

# SIEMENS

## SIMATIC

## STEP 7 V5.5

### 使用入门

欢迎使用 STEP 7,  
目录

介绍 STEP 7	1
SIMATIC 管理器	2
用符号编程	3
在 OB1 中创建程序	4
创建一个具有功能块和数据块的程序	5
配置中央机架	6
下载和调试程序	7
编程一个功能	8
编程一个共享数据块	9
编程一个多重背景	10
组态分布式 I/O	11
附录	
附录 A	A

索引

本手册是文档包的一部分  
订货号为：  
**6ES7810-4CA10-8KW0**

**2010 年10 月版**  
A5E03290305-01

## 安全指南

本手册包括了应遵守的注意事项，以保证您自己的人身安全以及保护产品及所连接的设备。这些注意事项在本手册中均用下面所示的符号突出强调，并根据危险等级注明如下：



---

### 危险

表示如果不采取适当的预防措施，将会导致死亡、严重的人身伤害或重大的财产损失。

---



---

### 警告

表示如果不采取适当的预防措施，可能导致死亡、严重的人身伤害或重大的财产损失。

---



---

### 当心

表示如果不采取适当的预防措施，可能导致轻微的人身伤害。

---

---

### 当心

表示如果不采取适当的预防措施，可能造成财产损失。

---

---

### 注意

提醒您对产品有关的重要信息、产品的处置或文件的特别部分格外注意。

---

## 合格人员

只有合格人员才允许安装和操作这一设备。合格人员规定为根据既定的安全惯例和标准，对线路、设备和系统进行调试、接地和标记的人员。

## 正确使用

请注意下列事项：



---

### 警告

本设备及其组件只能用于产品目录或技术说明书中所描述的应用，并且只能与由西门子公司批准或推荐的其他生产厂商提供设备或元件相连接。

只有正确地运输、贮存、组装和安装本产品，并且按照推荐的方式运行和维护，才能正确安全地发挥其功能。

---

## 商标

SIMATIC®、SIMATIC HMI®和 SIMATIC NET®是 SIEMENS AG 的注册商标。

本文档中的其它一些标志也是注册商标，如果任何第三方出于己方的目的而使用，都会侵犯商标所有者的权利。

### Copyright © Siemens AG 2010 保留所有权利

未经明确的书面许可，不得复制、传送或使用本手册或其中的内容。违者要对造成的损失承担责任。保留所有权包括实用新型或设计的专利许可权及注册权。

### 免责声明

我们已核对本手册中的内容与所描述的硬件和软件相符。由于差错在所难免，所以我们不能保证完全一致。但是，我们会定期审查本手册中的数据，并在后续版本中进行必要的更正。欢迎提出改进意见。

Siemens AG  
Bereich Automation and Drives  
Geschaeftsgebiet Industrial Automation Systems  
Postfach 4848, D- 90327 Nuernberg

©Siemens AG 2010  
技术数据如有改变，恕不另行通知。

Siemens Aktiengesellschaft

A5E03290305-01

# 欢迎使用 STEP 7...

...STEP 7 是用于 SIMATIC S7-300/400 站创建可编程逻辑控制程序的标准软件，可使用梯形图、功能块图或语句表。

## 关于这本使用入门手册

在本手册中，您将会了解 SIMATIC STEP 7 的基础知识。我们将向您显示最重要的屏幕对话框并通过实际练习显示应遵循的步骤，而这些内容都有独立的结构，您几乎可以从任意一章开始。

每一章节分为两个部分：说明部分标以灰色；操作过程部分则标以绿色。指令从绿色页边内的箭头处开始，并且可能分布在多页中，结束处是一个停止符号，有一个包含相关主题的方框。

有关鼠标、窗口操作、下拉菜单等的预先经验是有用的，您最好熟悉可编程逻辑控制器的基本原理。

STEP 7 的培训课程则为您提供入门手册之外的更深入的知识，教您如何用 STEP 7 创建完整的自动化解决方案。

## 使用入门手册的要求

为完成这本入门手册中 STEP 7 的实际练习，您需要具备如下条件：

- 一台西门子编程设备或一台 PC
- STEP 7 软件包和相应的许可证密钥
- 一个 SIMATIC S7-300 或 S7-400 可编程控制器 (用于第 7 章“下载和调试程序”)。

## 有关 STEP 7 的其它资料

- STEP 7 基本信息
- STEP 7 参考信息

在您安装了 STEP 7 之后，在“开始”菜单的 **Simatic >** 文档下可以找到电子手册，或者也可以从西门子的任何一个销售中心订购它们。手册中的所有信息均可在 STEP 7 中从在线帮助中调出。

祝您好运！

Siemens AG



# 目录

<b>1</b>	<b>介绍 STEP 7</b>	
1.1	您将学到的内容	1-1
1.2	组合硬件和软件	1-3
1.3	使用 STEP 7 的基本步骤	1-4
1.4	安装 STEP 7	1-5
<b>2</b>	<b>SIMATIC 管理器</b>	
2.1	启动 SIMATIC 管理器并创建一个项目	2-1
2.2	SIMATIC 管理器中的项目结构以及如何调用在线帮助	2-4
		在第 3 到第 5 章中，您将创建一个简单的程序。
<b>3</b>	<b>用符号编程</b>	
3.1	绝对地址	3-1
3.2	符号编程	3-2
<b>4</b>	<b>在 OB1 中创建程序</b>	
4.1	打开 LAD/STL/FBD 编程窗口	4-1
4.2	用梯形图编程 OB1	4-4
4.3	用语句表编程 OB1	4-8
4.4	用功能块图编程 OB1	4-11
<b>5</b>	<b>创建一个具有功能块和数据块的程序</b>	
5.1	创建与打开功能块(FB)	5-1
5.2	用梯形图编程 FB1	5-3
5.3	用语句表编程 FB1	5-7
5.4	用功能块图编程 FB1	5-10
5.5	生成背景数据块和修改实际值	5-14
5.6	用梯形图编程块调用	5-16
5.7	用语句表编程块调用	5-19
5.8	用功能块图编程块调用	5-21

	在第 6 和第 7 章中，您将配置硬件并测试程序。	
<b>6</b>	<b>组态中央机架</b>	
6.1	配置硬件	6-1
<b>7</b>	<b>下载和调试程序</b>	
7.1	建立一个在线连接	7-1
7.2	下载程序到可编程控制器	7-3
7.3	用程序状态测试程序	7-6
7.4	用变量表测试程序	7-8
7.5	评估诊断缓冲区	7-12
	在第 8 章到第 11 章中，您将扩展包括新功能在内的知识面。	
<b>8</b>	<b>编程一个功能</b>	
8.1	创建并打开功能(FC)	8-1
8.2	编程功能	8-3
8.3	在 OB1 中调用功能	8-6
<b>9</b>	<b>编程一个共享数据块</b>	
9.1	创建和打开共享数据块	9-1
<b>10</b>	<b>编程一个多重背景</b>	
10.1	创建和打开较高一级的功能块	10-1
10.2	编程 FB10	10-3
10.3	生成 DB10 并调整实际值	10-7
10.4	在 OB1 中调用 FB10	10-9
<b>11</b>	<b>组态分布式 I/O</b>	
11.1	用 PROFIBUS DP 组态分布式 I/O	11-1
	<b>附录 A</b>	<b>A-1</b>
	入门手册的示例项目概述	
	<b>索引</b>	<b>索引 1</b>

# 1 介绍 STEP 7

## 1.1 您将学到的内容

通过实际的练习，我们将向您展示使用 STEP7 的梯形图、语句表或功能块图编程是多么的容易。

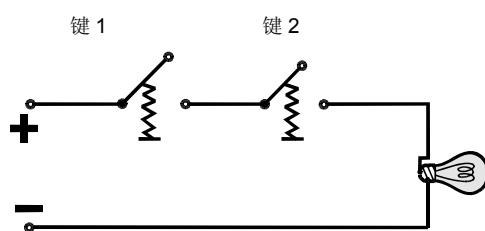
各章中详细的指导将逐步地为您介绍使用 STEP 7 的诸多方法。

### 用二进制逻辑创建一个程序

在第 2 章到第 7 章中，介绍了如何使用二进制逻辑创建一个程序。使用已编程的逻辑操作，可以寻址 CPU (如果存在的话)的输入和输出。

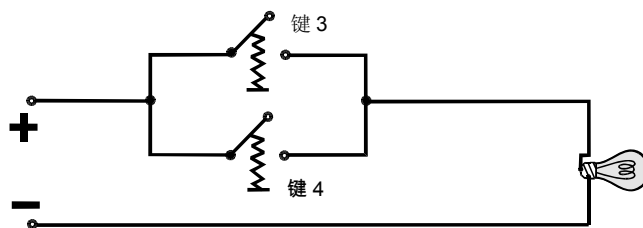
入门手册中的编程示例主要基于三个基本的二进制逻辑运算。

第一个二进制逻辑运算是 AND (与)功能，稍后我们将用它进行编程。下面的具有两个键的电路图可以很好地说明 AND 功能。



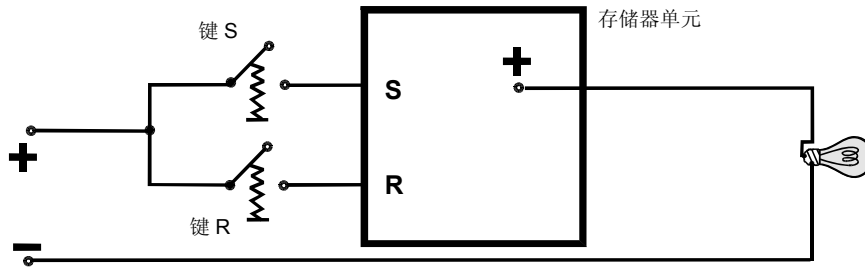
只有同时按下键 1 和键 2，灯泡才会点亮。

第二个二进制逻辑运算是 OR (或)功能。OR 功能可由以下电路图来表达。



不管按下键 3 或者键 4，灯泡都将点亮。

第三个二进制逻辑操作是存储器单元。在电路图中，SR 功能对某一电压状态作出响应并相应地传递这一状态。



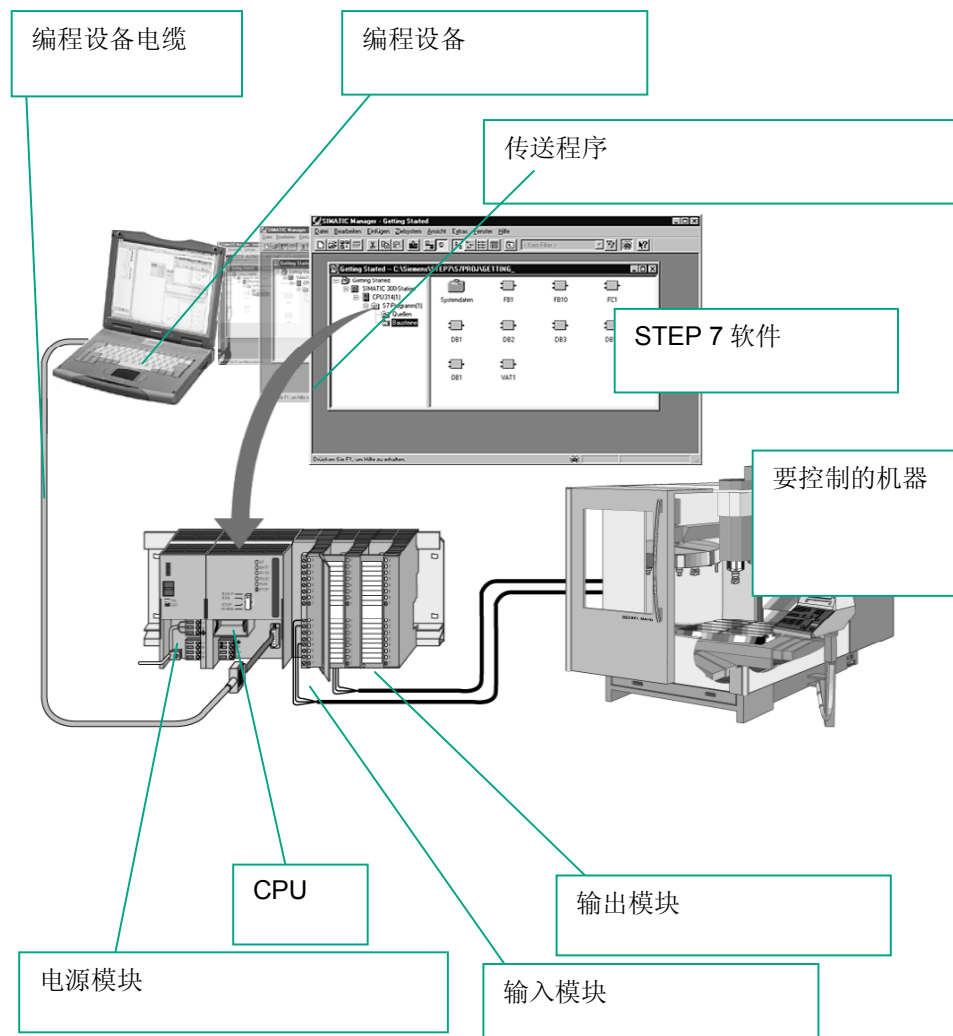
如果按下 S 键，则点亮灯泡并一直保持，直到按下 R 键。



## 1.2 组合硬件和软件

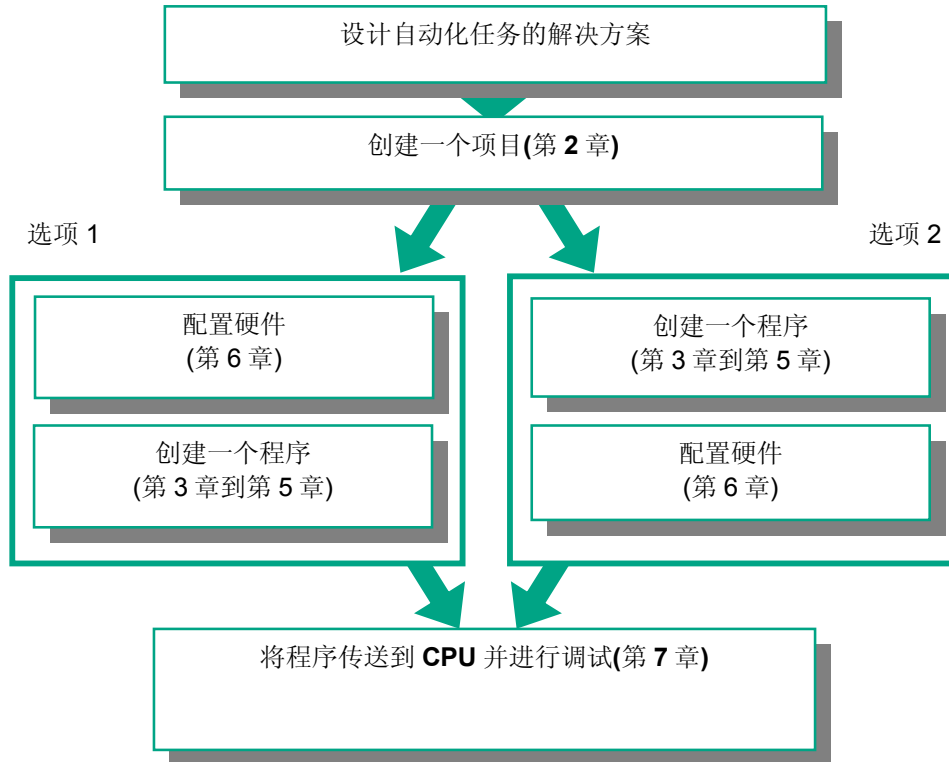
使用 STEP 7 软件，可以在一个项目中创建 S7 程序。S7 可编程控制器包括一个供电单元、一个 CPU，以及输入和输出模块(I/O 模块)。

可编程逻辑控制器(PLC)通过 S7 程序监控机器。在 S7 程序中通过地址寻址 I/O 模块。



### 1.3 使用 STEP 7 的基本步骤

在创建一个项目之前，您应该了解 STEP 7 项目可以按不同的顺序创建。



如果要创建一个使用许多输入和输出的综合程序，我们建议先做硬件配置。这样做的优点在于 STEP 7 在硬件配置编辑器中会显示可能的地址。

如果选择第二个选项，那么您只能根据所选组件来自行确定每个地址，而不能通过 STEP 7 调用这些地址。

在硬件配置中，您不仅可以定义地址，还可以改变模块的参数和属性。例如，如果要操作多个 CPU，则必须区分各个 CPU 的 MPI 地址。

由于在使用入门手册中我们只使用了少量的输入和输出，我们可以暂时跳过硬件配置，从编程开始。

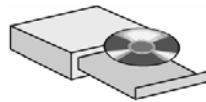
## 1.4 安装 STEP 7

无论您想从编程开始还是想从硬件配置开始，首先必须安装 STEP 7。如果使用的是 SIMATIC 编程设备，则 STEP 7 已经事先安装完毕。



在编程设备或者 PC 上安装 STEP 7 软件时，如果该设备以前没有安装过 STEP 7，则要注意安装 STEP 7 对软件和硬件要求。这些要求可以在 STEP 7 光盘的 Readme.wri 文件中找到，该文件所在的路径为

**<驱动器>:\STEP 7 \Disk1。**



如果您需要先安装 STEP 7，则现在就将 STEP 7 光盘插入到光盘驱动器中。安装程序将自动启动。按照屏幕上的指令进行操作。

如果安装程序没有自动启动，则可以在光盘驱动器的以下路径中找到安装程序

**<驱动器>:  
\STEP 7 \Disk1\setup.exe。**



SIMATIC Manager

一旦安装完成并重新启动计算机后，“SIMATIC 管理器”的图标将显示在 Windows 桌面上。

安装之后，双击“SIMATIC 管理器”图标，STEP 7 向导将自动启动。

在 STEP 7 光盘的 Readme.wri 文件中可以找到关于安装的其他注意事项，该文件位于

**<驱动器>:\STEP 7 \Disk1\Readme.wri。**

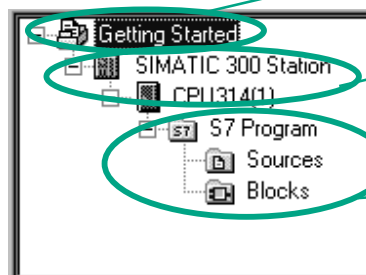


## 2 SIMATIC 管理器

### 2.1 启动 SIMATIC 管理器并创建一个项目

SIMATIC 管理器是 STEP 7 的中央窗口，在 STEP 7 启动时激活。缺省设置启动 STEP 7 向导，它可以在您创建 STEP 7 项目时提供支持。用项目结构来按顺序存储和排列所有的数据和程序。

在一个项目中，数据在分层结构中以对象的形式保存



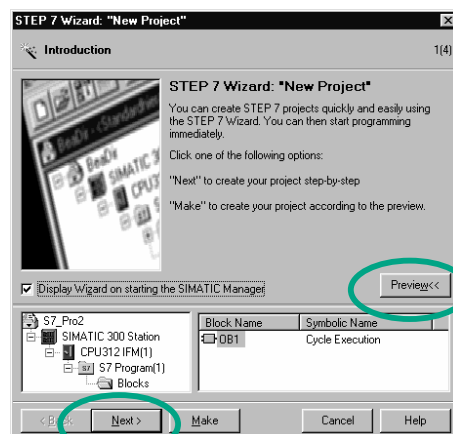
SIMATIC 站和 CPU 包含硬件的配置和参数数据

S7 程序包含了所有的块，这些块中有控制机器所需的程序



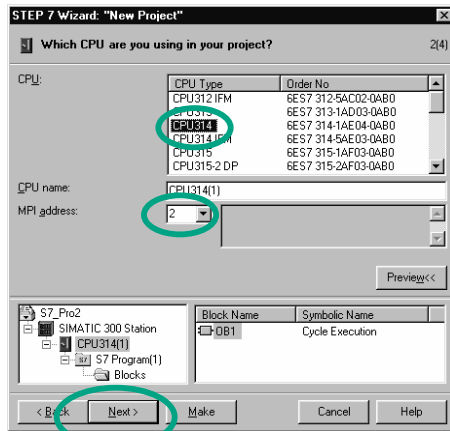
SIMATIC Manager

双击 Windows 桌面上的 **SIMATIC 管理器** 图标，如果向导没有自动启动，请选择菜单命令文件 > “新建项目” 向导。



在预览中，您可以显示或隐藏正在创建的项目结构的视图。

要转到下一个对话框，请单击下一步。



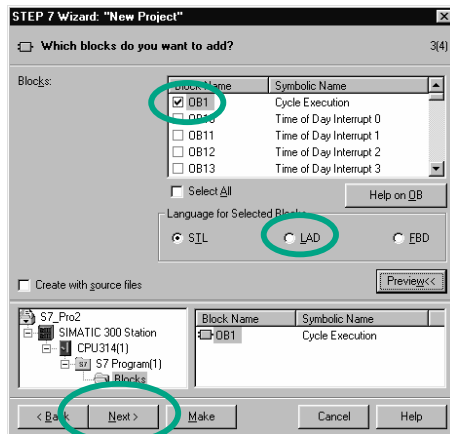
对于“Getting Started”示例项目，请选择 CPU 314。实际上，该示例支持您随时选择您所得到的 CPU。

MPI 地址的缺省设置为 2。

单击下一步确认设置，进入下一个对话框。

每个 CPU 都有某些特性；例如，关于其存储器组态或地址区域。这也是为什么在编程前必须要选择 CPU。

为了使 CPU 与编程设备或 PC 之间进行通讯，需要设置 MPI 地址(多点接口)。



请选择组织块 **OB1** (如果尚未选中)。

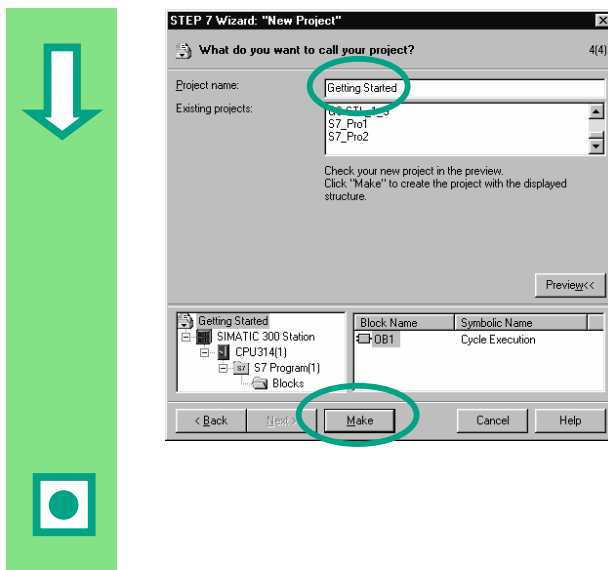
请选择以下一种编程语言：梯形图 (LAD)、语句表(STL)或功能块图 (FBD)。

