通常，在编写程序时，会多次重复使用某些程序块，这些程序块可以以子程序的方式创建 。

当调用子程序时，程序分支从当前任务跳转到子程序，开始执行其中的指令。执行完毕，程序跳转回原来激活的任务。

SIMOTION 设备的一个或多个程序（如 MCC，LAD/FBD，ST 程序），可以根据需要重复调用子程序。

调用子程序示意图如图 1 所示。

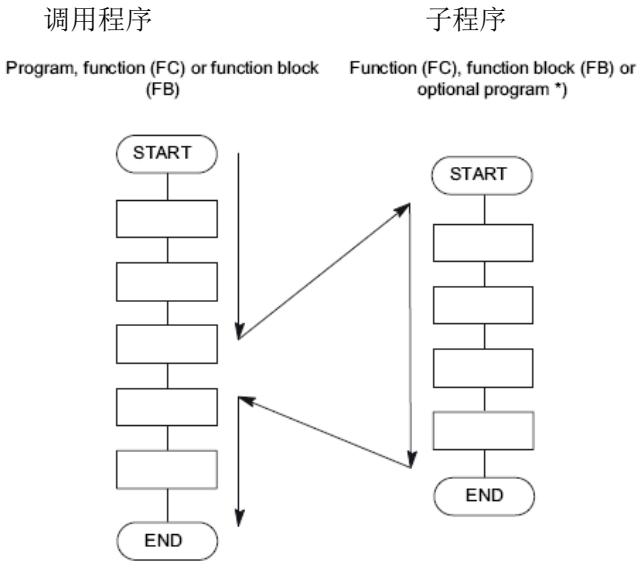


图 1 子程序的执行

FC（Function）是一个无静态数据的子程序，即当 FC 执行后，所有本地变量的值就丢失了，当 FC 下次执行时再进行初始化。

可使用输入参数或输入/输出参数把数据传入 FC，也可输出 FC 的返回值。

通过传递参数或全局变量在子程序和调用程序间传递信息。

传递参数可以是输入，输入/输出或输出参数。这些在变量声明表中定义。

- 输入参数： 变量类型 VAR_INPUT
- 输入/输出参数： 变量类型 VAR_IN_OUT

FB （Function Block）是一个有静态数据的子程序，即当 FB 执行后，所有的本地变量会保持他们原有的值，只有那些明确声明为临时变量的值会丢失。

在使用 FB 之前，必须定义一个背景数据块：即 VAR 或 VAR_GLOBAL，然后输入 FB 的名称作为数据类型。FB 的静态数据存储在此背景数据块中。可以定义多个 FB 背景数据块，每个背景数据块相对独立。

FB 背景数据块的静态数据一直保持，直到该背景数据块再次调用。当 FB 背景数据块的变量类型被再次初始化时，他们也被重新初始化。

可使用输入参数或输入/输出参数把数据传入 FB，通过输入/输出参数或输出参数从 FB 返回数据。通过传递参数或全局变量在子程序和调用程序间传递信息。

传递参数可以是输入，输入/输出或输出参数。这些在变量声明表中定义。

- 输入参数： 变量类型 VAR_INPUT
- 输入/输出参数： 变量类型 VAR_IN_OUT
- 输出参数： 变量类型 VAR_OUT

下面分别介绍在 MCC、LAD/FBD、ST 语言下如何创建和使用 FC、FB。

1 MCC 语言下创建和使用 FC、FB

1.1 使用 FC 子程序

1.1.1 创建一个 FC

首先打开 SCOUT，创建一个新项目。在“PROGRAMS”目录树下双击“Insert MCC unit”，插入一个 MCC 单元。如图 2 所示，Name 栏名称为“MCCunit_1”，点击 OK 确认。

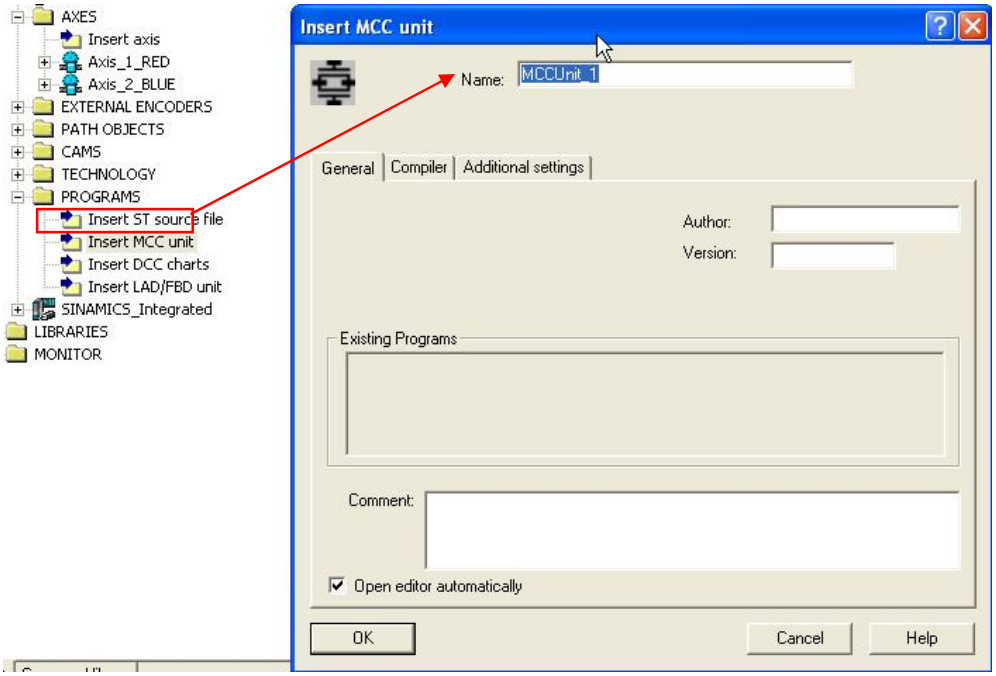


图 2 插入 MCC 单元

下面举例说明：

创建一个计算圆周长的子程序，程序类型为 FC，名称为 “Circumference”。此圆周长计算可作为子程序在任何程序任务中调用。

圆周长计算公式：Circumference = PI*2*radius

可在 FC 变量声明表中定义 Radius（半径）和 π（PI，圆周率）的值。

步骤如下：

鼠标左键双击“Insert MCC chart”，插入一个程序。程序名为“Circumference”。创建类型（Creation type）选择“Function”，返回类型（Return type）选择“REAL”，若选择“<-->”则无返回值。

检查 “Exportable”选项，如果此 FC 程序将要在其他程序源文件中使用（LAD/FBD，MCC 或 ST 源文件），则勾选；如果没有勾选，则此程序只能在本 MCC 单元中使用。

还可以输入作者，版本和评论等。最后点击 OK 确认，如图 3 所示。

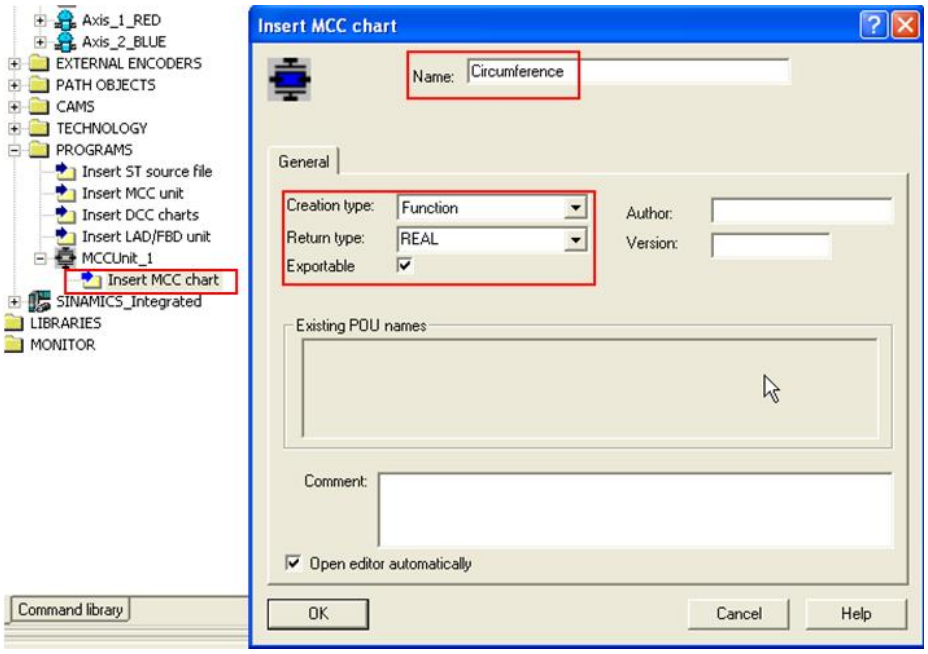


图 3 插入 MCC 程序图

1.1.2 编写 FC 程序

在创建的 FC 程序的变量声明表中定义半径（radius）的变量类型为输入 VAR_INPUT，数据类型为 REAL；圆周率 PI 的变量类型为常数 VAR CONSTANT，数据类型为 REAL，初始值为 3.14159。

Parameters/variables | I/O symbols | Structures | Enumerations

	Name	Variable type	Data type	Array length	Initial value	Comment
1	radius	VAR_INPUT	REAL			
2	PI	VAR CONSTANT	REAL		3.14159	
3						

图 4：FC 变量声明表

编写变量赋值程序并赋给返回值，然后编译保存，FC 程序就编写完毕了。

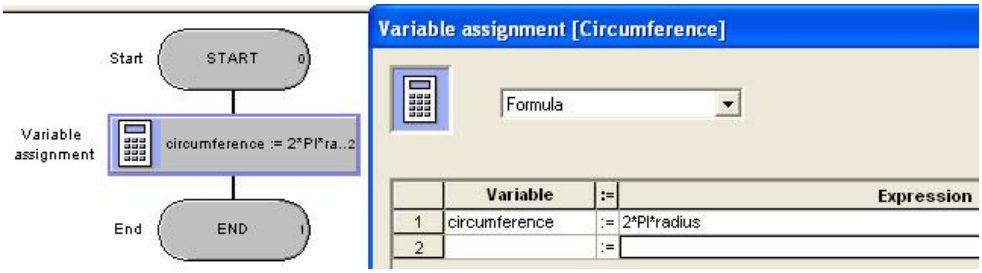


图 5：编写圆周长计算指令

1.1.3 调用 FC 程序

在同一 MCC unit 下生成一个新的程序。同样，鼠标左键双击“Insert MCC chart”，插入一个程序。程序名为“Program_circumference”。创建类型（Creation type）选择“Program”，点击 OK 确认。

