8.1 基本功能指令

8.1.1 使用组织块 OB

组织块 0B 是操作系统与用户程序的接口，由操作系统调用。组织块中除可以用来实现 PLC 循环扫描控制以外，还可以完成 PLC 的启动、中断程序的执行和错误处理等功能。

（1）事件和组织块

事件是 S7-1200 PLC 操作系统运行的基础，分为能够启动 OB 的和无法启动 OB 的两类事件。

用户程序循环取决于事件和给这些事件分配的 OB，以及包含在 OB 中的程序代码或在 OB 中调用的程序代码。表 8.1 所示为能够启动 OB 的事件，其中包括相关的事件类别。

表 8.1 能够启动 OB 的事件

事件类别	OB 号	OB 数目	启动事件	OB 优先
循环程序	1，〉=200	〉=1	启动或结束上一个循环 OB	1
启动	100，〉=200	〉=0	STOP 到 RUN 的转换	1
延时中断	〉=200	最多 4 个	延时时间结束	3
循环中断	〉=200		等长间隔时间结束	4
硬件中断	〉=200	最多 50 个（通过 DETACH 和 ATTACH 指令可使用更多）	上升沿（最多 16 个） 下降沿（最多 16 个）	5
			HSC：计数值 = 参考值（最多 6 次） HSC：计数方向变化（最多 6 次） HSC：外部复位（最多 6 次）	6
中断错误中断	82	0 或 1	模块检测到错误	9
时间错误	80	0 或 1	超出最大循环时间	26
			仍在执行所调用的 OB 队列溢出，因中断负载过高而导致中断丢失	

无法启动 OB 的事件见表 8.2，其响应由操作系统完成。

表 8.2 无法启动 OB 的事件

事件类别	事 件	事件优先级	系统响应
插入 / 卸下	插入 / 卸下模块	21	STOP
访问错误	过程映像更新期间的 I/O 访问错误	22	忽略
编程错误	块中的编程错误（如果激活了本地错误处理，则会执行块中的错误程序）	22	STOP
I/O 访问错误	块中的 I/O 访问错误（如果激活了本地错误处理，则会执行块程序中的错误处理程序）	24	STOP
超出最大循环时间两倍	超出最大循环时间两倍	27	STOP

（2）启动组织块

接通 CPU 后， S7-1200 PLC 在开始执行用户程序之前首先执行启动程序，可以在启动 OB 中完成程序的初始化。

S7-1200 PLC 支持三种启动模式：不重新启动模式、暖启动 - RUN 模式和暖启动断电前的工作模式。不管选择哪种启动模式，已编写的所有启动 OB 都会执行。

表 8.3 启动 OB 声明表中变量的含义

变量	类型	描述
LostRetentive	BOOL	=1，如果保持性数据存储区已丢失
LostRTC	BOOL	=1，如果实时时钟已丢失

【例 8-1 】 S7-1200 PLC 中要利用实时时钟，如交通灯不同时间段切换不同的控制策略等，则启动运行时，需要检测实时时钟是否丢失，若丢失，则警示灯 Q0.0 亮。

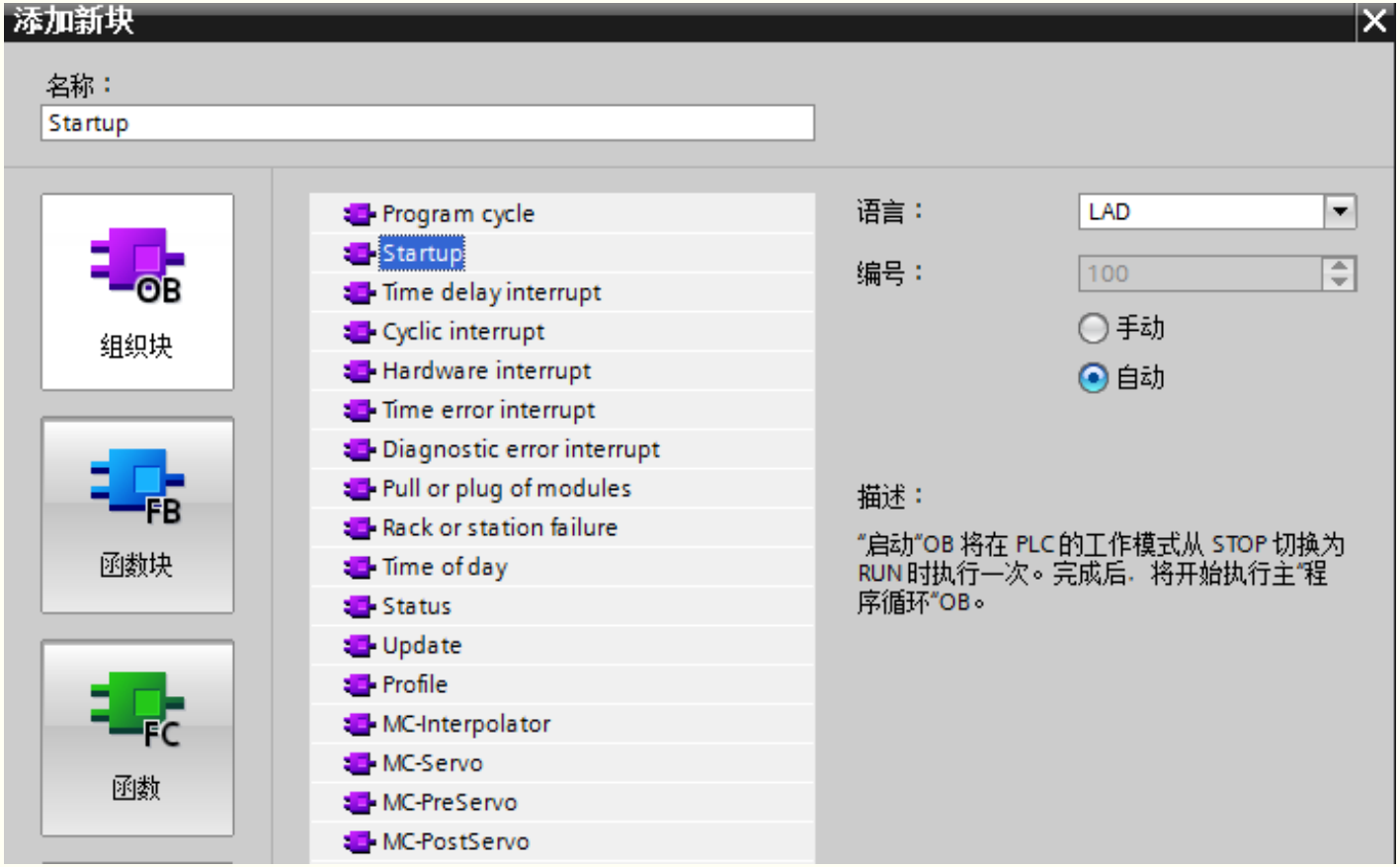


图 8.1 新建启动组

在 OB100 中编写程序如图 8.2 所示，则当 S7-1200 PLC 从 STOP 转到 RUN 时，若实时时钟丢失则输出 Q0.0 指示灯亮。

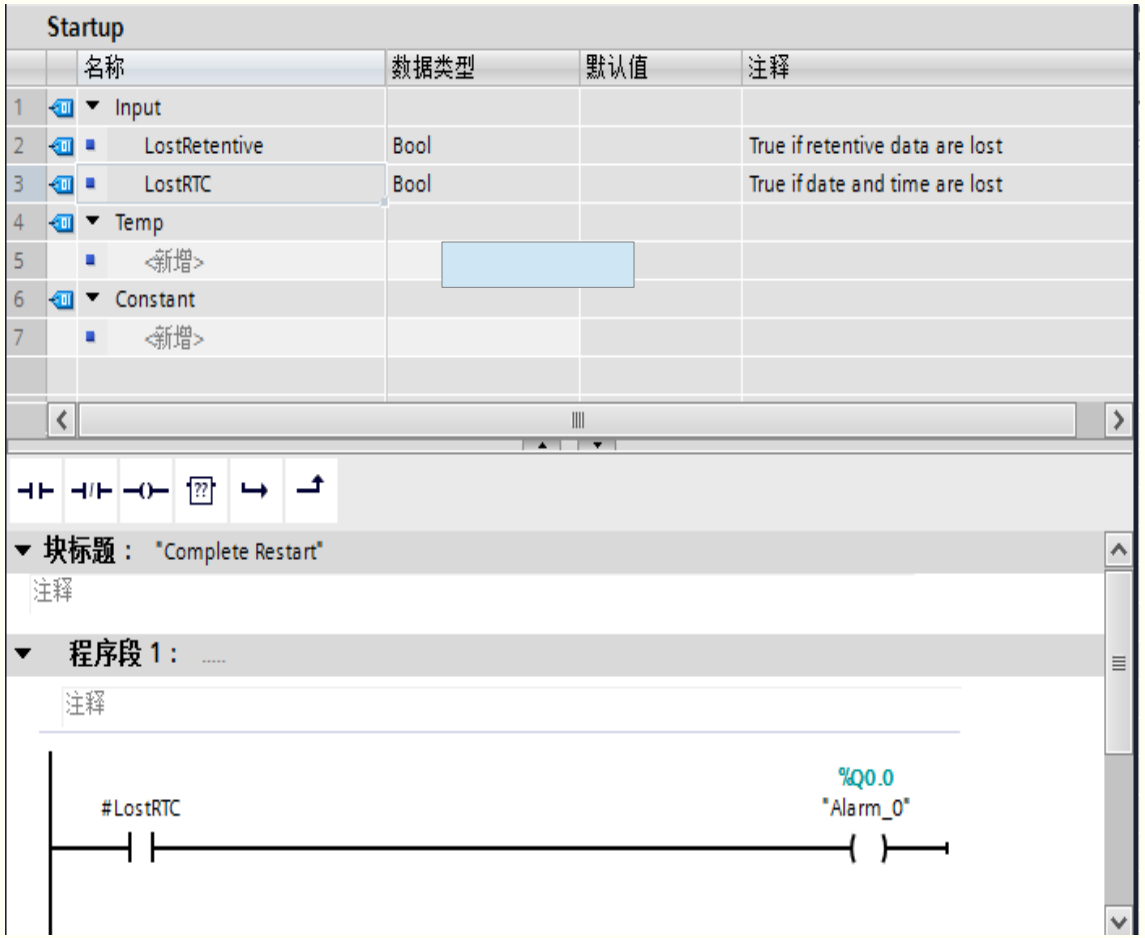


图 8.2　启动 OB 的局部变量和应用

（3）循环中断组织块

循环中断组织块用于按一定时间间隔循环执行中断程序，例如周期性地定时执行闭环控制系统的 PID 运算程序等。

【例 8-2 】使用循环中断组织块，每隔 1.5 秒 MW20 的值加 1 。

在项目视图项目树中，双击 PLC 设备程序块下的“添加新块”项，选择添加“ Cyclic·interrupt” 类型的 OB 块，用手动方式设置程序的编号，并将循环时间设为 1.5 秒，如图 8.3 所示，对应的 PLC 程序如图 8.4 。

