说明

下面为课题提示：

如何进行实验?

如何安装 Step7?

如何在网络实验室注册?

如何更该 IP 地址?

计算机和实验连接的作用?

如何进行实验?

通过互联网进行编程的 step7 程序是试验的基础。为了使实验能够顺利进行,您必须先在虚拟实验室注册.(见"实验室/计划试验人员")。您可以在那里选择您想做实验的时间。

如何安装 Step7?

在 pdf 文件中将为您详细介绍如何安装 Step7。

如何在虚拟实验实注册?

用户名和密码是进入实验室工作所必需的。您可以通过网络或是亲自注册获得它们。可以使用 Applet 的注册方式：

Applet 注册：

输入用户名和密码，点击“connent”连通。

您的用户名和对应的 IP 地址将在第一个表格中显示出来。点击输入您个人信息的那一行，它将被启动。（这一步不能省略！）

如果点击“list destinations”，试验将被显示。您可以选择感兴趣的进行编程。

如果要进入数据库，“list sources ”和“list destinations”两行，一定要有颜色的。否则，您将不能成功登陆。最后点击“list times”。

用<<,<,> 和>>健，您可以看见是否已有人在数据库中。如果您想在余下的时间里安排自己的时间，请选择这个时间。点击“send”你的时间将被存入数据库中。您可以先后安排不同的时间。（每次都要选择和点击“send”）。

如何更该 IP 地址?

小应用程序 IP-Change :

如果有了固定地 IP 地址，就不要随便更改。学生公寓和校园里的每一台电脑都有固定的 IP 地址。

如果用其它的电脑登陆，每次上网时都得到一个新的动力 IP 地址。因此，需要使用小应用程序 IP-Change 。

请注意以下几点:

在上网完毕后，应按照下面方法找出自己的 IP 地址:

点击“Start”，再点“Ausführen”。在输入栏中输入命令“cmd" 。DOS 窗口被打开。输入命令 ipconfig.您将看到目前您的 IP 地址。

接下来可以使用小应用程序 IP-Change .

输入用户名和密码然后点击 *verbinden* .

用户将看到自己的用户名和原始的 IP 地址。

在输入栏中输入新的 IP 地址，点击 *neue IP Adresse speichern*。同时新的地址将被存入数据库中。如果地址输错，还可以重新进行保存。

为了能应用 Step7 对一定的实验课题进行编程,还需要在 Step7 中输入新的 IP 地址。

注意：如果暂时退出，一段时间后又从新登陆。 这样还会得到一个新的 IP 地址。所以，在有限的时间内最好一直保持在线状态。

注册 IP 更改 账号申请

注册

IP 更改

用户名

使用用户名和密码登记，是进行试验的前提。您可以通过 E-mail 注册或是直接注册来获得它们。

通过 E-mail 注册，所需的个人信息：

名字

姓

E-mail 地址

计算机的 IP 地址

如果您有固定的 IP 地址，可以直接填写上面的表格。如果您的 IP 地址是不固定的，请随意填写一个地址。之后，您可以用 小应用程序 IP-Change 来进行更改。

Hochschule Mittweida (FH) 自动化专业以网络为基础的教学模式

这里第一次提供以物理 PLC 为基础通过网络来编程和操纵的可能性。

继续

内容提要

实验 1：7 段显示
简短描述：

为了能够操纵由 0-F 来表示的 4 位解码器，必须编写相应的 SIMATIC S7 程序。

回顶部　　点击

试验 2：工地的交通灯
简短描述：
公路进行扩建，车辆只能使用一个车道。需要安装交通灯提醒过往车辆。

回顶部　　点击

试验 3：两点控制器
简短描述：
可以通过 两点控制器进行对容器中水量的测量。实验装置通过模拟容器模型实现。
回顶部　　点击

试验 4：PI(D)-控制器
简短描述：
可以通过 PI(D)-控制器进行对容器中水量的测量。实验装置通过模拟容器模型实现。

回顶部　　点击

试验概况　返回

重要提示！

使用硬件配置中所提供的组成配件正确组装是非常重要的。同时交流处理器要按要求进行配置。
您必须使用固定定义的开关，标记符和标记词。
在硬件被储存和翻译后，就不允许在 S7 中按装硬件！
如忽视上述提示，与 PLC 连接的互联网络将被中断，需要一位本地主管才能从新接通。
实验概况　返回　继续

试验概况　　返回

试验内容
可以应用三种基础编程语言 （LAD 标准语言梯形逻辑，STL 语句表或 FBD 功能块图。)在 SIMATIC S7 中编写程序，对 7 段显示进行操纵。

7 段显示的技术图示：

4 位译码器可以通过向前和向后从 0 到 F 调节。通过标记符新的数值可调对。

实验概况　返回　继续

内容索引

硬件配置

机架：

机架0，插槽1

简称：PS 307 2A
序号：6ES7 307- 1BA00 -0AA0
名称：PS 307 2A
宽度：1

顶部
实验概况 返回 继续

机架0，插槽2

简称：CPU 315-2DP
序号：6ES7 315-2AF03-0AB0
名称：CPU 315-2 DP
宽度：1
MPI-地址：23
调制速率：187.5 kBit/s

顶部
实验概况 返回 继续

机架0，插槽3 界面X2

简称：DP –Master
序号：- - - - - -
名称：DP-Master
宽度：1
23
- - - - - -
地址
入口
开始：1023
结束：1023

顶部
实验概况 返回 继续

机架0，插槽4

简称：DI8/DO8xDC24V/0,5A
代号：6ES7 323-1BH01-0AA0
名称：DI8/DO8DC24V/0.5A

数字通道:8 个入口
8 个出口
宽度：
- - - - - -
地址
入口
开始:0
结束:0
出口：
开始：0
结束：0

顶部
实验概况 返回 继续

机架 0，插槽 5
简称：CP 343-1 TCP
代号：6GK7 343-1EX00-0XE0
名称：CP 343-1 TCP
地点
站点：SIMATIC 300(1)
宽度：1
MPI-地址：24
MPI-网络名称：MPI（1）
网络
网络类型：Ind. Ethernet
网络名称：Ethernet（1）
IP 地址：141.55.129.2
子网络掩码： 255.255.252.0
路线地址：141.55.129.254
标准路线：是
MAC-地址：08.00.06.01.00.07 地址
- - - - - -
地址
入口
开始:272
结束:16
出口：
开始：272
结束：16

顶部
实验概况 返回 继续

标记符：
注意，下面的数值再 SPS 程序中将用到！

M1.0	开关 1	向前的数值
M1.1	开关 2	向后的数值

顶部
实验概况 返回 继续

实验概况 返回

实验过程

给可调变量赋值：

输入的变量	**使用设备**	**逻辑赋值**
增量器	M1.0	开关 1“开” M1.0=1
减量器	M1.1	开关 2“开” M1.1=1

输出变量

段 a	A0.0	数字输出 0..1
段 b	A0.1	数字输出 0..1
段 c	A0.2	数字输出 0..1
段 d	A0.3	数字输出 0..1
段 e	A0.4	数字输出 0..1
段 f	A0.5	数字输出 0..1
段 g	A0.6	数字输出 0..1

顶部

步骤 1：

创建一个 Step7 项目，然后在 CPU 315-2DP 中安装硬件设备。
被您确定参数的硬件不能随意下载到控制器中，您一定要使在硬件配置中为您提供的组成器件。
任何改动都会导致试验的失败。
图示：（全屏模）
硬件配置
MPI 的介绍
信息处理机

帮助：

您可以在西门子公司提供的教育材料 的 24 页起,找到关于自动化 SIMATIC S7-300 的相关资料。
在模 4 的第六页您可以找到关于 STEP 7 项目的详细描述。您同样可以看到对 CPU 315-2DP 硬件配置的介绍。
参考书：

- Hardware konfigurieren und Verbindungen projektieren mit STEP 7
- Erste Schritte und Übungen mit STEP 7

顶部

步骤 2：

对实验的解答，存在多种方法。

参考答案是通过 AWL 语言所编写的程序，使用 KOP 和 FUP 语言所用的程序当然也不同。
在 AWL 语言中，已经保存的二进制数位组合的每个数字以全程数据组合的形势被使用。
因此，在使用 AWL 语言编程时，要在所在的项目中添加一个数据库。
在数据组合中，二进制数位组合可按任意顺序保存，以便在 OB1 也可以调用这个数据。
数据组合是通过在实验项目中先选择文件夹组合元件，然后通过Einfügen -> S7-Bausteine -> Datenbausteine 而产生的(图)。在下面的窗口中能找到数据组合的号码然后点击OK 确认(图)。
图示：（全屏模）
组合结构
帮助：
在模 B4西门子教育材料中，将对数据组合做详细介绍。
在此第 10 页中，将对数据组合的其它可能性作以说明。
参考书：
Programmieren mit STEP 7

顶部

步骤 3：
在硬件组装完毕，缺少的组件组合后。现在可以对组件进行编程。
OB1:
在 OB1 中向前和向后的计数器将被启动（见模 III 第 20 页）。要避免超值。最低数值不能够准确读出，因为计数器不能被调到小于 0 的位置（见模 III 第 20 页）。
在 AWL 中通过调用数据库（见模 III 第 20 页）来下载二进制数位组合最后传送给输出字节。
在 KOP 和 FUP 中计数器的状态将被直接衡量（见模 III 第 26 页）相应的二进制数位组合被直接传送给输出字节。
参考答案：（全屏模）
OB 1 在 AWL
OB 1 在 KOP
OB 1 在 FUP
DB1：（在 AWL 中）
这里合适的二进制数将以具代表性的数字被保存（见模型 B4 第 13 页）。在前面给出的答案中将以十六进制的形式被保存。
如果之后对数据组合中的数值做改变，要切入到数据视图才能更改（见模型 B4 第 15 页）。
参考答案：（全屏模）
DB 1
参考书：

- Erste Schritte und Übungen mit STEP 7
- Programmieren mit STEP 7
- Anweisungsliste (AWL) für S7-300/400
- Kontaktplan (KOP) für S7-300/400

