定	设定
输入	Execute	B-VAL	故障示踪调出指令
	Reset	B-VAL	故障示踪缓冲复位指令
	Type	I-REG	调出数据的类型 1：发生的数据 2：修复的数据
	Rec-Size	I-REG	调出记录数 发生：1～64　　修复：450
	Dat-Adr	地址输入	调出的首寄存器地址 （MW、DW 的地址）
输出	Complete	B-VAL	故障示踪调出指令结束
	Error	B-VAL	发生错误
	Status	I-REG	故障示踪调出指令执行状态
	Rec-Size	I-REG	调出记录数
	Rec-Len	I-REG	调出记录长度

下表给出了变频器常数写入执行状态 (STATUS) 的构成。

名称	Bit 名称	备考
系统预约	Bit 0 ～ Bit 7	
无示踪定义	Bit8	函数未被执行。
指定类型 No. 错误	Bit9	函数未被执行。
系统预约	Bit10	
指定记录数错误	Bit11	函数未被执行。
数据存储错误	Bit12	函数未被执行。
系统预约	Bit13	
系统预约	Bit14	
系统预约	Bit15	函数未被执行。

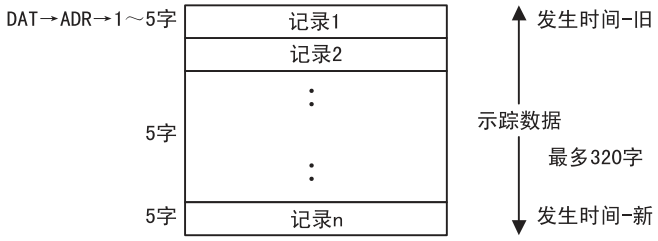
■ 故障发生数据的调出

如下图所示进行故障发生数据的调出。始终从最新的记录开始调出。

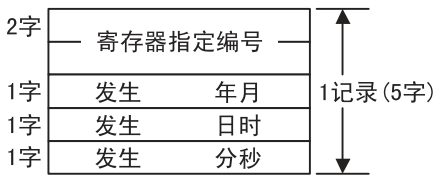


■ 调出数据的构成（故障发生数据）

数据构成



记录构成



寄存器指定编号的构造 (2 字)

有故障检测继电器信息。

下表给出了位的构成。

No.	① 的位构成	② 的位构成
7	已定义的标识 (1 ＝已定义，0 ＝未定义)	系统预约 (＝ 0)
6 5 4	系统预约 (＝ 0) 0 ＝ A 触点指定，1 ＝ B 触点指定	数据类型 位＝ 0， 整数＝ 1， 长整数＝ 2， 实数＝ 3
3 2 1 0	寄存器的种类 S ＝ 0， I ＝ 0， O ＝ 0， M ＝ 3	位地址 0 ～ F

