7.6 /I 作为整数(16位),将累加器2的内容除以累加器1的内容

格式

/I

说明

使用 16 位整数除法指令（/I），可以将累加器 2 低字中的内容乘以累加器 1 低字中的内容。累加器 1 和累加器 2 低字中的内容作为 16 位整数编译。结果保存在累加器 1 中，并由两个 16 位整数：商和余数组成。商保存在累加器 1 低字中，余数保存在累加器 1 高字中。指令的执行与 RLO 无关，而且对 RLO 没有影响。状态字位 CC 1、CC 0、OS 和 OV 都设定为指令结果的一个功能。

对于具有两个累加器的 CPU，累加器 2 的内容保持不变；对于具有 4 个累加器的 CPU，将累加器 3 的内容拷入累加器 2 中，将累加器 4 的内容拷入累加器 3 中。而累加器 4 的内容保持不变。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	x	x	x	x	-	-	-	-

状态位生成	CC 1	CC 0	OV	OS
商 = 0	0	0	0	-
- 32768 ≤ **商** < 0	0	1	0	-
32767 ≥ **商** 0	1	0	0	-
商 = 32768	1	0	1	1
被零除	1	1	1	1

举例

STL		**解 释**
L	IW10	// **将输入字** IW10 **的数值装入累加器** 1 **低字节。**
L	MW14	// **将累加器** 1 **低字中的内容装入累加器** 2 **低字中。将存储字** MW14 **的值装入累加器** 1 **低字。**
/I		// **将累加器** 2 **低字中的内容除以累加器** 1 **低字中的内容；结果保存到累加器** 1 **中：累加器** 1 **低字：商，累加器** 1 **高字：余数**
T	MD20	// **累加器** 1 **的内容（结果）被传送到存储双字** MD20。

例如：13 **被** 4 **除**

指令执行之前累加器 2 **低字中的内容（**IW10**）：** "13"
指令执行之前累加器 1 **低字中的内容（**MW14**）：** "4"
指令 /I**（累加器** 2 **低字中的内容 /** 累加器 1 **低字中的内容）：** "13/4"
指令执行之后累加器 1 **低字中的内容（商）：** "3"
指令执行之后累加器 1 **高字中的内容（余数）：** "1"

7.7 + 加上一个整数常数（16 位，32 位）

格式

+ <整数常数>

地 址	数据类型	说 明
<整数常数>	（16 位或 32 位整数）	要加的常数

说明

使用加上一个整数常数指令（+ <整数常数>），可以对累加器 1 中的内容加上一个整数常数，结果保存在累加器 1 中。指令的执行与状态字位无关，而且对状态字位没有影响。

对于具有两个累加器的 CPU，累加器 2 的内容保持不变。

对于具有 4 个累加器的 CPU，将累加器 3 的内容拷入累加器 2 中，将累加器 4 的内容拷入累加器 3 中。而累加器 4 的内容保持不变。

+ <16 位整数常数>：对累加器 1 低字中的内容加上一个 16 位整数常数（范围 –32768 - +32767），结果保存在累加器 1 低字中。

+ <32 位整数常数>：对累加器 1 中的内容加上一个 32 位整数常数（范围 -2,147,483,648 - +2,147,483,647），结果保存在累加器 1 中。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	-	-	-	-

举例 1

STL		解 释
L	IW10	// 将输入字 IW10 的数值装入累加器 1 低字节。
L	MW14	// 将累加器 1 低字中的内容装入累加器 2 低字中。将存储字 MW14 的值装入累加器 1 低字。
+I		// 将累加器 2 低字和累加器 1 低字中的内容相加；结果保存到累加器 1 低字中。
+	25	// 将累加器 1 低字中的内容加上“25”；结果保存到累加器 1 低字中。
T	DB1.DBW25	// 将累加器 1 低字中的内容（结果）传送到 DB1 的 DBW25 中。

举例 2

STL		解 释
L	IW12	
L	IW14	
+	100	// 将累加器 1 低字中的内容加上“100”；结果保存到累加器 1 低字中。
>I		// 如果累加器 2 中的内容大于累加器 1 中的内容，或输入字 IW12 > （输入字 IW14 + 100），
JC	NEXT	// 则条件跳转到跳转标号 NEXT。。

举例 3

STL		解 释
L	MD20	
L	MD24	
+D		// 将累加器 1 和累加器 2 中的内容相加；结果保存到累加器 1 中。
+	L#-200	// 将累加器 1 中的内容和“-200”相加；结果保存到累加器 1 中。
T	MD28	

7.8 +D 作为双整数(32位)，将累加器1和累加器2的内容相加

格式

+D

说明

使用 32 位整数加法指令（+D），可以将累加器 1 中的内容与累加器 2 中的内容相加，结果保存在累加器 1 中。累加器 1 和累加器 2 中的内容作为 32 位整数编译。执行指令与 RLO 无关，而且对 RLO 没有影响。状态字位 CC 1、CC 0、OS 和 OV 都设定为指令结果的一个功能。

对于具有两个累加器的 CPU，累加器 2 的内容保持不变。

对于具有 4 个累加器的 CPU，将累加器 3 的内容拷入累加器 2 中，将累加器 4 的内容拷入累加器 3 中。而累加器 4 的内容保持不变。

请参见“判断整数算术运算指令后状态字的位”。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	x	x	x	x	-	-	-	-

状态位生成	CC 1	CC 0	OV	OS
和 = 0	0	0	0	-
-2147483648 ≤ 和 < 0	0	1	0	-
2147483647 ≥ 和 > 0	1	0	0	-
和 = -4294967296	0	0	1	1
4294967294 ≥ 和 > 2147483647	0	1	1	1
-4294967295 ≤ 和 < -2147483648	1	0	1	1

举例

```
STL                 解  释
L    ID10           // 将输入双字 ID10 的数值装入累加器 1。
L    MD14           // 将累加器 1 中的内容装入累加器 2 中。将存储双字 MD14 的值装入累加
                       器 1 中。
+D                  // 将累加器 2 中的内容和累加器 1 中的内容相加；结果保存到累加器 1 中。
T    DB1.DBD25      // 累加器 1 的内容（结果）被传送到 DB1 的 DBD25。
```

7.9 –D 作为双整数(32位)，累加器2的内容减累加器1的内容

格式

-D

说明

使用 32 位整数减法指令（-D），可以将累加器 2 中的内容减去累加器 1 中的内容，结果保存在累加器 1 中。累加器 1 和累加器 2 中的内容作为 32 位整数编译。执行指令与 RLO 无关，而且对 RLO 没有影响。状态字位 CC 1、CC 0、OS 和 OV 都设定为指令结果的一个功能。

对于具有两个累加器的 CPU，累加器 2 的内容保持不变。

对于具有 4 个累加器的 CPU，将累加器 3 的内容拷入累加器 2 中，将累加器 4 的内容拷入累加器 3 中。而累加器 4 的内容保持不变。

请参见“判断整数算术运算指令后状态字的位”。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	x	x	x	x	-	-	-	-

状态位生成	CC 1	CC 0	OV	OS
差 = 0	0	0	0	-
-2147483648 ≤ 差 < 0	0	1	0	-
2147483647 ≥ 差 > 0	1	0	0	-
4294967295 ≥ 差 > 2147483647	0	1	1	1
-4294967295 ≤ 差 < -2147483648	1	0	1	1

举例

STL		解 释
L	ID10	// 将输入双字 ID10 的数值装入累加器 1。
L	MD14	// 将累加器 1 中的内容装入累加器 2 中。将存储双字 MD14 的值装入累加器 1 中。
-D		// 将累加器 2 中的内容减去累加器 1 中的内容；结果保存到累加器 1 中。
T	DB1.DBD25	// 累加器 1 的内容（结果）被传送到 DB1 的 DBD25。