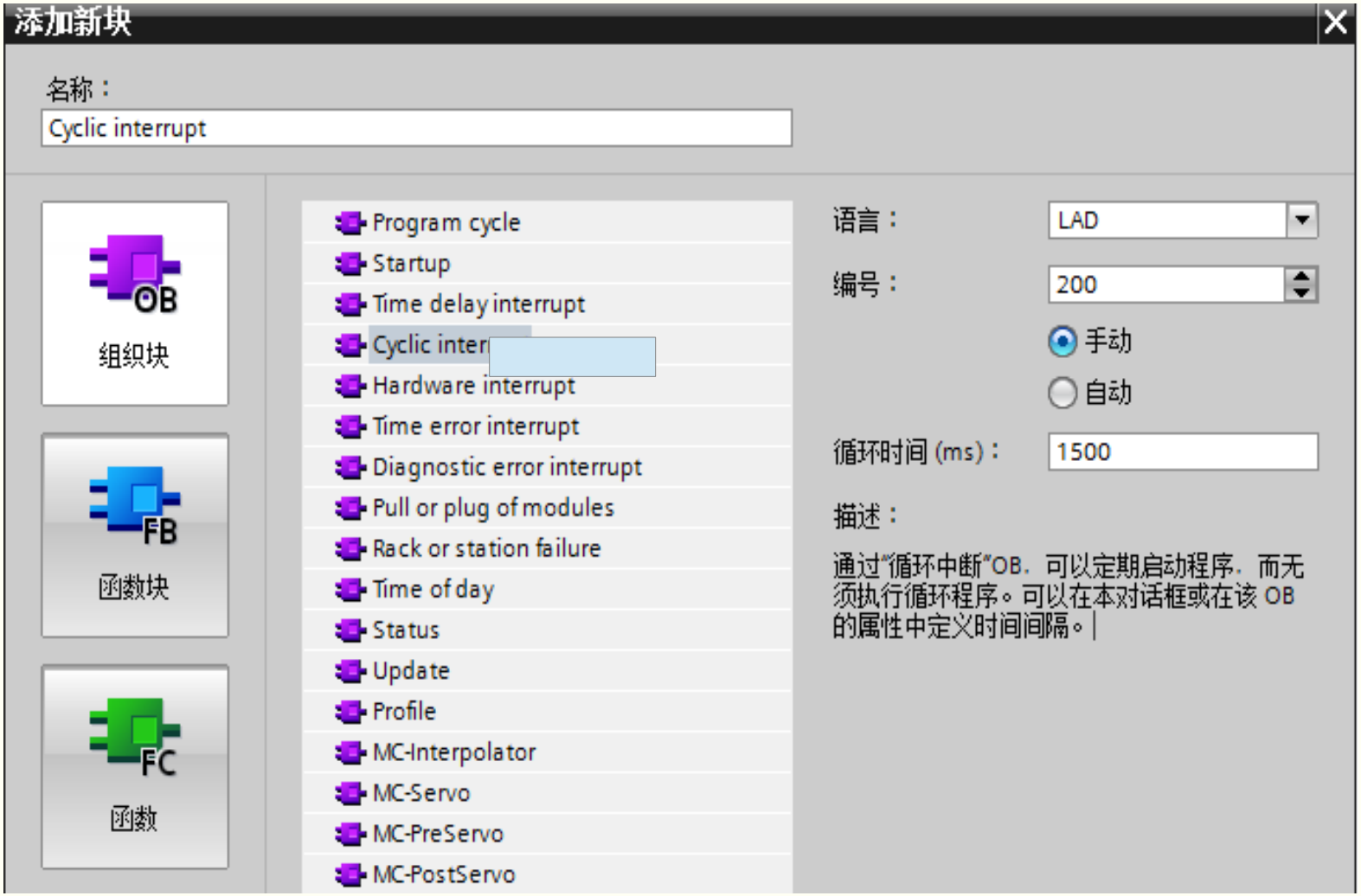


图 8.3 设置循环中断组织块属性

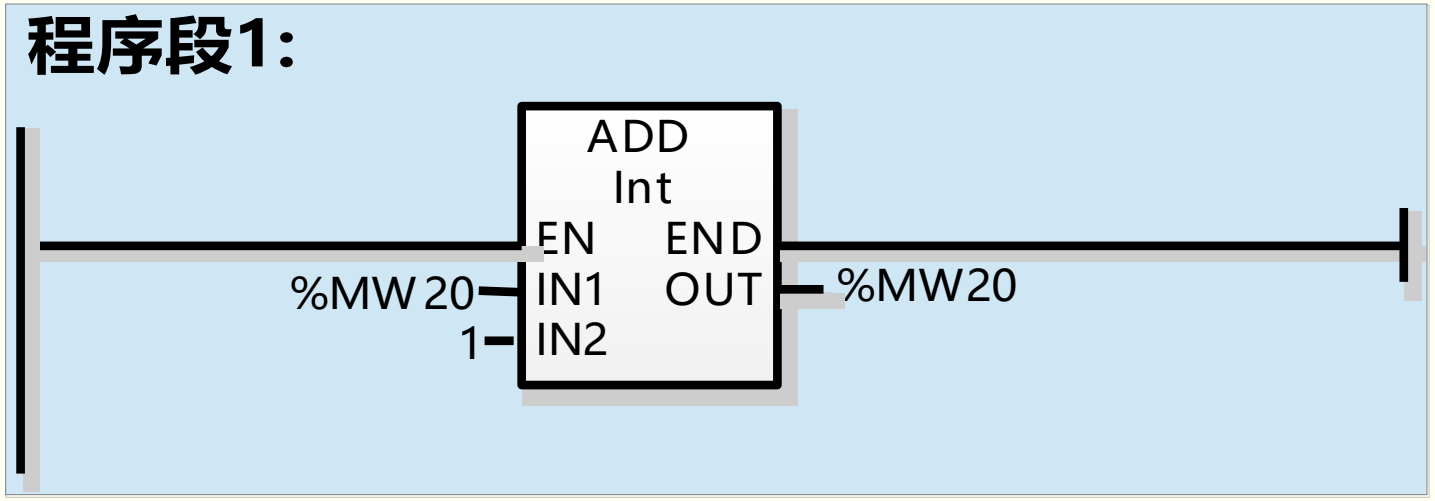


图 8.4 OB200 程序

（4）硬件中断组织块

硬件中断 OB 用来响应特定事件，最多可使用 50 个硬件中断 OB，它们在用户程序中彼此独立。使用时只能将触发报警的事件分配给一个硬件中断 OB，而一个硬件中断 OB 可以对应给多个事件。

【例 8-3 】新建一个硬件中断组织块 OB200，通过硬件中断在 I0.0 上升沿时将 Q1.0 置位，在 I0.1 下降沿时将 Q1.0 复位。

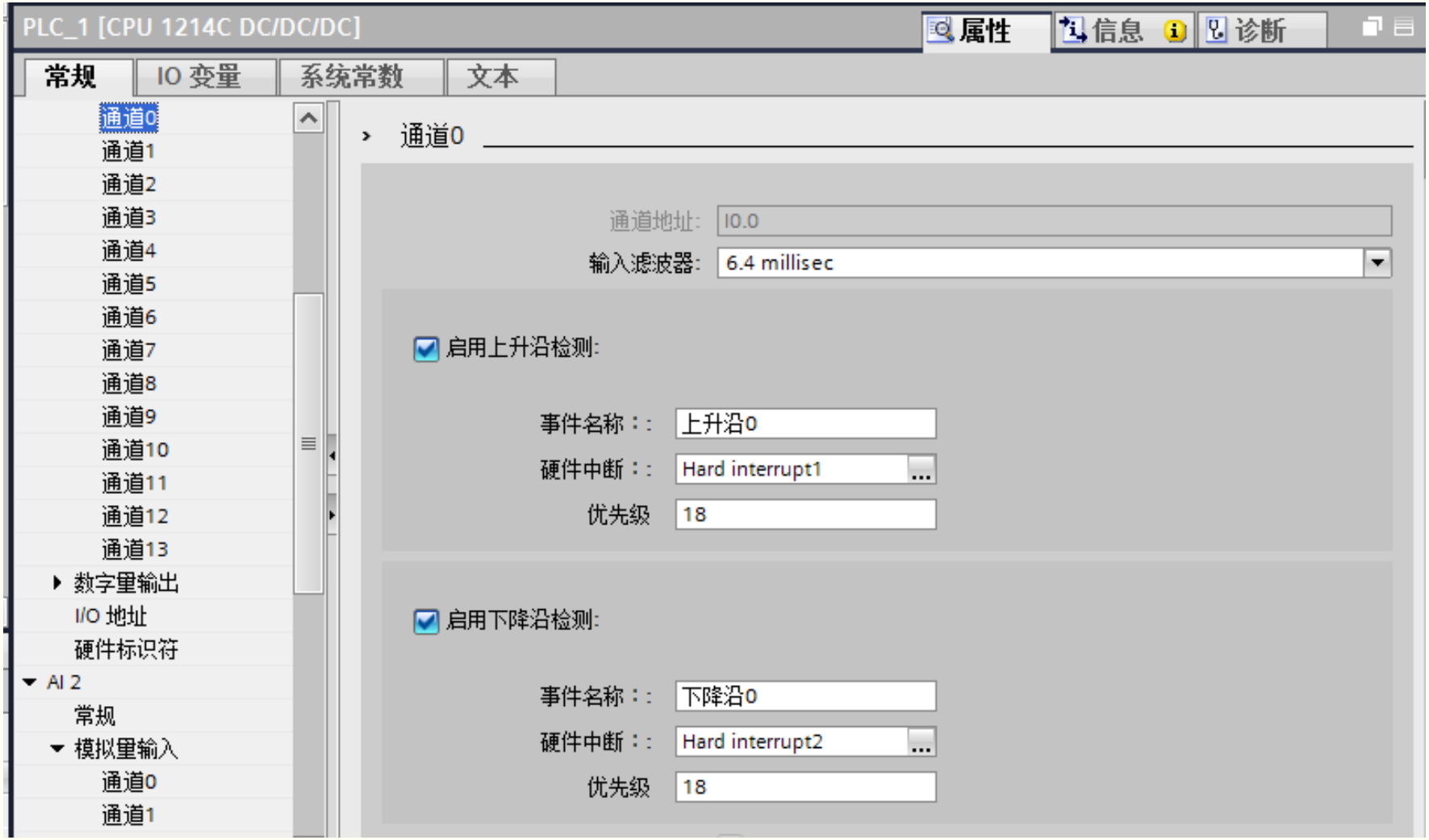
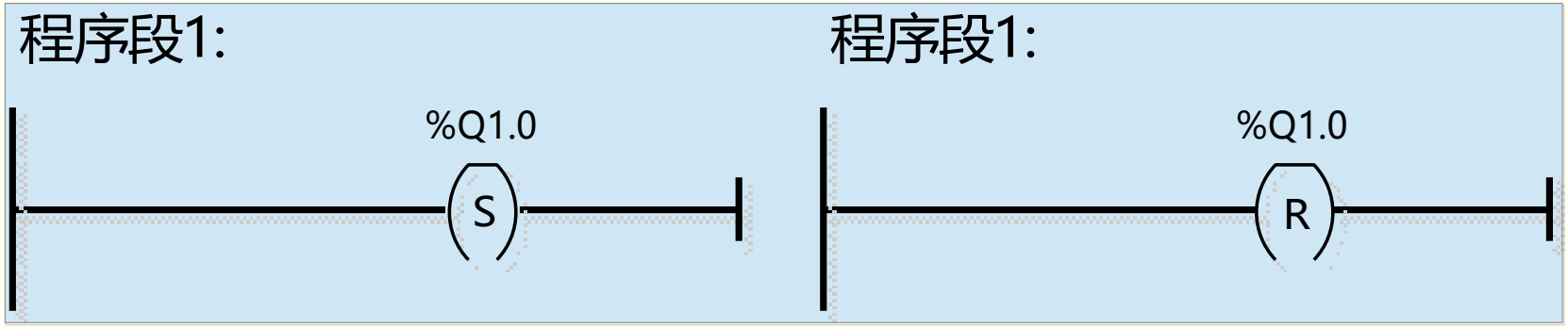