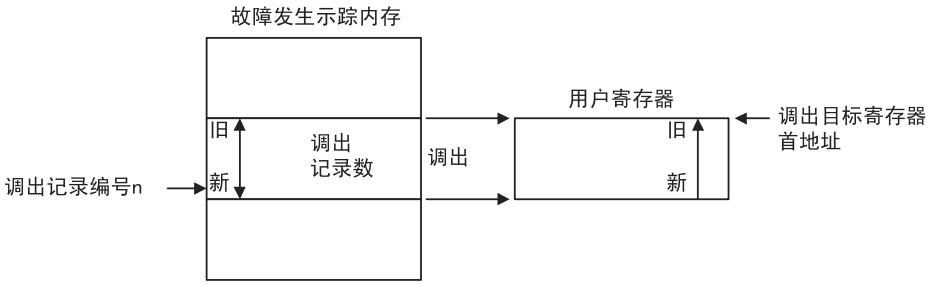
记录数

记录数如下表所示。

最小记录数	0(无故障发生数据)
最大记录数	64

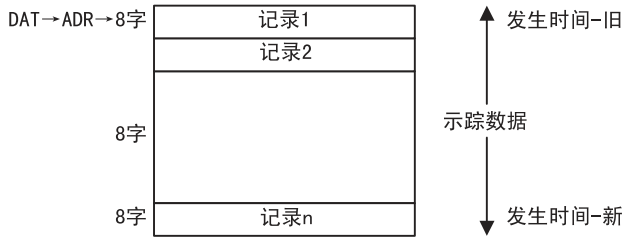
■ 故障修复数据的调出

故障修复数据的调出如下图所示。修复数据被存储在 SW00093(1 ～ 9999 的环形计数器) 中。



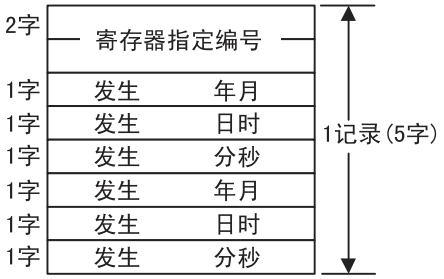
■ 调出数据（故障修复数据）的构成

下图给出了数据的构成。



记录构成

下图给出了记录的构成。



记录数

记录数如下所示。

最小记录数	0（无故障发生数据）
最大记录数	450

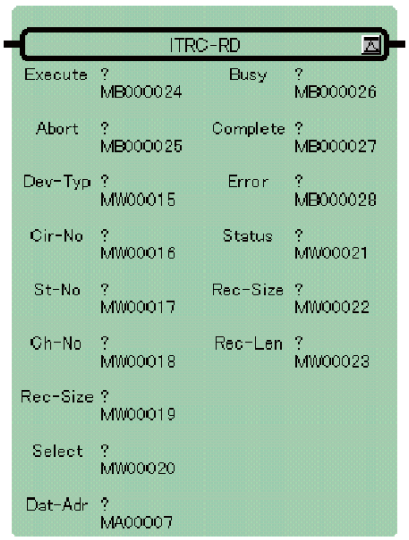
2.2.4　变频器示踪调出函数（ITRC-RD）

■ 概要

调出变频器的示踪数据，将其存储在用户寄存器中。可从示踪缓冲中指定并调出所需的记录数。仅指定记录内的必要项目也可调出。

＜对象变频器＞　连接 MP930，SVB-01，215IF 的变频器

■ 格式



标记：ITRC-RD

全称：Inverter-Trace Read

类别：SYSTEM

图标：

■ 参数

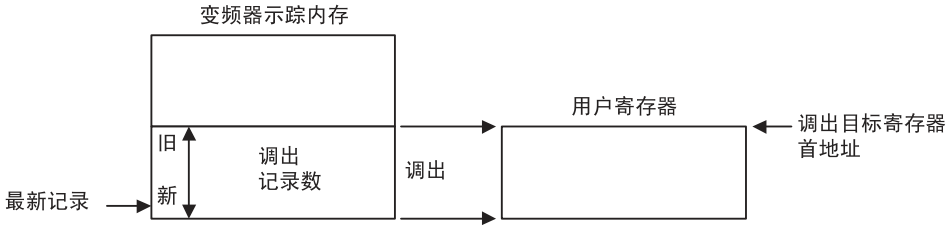
输入输出	参数名称	输入输出指定	设定
输入	Execute	B-VAL	变频器示踪调出指令
	Abort	B-VAL	变频器示踪调出强制中止指令
	Dev-Typ	I-REG	传送器件类别 215IF ＝ 1　MP930 ＝ 4　SVB-01 ＝ 11
	Cir-No	I-REG	线路编号 215IF ＝ 1，2　MP930 ＝ 1　SVB-01 ＝ 1 ～ 16
	St-No	I-REG	子局编号 215IF ＝ 1 ～ 64　MP930 ＝ 1 ～ 14 SVB-01 ＝ 1 ～ 14
	Dh-No	I-REG	传送缓冲通道编号 215IF ＝ 1 ～ 3　MP930 ＝ 1　SVB-01 ＝ 1 ～ 8
	Rec-Size	I-REG	调出记录数 (1 ～ 64)
	Select	I-REG	调出项目 (0001H ～ FFFFH) 位 0 ～ F 与示踪数据项目 1 ～ 26 相对应
	Dat-Adr	地址输入	数据缓冲寄存器首寄存器地址 (MW、DW 的地址)
输出	Busy	B-VAL	变频器示踪调出中
	Complete	B-VAL	变频器示踪调出结束
	Error	B-VAL	发生错误
	Status	I-REG	变频器示踪调出执行状态
	Rec-Size	I-REG	调出记录数
	Rec-Len	I-REG	调出记录长度

下表给出了变频器常数调出执行状态 (STATUS) 的构成。

名称	Bit 名称	备考
系统预约	Bit 0 ～ Bit 8	
传送参数错误	Bit9	函数未被执行。
系统预约	Bit10	
指定记录数错误	Bit11	函数未被执行。
数据存储错误	Bit12	函数未被执行。
传送错误	Bit13	函数未被执行。
系统预约	Bit14	
地址输入错误	Bit15	函数未被执行。