单击下一步确认设置。

OB1 代表最高的编程层次，它负责组织 S7 程序中的其它块。

您也可以在以后重新改变编程语言。





在“项目名称”域中双击选中默认的名称，并用“Getting Started”重写。

请单击生成，如前面预览的那样生成新项目。

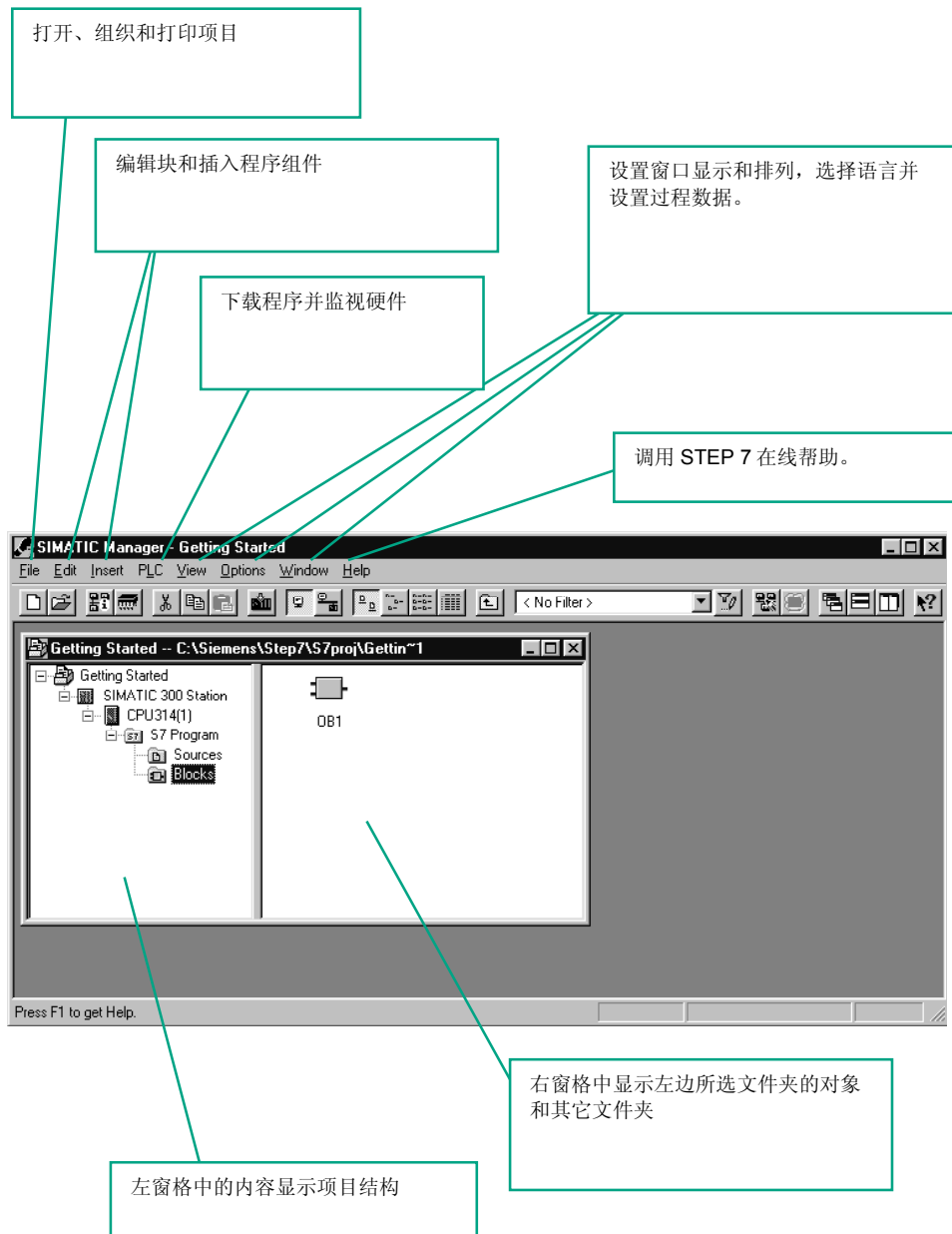
当单击生成按钮时，将一同打开 SIMATIC 管理器和刚刚创建的“Getting Started”项目的窗口。在随后的几页中，我们将向您说明创建文件和文件夹的目的以及如何有效地使用它们。

每次启动程序时都将激活 STEP 7 向导。您可以在向导的第一个对话框中取消这个缺省设置。但是，如果不使用 STEP 7 向导，则创建项目时您必须自行创建项目的每个目录。

在帮助 > 目录下的主题“建立和编辑项目”中可以找到更多的信息。

## 2.2 SIMATIC 管理器中的项目结构以及如何调用在线帮助

STEP 7 向导关闭后，立即出现 SIMATIC 管理器以及打开的“Getting Started”项目窗口。从这里可以启动所有的 STEP 7 功能和窗口。





## 调用 STEP 7 中的帮助

## F1

## 方法 1:

将光标放在任意菜单命令上并按 **F1** 键。出现所选菜单命令的上下文相关的帮助。

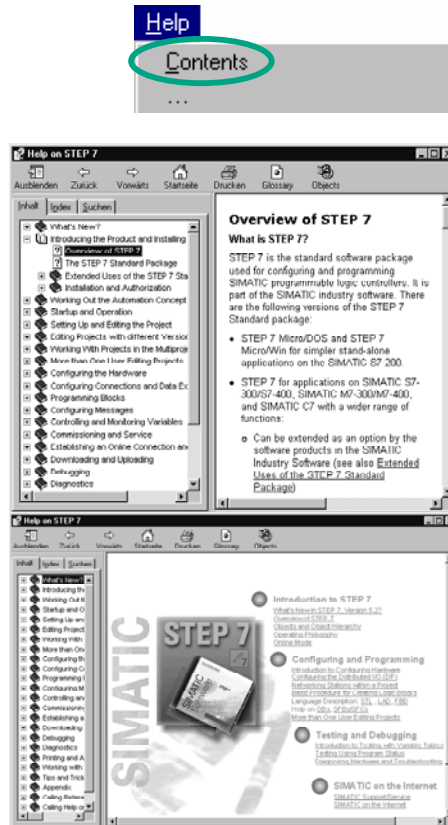
## 方法 2:

用菜单打开 STEP 7 的在线帮助。

包含各种帮助主题的目录页出现在左窗格中，而所选主题的内容显示在右窗格中。

单击目录列表中的 **+** 号可以查找到您想查看的主题。同时，所选择主题的内容显示在右窗格中。

使用索引和查找，可以输入字符串来查找所需要的特定主题。



## 方法 3:

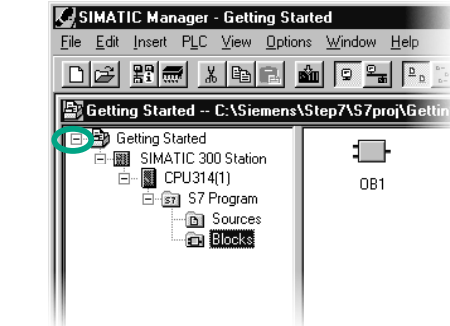
单击 STEP 7 在线帮助中的“起始页”图标，打开信息入口。可在该入口中直接访问在线帮助的主要主题，例如：

- STEP 7 使用入门
- 组态与编程
- 测试与调试
- Internet 上的 SIMATIC

## 方法 4:

单击工具栏中的问号按钮，将鼠标变成帮助光标。这样，下次单击一个特定的对象时，将激活在线帮助功能。

## 项目结构



将显示所创建的项目以及所选的 S7 站和 CPU。

单击+号或者-号可打开或关闭文件夹。

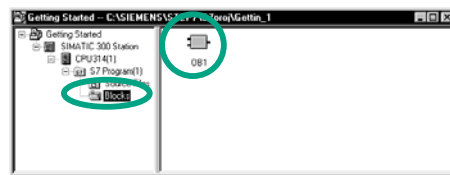
之后您可以单击右窗格中显示的符号来启动其它功能。



单击 **S7 程序(1)** 文件夹。这里包含了所有必须的程序组件。

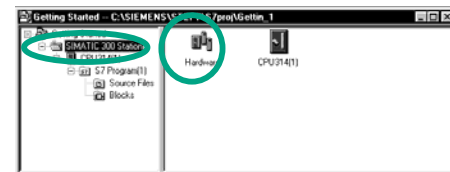
在第 3 章中将使用符号组件来给地址定义符号名。

源文件组件用来存储源文件。本使用入门手册不涉及这一部分。



单击 **Blocks** 文件夹。这里包含已经创建的 **OB1** 以及以后将创建的所有其它块。

在这里，您就可以开始使用第 4 章和第 5 章中的梯形图、语句表，或者功能块图进行编程。



单击 **SIMATIC 300 站** 文件夹。所有与硬件相关的项目数据都存储在这里。

在第 6 章中将使用硬件组件来指定可编程控制器的参数。

您的自动化任务可能还需要其它的 SIMATIC 软件；例如，可选软件包 **PLCSIM** (硬件模拟程序)或 **S7 Graph** (图形编程语言)，它们都集成在 **STEP 7** 中。例如，使用 **SIMATIC 管理器**，可以直接打开像 **S7 Graph** 功能块这样的相关对象。

可以在**帮助 > 目录**下的主题“设计自动化概念”和“设计程序结构的基础”中找到更多的信息。

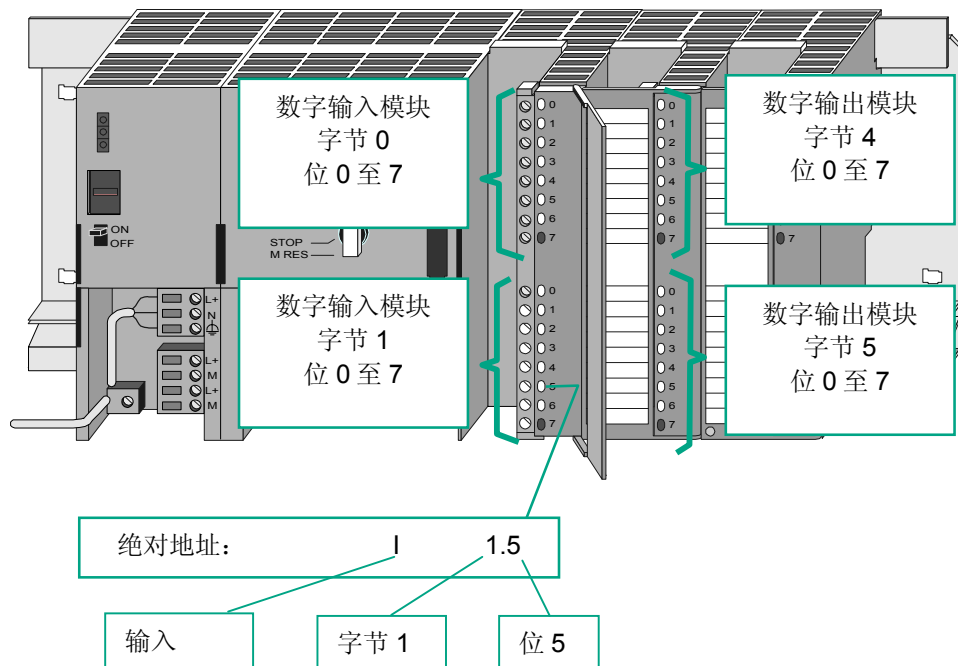
在 **SIMATIC** 目录 **ST 70** 的“完全集成自动化组件”中可以找到更多的关于可选软件包的信息。

## 3 使用符号编程

### 3.1 绝对地址

每个输入和输出都有一个由硬件配置预定义的绝对地址。该地址是直接指定的，即为绝对地址。

该绝对地址可以用您所选择的任何符号名替换。



如果在您的 S7 程序中寻址的输入与输出并不多，应该只使用绝对地址编程。

### 3.2 符号编程

在符号表中，可以为所有要在程序中寻址的绝对地址分配符号名和数据类型；例如，为输入 I1.0 分配符号名 **Key1**。这些名称可以用在程序的所有部分，即是所说的全局变量。

使用符号编程可以大大地提高已创建的 **S7** 程序的可读性。

#### 使用符号编辑器



在“Getting Started”项目窗口查找到 **S7 程序(1)**，然后双击打开符号组件。



当前符号表中只包括预定义的组织块 **OB1**。

Status	Symbol	Address	Data type
1	Cycle Execution	OB 1	OB 1
2			

单击**循环执行**，且用“主程序”作为我们的示例将其重写。

Status	Symbol	Address	Data type
1	Main Program	OB 1	OB 1
2	Green Light	Q 4.0	BOOL

在第二行输入“Green Light”和“Q 4.0”。将自动添加数据类型。

Comment				

单击第一行或第二行的注释栏，为符号输入注释。完成一行后按**回车**键，会自动添加一新行。

Status	Symbol	Address	Data type
1	Main Program	OB 1	OB 1
2	Green Light	Q 4.0	BOOL
3	Red Light	Q 4.1	BOOL

在第三行输入“Red Light”和“Q 4.1”，按回车键结束该项。

用这种方式可以为程序需要的所有输入与输出的绝对地址分配符号名。



保存符号表中已经完成的输入或修改并关闭窗口。

因为在整个“Getting Started”项目中有很多名称，您可以在第 4.1 节中将符号表复制到“Getting Started”项目中。

Status	Symbol	Address	Data type	Comment
1	Automatic_Mode	Q 4.2	BOOL	Retentive output
2	Automatic_On	I 0.5	BOOL	For the memory function (switch on)
3	DE_Actual_Speed	MW 4	INT	Actual speed for diesel engine
4	DE_Failure	I 1.6	BOOL	Diesel engine failure
5	DE_Fan_On	Q 5.6	BOOL	Command for switching on diesel engine fan
6	DE_Follow_On	T 2	TIMER	Follow-on time for diesel engine fan
7	DE_On	Q 5.4	BOOL	Command for switching on diesel engine
8	DE_Preset_Speed_R...	Q 5.5	BOOL	Display "Diesel engine preset speed reached"
9	Diesel	DB 2	FB 1	Data for diesel engine
10	Engine	FB 1	FB 1	Engine control
11	Fan	FC 1	FC 1	Fan control
12	Green_Light	Q 4.0	BOOL	Result of AND query
13	Key_1	I 0.1	BOOL	For the AND query
14	Key_2	I 0.2	BOOL	For the AND query
15	Key_3	I 0.3	BOOL	For the OR query
16	Key_4	I 0.4	BOOL	For the OR query
17	Main_Program	OB 1	OB 1	This block contains the user program
18	Manual_On	I 0.6	BOOL	For the memory function (switch off)
19	PE_Actual_Speed	MW 2	INT	Actual speed for petrol engine
20	PE_Failure	I 1.2	BOOL	Petrol engine failure
21	PE_Fan_On	Q 5.2	BOOL	Command for switching on petrol engine fan
22	PE_Follow_On	T 1	TIMER	Follow-on time for petrol engine fan
23	PE_On	Q 5.0	BOOL	Command for switching on petrol engine
24	PE_Preset_Speed_Re	Q 5.1	BOOL	Display "Petrol engine preset speed reached"
25	Petrol	DB 1	FB 1	Data for petrol engine
26	Red_Light	Q 4.1	BOOL	Result of OR query
27	S_Data	DB 3	DB 3	Shared data block
28	Switch_Off_DE	I 1.5	BOOL	Switch off diesel engine
29	Switch_Off_PE	I 1.1	BOOL	Switch off petrol engine
30	Switch_On_DE	I 1.4	BOOL	Switch on diesel engine
31	Switch_On_PE	I 1.0	BOOL	Switch on petrol engine
32				

在这里您可以看到“Getting Started”示例中语句列表的 S7 程序的符号表。

一般说来，不论选用哪种编程语言，每个 S7 程序只创建一个符号表。

所有可打印的字符(如，特殊字符、空格)都可以在符号表中使用。

以前自动添加到符号表中的数据类型决定了将由 CPU 处理的信号的类型。STEP 7 还可以使用以下数据类型：

BOOL BYTE WORD DWORD	这种类型的数据是位的组合。1 位(布尔型)至 32 位(双字型)。
CHAR	这种类型的数据只占 ASCII 字符集中的一个字符。
INT DINT REAL	它们可用于处理数值(例如，计算数学表达式)。
S5TIME TIME DATE TIME_OF_DAY	这种类型的数据在 STEP 7 中代表不同的时间和日期值(例如，设定日期或为定时器输入时间值。)

在帮助 > 目录的主题“编程块”和“定义符号”中可以找到更多的信息。



## 4 在 OB1 中创建程序

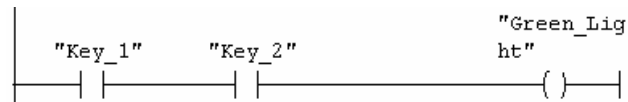
### 4.1 打开 LAD/STL/FBD 编程窗口

选择梯形图、语句表、或功能块图

在 STEP 7 中，可以用标准语言梯形图(LAD)、语句表(STL)或功能块图(FBD)创建 S7 程序。在实际使用时，您必须决定使用哪种语言，在本章也是如此。

#### 梯形图(LAD)

例如，适用于电气行业的用户。



#### 语句表(STL)

例如，适用于计算机技术领域的用户。

```
A    "Key_1"  
A    "Key_2"  
=    "Green_Light"
```

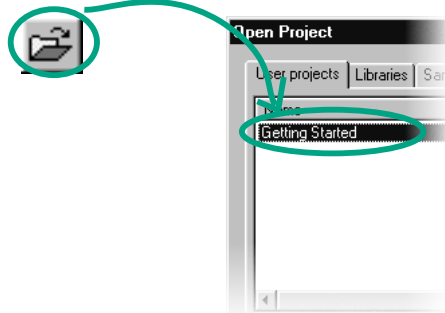
#### 功能块图(FBD)

例如，适用于电气工程领域的用户。



在项目向导中创建该块时所选择的语言打开 OB1 块。然而，您可以随时更改这个缺省的编程语言。

### 复制符号表并打开 OB1



如有必要，打开“Getting Started”项目。为此，单击工具栏中的打开按钮，选择所创建的“Getting Started”项目，并按确定确认。

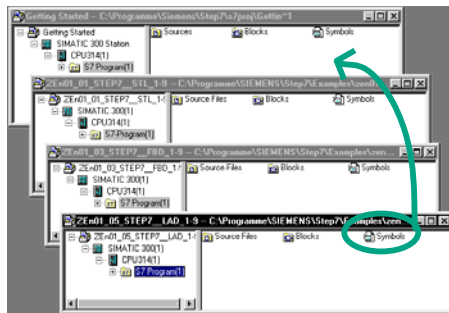
根据您所选用的编程语言，在“示例项目”标签中打开下列项目之一：

- ZEn01\_05\_STEP7\_LAD\_1-9
- ZEn01\_01\_STEP7\_STL\_1-9

或

- ZEn01\_03\_STEP7\_FDB\_1-9

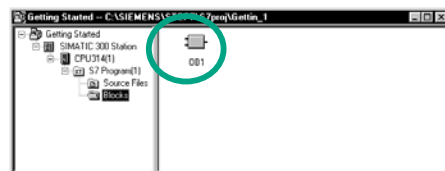
在这里您可以看到显示的所有三个示例项目。



在“ZEN01\_XXX”中浏览，直到找到符号组件，用拖放功能将该符号组件复制到项目窗口“Getting Started”的 S7 程序文件夹中。

然后，关闭窗口“ZEN01\_XXX”。

拖放功能就是用鼠标单击任意对象，按住鼠标的同时移动。当松开鼠标时，对象将被粘贴到所选择的位置。



双击“Getting Started”项目中的 OB1。打开 LAD/STL/FBD 编程窗口。

在 STEP 7 中，CPU 循环处理 OB1。CPU 逐行地读取并执行程序命令。当 CPU 返回到第一个程序行时，它已经完成一个循环。所需要的时间即所说的扫描循环时间。

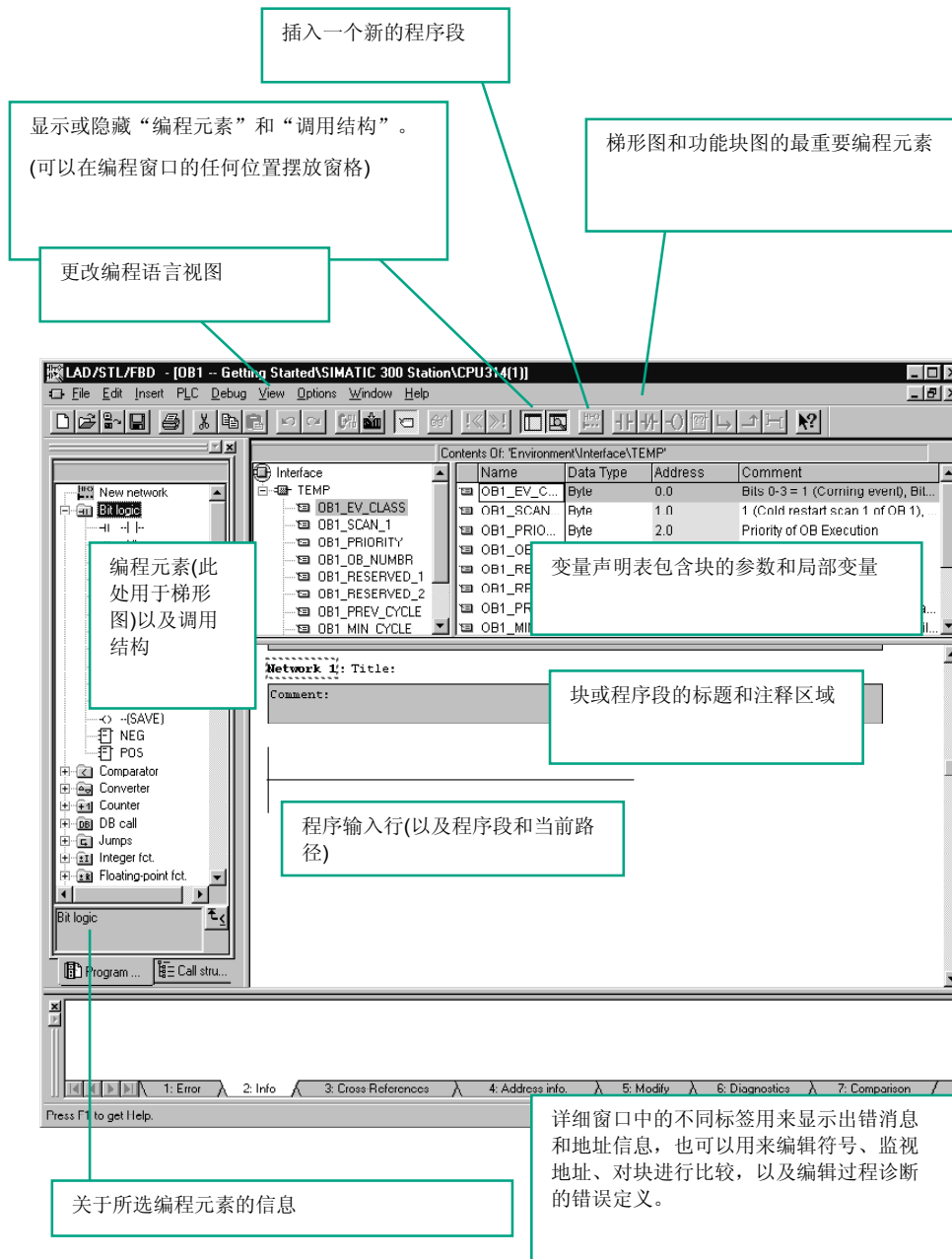
根据所选择的编程语言，继续阅读第 4.2 节(用梯形图编程)、第 4.3 节(用语句表编程)或第 4.4 节(用功能块图编程)。

在帮助 > 目录下的主题“编程块”和“创建块和库”中可以找到更多的信息。



## LAD/STL/FBD 编程窗口

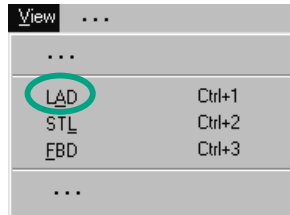
所有块都在 LAD/STL/FBD 编程窗口中进行编辑。这里，您可以看到梯形图的视图。



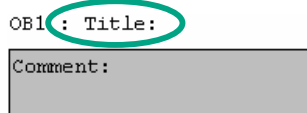
## 4.2 用梯形图编程 OB1

在下面的章节中，将使用梯形图(LAD)编程一个串联电路、一个并联电路和置位/复位存储器功能。

### 使用梯形图编程一个串联电路



如有必要，请在视图菜单中将 **LAD** 设置为编程语言。



单击 OB1 中的标题区域，作为示例，输入“循环处理的主程序”。



请为第一个元素选择电流通路。



请单击工具栏中的按钮，并插入一个常开触点。



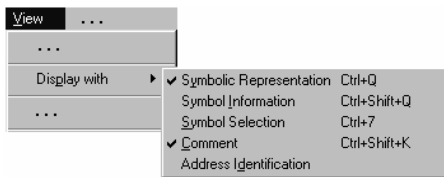
以同样的方式，插入第二个常开触点。



在电流通路的右端插入一个线圈。



串联电路中的常开触点和线圈还没有地址。



请检查符号表达式是否已经激活。



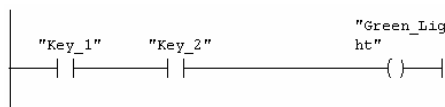
请单击“???”符号并输入符号名“Key\_1”（不包括引号）。同样，也可以从所显示的下拉列表中选择名称。用回车键确认。