PLC_1 [CPU 1214C DC/DC/DC]
属性
信息
诊断
常规
IO 变量
系统常数
文本
通道0
通道1
通道2
通道3
通道4
通道5
通道6
通道7
通道8
通道9
通道10
通道11
通道12
通道13
数字量输出
I/O 地址
硬件标识符
AI 2
常规
模拟量输入
通道0
通道1
通道0
通道地址: I0.0
输入滤波器: 6.4 millisec
启用上升沿检测:
事件名称：: 上升沿0
硬件中断：: Hard interrupt1
优先级 18
启用下降沿检测:
事件名称：: 下降沿0
硬件中断：: Hard interrupt2
优先级 18

图 8.5 设置硬件中断

在 OB200 中编写程序，如图 8.6a 所示，在 OB201 中编写程序，如图 8.6b 所示。



a ）置位 Q1.0　b ）复位 Q1.0

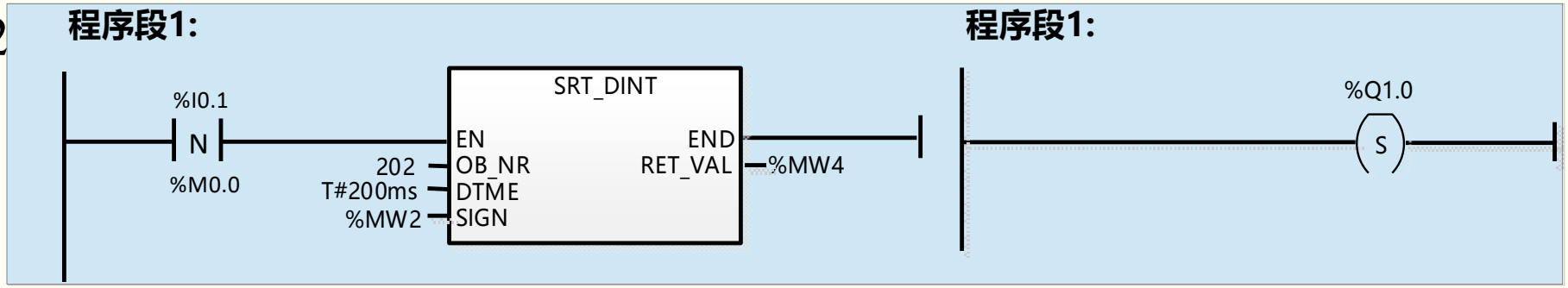
图 8.6　编写程序

（5）延时中断组织块

延时中断组织块用来编写延时中断程序，在某个特定事件出现时，如特定的输入出现下降沿或上升沿，可以通过 SRT_DINT 指令启动延时中断程序，延时时间在 SRT_DINT 中指定，时间精度为 1ms，同时还可通过参数 RET_VAL 检测指令执行是否正常。

【例 8-4 】在 I0.1 的下降沿启动延时中断程序 OB202 的调用，

2



a） OB1 程序　b） OB202 程序

图 8.7　示例程序

（6）时间错误组织块

在用户程序中只能使用一个时间错误中断组织块，在以下事件发生时，操作系统将自动进行调用：

1）循环程序超出最大循环时间。

2）被调用 0B （如延时中断 OB 和循环中断 0B ）当前正在执行。

3）中断 OB 队列发生溢出。

4）由于中断负载过大而导致中断丢失。

时间错误中断 OB 的启动信息含义如下表所示。

表 8.4 时间错误中断 OB 启动信息

变　量	数据类型	描述
Fault_id	BYTE	0x01：超出最大循环时间 0x02：仍在执行被调用 OB 0x07：队列溢出 0x09: 中断负载过大导致中断丢失
Csg_OBnr	OB_ANY	出错时要执行的 OB 编号
Csg_prio	UINT	出错时执行的 OB 的优先级

（7）诊断组织块

可以为具有诊断功能的模块启用诊断错误中断功能，使模块能检测到 I/O 状态变化，因此模块会在出现故障（进入事件）或故障不再存在（离开事件）时触发诊断错误中断。如果没有其他中断 OB 激活，则调用诊断错误中断 OB；若已经在执行其他中断 OB，诊断错误中断将置于同优先级的队列中。

在用户程序中只能使用一个诊断错误中断 OB。诊断错误中断 OB 的启动信息如表 8.5 所示。表 8.6 列出了局部变量 IO-state 所能包含的可能 I/O 状态。

表 8.5 诊断错误中断 OB 启动信息

变量	数据类型	描述
IO_state	WORD	包含具有诊断功能的模板的 I/O 状态
laddr	HW_ANY	HW-ID
Channel	UINT	通信编号
multi_error	BOOL	为 1 表示有多个错误

表 8.6 IO_state 状态

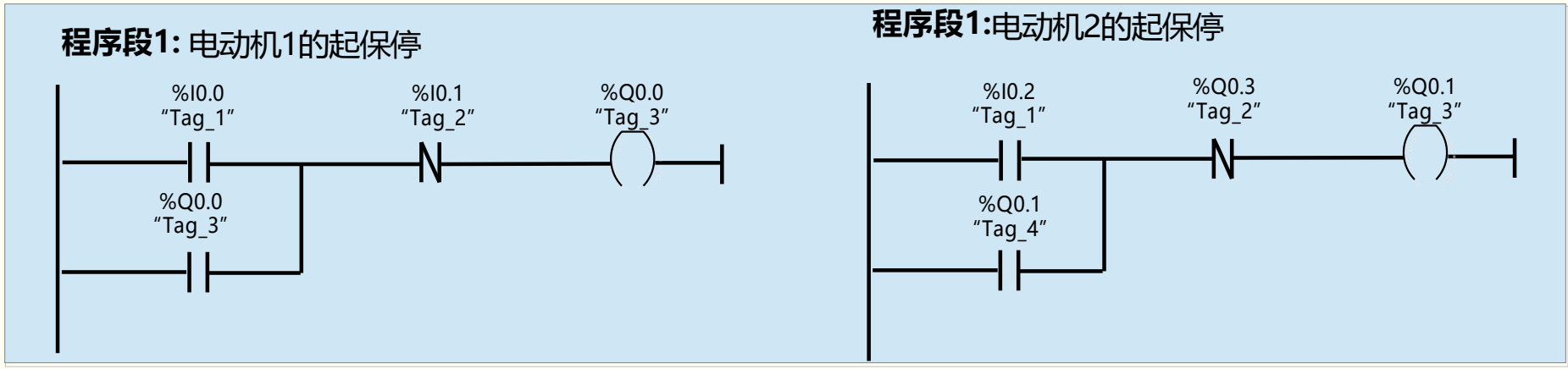
IO_state	含义
位 0	组态是否正确，为 1 表示组态正确
位 4	为 1 表示存在错误，如断路等
位 5	为 1 表示组态不正确
位 6	为 1 表示发生了 I/O 访问错误，此时程序包含存在访问错误 I/O 的硬件标识符

8.1.2 使用功能 FC 和功能块 FB

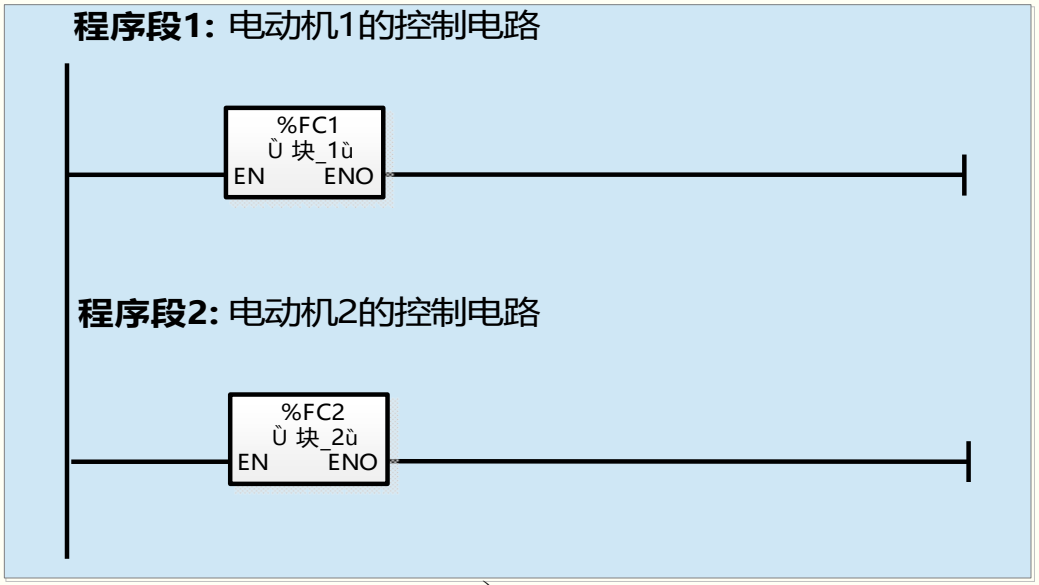
PLC 有三种编程方法：线性化编程、模块化编程和结构化编程。

【例 8-5 】有两台电动机，控制模式是相同的：按下启动按钮（电动机 1 对应 I0.0 ，电动机 2 对应 I0.2 ），电动机启动运行（电动机 1 对应 Q0.0 ，电动机 2 对应 Q0.1 ），按下停止按钮（电动机 1 对应 I0.1 ，电动机 2 对应 I0.3 ），电动机停止运行。