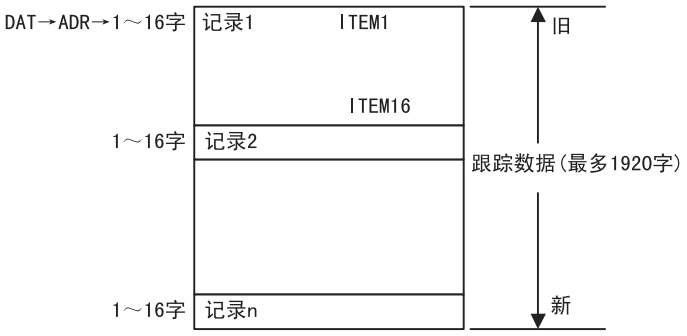
■ 数据的调出

始终从最新的记录开始调出。



■ 调出数据的构成

数据构成



记录长度

记录由所选项目的数据构成。
1 个记录的字长＝ 1 ～ 16 字

记录数

最大记录数＝ 120

2.3 变频器函数

2.3.1 变频器常数写入函数（ICNS-WR）

■ 概要

写入变频器常数。可指定写入对象的变频器常数的种类与范围。

＜对象变频器＞ 连接 MP930，SVB-01，215IF 的变频器

■ 格式

标记：ICNS-WR

全称：Inverter-Constant Write

类别：SYSTEM

图标： 

■ 参数

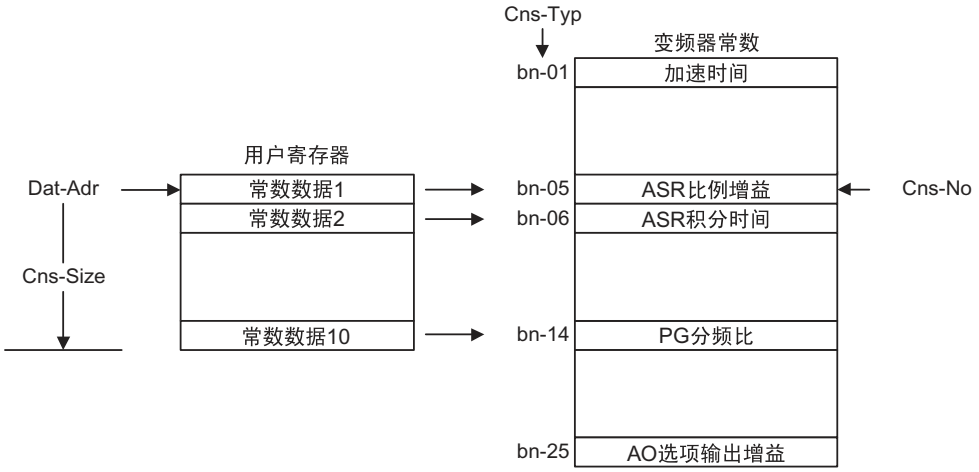
输入输出	参数名称	输入输出指定	设定
输入	Execute	B-VAL	变频器常数写入指令
	Abort	B-VAL	变频器常数写入强制中止指令
	Dev-Typ	I-REG	传送器件类别 215IF ＝ 1　MP930 ＝ 4　SVB-01 ＝ 11
	Cir-No	I-REG	线路编号 215IF ＝ 1，2　MP930 ＝ 1　SVB-01 ＝ 1 ～ 16
	St-No	I-REG	子局编号 215IF ＝ 1 ～ 64　MP930 ＝ 1 ～ 14 SVB-01 ＝ 1 ～ 14
	Ch-No	I-REG	传送缓冲通道编号 215IF ＝ 1 ～ 3　MP930 ＝ 1　SVB-01 ＝ 1 ～ 8
	Cns-Typ	I-REG	变频器常数的种类 0 ＝参考编号直接指定，1 ＝ An， 2 ＝ Bn，3 ＝ Cn，4 ＝ Dn，5 ＝ En，6 ＝ Fn， 7 ＝ Hn，8 ＝ Ln，9 ＝ On，10 ＝ Tn
	Cns-No	I-REG	变频器常数的编号（1 ～ 99） 上限根据变频器的种类而不同。 当 CNS-TYP ＝ 0 时，指定参考编号
	Cns-Size	I-REG	变频器常数的写入数量 （写入数据数量）1 ～ 100
	Dat-Adr	地址输入	设定数据的寄存器地址 （MW、DW、#W 的地址）
输出	Busy	B-VAL	变频器常数写入中
	Complete	B-VAL	变频器常数写入结束
	Error	B-VAL	发生错误
	Status	I-REG	变频器常数写入执行状态

下表给出了变频器常数写入执行状态（STATUS）的构成。

名称	Bit 名称	备考
系统预约	Bit 0 ～ Bit 7	
执行顺序错误	Bit8	函数未被执行。
传送参数错误	Bit9	函数未被执行。
指定种类错误	Bit10	函数未被执行。
指定编号错误	Bit11	函数未被执行。
指定数据错误	Bit12	函数未被执行。
传送错误	Bit13	函数未被执行。
变频器响应错误	Bit14	函数未被执行。
地址输入错误	Bit15	函数未被执行。

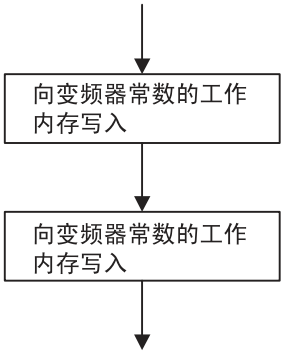
（注）当为变频器响应错误时，在 Bit 0 ～ Bit 7 中表示来自变频器的
错误代码。
01H(1) ：功能代码错误
02H(2) ：参考编号错误
03H(3) ：写入个数错误
21H(33)：写入数据上下限错误
22H(34)：写入异常（运行中、UV 中）