- Funktionsplan (FUP) für S7-300/400

当对所有组合元件编程完毕，可进行一般的保存，和重新读取。

顶部

步骤 4：
当您对组合元件的编程完成后，可以将之下载到操作系统中。接下来可以验证它的正确性。
有两种方法。可以通过 SIMATIC 的软件中的变量表格，或是使用这里提供的使用工具。
为了确保正确性，可以两种方法兼用。
方法 1：
在 STEP 7 中提供多种测试和分析的方法。
如何调用和正确使用将在Modul A7 ab Seite 6 进行详细介绍。
使用变量表格可以对您项目中的标记符进行观察和控制。

方法 2：
使用这里提供的工具。
您可以设置和删除您的标记符，还可以观察其在输出时的状态。
使用手册：

- Programmieren mit STEP 7

还用一种可能性，通过 WebCams 可以随时观察您的实验过程。

WebCam 1　　WebCam2

实验概况 返回　　回顶部

实验内容
由于施工，去工厂的车辆只能在公路的一个车道行驶。因此，需要安装交通灯来指挥交通。打开装置（开关 S0）两个交通灯应同时显示红色。启动启始器（J1 或着 J2)，相应的灯应该在十秒钟后变绿。
绿灯应该至少持续 20 秒钟，在点击另一个起始器使两个绿灯再次变红。10 秒钟之后，绿灯再次亮起。
如果不在使用启始器，路灯将保持最后的状态。只能在绿灯后，才能关闭路灯。

实验示意图：

内容索引
插槽 1
插槽 2
插槽 4

插槽 5
M1.0
M1.1
M1.2
硬件配置
机架 0，插槽 1
简称：PS 307 2A
序号：6ES7 307-1BA00-0AA0
名称：PS 307 2A
宽度：1
实验概况　　返回　继续

机架 0，插槽 2
简称：CPU 314
序号：6ES7 314-1AF10-0AB0
名称：CPU 314
宽度：2
MPI-地址：21
调制速率：187.5 kBit/s
实验概况　　返回　　继续

机架 0，插槽 4
简称：DI8/DO8xDC24V/0,5A
序号：6ES7 323-1BH01-0AA0
名称：DI8/DO8xDC24V/0.5A
数字通道:　8 个入口
　　8 个出口
宽度:1
- - - - - -
地址
入口
开始:0
结束:0
出口：
开始：0
结束：0
实验概况　返回　继续

机架 0，插槽 5
简称：CP 343-1 IT
序号：6GK7 343-1GX11-0XE0
名称：CP 343-1 IT
地点
站点：SIMATIC 300(1)
宽度：1

MPI-地址：22
MPI-网络名称：MPI（1）
网络
网络类型：Ind. Ethernet
网络名称：Ethernet（1）
IP 地址：141.55.129.2
子网络掩码：255.255.252.0
路线地址：141.55.129.254
标准线路：是
MAC-地址：08.00.06.01.00.08
地址
入口
开始:272
结束:287
出口：
开始：272
结束：287
实验概况　返回 继续
标记符
以下标记符一定要在 SPS 程序中使用。
M1.0 开关 0　　开关　开/关
M1.1 启始器 1　　启始器 1 开/关
M1.2 启始器 2　　启始器 2　　开/关

试验概况　返回 继续

试验概况　　返回
内容索引：

实验过程
给可调变量赋值：
输入的变量　使用设备　逻辑赋值
开关 0　　M1.0　　开关 1“开”　M1.0 =1
起始器 1　　M1.1　　起始器“开”M1.1=1
起始器 2　　M1.2　　起始器“开”M1.2=1
输出变量
绿灯 1　　A0.0　　数字输出 0
绿灯 2　　A0.1　　数字输出 0
红灯 1　　A0.2　　数字输出 0
红灯 2　　A0.3　　数字输出 0
步骤 1：

创建一个 Step7 项目，然后在 CPU 315-2DP 中安装硬件设备。
被您确定参数的硬件不能随意下载到控制器中，您一定要使在硬件配置中为您提供的组成器件。
任何改动都会导致试验的失败。
图示：（全屏模）
硬件配置
MPI 的介绍
信息处理机

帮助：
您可以在西门子公司提供的教育材料 的 24 页起,找到关于自动化 SIMATIC S7-300 的相关资料。
在模 4 的第六页您可以找到关于 STEP 7 项目的详细描述。您同样可以看到对 CPU 315-2DP 硬件配置的介绍。
参考书：

- Hardware konfigurieren und Verbindungen projektieren mit STEP 7
- Erste Schritte und Übungen mit STEP 7

顶部

步骤 2：
从试验内容来看，程序 生成一个有限状态机。
接下来想清楚交通灯是自动操作还是通过优先电路来控制。
先从优先电路来看。
值得注意的是，输入的条件要相互联系。有 4 种不同的开始条件：
• 在 J1 和 J2 处有一辆车(a)
• 只有一辆车在 J1 处(b)
• 只有一辆车在 J2 处(c)
• 在 J1 和 J2 都没有车(d)
如果条件 J1 和 J2 的同时被满足，将出现先行状态（比如，M10.2）。
下面 M 10.4 和 M 10.7 的状态 一定要统一。我们提供的答案是 M10.0 具有优先权。
结果将生成一个顺序操作系统，通过此系统从一步到下一步的分级，由每步的条件而定。从中将生成下面状态图示：
这只是其中的一种可能性，被我们采用做为参考答案。
帮助：
在Modul C1西门子教育材料对此进行详细地介绍。
顶部
步骤 3：
在硬件组装完毕后，您将对程序的结构有清楚的了解，这是就可以开始编程了。
OB1:
在 OB1 中包含所有的程序。
它们将以单一的形式被分开，每一步按照状态图示而生成。
请注意，在这里使用 S5-Time,因为如使用旧的系统，新的 IEC-clock (1ms)会导致一些问题。

延长脉冲的计时器“SV”被使用（见TIA Module III 17 页 ）。因为，编程时在“1”停留的时间的输出信号是不受输入信号在“1“停留时间的影响。
参考答案：（全屏图）
OB1 在 AWL
OB1 在 KOP
OB1 在 FUP
参考书：

- Erste Schritte und Übungen mit STEP 7
- Anweisungsliste (AWL) für S7-300/400
- Kontaktplan (KOP) für S7-300/400
- Funktionsplan (FUP) für S7-300/400

当对所有组合元件编程完毕，可进行一般的保存，和重新读取。

顶部

步骤 4：
当您对组合元件的编程完成后，可以将之下载到操作系统中。接下来可以验证它的正确性。
有两种方法。可以通过 SIMATIC 的软件中的变量表格，或是使用这里提供的使用工具。
为了确保正确性，可以两种方法兼用。
方法 1：
在 STEP 7 中提供多种测试和分析的方法。
如何调用和正确使用将在Modul A7 ab Seite 6 进行详细介绍。
使用变量表格可以对您项目中的标记符进行观察和控制。

方法 2：
使用这里提供的工具。
您可以设置和删除您的标记符，还可以观察其在输出时的状态。
使用手册：

- Programmieren mit STEP 7

还用一种可能性，通过 WebCams 可以随时观察您的实验过程。

WebCam 1　　WebCam2

实验概况 返回　　回顶部

试验 3：两点控制器
简短描述：
可以通过 两点控制器进行对容器中水量的测量。容器器件模拟型模实现本试验装置。

试验任务
过两点控制器来实现对一个模拟容器得容量控制。实验者可以通过标记符和标记数字对电子和模拟入口的大小进行调节，来实现以网络为基础的对试验过程的控制和观察。
调制器线路的技术图解：
请把控制变量连接到数字输入的阀门上（Q1)。 当前的数值（输入批 PIW290)显示水位的高低，被看作是模型的模拟数值。再 PLC 中使用标记数字 10（MW10) 作为定位点。
下面的表格显示对模拟数据的处理条件，其中被测量和小数/十六进制表示形式形成的线性关系。

输入变量（测量范围）	单位（小数）	单位（十六进制）
0V	0(0%)	0
10V	27648(100%)	6C00

试验概况　　返回　　继续

内容索引

硬件配置
机架 0，插槽 1
简称：PS 307 5A
序号：6ES7 307-1EA00-0AA0
名称：PS 307 5A
宽度：1
实验概况　　返回　继续

机架 0，插槽 2
简称：CPU 315-2DP
代号：6ES7 315-2AF03-0AB0
名称：CPU 315-2 DP
宽度：2
MPI-地址：25
调制速率：187.5 kBit/s
实验概况　　返回　　继续

机架 0，插槽 3
简称：DP –Master

代号：- - - - - -
名称：DP-Master
宽度：1
专业总线地址：2
- - - - - -
地址
入口
开始：1023
结束：1023
实验概况　　返回　继续

机架 0，插槽 4
简称：DI16xDC24V
代号：6ES7 323-1BH02-0AA0
名称：　DI16xDC24V
数字通道:16 个入口
宽度: 1
- - - - - -
地址
入口
开始:0
结束:1

实验概况　　返回　继续

机架 0，插槽 5
简称：DO16xDC24V/0.5A
代号：6ES7 322-1BH01-0AA0
名称：DO16xDC24V/0.5A
电子通道：16 出口
宽度：1
- - - - - -
地址
入口
开始：4
结束：5
实验概况　　返回 继续
机架 0，插槽 6
简称：AI4/AO2x8/8Bit
代号：6ES7 334-0CE01-0AA0
名称：AI4/AO2x8/8Bit
模拟通道：4 个入口

2 个出口
宽度：1
- - - - - -
地址
入口
开始：288
结束：295
出口
开始 ： 288
结束：291
实验概况　返回 继续
机架 0，插槽 7
简称：CP 343-1 IT
代号：6GK7 343-1GX11-0XE0
名称：CP 343-1 IT
地点
站点：SIMATIC 300(1)
宽度：1
MPI-地址：26
MPI-网络名称：MPI（1）
网络
网络类型：Ind. Ethernet
网络名称：Ethernet（1）
IP 地址：141.55.129.1
子网络掩码：255.255.252.0
路线地址：141.55.129.254
标准线路：是
MAC-地址：08.00.06.01.00.08
地址
入口
开始:304
结束:16
出口：
开始：304
结束：16
实验概况　返回 继续
标记符：
以下标记符一定要在 SPS 程序中使用。
M1.0 开关 1　调节器 开/关
M1.1 开关 2　　活门 开/关
M1.2 开关 3　活门 开/关
MW10 PEW288 期望值