7.10 *D 作为双整数(32位)，将累加器1和累加器2的内容相乘

格式

*D

说明

使用 32 位整数乘法指令（*D），可以将累加器 2 中的内容乘以累加器 1 中的内容。累加器 1 和累加器 2 中的内容作为 32 位整数编译。结果作为一个 32 位整数保存在累加器 1 中。如果状态字位 OV1 = 1 且 OS = 1，则结果超出 32 位整数的范围。

执行指令与 RLO 无关，而且对 RLO 没有影响。状态字位 CC 1、CC 0、OS 和 OV 都设定为指令结果的一个功能。

对于具有两个累加器的 CPU，累加器 2 的内容保持不变。

对于具有 4 个累加器的 CPU，将累加器 3 的内容拷入累加器 2 中，将累加器 4 的内容拷入累加器 3 中。而累加器 4 的内容保持不变。

请参见“判断整数算术运算指令后状态字的位”。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	x	x	x	x	-	-	-	-

状态位生成	CC 1	CC 0	OV	OS
积 = 0	0	0	0	-
-2147483648 ≤ 积 < 0	0	1	0	-
2147483647 ≥ 积 > 0	1	0	0	-
积 > 2147483647	1	0	1	1
积 < -2147483648	0	1	1	1

举例

STL		解 释
L	ID10	// 将输入双字 ID10 的数值装入累加器 1。
L	MD14	// 将累加器 1 中的内容装入累加器 2 中。将存储双字 MD14 的值装入累加器 1 中。
*D		// 将累加器 2 中的内容和累加器 1 中的内容相乘；结果保存到累加器 1 中。
T	DB1.DBD25	// 累加器 1 的内容（结果）被传送到 DB1 的 DBD25。

7.11 /D 作为双整数(32位)，累加器2的内容除以累加器1的内容

格式

/D

说明

使用 32 位整数除法指令（/D），可以将累加器 2 中的内容除以累加器 1 中的内容。累加器 1 和累加器 2 中的内容作为 32 位整数编译。结果保存在累加器 1 中。结果只给出了商，没有余数。（使用指令 MOD，可以获得余数）。

执行指令与 RLO 无关，而且对 RLO 没有影响。状态字位 CC 1、CC 0、OS 和 OV 都设定为指令结果的一个功能。

对于具有两个累加器的 CPU，累加器 2 的内容保持不变。

对于具有 4 个累加器的 CPU，将累加器 3 的内容拷入累加器 2 中，将累加器 4 的内容拷入累加器 3 中。而累加器 4 的内容保持不变。

请参见“判断整数算术运算指令后状态字的位”。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	x	x	x	x	-	-	-	-

状态位生成	CC 1	CC 0	OV	OS
商 = 0	0	0	0	-
-2147483648 ≤ **商** < 0	0	1	0	-
2147483647 ≥ **商** > 0	1	0	0	-
商 = 2147483648	1	0	1	1
被零除	1	1	1	1

举例

STL		解 释
L	ID10	// 将输入双字 ID10 的数值装入累加器 1。
L	MD14	// 将累加器 1 中的内容装入累加器 2 中。将存储双字 MD14 的值装入累加器 1 中。
/D		// 将累加器 2 中的内容除以累加器 1 中的内容；结果（商）保存到累加器 1 中。
T	MD20	// 累加器 1 的内容（结果）被传送到存储双字 MD20。

例如：13 被 4 除

指令执行之前累加器 2 中的内容（ID10）： "13"
指令执行之前累加器 1 中的内容（MD14）： "4"
指令 /D（累加器 2 中的内容 / 累加器 1 中的内容）： "13/4"
指令执行之后累加器 1 中的内容（商）： "3"

7.12 MOD 双整数除法的余数（32位）

格式

MOD

说明

使用 MOD 指令（32 位整数除法的余数），可以将累加器 2 中的内容除以累加器 1 中的内容。累加器 1 和累加器 2 中的内容作为 32 位整数编译。结果保存在累加器 1 中。结果只给出了余数，没有商。（使用指令 /D，可以获得商）。

执行指令与 RLO 无关，而且对 RLO 没有影响。状态字位 CC 1、CC 0、OS 和 OV 都设定为指令结果的一个功能。

对于具有两个累加器的 CPU，累加器 2 的内容保持不变。

对于具有 4 个累加器的 CPU，将累加器 3 的内容拷入累加器 2 中，将累加器 4 的内容拷入累加器 3 中。而累加器 4 的内容保持不变。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	x	x	x	x	-	-	-	-

状态位生成	CC 1	CC 0	OV	OS
余数 = 0	0	0	0	-
- 2147483648 ≤ **余数** < 0	0	1	0	-
2147483647 ≥ **余数** > 0	1	0	0	-
被零除	1	1	1	1

举例

STL	解 释
L ID10	// 将输入双字 ID10 的数值装入累加器 1。
L MD14	// 将累加器 1 中的内容装入累加器 2 中。将存储双字 MD14 的值装入累加器 1 中。
MOD	// 将累加器 2 中的内容除以累加器 1 中的内容 ;结果 (余数) 保存到累加器 1 中。
T MD20	// 累加器 1 的内容（结果）被传送到存储双字 MD20。

例如：13 **被** 4 **除**

指令执行之前累加器 2 **中的内容（**ID10**）：** "13"

指令执行之前累加器 1 **中的内容（**MD14**）：** "4"

指令 MOD**（累加器** 2 **中的内容** / **累加器** 1 **中的内容）：** "13/4"

指令执行之后累加器 1 **中的内容（余数）：** "1"

8 浮点算术运算指令

8.1 浮点算术运算指令概述

说明

算术运算指令针对累加器 1 和 2 的内容。其结果保存在累加器 1 中。累加器 1 的原有内容被移入累加器 2 中。累加器 2 的内容保持不变。

如果 CPU 具有 4 个累加器，将累加器 3 的内容拷入累加器 2 中，将累加器 4 的内容拷入累加器 3 中。

而累加器 4 中原有的内容保持不变。

标准 IEEE 32 位浮点数所属的数据类型称为实数“ REAL ”。

应用浮点算术运算指令，可以对于两个 32 位标准 IEEE 浮点数完成以下算术运算：

- +R **将累加器 1 和累加器 2 中的内容相加**
- -R **将累加器 2 中的内容减去累加器 1 中的内容**
- *R **将累加器 1 和累加器 2 中的内容相乘**
- /R **将累加器 2 中的内容除以累加器 1 中的内容**

应用浮点算术运算指令，可以对于一个 32 位标准 IEEE 浮点数完成以下算术运算：

- ABS **浮点数取绝对值**
- SQR **浮点数平方**
- SQRT **浮点数开方**
- EXP **浮点数指数运算**
- LN **浮点数自然对数运算**
- SIN **浮点数正弦运算**
- COS **浮点数余弦运算**
- TAN **浮点数正切运算**
- ASIN **浮点数反正弦运算**
- ACOS **浮点数反余弦运算**
- ATAN **浮点数反正切运算**

请参见“ 判断浮点算术运算指令后状态字的位 ”。

8.2 判断浮点算术运算指令后状态字的位

说明

基本算术类型可以影响以下状态字中的位：CC 1 **和** CC 0**，**OV **和** OS**。**

下表所示为使用了浮点数（32 **位）运算指令结果的状态字中各位的信号状态：**

有效的结果范围	CC 1	CC 0	OV	OS
+0，- 0**（零）**	0	0	0	*
-3.402823E+38 < **结果** < -1.175494E-38**（负数）**	0	1	0	*
+1.175494E-38 < **结果** < 3.402824E+38**（正数）**	1	0	0	*