根据模块化编程的思想，分别在 FC1 和 FC2 两个子程序中利用启保停电路设计不同电机的控制程序，如图 8.8a 和图 8.8b 所示，最后在主程序 OB1 中调用此两程序。



a）



c）

图 8.8 电动机控制的模块化编程

【例 8-6 】采用模块化编程思想实现公式：$\sqrt{a^3 + b^2}$ 。

首先建立一个子程序 FC3 ，实现上式的功能，然后在主程序 OB1 中进行调用，这就是模块化程序设计的思想。

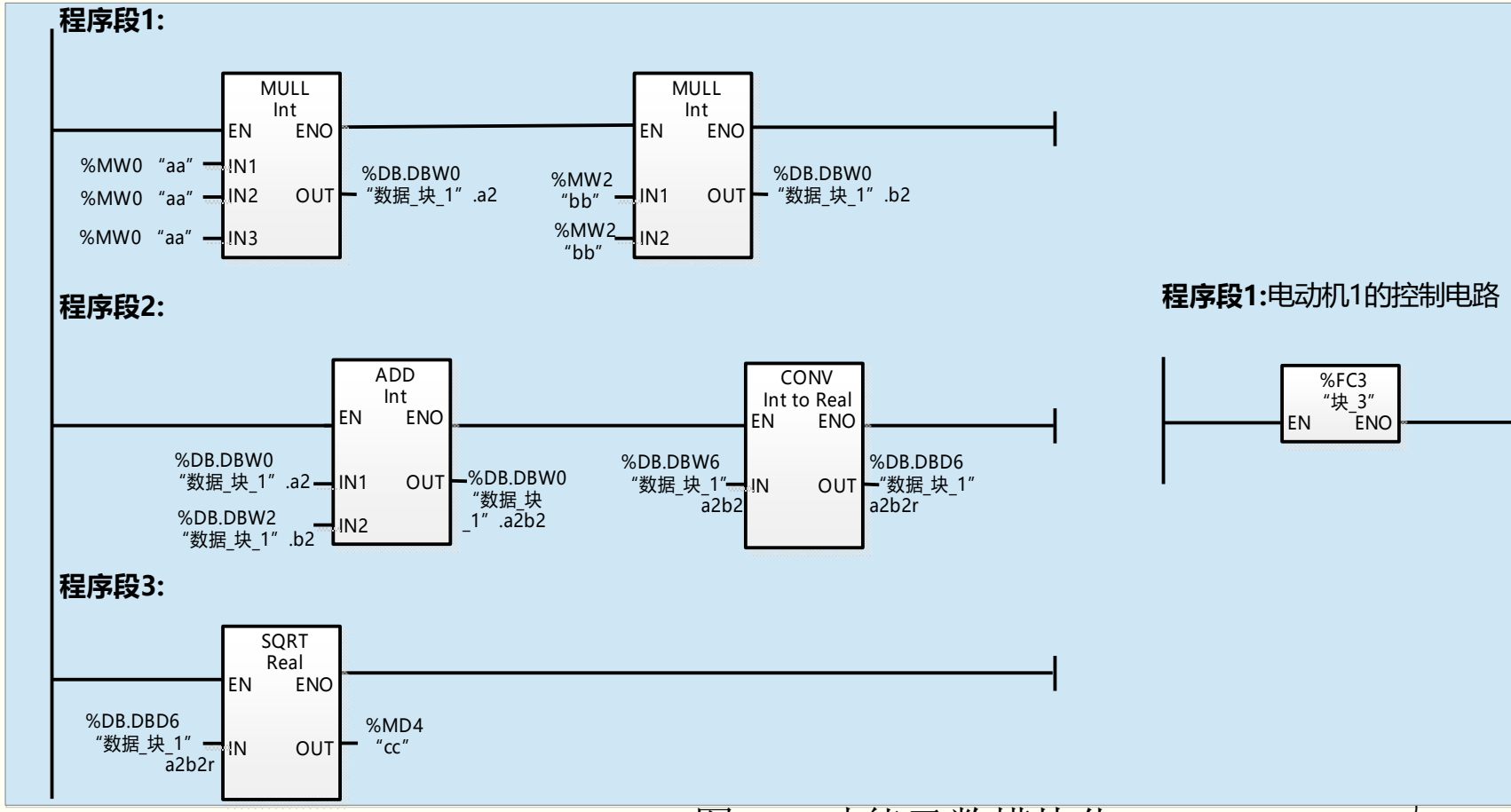
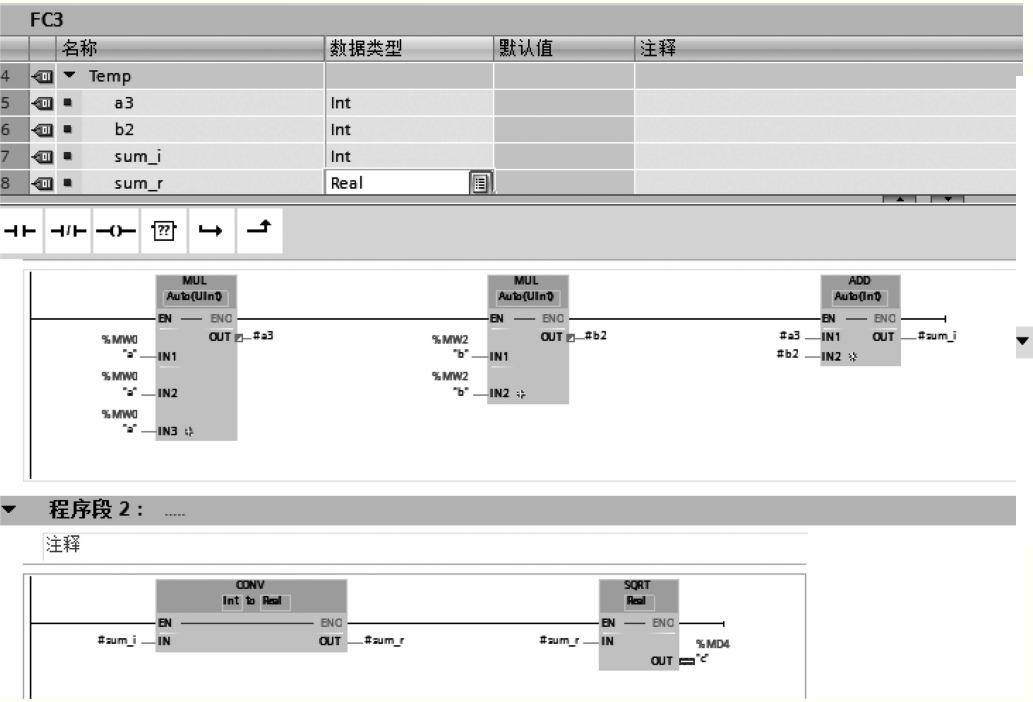


图 8.9 功能函数模块化编程

（2）临时变量

临时变量可以用于所有块（OB、FC、FB）中。当块执行的时候，它们被用来存储临时数据，退出该块时，这些数据将丢失。

临时变量在块的变量声明表中定义。点击程序编辑器工具栏间的上下箭头，可以收缩或展开块的变量声明表，其中 Input 为输入参数， Output 为输出参数， InOut 为输入输出参数， Temp 为临时工作变量。在此程序设计中，定义了整型临时变量 a3 ， b2 和 sum_i ，以及实型临时变量 sum_r ，分别存储变量的立方、平方，以及它们的和。程序相对简单，在此不再赘述。定义好临时变量后，在子程序中就能使用所定义的临时变量和全局变量，临时变量的标识是在前面加 # ，全局变量的标识是在前面加 % ，当二者同名时，优先使用临时变量，除非设计人员人为选择全局变量



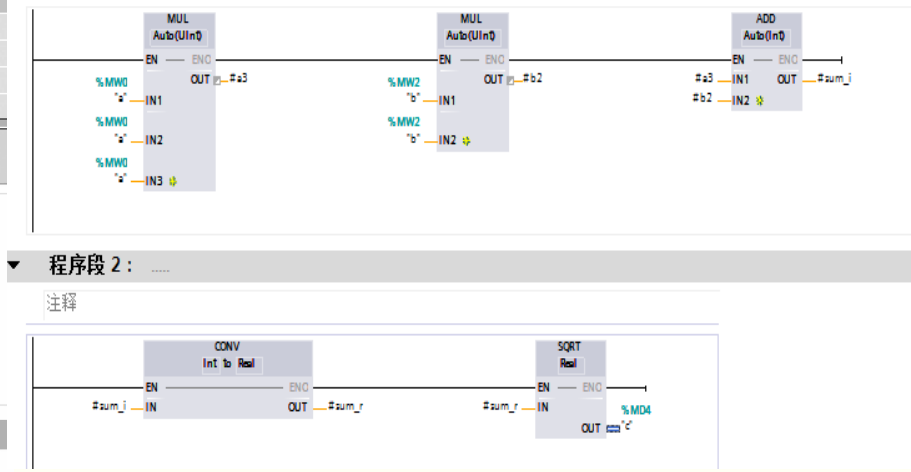


图 8.10　定义临时变量编程

（3）结构化编程

当任务要求中出现多个类似的功能时，采用模块化编程方法将不可避免地出现大量的重复代码，这时可以利用结构化编程方法对其进行优化，即对功能块的输入输出进行封装，当采用不同的参数进行功能（FC、FB）调用时，程序实现对不同对象的控制，这就是结构化编程的意义所在。