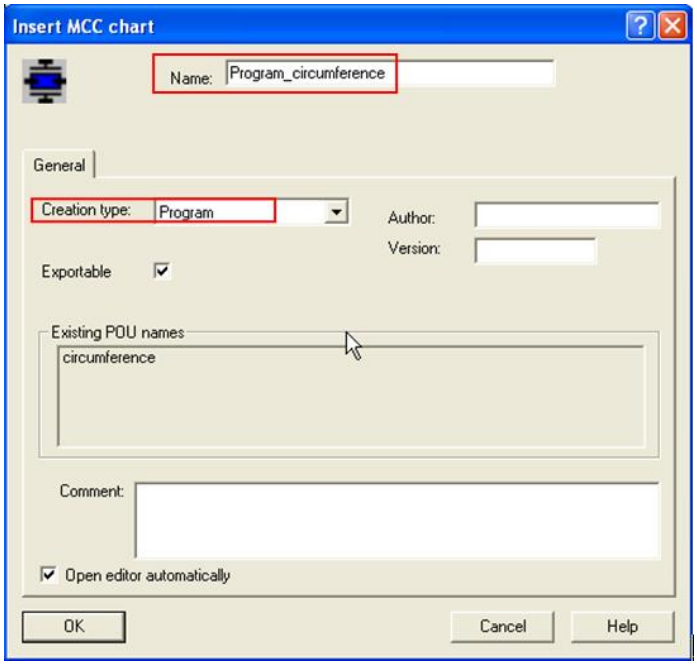


图 6：插入程序

在 MCC unit 或 MCC chart 中，声明下列内容：（本例在 MCC chart 中声明）

- *mycircum* 变量

  FC “Circumference”的返回值赋给此变量。

- *myradius* 变量

  此变量包含半径的数据，赋给 FC“Circumference”的输入参数 Radius。

Parameters/variables | I/O symbols | Structures | Enumerations

	Name	Variable type	Data type	Array length	Initial value	Comment
1	mycircum	VAR	REAL			
2	myradius	VAR	REAL			
3						

图 7：在 MCC chart 中声明变量

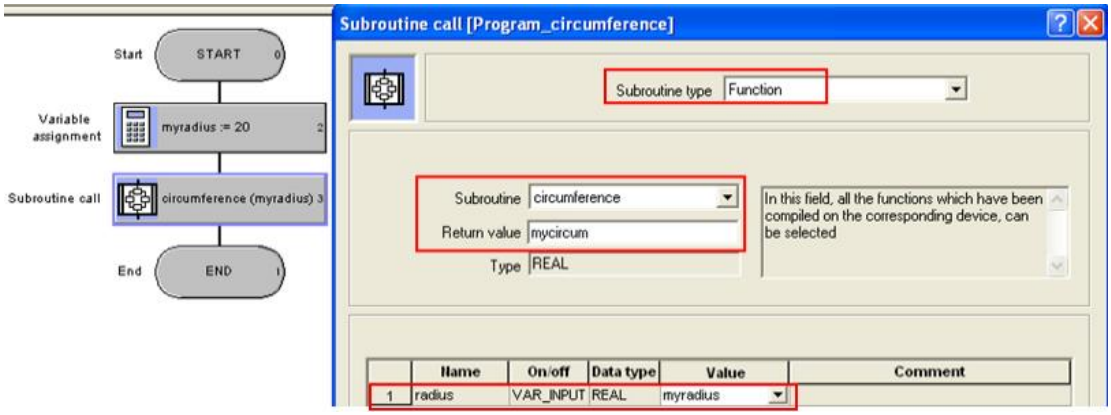


图 8：调用子程序的设定

打开调用子程序设定界面，subroutine type 子程序类型选择“Function”。Subroutine 子程序选择之前创建的“circumference”，Return value 返回值选择“mycircum”。把“myradius”的值赋给 FC 中的变量“radius”。

编译保存 MCC 源文件，这样就完成了 FC 的调用。

需要注意 FC 和 Program 在 MCC unit 中的顺序，FC 必须处在 Program 之上的位置。如果不是，可以点击鼠标右键，选择“Down”或“Up”来调整位置。

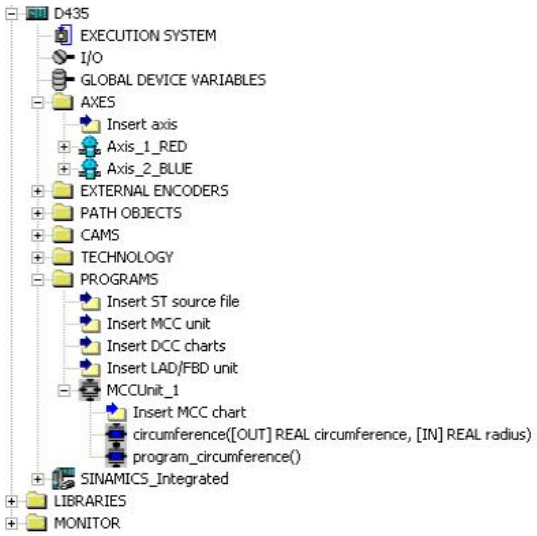


图 9：MCC Charts 的顺序

1.2 使用 FB 子程序

1.2.1 创建一个 FB

首先打开 SCOUT，创建一个新项目。在“PROGRAMS”目录树下双击“Insert MCC unit”，插入一个 MCC 单元。Name 栏名称为“MCCunit_1”，点击 OK 确认。

如计算跟随误差，可创建一个计算跟随误差的子程序，程序类型为 FB，名称为 “FollError”。此跟随误差计算可作为子程序在任何程序任务中调用。

跟随误差计算公式：Difference = Specified position – Actual position

可在 MCC chart 或 MCC unit 中定义所需的输入和输出参数，如设定位置值，实际位置值和偏差。如果需要的话，还可定义其他变量。

步骤如下：

鼠标左键双击“Insert MCC chart”，插入一个程序。程序名为“FollError”。创建类型（Creation type）选择“Function block”。

还可以输入作者，版本和评论等。最后点击 OK 确认。

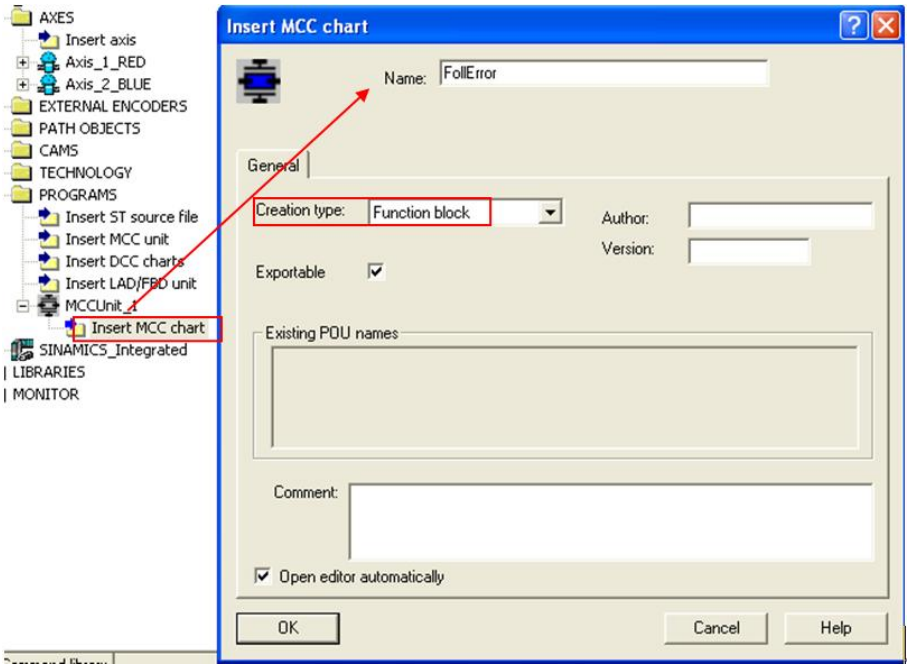


图 10：插入 FB 程序

1.2.2 编写 FB 程序块

在创建的 FB 程序的变量声明表中定义变量，如输入和输出参数，见图 11。

Parameters/variables | I/O symbols | Structures | Enumerations

	Name	Variable type	Data type	Array length	Initial value	Comment
1	Setpoint_position	VAR_INPUT	LREAL			
2	Actual_position	VAR_INPUT	LREAL			
3	Difference	VAR_OUTPUT	LREAL			
4						

图 11：FB 程序的变量声明列表

在 MCC chart 程序编辑区编写程序，使用变量赋值命令，编写计算公式为：Difference = Setpoint_position – Actual_position，然后编译保存，FB 功能块“FollError”编写完成。

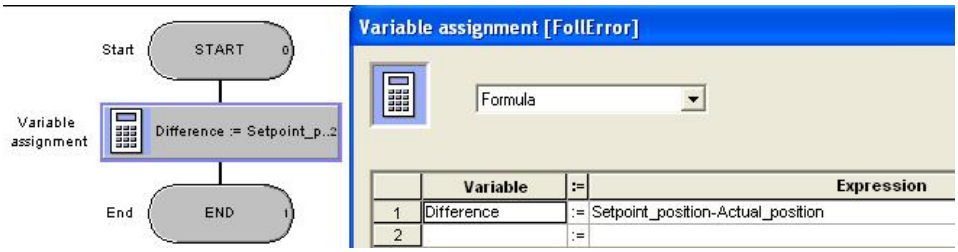


图 12：编写 FB 程序

1.2.3 调用 FB 功能块

基本步骤主要有：

• 创建一个新的 MCC chart，创建类型为“program”

•在 MCC unit 或 MCC chart 中声明 FB 背景数据块

•编写调用程序调用 FB

•执行 FB 背景数据块后，在调用程序中编写变量赋值程序访问输出参数

•编译保存程序

这样就完成了 FB 子程序调用程序的编写。

在使用 FB 之前，必须定义一个背景数据块。FB 的每个背景数据块都相互独立。一旦使用背景数据块结束，静态数据会保留。

可在 MCC unit 或 MCC chart 的变量声明表中定义 FB 的背景数据块。背景数据块声明有效的范围取决于声明的位置。

• 在 MCC unit 的接口部分（interface）的变量声明表

背景数据块如同一个单元变量，对整个 MCC unit 都有效。所有 MCC unit 中的 MCC chart（programs, FC, FB）都能访问此背景数据块。

另外，在 HMI 设备上也可显示此背景数据块；也可引入到其他 MCC unit 中使用。

在接口部分的所有单元变量的大小不能超过 64K 字节。

• 在 MCC unit 的执行部分（implementation）的变量声明表

背景数据块如同一个单元变量，但只对此 MCC unit 有效。所有在此 unit 的 MCC chart（programs, FC, FB）都能访问此背景数据块。