为第二个常开触点输入符号名“Key\_2”。



为线圈输入名称“Green\_Light”。



现在您已经编程了一个完整的串联电路。



如果没有符号显示为红色，则保存该块。

如果符号不存在于符号表中，或者有语法错误，则该符号显示为红色。

### 使用梯形图编程一个并联电路



选择程序段 1。



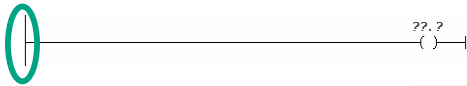
插入一个新的程序段。



再次选择电流通路。



插入一个常开触点和一个线圈。



选择电流通路的垂直线。



插入一个并行分支。



在并行分支上添加另一个常开触点。



闭合分支(如有必要, 可选择向下的箭头)。



在并联电路中还没有输入地址。

要分配符号地址, 可按照与串联电路相同的方法进行。



用“Key\_3”来覆盖上面的常开触点, 用“Key\_4”覆盖下面的触点, 线圈则为“Red\_Light”。



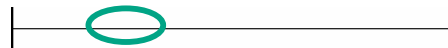
保存该块。



### 使用梯形图编程一个存储器功能



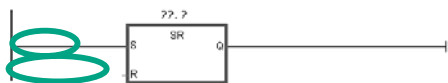
选择程序段 2 并插入另一程序段。



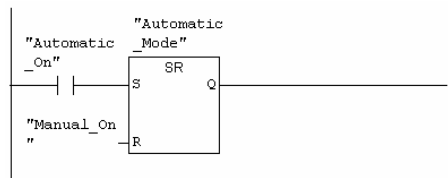
再次选择电流通路。



在编程元素目录的位逻辑下查找到 **SR** 元素。双击插入该元素。



分别在 **S** 和 **R** 的输入之前插入一个常开触点。

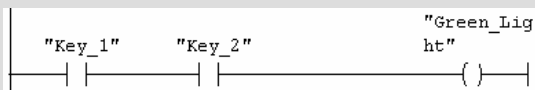


请为 **SR** 元素输入以下符号名：  
上面触点的名称为 “Automatic\_On”  
下面触点的名称为 “Manual\_On”  
**SR** 元素的名称为 “Automatic\_Mode”

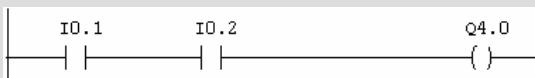


保存该块，并关闭窗口。

如果要查看绝对寻址和符号寻址之间的差别，请释放菜单命令 **视图 > 显示 > 符号表达式**。



示例：  
LAD 中的符号寻址



示例：  
LAD 中的绝对寻址

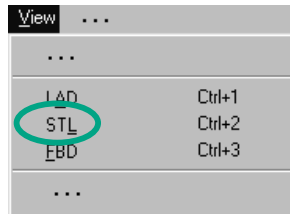
要改变 LAD/STL/FBD 编程窗口符号寻址的行断，可使用菜单命令 **选项 > 自定义**，然后选择 “LAD/FBD” 标签中的 “地址区域的宽度”。这里，可以将行断设置为 10 到 26 个字符。

在 **帮助 > 目录** 下的主题 “编程块”、“创建逻辑块” 和 “编程梯形图指令” 中可以找到更多的信息。

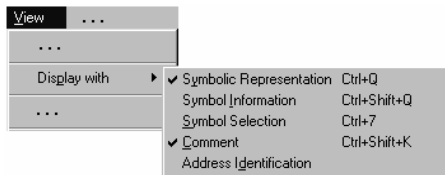
### 4.3 用语句表编程 OB1

在下面的章节中，将使用语句表(STL)编程一个 AND 指令、一个 OR 指令和存储器指令置位/复位。

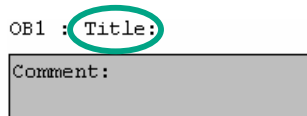
#### 使用语句表编程一个 AND 指令



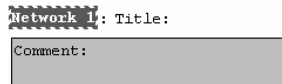
如有必要，请在视图菜单中将 **STL** 设置为编程语言。



请检查符号表达式是否已经激活。



单击 **OB1** 中的标题区域，作为示例，输入“循环处理的主程序”。



为第一条语句选择一个区域。



在第一个程序行输入 A (表示 AND) 和一个空格，然后输入符号名 “Key\_1” (不包括引号)。

用 **回车** 键完成该行。光标跳到下一行。



```
A    "Key_1"
A    "Key_2"
=    "Green_Light"
```

按同样的方法，完成所示的 AND 指令。



现在您已经编程了一条完整的 AND 指令。如果没有符号显示为红色，则保存该块。

如果符号不存在于符号表中，或者有语法错误，则该符号显示为红色。  
您还可以从符号表中直接插入符号名。请单击??符号，然后选择菜单命令插入 > 符号。滚动下拉列表，找到相应的名称并选中它。符号名则自动添加。

### 使用语句表编程一个 OR 指令

**Network 1** Title: \_\_\_\_\_  
Comment: \_\_\_\_\_

选择程序段 1。



插入一个新的程序段并再次选择输入区域。

```
O    "Key_3"

O    "Key_3"
O    "Key_4"
=    "Red_Light"
```

输入一个 O (表示 OR)和符号名 “Key\_3” (与 AND 指令的方法相同)。

完成 OR 指令并保存。

### 使用语句表编程一个存储器指令



选中程序段 2 并插入另一程序段。

```
A      "Automatic_On"
```

在第一行中输入指令 A 和符号名 "Automatic\_On"。

```
A      "Automatic_On"
```

完成存储器指令并保存。关闭该块。

```
S      "Automatic_Mode"
```

```
A      "Manual_On"
```

```
R      "Automatic_Mode"
```

如果要查看绝对寻址和符号寻址之间的差别，请释放菜单命令视图 > 显示 > 符号表达式。

```
A      "Key_1"  
A      "Key_2"  
=      "Green_Light"
```

示例：  
STL 中的符号寻址

```
A      I      0.1  
A      I      0.2  
=      Q      4.0
```

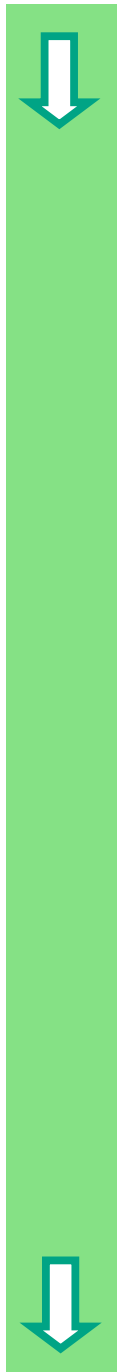
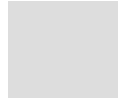
示例：  
STL 中的绝对寻址

在帮助 > 目录下的主题“编程块”、“创建逻辑块”和“编程 STL 语句”中可以找到更多的信息。

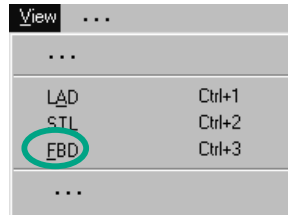


## 4.4 用功能块图编程 OB1

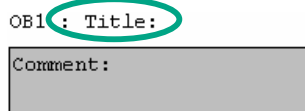
在下面的章节中，将使用功能块图(FBD)编程一个 AND 功能、一个 OR 功能和一个存储器功能。



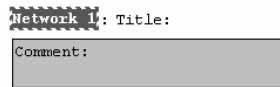
### 使用功能块图编程一个 AND 功能



如有必要，请在视图菜单中将 **FBD** 设置为编程语言。



单击 OB1 中的标题区域，作为示例，输入“循环处理的主程序”。



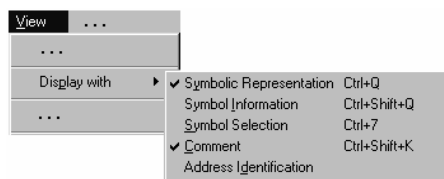
选择 AND 功能的输入区域(在注释区域下面)。



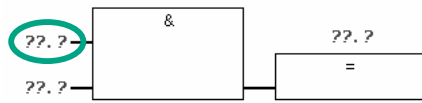
插入一个 AND 逻辑框(&)和一个赋值符号(=)。



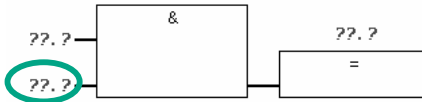
AND 功能中各元素的地址还未输入。



请检查符号表达式是否已经激活。



请单击`???`符号并输入符号名“Key\_1” (不包括引号)。同样，也可以从所显示的下拉列表中选择名称。用**回车**键确认。



为第二个输入输入符号名“Key\_2”。



为赋值输入名称“Green\_Light”。



现在您已经编程了一个完整的 AND 功能。



如果没有符号显示为红色，则可以保存该块。

如果符号不存在于符号表中，或者有语法错误，则该符号显示为红色。



使用功能块图编程一个 OR 功能



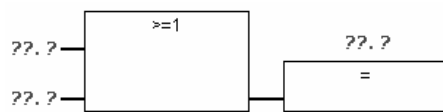
插入一个新的程序段。

Network 2: Title:  
Comment:

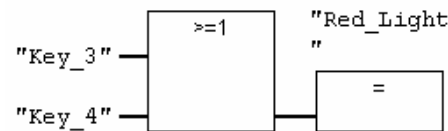
再次选择 OR 功能的输入域。



插入一个 OR 逻辑框( $\geq 1$ )和一个赋值符号(=)。



在 OR 功能中还没有输入地址。请按照与 AND 功能同样的方法，完成所示的 OR 功能。

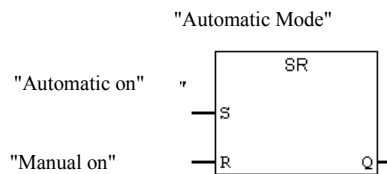
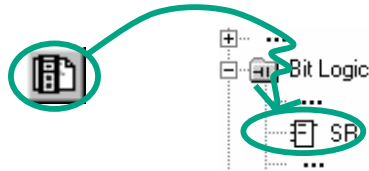


请为上面的输入端输入“Key\_3”，为下面的输入端输入“Key\_4”，为赋值输入“Red\_Light”。



保存该块。

### 使用功能块图编程一个存储器功能



选中程序段 2 并插入另一程序段。再次选择输入域(在注释域下面)。

在编程元素目录的位逻辑下查找到 SR 元素。双击插入该元素。

为 SR 元素输入以下符号名：  
置位 “Automatic\_On”  
复位 “Manual\_On”  
存储器位 “Automatic\_Mode”。



保存该块，并关闭窗口。

如果要查看绝对寻址和符号寻址之间的差别，释放菜单命令视图 > 显示 > 符号表达式。



示例：  
FBD 中的符号寻址



示例：  
FBD 中的绝对寻址

要改变 LAD/STL/FBD 编程窗口符号寻址的行断，可使用菜单命令选项 > 自定义，然后选择“LAD/FBD”标签中的“地址区域的宽度”。这里，可以将行断设置为 10 到 26 个字符。

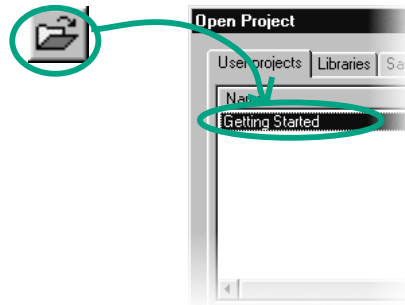
在帮助 > 目录下的主题“编程块”、“创建逻辑块”和“编程 FBD 语句”中可以找到更多的信息。

## 5 创建一个带有功能块和数据块的程序

### 5.1 创建并打开功能块(FB)

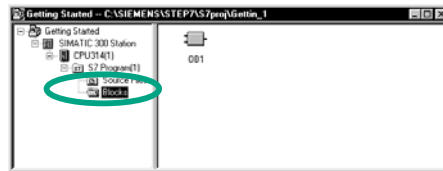
功能块(FB)在程序的体系结构中位于组织块之下。它包含程序的一部分，这部分程序在 OB1 中可以多次调用。功能块的所有形参和静态数据都存储在一个单独的、被指定给该功能块的数据块(DB)中。

然后在您所熟悉的 LAD/STL/FBD 编程窗口编程功能块(FB1，符号名“Engine”；请参见 3-3 页的符号表)。为此编程时应使用和第 4 章(编程 OB1)相同的编程语言。



此时，应当已将符号表复制到项目“Getting Started”中。如果尚未复制，请参见第 4-2 页上有关如何进行复制的信息，然后再返回到该章节。

如果必要，打开“Getting Started”项目。

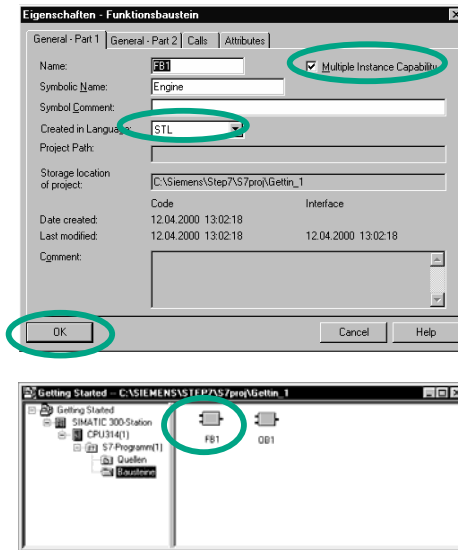


找到 **Blocks** 文件夹并打开它。

用鼠标右击右窗口。



按鼠标右键出现的弹出菜单中包含菜单栏中最重要的命令。插入一个功能块作为新对象。



在“属性 - 功能块”对话框中，选择用以生成块的语言，激活多重背景 **FB** 的检查框，用确定确认其余的设置。

将功能块 **FB1** 插入到 **Blocks** 文件夹中。

双击 **FB1**，打开 LAD/STL/FBD 编程窗口。

为此，根据所选择的编程语言，继续阅读第 5.2 节(用梯形图编程)、第 5.3 节(用语句表编程)或第 5.4 节(用功能块图编程)。

在帮助 > 目录的主题“编程块”和“创建块和库”中可以找到更多的信息。

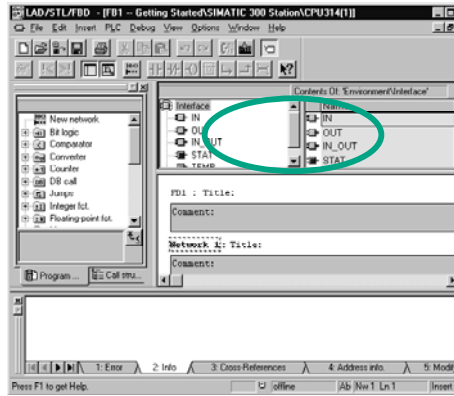
## 5.2 用梯形图编程 FB1

我们将向您说明如何编程一个功能块，在本例中，该功能块使用两个不同的数据块控制和监视汽油或柴油发动机。

所有“发动机特定的”信号都是作为块参数从组织块传送给功能块的，因此必须作为输入和输出参数在变量声明表中列出(声明“in”和“out”)。

我们假定您已经掌握了如何使用 STEP 7 输入一个串联电路、一个并联电路和一个存储器功能。

### 声明/定义变量



LAD/STL/FBD 编程窗口已经打开，并已激活选项视图 > LAD (编程语言)。

注意，FB1 现在显示在标题栏中，因为您是通过双击 FB1 打开的编程窗口。

变量声明区域由变量总览视图(左窗格)和变量详细视图(右窗格)组成。

在变量总览视图中，依次选择声明类型“IN”，“OUT”和“STAT”，并在相应的变量详细视图中输入如下声明。

在变量总览视图中，单击相应的单元并在随后出现的图中应用条目。您可以从所显示的下拉列表中选择数据类型。



Contents Of: 'Environment\Interface\IN'					
Name	Data Type	Address	Initial Value	Comment	
Switch_On	Bool	0.0	FALSE	Switch on engine	
Switch_Off	Bool	0.1	FALSE	Switch off engine	
Failure	Bool	0.2	FALSE	Engine failure, causes the engine to switch off	
Actual_Speed	Int	2.0	0	Actual engine speed	

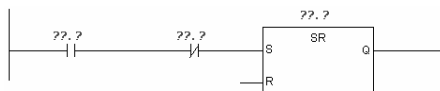
Contents Of: 'Environment\Interface\OUT'					
Name	Data Type	Address	Initial Value	Comment	
Engine_On	Bool	4.0	FALSE	Engine is switched on	
Preset_Speed_Reached	Bool	4.1	FALSE	Preset speed reached	

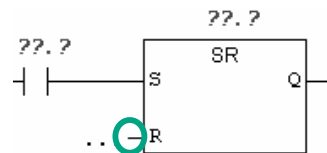
Contents Of: 'Environment\Interface\STAT'					
Name	Data Type	Address	Initial Value	Comment	
Preset_Speed	Int	6.0	1500	Requested engine speed	

只有字母、数字和下划线是变量声明表中的块参数名称所允许使用的字符。  
如果在变量详细视图中没有显示所有需要的栏，您可以通过快捷菜单来显示(使用鼠标右击)。

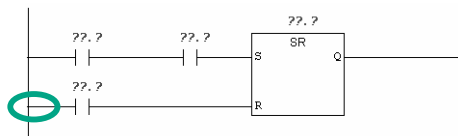
### 编程一个发动机的开机和停机



使用工具栏中相应的按钮或编程元素目录在程序段 1 中依次插入一个常开触点、一个常闭触点和一个 SR 元素。



然后在输入 R 之前选择电流通路。

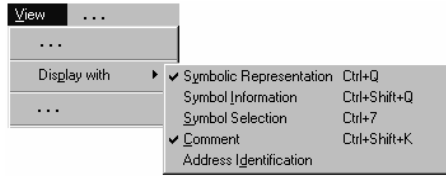


插入另一个常开触点。在该触点前选择电流通路。



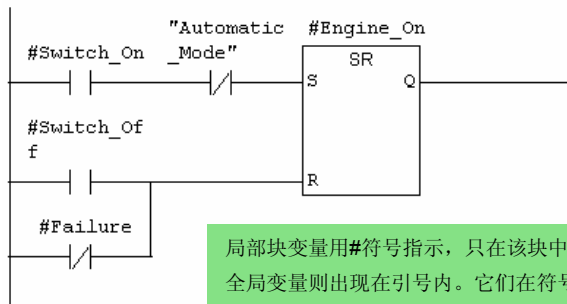
插入一个与常开触点并联的常闭触点。





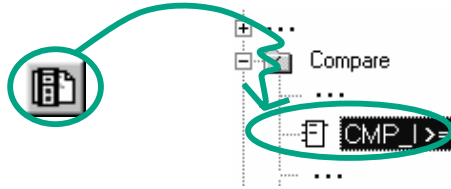
检查符号表达式是否已激活。

选中问号并输入变量声明表中相应的名称(自动分配符号#)。为串联电路中的常闭触点输入符号名“Automatic\_Mode”。然后保存程序。



局部块变量用#符号指示，只在该块中有效。  
全局变量则出现在引号内。它们在符号表中定义，在整个程序内都有效。  
信号状态“Automatic\_Mode”由OB1中(请参见4-7页程序段3)的另一个SR元素定义，现在由FB1查询。

### 编程速度监视

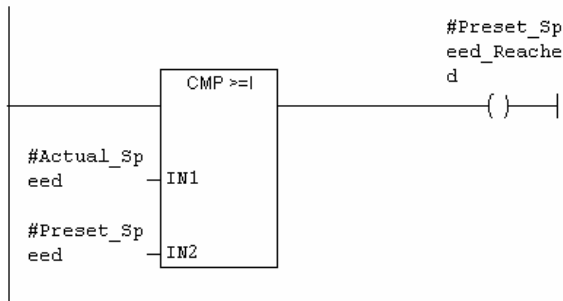


插入一个新的程序段并选择电流通路。  
然后在编程元素目录中浏览直至找到比较功能并插入 **CMP >= I**。



另外在电流通路中插入一个线圈。

再次选择问号，并使用变量声明表中的名称标定线圈和比较器。  
然后保存程序。



#### 何时开动和关停发动机？

当变量 **#Switch\_On** 的信号状态为“1”并且变量“**Automatic\_Mode**”的信号状态为“0”时，开动发动机。只有当对“**Automatic\_Mode**”取反时(常闭触点)，才能够启用该功能。  
当变量 **#Switch\_Off** 的信号状态为“1”或变量 **#Fault** 的信号状态为“0”时，发动机关闭。同样，可以通过取反 **#Fault** 实现该功能(**#Fault** 是一个“0 激活”信号，它在常态下的信号为“1”，如果出现故障则为“0”)。

#### 比较器如何监视发动机速度？

比较器比较变量 **#Actual\_Speed** 和 **#Setpoint\_Speed**，并将结果赋值给 **#Setpoint\_Speed\_Reached** (信号状态“1”)。

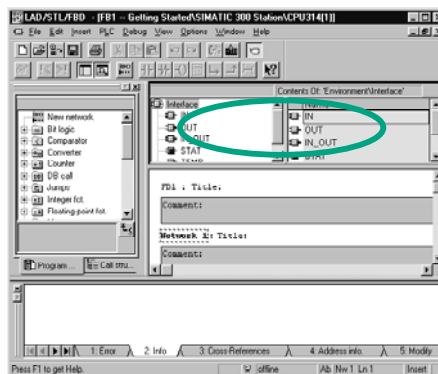
在帮助 > 目录下主题“编程块”，“创建逻辑块”和“编辑变量声明”或“编辑 LAD 指令”中可以找到更多的信息。

## 5.3 用语句表编程 FB1

我们将向您介绍如何编程一个功能块，在本例中，该功能块使用两个不同的数据块控制和监视汽油或柴油发动机。

所有“发动机特定的”信号都是作为块参数从组织块传送给功能块的，因此必须作为输入和输出参数在变量声明表中列出(声明“in”和“out”)。

### 声明/定义变量



LAD/STL/FBD 编程窗口已经打开，并已激活选项视图 > STL (编程语言)。

注意，FB1 现在显示在标题栏中，因为您是通过双击 FB1 打开的编程窗口。

变量声明区域由变量总览视图(左窗格)和变量详细视图(右窗格)组成。

在变量总览视图中，依次选取声明类型“IN”，“OUT”和“STAT”，并在相应的变量详细视图中输入如下声明。

在变量总览视图中，单击相应的单元并在随后出现的图中应用条目。您可以从所显示的下拉列表中选择数据类型。



Contents Of: 'Environment\Interface\IN'					
Name	Data Type	Address	Initial Value	Comment	
Switch_On	Bool	0.0	FALSE	Switch on engine	
Switch_Off	Bool	0.1	FALSE	Switch off engine	
Failure	Bool	0.2	FALSE	Engine failure, causes the engine to switch off	
Actual_Speed	Int	2.0	0	Actual engine speed	

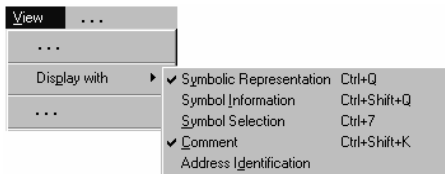
Contents Of: 'Environment\Interface\OUT'					
Name	Data Type	Address	Initial Value	Comment	
Engine_On	Bool	4.0	FALSE	Engine is switched on	
Preset_Speed_Reached	Bool	4.1	FALSE	Preset speed reached	

Contents Of: 'Environment\Interface\STAT'					
Name	Data Type	Address	Initial Value	Comment	
Preset_Speed	Int	6.0	1500	Requested engine speed	

只有字母、数字和下划线这些字符是变量声明表中的块参数名称所允许使用的字符。

### 编程发动机的开动和关停



检查符号表达式是否已激活。

```

A      #Switch_On
AN    "Automatic_Mode"
S      #Engine_On
O      #Switch_Off
ON    #Failure
R      #Engine_On
    
```

在程序段 1 中输入相应的指令。

局部块变量用#符号指示，只在该块中有效。  
 全局变量则出现在引号内。它们在符号表中定义，在整个程序内都有效。  
 信号状态“Automatic\_Mode”由 OB1 中(请参见 4-10 页程序段 3)的另一个 SR 元素定义，现在由 FB1 查询。

## 编程速度监视

```
L   #Actual_Speed
L   #Preset_Speed
>=I
=   #Preset_Speed_Reached
```

插入一个新的程序段并输入相应的指令。然后保存程序。



### 何时开动和关停发动机？

当变量#Switch\_On的信号状态为“1”并且变量“Automatic\_Mode”的信号状态为“0”时，发动机开动。只有当对“Automatic\_Mode”取反时(常闭触点)，才能够启用该功能。