■ 写入数据的构成

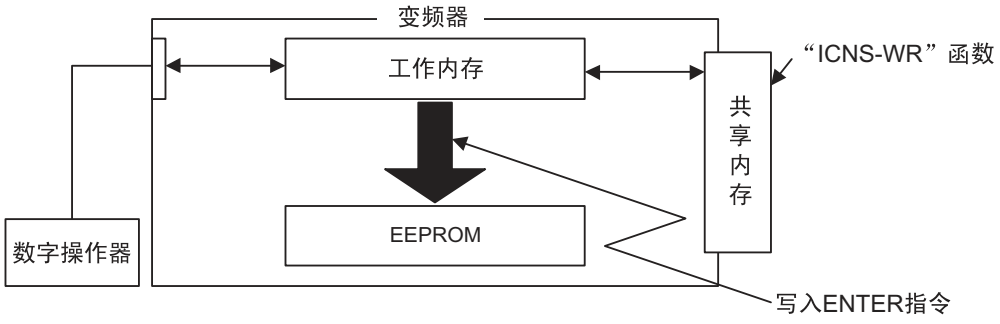


■ EEPROM 的写入方法

向 EEPROM（变频器内部的常数存储内存）写入常数的步骤如下图所示。



用系统函数 「ICNS-WR」写入的常数，先被存储在变频器的工作内存中。要将其实际写入 EEPROM 中时，需要发出下图所示的写入 ENTER 指令。

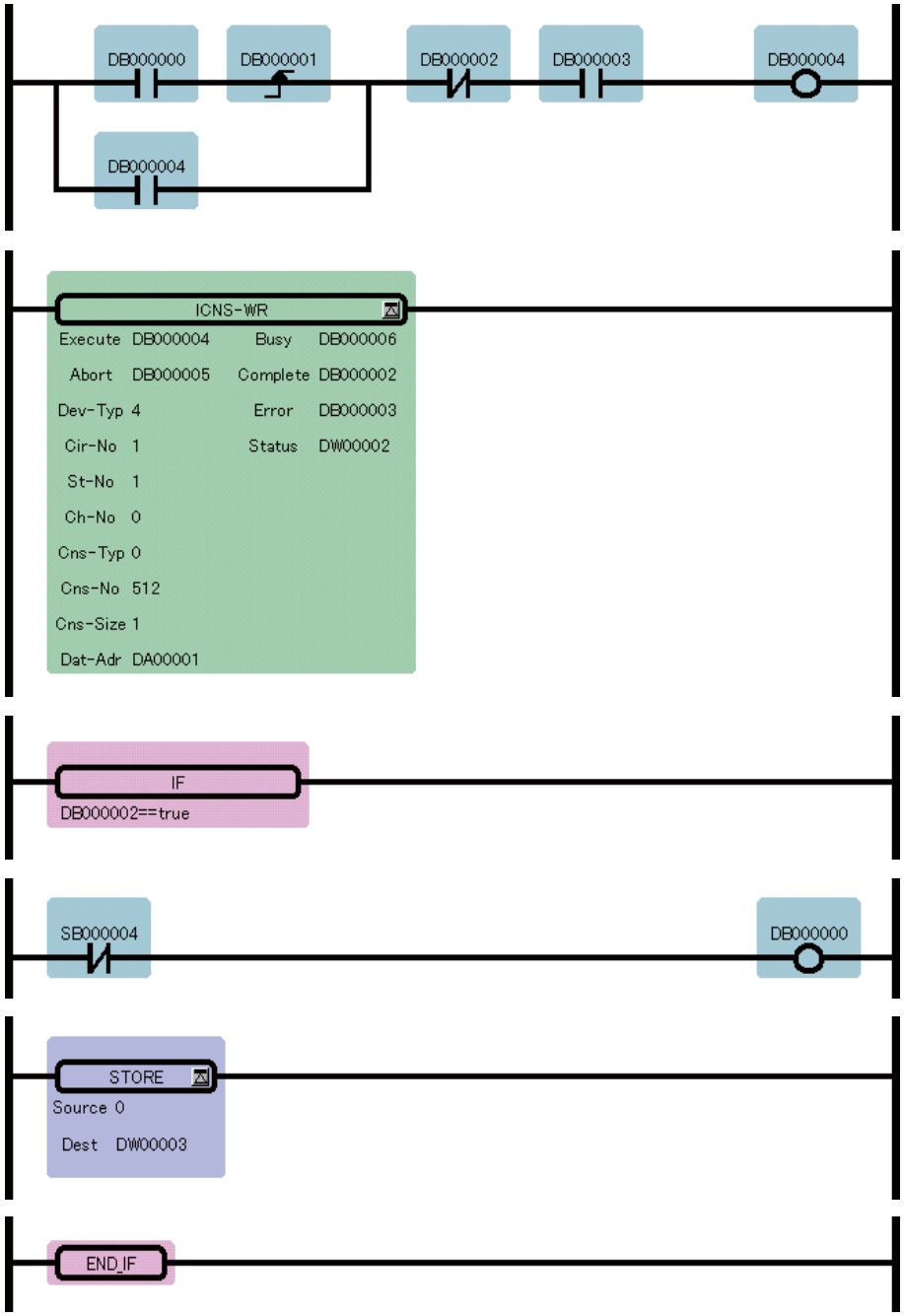


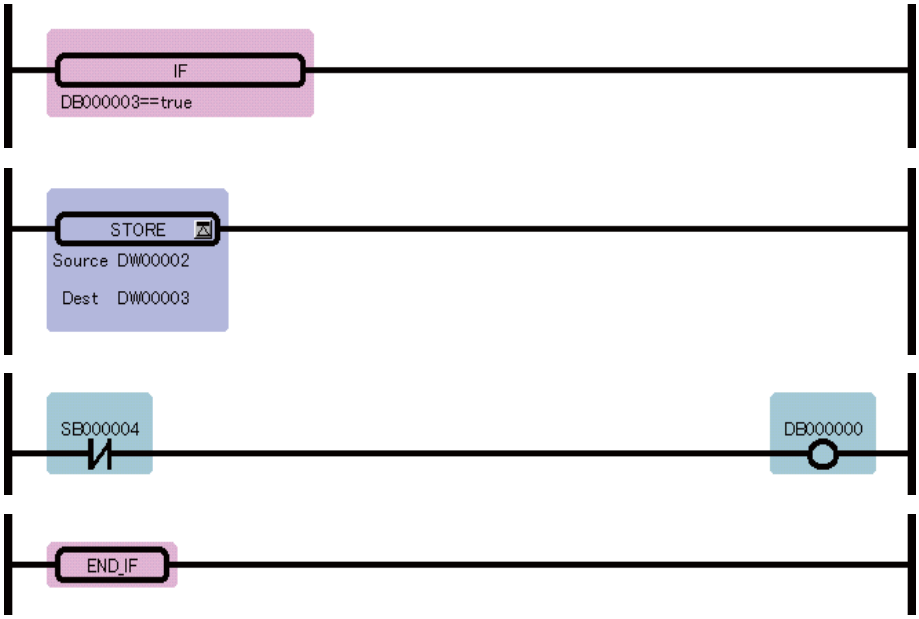
写入 ENTER 指令

利用 INCS-WR 函数，通过在参考编号 “FEED”中写入数据 “0”，可对变频器发送出写入 ENTER 指令。

■ 程序举例

向常数 “CI-01”中写入 “200”的程序实例（MP930 时）如下图所示。




IF
DB000003==true
STORE
Source DW00002
Dest DW00003
SB000004
DB000000
END_IF

2.3.2　变频器常数调出函数（ICNS-RD）

■ 概要

调出变频器常数。可指定调出对象的变频器常数的种类与范围。

■ 格式

标记：ICNS-RD
全称：Inverter-Constant Read
类别：SYSTEM
图标： 

■ 参数

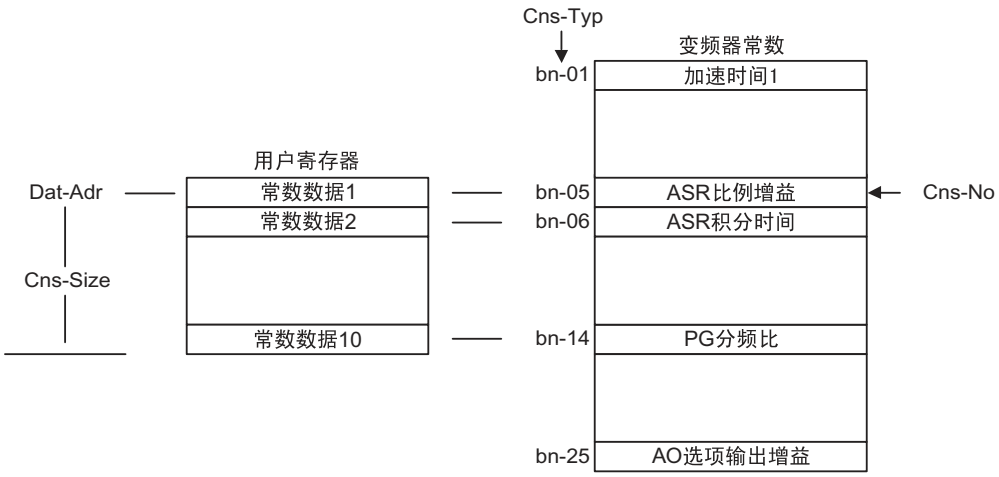
输入输出	参数名称	输入输出指定	设定