• 在 MCC chart 中的变量声明表

背景数据块如同一个本地变量，只能在被声明的 LAD/FBD program 中使用。

本例在 MCC chart 中声明背景数据块。

在同一 MCC unit 下生成一个新的程序。鼠标左键双击“Insert MCC chart”，插入一个程序。程序名为“Prog_FollError”。创建类型（Creation type）选择“Program”，点击 OK 确认。然后声明 FB 背景数据块和其他变量。

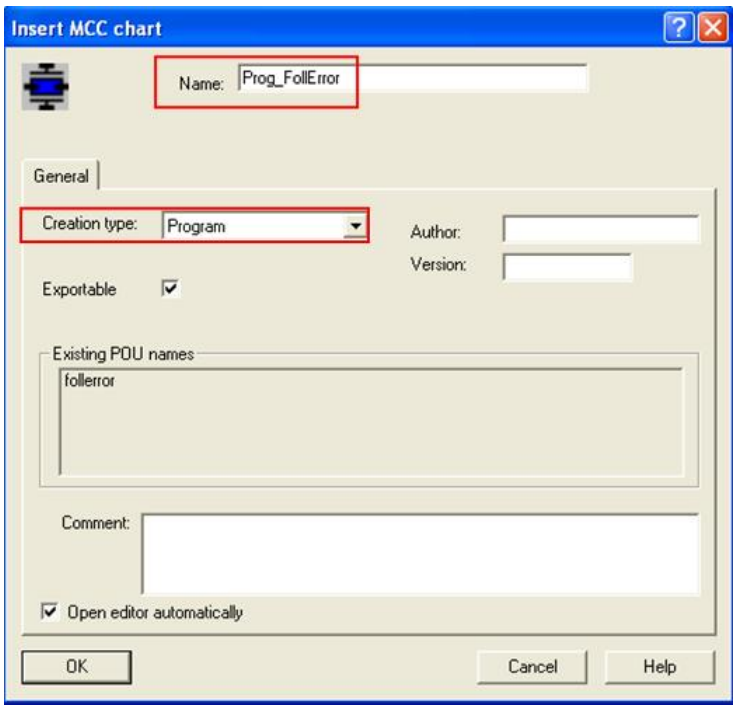


图 13：插入程序

Parameters/variables | I/O symbols | Structures | Enumerations

	Name	Variable type	Data type	Array length	Initial value	Comment
1	myFollErr	VAR	FOLLERR			
2	Result	VAR	LREAL			
3	Result_2	VAR	LREAL			
4						

图 14：声明 FB 背景数据块和其他变量

在 MCC chart 中插入“Subroutine call”调用子程序指令。打开调用子程序设定界面，subroutine type 子程序类型选择“Function block”。Subroutine 子程序选择之前创建的“follerror”，Instance 背景数

据块选择“myFollErr”。把轴的设定位置值和实际位置值分别赋给 FB 中的变量“setpoint_position”和“actual_position”。“Result”参数就是经过 FB 块计算后输出的值。点击 OK 确认。

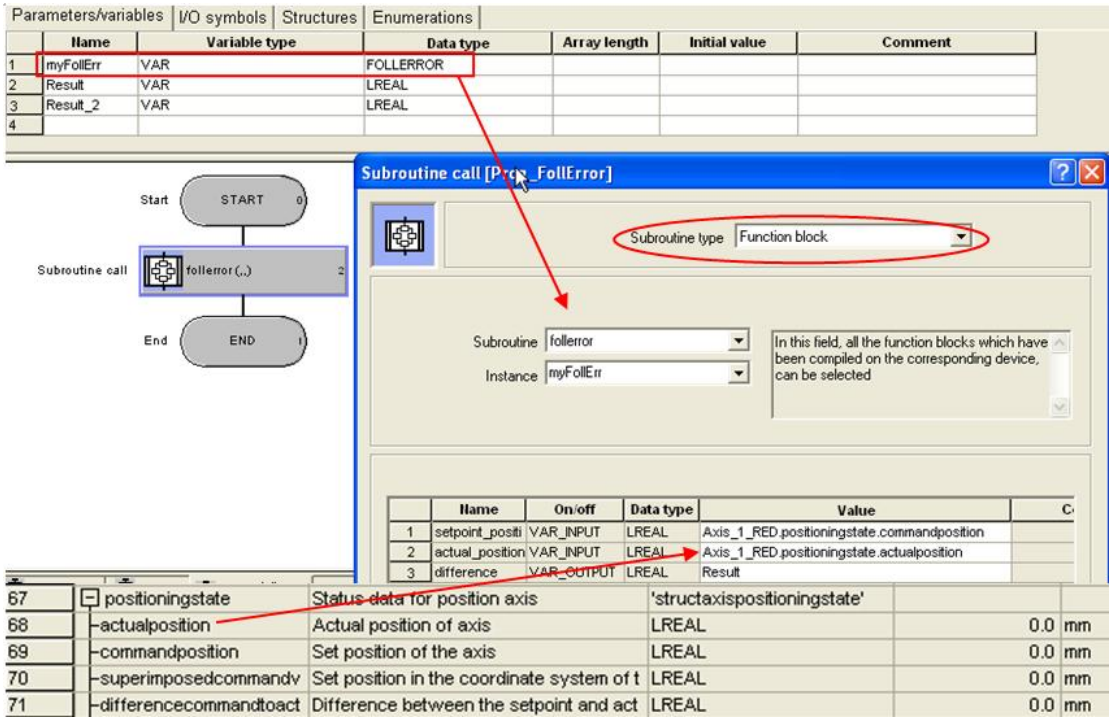


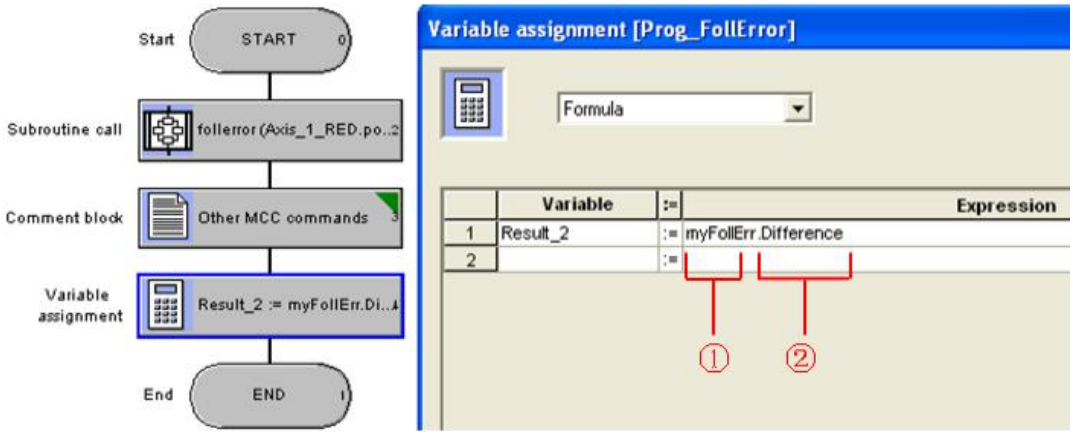
图 15：声明 FB 背景数据块和其他变量

编译保存 MCC 源文件，这样就完成了 FB 的调用。

需要注意 FB 和 Program 在 MCC unit 中的顺序，FB 必须处在 Program 之上的位置。如果不是，可以点击鼠标右键，选择“Down”或“Up”来调整位置。

在 FB 功能块执行后，背景数据块中的静态数据（包括输出参数）仍然保留。可以在调用程序中访问输出参数。如果把 FB 背景数据块定义成 VAR_GLOBAL，还可以在其他 MCC charts 程序中访问输出参数。

在 MCC chart 中插入“Variable assignment”指令，编写指令 Result_2 := myFollErr.Difference，点击 OK 确认。这样就把输出参数 myFollErr.Difference 的值赋给了 Result_2。



① FB 背景数据块名称　　② 输出参数

图 16：程序的变量分配

2 LAD/FBD 语言下创建和使用 FC、FB

2.1 使用 FC 子程序

2.1.1 创建一个 FC

首先打开 SCOUT，创建一个新项目。在“PROGRAMS”目录树下双击“Insert LAD/FBD unit”，插入一个 LAD/FBD 单元。Name 栏名称为“LFunit_1”，点击 OK 确认。

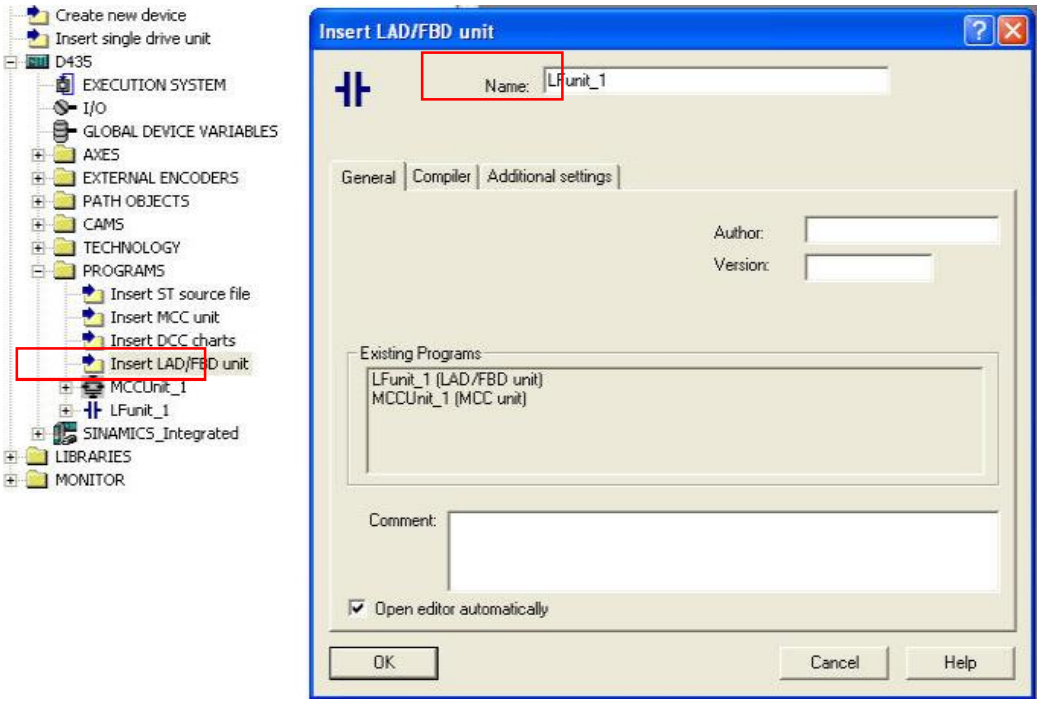


图 17：插入 LAD/FBD 单元

下面举例说明：

创建一个计算圆周长的子程序，程序类型为 FC，名称为 “Circumference”。此圆周长计算可作为子程序在任何程序任务中调用。

圆周长计算公式：Circumference = PI*2*radius

可在 FC 变量声明表中定义 Radius（半径）和 π（PI，圆周率）的值。

步骤如下：

鼠标左键双击“Insert LAD/FBD program”，插入一个程序。程序名为“Circumference”。创建类型（Creation type）选择“Function”，返回类型（Return type）选择“REAL”，若选择“<-->”则无返回值。