* OS **位不受指令结果的影响。**

无效的结果范围	CC 1	CC 0	OV	OS
下溢 -1.175494E-38 < **结果** < - 1.401298E-45**（负数）**	0	0	1	1
下溢 +1.401298E-45 < **结果** < +1.175494E-38**（正数）**	0	0	1	1
上溢 **结果** < -3.402823E+38**（负数）**	0	1	1	1
上溢 **结果** > 3.402823E+38**（正数）**	1	0	1	1
非有效浮点数或非法指令 **（输入值超出有效范围）**	1	1	1	1

8.3 浮点算术运算指令：基本指令

8.3.1 +R 作为浮点数(32位，IEEE-FP)，将累加器1和累加器2中的内容相加

格式

+R

指令说明

使用 32 位 IEEE 浮点数加法指令（+R），可以将累加器 1 中的内容与累加器 2 中的内容相加，结果保存在累加器 1 中。累加器 1 和累加器 2 中的内容作为 32 位 IEEE 浮点数编译。执行指令与 RLO 无关，而且对 RLO 没有影响。状态位 CC 1、CC 0、OS 和 OV 都设定为指令结果的一个功能。

对于具有两个累加器的 CPU，累加器 2 的内容保持不变。

对于具有 4 个累加器的 CPU，将累加器 3 的内容拷入累加器 2 中，将累加器 4 的内容拷入累加器 3 中。而累加器 4 的内容保持不变。

结果

累加器 1 中的结果	CC 1	CC 0	OV	OS	备 注
+qNaN	1	1	1	1	
+infinite（无穷大）	1	0	1	1	上溢
+normalized（规格化）	1	0	0	-	
+denormalized（反向规格化）	0	0	1	1	下溢
+zero（零）	0	0	0	-	
-zero（零）	0	0	0	-	
-denormalized（反向规格化）	0	0	1	1	下溢
-normalized（规格化）	0	1	0	-	
-infinite（无穷小）	0	1	1	1	上溢
-qNaN	1	1	1	1	

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	x	x	x	x	-	-	-	-

举例

STL		解 释
OPN	DB10	
L	ID10	// 将输入双字 ID10 的数值装入累加器 1。
L	MD14	// 将累加器 1 中的内容装入累加器 2 中。将存储双字 MD14 的值装入累加器 1 中。
+R		// 将累加器 2 中的内容和累加器 1 中的内容相加；结果保存到累加器 1 中。
T	DBD25	// 累加器 1 的内容（结果）被传送到 DB10 的 DBD25。

8.3.2 –R 作为浮点数(32位，IEEE-FP)，将累加器2中的内容减去累加器1中的内容

格式

-R

说明

使用 32 位 IEEE 浮点数减法指令（-R），可以将累加器 2 中的内容减去累加器 1 中的内容，结果保存在累加器 1 中。累加器 1 和累加器 2 中的内容作为 32 位 IEEE 浮点数编译。执行指令与 RLO 无关，而且对 RLO 没有影响。状态位 CC 1、CC 0、OS 和 OV 都设定为指令结果的一个功能。

对于具有两个累加器的 CPU，累加器 2 的内容保持不变。

对于具有 4 个累加器的 CPU，将累加器 3 的内容拷入累加器 2 中，将累加器 4 的内容拷入累加器 3 中。而累加器 4 的内容保持不变。

结果

累加器 1 中的结果	CC 1	CC 0	OV	OS	备 注
+qNaN	1	1	1	1	
+infinite（无穷大）	1	0	1	1	上溢
+normalized（规格化）	1	0	0	-	
+denormalized（反向规格化）	0	0	1	1	下溢
+zero（零）	0	0	0	-	
-zero（零）	0	0	0	-	
-denormalized（反向规格化）	0	0	1	1	下溢
-normalized（规格化）	0	1	0	-	
-infinite（无穷小）	0	1	1	1	上溢
-qNaN	1	1	1	1	

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	x	x	x	x	-	-	-	-

举例

```
STL              解  释
OPN DB10
L    ID10        // 将输入双字 ID10 的数值装入累加器 1。
L    MD14        // 将累加器 1 中的内容装入累加器 2 中。将存储双字 MD14 的值装入累加器 1 中。
-R               // 将累加器 2 中的内容减去累加器 1 中的内容；结果保存到累加器 1 中。
T    DBD25       // 累加器 1 的内容（结果）被传送到 DB10 的 DBD25。
```

8.3.3 *R 作为浮点数(32位，IEEE-FP)，将累加器1和累加器2中的内容相乘

格式

*R

指令说明

使用 32 位 IEEE 浮点数乘法指令（*R），可以将累加器 2 中的内容乘以累加器 1 中的内容。累加器 1 和累加器 2 中的内容作为 32 位 IEEE 浮点数编译。结果作为一个 32 位 IEEE 浮点数保存在累加器 1 中。执行指令与 RLO 无关，而且对 RLO 没有影响。状态位 CC 1、CC 0、OS 和 OV 都设定为指令结果的一个功能。

对于具有两个累加器的 CPU，累加器 2 的内容保持不变。

对于具有 4 个累加器的 CPU，将累加器 3 的内容拷入累加器 2 中，将累加器 4 的内容拷入累加器 3 中。而累加器 4 的内容保持不变。

结果

累加器 1 中的结果	CC 1	CC 0	OV	OS	备 注
+qNaN	1	1	1	1	
+infinite（无穷大）	1	0	1	1	上溢
+normalized（规格化）	1	0	0	-	
+denormalized（反向规格化）	0	0	1	1	下溢
+zero（零）	0	0	0	-	
-zero（零）	0	0	0	-	
-denormalized（反向规格化）	0	0	1	1	下溢
-normalized（规格化）	0	1	0	-	
-infinite（无穷小）	0	1	1	1	上溢
-qNaN	1	1	1	1	

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	x	x	x	x	-	-	-	-

举例

STL		解 释
OPN	DB10	
L	ID10	// 将输入双字 ID10 的数值装入累加器 1。
L	MD14	// 将累加器 1 中的内容装入累加器 2 中。将存储双字 MD14 的值装入累加器 1 中。
*R		// 将累加器 2 中的内容和累加器 1 中的内容相乘；结果保存到累加器 1 中。
T	DBD25	// 累加器 1 的内容（结果）被传送到 DB10 的 DBD25。

8.3.4 /R 作为浮点数(32位，IEEE-FP)，累加器2的内容除以累加器1的内容

格式

/R

指令说明

使用 32 位 IEEE 浮点数除法指令（/R），可以将累加器 2 中的内容除以累加器 1 中的内容。累加器 1 和累加器 2 中的内容作为 32 位 IEEE 浮点数编译。结果作为一个 32 位 IEEE 浮点数保存在累加器 1 中。执行指令与 RLO 无关，而且对 RLO 没有影响。状态位 CC 1、CC 0、OS 和 OV 都设定为指令结果的一个功能。

对于具有两个累加器的 CPU，累加器 2 的内容保持不变。

对于具有 4 个累加器的 CPU，将累加器 3 的内容拷入累加器 2 中，将累加器 4 的内容拷入累加器 3 中。

结果

累加器 1 中的结果	CC 1	CC 0	OV	OS	备 注
+qNaN	1	1	1	1	
+infinite（无穷大）	1	0	1	1	上溢
+normalized（规格化）	1	0	0	-	
+denormalized（反向规格化）	0	0	1	1	下溢
+zero（零）	0	0	0	-	
-zero（零）	0	0	0	-	
-denormalized（反向规格化）	0	0	1	1	下溢
-normalized（规格化）	0	1	0	-	
-infinite（无穷小）	0	1	1	1	上溢
-qNaN	1	1	1	1	

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	x	x	x	x	-	-	-	-

举例

STL		解 释
OPN	DB10	
L	ID10	// 将输入双字 ID10 的数值装入累加器 1。
L	MD14	// 将累加器 1 中的内容装入累加器 2 中。将存储双字 MD14 的值装入累加器 1 中。
/R		// 将累加器 2 中的内容除以累加器 1 中的内容；结果保存到累加器 1 中。
T	DBD20	// 累加器 1 的内容（结果）被传送到 DB10 的 DBD20。