输入	Execute	B-VAL	变频器常数调出指令
	Abort	B-VAL	变频器常数调出强制中止指令
	Dev-Typ	I-REG	传送器件类别 215IF ＝ 1　MP930 ＝ 4　SVB-01 ＝ 11
	Cir-No	I-REG	线路编号 215IF ＝ 1、2　MP930 ＝ 1 SVB-01 ＝ 1 ～ 16
	St-No	I-REG	子局编号 215IF ＝ 1 ～ 64　MP930 ＝ 1 ～ 14 SVB-01 ＝ 1 ～ 14
	Dh-No	I-REG	传送缓冲通道编号 215IF ＝ 1 ～ 3　MP930 ＝ 1 SVB-01 ＝ 1 ～ 8
	Cns-Typ	I-REG	变频器常数的种类 0 ＝参考编号直接指定，1 ＝ An，2 ＝ Bn， 3 ＝ Cn，4 ＝ Dn，5 ＝ En，6 ＝ Fn，7 ＝ Hn， 8 ＝ Ln，9 ＝ On，10 ＝ Tn
	Cns-No	I-REG	变频器常数的编号（1 ～ 99） 上限根据变频器的种类而不同。 当 CNS-TYP ＝ 0 时，指定参考编号
	Cns-Size	I-REG	变频器常数的写入数量 （写入数据数量）1 ～ 100
	Dat-Adr	地址输入	设定数据的寄存器地址 （MW，DW，#W 的地址）
输出	Busy	B-VAL	变频器常数调出中
	Complete	B-VAL	变频器常数调出结束
	Error	B-VAL	发生错误
	Status	I-REG	变频器常数调出执行状态