检查 “Exportable”选项，如果此 FC 程序将要在其他程序源文件中使用（LAD/FBD，MCC 或 ST 源文件），则勾选；如果没有勾选，则此程序只能在本 LAD/FBD 单元中使用。

还可以输入作者，版本和评论等。最后点击 OK 确认。

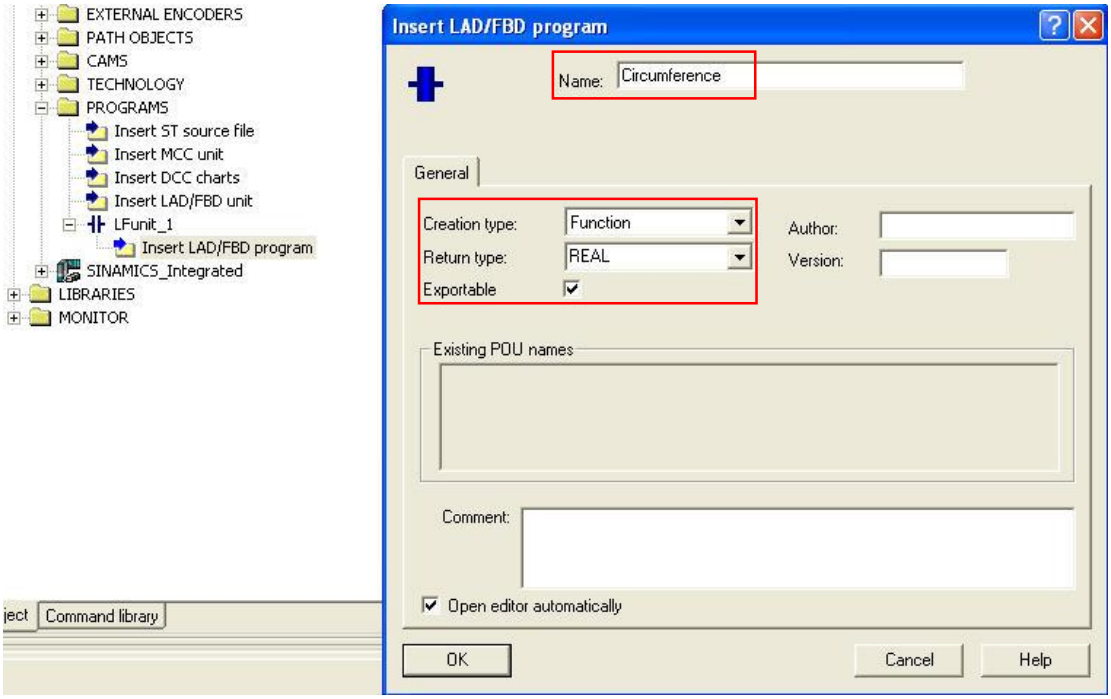


图 18：插入 LAD/FBD 程序

2.1.2 编写 FC 程序

在创建的 FC 程序的变量声明表中定义半径（radius）的变量类型为输入 VAR_INPUT，数据类型为 REAL；圆周率 PI 的变量类型为常数 VAR CONSTANT，数据类型为 REAL，初始值为 3.14159；直径 diameter 的变量类型为 VAR，数据类型为 REAL。

Parameters/variables | I/O symbols | Structures | Enumerations

	Name	Variable type	Data type	Array length	Initial value	Comment
1	radius	VAR_INPUT	REAL			
2	PI	VAR CONSTANT	REAL		3.14159	
3	diameter	VAR	REAL			
4						

图 19：FC 变量声明表

在程序编辑区域右键点击“Insert network”，或者左键点击按钮“  ”插入一个网络。

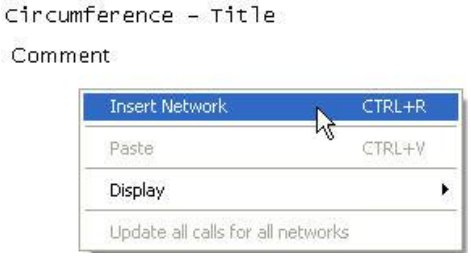


图 20：插入网络

在命令库中拖出两个乘法器命令到程序编辑区，如图 21 所示。

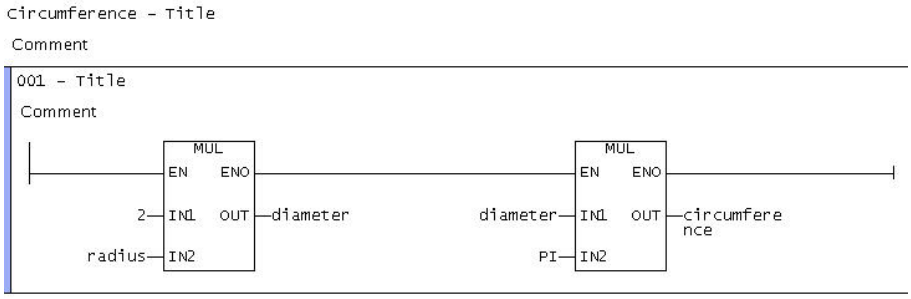


图 21：FC 程序

编写周长计算程序，把这两个乘法器的输出赋值给返回值，然后编译保存。

2.1.3 调用 FC 程序

在同一 LAD/FBD unit 下生成一个新的程序。同样，鼠标左键双击“Insert LAD/FBD program”，插入一个程序。程序名为“Program_circumference”。创建类型（Creation type）选择“Program”，点击 OK 确认。

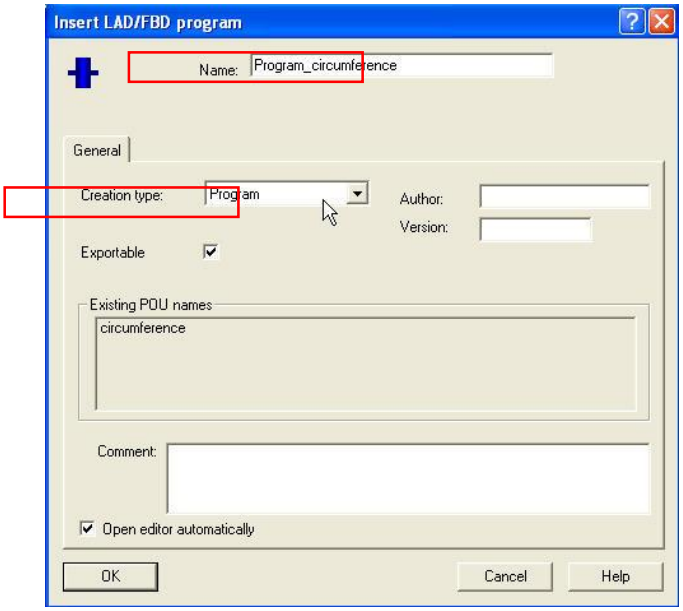


图 22：插入程序

插入一个网络，然后把 FC “Circumference”拖入该网络。

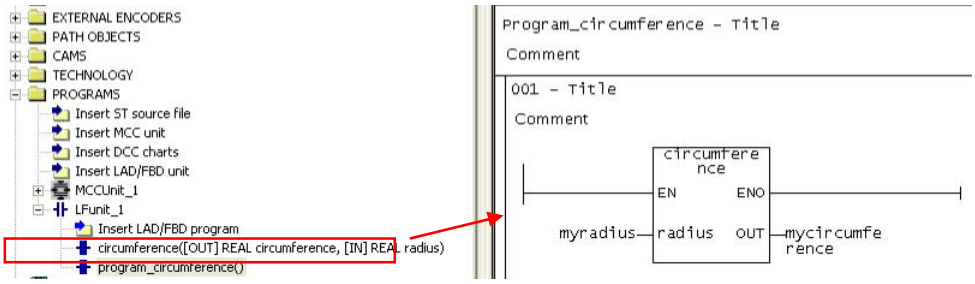


图 23：调用程序

在 LAD/FBD unit 或 LAD/FBD program 中声明下列内容：

• *mycircumference* 变量

FC “Circumference”的返回值赋给此变量。

• *myradius* 变量

此变量包含半径，赋给 FC“Circumference”的输入参数 Radius。

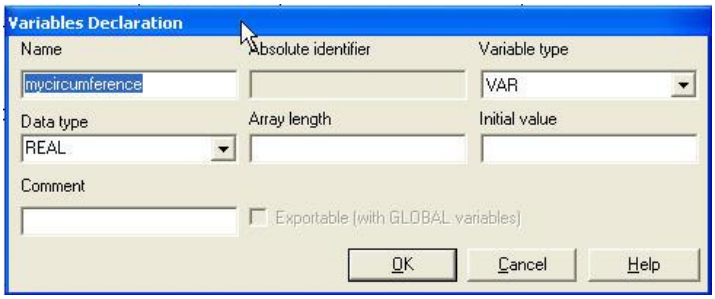


图 24：声明变量

Parameters/variables | I/O symbols | Structures | Enumerations

	Name	Variable type	Data type	Array length	Initial value	Comment
1	myradius	VAR	REAL			
2	mycircumferen	VAR	REAL			
3						

图 25：声明变量及数据类型

双击该程序可看到子程序调用的返回值等信息。

需要注意 FC 和 Program 在 LAD/FBD unit 中的顺序，FC 必须处在 Program 之上的位置。如果不是，可以点击鼠标右键，选择“Down”或“Up”来调整位置。

2.2 使用 FB 子程序

2.2.1 创建一个 FB

首先打开 SCOUT，创建一个新项目。在“ROGRAMS”目录树下双击“Insert LAD/FBD unit”，插入一个 LAD/FBD 单元。如图 26 所示，Name 栏名称为“LFunit_1”，点击 OK 确认。