结构化编程有如下优点：

1）程序只需生成一次，显著减少了编程时间。

2）该块只在用户存储器中保存一次，显著降低了存储容量。

3）该块可以利用不同的参数多次调用，完成性质相同的一类功

形式参数的类型及作用列于表 8.7 。

表 8.7 形式参数的类型

参数类型	定义	使用方法	图形显示
输入参数	Input	只读，将数据传入程序	在块的左侧
输入参数	Output	只写，将数据传出程序	在块的右侧
输入 / 输出参数	InOut	可读写，数据的输入输出	在块的左侧
返回参数	Return	只写，将数据传出程序	在块的右侧
临时参数	Temp	可读写，程序内使用	不显示

【例 8-7 】用结构化编程方法重新编写前述电动机的控制电路程序。新建块 FC4 ： Motor ， 其形式参数定义和程序实现示意如下。

Motor

		名称	数据类型	默认值
1		Input		
2		start	Bool	
3		stop	Bool	
4		Output		
5		<新增>		
6		InOut		
7		motor	Bool	

图 8.11 结构化编程中的参数定义

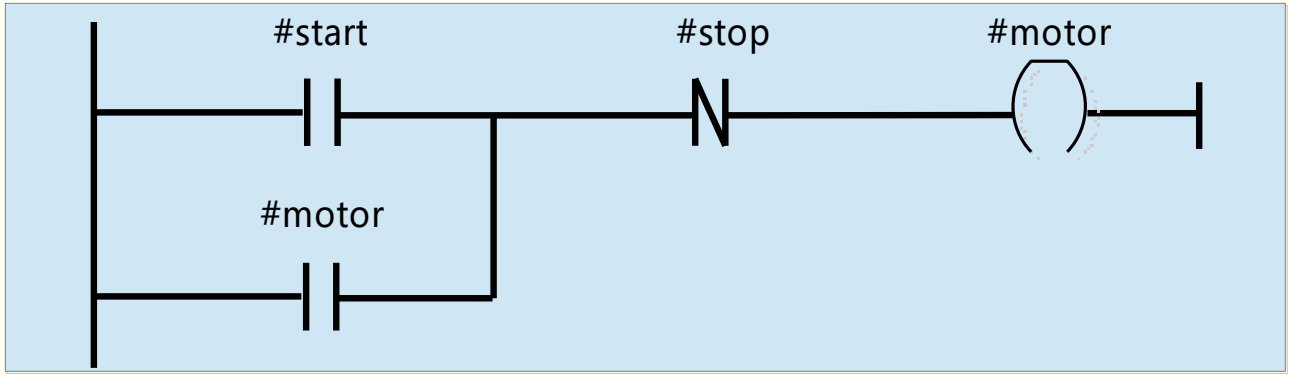


图 8.12 结构化编程中的程序设计

要注意以下问题：

1）如果在编辑一个块的程序时使用符号名，编辑器将在该块的变量声明中查找该符号名。如果该符号名存在，编辑器将把它当做局部变量，并在符号名前加“ #”号。

2）如果它不属于局部变量，则编辑器将在全局符号表中搜索。如果找到该符号名，编辑器将把它当做全局变量，并在符号名上加引号。

3）如果在全局变量表和变量声明表中使用了相同的符号名，编辑器将始终把它当做局部变量，除非输入该符号名时加了引号，则成为全局变量。

与模块化方法在使用上的不同就是，结构化编程中的输入输出参数是可以根据需要进行改变的，因而程序可以重用。下图为在 OB1 中调用函数 FC4:Motor 控制 2 台电动机。

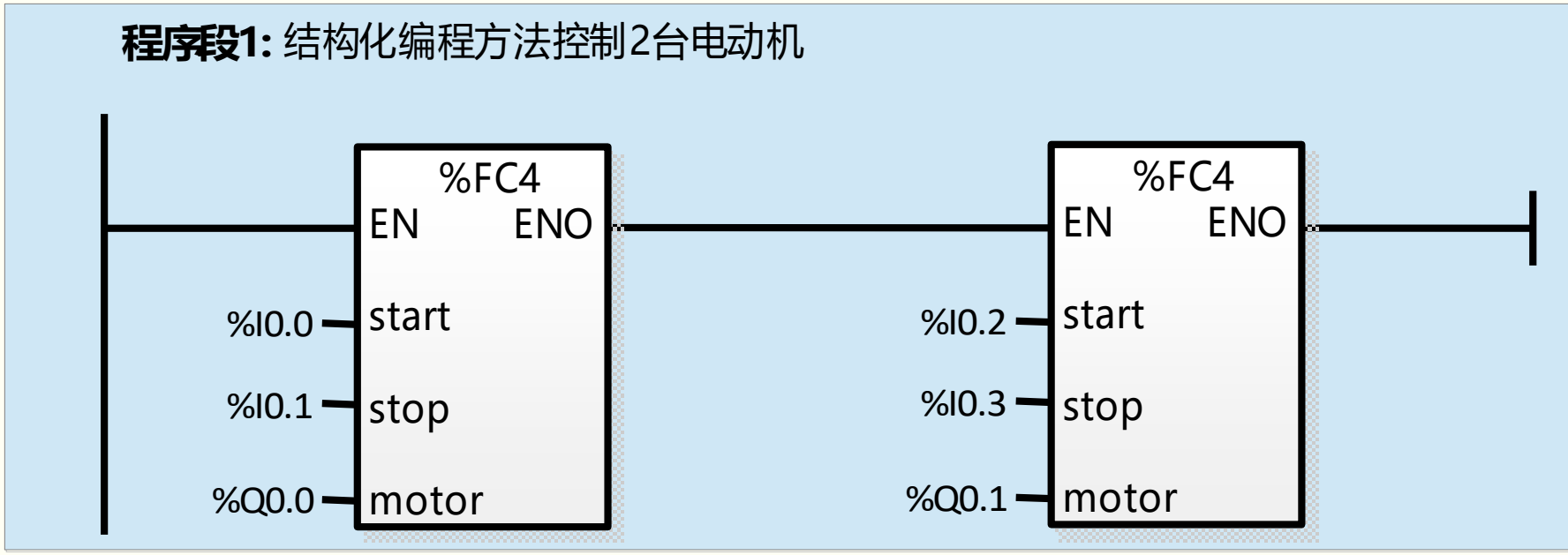


图 8.13 结构化程序设计中的程序调用

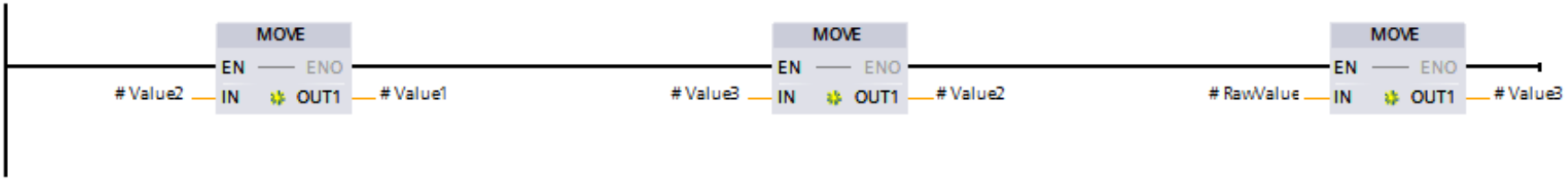
【例 8-8 】在工业生产中，经常需要对采集的模拟量进行滤波处理。本例将最近采集的三个采样值进行均值滤波，即将三个采样值求和，然后除以 3 。假设最新采集的模拟量工程值存储在 MD0 中，处理结果存储在 MD16 中，所有数据均为浮点型。

编程思路：利用堆栈概念，将最近采集的三个数保存在固定的全局地址中，并在每个控制周期进行数据更新，以确保参与运算的数据是最新的，因而此子程序 FC5 可以在循环中断处理程序中调用。

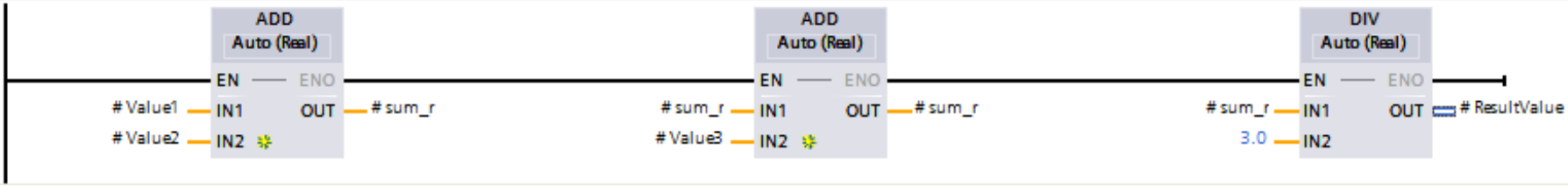
FC5

	名称	数据类型	默认值	注释
1	Input			
2	RawValue	Real		
3	Output			
4	ResultValue	Real		
5	InOut			
6	Value1	Real		
7	Value2	Real		
8	Value3	Real		