当变量#Switch\_Off的信号状态为“1”或变量#Fault的信号状态为“0”时，发动机关闭。同样，可以通过取反#Fault实现该功能(#Fault是一个“0激活”信号，它在常态下的信号为“1”，如果出现故障则为“0”)。

### 比较器如何监视发动机速度？

比较器比较变量#Actual\_Speed和#Setpoint\_Speed，并将结果赋值给#Setpoint\_Speed\_Reached(信号状态“1”)。

在帮助 > 目录下主题“编程块”，“创建逻辑块”和“编辑变量声明表”或者“编辑 STL 声明”中可以找到更多的信息。

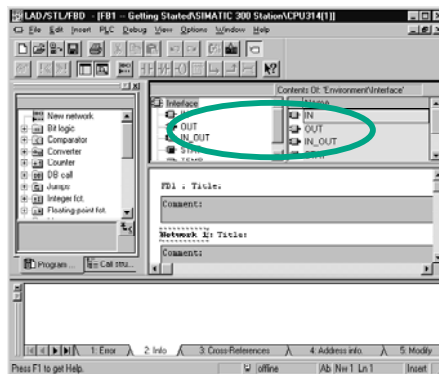
## 5.4 用功能块图编程 FB1

我们将向您介绍如何编程一个功能块，在本例中，该功能块使用两个不同的数据块控制和监视汽油或柴油发动机。

所有“发动机特定的”信号都是作为块参数从组织块传送给功能块的，因此必须作为输入和输出参数在变量声明表中列出(声明“in”和“out”)。

我们假定您已经掌握了如何使用 STEP 7 输入 AND 功能、OR 功能和存储器指令。

### 声明/定义变量



LAD/STL/FBD 编程窗口已经打开，并已激活选项视图 > FBD (编程语言)。

注意，FB1 现在会显示在标题栏中，因为您是通过双击 FB1 打开的编程窗口。

变量声明区域由变量总览视图(左窗格)和变量详细视图(右窗格)组成。

在变量总览视图中，依次选取声明类型“IN”、“OUT”和“STAT”，并在相应的变量详细视图中输入如下声明。

在变量总览视图中，单击相应的单元并在随后出现的图中应用条目。您可以从所显示的下拉列表中选择数据类型。



Contents Of: 'Environment\Interface\IN'

Name	Data Type	Address	Initial Value	Comment
Switch_On	Bool	0.0	FALSE	Switch on engine
Switch_Off	Bool	0.1	FALSE	Switch off engine
Failure	Bool	0.2	FALSE	Engine failure, causes the engine to switch off
Actual_Speed	Int	2.0	0	Actual engine speed

Contents Of: 'Environment\Interface\OUT'

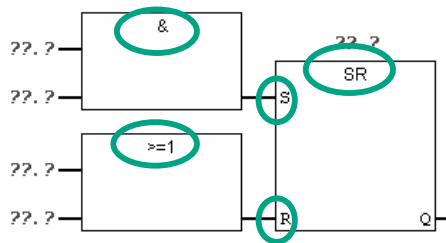
Name	Data Type	Address	Initial Value	Comment
Engine_On	Bool	4.0	FALSE	Engine is switched on
Preset_Speed_Reached	Bool	4.1	FALSE	Preset speed reached

Contents Of: 'Environment\Interface\STAT'

Name	Data Type	Address	Initial Value	Comment
Preset_Speed	Int	6.0	1500	Requested engine speed

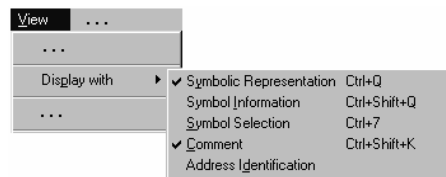
局部块变量用#符号指示，只在该块中有效。  
全局变量则出现在引号内。它们在符号表中定义，在整个程序内都有效。

编程一个发动机的开动和关停



在程序段 1 中用编程元素目录(位逻辑文件夹)插入一个 SR 功能。

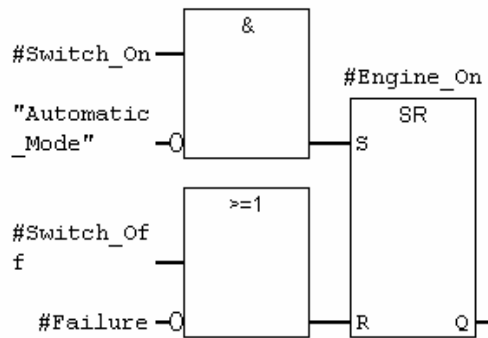
在 S (置位)输入端添加一个 AND 逻辑框，在 R (复位)输入端添加一个 OR 逻辑框。



检查符号表达式是否已激活。



请单击`???`符号并输入变量声明表中相应的名称(自动分配符号`#`)。  
确认 AND 功能的一个输入端的地址是符号名“Automatic\_Mode”。  
用工具栏中相应的按钮对输入“Automatic\_Mode”和`#Fault`取反。  
然后保存程序。

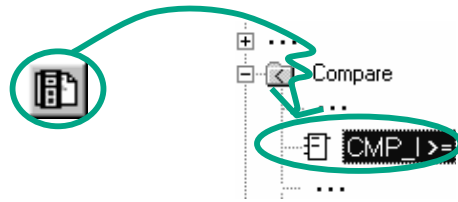


局部块变量用`#`符号指示，只在该块中有效。  
全局变量则出现在引号内。它们在符号表中定义，在整个程序内都有效。  
信号状态“Automatic\_Mode”由 OB1 中(请参见 4-14 页程序段 3)的另一个 SR 元素定义，现在由 FB1 查询。





### 编程速度监视

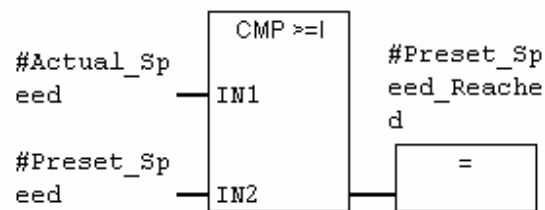


插入一个新的程序段并选择输入域。

然后在编程元素目录中查找到比较功能并插入一个 **CMP >= I**。

在比较器后面附上一个输出赋值，用变量声明表中的名称作为输入的地址。

然后保存程序。



#### 如何开动和关停发动机？

当变量 **#Switch\_On** 的信号状态为 “1” 并且变量 “**Automatic\_Mode**” 的信号状态为 “0” 时，发动机开动。只有当对 “**Automatic\_Mode**” 取反时(常闭触点)，才能够启用该功能。

当变量 **#Switch\_Off** 的信号状态为 “1” 或变量 **#Fault** 的信号状态为 “0” 时，发动机停机。同样，可以通过取反 **#Fault** 实现该功能(**#Fault** 是一个 “0 激活” 信号，它在常态下的信号为 “1”，如果出现故障则为 “0”)。

#### 比较器如何监视发动机速度？

比较器比较变量 **#Actual\_Speed** 和 **#Setpoint\_Speed**，并将结果赋值给 **#Setpoint\_Speed\_Reached** (信号状态 “1”)。

在帮助 > 目录下主题 “编程块”，“创建逻辑块” 和 “编辑变量声明” 或者 “编辑 FBD 指令” 中可以找到更多的信息。

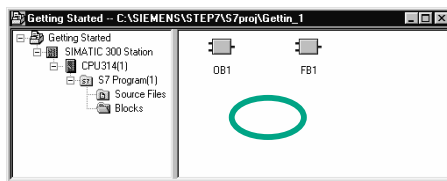
## 5.5 生成背景数据块和修改实际值

您已经编写了功能块 FB1 (“Engine”)并且还在变量声明表中定义了发动机特定的参数。

为了以后能在 OB1 中编写指令调用此功能块，必须生成相应的数据块。一个背景数据块(DB)总是被指定给一个功能块。

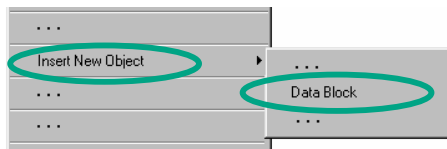
这个功能块用于控制和监视一台汽油或柴油发动机。不同的发动机的预设速度分别存储在两个数据块中，可在其中修改实际值(#Setpoint\_Speed)。

通过一次性集中编写功能块，可以减少相关的编程量。

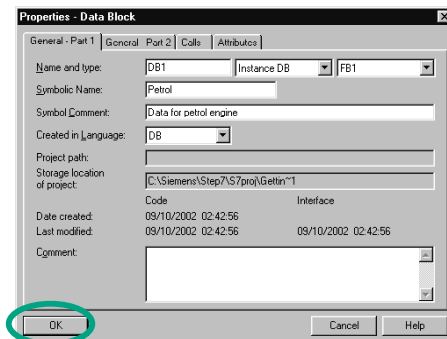


在 SIMATIC 管理器中打开项目“Getting Started”。

查找到 **Blocks** 文件夹并用鼠标右击右窗口。



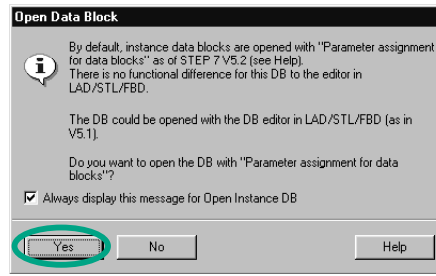
右击鼠标，使用弹出菜单插入一个数据块。



在“数据块属性”对话框中使用名称 DB1，然后在相邻的下拉列表中选择应用程序“背景 DB”，并应用所分配的功能块名“FB1”。确认“属性”对话框中的所有设置。

数据块 DB1 被添加到“Getting Started”项目中。

双击打开 DB1。



Address	Declaration	Name	Type	Initial value	Actual value	Comment
1	0.0 in	Switch_On	BOOL	FALSE	FALSE	Switch on engine
2	0.1 in	Switch_Off	BOOL	FALSE	FALSE	Switch off engine
3	0.2 in	Failure	BOOL	FALSE	FALSE	Engine failure, causes the engine to switch off
4	2.0 in	Actual_Speed	INT	0	0	Actual engine speed
5	4.0 out	Engine_On	BOOL	FALSE	FALSE	Engine is switched on
6	4.1 out	Present_Speed_Reached	BOOL	FALSE	FALSE	Present speed reached
7	6.0 int	request_speed	INT	1500	1500	Requested engine speed

Address	Declaration	Name	Type	Initial value	Actual value	Comment
1	0.0 in	Switch_On	BOOL	FALSE	FALSE	Switch on engine
2	0.1 in	Switch_Off	BOOL	FALSE	FALSE	Switch off engine
3	0.2 in	Failure	BOOL	FALSE	FALSE	Engine failure, causes the engine to switch off
4	2.0 in	Actual_Speed	INT	0	0	Actual engine speed
5	4.0 out	Engine_On	BOOL	FALSE	FALSE	Engine is switched on
6	4.1 out	Present_Speed_Reached	BOOL	FALSE	FALSE	Present speed reached
7	6.0 int	request_speed	INT	1200	1200	Requested engine speed

单击是确认随后出现的对话框，可将参数分配给背景数据块。

接着在“实际值”栏中为汽油机输入数值“1500”（在“Setpoint\_Speed”行中）。现在您已经为该发动机定义了最大速度。

保存 DB1，并关闭编程窗口。

按相同的方法，为 FB1 生成另一个数据块 DB2。

现在为柴油机输入实际值“1200”。

保存 DB2，并关闭编程窗口。

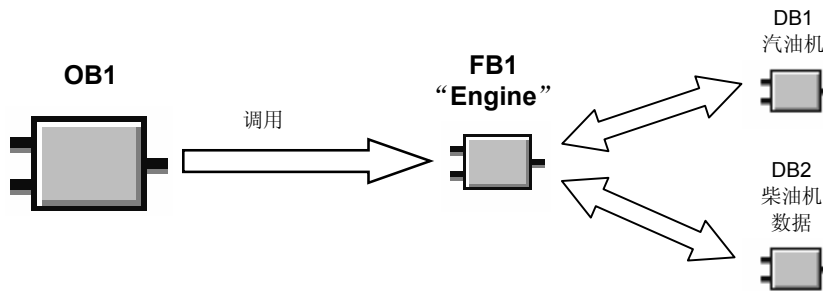
通过修改实际值，您已经完成用一个功能块控制两个发动机的准备工作。要控制更多的发动机，您所要做的就是生成其它的数据块。

您要做的下一件事就是在 OB1 中编程来调用功能块。为此，根据所选择的编程语言，继续阅读第 5.6 节(使用梯形图编程)、第 5.7 节(使用语句表编程)或第 5.8 节(使用功能块图编程)。

在帮助 > 目录的主题“编程块”和“创建数据块”中可以找到更多的信息。

## 5.6 用梯形图编程块调用

为编程功能块所做的所有工作，只有当您在 OB1 中调用该功能块时才有用处。一个功能块调用使用一个数据块，这样两个发动机您都可以进行控制。



SIMATIC 管理器随着项目 “Getting Started” 一起打开。

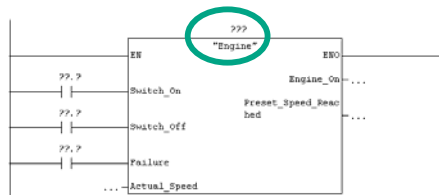
查找到 **Blocks** 文件夹并打开 **OB1**。



选择程序段 3，然后在 LAD/STL/FBD 编程窗口插入程序段 4。

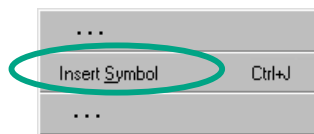


在编程元素目录中找到 **FB1**，并双击将其插入。



在以下各项前面插入一个常开触点：  
**Switch\_On**、**Switch\_Off** 和 **Fault**。

单击 “Engine” 上面的 **???** 符号，然后将光标保持在同一位置，用鼠标右击输入框。



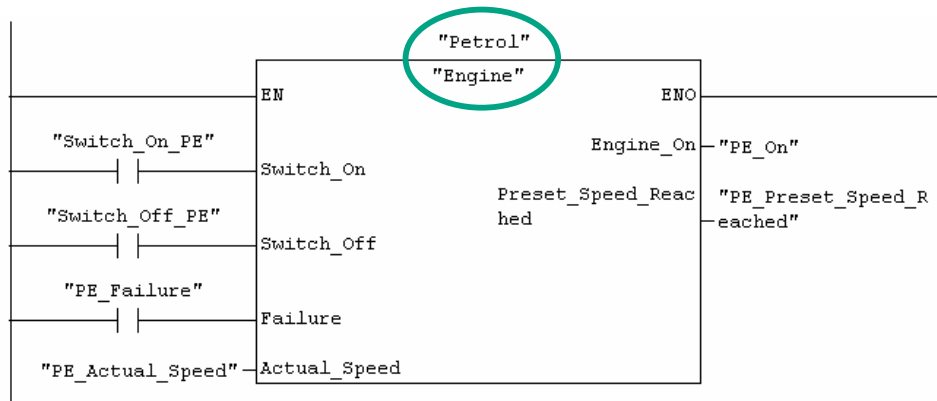
右击鼠标，在显示的快捷菜单中单击 **插入符号**。将会出现一个下拉列表。



Petrol	FB	1	DB	1
				Data for petrol engine

双击数据块 **Petrol**。然后该块则自动被输入到输入框中，并加上引号。

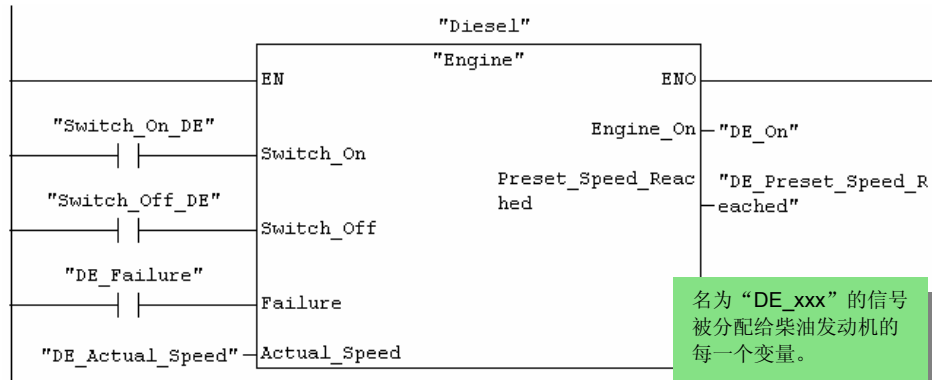
单击问号，然后使用下拉列表中的相应符号名输入到引号中，为功能块中的其它参数输入地址。



发动机特定的输入和输出变量(声明“in”和“out”)显示在FB“Engine”中。  
名为“PE\_xxx”的信号将被分配给汽油发动机的每一个变量。



在一个新程序段中，使用下拉列表中的相应的地址，对带有数据块“Diesel” (DB2) 的功能块“Engine” (FB1)的调用进行编程。



保存程序并关闭块。

当您创建一个具有组织块、功能块和数据块的程序结构时，必须在体系中子块(如 FB1)上一级块中(如，OB1)对子块的调用指令进行编程。这个过程都是相同的。

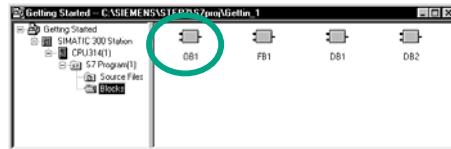
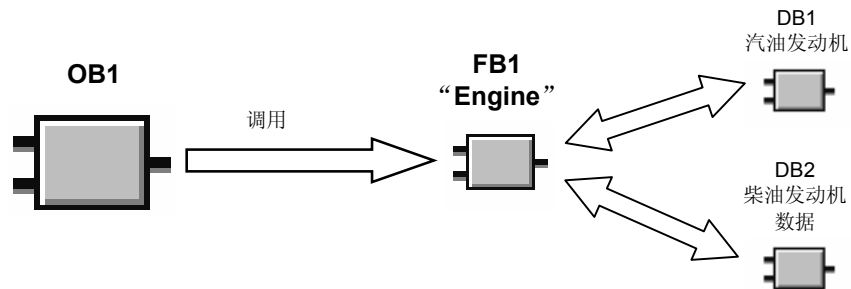
您还可以在符号表中输出各个块的符号名(例如，FB1 的名称为“Engine”，DB1 的名称为“Petrol”)。

您可以随时归档或者打印编程的块。对应的功能可以在 **SIMATIC 管理器**下的菜单命令**文件 > 归档或文件 > 打印**中找到。

在**帮助 > 目录**下的主题“调用参考帮助”、“语言描述：LAD”和“程序控制指令”中可以找到更多的信息。

## 5.7 用语句表编程块调用

为编程功能块所做的所有工作，只有当您在 **OB1** 中调用该功能块时才有用处。每个功能块调用使用一个数据块，这样两个发动机您都可以进行控制。



SIMATIC 管理器随着项目 “Getting Started” 一起打开。

查找到 **Blocks** 文件夹并打开 **OB1**。



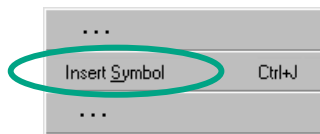
选择程序段 3，然后在 LAD/STL/FBD 编程窗口中插入程序段 4。

```
CALL "Engine", "Petrol"
Switch_On      :=
Switch_Off     :=
Failure        :=
Actual_Speed   :=
Engine_On      :=
Preset_Speed_Reached:=
```

在代码段输入 **CALL "Engine"**，  
**"Petrol"**，然后按下回车键。

将显示出功能块 **"Petrol"** 的所有参数。

将光标放在 **Switch\_On** 的等号后然后按下鼠标右键。



单击鼠标右键，在显示的快捷菜单中单击**输入符号**。将会出现一个下拉列表。



OB1_SCAN_1	Byte	1.0
PE_Actual_Speed	INT	MW 2
PE_Failure	BOOL	I 1.2
PE_Fan_On	BOOL	Q 5.2
PE_Follow_On	TIMER	T 1
PE_On	BOOL	Q 5.0
PE_Preset_Speed_Reached	BOOL	Q 5.1
Petrol	FB 1	DB 1
Red_Light	BOOL	Q 4.1
Switch_Off_DE	BOOL	I 1.5
Switch_Off_PE	BOOL	I 1.1
Switch_On_DE	BOOL	I 1.4
Switch_On PE	BOOL	I 1.0

单击名称 **Switch\_On\_PE**。将从下拉列表选取该名称并自动将其添加到引号内。

```
CALL "Engine" , "Petrol"
Switch_On      := "Switch_On_PE"
Switch_Off     := "Switch_Off_PE"
Failure       := "PE_Failure"
Actual_Speed  := "PE_Actual_Speed"
Engine_On     := "PE_On"
Preset_Speed_Reached := "PE_Preset_Speed_Reached"
```

用下拉列表中相应的符号名为功能块的变量分配所有需要的地址。

名为“PE\_xxx”的信号被分配给汽油发动机的每一个变量。

```
CALL "Engine" , "Diesel"
Switch_On      := "Switch_On_DE"
Switch_Off     := "Switch_Off_DE"
Failure       := "DE_Failure"
Actual_Speed  := "DE_Actual_Speed"
Engine_On     := "DE_On"
Preset_Speed_Reached := "DE_Preset_Speed_Reached"
```

在一个新的程序段中，编程对带数据块“Diesel” (DB2)的功能块“Engine” (FB1)的调用。按同样的方法，完成其它调用。



保存程序并关闭块。

当您创建一个具有组织块、功能块和数据块的程序结构时，必须在分级结构中的高级块中(如，OB1)编写子程序块(如FB1)的调用。这个过程都是相同的。

您还可以在符号表中输出各个块的符号名(例如，FB1的名称为“Engine”、DB1的名称为“Petrol”)。

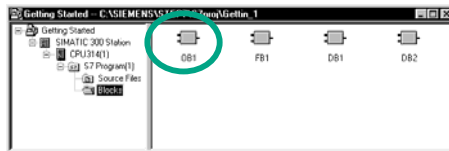
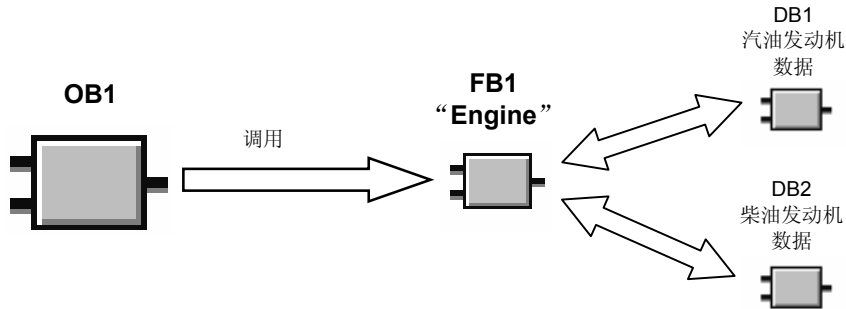
您可以随时归档或者打印编程的块。对应的功能可以在 **SIMATIC 管理器**下的菜单命令**文件 > 归档或文件 > 打印**中找到。