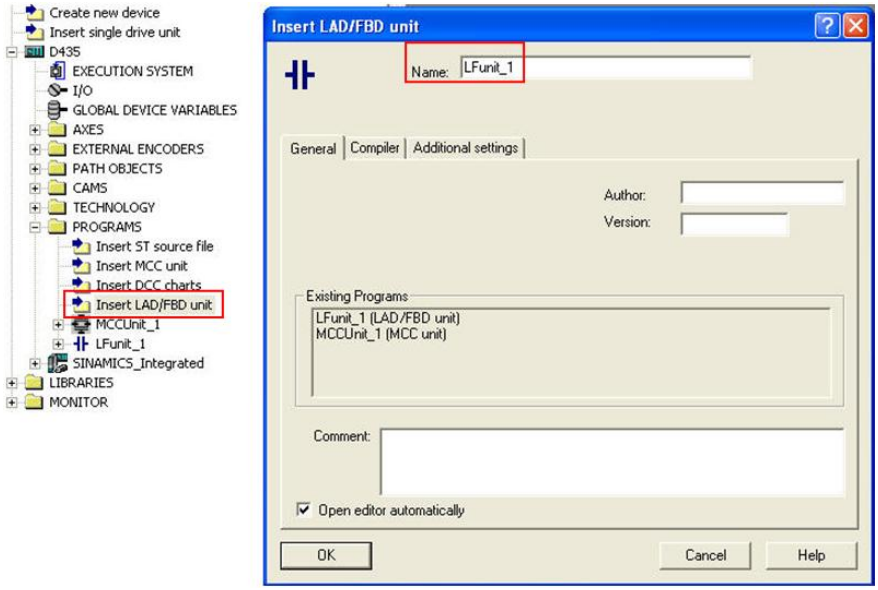


图 26：插入 LAD/FBD 单元

下面举例说明：

如计算跟随误差，可创建一个计算跟随误差的子程序，程序类型为 FB，名称为 “FollError”。此跟随误差计算可作为子程序在任何程序任务中调用。

跟随误差计算公式：Difference = Specified position – Actual position

可在 LAD/FBD 程序（FB）或 LAD/FBD unit 中定义所需的输入和输出参数，如设定位置值，实际位置值和偏差。如果需要的话，还可定义其他变量。

步骤如下：

鼠标左键双击“Insert LAD/FBD program”，插入一个程序。程序名为“FollError”。创建类型（Creation type）选择“Function block”。

还可以输入作者，版本和评论等。最后点击 OK 确认。

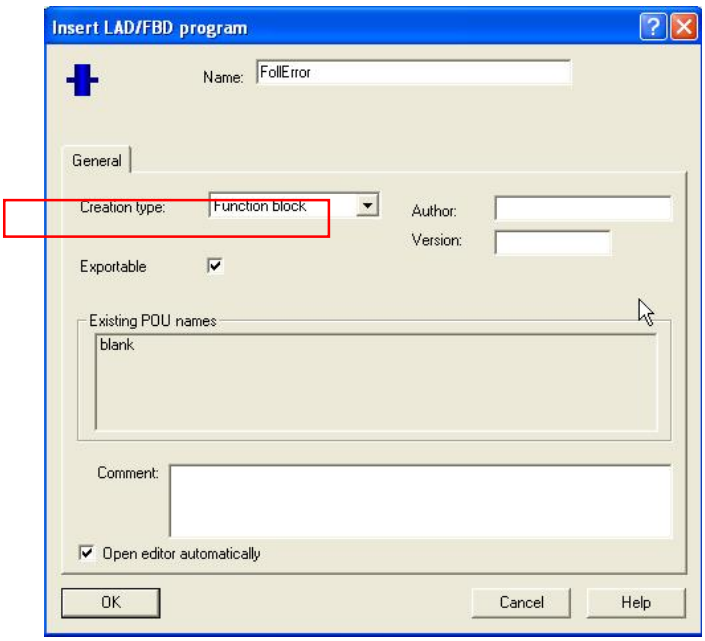


图 27：插入 FB 程序

2.2.2 编写 FB 程序块

在创建的 FB 程序的变量声明表中定义变量，如输入和输出参数，见图 28。

Parameters/variables | I/O symbols | Structures | Enumerations

	Name	Variable type	Data type	Array length	Initial value	Comment
1	Setpoint_position	VAR_INPUT	LREAL			
2	Actual_position	VAR_INPUT	LREAL			
3	Difference	VAR_OUTPUT	LREAL			
4						

图 28：FB 程序的变量声明列表

在程序编辑区域右键点击“Insert network”，或者左键点击按钮“  ”插入一个网络。

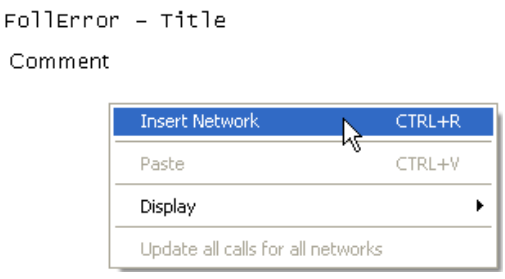


图 29：插入网络

在命令库中拖出一个减法器命令到程序编辑区，如图 30 所示。然后编译保存。

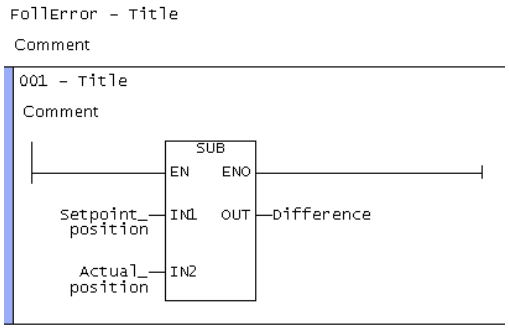


图 30：编写 FB 程序

2.2.3 调用 FB 程序块

在使用 FB 之前，必须定义一个背景数据块。FB 的每个背景数据块都相互独立。一旦使用背景数据块结束，静态数据会保留。

可在 LAD/FBD unit 或 LAD/FBD program 的变量声明表中定义 FB 的背景数据块。背景数据块声明的有效范围取决于声明的位置。

• 在 LAD/FBD unit 的接口部分（interface）的变量声明表

背景数据块如同一个单元变量，对整个 LAD/FBD unit 都有效。所有 LAD/FBD unit 的 LAD/FBD 程序（programs, FC, FB）都能访问此背景数据块。

另外，在 HMI 设备上也可显示此背景数据块；也可引入到其他 LAD/FBD unit 中使用。

在接口部分的所有单元变量的大小不能超过 64K 字节。

• 在 LAD/FBD unit 的执行部分（implementation）的变量声明表

背景数据块如同一个单元变量，但只对此 LAD/FBD unit 有效。所有在此 unit 的 LAD/FBD 程序（programs, FC, FB）都能访问此背景数据块。

• 在 LAD/FBD program 的变量声明表

背景数据块如同一个本地变量，只能在被声明的 LAD/FBD program 中使用。

本例在 LAD/FBD unit 的接口部分声明背景数据块。

注意，如果在此 LAD/FBD unit 下生成一个 FOLLERROR 类型的全局背景数据块就会编译出错。正确做法是必须再创建一个新的 LAD/FBD unit：LFunit_2，与 LFunit_1 建立连接，之后在 LFunit_2 的接口部分（interface）中声明背景数据块，名称为“myFollError”，创建的 FB 作为数据类型。还可声明其他变量，自定义名称、变量类型和数据类型。如图 32 所示。

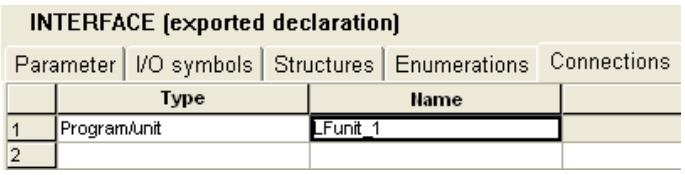


图 31：建立连接

INTERFACE (exported declaration)

Parameter | I/O symbols | Structures | Enumerations | Connections

	Name	Variable type	Data type	Array length	Initial value	Comment
1	myFollError	VAR_GLOBAL	FOLLERROR (* LFUNIT_1 *)			
2	result	VAR_GLOBAL	LREAL			
3	result_2	VAR_GLOBAL	LREAL			
4						

图 32：声明 FB 背景数据块和其他变量

在 LFunit_2 中可看到相应的变量，能够正常使用该背景数据块了。

D435.LFunit_2:

	Name	Data type	Initial value	
1	myfollerror	'follerror'		
2	setpoint_position	LREAL	0.0	DEC
3	actual_position	LREAL	0.0	DEC
4	difference	LREAL	0.0	DEC
5	result	LREAL	0.0	DEC
6	result_2	LREAL	0.0	DEC

图 33：LFunit_2 中的变量

在 LFunit_2 中创建一个程序。鼠标左键双击“Insert LAD/FBD program”，插入一个程序。程序名为“Program_FollError”。创建类型（Creation type）选择“Program”，点击 OK 确认。