8.3.5 ABS 浮点数取绝对值（32 位，IEEE-FP）

格式

ABS

说明

使用 ABS（对 32 位 IEEE 浮点数取绝对值）指令，可以对累加器 1 中的浮点数（32 位，IEEE-FP）取绝对值。其结果保存在累加器 1 中。该指令的执行与状态位无关，对状态位也没有影响。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	-	-	-	-

举例

STL		解 释
L	ID8	// 将数值装入累加器 1 中（例如：ID8 = -1.5E+02）。
ABS		// 求绝对值；结果保存到累加器 1 中。
T	MD10	// 将结果传送到存储双字 MD10（例如：结果 = 1.5E+02）。

8.4 浮点算术运算指令：扩展指令

8.4.1 SQR 浮点数平方运算（32 位）

格式

SQR

指令说明

使用 SQR（对 32 位 IEEE 浮点数求平方）指令，可以对累加器 1 中的浮点数（32 位，IEEE-FP）求平方。其结果保存在累加器 1 中。该指令的执行影响 CC 1、CC 0、OV 和 OS 状态字位。

累加器 2 的内容（以及累加器 3 和累加器 4 的内容，对于具有 4 个累加器的 CPU）保持不变。

结果

累加器 1 中的结果	CC 1	CC 0	OV	OS	备 注
+qNaN	1	1	1	1	
+infinite（无穷大）	1	0	1	1	上溢
+normalized（规格化）	1	0	0	-	
+denormalized（反向规格化）	0	0	1	1	下溢
+zero（零）	0	0	0	-	
-qNaN	1	1	1	1	

举例

```
STL                 解  释
     OPN DB17       // 打开数据块 DB17。
     L   DBD0       // 数据双字 DBD0 的值装入累加器 1 中。 （该值必须为浮点数格式）。
     SQR            // 在累加器 1 中求浮点数（32 位，IEEE FP）的平方。结果保存到累加器 1 中。
     AN  OV         // 扫描状态字中的 OV 位是否为“0”。
     JC  OK         // 如果在 SQR 执行过程中没有出现错误，则跳转到 OK 跳转标号。
     BEU            // 如果在 SQR 执行过程中出现错误，则块无条件结束。
OK:  T   DBD4       // 将累加器 1 中的内容（结果）传送到数据双字 DBD4。
```

8.4.2 SQRT 浮点数开方运算（32 位）

格式

SQRT

指令说明

使用 SQRT（对 32 位 IEEE 浮点数求平方根）指令，可以对累加器 1 中的浮点数（32 位，IEEE-FP）求平方根。其结果保存在累加器 1 中。输入值必须大于或等于“0”。结果为正值。“-0”的平方根为“–0”例外。

该指令会影响 CC 1、CC 0、OV 和 OS 状态字位。

累加器 2 的内容（以及累加器 3 和累加器 4 的内容，对于具有 4 个累加器的 CPU）保持不变。

结果

累加器 1 中的结果	CC 1	CC 0	OV	OS	备 注
+qNaN	1	1	1	1	
+infinite（无穷大）	1	0	1	1	上溢
+normalized（规格化）	1	0	0	-	
+denormalized（反向规格化）	0	0	1	1	下溢
+zero（零）	0	0	0	-	
-zero（零）	0	0	0	-	
-qNaN	1	1	1	1	

举例

```
STL                 解  释
      L    MD10     // 将存储双字 MD10 的值装入累加器 1 中。 （该值必须为浮点数格式）。
      SQRT          // 在累加器 1 中求浮点数(32 位,IEEE FP)的平方根。结果保存到累加器 1 中。
      AN   OV       // 扫描状态字中的 OV 位是否为“0”。
      JC   OK       // 如果在 SQRT 执行过程中没有出现错误，则跳转到 OK 跳转标号。
      BEU           // 如果在 SQRT 执行过程中出现错误，则块无条件结束。
OK:   T    MD20     // 将累加器 1 中的内容（结果）传送到存储双字 MD20。
```

8.4.3 EXP 浮点数指数运算（32 位）

格式

EXP

指令说明

使用 EXP（对 32 位 IEEE 浮点数求指数）指令，可以对累加器 1 中的浮点数（32 位，IEEE-FP）求指数（以 e 为底）。其结果保存在累加器 1 中。该指令的执行影响 CC 1、CC 0、OV 和 OS 状态字位。

累加器 2 的内容（以及累加器 3 和累加器 4 的内容，对于具有 4 个累加器的 CPU）保持不变。

结果

累加器 1 中的结果	CC 1	CC 0	OV	OS	备 注
+qNaN	1	1	1	1	
+infinite（无穷大）	1	0	1	1	上溢
+normalized（规格化）	1	0	0	-	
+denormalized（反向规格化）	0	0	1	1	下溢
+zero（零）	0	0	0	-	
-qNaN	1	1	1	1	

举例

```
STL                 解 释
    L    MD10       // 将存储双字 MD10 的值装入累加器 1 中。（该值必须是浮点数格式）。
    EXP             // 在累加器 1 中求浮点数（32 位，IEEE FP）的指数（以 e 为底数）。结果
                       保存到累加器 1 中。
    AN   OV         // 扫描状态字中的 OV 位是否为“0”。
    JC   OK         // 如果在 EXP 执行过程中没有出现错误，则跳转到 OK 跳转标号。
    BEU             // 如果在 EXP 执行过程中出现错误，则块无条件结束。
OK: T    MD20       // 将累加器 1 中的内容（结果）传送到存储双字 MD20。
```

8.4.4 LN 浮点数自然对数运算（32 位）

格式

LN

指令说明

使用 LN（对 32 位 IEEE 浮点数求自然对数）指令，可以对累加器 1 中的浮点数（32 位，IEEE-FP）求自然对数。其结果保存在累加器 1 中。输入值必须大于“0”。该指令会影响 CC 1、CC 0、OV 和 OS 状态字位。

累加器 2 的内容（以及累加器 3 和累加器 4 的内容，对于具有 4 个累加器的 CPU）保持不变。

结果

累加器 1 中的结果	CC 1	CC 0	OV	OS	备 注
+qNaN	1	1	1	1	
+infinite（无穷大）	1	0	1	1	上溢
+normalized（规格化）	1	0	0	-	
+denormalized（反向规格化）	0	0	1	1	下溢
+zero（零）	0	0	0	-	
-zero（零）	0	0	0	-	
-denormalized（反向规格化）	0	0	1	1	下溢
-normalized（规格化）	0	1	0	-	
-infinite（无穷小）	0	1	1	1	上溢
-qNaN	1	1	1	1	

举例

```
STL                 解 释
     L    MD10     // 将存储双字 MD10 的值装入累加器 1 中。（该值必须是浮点数格式）。
     LN            // 在累加器 1 中求浮点数(32 位，IEEE FP)的自然对数。结果保存到累加器 1 中。
     AN   OV       // 扫描状态字中的 OV 位是否为“0”。
     JC   OK       // 如果在指令执行过程中没有出现错误，则跳转到 OK 跳转标号。
     BEU           // 如果在指令执行过程中出现错误，则块无条件结束。
OK:  T    MD20     // 将累加器 1 中的内容（结果）传送到存储双字 MD20。
```

8.4.5 SIN 浮点数正弦运算（32 位）

格式

SIN

指令说明

使用 SIN（对 32 位 IEEE 浮点数求正弦）指令，可以计算角度为弧度的正弦。角度必须在累加器 1 中以浮点数表示。其结果保存在累加器 1 中。该指令的执行影响 CC 1、CC 0、OV 和 OS 状态字位。累加器 2 的内容（以及累加器 3 和累加器 4 的内容，对于具有 4 个累加器的 CPU）保持不变。