在**帮助 > 目录**下的主题“调用参考帮助”、“语言描述：STL”和“程序控制指令”中可以找到更多的信息。



## 5.8 用功能块图编程块调用

为编程功能块所做的所有工作，只有当您在 **OB1** 中调用该功能块时才有用处。每个功能块的调用都使用一个数据块，这样两个发动机您都可以进行控制。

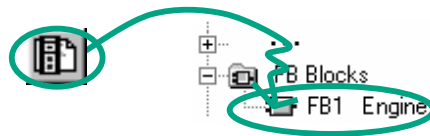


SIMATIC 管理器随着项目 “Getting Started” 一起打开。

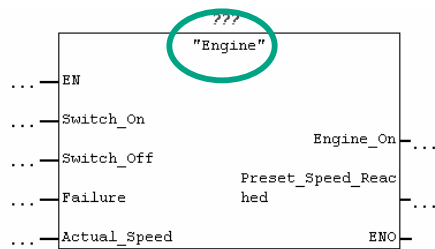
查找到 **Blocks** 文件夹并打开 **OB1**。



选择程序段 3，然后在 LAD/STL/FBD 编程窗口插入程序段 4。

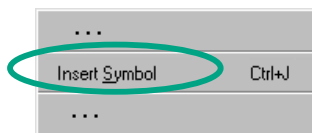


在编程元素目录中查找到 **FB1**，然后插入该块。




发动机特定的所有输入和输出变量都将显示。

单击 “Engine” 上的 ??? 符号，然后将光标保持在同一位置，用鼠标右击输入框。



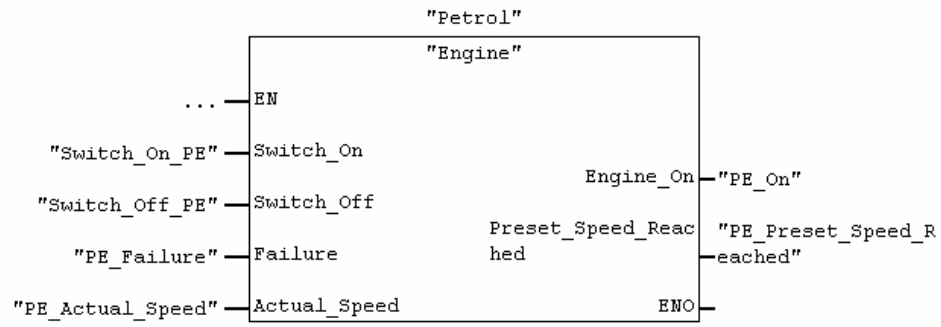
单击鼠标右键，在显示的快捷菜单中单击 **插入符号**。显示一个下拉列表。



 Petrol	FB	1	DB	1
				Data for petrol engine

双击数据块 **Petrol**。将从下拉列表中将  
其取出并自动地加在引号内输入到输入  
框中。

用下拉列表中相应的符号名为功能块的所有其它参数分配地址。

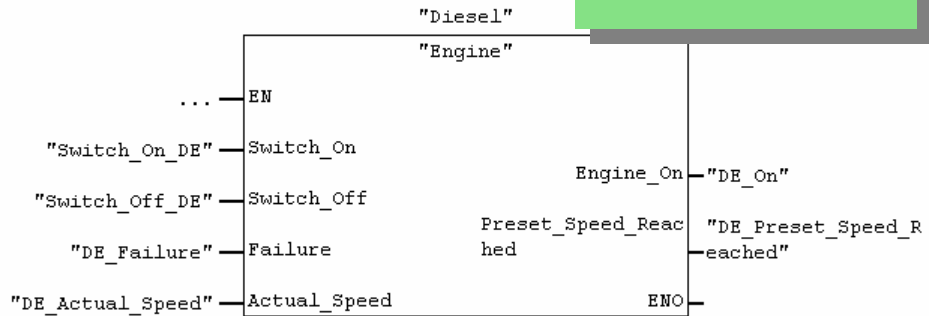


名为“PE\_xxx”的信号被分配  
给汽油机的每一个变量。





在一个新程序段中，使用下拉列表中的相应的地址，对带有数据块“Diesel” (DB2) 的功能块“Engine” (FB1)的调用进行编程。



名为“DE\_xxx”的信号被分配给柴油机的每一个变量。



保存程序并关闭块。

当您创建一个具有组织块、功能块和数据块的程序结构时，必须在分级结构的高一级块中(如，OB1)编写对子程序块(如FB1)的调用。这个过程都是相同的。

您还可以在符号表中输出各个块的符号名(例如，FB1的名称为“Engine”、DB1的名称为“Petrol”)。

您可以随时归档或者打印编程的块。对应的功能可以在 SIMATIC 管理器下的菜单命令文件 > 归档或文件 > 打印中找到。

在帮助 > 目录下的主题“调用参考帮助”、“语言描述：FBD”和“程序控制指令”中可以找到更多的信息。

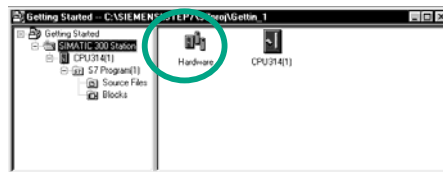


## 6 配置中央机架

### 6.1 配置硬件

一旦您创建了一个带有 SIMATIC 站的项目，就可以配置硬件了。在第 2.1 节中用 STEP 7 向导创建的项目结构完全能够满足这些要求。

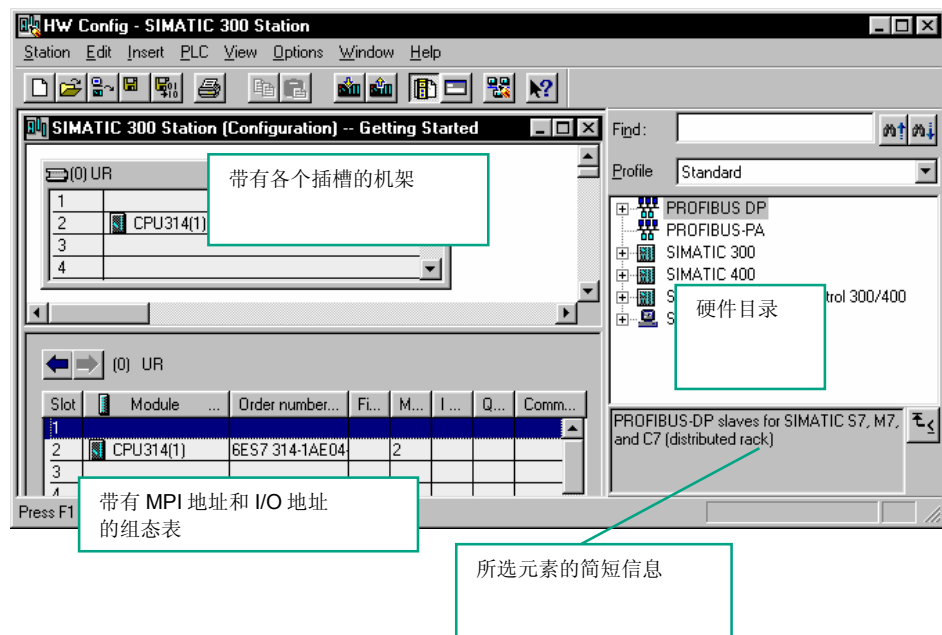
使用 STEP 7 对硬件进行配置。这些配置的数据以后可以通过“下载”(请参见第 7 章)传送到可编程控制器。

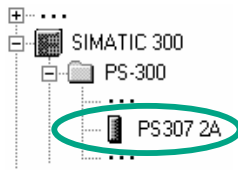


假设已打开了 SIMATIC 管理器和“Getting Started”项目。

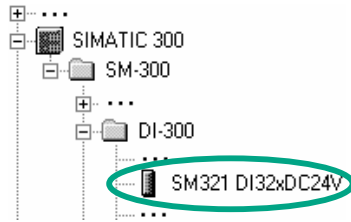
打开 **SIMATIC 300 站** 文件夹，并双击硬件符号。

“HW Config”窗口打开。您在创建项目时所选择的 CPU 将显示出来。对于“Getting Started”项目，是 CPU 314。

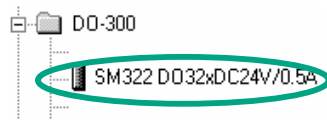




首先您需要一个电源模块。在目录中找到 **PS307 2A**，然后将该模块拖放到插槽 1。



查找到输入模块(DI, 数字输入) **SM321 DI32xDC24V**，将其插入到插槽 4。插槽 3 保留为空。

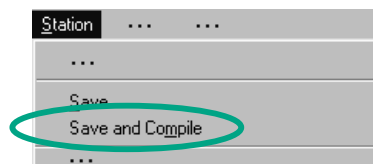


按照相同的方法，在插槽 5 插入输出模块 **SM322 DO32xDC24V/0.5A**。

要在一个项目中修改模块的参数(例如，地址)，请双击该模块。但是，您应该只在确信知道改变这些参数会对可编程控制器有何影响时方可改变它们。

对于“Getting Started”项目，不需改变任何参数。

Slot	Module	Order Number	MPI Address	I Add...	Q...	Comment
1	PS 307 2A	6ES7 307-1BA00-0AA0				
2	CPU314(1)	6ES7 314-1AE04-0AB0	2			
3						
4	DI32xDC24V	6ES7 321-1BL00-0AA0		0...3		
5	DO32xDC24V/0.5A	6ES7 322-1BL00-0AA0			4..7	
6						
7						
8						
9						
10						
11						



使用菜单命令 **保存和编译** 为向 CPU 传送准备好的数据。

一旦关闭“HW Config”应用程序，在 **Blocks** 文件夹中将会出现系统数据的符号。

使用菜单命令 **站 > 一致性检查** 还可以检查组态错误。对任何可能出现的错误，**STEP 7** 都为您提供了解决方案。

在 **帮助 > 目录** 下的主题“配置硬件”和“配置中央机架”中可以找到更多的信息。

## 7 下载和调试程序

### 7.1 建立一个在线连接

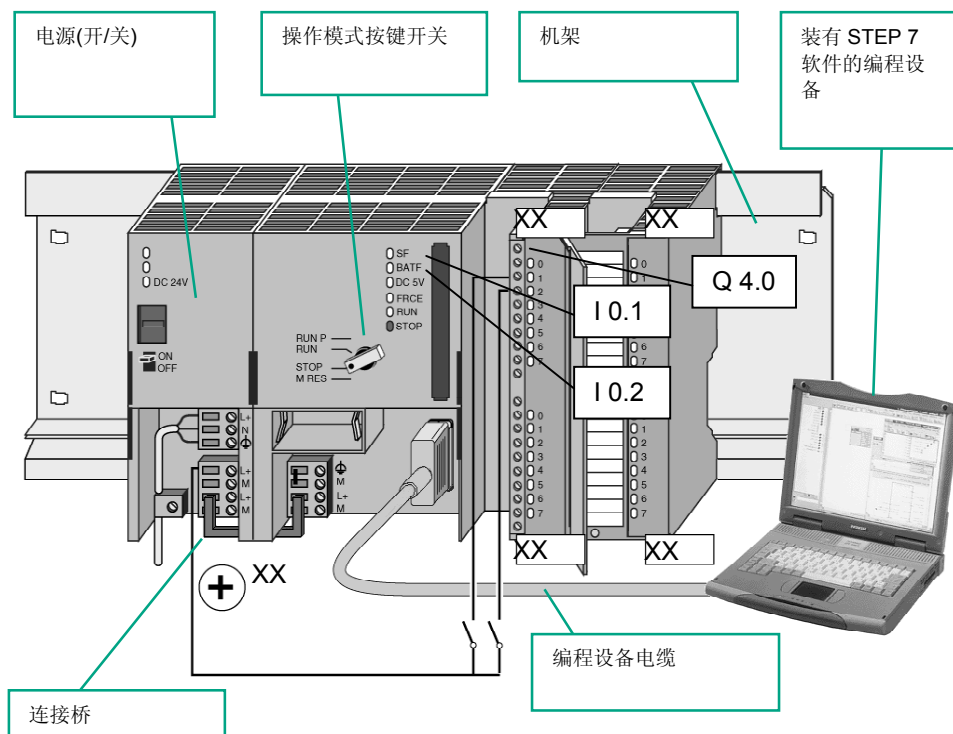
使用提供的项目“GS-LAD\_Example”或者已经创建的项目“Getting Started”和一个简单的测试组态，我们将向您显示如何将程序下载到可编程逻辑控制器(PLC)，然后如何调试它。

您应该已经：

- 为“Getting Started”项目配置了硬件(请参见第 6 章)
- 按照安装手册的说明设置了硬件

串联电路(AND)的：

只有当键 I 0.1 和 I 0.2 都按下时，输出 Q 4.0 才点亮(数字输出模块上的二极管 Q 4.0 点亮)。用接线与您的 CPU 建立如下的测试组态。





### 配置硬件

要将模块组装到导轨，可以按照如下的操作顺序进行：

- 将模块与总线连接器连接
- 将模块挂在导轨上并向下摆动
- 将模块旋紧就位
- 组装其余的模块
- 一旦完成所有模块的组装后，请将钥匙开关插在 CPU 上



即使您所使用的硬件与图示不同，也仍然可以进行测试。只要使地址与输入和输出相符就可以。

STEP 7 为您提供了多种方法来进行程序调试；例如，使用程序状态或通过变量表。

在手册“S7-300, 硬件和安装/模块技术规范”和“S7-400/M7-400-硬件”中可以找到更多的关于配置中央机架的信息。



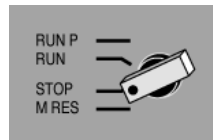
## 7.2 下载程序到可编程控制器

您应该已经建立了一个在线连接，以便下载程序。

### 接通电源

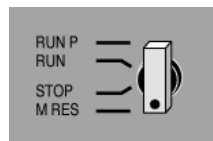


使用 ON/OFF 开关接通电源。CPU 上的二极管“DC 5V”将点亮。



将操作模式开关转到 **STOP** 位置(如果还没有在 **STOP** 位置的话)。红色的“STOP”发光二极管将点亮。

### 复位 CPU 并切换到 RUN



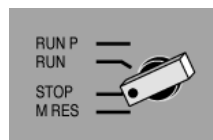
将操作模式开关转换到 **MRES** 位置并保持至少 3 秒钟，直到红色的“STOP”发光二极管开始慢闪为止。

请释放开关，并且最多在 3 秒内将开关再次转到 **MRES** 位置。当“STOP”LED 快闪时，CPU 已经被复位。

如果“STOP”发光二极管没有开始快闪，请重复执行此过程。

存储器复位功能将删除 CPU 上的所有数据。然后 CPU 就会进入初始状态。

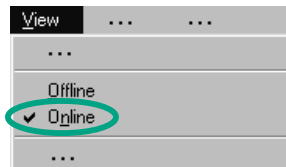
### 将程序下载到 CPU



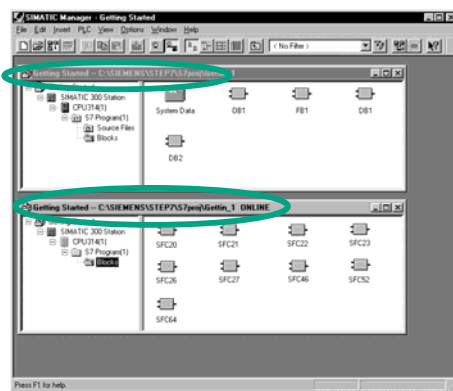
现在将操作模式开关重新切换到“STOP”位置，以便下载程序。



启动 SIMATIC 管理器，在“打开”对话框中打开“Getting Started”项目(如果该项目还没有打开的话)。



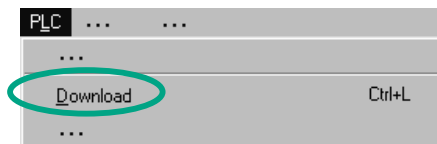
除了“Getting Started 离线”窗口外，也打开“Getting Started 在线”窗口。在线或者离线状态通过不同颜色的标题栏加以区分。



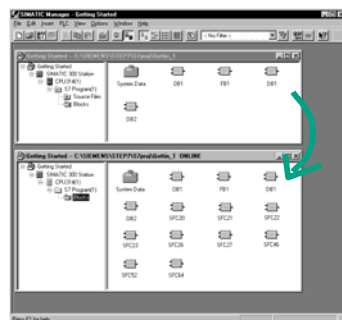
在两个窗口中查找到 **Blocks** 文件夹。

离线窗口显示编程设备上的情形；在线窗口则显示 CPU 上的情形。

即使执行了存储器复位，系统功能(SFC)也会保留在 CPU 中。CPU 提供操作系统的这些功能。无须下载这些功能，但也无法删除它们。



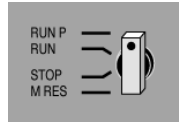
在离线窗口中选择 **Blocks** 文件夹，然后用菜单命令 **PLC > 下载** 将程序下载到 CPU。使用 **确定** 对提示信息进行确认。



当完成下载后，这些程序块就显示在在线窗口中。

您还可以使用工具栏中的相关按钮或者通过鼠标右键的弹出式菜单来调用菜单命令 **PLC > 下载**。

## 接通 CPU 并检查操作模式



将操作开关转到 **RUN-P** 位置。绿色的“RUN”发光二极管点亮，而红色的“STOP”发光二极管熄灭。CPU 的操作准备工作就绪

当绿色发光二极管点亮时，您就可以开始程序测试工作。

如果红色的发光二极管仍然处于点亮状态，就说明有错误出现。您需要通过评估诊断缓冲区来诊断错误。

### 下载单个块

在实际使用过程中，为了对错误作出快速的反应，可以使用拖放功能将块一个一个传送到 CPU 中。

在下载块时，CPU 上的操作模式开关必须在“RUN-P”或者“STOP”模式下。在“RUN-P”模式下下载的块被立即被启动。因此，您必须要记住如下注意事项：

- 如果没有错误的块被错误块重写，则将导致系统故障。要避免这样的情况发生，您可以在下载块之前对它们进行测试。
- 如果您没有按照一定的顺序下载块 – 首先是子程序块，然后是更高级的块 – CPU 将进入“STOP”模式。您可以将整个程序下载到 CPU，从而避免这样的情况发生。

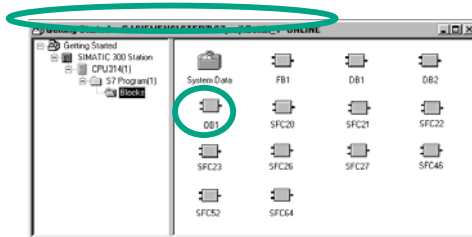
### 在线编程

在实际使用过程中，通常为了进行测试，可能需要修改已经下载到 CPU 的块。如果要进行修改，请在在线窗口中双击所需要的块，以打开 LAD/STL/FBD 编程窗口。然后按照常规方法编程该块。请注意，这个已编程的块将会立即在你的 CPU 中生效。

在帮助 > 目录下的主题“下载与上传”和“建立在线连接并进行 CPU 设置”中可以找到更多信息。

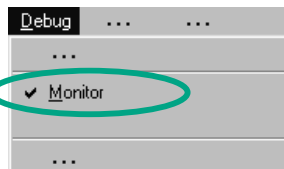
### 7.3 用程序状态测试程序

使用程序状态功能，可以在一个块中测试程序。要实现这一功能的前提是：您已经建立了与 CPU 的在线连接，该 CPU 处于 RUN 模式或 RUN-P 模式，并且程序已经下载。



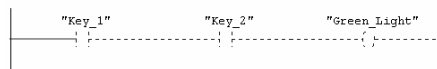
在项目窗口“Getting Started 在线”中打开 **OB1**。

打开 LAD/STL/FBD 编程窗口



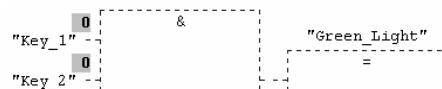
激活功能调试 > 监视。

#### 用梯形图进行调试



程序段 1 中的串联电路以梯形图的形式显示。当前支路一直到 Key 1 (I 0.1) 表示为一条实线；这表明正在为该电路供电。

#### 用功能块图进行调试



信号状态由“0”和“1”来指示。虚线表示没有逻辑运算结果。

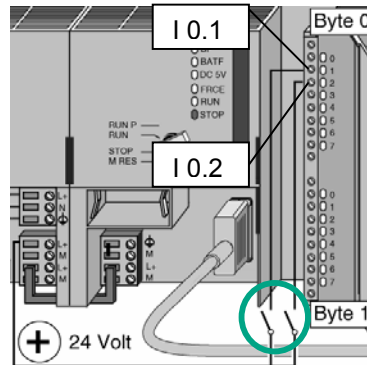
#### 用语句表进行调试

	RLO	STA	Standard
A "Key_1"	0	0	0
A "Key_2"	0	0	0
= "Green_Light"	0	0	0

在语句表中，以表格形式显示以下内容：

- 逻辑运算结果(RLO)
- 状态位(STA)
- 标准状态(STANDARD)

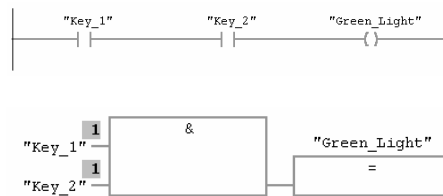
在测试过程中可以使用选项 > 自定义来改变编程语言的表达方式。



现在，将测试组态中的两个开关都按下。

在输入模块上的输入二极管 I 0.1 和 I 0.2 点亮。

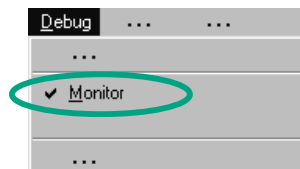
输出模块上的输出二极管 Q 4.0 点亮。



	RLO	STA	Standard
A "Key_1"	1	1	0
A "Key_2"	1	1	0
= "Green_Light"	1	1	0

在图形编程语言梯形图和功能块图中，您可以通过编程的程序段中颜色的变化来跟踪测试结果。这种颜色变化显示该点逻辑运算结果满足要求。

使用语句表编程语言，当逻辑运算结果满足要求时，STA 栏和 RLO 栏中的显示内容将发生变化。



释放调试 > 监视功能，并关闭窗口。

然后关闭 SIMATIC 管理器中的在线窗口。

我们建议您不要将扩展的程序全部下载到 CPU 中运行，因为，可能的错误源的数量过多会使错误诊断更加困难。所以，为了获得更好的总览，应该将块单个地下载并按顺序测试它们。

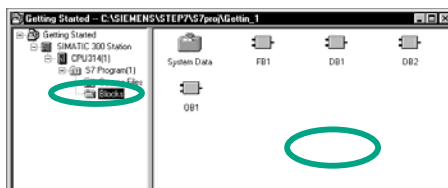
在帮助 > 目录的主题“调试”和“用程序状态测试”中您可以找到更多的信息。

### 7.3 用变量表测试程序

您可以通过监视和修改各个程序的变量来对它们进行测试。要实现这一功能的要求是：您已经建立了与 CPU 的在线连接，该 CPU 在 RUN-P 模式，并且程序已经下载。

和用程序状态测试一样，您可以在变量表中监视程序段 1 (串联电路或 AND 功能)中的输入和输出。您还可以通过预置实际速度测试 FB1 中用于发动机速度比较的比较器。

#### 创建变量表

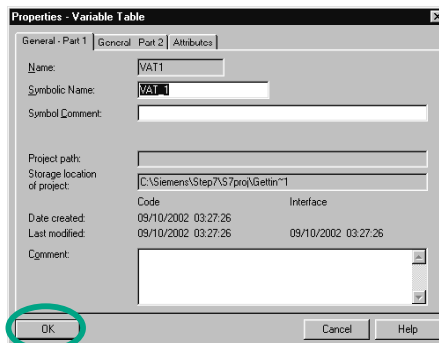


起始点还是在打开的 SIMATIC 管理器以及“Getting Started 离线”项目窗口。

查找到 **Blocks** 文件夹并用鼠标右键单击窗口右边。

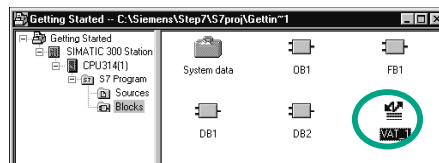


使用鼠标右键的弹出菜单插入一个变量表。



用确定关闭“属性”对话框，接受缺省设置。

或者，您可以为变量表分配一个符号名并输入符号注释。

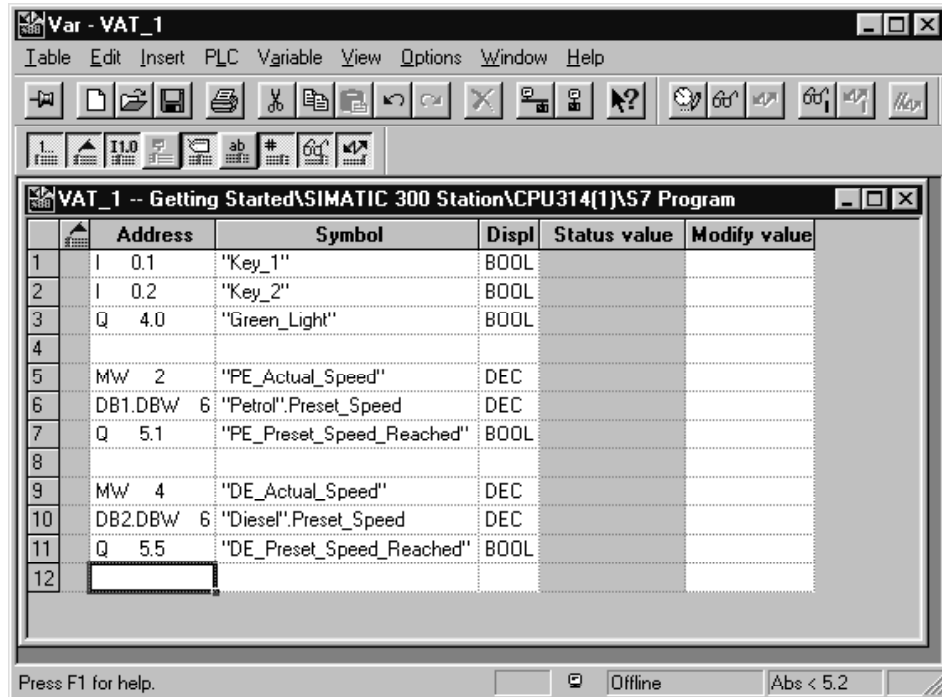


在 **Blocks** 文件夹中将创建一个 VAT1 (变量表)。

双击打开 **VAT1**；“监视和修改变量”窗口将打开。



变量表起初是空的。按照下面插图所示，为“Getting Started”示例输入符号名或地址。当用回车键完成输入项时，其余的详细资料会添加进来。

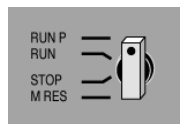


请保存变量表。

将变量表切换到在线方式



建立与已组态的 CPU 之间的连接。将会在状态栏中显示出 CPU 的操作模式。



将 CPU 的钥匙开关设置为 **RUN-P** (如果您尚未将其设置为 **RUN-P** 模式的话)。



### 监视变量



单击工具栏中的监视变量。

Address	Symbol	Display format	Status value	Modify value
I 0.1	"Key_1"	BOOL	true	
I 0.2	"Key_2"	BOOL	true	
Q 4.0	"Green_Light"	BOOL	true	
MW 2	"PE_Actual_Speed"	DEC	0	