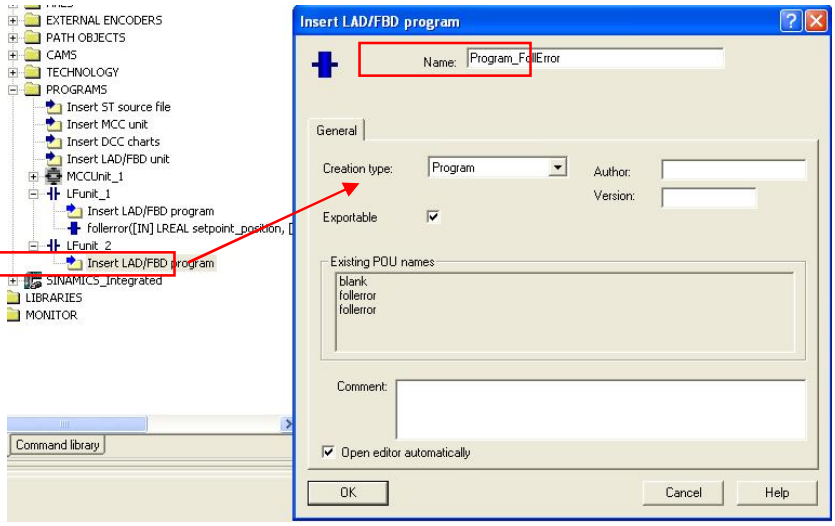


图 34：LFunit_2 中创建程序

插入一个网络，然后把 LFunit_1 中的 FB “FollError”拖入该网络。并选择“myfollerror”作为背景数据块。

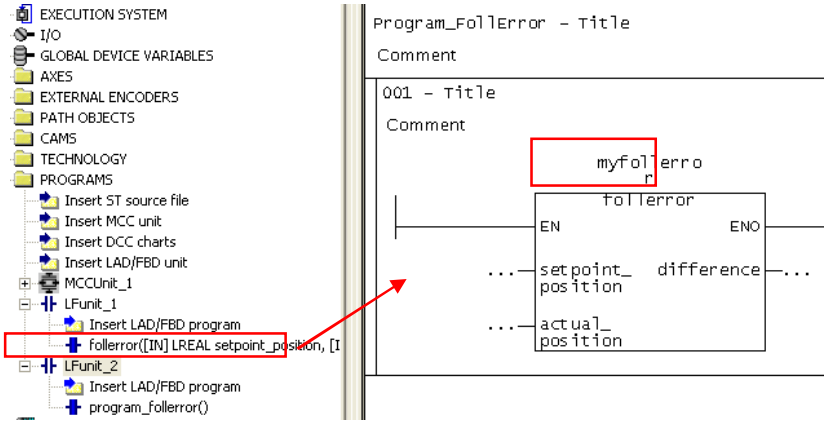


图 35：把 FB 拖入 Program 中

鼠标右键点击编辑区域，选择“Display>All Box Parameters”，FB 块的所有参数就显示出来了。

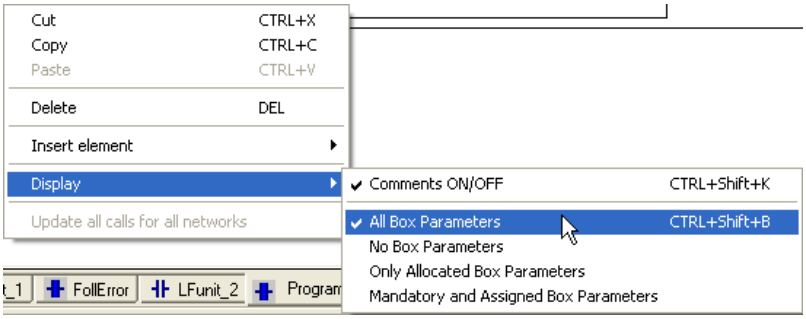


图 36：选择显示所有参数

双击调用的子程序，在弹出“Enter Call Parameter”画面中分配参数，比如选择变量“result”作为“difference”的输出参数。

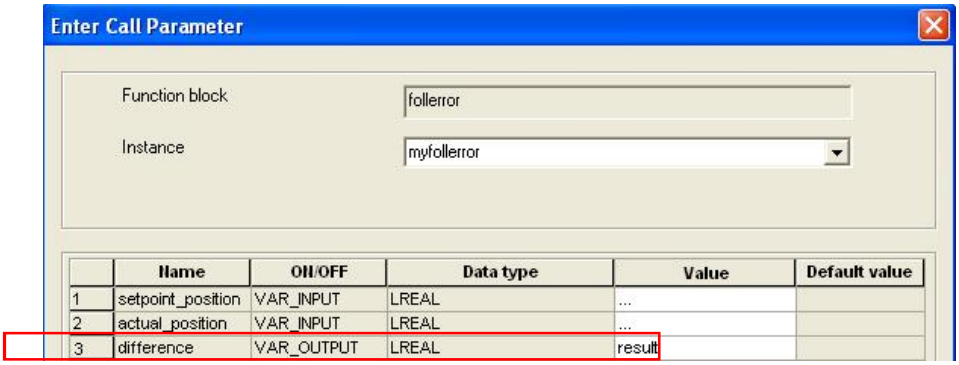


图 37：选择全部参数

也可直接将变量分配到输入/输出管脚上：

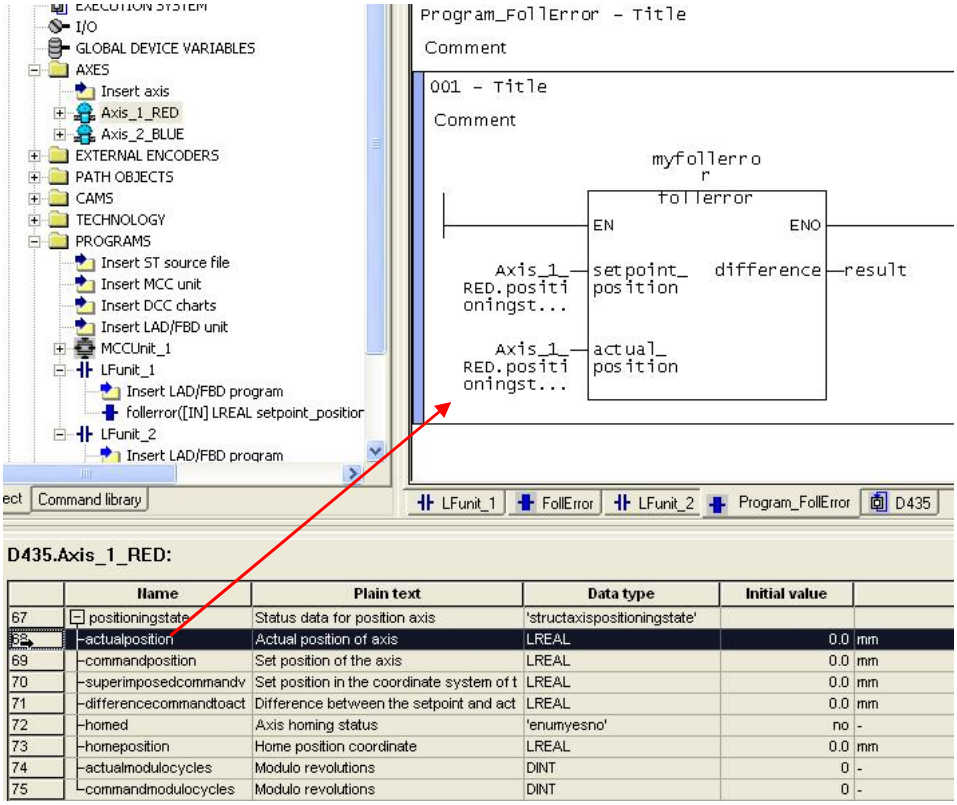
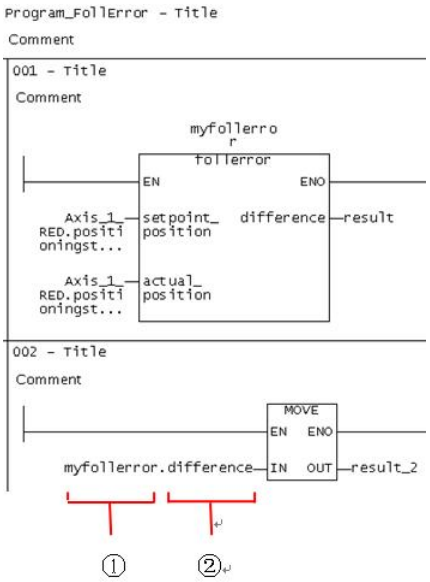


图 38：把变量分配给输入/输出参数

把轴的设定位置和实际位置的变量分配给输入/输出参数。 “result”参数就是经过 FB 块计算后输出的值。

在 FB 块执行后，背景数据块中的静态数据仍然保留。可以在调用程序中访问输出参数。如果把 FB 背景数据块定义成 VAR_GLOBAL，还可以在其他 LAD/FBD 程序中访问输出参数。

举例如下，在程序的 network2 中插入 MOVE 指令，并编辑输入输出参数，见图 39。这样就把输出参数 myFollError.Difference 的值赋给了 Result_2。



① FB 背景数据块名称　　② 输出参数

图 39：程序的变量分配

3 ST 语言下创建和使用 FC、FB

3.1 定义功能 FC（Function）

在调用 FC 的源文件段（program，FB，FC）前，用户可以在执行区域（implementation section）的声明部分（declaration part）定义一个 FC。

使用下列语法：

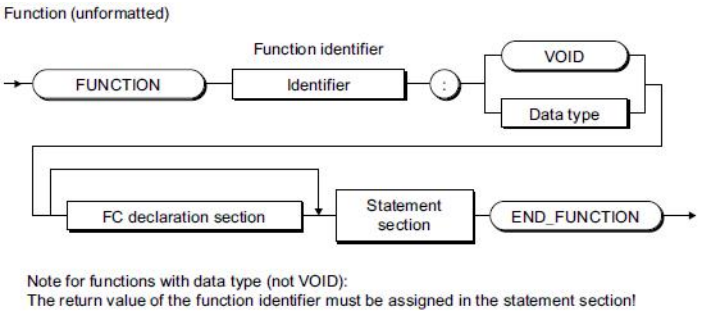


图 40：语法：定义 FC

在关键词 FUNCTION 后需输入一个标识符，作为 FC 的名称和返回值的数据类型。如果 FC 无返回值，则输入 VOID 作为数据类型。

然后输入：

- 其他的声明区域

• 程序段

•关键词 END_FUNCTION

3.2 定义功能块 FB（Function Block）

在调用 FB 的源文件段（program，FB，FC）前，用户可以在执行区域（implementation section）的声明部分（declaration part）定义一个 FB。

使用下列语法：

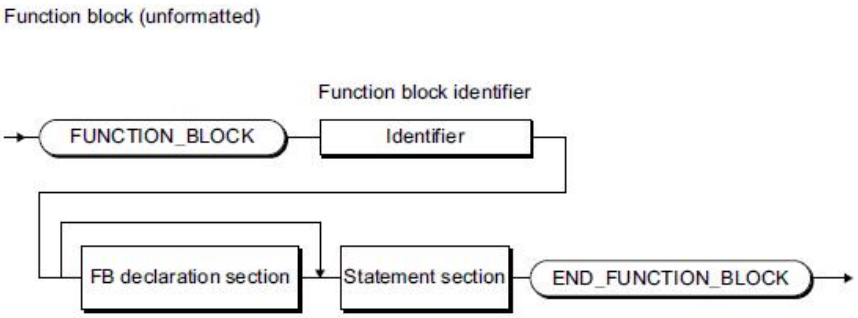


图 41：语法：定义 FB

在关键词 FUNCTION_BLOCK 后需输入一个标识符作为 FB 的名称。

然后输入：

• 其他的声明区域

• 程序段

•关键词 END_FUNCTION_BLOCK

3.3 FB 和 FC 的声明区域（Declaration section）和语句区域（Statement section）

声明区域可被细分为多个不同的声明块，每个声明块都有独立成对的关键词来标识。每个块中包含一个声明列表，比如常数，本地变量和参数。每种类型的块只能出现一次，但顺序可任意调换。

下列选项可用做 FC 和 FB 的声明区域。

Data	Syntax	FB	FC
Constant	VAR CONSTANT *Declaration list* END_VAR	X	X
Input parameters	VAR_INPUT *Declaration list* END_VAR	X	X
In/out parameter	VAR_IN_OUT *Declaration list* END_VAR	X	X
Output parameters	VAR_OUTPUT *Declaration list* END_VAR	X	–
Local variable (for FC and FB)	VAR *Declaration list* END_VAR	X (static)	X (temporary)
Local variable (for FB)	VAR_TEMP *Declaration list* END_VAR	X (temporary)	-
Declaration list: The list of identifiers of the type to be declared			