结果

累加器 1 中的结果	CC 1	CC 0	OV	OS	备 注
+qNaN	1	1	1	1	
+normalized（规格化）	1	0	0	-	
+denormalized（反向规格化）	0	0	1	1	上溢
+zero（零）	0	0	0	-	
-zero（零）	0	0	0	-	
-denormalized（反向规格化）	0	0	1	1	下溢
-normalized（规格化）	0	1	0	-	
-qNaN	1	1	1	1	

举例

```
STL                 解 释
L    MD10           // 将存储双字 MD10 的值装入累加器 1 中。（该值必须是浮点数格式）。
SIN                 // 在累加器 1 中求浮点数(32 位,IEEE FP)的正弦。结果保存到累加器 1 中。
T    MD20           // 将累加器 1 中的内容（结果）传送到存储双字 MD20。
```

8.4.6 COS 浮点数余弦运算（32 位）

格式

COS

指令说明

使用 COS（对 32 位 IEEE 浮点数求余弦）指令，可以计算角度为弧度的余弦。角度必须在累加器 1 中以浮点数表示。其结果保存在累加器 1 中。该指令的执行影响 CC 1、CC 0、OV 和 OS 状态字位。

累加器 2 的内容（以及累加器 3 和累加器 4 的内容，对于具有 4 个累加器的 CPU）保持不变。

结果

累加器 1 中的结果	CC 1	CC 0	OV	OS	备 注
+qNaN	1	1	1	1	
+normalized（规格化）	1	0	0	-	
+denormalized（反向规格化）	0	0	1	1	上溢
+zero（零）	0	0	0	-	
-zero（零）	0	0	0	-	
-denormalized（反向规格化）	0	0	1	1	下溢
-normalized（规格化）	0	1	0	-	
-qNaN	1	1	1	1	

举例

```
STL                 解 释
L    MD10           // 将存储双字 MD10 的值装入累加器 1 中。（该值必须是浮点数格式）。
COS                 // 在累加器 1 中求浮点数(32 位,IEEE FP)的余弦。结果保存到累加器 1 中。
T    MD20           // 将累加器 1 中的内容（结果）传送到存储双字 MD20。
```

8.4.7 TAN 浮点数正切运算（32 位）

格式

TAN

指令说明

使用 TAN（对 32 位 IEEE 浮点数求正切）指令，可以计算角度为弧度的正切。角度必须在累加器 1 中以浮点数表示。其结果保存在累加器 1 中。该指令的执行影响 CC 1、CC 0、OV 和 OS 状态字位。

累加器 2 的内容（以及累加器 3 和累加器 4 的内容，对于具有 4 个累加器的 CPU）保持不变。

结果

累加器 1 中的结果	CC 1	CC 0	OV	OS	备 注
+qNaN	1	1	1	1	
+infinite（无穷大）	1	0	1	1	上溢
+normalized（规格化）	1	0	0	-	
+denormalized（反向规格化）	0	0	1	1	下溢
+zero（零）	0	0	0	-	
-zero（零）	0	0	0	-	
-denormalized（反向规格化）	0	0	1	1	下溢
-normalized（规格化）	0	1	0	-	
-infinite（无穷小）	0	1	1	1	上溢
-qNaN	1	1	1	1	

举例

```
STL                    解 释
      L    MD10        // 将存储双字 MD10 的值装入累加器 1 中。（该值必须是浮点数格式）。
      TAN              // 在累加器 1 中求浮点数(32 位,IEEE FP)的正切。结果保存到累加器 1 中。
      AN   OV          // 扫描状态字中的 OV 位是否为“0”。
      JC   OK          // 如果在 TAN 执行过程中没有出现错误，则跳转到 OK 跳转标号。
      BEU              // 如果在 TAN 执行过程中出现错误，则块无条件结束。
OK:   T    MD20        // 将累加器 1 中的内容（结果）传送到存储双字 MD20。
```

8.4.8 ASIN 浮点数反正弦运算（32 位）

格式

ASIN

指令说明

使用 ASIN（对 32 位 IEEE 浮点数求反正弦）指令，可以计算累加器 1 中浮点数的反正弦。输入值的允许范围：

-1 ≤ 输入值 ≤ +1

结果是以弧度为单位的角度。其值范围为：

$-\pi/2 \leq$ 反正弦（累加器 1 中的内容）$\leq +\pi/2$，其中 $\pi = 3.14159...$

该指令会影响 CC 1、CC 0、OV 和 OS 状态字位。

累加器 2 的内容（以及累加器 3 和累加器 4 的内容，对于具有 4 个累加器的 CPU）保持不变。

结果

累加器 1 中的结果	CC 1	CC 0	OV	OS	备 注
+qNaN	1	1	1	1	
+normalized（规格化）	1	0	0	-	
+denormalized（反向规格化）	0	0	1	1	上溢
+zero（零）	0	0	0	-	
-zero（零）	0	0	0	-	
-denormalized（反向规格化）	0	0	1	1	下溢
-normalized（规格化）	0	1	0	-	
-qNaN	1	1	1	1	

举例

```
STL                 解  释
     L    MD10     // 将存储双字 MD10 的值装入累加器 1 中。（该值必须是浮点数格式）。
     ASIN          // 在累加器 1 中求浮点数(32 位，IEEE FP)的反正弦。结果保存到累加器 1 中。
     AN   OV       // 扫描状态字中的 OV 位是否为“0”。
     JC   OK       // 如果在 ASIN 执行过程中没有出现错误，则跳转到 OK 跳转标号。
     BEU           // 如果在 ASIN 执行过程中出现错误，则块无条件结束。
OK:  T    MD20     // 将累加器 1 中的内容（结果）传送到存储双字 MD20。
```

8.4.9 ACOS 浮点数反余弦运算（32 位）

格式

ACOS

指令说明

使用 ACOS（对 32 位 IEEE 浮点数求反余弦）指令，可以计算累加器 1 中浮点数的反余弦。输入值的允许范围：

-1 ≤ 输入值 ≤ +1

结果是以弧度为单位的角度。其值范围为：

0 ≤ 反余弦（累加器 1 中的内容）≤ π，其中 π = 3.14159...

该指令会影响 CC 1、CC 0、OV 和 OS 状态字位。

累加器 2 的内容（以及累加器 3 和累加器 4 的内容，对于具有 4 个累加器的 CPU）保持不变。

结果

累加器 1 中的结果	CC 1	CC 0	OV	OS	备 注
+qNaN	1	1	1	1	
+normalized（规格化）	1	0	0	-	
+denormalized（反向规格化）	0	0	1	1	上溢
+zero（零）	0	0	0	-	
-zero（零）	0	0	0	-	
-denormalized（反向规格化）	0	0	1	1	下溢
-normalized（规格化）	0	1	0	-	
-qNaN	1	1	1	1	

举例

```
STL                 解  释
      L    MD10     // 将存储双字 MD10 的值装入累加器 1 中。（该值必须是浮点数格式）。
      ACOS          // 在累加器 1 中求浮点数(32 位，IEEE FP)的反余弦。结果保存到累加器 1 中。
      AN   OV       // 扫描状态字中的 OV 位是否为“0”。
      JC   OK       // 如果在 ACOS 执行过程中没有出现错误，则跳转到 OK 跳转标号。
      BEU           // 如果在 ACOS 执行过程中出现错误，则块无条件结束。
OK:   T    MD20     // 将累加器 1 中的内容（结果）传送到存储双字 MD20。
```