试验概况　返回 继续

内容索引：

输入变量	使用工具	逻辑符值
实际值	PEW 290	模拟输入 0...10V
期望值	MW 10	类似输出 0...10V
调节器“开”	M1.0	调节器“开”M1.0=1
活门 2	M1.1	活门 2 “开” M1.1=1
活门 3	M1.2	活门 3 “开” M1.2=1

输出变量

调制大小（活门 2）	A4.0	数字输出 0...1
控制器显示	A4.0	控制器“开”= A4.0

控制器的功能示意图

控制器 控制系统

调节变量 y

创建控制函数 过程

控制差分值 w-x 传感器

实际值 /期望值的比较 控制变量 x 测量传感器

调整器 顶部

两点调节器的结构图式

步骤 1：

创建一个 Step7 项目，然后在 CPU 315-2DP 中安装硬件设备。

被您确定参数的硬件不能随意下载到控制器中，您一定要使在硬件配置中为您提供的组成器件。

任何改动都会导致试验的失败。

图示：（全屏模）

硬件配置

MPI 的介绍

信息处理机

帮助：

您可以在西门子公司提供的教育材料 的 24 页起,找到关于自动化 SIMATIC S7-300 的相关资料。

在模 4 的第六页您可以找到关于 STEP 7 项目的详细描述。您同样可以看到对 CPU 315-2DP 硬件配置的介绍。
参考书：

- Hardware konfigurieren und Verbindungen projektieren mit STEP 7
- Erste Schritte und Übungen mit STEP 7

顶部

步骤 2：
在试验任务和所提供的硬件的基础上，您要对能成功的程序结构进行思考。
基本上最好是以函数为导向进行编程。程序的结构对编程人员和其同事都一目了然。
当然也可以编写线性程序，如果程序不是很复杂的情况。操作系统将对你所存入的顺序进行加工。
一旦程序变得复杂或是您想增加函数，编程过程将不明显很难继续进行。（见附录 1 30 页）
因此，从功能上应把整个程序划分成许多小的程序。
按照我们对实验任务的分析，我们把程序按照不同的功能分为 2 个子程序。每个子程序通过组合调用命令被调用。（称作xx,CC xx [目录 III 27f]
帮助：
在西门子公司教育材料的目录中，您可以找到关于自动化系统 S-300 的概况和于之相关的 STEP 7 编程软件。从第 30 页起详细介绍对线性/构造的程序加工方法。
在西门子公司教育材料 A3 中，从第 31 页有关于其它组合的介绍。
工具书：
使用STEP 7 编程Programmieren mit STEP 7

步骤 3：
在硬件安装完毕后，您一定会对程序的结构有个清楚的了解，现在就开始编程。
OB1:
在 OB1 中应该完成对干扰值（阀门 V2 和 V3）以及对单一函数(FC1 和 FC2).的函数调用的编程。因为没有参数被传送，没有变量被写入，这个函数调用是通过有条件的模块调用来实现的。
参考答案：（全屏模）
FC 在 AWL
FC1 在 KOP
FC1 在 FUP
FC2:
在 FC2 中有针对两点控制器的真实的控制算法。
模拟值被读取（模 B2 12 页)，转换门限被算出，并将被比较。（目录 III 26 页）
在 AWL 编程语言中可以同过易位操作进行减法计算。
参考答案：（全屏模）
FC 在 AWL
FC1 在 KOP
FC1 在 FUP
参考书：

- Erste Schritte und Übungen mit STEP 7
- Anweisungsliste (AWL) für S7-300/400
- Kontaktplan (KOP) für S7-300/400
- Funktionsplan (FUP) für S7-300/400

在编程结束后，可以把他们存入控制器中。

顶部

步骤 4：
当您对组合元件的编程完成后，可以将之下载到操作系统中。接下来可以验证它的正确性。
有两种方法。可以通过 SIMATIC 的软件中的变量表格，或是使用这里提供的使用工具。
为了确保正确性，可以两种方法兼用。
方法 1：
在 STEP 7 中提供多种测试和分析的方法。
如何调用和正确使用将在Modul A7 ab Seite 6 进行详细介绍。
使用变量表格可以对您项目中的标记符进行观察和控制。

方法 2：
使用这里提供的工具。
您可以设置和删除您的标记符，还可以观察其在输出时的状态。
使用手册：

- Programmieren mit STEP 7

还用一种可能性，通过 WebCams 可以随时观察您的实验过程。

WebCam 1　　WebCam2

实验概况　返回　　回顶部

试验 4：PID -控制器
简短描述：
可以通过 PID -控制器进行对容器中水量的测量。容器器件模拟型模实现本试验装置。
试验任务
通过 PID -控制器来实现对一个模拟容器得容量控制。实验者可以通过标记符和标记数字对电子和模拟入口的大小进行调节，来实现以网络为基础的对试验过程的控制和观察。
调制器线路的技术图解：
请把控制变量连接到数字输入的阀门上（Q1)。 实际的数值 PEW290(显示水位的高低)，被看作是模型的模拟数值。在 SPS 中使用 MW10 作为期望值。
控制必须通过使用系统函数块"操纵 C"被实现。下面的表格显示对模拟数据的处理条件，其中被测量和小数/十六进制表示形式形成的线性关系。

输入变量（测量范围）	单位（小数）	单位（十六进制）
0V	0(0%)	0
10V	27648(100%)	6C00

试验概况　　返回　　继续

试验 4：PI(D)-控制器

内容索引

硬件配置
机架 0，插槽 1
简称：PS 307 5A
序号：6ES7 307-1EA00-0AA0
名称：PS 307 5A
宽度：1
实验概况　　返回　继续

机架 0，插槽 2
简称：CPU 315-2DP
代号：6ES7 315-2AF03-0AB0
名称：CPU 315-2 DP
宽度：2
MPI-地址：25
调制速率：187.5 kBit/s
实验概况　　返回　　继续

机架 0，插槽 3
简称：DP –Master

代号：- - - - - -
名称：DP-Master
宽度：1
专业总线地址：2
- - - - - -
地址
入口
开始：1023

结束：1023
实验概况　　返回　继续

机架 0，插槽 4
简称：DI16xDC24V
代号：6ES7 321-1BH01-0AA0
名称：　DI16xDC24V
电子通道:16 个入口
宽度: 1
- - - - - -
地址
入口
开始:0
结束:1

实验概况　返回　继续

机架 0，插槽 5
简称：DO16xDC24V/0.5A
代号：6ES7 322-1BH01-0AA0
名称：DO16xDC24V/0.5A
电子通道：16 出口
宽度：1
- - - - - -
地址
入口
开始：4
结束：5
实验概况　返回 继续
机架 0，插槽 6
简称：AI4/AO2x8/8Bit
代号：6ES7 334-0CE01-0AA0
名称：AI4/AO2x8/8Bit
模拟通道：4 个入口
　　2 个出口
宽度：1
- - - - - -
地址
入口
开始：288
结束：295
出口
开始 ：288
结束：291
实验概况　返回 继续

机架 0，插槽 7
简称：CP 343-1 IT
代号：6GK7 343-1GX00-0XE0
名称：CP 343-1 IT
地点
站点：SIMATIC 300(1)
宽度：1
MPI-地址：26
MPI-网络名称：MPI（1）
网络
网络类型：Ind. Ethernet
网络名称：Ethernet（1）
IP 地址：141.55.129.1
子网络掩码：255.255.252.0
路线地址：141.55.129.254
标准线路：是
MAC-地址：08.00.06.01.00.08
地址
入口
开始:304
结束:16
出口：
开始：304
结束：16
实验概况　返回 继续
以下标记符一定要在 SPS 程序中使用。
M1.0 开关 1　　调节器 开/关
M1.1 开关 2　　活门 开/关
M1.2 开关 3　　活门 开/关
M1.3 开关 4　　调节器重起
MW10 PEW288 期望值

试验概况　返回 继续

试验概况　　返回

内容索引：

给可调变量赋值
控制的工作原理图
控制器的组成部分
步骤 1
步骤 2
步骤 3
步骤 4

实验过程
给可调变量赋值：

输入变量	使用设备	逻辑赋值
实际值	PEW 290	数字输入 0...10V
期望值	MW 10	数字输出 0...10V
调节器“开”	M1.0	调节器“开”M1.0=1
活门 2	M1.1	活门 2 “开” M1.1=1
活门 3	M1.2	活门 3 “开” M1.2=1
重起“调节器”	M1.3	M1.3=1

输出变量
调制大小（活门 1）PAW 288　　数字输出 0...1
调节器的工作原理图

对一个PI-控制器进行赋值：
请在实验中使用一下参数：
比例组成部分　　KP=5.8
积分时间　　TN=1.72

步骤 1：
创建一个 Step7 项目，然后在 CPU 315-2DP 中安装硬件设备。
被您确定参数的硬件不能随意下载到控制器中，您一定要使在硬件配置中为您提供的组成器件。
任何改动都会导致试验的失败。
图示：（全屏模）
硬件配置
MPI 的介绍
信息处理机

帮助：
您可以在西门子公司提供的教育材料 的 24 页起,找到关于自动化 SIMATIC S7-300 的相关资料。
在模 4 的第六页您可以找到关于 STEP 7 项目的详细描述。您同样可以看到对 CPU 315-2DP 硬件配置的介绍。
参考书：

- Hardware konfigurieren und Verbindungen projektieren mit STEP 7
- Erste Schritte und Übungen mit STEP 7