按下测试组态中的 Key1 和 Key2，并监视变量表中的结果。

变量表中的状态值将由 false 变为 true。

### 修改变量

在“修改值”一栏中为地址 MW2 输入数值“1500”，为地址 MW4 输入“1300”。

Address	Symbol	Display format	Status value	Modify value
I 0.1	"Key_1"	BOOL	true	
I 0.2	"Key_2"	BOOL	true	
Q 4.0	"Green_Light"	BOOL	true	
MW 2	"PE_Actual_Speed"	DEC	0	1500
DB1.DBW 6	"Petrol".Preset_Speed	DEC	1500	
Q 5.1	"PE_Preset_Speed_Reached"	BOOL	true	
MW 4	"DE_Actual_Speed"	DEC	0	1300
DB2.DBW 6	"Diesel".Preset_Speed	DEC	1200	
Q 5.5	"DE_Preset_Speed_Reached"	BOOL	true	

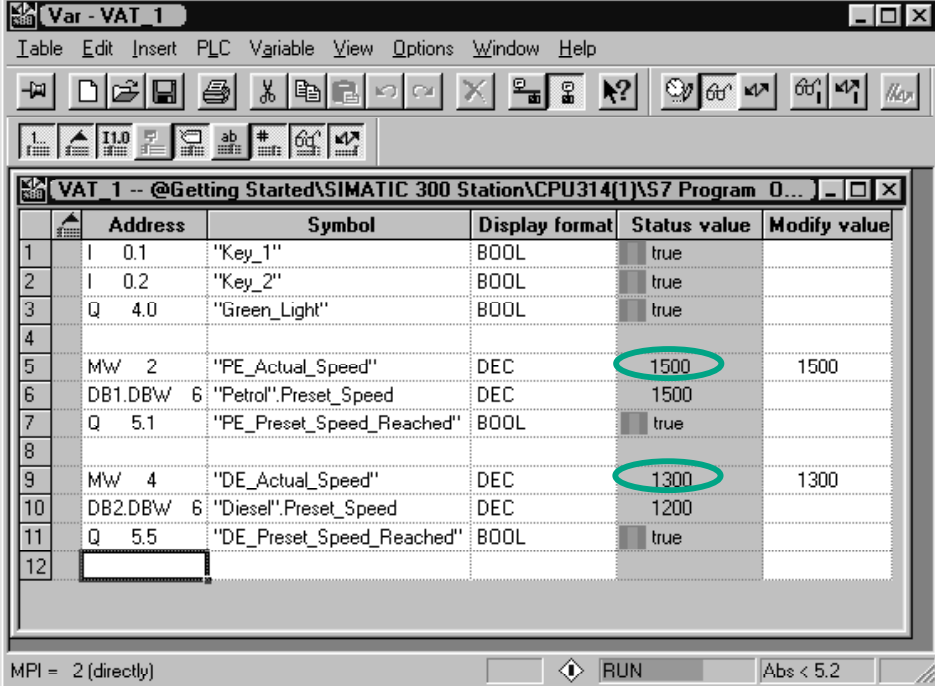


将修改的值传送到 CPU。





传送后，这些值将在 CPU 中进行处理。可以看到结果的比较。



	Address	Symbol	Display format	Status value	Modify value
1	I 0.1	"Key_1"	BOOL	true	
2	I 0.2	"Key_2"	BOOL	true	
3	Q 4.0	"Green_Light"	BOOL	true	
4					
5	MW 2	"PE_Actual_Speed"	DEC	1500	1500
6	DB1.DBW 6	"Petrol".Preset_Speed	DEC	1500	
7	Q 5.1	"PE_Preset_Speed_Reached"	BOOL	true	
8					
9	MW 4	"DE_Actual_Speed"	DEC	1300	1300
10	DB2.DBW 6	"Diesel".Preset_Speed	DEC	1200	
11	Q 5.5	"DE_Preset_Speed_Reached"	BOOL	true	
12					

MPI = 2 (directly)      RUN      Abs < 5.2

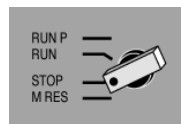
由于受到屏幕空间的限制，经常不能完全显示超大的变量表。如果您现在所使用的变量表过大，我们建议您使用 STEP 7 为一个 S7 程序生成多个变量表。您可以通过调整变量表，来使之满足自己的测试要求。您可以为每个变量表分配不同的名称，命名的方法与为块命名的方法相同(例如，用 OB1\_Network1 代替 VAT1)。使用符号表来分配新名称。

在帮助 > 目录的主题“调试”和“用变量表测试”中可以找到更多的信息。

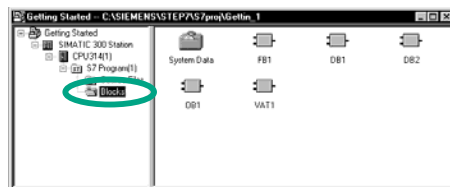
## 7.4 评估诊断缓冲区

在极端情况下，如果在处理一个 S7 程序时 CPU 进入了 STOP 状态，或者当您下载程序后无法将 CPU 切换为 RUN 状态，您可以从诊断缓冲区的事件列表中判断出现故障的原因。

实现这一功能的要求是：已经建立了与 CPU 的在线连接，并且 CPU 在 STOP 模式下。

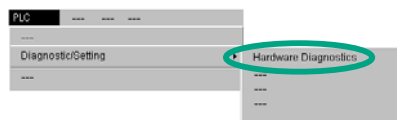


首先将 CPU 上的操作模式开关切换到 STOP 状态。

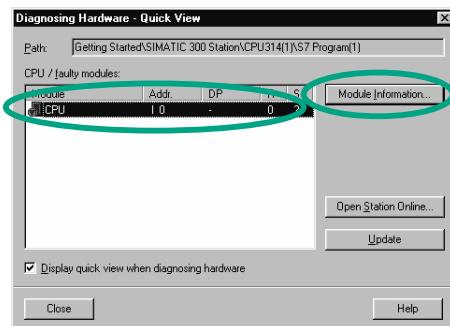


起始点还是在打开的 SIMATIC 管理器以及“Getting Started”项目窗口。

请选择 **Blocks** 文件夹。



如果您的项目中多个 CPU，首先要确定哪个 CPU 进入了 STOP 状态。



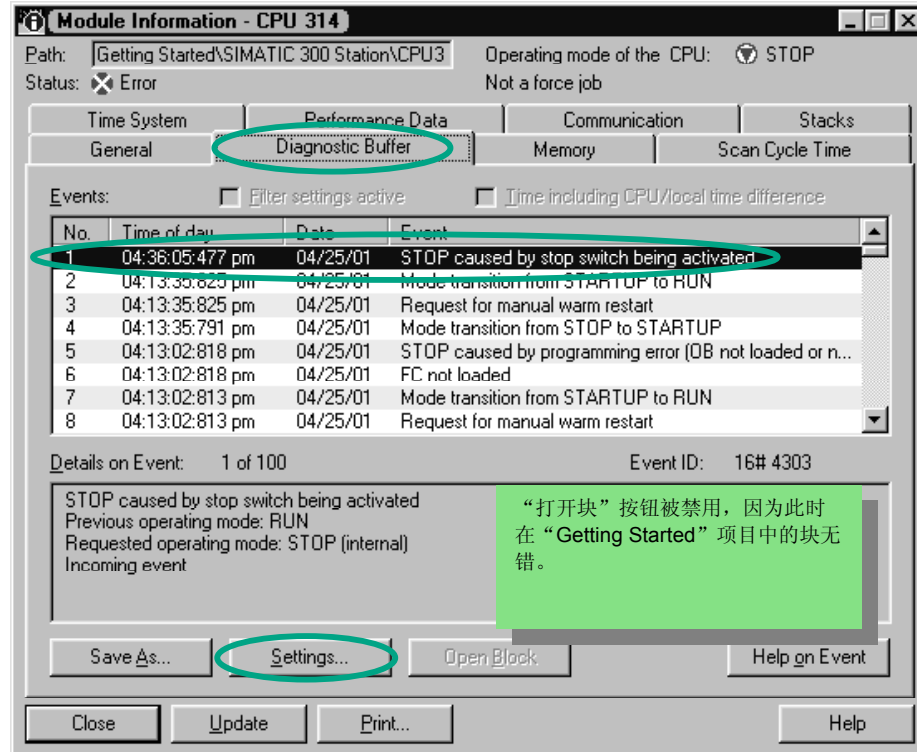
所有可访问的 CPU 都列在“诊断硬件”对话框中。处于 STOP 操作模式的 CPU 将高亮显示。

在“Getting Started”项目中只显示有一个 CPU。

单击**模块信息**，对该 CPU 诊断缓冲区进行评估。

如果只连接一个 CPU，您可以使用菜单命令 **PLC > 模块信息** 直接为该 CPU 查询模块信息。

“模块信息”窗口为您提供关于您的 CPU 的属性及参数的信息。现在选择“诊断缓冲区”标签，以确定造成进入 STOP 状态的原因。



最近的事件(编号为 1)在列表的最上面。显示进入 STOP 状态的原因。关闭除了 SIMATIC 管理器之外的所有窗口。

如果是由于编程错误造成 CPU 进入 STOP 模式，请选择该事件并单击“打开块”按钮。该块则在您所熟悉的 LAD/STL/FBD 编程窗口中打开，同时出错的程序段将高亮显示。通过本章您已经成功地完成了“Getting Started”示例项目，从创建一个项目开始，到调试完成结束。在下一章中，通过有选择性的练习，您可以进一步拓展您的知识。

在帮助 > 目录下的“诊断”下的“调用模块信息”中可以找到更多的信息。



## 8 编程一个功能

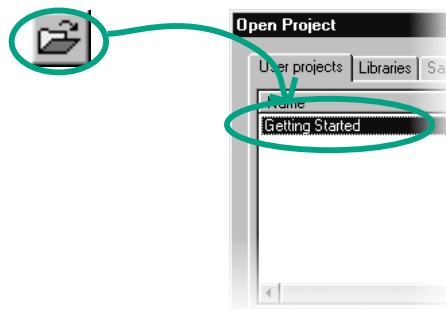
### 8.1 创建和打开功能(FC)

功能和功能块一样，在程序分级结构中位于组织块的下面。为使一个功能被 CPU 处理，必须在程序分级结构中的上一级调用它。与功能块不同的是，功能不需要数据块。

在功能中，参数也列在变量声明表中，但是不允许使用静态局部数据。

您可以使用 LAD/STL/FBD 编程窗口编程一个功能，其方法与编程一个功能块完全相同。

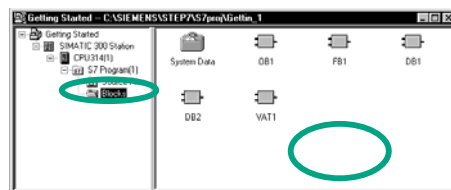
您应该已经熟悉了使用梯形图、功能块图或语句表(请参见第 4 章和第 5 章)编程，以及符号编程(请参见第 3 章)。



如果您已经在第 1 至第 7 章中创建了示例项目“Getting Started”，现在请打开该项目。

如果还没有创建该项目，请使用菜单命令文件 > “新建项目”向导在 SIMATIC 管理器中创建一个新项目。为此，可按照第 2.1 节中的指示进行操作，并且将项目重命名为“Getting Started Function”。

我们将继续使用“Getting Started”项目。而您可以使用新项目来完成每一个步骤。

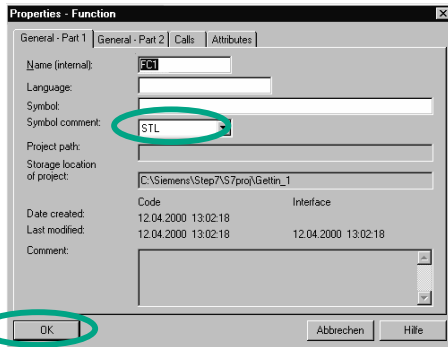


找到 **Blocks** 文件夹并打开它。

用鼠标右键单击右部窗口。

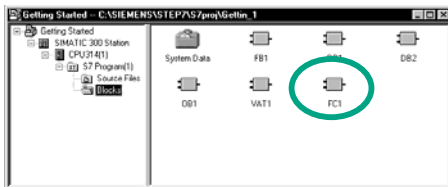


从弹出菜单中插入一个功能(FC)。



在“属性 - 功能”对话框中，接受名称 FC1 并选择所需要的编程语言。

用确定确认其余的缺省设置。



功能 FC1 被添加到 Blocks 文件夹中。

双击打开 FC1。

与功能块相反，功能的变量声明表中不能定义静态数据。

在功能块中定义的静态数据将在该块关闭时仍保留下来。例如，静态数据可以是用于“速度”限值的存储位(请参见第 5 章)。

要对功能进行编程，您可以使用符号表中的符号名。

在帮助 > 目录下的主题“设计自动化概念”和“设计程序结构的基础”以及“用户程序中的块”中，可以找到更多信息。

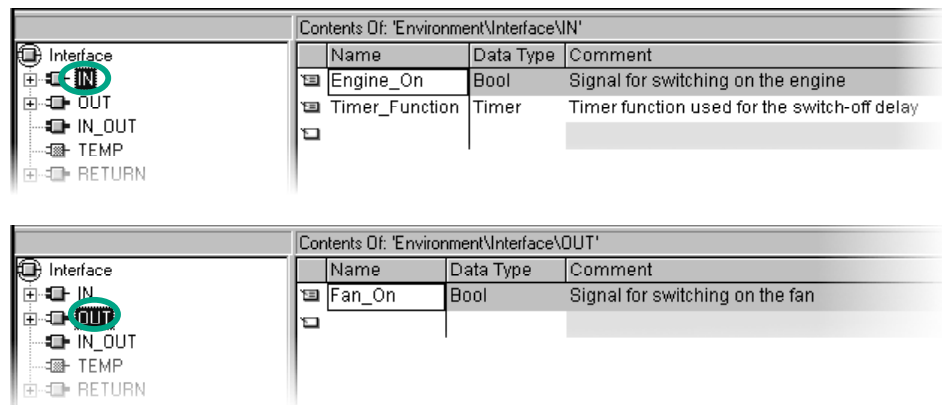
## 8.2 编程功能

在本节，您将根据我们的示例编程一个定时器功能。当发动机开动时，该定时器功能将使风扇打开(请参见第 5 章)，然后，在发动机停机后，该风扇继续运行 4 秒钟(延迟断开)。

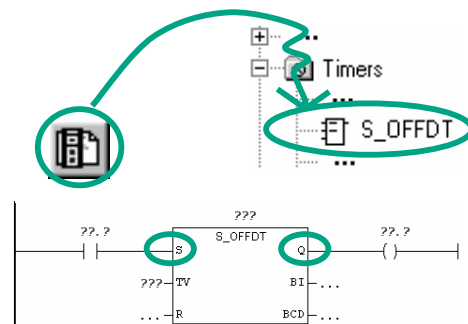
如前所述，您必须在变量详细视图中指定该功能的输入和输出参数(“in”和“out”声明)。

LAD/STL/FBD 编程窗口打开。使用该变量详细视图的方法与使用功能块的详细视图一样(请参见第 5 章)。

请输入如下声明：



### 用梯形图编程定时器功能



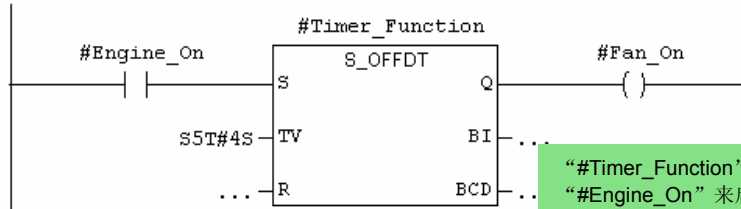
请选择输入梯形图指令的当前路径。

在编程元素目录中查找到 **S\_OFFDT** (启动延时断开定时器)，并选择该元素。

在输入 **S** 前插入一个常开触点。  
在输出 **Q** 之后插入一个线圈。



选择问号，输入“#”并选择相应的名称。



“#Timer\_Function”由输入参数“#Engine\_On”来启动。当该功能在OB1中调用后，它将被提供给汽油发动机参数一次，并提供给柴油发动机参数一次(例如，T1“PE\_Follow\_on”)。您以后将在符号表中输入这些参数的符号名。

#### 用语句表编程定时器功能

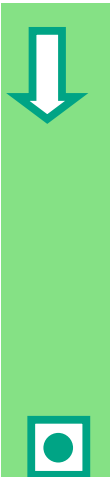
```
A #Engine_On
L S5T#4S
SF #Timer_Function
A #Timer_Function
= #Fan_On
```

如果使用语句表编程，则选择程序段下面的输入区域，并按所示内容输入语句。

保存该功能并关闭窗口。



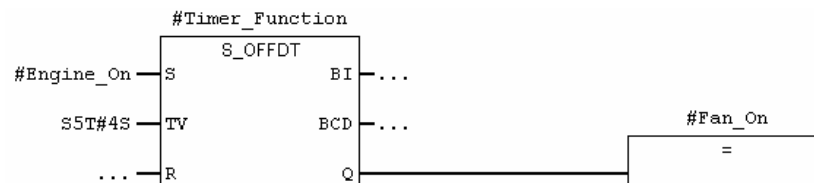




### 用功能块图编程定时器功能

如果您使用功能块图编程，则选择程序段下面的输入区域，并为定时器功能输入下面的 FBD 程序。

保存该功能，并关闭窗口。



为了处理定时器功能，您需要在块的分级结构中处于更高一级的块中调用该功能(在我们的示例中是在 OB1 中)。

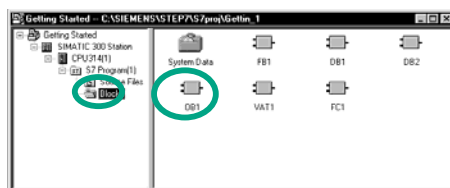
在帮助 > 目录下的主题“调用参考帮助”、“STL、FBD 或 LAD 语言描述”和“定时器指令”中可以找到更多的信息。

### 8.3 在 OB1 中调用功能

对功能 FC1 的调用的执行方式与在 OB1 中对功能块的调用相似，在 OB1 中用汽油机或者柴油机的相应的地址给功能的所有参数赋值。

由于这些地址还未在符号表中定义，现在将添加这些地址的符号名。

地址是 STEP 7 语句的一部分，它指定处理器应对什么执行指令。地址既可以是绝对地址也可以是符号地址。



SIMATIC 管理器连同项目“Getting Started”或您的新项目一同打开。

查找到 **Blocks** 文件夹并打开 **OB1**。

LAD/STL/FBD 编程窗口打开

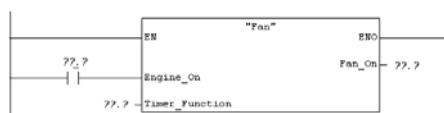
在梯形图中编程调用



您现在已经进入 **LAD** 视图。选择程序段 5 并插入一个新的程序段 6。



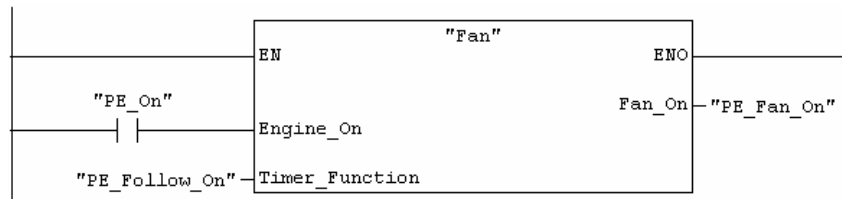
在编程元素目录中查找到 FC1，插入该功能。



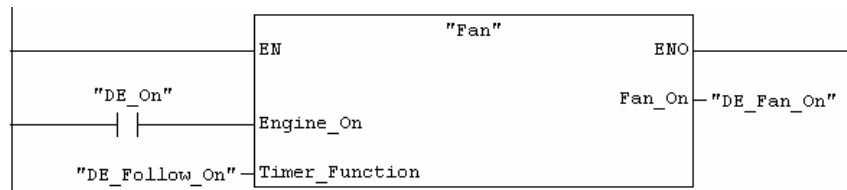
在“Engine\_On”前插入一个常开触点。

使用菜单命令视图 > 显示 > 符号表示法，可以在符号地址和绝对地址之间切换。

单击 FC1 调用的问号，插入符号名。



在程序段 7 中使用柴油机的地址对功能 FC1 的调用进行编程。可以采用与前一程序段相同的方法(您已经将柴油机的地址添加到了符号表中)。



保存该块，并关闭窗口。

激活菜单命令视图 > 显示 > 符号信息来查看每个程序段中各个地址的信息。

要在屏幕上显示多个程序段，可取消激活菜单命令视图 > 显示 > 注释，如有必要，请释放菜单命令视图 > 显示 > 符号信息。

使用菜单命令视图 > 缩放因子，您可以改变所显示的程序段的大小。





### 用语句表编程调用

```

Network 6 : Controlling the Fan for the Petrol Engine
CALL "Fan"
  Engine_On      := "PE_On"
  Timer_Function := "PE_Follow_On"
  Fan_On         := "PE_Fan_On"

Network 7 : Controlling the Fan for the Diesel Engine
CALL "Fan"
  Engine_On      := "DE_On"
  Timer_Function := "DE_Follow_On"
  Fan_On         := "DE_Fan_On"
    
```

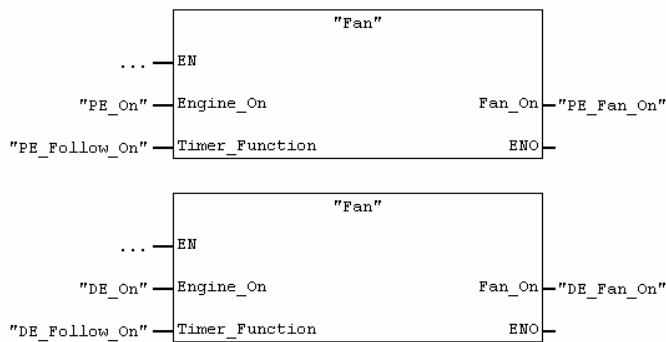
如果使用语句表编程，则选择新的程序段下面的输入区域，并按所示内容输入 STL 语句。