8.4.10 ATAN 浮点数反正切运算（32 位）

格式

ATAN

指令说明

使用 ATAN（对 32 位 IEEE 浮点数求反正切）指令，可以计算累加器 1 中浮点数的反正切。结果是以弧度为单位的角度。其值范围为：

？？ $-\pi/2 \leq$ 反正切（累加器 1 中的内容）$\leq +\pi/2$，其中 $\pi = 3.14159...$

该指令会影响 CC 1、CC 0、OV 和 OS 状态字位。

累加器 2 的内容（以及累加器 3 和累加器 4 的内容，对于具有 4 个累加器的 CPU）保持不变。

结果

累加器 1 中的结果	CC 1	CC 0	OV	OS	备 注
+qNaN	1	1	1	1	
+normalized（规格化）	1	0	0	-	
+denormalized（反向规格化）	0	0	1	1	上溢
+zero（零）	0	0	0	-	
-zero（零）	0	0	0	-	
-denormalized（反向规格化）	0	0	1	1	下溢
-normalized（规格化）	0	1	0	-	
-qNaN	1	1	1	1	

举例

```
STL                 解  释
      L    MD10     // 将存储双字 MD10 的值装入累加器 1 中。（该值必须是浮点数格式）。
      ATAN          // 在累加器 1 中求浮点数(32 位，IEEE FP)的反正切。结果保存到累加器 1 中。
      AN   OV       // 扫描状态字中的 OV 位是否为“0”。
      JC   OK       // 如果在 ATAN 执行过程中没有出现错误，则跳转到 OK 跳转标号。
      BEU           // 如果在 ATAN 执行过程中出现错误，则块无条件结束。
OK:   T    MD20     // 将累加器 1 中的内容（结果）传送到存储双字 MD20。
```

9 装入和传送指令

9.1 装入和传送指令概述

说明

使用装入（L）和传送（T）指令，可以对输入或输出模块与存储区之间的信息交换进行编程。CPU 在每次扫描中将无条件执行这些指令，也就是说，这些指令不受语句逻辑操作结果（RLO）的影响。

下述装入和传送指令可供使用：

- L 装入
- L STW 将状态字装入累加器 1
- LAR1 AR2 将地址寄存器 2 的内容装入地址寄存器 1
- LAR1 <D> 将两个双整数（32 位指针）装入地址寄存器 1
- LAR1 将累加器 1 中的内容装入地址寄存器 1
- LAR2 <D> 将两个双整数（32 位指针）装入地址寄存器 2
- LAR2 将累加器 2 中的内容装入地址寄存器 1

- T 传送
- T STW 将累加器 1 中的内容传送到状态字
- TAR1 AR2 将地址寄存器 1 的内容传送到地址寄存器 2
- TAR1 <D> 将地址寄存器 1 的内容传送到目的地（32 位指针）
- TAR2 <D> 将地址寄存器 2 的内容传送到目的地（32 位指针）
- TAR1 将地址寄存器 1 中的内容传送到累加器 1
- TAR2 将地址寄存器 2 中的内容传送到累加器 1
- CAR 交换地址寄存器 1 和地址寄存器 2 的内容

9.2 L 装入

格式

L <地址>

地 址	数据类型	存储区	源地址
<地址>	BYTE WORD DWORD	E，A，PE，M，L， D，指针，参数	0...65535 0...65534 0...65532

说明

使用该指令，可以在累加器 1 的原有内容保存到累加器 2 并复位累加器 1 为“0”之后，将寻址字节、字或双字装入累加器 1 中。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	-	-	-	-

举例

STL		解 释
L	IB10	// 将输入字节 IB10 装入累加器 1 低字低字节中。
L	MB120	// 将存储字节 MB120 装入累加器 1 低字低字节中。
L	DBB12	// 将数据字节 DBB12 装入累加器 1 低字低字节中。
L	DIW15	// 将背景数据字 DIW15 装入累加器 1 低字中。
L	LD252	// 将本地数据双字 LD252 装入累加器 1 中。
L	P# I 8.7	// 将指针装入累加器 1。
L	OTTO	// 将参数“OTTO”装入累加器 1。
L	P# ANNA	// 将指针装入累加器 1 的指定参数。（使用该指令可以装入指定参数的相对地址偏移量）。为了计算多背景功能块中背景数据块中的绝对偏移量，必须将 AR2 寄存器的内容加上该值。

累加器 1 的内容

累加器 1 的内容	累加器 1 高字中的高字节	累加器 1 高字中的低字节	累加器 1 低字中的高字节	累加器 1 低字中的低字节
在装入指令执行之前	XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX
在 L MB10（L <字节>）执行之后	00000000	00000000	00000000	<MB10>
在 L MW10（L <字>）执行之后	00000000	00000000	<MB10>	<MB11>
在 L MD10（L <双字>）执行之后	<MB10>	<MB11>	<MB12>	<MB13>
在 L P# ANNA（功能块中）执行之后	<86>		<相对于功能块，开始 ANNA 的位偏移> 为了计算多背景功能块中背景数据块中的绝对偏移量，必须将 AR2 寄存器的内容加上该值。	
在 L P# ANNA（功能中）执行之后	<ANNA 数据被传送到区域的交叉地址>			
	X = “1”或“0”			

9.3 L STW 将状态字装入累加器 1

格式

L STW

说明

使用 L STW（使用地址 STW 装入）指令，可以将状态字的内容装入累加器 1。指令的执行与状态位无关，而且对状态位没有影响。

注意

对于 S7-300 系列 CPU，L STW 语句不能装入状态字的 FC、STA 和 OR 位。只有位 1、4、5、6、7 和 8 才能装入累加器 1 低字中的相应位。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	-	-	-	-

举例

STL	解 释
L STW	// 将状态字的内容装入累加器 1 中。

执行 L STW 后累加器 1 的内容如下：

位	31-9	8	7	6	5	4	3	2	1	0
内容	0	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC

9.4 LAR1 将累加器 1 中的内容装入地址寄存器 1

格式

LAR1

说明

使用该指令，可以将累加器 1 的内容（32 位指针）装入地址寄存器 AR1。累加器 1 和累加器 2 保持不变。指令的执行与状态位无关，而且对状态位没有影响。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	-	-	-	-

9.5 LAR1 <D> 将两个双整数(32位指针)装入地址寄存器1

格式

LAR1 <D>

地 址	数据类型	存储区	源地址
<D>	DWORD 指针常数	D，M，L	0...65532

说明

使用该指令，可以将寻址双字<D>的内容或指针常数装入地址寄存器 AR1。累加器 1 和累加器 2 保持不变。指令的执行与状态位无关，而且对状态位没有影响。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	-	-	-	-

例如：直接寻址

STL		解 释
LAR1	DBD20	// 将数据双字 DBD20 中的指针装入 AR1 。
LAR1	DID30	// 将背景数据双字 DID30 中的指针装入 AR1 。
LAR1	LD180	// 将本地数据双字 LD180 中的指针装入 AR1。
LAR1	MD24	// 将存储数据双字 MD24 的内容装入 AR1。

例如：指针常数

STL		解 释
LAR1	P#M100.0	// 将一个 32 位指针常数装入 AR1。

9.6 LAR1 AR2 将地址寄存器 2 的内容装入地址寄存器 1

格式

LAR1 AR2

说明

使用该指令（带地址 AR2 的 LAR1 指令），可以将地址寄存器 AR2 的内容装入地址寄存器 AR1。累加器 1 和累加器 2 保持不变。指令的执行与状态位无关，而且对状态位没有影响。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	-	-	-	-

9.7 LAR2 将累加器 1 中的内容装入地址寄存器 2

格式

LAR2

说明

使用该指令，可以将累加器 1 的内容（32 位指针）装入地址寄存器 AR2。

累加器 1 和累加器 2 保持不变。指令的执行与状态位无关，而且对状态位没有影响。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	-	-	-	-