程序段 3：

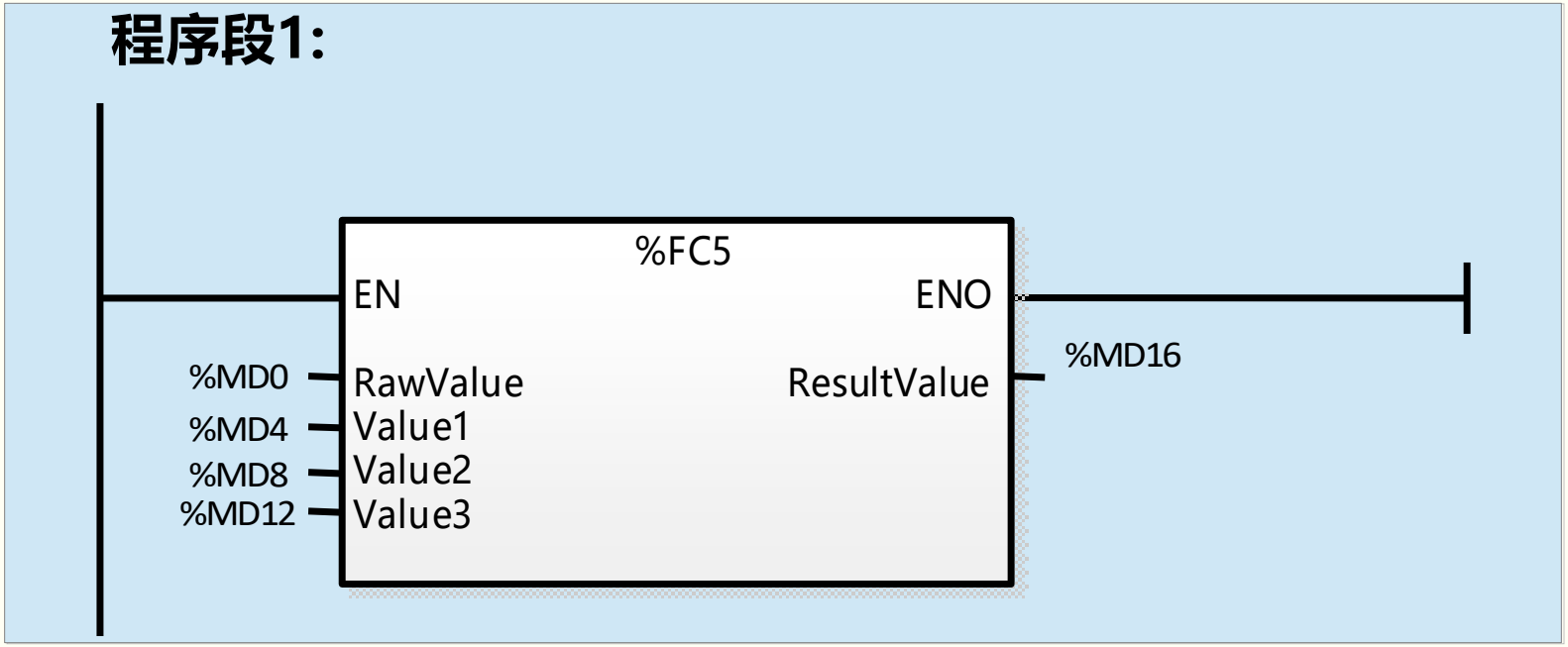


程序段 4：



a ）

图 8.14b 示意了调用 FC5 并赋值实际参数，将平均值存放在 MD16 中。这样，通过不同的实际参数可以重复调用 FC5 进行均值滤波。



b ）

图 8.14 子程序 FC5 b ）主程序中调用

（4） FB 的使用

FB 不同于 FC 之处是它带有一个存储区，也就是说，有一个局部数据块被分配给 FB，这个数据块称为背景数据块（Instance Data Block）。

每次调用 FB 时可以指定不同的实际参数。当块退出时，背景数据块中的数据仍然保持。可以看到， FB 具有以下优点：

1）当编写 FC 程序时，必须寻找空的标志区或数据区来存储需保持的数据，并且要自己编写程序来保存，而 FB 中的静态变量可由系统自动保存。

2）使用静态变量可避免两次分配同一存储区的危险。

结合前面例子，如果用 FB 块实现 FCl 的功能，并用静态变量 EarlyValue 、 LasfValue 和 LalestValue 来代替原来的形式参数，如表 8.8 所示，将可省略这三个形式参数，简化了块的调用。

表 8.8 定义 FB 的形式参数

参数类型	名称	数据类型	注释
IN	RawValue	REAL	要处理的原始数值
STAT	EarlyValue	REAL	最早的一个数
STAT	LastValue	REAL	较早的一个数
STAT	LatestValue	REAL	最近的一个数
OUT	ProcessedValue	REAL	处理后的数
TEMP	Temp1	REAL	中间结果
TEMP	Temp2	REAL	中间结果

在 FB1 中定义形式参数，编写程序同图 8.14 a，图 8.15 所示为调用 FB1 子程序，其中 DB10 为 FB1 的背景数据块，在输入时若 DB10 不存在则将自动生成该背景数据块。

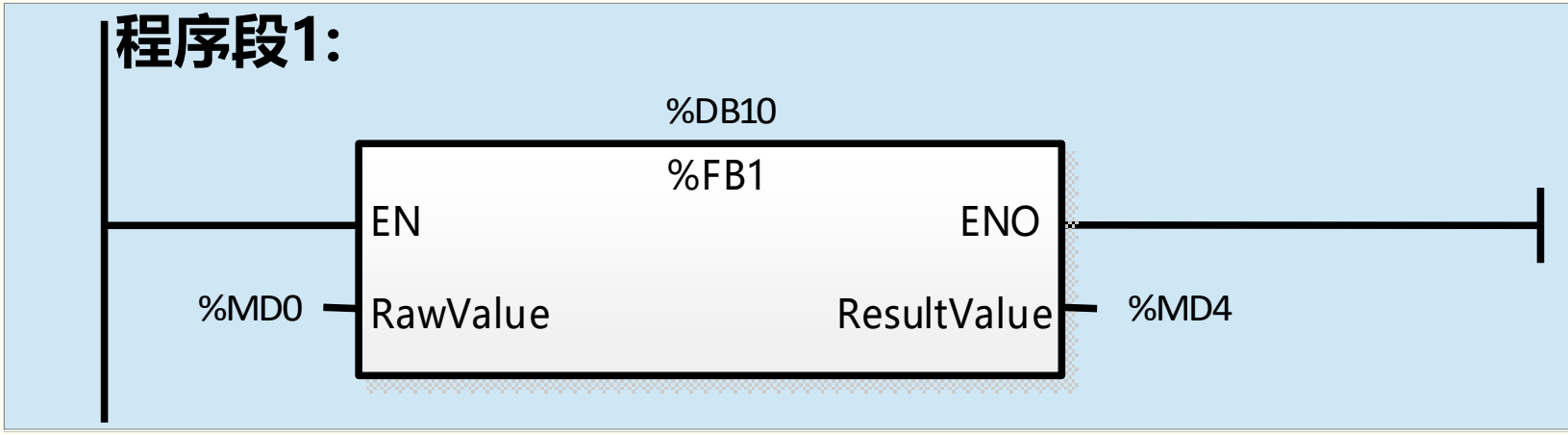


图 8.15 调用 FB1 子程序

双击打开背景数据块 DB10，可以看到 DB10 中保存的正是在 FB 的接口中定义的形式参数，如图 8.16 所示。对于背景数据块，无法进行编辑修改．而只能读写其中的数据。

		名称	数据类型	起始值	保持	可从 HMI/...	从 H...	在 HMI ...	设定值	注
1		Input								
2		RawValue	Real	0.0		☑	☑	☑		
3		Output								
4		ResultValue	Real	0.0		☑	☑	☑		
5		InOut								
6		Static								
7		EarlyValue	Real	0.0		☑	☑	☑		
8		LastValue	Real	0.0		☑	☑	☑		
9		LatestValue	Real	0.0		☑	☑	☑		

图 8.16 背景数据块

（5）检查块的一致性

如果在程序生成期间或之后调整或增加某个块（FC 或 FB）的接口或代码，可能导致时间标签冲突。反过来，时间标签冲突可能导致在调用的和被调用的或有关的块之间不一致。针对这种情况，当一个块已在程序中被调用之后，再增加或删除块的参数，必须更新其他块中该块的调用。否则， CPU 会进入 STOP 状态或者块的功能不能实现。在项目视图中打开程序编辑器，通过菜单命令“选项”→“块调用”，点击“更新所有块调用”，可以更新所有块的时间标签冲突和块不一致的调用。

8.1.3 使用数据块 DB

数据是以变量的形式进行存储的，通过存储地址和数据类型来确保数据的唯一性。数据的存储地址包括 I/O 映像区、位存储器、局部存储区和数据块等，数据块需要占用用户的存储器空间；数据类型有位、字节、字、双字等形式，访问数据块中的数据是通过符号或绝对地址的形式进行的。

根据数据块的使用范围，可将其分为全局数据块（也叫共享数据块）和背景数据块。用户程序中的所有逻辑块都可以访问全局数据块中的信息，而背景数据块只能分配给特定的 FB，仅在所分配的 FB 中使用。

（1）定义数据块

在项目视图左侧项目树中的 PLC 设备项下双击“程序块”下的“添加新块”，打开“添加新块”对话框，如图 8.17 所示。

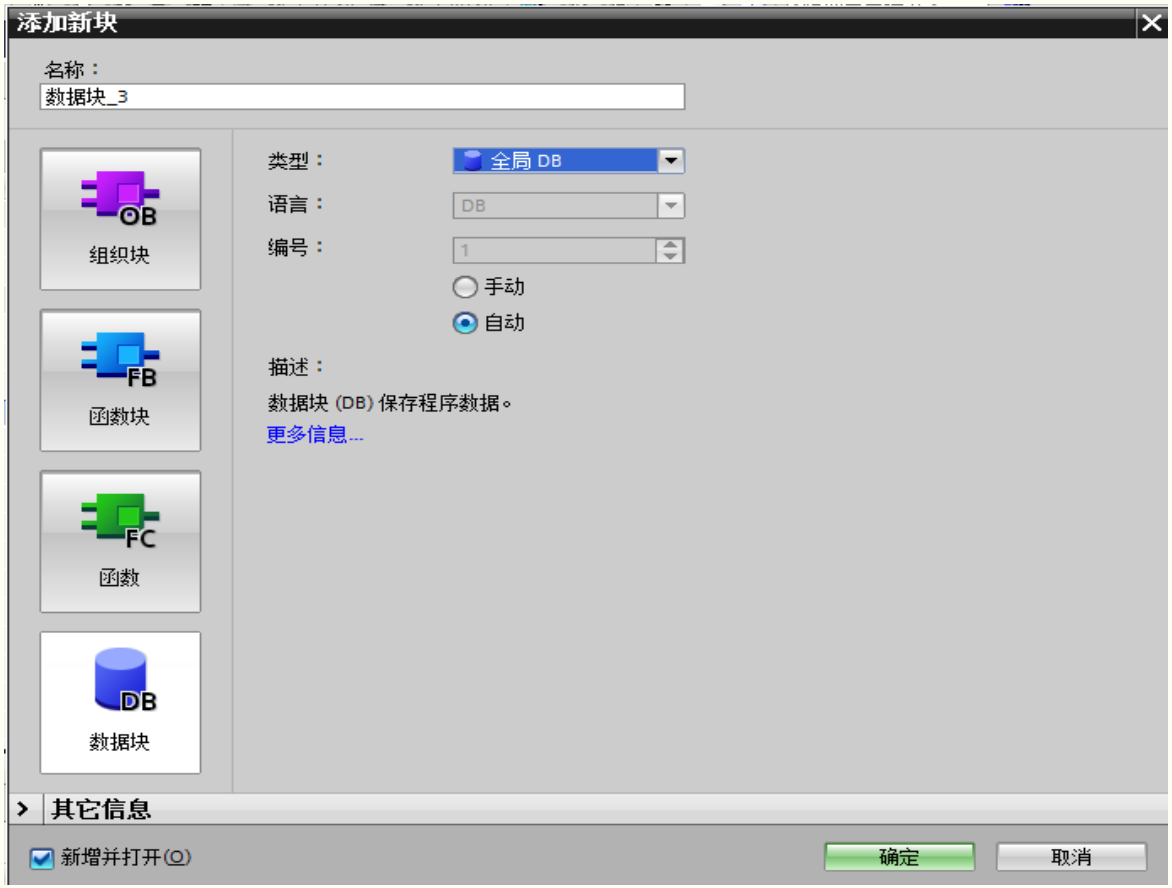


图 8.17“添加新块”对

单击“确定”按钮，则可以打开图 8.18 所示新建数据块，各变量默认情况下是可以在组态中可见的，并可以从“ HMI/OPC UA”中读或写，如 PLC 外部没有接 HMI 设备，也没有通过 OPC 进行数据通信，则可以不勾选这些功能，否则一定要勾选。