保存该调用，并关闭窗口。

### 用功能块图编程调用

如果使用功能块图编程，则选择新的程序段下面的输入区域，并输入如下所示的 FBD 指令。

保存该调用并关闭窗口。



在示例中所编程的对功能的调用是一个无条件调用；即该功能总会被处理。  
 根据您的自动化任务的要求，可以根据某些条件来调用功能或功能块；例如，一个输入或者一个前面的逻辑运算结果。框图中的输入 EN 和输出 ENO 就是用于程序的条件调用。

在帮助 > 目录下的主题“调用参考帮助”，“LADL、FBD 或 STL 语言描述”或“程序控制指令”中可以找到更多的信息。

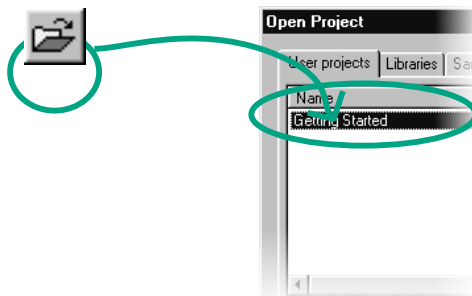
## 9 编程一个共享数据块

### 9.1 创建和打开共享数据块

如果 CPU 中没有足够的内部存储位来保存所有数据，可将一些指定的数据存储到一个共享数据块中。

存储在共享数据块中的数据可以被其它的任意一个块使用。而一个背景数据块被指定给一个特定的功能块，它的数据只在这个功能块中有效(请参见第 5.5 节)。

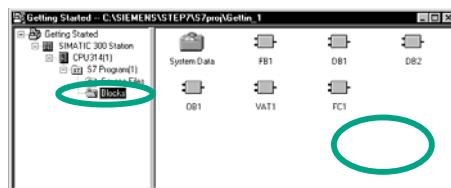
您应该已经熟悉了使用梯形图、功能块图或语句表编程(请参见第 4 章和第 5 章)以及符号编程(请参见第 3 章)。



如果您已经在第 1 至第 7 章中创建并使用了样板项目“Getting Started”，请立即打开。

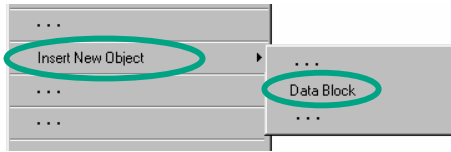
如果没有，请使用菜单命令文件 > “新建项目”向导在 SIMATIC 管理器中创建一个新的项目。为此，可按照第 2.1 节中的指导进行操作，并重新命名该项目为“Getting Started Function”。

我们将继续使用“Getting Started”项目。而您使用一个新项目也可以实现每一步。

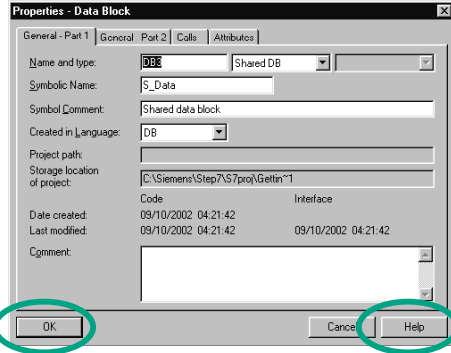


查找到 **Blocks** 文件夹并打开它。

用鼠标右键点击右部窗口。



从弹出菜单中插入一个数据块(DB)。



在“属性 - 数据块”对话框中用确定接受所有的缺省设置。

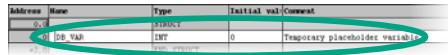
需要进一步的信息，请点击“帮助”按钮。

数据块 DB3 已经被加入到 **Blocks** 文件夹。

双击打开 **DB3**。

提示：在第 5.5 节中，可通过激活选项“引用功能块的数据块”生成一个背景数据块。与之相反，用“数据块”可以创建一个共享的数据块。

### 在数据块中编写变量程序



在名称栏中输入“PE\_Actual\_Speed”。

点击鼠标右键选择类型，使用弹出式菜单中的菜单命令元素类型 > INT。

下面的示例中，DB3 中定义了三个共享数据。将这些数据相应地输入变量声明表中。

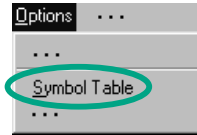
Address	Name	Type	Initial val	Comment
0.0		STRUCT		
+0.0	DB_VAR	INT	0	Temporary placeholder variable
=2.0		END_STRUCT		

对于数据块中用于实际速度的变量“PE\_Actual\_Speed”以及“DE\_Actual\_Speed”的处理，与存储字 MW2 (PE\_Actual\_Speed)和 MW4 (DE\_Actual\_Speed)相同。这一部分将在下一章中进行讨论。



保存该共享数据块。

## 分配符号



您也可以给数据块分配符号名。

打开符号表并为数据块 DB3 输入符号名“S\_Data”。

如果您在第 4 章中从样例项目 (zEn01\_02\_STEP7\_\_STL\_1-10, zEn01\_06\_STEP7\_\_LAD\_1-10 或 zEn01\_04\_STEP7\_\_FBD\_1-10) 中复制符号表到您的“Getting Started”项目中，现在您不需要增加任何符号。

Symbol	Address	Data Type	Comment
...	...	...	...
S_Data	DB 3	DB 3	Shared data block



保存符号表并关闭“符号编辑器”窗口。

同样，关闭共享数据块。



#### 变量声明表中的共享数据块：

使用菜单命令视图 > 数据视图，可为共享数据块修改表中数据类型 INT 的实际值(请参见第 5.5 节)。

#### 符号表中的共享数据块：

与背景数据块相反，在符号表中共享数据块的数据类型总是绝对地址。在我们的示例中，数据类型是“DB3”。对于背景数据块，相应的功能块总是指定的数据类型。

在帮助 > 目录的主题“编程块”和“创建数据块”中可以找到更多的信息。





## 10 编程多重背景

### 10.1 创建和打开较高一级的功能块

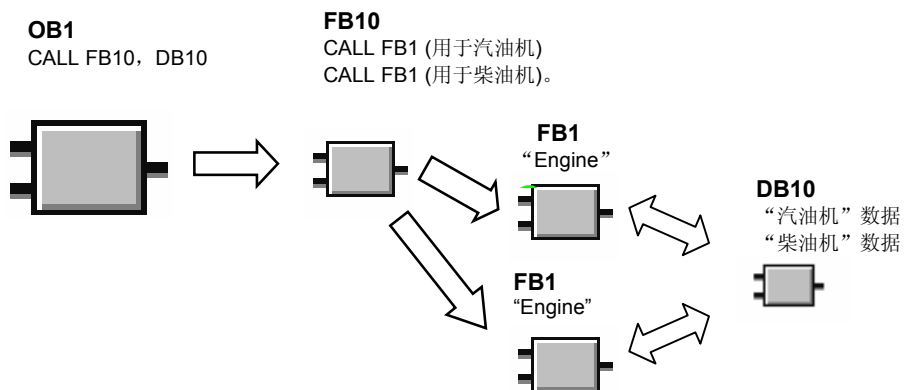
在第 5 章中，您创建了一个功能块“Engine” (FB1) 控制一台发动机的程序。当功能块 FB1 在组织块 OB1 中调用时，它使用了数据块“Petrol” (DB1) 和“Diesel” (DB2)。每个数据块包含发动机的不同数据(例如，#Setpoint\_Speed)。

现在想象一下，您的自动化设备还需要其它的程序控制发动机；例如，用于菜籽油发动机的控制程序，或者用于氢发动机的控制程序，等等。

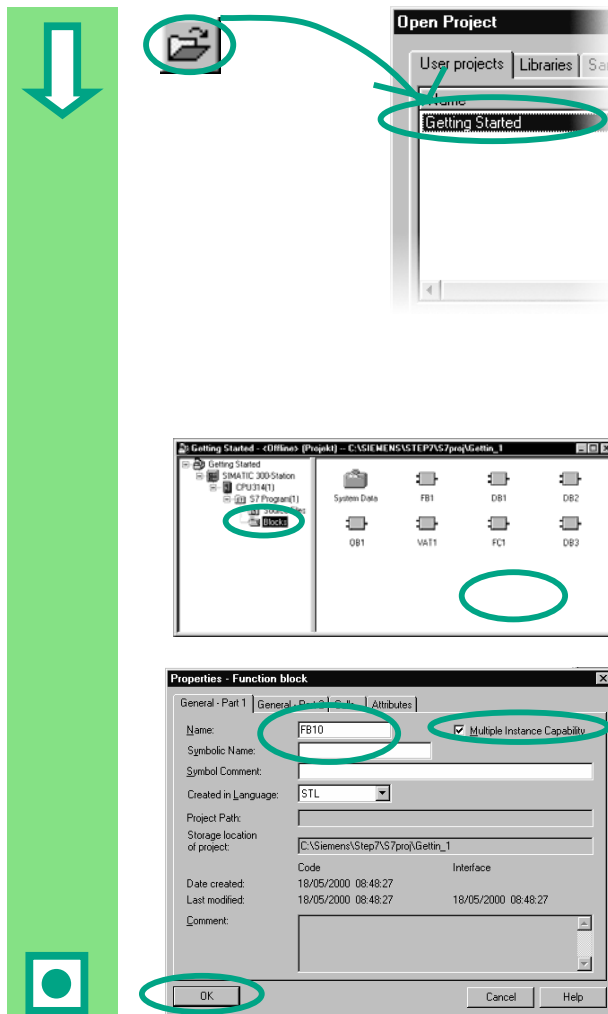
按照目前您已经学习过的步骤，现在要为一个附加的发动机控制程序使用 FB1，并且每次为发动机的数据分配新的数据块；例如，FB1 和 DB3 用于控制菜籽油发动机，FB1 和 DB4 用于控制氢发动机，等等。当您创建新的发动机控制程序时，块数量的增加是非常大的。

另一方面，通过使用多重背景可以减少块的数量。为此，您要创建一个新的、更高级别的功能块(在我们的示例中是 FB10)，并在其中调用未作任何修改的 FB1 作为“局部背景”。对每一个调用，子程序 FB1 将它的数据存储于较高一级 FB10 的数据块 DB10 中。这就意味着您无需给 FB1 分配任何数据块。所有的功能块都指向一个数据块(此处是 DB10)。

数据块 DB1 和 DB2 被集成在 DB10 中。为此，必须在 FB10 的静态局域数据中声明 FB1。



您应该已经熟悉了使用梯形图、功能块图或语句表编程(请参见第 4 章和第 5 章)以及符号编程(请参见第 3 章)。



如果您已经在第 1 至第 7 章中创建并使用了示例项目“Getting Started”，则将它打开。

如果没有，可在 SIMATIC 管理器中打开以下项目之一：

ZEn01\_05\_STEP7\_\_LAD\_1-9 用于梯形图，

ZEn01\_01\_STEP7\_\_STL\_1-9 用于语句表

ZEn01\_03\_STEP7\_\_FBD\_1-9 用于功能块图。

查找到 **Blocks** 文件夹并打开它。

在右半窗口中击鼠标右键，然后使用弹出菜单插入一个功能块。

将块名改为 **FB10** 并选择所需要的编程语言。

如有必要，激活**多重背景 FB**，并用确定确认其余的缺省设置。

**FB10** 被加入到 **Blocks** 文件夹。双击以打开 **FB10**。

您可以为任意功能块创建多重背景，例如，为阀门控制程序。如果您要使用多重背景，注意调用块和被调用块都必须具有多重背景功能。

在帮助 > 目录的主题“编程块”和“创建块和库”中可以找到更多的信息。

## 10.2 编程 FB10

要将 FB1 作为 FB10 的一个“局部背景”调用，则需要在变量详细视图中为每一个计划调用的 FB1 声明一个具有不同名字的静态变量。这里，数据类型是 FB1 (“Engine”)。

### 声明/定义变量

FB10 在 LAD/STL/FBD 编程窗口中打开。将顺序映像的声明传送到变量详细视图中。为此，请依次选择声明类型“OUT”、“STAT”以及“TEMP”，并在变量详细视图中进行输入。从下拉列表中选择“FB <nr>”作为声明类型“STAT”的数据类型，并用“1”来替换字符串“<nr>”。

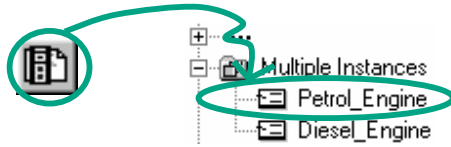
Contents Of: 'Environment\Interface\OUT'					
	Name	Data Type	Address	Initial Value	Comment
Interface	IN				
	OUT				
	IN_OUT				
	STAT				
	TEMP				
	PE_Preset_Speed_Reached	Bool	0.0	FALSE	Both engines have reached the preset speed

Contents Of: 'Environment\Interface\STAT'					
	Name	Data Type	Address	Initial Value	Comment
Interface	IN				
	OUT				
	IN_OUT				
	STAT				
	TEMP				
	Petrol_Engine	Engine	2.0		First local instance of FB1 "Engine"
	Diesel_Engine	Engine	10.0		Second local instance of FB1 "Engine"

Contents Of: 'Environment\Interface\TEMP'					
	Name	Data Type	Address	Initial Value	Comment
Interface	IN				
	OUT				
	IN_OUT				
	STAT				
	TEMP				
	PE_Preset_Speed_Reached	Bool	0.0		Preset speed reached (petrol engine)
	DE_Preset_Speed_Reached	Bool	0.1		Preset speed reached (diesel engine)

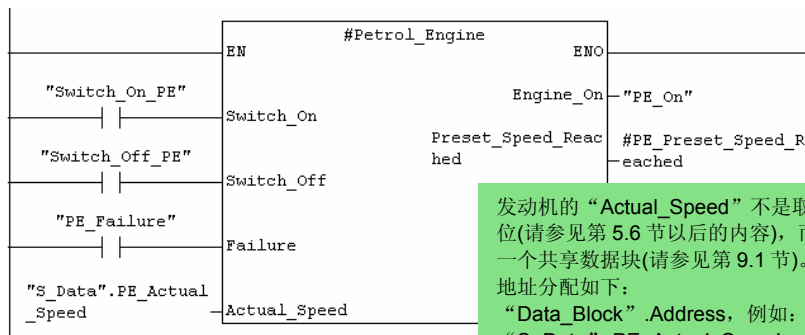
声明的局部背景将出现在“编程元素”标签中的“多重背景”下面。

### 用梯形图编程 FB10



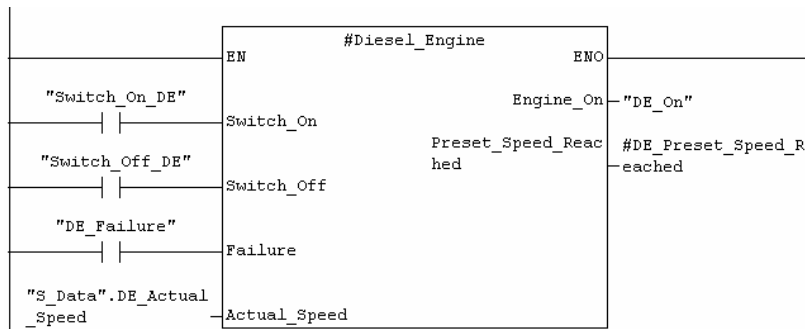
在程序段 1 中插入调用  
“Petrol\_Engine” 作为多重背景数据块  
“Petrol\_Engine”。

然后插入所需的常开触点并用符号名完成调用。



发动机的“Actual\_Speed”不是取自存储位(请参见第 5.6 节以后的内容),而是来自一个共享数据块(请参见第 9.1 节)。通用的地址分配如下:  
“Data\_Block”.Address, 例如:  
“S\_Data”.PE\_Actual\_Speed.

插入一个新的程序段, 并为柴油机编写调用程序。按照与程序段 1 相同的方法进行。





插入一个新的程序段，并用相应的地址编程一个串联电路。保存程序并关闭块。



使用相应的临时变量。您将通过下拉菜单中左边所显示的符号来识别临时变量。

保存程序并关闭块。

```
#PE_Preset_Speed_R #DE_Preset_Speed_R #Preset_Speed_Reach
eached eached hed
```

临时变量  
(“PE\_Setpoint\_Reached”以及  
“DE\_Setpoint\_Reached”)用于输出  
参数“Setpoint\_Reached”，参  
数将在 OB1 中进行进一步的处理。

## 用语句表编程 FB10

```
CALL #Petrol_Engine
Switch_On := "Switch_On_PE"
Switch_Off := "Switch_Off_PE"
Failure := "PE_Failure"
Actual_Speed := "S_Data".PE_Actual_Speed
Engine_On := "PE_On"
Preset_Speed_Reached := #PE_Preset_Speed_Reached

CALL #Diesel_Engine
Switch_On := "Switch_On_DE"
Switch_Off := "Switch_Off_DE"
Failure := "DE_Failure"
Actual_Speed := "S_Data".DE_Actual_Speed
Engine_On := "DE_On"
Preset_Speed_Reached := #DE_Preset_Speed_Reached

A #PE_Preset_Speed_Reached
A #DE_Preset_Speed_Reached
= #Preset_Speed_Reached
```

如果使用语句表编程，则选择新的程序段下面的输入区域，并按所示内容输入 STL 指令。

保存程序并关闭块。

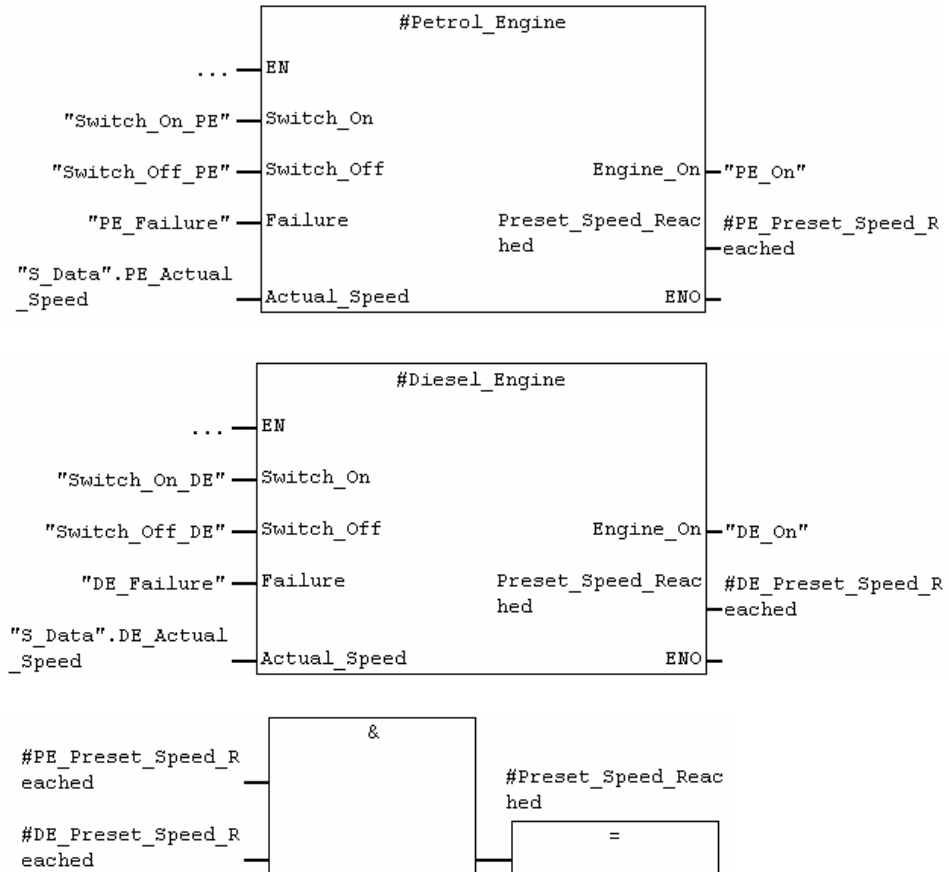




### 用功能块图编程 FB10

如果使用功能块图编程，则选择新的程序段下面的输入区域，并输入如下所示的 FBD 指令。

保存程序并关闭块。

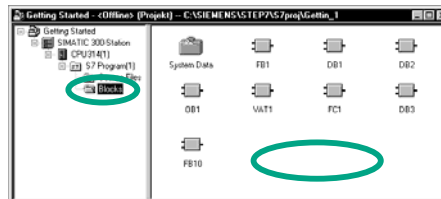


要在 FB10 中编辑对 FB1 的两次调用，FB10 本身必须被调用。  
多重背景只能用于功能块的编程。不能为功能(FC)创建多重背景。

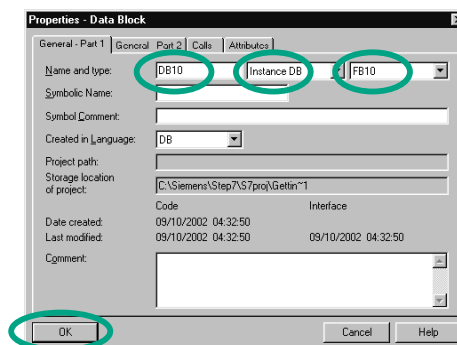
在帮助 > 目录下主题“编程块”，“创建逻辑块”和“变量声明表中的多重背景”中可以找到更多的信息。

## 10.3 生成 DB10 并调整实际值

新的数据块 DB10 将替代数据块 DB1 以及 DB2。用于汽油机和柴油机的数据存储于 DB10 中，后面在 OB1 中调用 FB1 时将会用到(请参见第 5.6 节以后的“在 OB1 中调用 FB1”)。



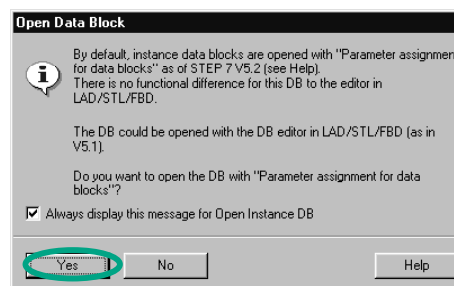
在 SIMATIC 管理器中，使用弹出菜单在项目“Getting Started”的 **Blocks** 文件夹中创建数据块 DB10。



为此，在出现的对话框“属性-数据块”中将数据块名修改为 DB10，并在相邻的下拉列表中选择应用程序“背景数据块”。在右边的下拉式列表中，选择要分配的功能块“FB10”并且用确定来确认其余的设置。

数据块 DB10 已经被添加到“Getting Started”项目中。

双击 DB10。



在下面的对话框中，使用是进行确认以打开背景数据块。选择菜单命令视图 > 数据视图。

数据视图将在 DB10 中显示各个变量，包括 FB1 两个调用的“内部”变量(“局部背景”)。

声明视图中变量的显示内容和它们在 FB10 中声明的显示内容相同。

将柴油机的实际值更改为“1300”，存储该块然后关闭它。

	Address	Declaration	Name	Type	Initial value	Actual value	Comment
1	0.0	out	Preset_Speed_Reached	BOOL	FALSE	FALSE	Both engines have reached the preset speed
2	2.0	stat:in	Petrol_Engine.Switch_On	BOOL	FALSE	FALSE	Switch on engine
3	2.1	stat:in	Petrol_Engine.Switch_Off	BOOL	FALSE	FALSE	Switch off engine
4	2.2	stat:in	Petrol_Engine.Failure	BOOL	FALSE	FALSE	Engine failure, causes the engine to switch off
5	4.0	stat:in	Petrol_Engine.Actual_Speed	INT	0	0	Actual engine speed
6	6.0	stat:out	Petrol_Engine.Engine_On	BOOL	FALSE	FALSE	Engine is switched on
7	6.1	stat:out	Petrol_Engine.Preset_Speed_Reached	BOOL	FALSE	FALSE	Preset speed reached
8	8.0	stat	Petrol_Engine.Preset_Speed	INT	1500	1500	Requested engine speed
9	10.0	stat:in	Diesel_Engine.Switch_On	BOOL	FALSE	FALSE	Switch on engine
10	10.1	stat:in	Diesel_Engine.Switch_Off	BOOL	FALSE	FALSE	Switch off engine
11	10.2	stat:in	Diesel_Engine.Failure	BOOL	FALSE	FALSE	Engine failure, causes the engine to switch off
12	12.0	stat:in	Diesel_Engine.Actual_Speed	INT	0	0	Actual engine speed
13	14.0	stat:out	Diesel_Engine.Engine_On	BOOL	FALSE	FALSE	Engine is switched on
14	14.1	stat:out	Diesel_Engine.Preset_Speed_Reached	BOOL	FALSE	FALSE	Preset speed reached
15	16.0	stat	Diesel_Engine.Preset_Speed	INT	1500	1300	Requested engine speed

现在所有的变量都包含在 DB10 的变量声明表中。在前半部分，您可以看到调用功能块“Petrol\_Engine”的变量，在后半部分中则是调用功能块“Diesel\_Engine” (请参见第 5.5 节)。

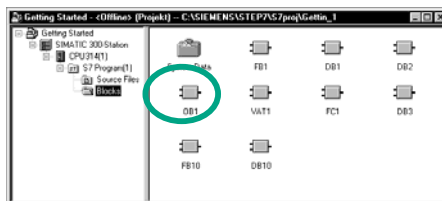
FB1 的“内部”变量仍然保持它们的符号名；例如，“Switch\_On”。现在局域背景的名字则被加在这些符号名的前面；例如，“Petrol\_Engine.Switch\_On”。

在帮助 > 目录的主题“编程块”和“创建数据块”中可以找到更多的信息。



## 10.4 在 OB1 中调用 FB10

在我们的示例中，在 OB1 中调用 FB10。该调用与您已经学过的在 OB1 中对 FB1 的调用，具有相同的功能(请参见第 5.6 节以后的部分)。通过使用多重背景，您可以替换前面第 5.6 节中编好的程序段 4 和 5。



打开项目中的 **OB1**，您刚在该项目中编写过 FB10。

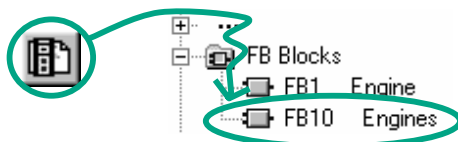
### 定义符号名

LAD/STL/FBD 编程窗口打开。用菜单命令选项 **> 符号表** 打开符号表，为功能块 FB10 和数据块 DB10 在符号表中输入符号名。

保存符号表，并关闭窗口。

Symbol	Address	Data Type	Comment
...	...	...	...
Engines	FB 10	FB 10	Example of multiple instances
Engine_Data	DB 10	FB 10	Instance data block for FB10 10
...	...	...	...