9.8 LAR2 <D> 将两个双整数(32位指针)装入地址寄存器2

格式

LAR2 <D>

地 址	数据类型	存储区	源地址
<D>	DWORD 指针常数	D，M，L	0...65532

说明

使用该指令，可以将寻址双字<D>的内容或指针常数装入地址寄存器 AR2。累加器 1 和累加器 2 保持不变。指令的执行与状态位无关，而且对状态位没有影响。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	-	-	-	-

例如：直接寻址

STL		解 释
LAR2	DBD 20	// 将数据双字 DBD20 中的指针装入 AR2。
LAR2	DID 30	// 将背景数据双字 DID30 中的指针装入 AR2。
LAR2	LD 180	// 将本地数据双字 LD180 中的指针装入 AR2。
LAR2	MD 24	// 将存储双字 MD24 中的指针装入 AR2。

例如：指针常数

STL		解 释
LAR2	P#M100.0	// 将一个 32 位指针常数装入 AR2。

9.9 T 传送

格式

T <地址>

地 址	数据类型	存储区	源地址
<地址>	BYTE WORD DWORD	I，Q，PQ，M，L， D	0...65535 0...65534 0...65532

说明

使用该指令，如果主控继电器接通（MCR = 1），可以将累加器 1 中的内容传送（复制）到目的地址。如果 MCR = 0，则目的地址写入“0”。从累加器 1 中复制的字节数量取决于目的地址规定的大小。在传送指令执行完后，累加器 1 还可保存数据。到直接 I/O 区的传送（存储器类型 PQ）也可将累加器 1 的内容或“0”（如果 MCR = 0）传送到过程映像输出表的相应地址（存储器类型 Q）。指令的执行与状态位无关，而且对状态位没有影响。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	-	-	-	-

举例

STL		解 释
T	QB10	// 将累加器 1 低字低字节中的内容传送到输出字节 QB10 中。
T	MW14	// 将累加器 1 低字中的内容传送到存储字 MW14。
T	DBD2	// 将累加器 1 中的内容传送到数据双字 DBD2。

9.10 T STW 将累加器 1 中的内容传送到状态字

格式

T STW

说明

使用 T STW（使用地址 STW 传送）指令，可以将累加器 1 的位 0 到位 8 传送到状态字。

指令的执行与状态位无关。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	x	x	x	x	x	x	x	x	x

举例

STL	**解 释**
T STW	// **将累加器 1 的位 0 到位 8 传送到状态字。**

累加器 1 中的位包含以下状态位：

位	31-9	8	7	6	5	4	3	2	1	0
内容	*)	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC

*） **该位不被传送**

9.11 CAR 交换地址寄存器 1 和地址寄存器 2 的内容

格式

CAR

说明

使用 CAR（交换地址寄存器）指令，可以将地址寄存器 AR1 和 AR2 中的内容进行交换。指令的执行与状态位无关，而且对状态位没有影响。

地址寄存器 AR1 中的内容移至地址寄存器 AR2 中，地址寄存器 AR2 中的内容移至地址寄存器 AR1 中。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	-	-	-	-

9.12 TAR1 将地址寄存器 1 中的内容传送到累加器 1

格式

TAR1

说明

使用该指令，可以将地址寄存器 AR1 的内容传送到累加器 1（32 位指针）。

累加器 1 的原有内容保存到累加器 2 中。指令的执行与状态位无关，而且对状态位没有影响。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	-	-	-	-

9.13 TAR1 <D>将地址寄存器1的内容传送到目的地(32位指针)

格式

TAR1 <D>

地　址	数据类型	存储区	源地址
<D>	DWORD	D，M，L	0...65532

说明

使用该指令，可以将地址寄存器 AR1 **的内容传送到寻址双字** <D>**。可能的目的区域有存储双字（**MD**）、本地数据双字（**LD**）、数据双字（**DBD**）和背景数据双字（**DID**）。**

累加器 1 **和累加器** 2 **保持不变。指令的执行与状态位无关，而且对状态位没有影响。**

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	-	-	-	-

举例

STL		解　释
TAR1	DBD20	// 将 AR1 中的内容传送到数据双字 DBD20。
TAR1	DID30	// 将 AR1 中的内容传送到背景数据双字 DID30。
TAR1	LD18	// 将 AR1 中的内容传送到本地数据双字 LD18。
TAR1	MD24	// 将 AR1 中的内容传送到存储双字 MD24。

9.14 TAR1 AR2 将地址寄存器1的内容传送到地址寄存器2

格式

TAR1 AR2

说明

使用该指令（使用地址 AR2 的 TAR1 指令），可以将地址寄存器 AR1 的内容传送到地址寄存器 AR2。

累加器 1 和累加器 2 保持不变。指令的执行与状态位无关，而且对状态位没有影响。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	-	-	-	-

9.15 TAR2 将地址寄存器 2 中的内容传送到累加器 1

格式

TAR2

说明

使用该指令，可以将地址寄存器 AR2 的内容传送到累加器 1（32 位指针）。

累加器 1 的原有内容保存到累加器 2 中。指令的执行与状态位无关，而且对状态位没有影响。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	-	-	-	-

9.16 TAR2 <D>将地址寄存器2的内容传送到目的地(32位指针)

格式

TAR2 <D>

地　址	数据类型	存储区	源地址
<D>	DWORD	D，M，L	0...65532

说明

使用该指令，可以将地址寄存器 AR2 的内容传送到寻址双字 <D>。可能的目的区域有存储双字（MD）、本地数据双字（LD）、数据双字（DBD）和背景双字（DID）。

累加器 1 和累加器 2 保持不变。指令的执行与状态位无关，而且对状态位没有影响。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	-	-	-	-

举例

STL		解　释
TAR2	DBD20	// 将 AR2 中的内容传送到数据双字 DBD20。
TAR2	DID30	// 将 AR2 中的内容传送到背景双字 DID30。
TAR2	LD18	// 将 AR2 中的内容传送到本地数据双字 LD18。
TAR2	MD24	// 将 AR2 中的内容传送到存储双字 MD24。

10 程序控制指令

10.1 程序控制指令概述

说明

下述程序控制指令可供使用：

- BE **块结束**
- BEC **条件块结束**
- BEU **无条件块结束**
- CALL **块调用**
- CC **条件调用**
- UC **无条件调用**

- **调用功能块**
- **调用功能**
- **调用系统功能块**
- **调用系统功能**
- **调用多背景块**
- **从库中调用块**

- MCR **（主控继电器）**
- **使用** MCR **功能的重要注意事项**
- MCR(**将** RLO **存入** MCR **堆栈，开始** MCR
-)MCR **结束** MCR
- MCRA **激活** MCR **区域**
- MCRD **去活** MCR **区域**

10.2 BE 块结束

格式

BE

说明

使用该指令，可以中止在当前块中的程序扫描，并跳转到调用当前块的程序块。然后从调用程序中块调用语句后的第一个指令开始，重新进行程序扫描。并将当前的本地数据区域释放，前一本地数据区域即成为当前本地数据区域。调用块时打开的数据块将被重新打开。另外，还恢复调用块的 MCR 相关性，并将 RLO 从当前块传送到调用当前块的程序块。该指令与任何条件无关。但是，如果该指令被跳转，则当前程序扫描不结束，而是从块内跳转到目的地处继续。

当用于 S5 软件时，该指令略有不同。当用于 S7 软件时，该指令的功能与 BEU 相同。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	0	0	1	-	0

举例

```
STL                    解  释
      A    I 1.0
      JC   NEXT        // 如果 RLO = 1 (I 1.0 = 1)，则跳转到 NEXT 跳转标号。
      L    IW4         // 如果没有执行跳转，则继此继续程序扫描。
      T    IW10
      A    I 6.0
      A    I 6.1
      S    M 12.0
      BE               // 块结束
NEXT: NOP 0            // 如果执行了跳转，则继此继续程序扫描。
```