数据块_1

	名称	数据类型	起始值	保持	可从 HMI/OPC UA 访问	从 H...	在 HMI ...	设定值	注释
1	Static								
2	sample1	Bool	false		☑	☑	☑		采样值
3	temp1	Int	0		☑	☑	☑		温度
4	volt	Array[1..10] ...			☑	☑	☑		电压数组
5	volt[1]	Int	0		☑	☑	☑		
6	volt[2]	Int	0		☑	☑	☑		
7	volt[3]	Int	0		☑	☑	☑		
8	volt[4]	Int	0		☑	☑	☑		
9	volt[5]	Int	0		☑	☑	☑		
10	volt[6]	Int	0		☑	☑	☑		
11	volt[7]	Int	0		☑	☑	☑		
12	volt[8]	Int	0		☑	☑	☑		
13	volt[9]	Int	0		☑	☑	☑		
14	volt[10]	Int	0		☑	☑	☑		

图 8.18 数据块编辑器

（2）使用全局数据块举例

下面通过一个计算平方根的例子介绍全局数据块的使用。

【例 8-9 】计算 $c=\sqrt{a^2+b^2}$，其中 a 为整数，存储在 MW0 中， b 为整数，存储在 MW2 ， c 为实数，存储在 MD4 中。

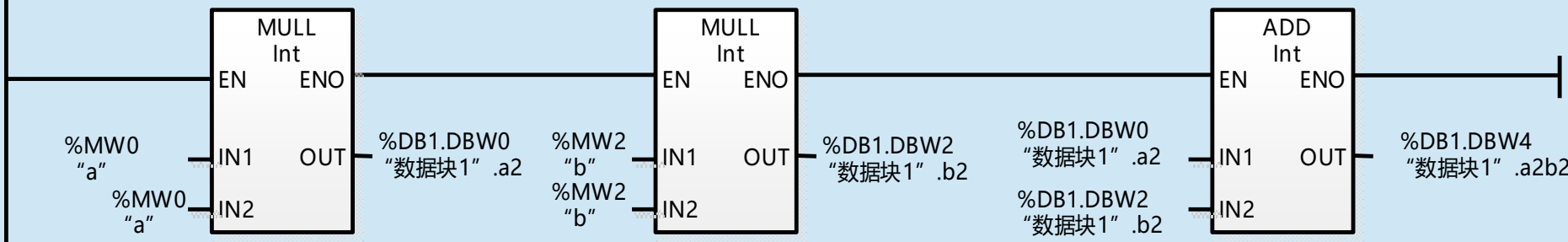
建立全局数据块“数据一块 _1”，选择自动编号，仅符号访问，定义存储中间计算结果的变量，如图 8.19 所示。编写的程序如图 8.20

数据块1

	名称	数据类型	偏移量	起始值	保持	可...	...	在 ...	设定值	注释
	Static									
	a2	Int	0.0	0						a的平方
	b2	Int	2.0	0						b的平方
	a2b2	Int	4.0	0						a的平方与b的平方和
	a2b2r	Real	6.0	0.0						平方和的实数形式

图 8.19 定义数据块中的变量

程序段1： 计算平方根



程序段2： 将整数的平方和转换为实数的平方和，以便使用平方根指令

程序段3： 求平方根

图 8.20 例 8-6 程序

（3）访问数据块

数据块用来存储程序设计过程中所用到的数据信息，用户在程序中需要对数据块中的数据进行读写访问，访问数据块内容的方法有两种：符号寻址和绝对地址寻址。

数据块的寻址格式类似于

DB10.DBB0，DB10.DBW2，DB1.DBD4，DB10.DBX6.3，其中 DB10 为数据块编号，点后面的 DB 表示寻址数据块，最后的数字 0、2、4、6 表示寻址的起始字节地址，B、W、D、X 分别表示寻址宽度为一个字节（Byte）、一个字（Word）、一个双字（Double Word）和一个位（Bit）。字节、字、双字和位

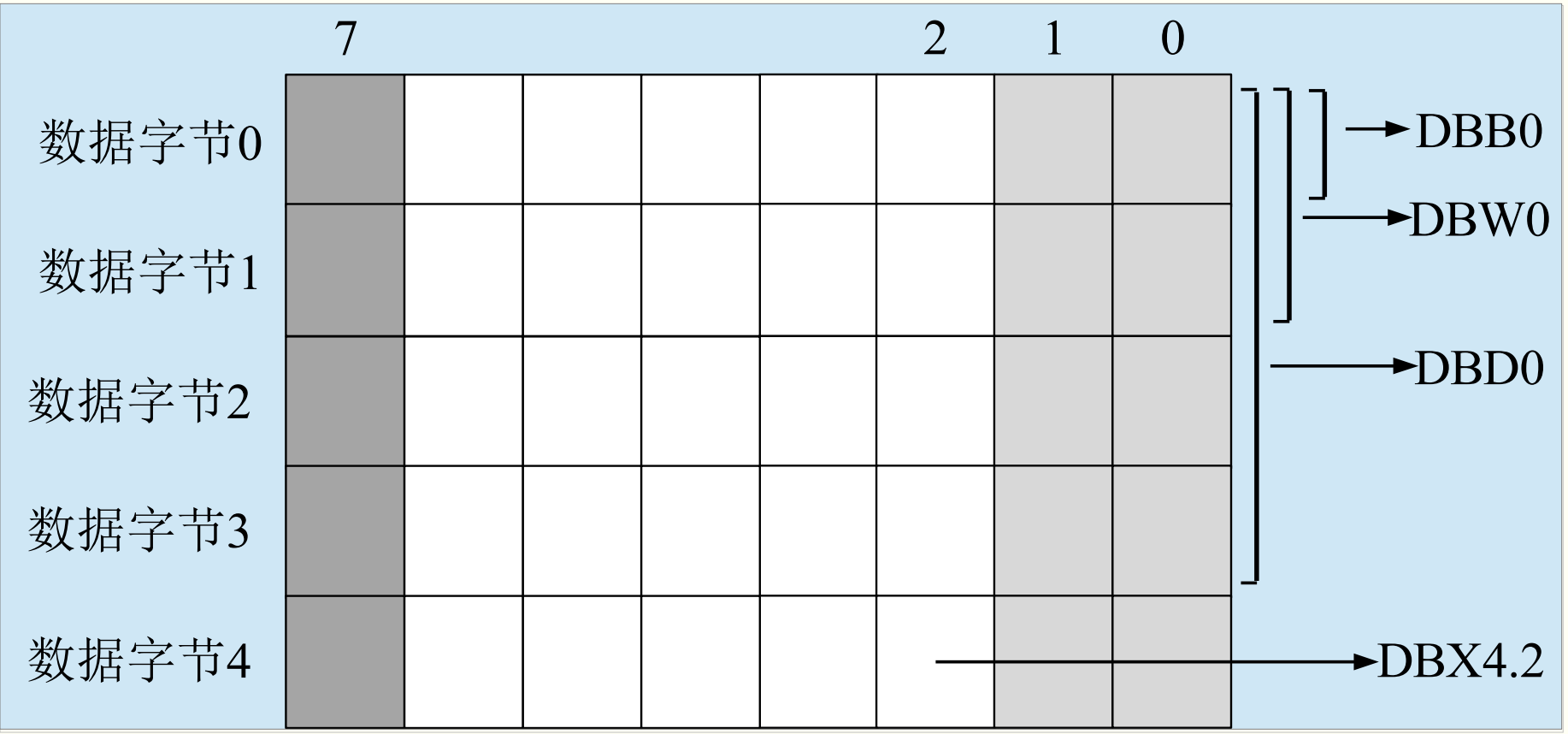


图 8.21 数据单元示意图

由上可知， DB1.DBW200 是由 DBB200 和 DBB201 构成的，在 S7-1200 系列 PLC 中， DBB200 为高位， DBB201 为低位，也就是说，如果 MB200=16#78 ， MB201=16#12 ，则 MW200=16#7812 。下图示意了 MW200 和 MD200 的高低位组成。