顶部

步骤 2：
因为西门子公司为我们提供了 PID-调节器的函数部分。
他们为 FB 41 (常数 _C)针对与连续控制，FB 42 (常数 _S)针对分步控制，FB 43 (PUSLSGEN)针对脉冲宽度调制。
因为我们的实验是关于 PI(D)-调节器，所以有必要把所需的函数组成部分复制到新的项目中。
它们位于程序库目录\Siemens\Step7\S7libs\Stdlib30。复制前应先打开程序库。
先选 *文件* -> *打开* 进入程序库。选择标准程序库点击 OK，接下来把 FB41 可以拖到您的项目中。
创建 OB 35，先在您的项目中部件上点击鼠标的右健，然后选择*新对象* -> *添加*，选择部件类型（图）。
通过 西门子提供的 PID-控制器工具，可以建成部件实例数据 DB 101。您所要做的是调用和（图）。确定 *PID-Control* 的参数。可以在程序库 *SIMATIC\STEP7\PID Control* 找到它。选择 *文件*-> *新建*，在下一个窗口中选项目并给对象命名为 DB101（图）。在 OK 后 DB 101 将作为部件实例数据给 FB41 赋值（图）。通过这一步你已经在您的项目里建成 BD，在他们被保存后，可以关闭窗口（图）。之后他们也将被确定参数。
图：（全屏模）
图书室
PID 控制部件
部件结构

帮助:

如何在您的项目里创建新的部件将在 Modul A3 第 35 页 5 起以 PC 为例，为例做详细介绍。创建新的部件方法雷同。

参考书：

- Programming with STEP 7 V5.1
- Standard Software for S7 - 300 and S7 - 400 PID Control

步骤 3：

在硬件配置后和缺少的组合分程序生成后或是从图书室把它们复制后，就可以进行编程。

OB1:
在 OB1 中应该完成对干扰值（活门 V2 和 V3）的编程，以及对期望值的读取和换算。在控制器组成原件中所期待的数的类型(真实)必须正常化到 100%。

对模拟值的换算以及所需的换算规则的提示和讲解在Modul I auf den Seiten 8 und 14 中。
参考答案：（全屏模）

OB1 在 AWL
OB1 在 KOP
OB1 在 FUP

OB35:

OB 35 是一个循环中断组织组合，每隔 100ms 将被调用一次。在 OB1 中换算的期望值和实际值在这里被输入。
在 OB 35 中调用 FB 41 后，参数将传给控制器。分别在 OB35 中键入输入 /输出变量。这些将会确保控制器正常运行，没被使用的参数将被 DB101 中的原始值取代。如果涉及可能存在的参数时有问题，请参看 OB35, 您可以找到相关注释（注释 OB35）。

参考答案：（全屏模）

OB35 在 STL
OB 35 在 LAD
OB35 在 FBD

DB101:

用来调节 PID 控制器的所用参数，P-，I-，和 D-组成部分，将由西门子公司提供的“控制器参数确定”通过 DB101 事例数据部件进行安装。
如果想在此程序中实现参数确定，DB 101 必须在此程序中更新。通过程序管理 \SIMATIC\STEP7\PID-Control 参数确定可以调用这个工具（图）。在那里可以选择您的项目和 DB 101（图）然后在数据部分的掩码中可以给值加参数。
控制系统运行和缓慢。因此有必要使用 P- ，I-组成部分。如果您想使控制路线达到最佳状态，您必须对控制系统的传送功能有所了解。这应该不成问题：请参考下面的答案。

参考答案：（全屏模）

DB 101

参考书：

- Working with STEP 7 V5.1
- Statement List (STL) for S7 - 300 and S7 - 400 Programming
- Ladder Logic (LAD) for S7 - 300 and S7 - 400 Programming
- Function Block Diagram (FBD) for S7 - 300 and S7 - 400 Programming
- Standard Software for S7 - 300 and S7 - 400 PID Control

在编程结束后，可以把他们存入控制器中。

顶部

步骤 4：
当您对组合元件的编程完成后，可以将之下载到操作系统中。接下来可以验证它的正确性。
有两种方法。可以通过 SIMATIC 的软件中的变量表格，或是使用这里提供的使用工具。
为了确保正确性，可以两种方法兼用。
方法 1：
在 STEP 7 中提供多种测试和分析的方法。
如何调用和正确使用将在Modul A7 ab Seite 6 进行详细介绍。
使用变量表格可以对您项目中的标记符进行观察和控制。

方法 2：
使用这里提供的工具。
您可以设置和删除您的标记符，还可以观察其在输出时的状态。
使用手册：

- Programmieren mit STEP 7

还用一种可能性，通过 WebCams 可以随时观察您的实验过程。

WebCam 1 WebCam2

实验概况 返回 回顶部

SIEMENS

SIMATIC S7

SETP 7 V 5.0

使用入门

6ES7 810-4CA04-5D00

安全指南

本手册包括应该遵守的注意事项，以保证你自己的生命安全以及保护产品和所连接的设备。这些注意事项在本手册中是用警示三角形突出强调的并根据危险等级注明如下：

危险(Danger)
表示若不采取适当的预防措施的话，将造成死亡、严重的人身伤害或重大的财物损失

警告(Warning)
表示假若不采取适当的预防措施的话，将可能造成死亡、严重的人身伤害或重大的财物损失

告戒(Caution)
表示假如不采取适当的预防措施的话，可能造成轻微的人身伤害或财物损失

注意(Note)
提醒你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批准进行试运行、接地和为电路、设备和系统加装标签的人员

正确使用

注意如下：

警告
本装置及其元件只能用于产品目录或技术说明书阐述的应用，并且只能与西门子公司批准或
推荐的其它生产厂购买的装置或元件相连接
本产品只有在正确的运输、贮存、组装和安装的情况下，按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。

Siemens AG
Automation Group
Industrial Automation Systems
Postfach 4848, D-90327 Nurnberg

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，可使用梯形逻辑、功能块图或语句表。

关于这本入门手册

在本手册中，你将会了解 SIMATIC STEP 7 的基础知识，我们将向你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，你几乎可以从任意一章开始。

每一章节分为两个部分：说明部分和操作过程部分。指令开始处是一个有箭头并且可能分布在几页中，结束处是一个停止符号，有一个方框，包含相关的主题。

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

STEP 7 的培训课则为您提供入门手册之外的更深入的知识，教你如何用 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，你需要以下各项：

- 一个西门子编程设备或一个 PC
- STEP 7 软件包和授权盘
- 一个 SIMATIC S7-300 或 S7-400 可编程控制器

  (用于第七章“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

在你安装了 STEP 7 之后，在 Start 菜单下的 Simatic>S7 Manuals 可以找到电子手册，或者从西门子的任何一个销售中心定购它们。手册中的所有信息都可以在 STEP 7 中从在线帮助中调出。

祝好运!