下表给出了变频器常数调出执行状态（STATUS）的构成。

名称	Bit 名称	备考
系统预约	Bit 0～Bit 7	
执行顺序错误	Bit8	
传送参数错误	Bit9	函数未被执行。
指定种类错误	Bit10	函数未被执行。
指定编号错误	Bit11	函数未被执行。
指定数据错误	Bit12	函数未被执行。
传送错误	Bit13	函数未被执行。
变频器响应错误	Bit14	函数未被执行。
地址输入错误	Bit15	函数未被执行。

（注）当为变频器响应错误时，在 Bit 0～Bit 7 中表示来自变频器的错误代码。
01H(1)：功能代码错误
02H(2)：参考编号错误
（ ）内为 10 进制表示

■ 调出数据的构成



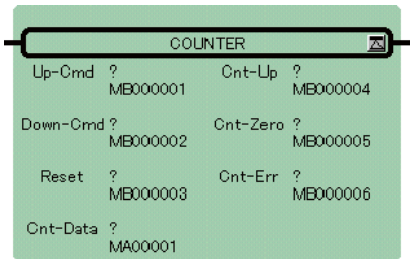
2.4 其他的函数

2.4.1 计数函数（COUNTER）

■ 概要

当正计数、倒计数（Up-cmd，Down-cmd）为 OFF → ON 时，对当前值进行正计数倒计数。当计数器复位指令（RESET）变为 ON 时，计数器将当前值清除为零。另外，将计数器的当前值与设定值进行比较，并将其结果输出。当发生计数器错误（当前值＞设定值时）时，不再对当前值进行正倒计数。

■ 格式



标记：COUNTER
全称：Counter
类别：SYSTEM
图标： 

■ 参数

输入输出	参数名称	输入输出指定	设定	
输入	Up-Cmd	B-VAL	正计数指令 (OFF → ON)	计数器处理用数据域 1：设定值 2：当前值 3：工件标识
	Down-Cmd	B-VAL	倒计数指令 (OFF → ON)	
	Reset	B-VAL	计数器复位指令	
	Cnt-Data	地址输入	计数器处理用数据域首地址（MW、DW 寄存器）	
输出	Cnt-Up	B-VAL	计数器当前值＝设定值时 ON	
	Cnt-Zero	B-VAL	计数器当前值＝ 0 时 ON	
	Cnt-Err	B-VAL	计数器当前值＞设定值时 ON	

输入输出参数的数据格式如下表所示。

输入数据格式	输入指定	说明
位输入	B-VAL	输入指定为比特型。 该比特型数据为函数的输入。
整型输入	I-VAL	输入指定为整型。 指定了寄存器编号的内容（整型数据）为函数的输入。
	I-REG	输入指定为整型寄存器编号的内容。调用函数时，指定整型寄存器编号。 指定了寄存器编号的内容（整型数据）为函数的输入。
长整型输入	L-VAL	输入指定为长整型。 调用函数时，指定了寄存器编号的内容（长整型数据）为函数的输入。
	L-REG	输入指定为长整型寄存器编号的内容。 调用函数时，长整型寄存器编号的内容（长整型数据）为函数的输入。
实型输入	F-VAL	输入指定为实型。 指定了寄存器编号的内容（实型数据）为函数的输入。
	F-REG	输入指定为实型寄存器编号的内容。 调用函数时，指定实型寄存器编号。 指定了寄存器编号的内容（实型数据）为函数的输入。
地址输入	—	将指定寄存器（任意的整型寄存器）的地址传给函数。用户函数时仅1个输入。

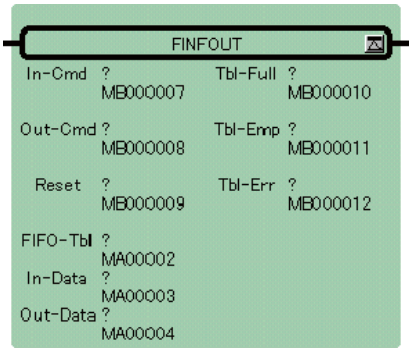
2.4.2 先进先出函数（FINFOUT）

■ 概要

这是先进先出型的块数据转发函数。FIFO 数据表由 4 个字的标题部和数据缓冲构成。请在调出本函数前设定标题部的 3 个字（数据大小、输入大小、输出大小）。

- 数据输入指令（IN-CMD）为 ON 时，可按顺序将指定了个数的数据从指定的输入数据域存储到 FIFO 表的数据域中。
- 数据输出指令（OUT-CMD）为 ON 时，从 FIFO 表的数据域前部将指定了个数的数据转发到指定的输出数据域。
- 当复位指令（RESET）为ON时，数据存储数被清除为零，FIFO表清空（TBL-EMP）为ON。
- 当 “数据可用大小＜输入大小”，或 “数据大小＜输出大小”时，FIFO 表错误（TBL-ERR）为 ON。