10.3 BEC 条件块结束

格式

BEC

说明

如果 RLO = 1，**使用该指令，可以中断在当前块中的程序扫描，并跳转到调用当前块的程序块。然后从块调用语句后的第一个指令开始，重新进行程序扫描。并将当前的本地数据区域释放，前一本地数据区域即成为当前本地数据区域。调用块时打开的数据块将被重新打开。调用块的** MCR **相关性被恢复。**

RLO (= 1) **从被中止的块传送到调用块。如果** RLO = 0**，则不执行该指令。**RLO **被置为“** 1 **”，程序扫描从该指令后的下一指令继续。**

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	x	0	1	1	0

举例

STL	**解 释**
A I 1.0	// **刷新** RLO**。**
BEC	// **如果** RLO = 1**，结束块。**
L IW4	// **如果没有执行** BEC**，**RLO = 0**，则继此继续程序扫描。**
T MW10	

10.4 BEU 无条件块结束

格式

BEU

说明

使用该指令，可以中止在当前块中的程序扫描，并跳转到调用当前块的程序块。然后从块调用语句后的第一个指令开始，重新进行程序扫描。并将当前的本地数据区域释放，前一本地数据区域即成为当前本地数据区域。调用块时打开的数据块将被重新打开。另外，还恢复调用块的 MCR 相关性，并将 RLO 从当前块传送到调用当前块的程序块。该指令与任何条件无关。但是，如果该指令被跳转，则当前程序扫描不结束，而是从块内跳转目的地处继续。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	0	0	1	-	0

举例

```
STL                    解  释
      A    I 1.0
      JC   NEXT    // 如果 RLO = 1 (I 1.0 = 1)，则跳转到 NEXT 跳转标号。
      L    IW4     // 如果没有执行跳转，则继此继续程序扫描。
      T    IW10
      A    I 6.0
      A    I 6.1
      S    M 12.0
      BEU          // 无条件块结束
NEXT: NOP 0        // 如果执行了跳转，则继此继续程序扫描。
```

10.5 CALL 块调用

格式

CALL <**逻辑块标识符**>

说明

使用该指令，可以调用功能（FC**）或功能块（**SFB**）、系统功能（**SFC**）或系统功能块（**SFB**），或调用由西门子公司提供的标准预编程块。使用该指令，可以调用可作为地址输入的** FC **和** SFC **或** FB **和** SFB**，与** RLO **或其它条件无关。如果使用该指令调用一个** FB **或** SFB**，必须提供具有相关背景数据块的程序块。在被调用块处理完后，调用块程序继续逻辑处理。逻辑块的地址可以绝对指定，也可相对指定。在** SFB/SFC **调用后，保存寄存器的内容。**

例如：CALL FB1，DB1 **或** CALL FILLVAT1，RECIPE1

逻辑块	**块类型**	**绝对地址调用语法**
FC	**功能**	CALL FCn
SFC	**系统功能**	CALL SFCn
FB	**功能块**	CALL FBn1,DBn2
SFB	**系统功能块**	CALL SFBn1,DBn2

注意

如果使用的是语句表编辑器（STL Editor**），上表中的** n、n1 **和** n2 **必须是有效的现有块。同样，在使用之前必须定义符号名。**

传送参数（增量编辑方式）

调用块可通过一个变量表与被调用块交换参数。

当你输入一个有效的调用语句时，语句表程序中的变量表可自动扩展。

如果调用一个功能块（FB）、系统功能块（SFB）、功能（FC）或系统功能（SFC），并且被调用块的变量声明表中有 IN、OUT 和 IN_OUT 声明，则这些变量作为一个形式参数表被添加到调用块中。

如果调用的是一个功能（FC）和系统功能（SFC），则必须在调用逻辑块中为声明的形式参数赋值实际参数。

如果调用的是功能块（FB）和系统功能块（SFB），只需定义与以前调用相比必须进行修改的实际参数。在处理完功能块后，实际参数保存在背景数据块中。如果实际参数是一个数据块，则必须指定完整的绝对地址，例如 DB1，DBW2。

IN 参数可作为常数、绝对地址或符号地址定义。OUT 和 IN_OUT 参数必须作为绝对地址或符号地址定义。必须保证所有地址和常数与要传送的数据类型相符。

调用指令可将返回地址（选择符和相对地址）、两个当前数据块的选择符以及 MA 位保存在 B（块）堆栈中。除此之外，调用指令还可去活 MCR 的相关性，然后生成被调用块的本地数据范围。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	0	0	1	-	0

举例 1：为 FC6 调用赋值参数

```
CALL    FC6
        形式参数        实际参数
        NO OF TOOL      := MW100
        TIME OUT        := MW110
        FOUND           := Q 0.1
        ERROR           := Q 100.0
```

举例 2：无参数调用一个系统功能（SFC）

```
STL                     解  释
CALL    SFC43           // 调用 SFC43，重新触发看门狗定时器（无参数）。
```

举例 3：使用背景数据块 DB1 调用 FB99

```
CALL    FB99，DB1
        形式参数        实际参数
        MAX_RPM         := #RPM1_MAX
        MIN_RPM         := #RPM1
        MAX_POWER       := #POWER1
        MAX_TEMP        := #TEMP1
```

举例 4：使用背景数据块 DB2 调用 FB99

```
CALL    FB99，DB2
        形式参数        实际参数
        MAX_RPM         := #RPM2_MAX
        MIN_RPM         := #RPM2
        MAX_POWER       := #POWER2
        MAX_TEMP        := #TEMP2
```

注意

每一次功能块（FB）或系统功能块（SFB）调用都必须有一个背景数据块。在上述举例中，数据块 DB1 和 DB2 必须在调用之前已存在。

10.6 调用功能块

格式

CALL FB n1，DB n1

说明

使用该指令，可调用用户定义的功能块（FB）。调用指令能够调用你作为地址输入的功能块，与 RLO 或其它条件无关。如果使用调用指令调用一个功能块，必须为它提供一个背景数据块。在处理完被调用块后，调用块程序继续处理。逻辑块的地址可以绝对指定，也可相对指定。

传送参数（增量编辑方式）

调用块可通过一个变量表与被调用的块交换参数。当你输入一个有效的调用语句时，语句表程序中的变量表可自动扩展。

如果调用一个功能，并且调用块的变量声明表中有 IN、OUT 和 IN_OUT 声明，则这些变量作为一个形式参数表被添加到用于调用块的程序中。

由于在功能块处理完之后，实际参数保存在背景数据块中，当调用功能块时，只需定义与以前调用相比必须修改的实际参数。如果实际参数是一个数据块，则必须指定完整的绝对地址，例如 DB1，DBW2。

IN 参数可作为常数、绝对地址或符号地址定义。OUT 和 IN_OUT 参数必须作为绝对地址或符号地址定义。必须保证所有地址和常数与要传送的数据类型相符。

调用指令可将返回地址（选择符和相对地址）、两个当前数据块的选择符以及 MA 位保存在 B（块）堆栈中。除此之外，调用指令还可去活 MCR 的相关性，然后生成被调用块的本地数据范围。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	0	0	1	-	0

举例 1：使用背景数据块 DB1 调用 FB99

```
CALL    FB99，DB1
        形式参数        实际参数
        MAX_RPM         := #RPM1_MAX
        MIN_RPM         := #RPM1
        MAX_POWER       := #POWER1
        MAX_TEMP        := #TEMP1
```

举例 2：使用背景数据块 DB2 调用 FB99

```
CALL    FB99，DB2
        形式参数        实际参数
        MAX_RPM         := #RPM2_MAX
        MIN_RPM         := #RPM2
        MAX_POWER       := #POWER2
        MAX_TEMP        := #TEMP2
```

注意

每一次功能块（FB）调用都必须有一个背景数据块。在上述举例中，数据块 DB1 和 DB2 必须在调用之前已存在。