SIEMENS AG

目录

1 介绍 STEP 7

1.1 您将学到的内容

使用实际的练习，我们将向您显示使用 STEP 7 的梯形逻辑、语句表或功能图编程是多么的容易。

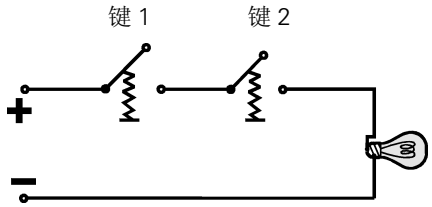
各章中详细的指导将逐步地为您显示使用 STEP 7 的诸多方法。

用二进制逻辑生成一个程序

在第二至第七章中，您将使用二进制逻辑生成一个程序。使用已编程的逻辑操作，可以寻址你的 CPU 的输入和输出(如果存在)。

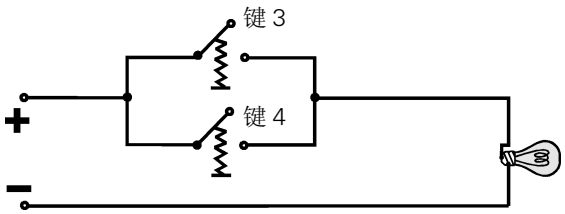
入门手册中的编程示例主要基于三个基本的二进制逻辑操作。

一会儿您将编程使用的第一个二进制逻辑操作是 AND(与)功能。AND 功能可通过下面使用两个按键的电路图作极好的说明。



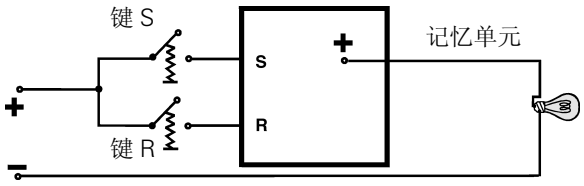
如果键 1 和键 2 都被按下则灯被点亮。

第二个二进制逻辑操作是 OR(或)功能。OR 功能可由以下电路图来表达。



如果键 3 或键 4 被按下则灯被点亮

第三个二进制逻辑操作是记忆单元。在一个电路内 SR 功能对某一电压状态作出响应并传递这一状态。

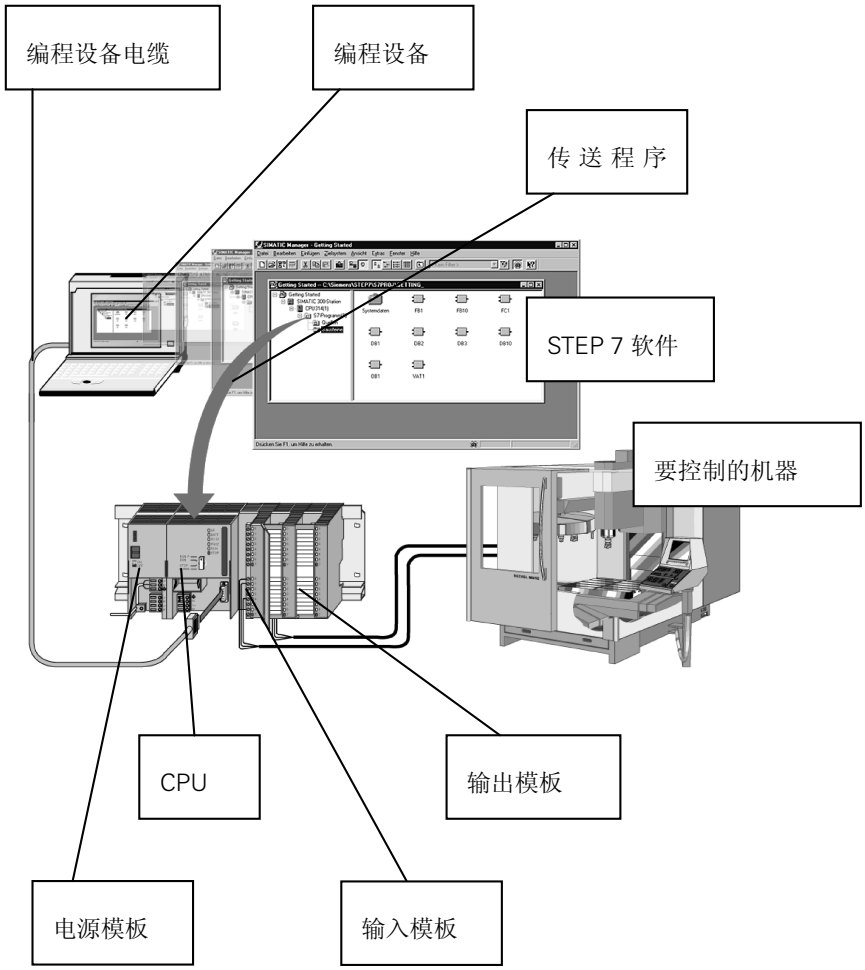


如果按下 S 键则灯被点亮并且一直保持直到按下 R 键

1.2 组合硬件和软件

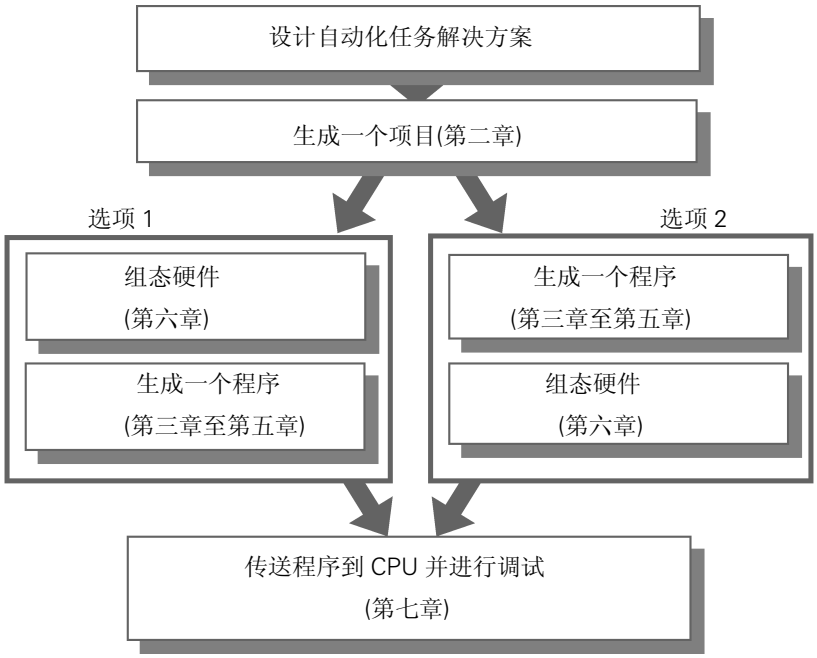
使用 STEP 7 软件，可以在一个项目下生成你的 S7 程序。S7 可编程控制器包括一个供电单元，一个 CPU，以及输入和输出模板(I/O 模板)。

可编程逻辑控制器(PLC)用 S7 程序监视和控制你的机器。在 S7 程序中通过地址寻址 I/O 模板。



1.3 使用 STEP 7 的基本步骤

在生成一个项目之前，你应该了解 STEP 7 项目可按不同的顺序生成。



如果你要生成一个使用了许多输入和输出的综合程序，我们建议您先作组态硬件。这样作的优势在于 STEP 7 在硬件组态编辑器中显示可能的地址。

如果你选择第二选项，则要自己决定每一个地址，只能依据你所选的组件，而不能通过 STEP 7 调入这些地址。

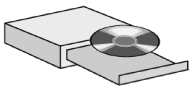
在硬件组态中，你不仅可以定义地址，还可以修改模板的参数和特性。例如，如果要操作几个 CPU，则必须区分各个 CPU 的 MPI 地址。

由于在入门手册中我们只使用少量的输入和输出，我们可以暂时跳过硬件组态从编程开始。

1.4 安装 STEP 7

无论您想从编程开始还是想从组态硬件开始，首先你必须安装 STEP7。如果您使用的是 SIMATIC 的编程设备，则 STEP 7 已经安装。

在编程设备或 PC 上安装 STEP 7 软件时，如果该设备没有以前安装的 STEP 7 版本，则要注意 STEP 7 安装对软件和硬件的要求。这些要求可以在 STEP 7 CD 的 Readme.wri 文件中找到，该文件所在的路径为：〈**驱动器**〉：**/STEP 7/Disk1**



如果你需要首先安装 STEP 7，现在可以将 STEP 7 CD 插入到 CD-ROM 驱动器，安装程序自动启动，跟随屏幕上的指令进行。

如果安装程序没有自动启动，可在 CD-ROM 的以下路径中找到安装程序〈驱动器〉：/STEP 7/Disk1/setup.exe

一旦安装完成并已重新启动计算机，“SIMATIC Manager (SIMATIC 管理器)”的图标将显示在你的 Windows 桌面上。

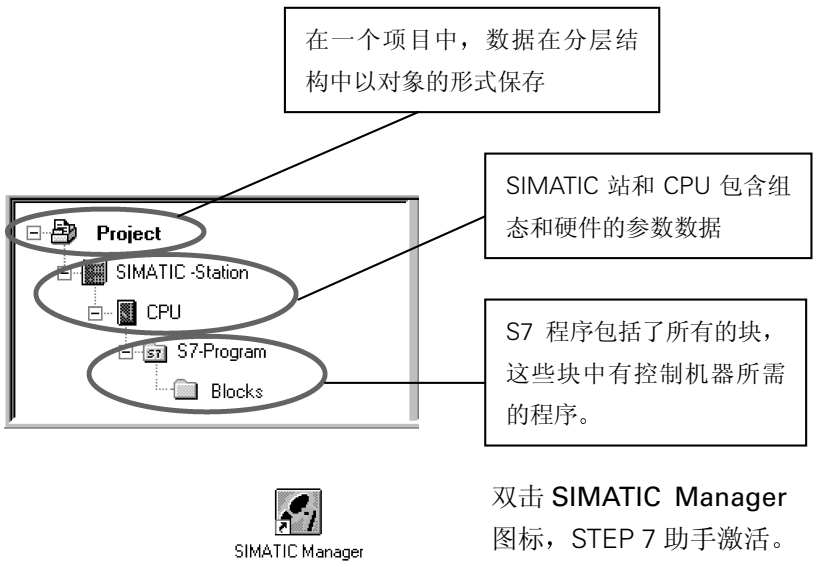
如果在安装之后，双击“SIMATIC Manager”图标，STEP 7 助手将被自动启动。

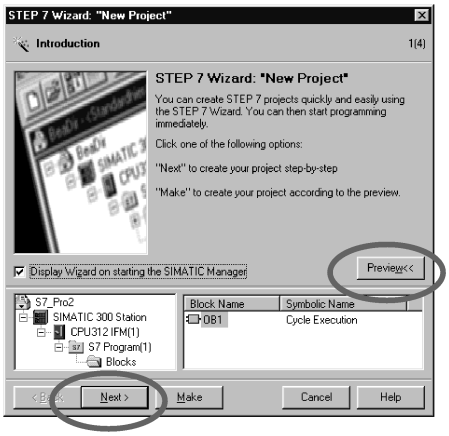
在 STEP 7 CD 的 Readme.wri 文件中可以找到关于安装的其它提示。
〈**驱动器>:/STEP 7/Disk1/Readme.wri**

2 SIMATIC 管理器

2.1 启动 SIMATIC 管理器并创建一个项目

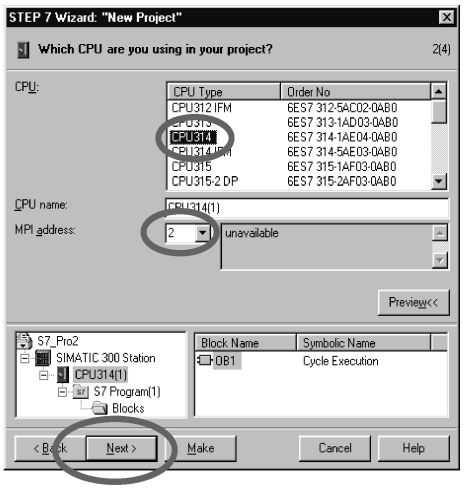
SIMATIC 管理器是一个中央窗口，STEP 7 启动时激活。缺省设置启动 STEP 7 助手，它可以在创建项目时给你支持。项目结构用来以一定的顺序保存和排列所有数据和程序。





在 **Preview** 中你可以将创建的项目结构的视图在 on 和 off 之间切换。

要转到下一个对话框，点击 **Next**.