■ 格式



标记：FINFOUT

全称：First-in First-out

类别：SYSTEM

图标：

■ 参数

输入输出	参数名称	输入输出指定	设定	
输入	In-Cmd	B-VAL	数据输入指令	FIFO 表的构成 0：数据大小 1：输入大小 2：输出大小 3：数据存储数量 4：数据
	Out-Cmd	B-VAL	数据输出指令	
	Reset	B-VAL	复位指令	
	FIFO-Tbl	地址输入	FIFO 表首地址 （MW、DW 的地址）	
	In-Data	地址输入	输入数据首地址 （MW、DW 的地址）	
	Out-Data	地址输入	输出数据首地址 （MW、DW 寄存器）	
输出	Tbl-Full	B-VAL	FIFO 表满	
	Tbl-Emp	B-VAL	FIFO 表空	
	Tbl-Err	B-VAL	FIFO 表错误	

附录 A

EXPRESSION

在梯形图程序里表述的指令中，需要在 IF、WHILE、EXPRESSION 指令中表述条件表达式和运算表达式。这些数式可通过 EXPRESSION 来表述。本章对 EXPRESSION 的使用规则进行了说明。

A.1 数式 ---------- A-2

A.1.1 运算符 ---------- A-2

A.1.2 运算对象 ---------- A-4

A.1.3 函数 ---------- A-4

A.2 可识别的表达式种类 ---------- A-5

A.2.1 算术运算符 ---------- A-5

A.2.2 比较运算符 ---------- A-5

A.2.3 逻辑运算符 ---------- A-5

A.2.4 赋值运算符 ---------- A-6

A.2.5 函数 ---------- A-6

A.2.6 其他 ---------- A-6

A.3 在梯形图程序中的应用 ---------- A-7

A.3.1 IF 指令句的条件表达式 ---------- A-7

A.3.2 WHILE 指令句的条件表达式 ---------- A-7

A.3.3 EXPRESSION 指令句的运算表达式 ---------- A-7

A.1　数式

数式由运算符、运算对象（常数和变量）及函数构成。一个数式的结束用分号 “;” 来表示。用 “(”, “)” 括号连接数式。本节对数式的构成要素进行了说明。

A.1.1　运算符

■ 可使用的运算符

可使用的运算符有下列几种。

算数运算符

＋	加法
-	减法
*	乘法
/	除法
%	求余
&	按位与
|	按位或

逻辑运算符（仅可用于比特型）

&&	逻辑与
||	逻辑或
!	逻辑异或

比较运算符

＝＝	等于
!＝	不等于
＞	大于
＞＝	大于等于
＜	小于
＜＝	小于等于

赋值运算符

＝	将右侧值赋给左侧变量

逻辑运算术语

true/false	逻辑表达式的值（真 / 假）

■ 优先级和结合规则

运算符有优先级，并且适用结合规则。
下表归纳了运算符的优先级和结合规则（所确定的运算对象的运算顺序）。
下表的运算符优先级按照从高到低的顺序排列。同一行的运算符的优先级相同，并按照结合规则被确定。

标记	说明	结合规则
[ ] ( )	其他	从左到右
- !	单目	从右到左
* / %	乘法、除法、求余	从左到右
+ -	加法、减法	从左到右
< > <= >=	关系	从左到右
== !=	关系（等值）	从左到右
&	按位与 AND	从左到右
|	按位或 OR	从左到右
&&	逻辑 AND	从左到右
||	逻辑 OR	从左到右

A.1.2 运算对象

■ 常数

常数可取整数或实数。

整数

整数可以取 32 位整型值可表示的范围内的数。
(-2147483648 ～ 2147483647)

实数

实数可以取 32 位实型值 (Float) 可表示的范围内的数。
± (1.175494351e-38F ～ 3.402823466e ＋ 38F)

■ 变量

在 Expression 中，C 语言所允许的任意变量名可通过与控制器寄存器的对应关系来表述。

C 语言中不存在 bool 型变量，但将控制器的比特型寄存器作为 bool 型来使用。bool 型变量的值只能取 true 或 false，只能用于逻辑表达式。

变量名的命名限制

可使用的变量名具有以下限制。

- 不能以数字开头
- 可使用的字符为 ASCII 码字符中的字母、下划线 “_” 及数字
- 不能使用与后述函数名相同的变量名

Abc	OK
get_input0	OK
1ab	NG
Sin	NG

A.1.3 函数

可以使用下列算数函数。

```
cos()
sin()
arctan()
tan()
```

A.2 可识别的表达式种类

表达式通过运算对象和运算符组合来表述，但其表述方式有几种限制条件。如果不满足限制条件，则无法作为表达式而被识别。

以下对限制条件进行了说明。

A.2.1 算术运算符

这种运算符只能用于整型及实型运算对象。单目负号只能使用一次。只能对整型进行位运算。不能对比特型运算对象进行算术运算。另外，即使运算值超出了寄存器的范围，也不会自动进行类型转换，因此需要用户来给变量分配适当的类型。