表 1：FB 和 FC 的声明块

参数是本地数据，而且是 FB 或 FC 的形式参数。当调用 FB 或 FC 时，由实参代替形参，从而在已调用和正在调用的源文件之间提供一个交换信息的手段 。

- 形式输入参数接收实际输入值 （数据流向内）
- 形式输出参数（只适用于 FB）用来传送输出值（数据流向外）
- 形式输入/输出参数作为输入和输出参数

下图显示了 FB 或 FC 声明参数的语法。

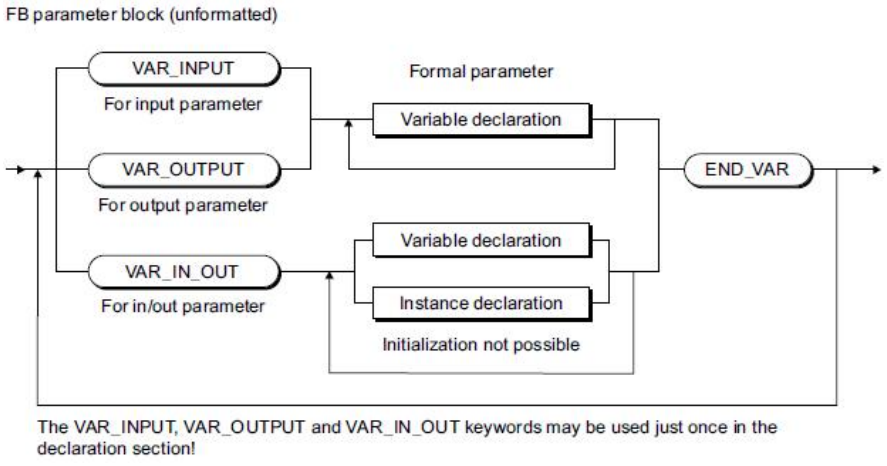


图 42：语法：FB 参数块

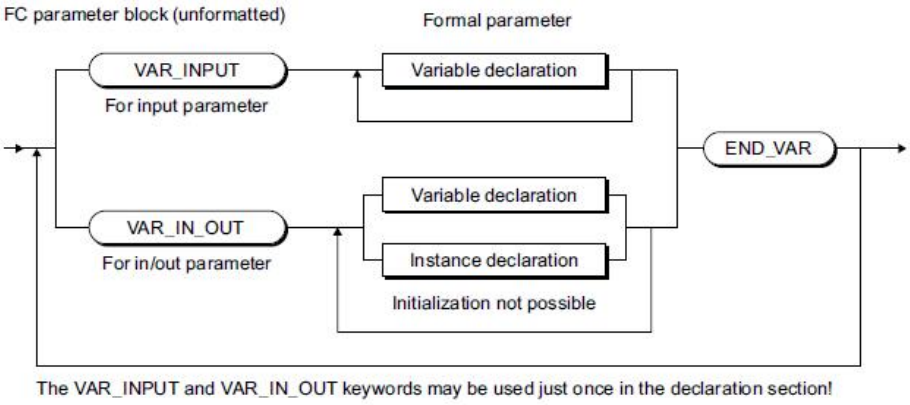


图 43：语法：FC 参数块

用户可以在 FB 或 FC 中和其他变量一样使用声明过的参数，但是需注意：不能把值赋给输入参数。

在 FB 或 FC 程序的外部接口，用户可以：

- 通过结构变量，访问 FB 的输入和输出参数

当且仅当“Permit language extensions”编辑器被激活时可访问输入参数。

- 通过使用表达式访问 FC 的返回值，并且可把此值赋给变量。

当调用 FB 或 FC 时，就会执行 FB 或 FC 中包含的语句。编写 FB 或 FC 的语法规则与编写其他程序并无区别。

3.4 FB 和 FC 的调用

3.4.1 参数传递规则

当调用 FB 或 FC 时，需要把传递的参数列入参数清单中，多个参数写在圆括号内，通过逗号分隔开。如图 44 所示。

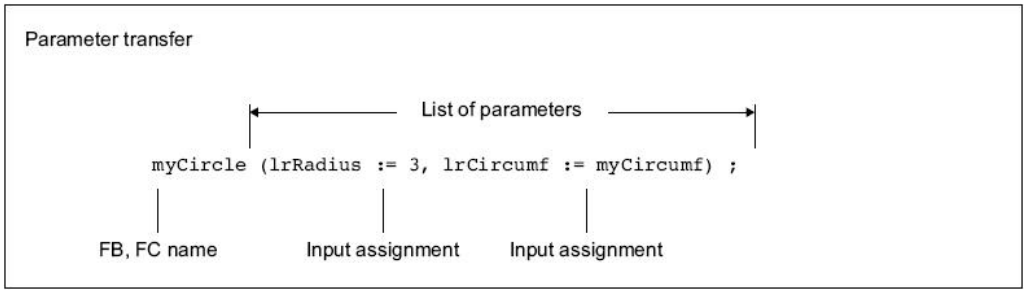


图 44：调用程序的参数传递规则

通常指定输入和输入/输出参数进行赋值。这样，用户可把实参的值赋给形参。形参在被调用块的声明部分中定义。

通过使用=>操作符对输出参数赋值。这样，用户可把实参的值赋给输出形参。形参在被调用块的声明部分中定义。

3.4.2 参数传递给输入参数

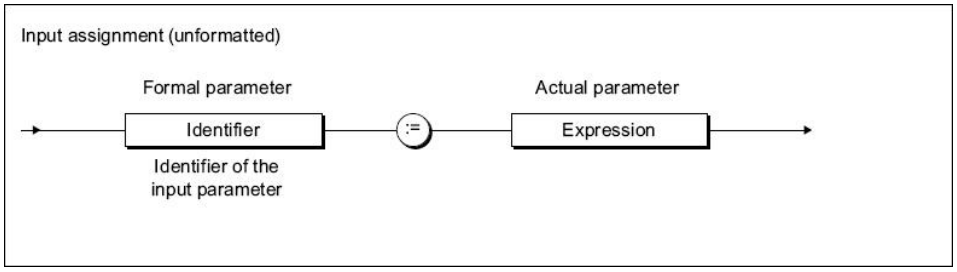


图 45：语法：输入赋值

用户通过分配输入的方式，传递数据（实参）到 FB 或 FC 的输入形参，可以以表达式的方式指定实参。在 FB 或 FC 内，可以使用输入形参，但是不能修改他们的值。

在 FB 中，可选择对实参赋值。如果输入赋值未指定，则保持上次调用时的值，因为 FB 带有背景数据块。

在 FC 中，当形参的声明指定了初始化表达式时，可选择对实参赋值。也可参考下文的示例。

3.4.3 参数传递给输入/输出参数

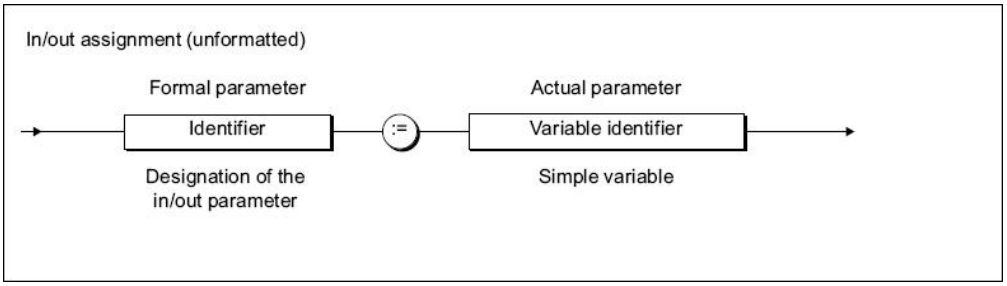


图 46：语法：输入/输出赋值

用户通过输入/输出赋值把数据（实参）传送到 FB 或 FC 的输入/输出形参。用户只能把相同类型的变量赋给输入/输出形参，无法进行数据格式转换。

在 FB 或 FC 语句内，可以使用和修改输入/输出形参。FB 或 FC 直接访问实参的变量并立即修改。

如果在输入/输出赋值中使用 STRING 字符串数据类型，那么实参的声明长度要大于或等于输入/输出形参，见下例。

```
FUNCTION_BLOCK REF_STRING
    VAR_IN_OUT
        io : STRING[80];
    END_VAR
;    // Statements
END_FUNCTION_BLOCK

FUNCTION_BLOCK test
    VAR
        my_fb : REF_STRING;
        str1 : STRING[100];
        str2 : STRING[50];
    END_VAR
    my_fb(io := str1);    // Permitted call
    my_fb(io := str2);    // Not permitted call,
                          // compiler error message

END_FUNCTION_BLOCK
```

图 47 在输入/输出赋值中使用 STRING 字符串数据类型

赋给输入/输出参数的变量必须能够直接读或写。因此，系统变量（SIMOTION 设备，工艺对象 TO），过程映像区的 I/O 变量不能赋给输入/输出参数。

3.4.4 参数传递给输出参数（只针对 FB）

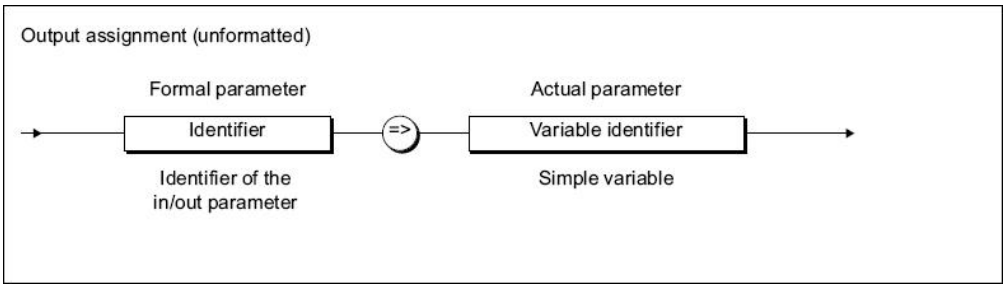


图 48 语法：输出赋值

用户使用输出赋值把 FB 的输出形参赋给变量（实参）。当 FB 调用后，变量接受输出形参的值。在 FB 语句中可以使用和修改输出形参。

对于参数传递而言，输出赋值是可选的。用户可以在任何时刻对 FB 输出参数进行读和写的操作。

3.4.5 参数访问时间

不同的访问类型导致访问参数的时间不同：

- 在输入赋值情况下，实参的值复制给了形参。如果大的数据结构，比如数组（ARRAY）被复制，而且经常调用 FC 或 FB，那么性能就会受到限制。
- 在输入/输出赋值情况下，不会复制参数的值，而是建立一个形参的存储地址和实参之间的链接。因此，传递变量比输入赋值的情况快（特别是包含大量数据的情况）。但是，从 FB 访问变量要慢一些。