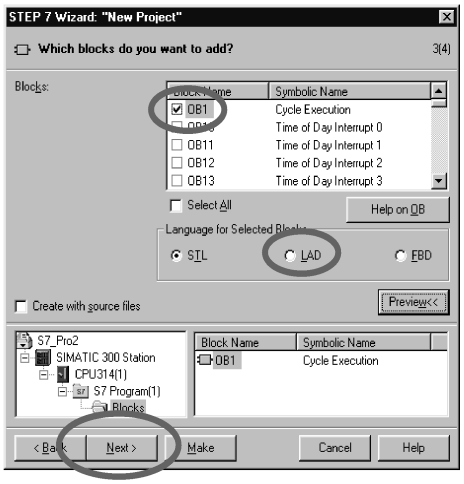


对于“入门”样板项目，选择 CPU314。你可以按照示例创建的方式在任何时候实际地选择你所得到的 CPU。

缺省设置 MPI 地址为 2。

点击 **Next** 确认设置并进入下一个对话框。

> 每个 CPU 有特定的特性；例如，关于它的存储器组态或地址区域。这也是为什么在你编程之前要选择 CPU。
>
> 需要 MPI 地址（多点接口）是为了你的 CPU 与编程设备或 PC 通讯

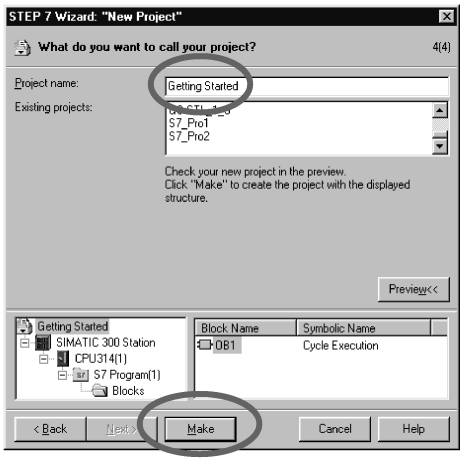


选择组织 **OB1**(如果尚未选中)。

选择以下编程语言之一：梯形逻辑 **(LAD)**、语句表 **(STL)** 或功能块图 **(FBD)**。

用 **Next** 确认你的设置。

> OB1 代表最高的编程层次，并组织 S7 程序中的其它块。
>
> 以后还可以重新改变编程语言。



在“Project name(项目名)”区域中用双击选中推荐的名字并用“Getting Started(入门)”重写。

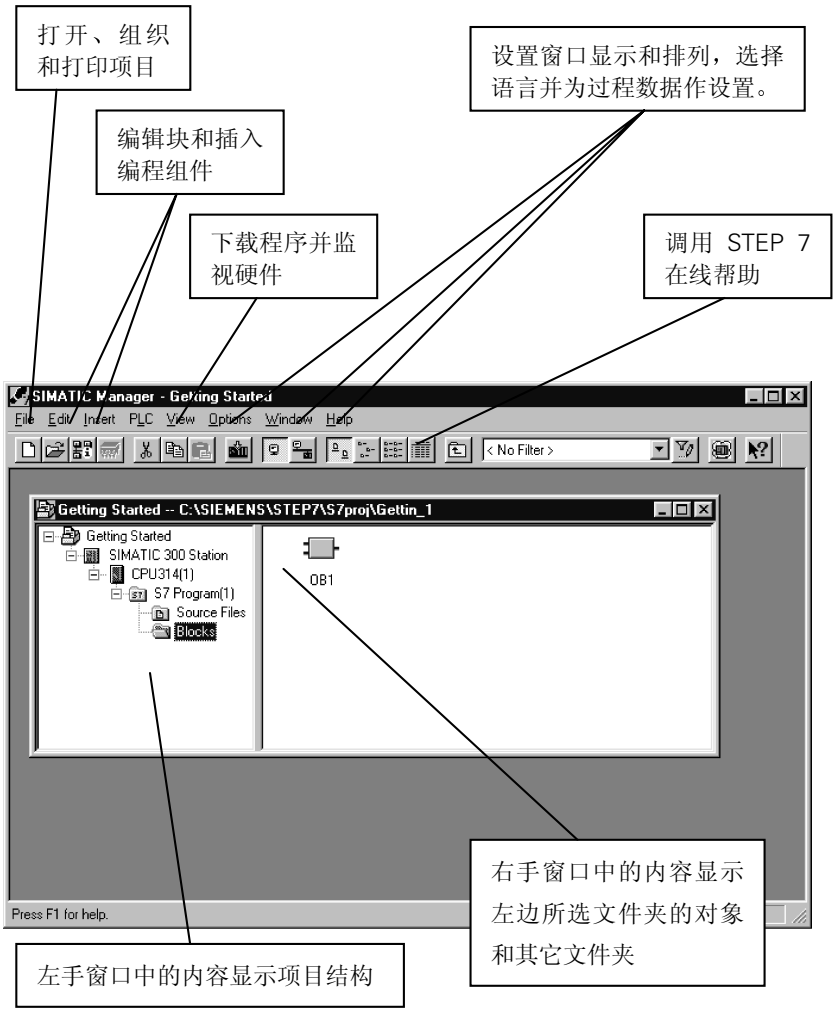
点击 **Make** 如前面预演的那样生成你的新项目。

当点击 **Make** 按钮时，SIMATIC 管理器连同已生成的“Getting Started”项目窗口一起打开。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。每次程序启动时 STEP 7 助手将被激活。你可以在助手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 助手，要创建项目则必须在项目中自己创建每一个目录。

在 **Help>Contents** 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

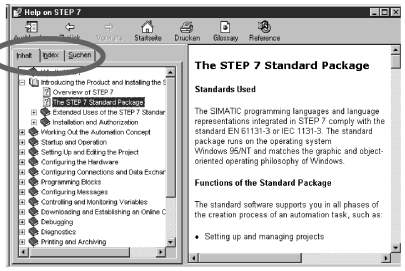
STEP 7 助手一关闭，就出现 SIMATIC 管理器及打开的“Getting Started”项目窗口。从这里可以启动所有的 STEP 7 功能和窗口。



调用 STEP 7 中的帮助

F1 选项 1：

将鼠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帮助



选项 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手窗口中，而所选主题的内容显示在右手窗口中。

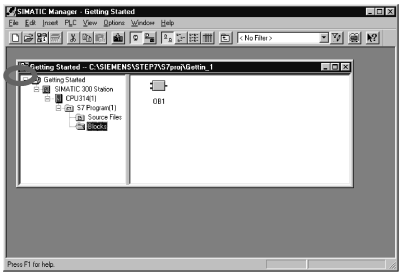
通过点击目录列表中的+号可以找到你想要的主题。同时，所选主题的目录显示在右手窗口中。

使用 **Index(索引)和 Find(查找)**，你可以输入字符串查找所需的特定的主题。

选项 3：

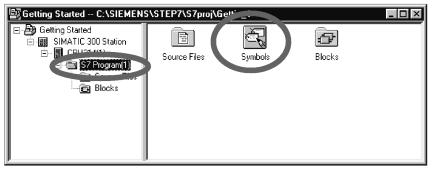
点击工具栏中的问号按钮，你的鼠标转换为帮助光标。下次当你点击一个特定的对象时，在线帮助就启动了。

Navigating in the Project Structure



你已创建的项目连同所选的 S7 站和 CPU 被显示。

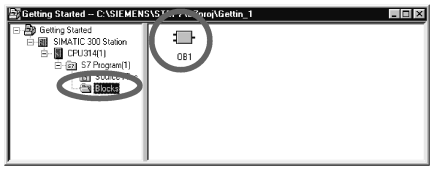
点击+或-号可打开或关闭文件夹。以后你可以点击右手窗口所显示的符号启动其它功能。



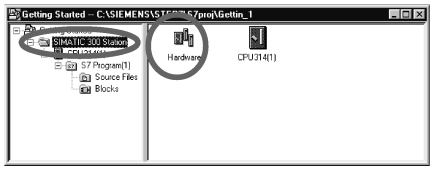
点击 **S7 Program (1)**文件夹。这里包含了所有必须的编程组件。

你将在第三章中用 Symbols(符号)给地址定义符号名。

组件 Source Files(源文件)用来保存源文件。在入门手册中将不涉及这一部分。



点击 **Blocks**(块)文件夹，这里包含你已生成的 OB1 以及以后将生成的所有其它块。



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

你将在第六章中用 Hardware(硬件)来指定你的可编程控制器的参数。

如果你的自动化任务还需要其它的 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)，这些也都集成在 STEP 7 中。例如，使用 SIMATIC 管理器，可以直接打开象 S7 Graph 功能块这些相关的对象。

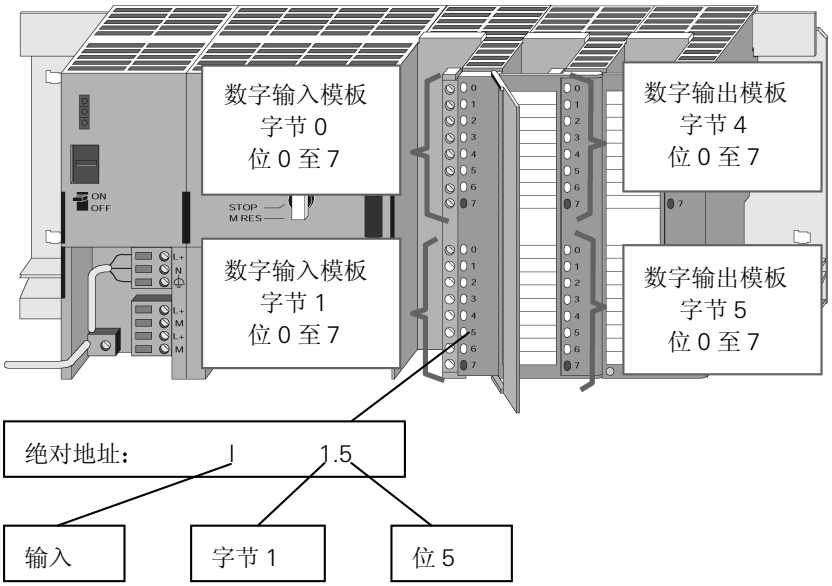
在 **Help>Contents** 下，你能够在主题“设计自动化概念”和“设计程序结构的基础”中找到更多的信息。你可以在 SIMATIC 目录 ST70 的“完全集成自动化组件”中找到更多的关于可选软件包的信息。

3 用符号编程

3.1 绝对地址

每个输入和输出都有由硬件组态预定义的一个绝对地址。该地址被直接指定，即绝对地。

该绝对地址可以由你所选择的任意符号名替代。



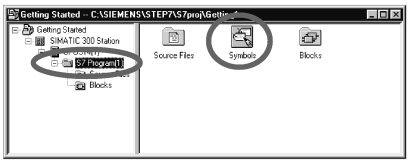
如果在你的 S7 程序中寻址的输入和输出并不多，应该使用绝对地址编程。

3.2 符号编程

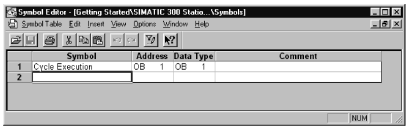
在符号表中，可以为所有的绝对地址分配符号名和数据类型，以后在用户程序中你将会寻址这些地址；例如，为输入 I0.1 指定符号名 Key1(键 1)。这些名字可以用在程序的所有部分，即是所说的全局变量。