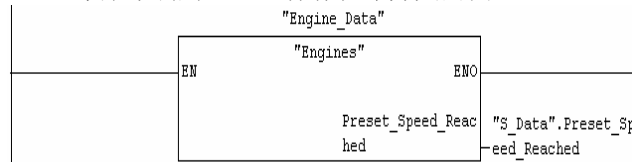
### 在梯形图中编写调用程序



在 OB1 的结尾处，插入一个新的程序段并添加对 **FB10** (“Engines”) 的调用。



用相应的符号名完成下面的调用。  
删除 OB1 中对 FB1 的调用(前面第 5.6 节中的程序段 4 和 5)，因为我们现在通过 FB10 来集中调用 FB1。保存程序并关闭块。



FB10 ( “Engines” ) 的输出信号  
“Setpoint\_Reached” 被传送给共享数据块中的  
变量。

### 用语句表编写调用程序

如果使用语句表编程，则选择新的程序段下面的输入区域，并将 STL 指令输入在下面。为此，请使用编程元素目录中的 **FB 块 > FB10 发动机**。

删除 OB1 中对 FB1 的调用(前面第 5.6 节中的程序段 4 和 5)，因为我们现在通过 FB10 来集中调用 FB1。

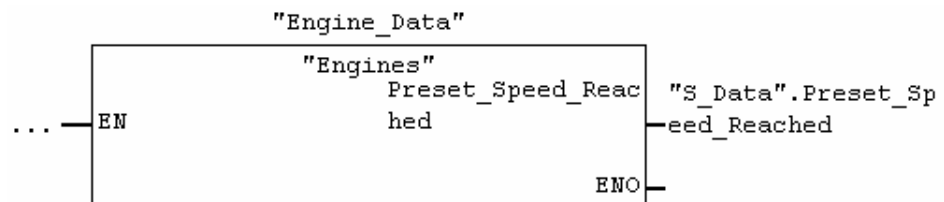
保存程序并关闭块。

```
CALL "Engines" , "Engine_Data"  
Preset_Speed_Reached:="S_Data".Preset_Speed_Reached
```



### 用功能块图编写调用程序

如果使用功能块图编程，则选择新的程序段下面的输入区域，并将 FBD 指令输入到下面。为此，请使用编程元素目录中的 **FB 块 > FB10 发动机**。删除 OB1 中对 FB1 的调用(前面第 5.6 节中的程序段 4 和 5)，因为我们现在通过 FB10 来集中调用 FB1。保存程序并关闭块。



如果您的自动化任务还需要更多的发动机控制程序；例如，用于汽油发动机、氢发动机等等.....，您可用相同的方法将它们编作多重背景并在 FB10 中进行调用。

为此，如 FB10 ( “Engine” ) 的变量声明表中所示的那样声明其它的发动机，并在 FB10 中编程对 FB1 的调用(编程元素目录中的多重背景)。之后您可以定义新的符号名；例如，在符号表中为开动和关停过程定义符号。

在帮助 > 目录下“调用参考帮助”的主题“STL、FBD 或 LAD 语言描述”中，可以找到更多的信息。



# 11 组态分布式 I/O

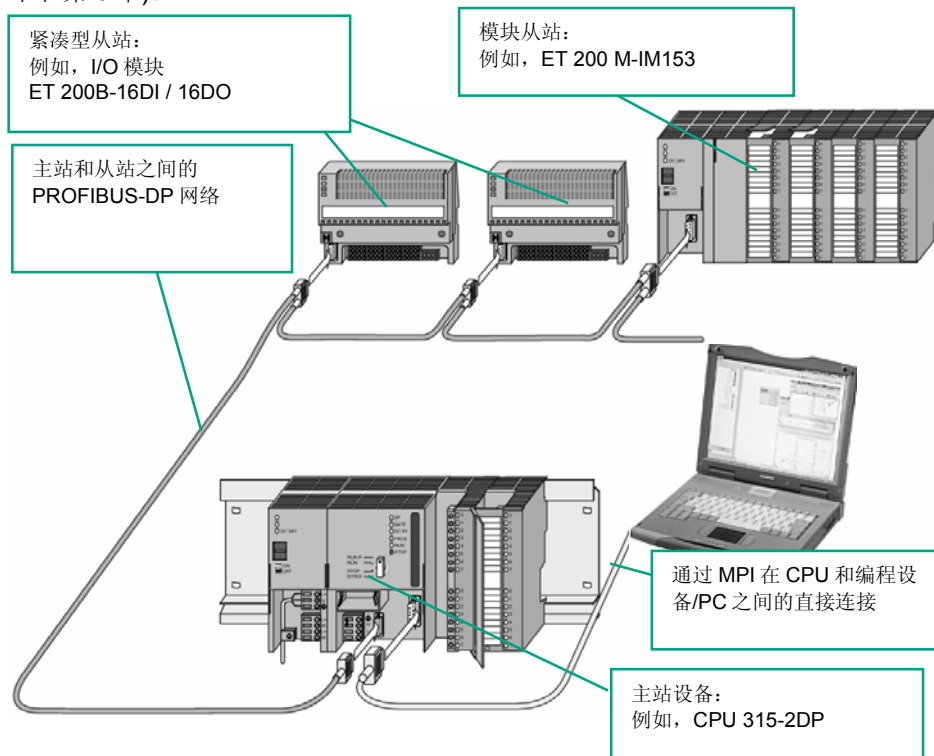
## 11.1 用 PROFIBUS DP 组态分布式 I/O

具有常规组态的自动化系统用电缆来连接传感器和执行器，这些电缆直接插入到中央可编程逻辑控制器的 I/O 模块上。这通常意味着需要用到大量的接线。

使用分布式组态，可将输入输出模块放置到离传感器和执行器很近的地方，从而减少接线量。您可以使用 PROFIBUS DP 来建立可编程控制器、I/O 模块和现场设备之间的连接。

您可在第 6 章中找到如何编程常规组态的信息。创建中央组态和创建分布式组态没有区别。从硬件目录中选择要使用的模块并将它们安置在机架上，按照您的要求修改其属性。

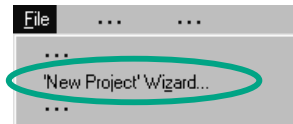
在阅读本章之前，您最好已经熟悉了创建项目和编程中央组态的内容(请参见第 2.1 节和第 6 章)。



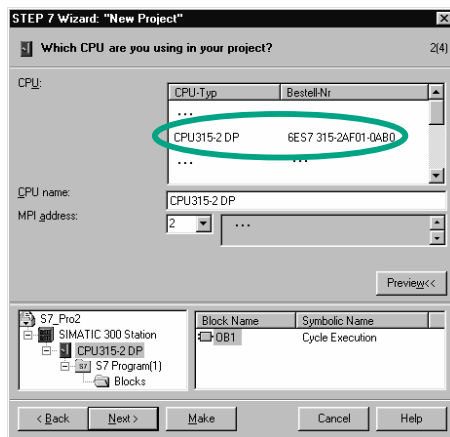
### 创建新项目



首先从 SIMATIC 管理器开始。为简化操作过程，请先关闭所有打开的项目。



创建一个新项目

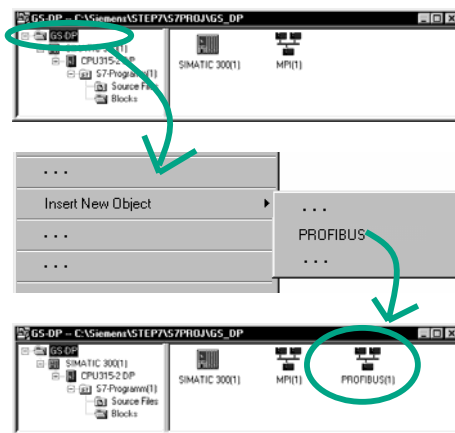


在相应的对话框中选择 **CPU 315-2DP**(带有 PROFIBUS-DP 网络的 CPU)。

现在按照与第 2.1 节中相同的方法进行，并为该项目分配名称“GS-DP”(Getting Started – Distributed I/O)。

如果您想在此处创建自己的组态，请现在在指定 CPU，请注意，您的 CPU 必须能够支持分布式 I/O。

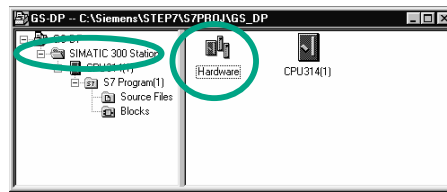
### 插入 PROFIBUS 网络



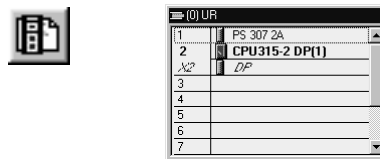
选择 **GS-DP** 文件夹。

在右半窗口中用击鼠标右键，然后插入 **PROFIBUS** 网络。

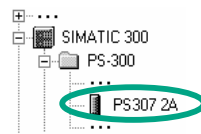
## 组态站



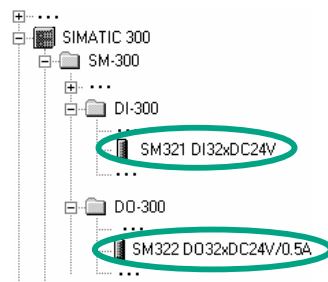
选择 **SIMATIC 300 站** 文件夹并双击硬件。  
“硬件配置”窗口打开(请参见第 6.1 节)。



CPU 315-2 DP 已经显示在机架中。如有必要，使用菜单命令视图 > 硬件配置或工具栏中的相应按钮打开硬件目录。



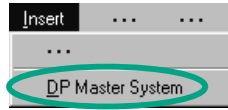
将电源模块 **PS307 2A** 拖放到插槽 1。



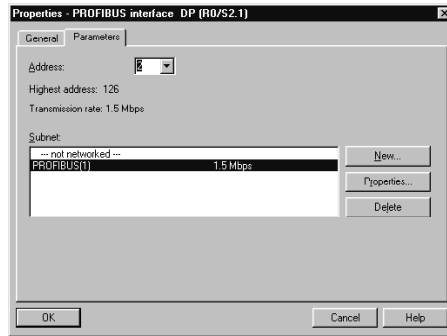
按照相同的方法，在插槽 4 和插槽 5 插入 I/O 模块 **DI32xDC24V** 和 **DO32xDC24V/0.5A**。

除了支持分布式 I/O 的 CPU 之外，您也可以在同一机架上放置其它的 CPU(此处不作讨论)。

### 组态 DP 主站系统



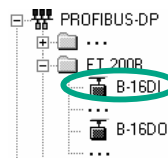
选择槽 2.1 中的 DP 主站，并插入 **DP 主站系统**。



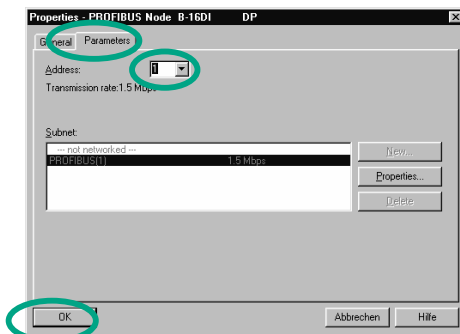
使用对话框中显示的建议地址。在“子网”域中选择“PROFIBUS(1)”，然后用**确定**应用设置。



您可以通过按住鼠标左键拖动您放置在主站系统中的任意对象，来移动它们。



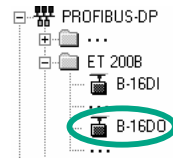
在硬件目录中查找到模块 **B-16DI**，将该模块插入到主站系统(拖动对象到主站系统，直到光标变为一个“+”号时放开该对象)。



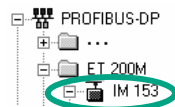
您可以在“属性”对话框的“参数”标签中修改您已插入的模块的节点地址。用**确定**确认推荐的地址。





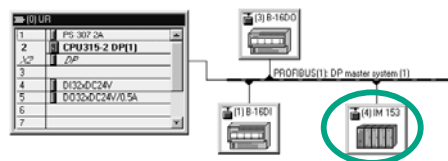


用同样的方式可将模块 **B-16DO** 拖放到主站系统。



将接口模块 **IM153** 拖放至主站系统并用 **确定** 再次确认站地址。

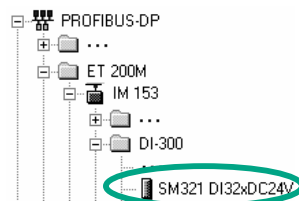
在我们的示例中，我们使用缺省站地址。但是，您可以随时按照您的要求修改这些地址。



在网络中选择 **ET200M**。  
**ET200M** 的空插槽显示在组态表的下部。

Slot	Module	Order Number
4		
5		
6		
7		
8		
9		
10		
11		

此处选择插槽 **4**。

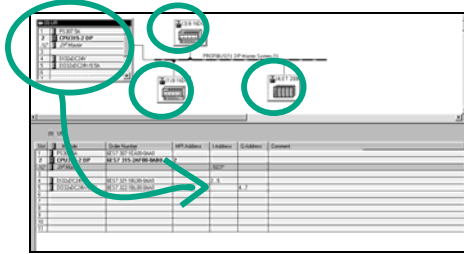


**ET200M** 本身可以有附加的 I/O 模块。例如，为插槽 **4** 选择模块 **DI32xDC24V**，然后双击该模块将其插入。

您应该始终确保在使用硬件目录时处于正确的文件夹中。例如，查找到 **ET200M** 文件夹，为 **ET200M** 选择模块。

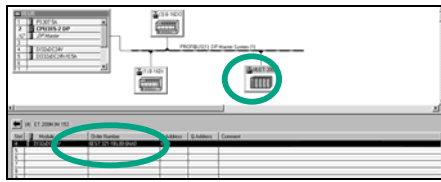


### 修改节点地址



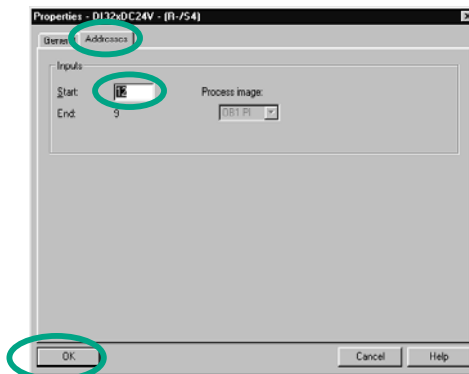
在此示例中，不需要修改节点地址。但是在实际的项目开发过程中，经常需要修改节点地址。

依次选择其它节点，检查输入地址和输出地址。“配置硬件”应用程序已调整了所有地址，所有不会出现双重赋值的情况。



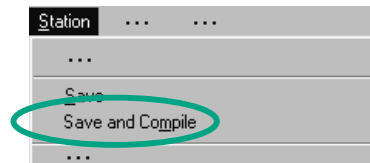
让我们设想一下，您想要修改 ET 200M 的地址：

选择 **ET200M** 并双击 **DI32xDC24V** (插槽 4)。



现在修改“属性”对话框的“地址”标签中的输入地址，将 **6** 改成 **12**。用**确定**关闭对话框。



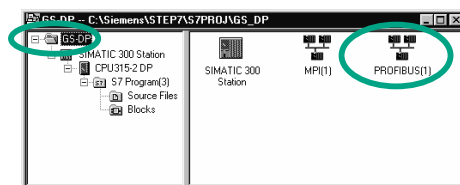


最后，**保存并编译**该分布式 I/O 组态。  
关闭窗口。

菜单命令**保存并编译**意味着自动对组态作一致性检查。如果没有出错，将生成系统数据，该系统数据可下载到可编程控制器。

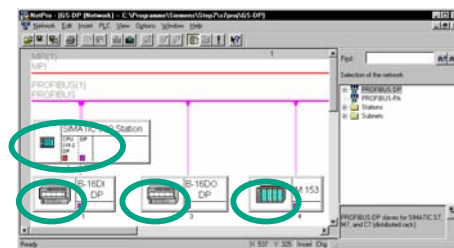
使用**保存**，即使组态有错误也会保存。但是，无法将该组态下载到可编程控制器。

### 另一种方法：组态网络



您也可以使用可选软件包“组态网络”来组态分布式 I/O。

在 SIMATIC 管理器中双击网络 **PROFIBUS (1)**。



“NetPro”窗口打开。

您可以从网络对象的目录中将其它 DP 从站拖放到 PROFIBUS DP。

双击任意元素以对其进行组态。“配置硬件”窗口打开。

使用菜单命令**站 > 一致性检查**(“配置硬件”窗口)以及**网络 > 一致性检查**(“组态网络”窗口)，可以在保存前检查组态是否有错。所有的错误均将显示，STEP 7 将会推荐可能的解决方案。

在**帮助 > 目录**下的主题“配置硬件”和“组态分布式 I/O”中可以找到更多的信息。

祝贺您！您已经读完了入门手册，并且学习了 **STEP 7** 最重要的术语、操作步骤和功能。您现在就可以开始您的第一个项目了。

在将来的项目中，如果要查找特定的功能或者忘记了 **STEP 7** 的操作指令，您都可以使用有关 **STEP 7** 的综合帮助信息。

如果您想要进一步地扩充自己对 **STEP 7** 的了解，我们设有很多专门的培训课程。您当地的西门子代表处非常乐意为您提供帮助。

我们希望您的项目获得成功！

Siemens AG

# 附录 A

## 使用入门手册中实例项目的概述

- **ZEn01\_02\_STEP7\_\_STL\_1-10:**  
用 STL 语言编程的第 1 章到第 10 章，包括符号表。
- **ZEn01\_01\_STEP7\_\_STL\_1-9:**  
用 STL 语言编程的第 1 章到第 9 章，包括符号表。
- **ZEn01\_06\_STEP7\_\_LAD\_1-10:**  
用 LAD 语言编程的第 1 章到第 10 章，包括符号表。
- **ZEn01\_05\_STEP7\_\_LAD\_1-9:**  
用 LAD 语言编程的第 1 章到第 9 章，包括符号表。
- **ZEn01\_04\_STEP7\_\_FBD\_1-10:**  
用 FBD 语言编程的第 1 章到第 10 章，包括符号表。
- **ZEn01\_03\_STEP7\_\_FBD\_1-9:**  
用 FBD 语言编程的第 1 章到第 9 章，包括符号表。
- **ZEn01\_07\_STEP7\_\_Dist\_IO:**  
使用分布式 I/O 编程的第 11 章。



# 索引

## 字母

AND 功能	1-1
CPU, 打开	7-5
DP 主站系统, 组态	11-4
OR 功能	1-1
SIMATIC, 其它软件	2-6
SIMATIC 管理器, 项目结构	2-4
SIMATIC 管理器, 启动	2-1
SIMATIC 管理器中的项目结构	2-4
SR 功能	1-2

## A

安装	1-5
----	-----

## B

帮助, 调用	2-5
背景数据块, 生成	5-14
编程定时器功能	8-4
编程多重背景	10-1
编程, 符号	3-2
编程共享数据块	9-1
变量表, 创建	7-8
变量, 监视	7-10
变量表, 切换到在线方式	7-9
变量, 修改	7-10
变量声明表中的共享数据块	9-3

## C

操作模式, 检查	7-5
程序, 下载到可编程控制器	7-3
创建变量表	7-8
创建功能	8-1
创建功能块	5-1
创建共享数据块	9-1
创建项目	2-1
创建一个带有功能块和数据块的程序	5-1

## D

打开功能	8-1
打开功能块	5-1
打开共享数据块	9-1
调用帮助	2-5
调用功能	8-6
对功能(FC)进行编程	8-1
多重背景, 编程	10-1

## F

分布式 I/O, 组态	11-1
符号编程	3-2
符号编辑器	3-2
符号表	3-2
符号表中的共享数据块	9-3
复位 CPU 并切换到 RUN	7-3

## G

功能, 创建	8-1
功能, 打开	8-1
功能, 调用	8-6
功能块, 创建	5-1
功能块, 打开	5-1
功能块, 用功能块图编程	5-10
功能块, 用梯形图编程	5-3
功能块, 用语句表编程	5-7
功能块图, 调试	7-6
功能块图, 块调用	5-21
功能块图中的块调用	5-21
功能块图, 编程定时器功能	8-5
共享数据块, 编程	9-1
共享数据块, 创建	9-1
共享数据块, 打开	9-1

## J

节点地址, 修改	11-6
介绍 STEP 7	1-1
接通电源	7-3
建立一个在线连接	7-1
监视变量	7-10
将变量表切换到在线方式	7-9
绝对地址	3-1

**M**

模块信息, 查询 ..... 7-12

**P**

配置硬件 ..... 6-1, 7-1

配置中央机架 ..... 6-1

评估诊断缓冲区 ..... 7-12

**Q**

启动 SIMATIC 管理器 ..... 2-1

**S**

声明变量

    FBD ..... 5-10

    LAD ..... 5-3

    STL ..... 5-7

实际值, 更改 ..... 5-14

使用 STEP 7 的步骤 ..... 1-4

数据块, 生成背景数据块 ..... 5-14

数据类型 ..... 3-3

**T**

梯形图, 块调用 ..... 5-16

梯形图, 调试 ..... 7-6

梯形图, 编程定时器功能 ..... 8-3

**X**

下载程序到可编程控制器 ..... 7-3

项目, 创建 ..... 2-1

项目结构, 浏览 ..... 2-6

修改变量 ..... 7-10

修改节点地址 ..... 11-6

**Y**

用功能块图编程定时器功能 ..... 8-5

用梯形图编程定时器功能 ..... 8-3

用语句表编程定时器功能 ..... 8-4

语句表, 用 PROFIBUS DP

    组态分布式 I/O ..... 11-1

用功能块图编程 FB1 ..... 5-10

用梯形图编程 FB1 ..... 5-3

用梯形图进行块调用 ..... 5-16

用语句表编程 FB1 ..... 5-7

语句表, 块调用 ..... 5-19

语句表中的块调用 ..... 5-19

硬件, 配置 ..... 6-1

用功能块图进行调试 ..... 7-6

用梯形图进行调试 ..... 7-6

用语句表进行调试 ..... 7-6

语句表, 调试 ..... 7-6

**Z**

在线连接, 建立 ..... 7-1

诊断缓冲区, 评估 ..... 7-12

组态 DP 主站系统 ..... 11-4

组态分布式 I/O ..... 11-1

组态网络 ..... 11-7