• 如果使用单元变量，则不会有任何变量复制到 FC 或 FB。因为这些变量在整个 ST 源文件中都是有效的。

3.4.6 调用 FC

可按照如下规则调用：

• 带返回值的 FC （数据类型不是 VOID）

函数被放置在赋值的右手边，通常也作为一个在表达式内部的操作数出现。调用 FC 后，计算表达式的结果为它的返回值。

举例：

```
y := sin (x) ;
y := sin(in := x) ;
y := sqrt (1-cos(x)*cos(x)) ;
```

• 不带返回值的 FC（VOID 数据类型）

变量赋值只由调用 FC 组成。

举例：

```
funct1  (in1 := var11, in2 := var12, inout1 := var13) ;
```

3.4.7 调用 FB（背景数据块调用）

在调用 FB 功能块之前，必须声明一个背景数据块。可声明一个变量，然后输入 FB 的名称作为数据类型。可以如下声明背景数据块：

• 本地的

在程序或 FB 的声明部分，VAR/END_VAR 范围内。

• 全局的

在执行部分的接口处，VAR_GLOBAL/END_VAR 范围内。

• 作为输入/输出参数

在 FB 或 FC 的声明部分，VAR_IN_OUT/END_VAR 范围内。

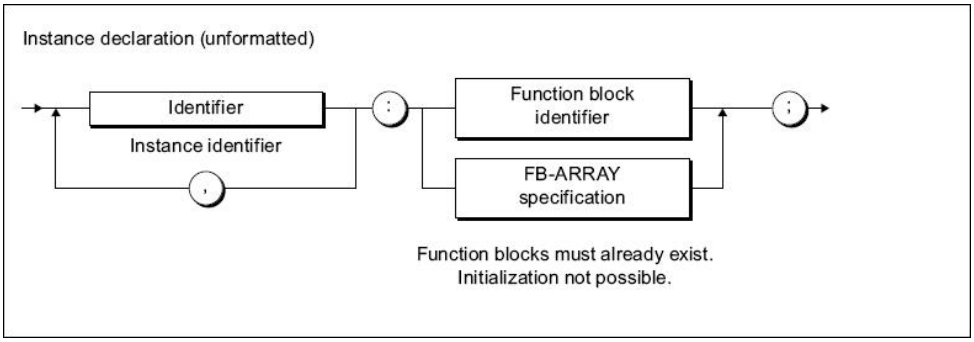


图 49 语法：声明背景数据块

背景数据块声明也可以是数组 ARRAY。比如：

FB_inst : ARRAY [1..2] OF FB_name

在程序组织单元 POU（Program Organization Units）的语句部分调用 FB 的背景数据块。FB 参数输入赋值和输入/输出赋值通过逗号分隔开。如图 50 所示。

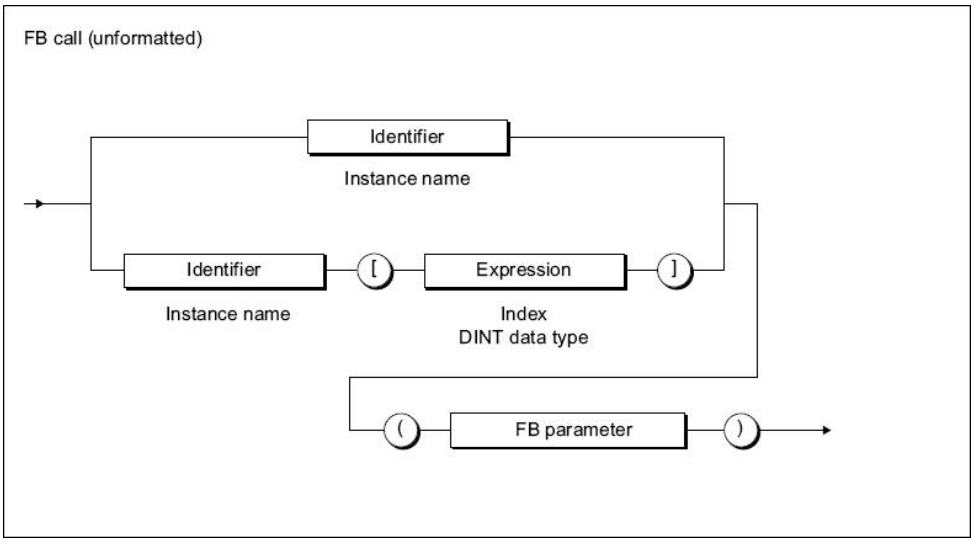


图 50 FB 调用语法

假定“supply”和“motor”FB 功能块已经定义完毕，举例如图 51。

- FB supply

输入参数 in1, in2；输入/输出参数 inout；输出参数 out

- FB motor

输入/输出参数 inout1，inout2；输出参数 out1，out2

```
VAR
    Supply1, Supply2: Supply;
    Motor1 : Motor;
END_VAR

// Parameter transfer (output assignment) when calling the instance of an FB
Supply1 (in1 := var11, in2 := expr12, inout := var13, out => var14) ;
Supply2 (in1 := var21, in2 := expr22, inout := var23, out => var24) ;
Motor1 (inout1 := var31, inout2 := var32, out1 => var33, out2 => var34);
// ...
// Accessing the FB's output parameter outside the FB
var15 := Supply1.out;
var25 := Supply2.out;
var35 := Motor1.out1;
var36 := Motor1.out2;
var41 := Motor1.out1 * Motor1.out2 * (Supply1.out + Supply2.out);
```

图 51 背景数据块声明，FB 调用，访问输出参数

在 FB 调用中，可使用结构变量从 FB 外部访问 FB 的输出参数。格式为 *FB instance name.output parameter* 比如，*Supply1.out*

也可使用结构变量从 FB 外部读或写 FB 的输入参数。格式为 *FB instance name.input parameter* 比如，*Supply1.in1*

FB 背景数据块名称本身不能用于变量赋值。

```
// Only with compiler option "Permit language extensions" activated
VAR
    var_fb   : _WORD_TO_2BYTE;
    var_word : WORD;
END_VAR
var_fb.wordin := var_word;
// ..
var_fb();
```

图 52 输入参数赋值

3.4.7 FB 调用注意事项

当调用 FB 背景数据块时，需注意：

- 只能把直接存储在存贮器中的变量赋给输入/输出参数。

  下列变量是允许的：

  -- 全局变量（单元变量和全局设备用户变量）

  -- 本地变量

  -- TO 的数据类型的变量（TO 背景数据块）

  下列变量是不允许的：

  -- 系统变量（TO 变量）

  -- 工程系统中的工艺对象名称

  -- I/O 变量

  -- 绝对和符号过程映像访问

- 不能把 FC 作为输入/输出参数

  调用 FC 时，不能在输入/输出赋值中把其当作实参。必须首先在本地变量中存储 FC 的结果，然后在输入/输出赋值中把此变量作为实参使用。

- 不能把常数作为输入/输出参数
- 输入/输出参数不能初始化

3.5 FB 和 FC 的比较

3.5.1 范例描述

为了简单起见，每种类型的参数只使用一次，但是在实际应用中，可以定义多个数量的参数。

在执行区域（implementation section）的声明部分（declaration part）定义 FB 和 FC，用来计算圆周长和面积，半径是输入变量。

- 定义输入参数为半径
- 定义输入/输出参数为圆周长。在调用 FB 或 FC 中，传递变量被直接赋值。
- 对 FB 和 FC，有多种方法定义圆面积
  -- 对 FB，可定义一个输出参数
  -- 对 FC，可使用其返回值，需恰当的定义返回值的数据类型
- 在程序区域，FB 或 FC 被调用，实参被赋值给下列形参：
  -- 对 FB：输入，输入/输出和输出参数
  -- 对 FC：输入，输入/输出参数

在调用 FB 或 FC 之后，圆周长和圆面积的计算值就可用了。

-- 对 FB：计算值在输入/输出和输出参数的实参中。可在 FB 的外部读取输出参数。

-- 对 FC：计算值在 FC 的返回值和输入/输出参数的实参中

3.5.2 程序源文件

Function block (FB)

```
INTERFACE
  PROGRAM CircleCalc1;
END_INTERFACE

  IMPLEMENTATION
    FUNCTION_BLOCK Circle1
      //Constant declaration
      VAR CONSTANT
        PI : LREAL := 3.1415 ;
      END_VAR
      //Input parameter
      VAR_INPUT
        Radius : LREAL;
      END_VAR
      //In/out parameter
      VAR_IN_OUT
        circumference : LREAL;
      END_VAR
      //Output parameter
      VAR_OUTPUT
        Area : LREAL;
      END_VAR
```

Function (FC)

```
INTERFACE
  PROGRAM CircleCalc2;
END_INTERFACE

IMPLEMENTATION
  FUNCTION Circle2 : LREAL
    //Constant declaration
    VAR CONSTANT
      PI : LREAL := 3.1415 ;
    END_VAR
    //Input parameter
    VAR_INPUT
      Radius : LREAL;
    END_VAR
    //In/out parameter
    VAR_IN_OUT
      circumference : LREAL;
    END_VAR
    //Output parameter
    // Not possible
```

```
    // Local variables, static
    VAR
      Counter : DINT;
      (* Variable retains its value
      between calls *)
    END_VAR
    //Call counter
    Counter := counter + 1 ;
    Circumference := 2 * PI * Radius ;
    Area := PI * Radius**2 ;
  END_FUNCTION_BLOCK
  PROGRAM CircleCalc1
    VAR
      myCircle1        : Circle1 ;
      myArea1, myArea2 : LREAL;
      myCircf          : LREAL;
    END_VAR;
    myCircle1(Radius := 3
      , Circumference := myCircf
      , Area => myArea1) ;
    myArea2 := myCircle1.Area ;
    // myCircf has the value 18,849
    // myArea1 has the value 28,274
    // myArea2 has the value 28,274
  END_PROGRAM
END_IMPLEMENTATION
```

```
    // Local variables, temporary
    VAR
      Counter : DINT;
      (* Variable will be initialized
      with 0 for each call *)
    END_VAR
    //Call counter
    Counter := Counter + 1 ;
    Circumference := 2 * PI * Radius ;
    Circle2 := PI * Radius**2 ;
  END_FUNCTION
  PROGRAM CircleCalc2
    VAR

      myArea  : LREAL;
      myCircf          : LREAL;
    END_VAR;
    myArea := Circle2(Radius := 3
      , Circumference := myCircf);


    // myCircf has the value 18,849
    // myArea has the value 28,274

  END_PROGRAM
END_IMPLEMENTATION
```

图 53 FB 和 FC 的例子程序