使用符号编程可以大大地改善你已生成的 S7 程序的可读性。

使用符号编辑器



在“Getting Started(入门)”窗口中浏览，直至找到 **S7 Pragram(1)**然后双击打开 **Symbols**。



你的符号表当前只包括预定义的组织块 OB1。

	Symbol	Address	Data Type
1	Cycle Execution	OB 1	OB 1
2			

点击 **Cycle Execution** 并且“Main Pragram”作为我们的示例来进行重写。

	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“Q4.0”，数据类型则自动添加。

Comment

点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**，随后会自动增加一新行。

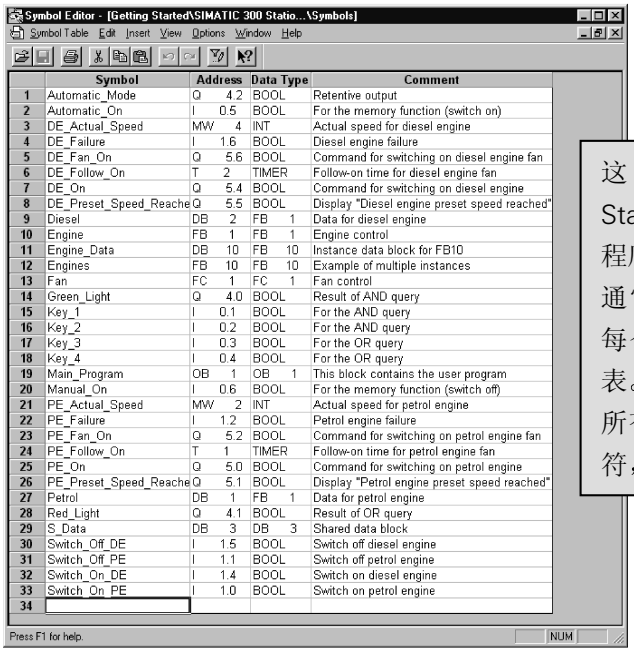
	Symbol	Address	Data Type
1	Main Program	OB 1	OB 1
2	Green Light	Q 4.0	BOOL
3	Red Light	Q 4.1	BOOL

在第三行输入“Red Light”和“Q4.1”，按回车键结束该输入项。

用这种方式可以为你的程序所需的所有输入和输出的绝对地址分配符号名

存储你在符号表中已作的输入或修改，关闭窗口。

因为在全部“Getting Started”项目中有很多名称，你可以 4.1 节中将符号表拷贝到“Getting Started”项目中。



Symbol Editor - [Getting Started\SIMATIC 300 Statio...\Symbols]

Symbol Table Edit Insert View Options Window Help

	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_Reache	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	Engine_Data	DB 10	FB 10	Instance data block for FB10
12	Engines	FB 10	FB 10	Example of multiple instances
13	Fan	FC 1	FC 1	Fan control
14	Green_Light	Q 4.0	BOOL	Result of AND query
15	Key_1	I 0.1	BOOL	For the AND query
16	Key_2	I 0.2	BOOL	For the AND query
17	Key_3	I 0.3	BOOL	For the OR query
18	Key_4	I 0.4	BOOL	For the OR query
19	Main_Program	OB 1	OB 1	This block contains the user program
20	Manual_On	I 0.6	BOOL	For the memory function (switch off)
21	PE_Actual_Speed	MW 2	INT	Actual speed for petrol engine
22	PE_Failure	I 1.2	BOOL	Petrol engine failure
23	PE_Fan_On	Q 5.2	BOOL	Command for switching on petrol engine fan
24	PE_Follow_On	T 1	TIMER	Follow-on time for petrol engine fan
25	PE_On	Q 5.0	BOOL	Command for switching on petrol engine
26	PE_Preset_Speed_Reache	Q 5.1	BOOL	Display "Petrol engine preset speed reached"
27	Petrol	DB 1	FB 1	Data for petrol engine
28	Red_Light	Q 4.1	BOOL	Result of OR query
29	S_Data	DB 3	DB 3	Shared data block
30	Switch_Off_DE	I 1.5	BOOL	Switch off diesel engine
31	Switch_Off_PE	I 1.1	BOOL	Switch off petrol engine
32	Switch_On_DE	I 1.4	BOOL	Switch on diesel engine
33	Switch_On_PE	I 1.0	BOOL	Switch on petrol engine
34				

Press F1 for help. NUM

这里你可以看到“Getting Started”示例中语句列表的 S7 程序的符号表。
通常不论你选择哪种编程语言每个 S7 程序只生成一个符号表。
所有可打印的字符(如，特殊字符，空格)都可以在符号表中使

自动填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。STEP7 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(类型 BOOL)至 32 位(DWORD)
CHAR	这种类型的数据只占有 ASCII 字集中的一个字符。
INT DINT REAL	它们用于处理数字值(例如，计算数学表达式)。
S5TIME TIME DATE TIME_OF_DAY	这种数据类型在 STEP7 中代表不同的时间和数值(例如，设定日期或为定时器输入时间值。)

在 **Help>Contents** 的主题“编程块”和“定义符号”中你可以找到更多的信息。

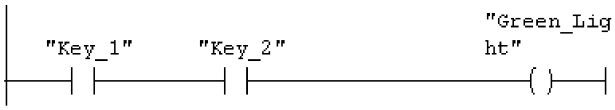
4 在 OB1 中创建程序

4.1 打开 LAD/STL/FBD 编程窗口

选择梯形逻辑、语句表、或功能块图

在 STEP 7 中，可以用标准语言梯形逻辑(LAD)，语句表(STL)或功能块图(FBD)生成 S7 程序。在实际中你必须决定使用哪种语言，在本章中也是如此。

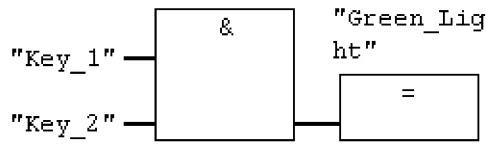
梯形逻辑（LAD）
例如：适用于来自电气工业的用户，



语句表（STL）
例如：适用于来自计算机领域的用户，

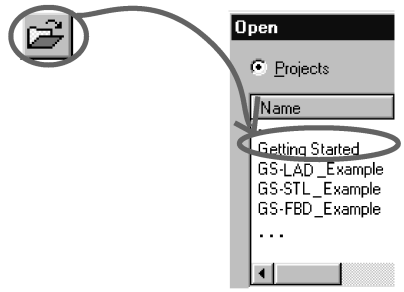
```
A      "Key_1"
A      "Key_2"
=      "Green_Light"
```

功能块图（FBD）
例如：适用于来自电路工程的用户，

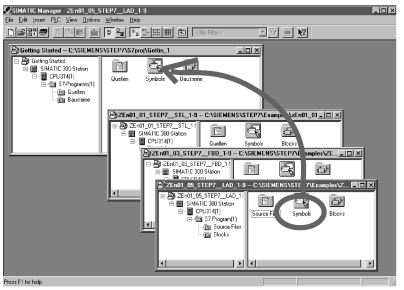


块 OB1 将以你的项目助手中生成该块时所选择的语言被打开。但是，你可以在任何时候重新修改这个缺省的编程语言。

拷贝符号表并打开 OB1



如果有必要，可打开你的“Getting Starrted”项目。要这么作，点击工具栏中的 Open(打开)按钮，选择你生成的“Getting Started”项目并用 OK 确认。



根据你所选择的要使用的语言，同时打开下列项目之一：

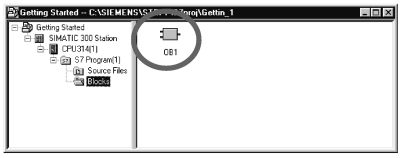
- zEn01_06_STEP 7_LAD_1-9
- zEn01_02_STEP 7_STL_1-9
- zEn01_04_STEP 7_FDB_1-9

这里你可以看到显示的所有的三个样板项目。

在“zEno1_×××”中查找直到找到组件 Symbols，用拖放功能将该符号表拷贝到你的“Getting Started”项目窗口的 S7 Pragram 文件夹中。