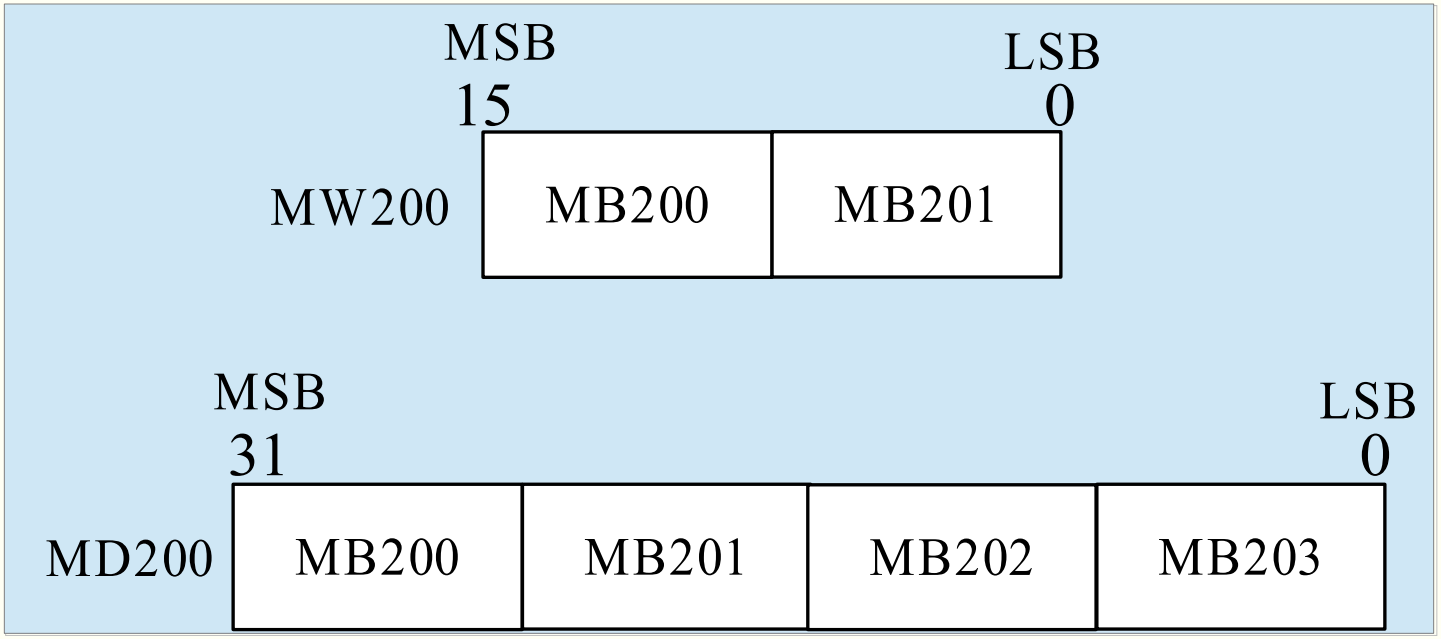


图 8.22　数据块中字及双字的构成

（4）复杂数据类型的使用

复杂数据类型是由其他数据类型组成的数据组，不能将任何常量用做复杂数据类型的实参。也不能将任何绝对地址作为实参传送给复杂数据类型。

1）数组（Array）

Array 数据类型表示的是由固定数目的同一数据类型的元素组成的一个集合。一维数组声明的形式为域名： ARRAY[最小索引 ... 最大索引]OF 数据类型；如一维数组： SampleValue ： ARRAY[1..10] OF REAL ， 数组声明中的索引数据类型为 INT ，其范围为 -32768~32767 ． 这也就反映了数组的最大数目。

blk10

		名称	数据类型	起始值	保持	可从 HMI/...	从 H...	在 HMI ...
1		Static			☐	☐	☐	☐
2		SampleValue	Array[1..10] of Real		☐	☑	☑	☑
3		SampleValue[1]	Real	2343.6	☐	☑	☑	☑
4		SampleValue[2]	Real	35.0	☐	☑	☑	☑
5		SampleValue[3]	Real	2780.8	☐	☑	☑	☑
6		SampleValue[4]	Real	0.0	☐	☑	☑	☑
7		SampleValue[5]	Real	0.0	☐	☑	☑	☑
8		SampleValue[6]	Real	0.0	☐	☑	☑	☑
9		SampleValue[7]	Real	0.0	☐	☑	☑	☑
10		SampleValue[8]	Real	0.0	☐	☑	☑	☑
11		SampleValue[9]	Real	0.0	☐	☑	☑	☑
12		SampleValue[10]	Real	0.0	☐	☑	☑	☑
13		TestValue	Array[-5..5] of Real		☐	☑	☑	☑
14		TestValue[-5]	Real	131.9	☐	☑	☑	☑
15		TestValue[-4]	Real	0.0	☐	☑	☑	☑
16		TestValue[-3]	Real	0.0	☐	☑	☑	☑
17		TestValue[-2]	Real	0.0	☐	☑	☑	☑
18		TestValue[-1]	Real	0.0	☐	☑	☑	☑
19		TestValue[0]	Real	0.0	☐	☑	☑	☑
20		TestValue[1]	Real	0.0	☐	☑	☑	☑
21		TestValue[2]	Real	0.0	☐	☑	☑	☑
22		TestValue[3]	Real	0.0	☐	☑	☑	☑
23		TestValue[4]	Real	0.0	☐	☑	☑	☑
24		TestValue[5]	Real	0.0	☐	☑	☑	☑

图 8.23 新建 Array 类型变量

2）结构（Struct）

Struct 作为一种复杂的数据类型，表示的是一组由若干相同类型或不同类型的数据构成的集合。S7-1200 中结构型变量不支持嵌套。

结构元素也可以在定义时进行初始化赋值，初始化值的数据类型必须与结构元素的数据类型相一致，否则系统会发出警告，当然也可在程序中通过 MOVE 指令进行赋值。

blk11

名称	数据类型	偏移量	起始值	保持	可从 HMI/...	从 H...	在 HMI ...
Static				☐	☐	☐	☐
Record	Struct	0.0		☐	☑	☑	☑
Number	Int	0.0	10870	☐	☑	☑	☑
Gender	Bool	2.0	true	☐	☑	☑	☑
Score	Real	4.0	94.5	☐	☑	☑	☑
Records	Array[1..10] of Struct	8.0		☐	☑	☑	☑
Records[1]	Struct	8.0		☐	☑	☑	☑
speed	Int	8.0	0	☐	☑	☑	☑
temp	Real	10.0	0.0	☐	☑	☑	☑
size	Int	14.0	0	☐	☑	☑	☑
Records[2]	Struct	16.0		☐	☑	☑	☑
speed	Int	16.0	0	☐	☑	☑	☑
temp	Real	18.0	0.0	☐	☑	☑	☑
size	Int	22.0	0	☐	☑	☑	☑
Records[3]	Struct	24.0		☐	☑	☑	☑
Records[4]	Struct	32.0		☐	☑	☑	☑
Records[5]	Struct	40.0		☐	☑	☑	☑
Records[6]	Struct	48.0		☐	☑	☑	☑
Records[7]	Struct	56.0		☐	☑	☑	☑
Records[8]	Struct	64.0		☐	☑	☑	☑
Records[9]	Struct	72.0		☐	☑	☑	☑
Records[10]	Struct	80.0		☐	☑	☑	☑

图 8.24 新建 Struct 类型变量

3）字符串（String）

String 数据类型的变量是用来存储字符串的，每个字符串变量的最大长度可由方括号中的关键字指定，如 String[4] 表示串中的字符数最多为 4，如果省略了最大长度信息，则相应的变量长度默认为 254。

新建一个全局数据块“blk12”，数据块编号为 DB8，不选中数据块的“优化的块访问”单选框，这样就可以允许绝对地址访问，可以查看各变量的地址偏移量。

blk12

	名称	数据类型	偏移量	起始值	保持	可从 HMI/...	从 H...	在 HMI ...
	Static							
	Msg1	String	0.0	'How are you'				
	Msg2	String[10]	256.0	'1234567890'				
	Msg3	String[20]	268.0	''				
	Msg4	String[2]	290.0	''				
	Msg5	String[2]	294.0	''				

图 8.25 新建 String 类型变量

4）长格式日期和时间（DTL）

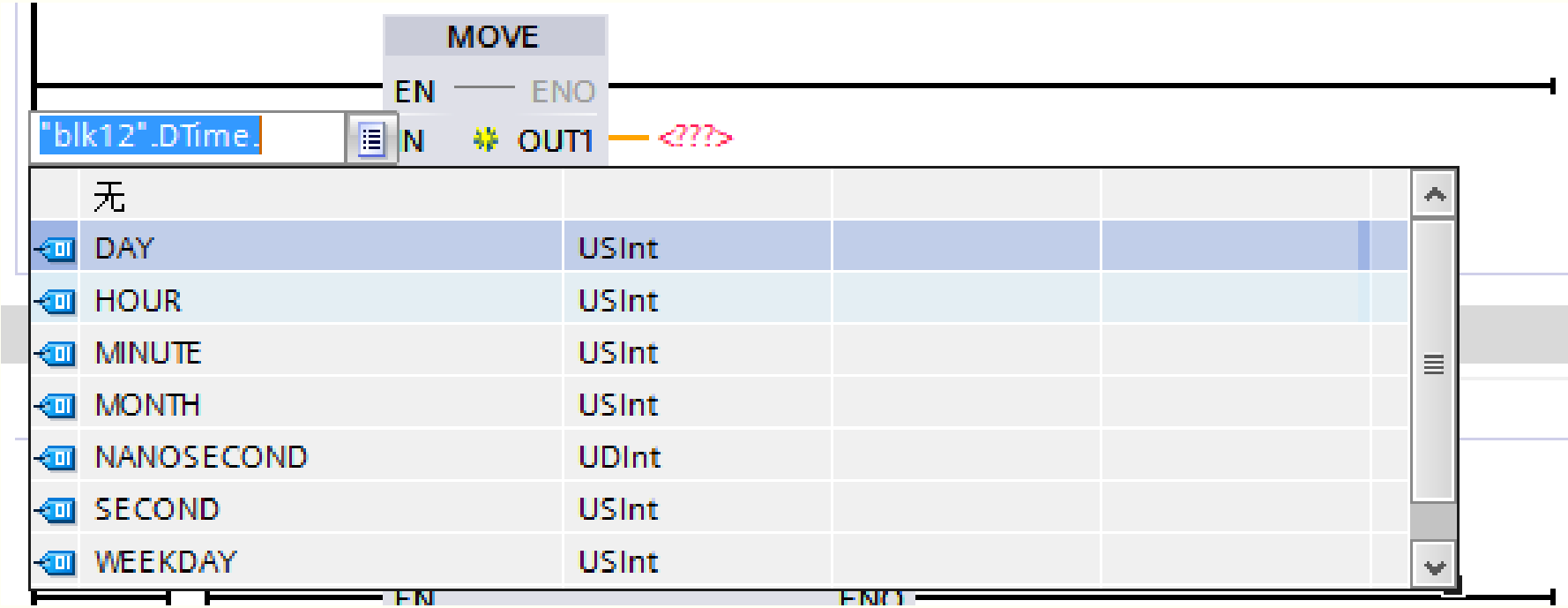
DTL 数据类型表示了一个日期时间值，共 12 个字节。

在全局数据块“blk12”中新建 DTL 型变量 DTime，系统会自动地赋予最小值 DTL#1970-01-01-00:00:00，DTL 型数据长度为 12 个字节，包括年、月、日、星期及时间，每个元素所占存储空间的字节数图 8.26a 所示，其数据格式为：DTL# 月－日－周－小时－分钟－秒－纳秒

blk12

名称	数据类型	起始值
DTime	DTL	DTL#1970-01-01-00:00:00
YEAR	UInt	1970
MONTH	USInt	1
DAY	USInt	1
WEEKDAY	USInt	5
HOUR	USInt	0
MINUTE	USInt	0
SECOND	USInt	0
NANOSECOND	UDInt	0

a



b ）

图 8.26 新建 DTL 类型变量

谢谢聆听