MW00001 = MW00002 + MW00003	OK
MW00001 = MW00002/345	OK
MF00002 = (MW00004 + MF00002)/(ML00018 = MW00008)	OK
MW00001 = MW00002 & 4096	OK
MB000010 = MB000011 - MB000012	NG
MW00001 = MB000011 * MW00001	NG

A.2.2 比较运算符

这种运算符只能用于整型及实型运算对象。左边必须为比特型寄存器。用“==”或“!=”对整数比特型的运算对象进行比较时，比较对象必须要用 true 和 false 来表示。

MB000010 = MW00002 != MW00003	OK
MB000010 = MF00002 < 99.99	OK
MB000010 = MW00002 >= MW00003	OK
MB000010 = MB000011 == true	OK
MB000010 = MB000011 != 0	NG
MB000010 = MB000011 == 1	NG

A.2.3 逻辑运算符

这种运算符仅可用于比特型运算对象。

MB000010 = MB000011 && MB000012	OK
MB000010 = !MB000011	OK
MB000010 = (MW000020 >= 50) && MB000011	OK
MB000010 = MW00001 || MW00002	NG
MB000010 = !MW00001	NG

A.2.4 赋值运算符

表达式左边和右边可以进行实型和整型不同类型的赋值。但是将实型数据赋给整型时，会产生取整误差。

只能将逻辑值（比特型寄存器或 true/false）赋给比特型寄存器将逻辑值以外的值赋给比特型寄存器时，与 0（或 0.0）比较，将其真假转化为赋值的编码。

不能将比特型赋给比特型寄存器以外的寄存器。

```
MW00001 = MW00002              OK
ML00003 = MW00002              OK
MF00006 = MW00002 * 343        OK
MB000010 = MB000011            OK
MW00001 = MF00012              OK
MB000102 = MW00010             OK
MB000102 = true                OK
MW00010 = MB000101             NG
MW00010 = true                 NG
```

A.2.5 函数

函数的形式参数和返回值取决于控制器的函数规格。
亦即，对 sin()、cos() 以及 arctan() 输入整数及整型寄存器时返回的输出值为整型，输入实数及实型寄存器时返回的输出值为实型。因为 tan() 的形式参数为实型，输入整型寄存器时也按实型处理。

```
MW00001 = sin(MW00002)          OK
MF00001 = cos(MF00002 * 3.14)   OK
MW00001 = - arctan(MF00002)     OK
```

A.2.6 其他

■ 括号

用 “(” 和 “)” 可将多个表达式连接在一起。

```
MW00001 = - ((MW00002 - MW00003)/(MW00004 + MW00005))        OK
```

■ 数组

数组和 C 语言相同，可用 “[” 和 “]” 来指定

```
MW00001 = MW00002[100]          OK
MW00001 = MW00002[MW00100]      OK
MB00001 = MB000020[0]           OK
```

A.3 在梯形图程序中的应用

梯形图程序中 Expression 的使用可分为以下 3 种。

- IF 指令句的条件表达式
- WHILE 指令句的条件表达式
- EXPRESSION 指令句的运算表达式

以下通过使用实例进行说明。

A.3.1 IF 指令句的条件表达式

表述在 IF 块及 ELSE 块的条件表达式表述范围内。但是只能表述产生 bool 型结果的 Expression 数式。因此无法识别含赋值运算符的数式。

MB000001 == true	OK
MW00002 < 100	OK
MW00003 ! = MW00004	OK
MB000005 = false	NG
MW00007 = MW00010	NG

A.3.2 WHILE 指令句的条件表达式

表述在 WHILE 块的条件表达式表述范围内。但是只能表述产生 bool 型结果的 Expression 数式。因此无法识别含赋值运算符的数式。

- 请参照附录 C.1 的示例。

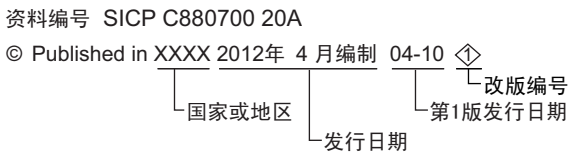
A.3.3 EXPRESSION 指令句的运算表达式

表述在 EXPRESSION 块的运算表达式表述范围内。可按照 Expression 数式规则来表述运算表达式。但是不能表述产生 bool 型结果的 Expression 数式。

MB000010 = MB000001 && MB000005;	OK
MB000011 = MB000010 == true;	OK
MW00000 = (MW00001 + MW00005)/MW00004;	OK
MW00003 = MW00000/50;	OK
MW00002 = MW00001 & 300;	OK
MW00010 = MW00003 - MW00002;	OK
MB000001 == true;	NG
MW00006 >= 100;	NG
MW00007 != MW00009;	NG

改版履历

资料的改版相关信息与资料编号一起记载于本资料封底的右下角。



印刷年 / 月	改版编号	改版追加编号	项目编号	更改之处
2012 年 4 月	◇1		封面	变更：格式、厂标
			封底	变更：格式、地址、厂标
			背脊	变更：厂标
2004 年 10 月	–			第 1 版发行