然后，关闭窗口“zEn01_×××”。

> 拖放功能就是用鼠标点击任意对象按住鼠标的同时移动，当松开鼠标时，对象被粘贴到所选的位置。



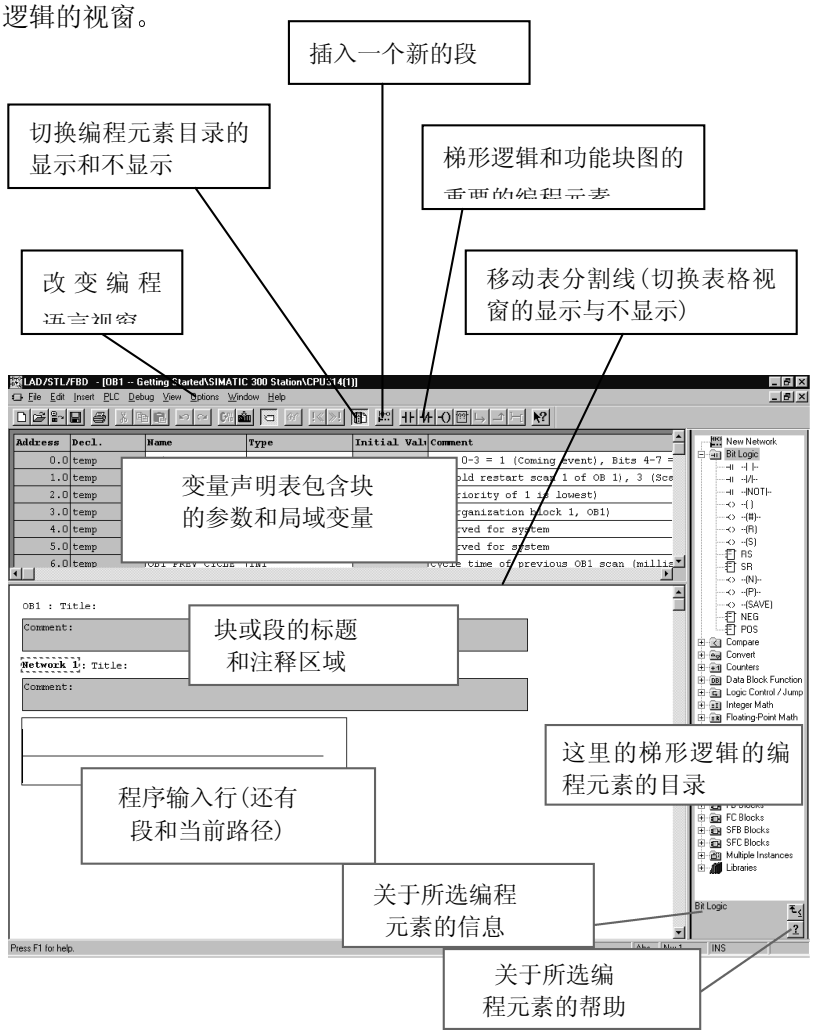
双击“Getting Started”项目中的 OB1。打开 LAD/STL/FBD 编程窗口。

在 STEP 7 中，CPU 循环处理 OB1。CPU 一行一行地读入并执行程序命令。当 CPU 返回到第一条程序时，它已完成一个循环。所需的时间即是所说的扫描循环时间。
根据你已选择的编程语言，可继续阅读 4.2 梯形逻辑编程，4.3 语句表或 4.4 功能块图。

> 在 Help>Contents 下的主题“编程块”和“生成块和库”中可以找到更多的信息。

LAD/STL/FBD 编程窗口

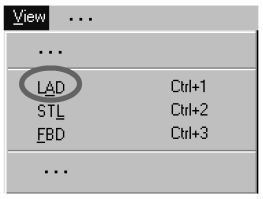
所有块都在 LAD/STL/FBD 编程窗口中编辑。这里，你可以看到梯形逻辑的视窗。



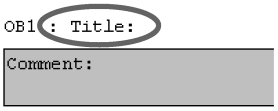
4.2 用梯形逻辑编程 OB1

在下一个部分你将用梯形逻辑(LAD)编程一个串联电路，一个并行电路以及置位/复位记忆功能。

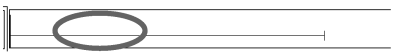
用梯形逻辑编程一个串联电路



如果有必要，在 View 菜单中设置 **LAD** 作为编程语言。



点击 OB1 的 **Title(标题)**区域并作为示例输入“Cyclically processed main program(循环处理的主程序)”。



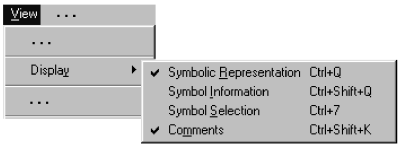
为你的第一个元素选择当前分支。

点击工具栏中的按钮并插入一个常开触点。

以同一方式插入第二个常开触点。

在当前分支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。

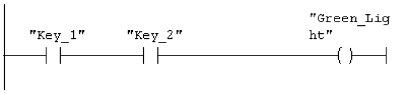


检查符号表达方式是激活的。

点击**??.?**符号并输入符号名“Key_1”(在引号中)。

用 **Enter(回车)**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”。

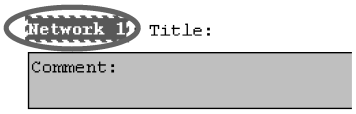


现在你已编程了一个完整的串联电路。

如果设有符号显示为红色，保存该块。

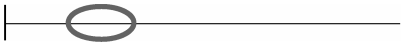
如果符号在符号表中不存在或有语法错误则该符号显示为红色。你也可以直接从符号表中插入符号名。点击??.?符号然后选择菜单命令 **Insert＞Symbol.**滚动下拉列表直到找到相应的名字并选中它。该符号名被自动加入。

用梯形逻辑编程一个并联电路



选择 Network(段)1。

插入一个新段。



再次选择当前分支

插入一个常开触点和一个线圈

选择当前分支的垂直线。

插入一个并行分支。

在并行分支上增加另一个常开触点。

关闭分支(如果有必要，可选择向下的箭头)。

在并联电路中还未输入地址。

要赋值符号地址，可按照与串联电路一样的方法进行。

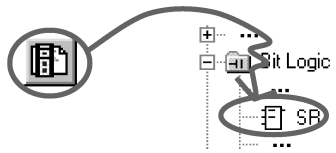
用“Key_3”重写上面的常开触点，用“Key_4”重写下面的触点，线圈则为“Red_Light”。

保存该块。

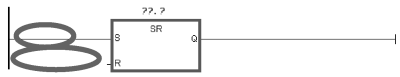
用梯形逻辑编程一个记忆功能

选中第二段并插入另一段。

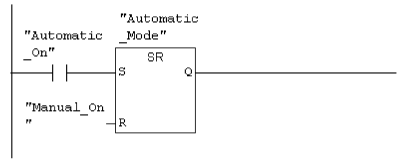
选择当前分支。



在编程元素目录的 **Bit Logic**(位逻辑)下查找直到找到 **SR**，双击插入该元素。



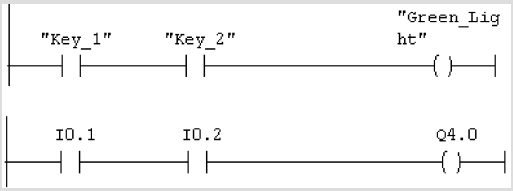
分别在 S 和 R 的输入之前插入一个常开触点。



为 SR 输入以下符号名：
上面触点“Automatic_On”
下面的触点“Manual_On”
SR 元素“Automatic_Mode”

保存该块并关闭窗口。

如果你想看一看绝对地址和符号地址之间的差别，释放菜单命令 **View＞Display＞Symbolic Representation**。



示例：
LAD 中的符号寻址

示例：
LAD 中的绝对寻址

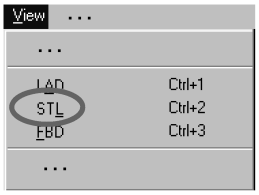
在 LAD/STL/FBD 编程窗口改变符号寻址的行断，可用菜单命令 Options＞Customize，然后选择“LAD/FBD”标签中的“Address Field Width (地址区域的宽度)”，这里，你可以将行断设为 10 至 24 个字符。

在 **Help＞Contents** 下面的主题“编程块”、“创建逻辑块”和“编辑梯形图指令”中可以找到更多的信息。

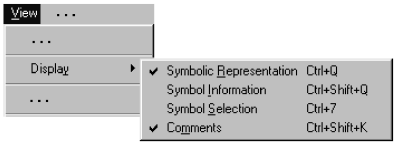
4.3 用语句表编程 OB1

下一部分中，你将用语句表(STL)编一个 AND 指令、一个 OR 指令和记忆指令置位/复位。

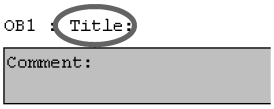
用语句表编程一个 AND 指令。



如需要，在 **View** 菜单中设置 **STL** 为编程语言。

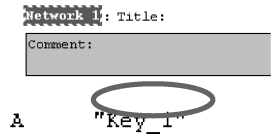


检查符号表达方式是否被激活。



点击 OB1 的 **title(标题)**区域并输入“Cyelically Processed main program”，作为示例。

为你的第一条指令选择一个区域。



在第一个程序行输入一个 A，空格，然后是符号名“Key_1”(在引号中)。

用 **Enter** 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。

现在你已编辑了一个完整的 AND 指令。
如果没有符号显示为红色，保存该块。

> 如果符号在符号表中不存在或有语法错误，则会显示为红色。
> 你还可以从符号表中直接插入符号名，点击??.?符号然后选择菜单命令 Insert＞Symbol。滚动下拉列表直至找到相应的名称并选中它，符号名则被自动添加。

用语句表编程一个 OR 指令。

选择 Network1(段 1)。

插入一个新的段并再次选择输入区域。

```
O      "Key_3"
```

输入一个 O(OR)和符号名“Key_3”(与 AND 指令方法相同)。

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

完成 OR 指令并保存它。

用语句表编程一个记忆指令

选中第 2 段然后插入另一段。

```
A      "Automatic_On"
```

在第一行中输入指令 A 和符号名“Automatic_On”。

```
A      "Automatic_On"
S      "Automatic_Mode"
A      "Manual_On"
R      "Automatic_Mode"
```

完成记忆指令并保存它。关闭该块。

如果你想看一看绝对寻址和符号寻址之间的区别，释放菜单命令 View＞Display＞Symbolic Representation。

```
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
```

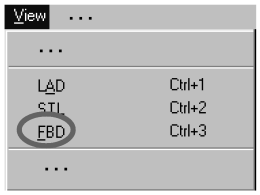
示例：
STL 中的绝对寻址

在 Help＞Contents 下的主题“编程块”，“创建逻辑块”以及“编辑 STL 指令”中能够找到更多的信息。

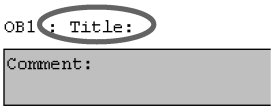
4.4 用功能块图编程 OB1

在下一部分中，你将用功能块图(FBD)编程一个 AND 功能，一个 OR 功能和一个记忆功能。

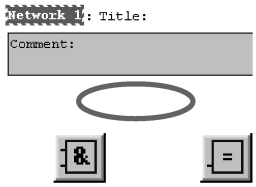
用功能块图编程一个 AND 功能。



如需要可在 **View** 菜单中将 **FBD** 设置为编程语言。

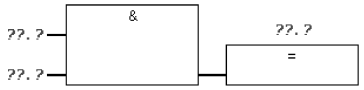


点击 **OB1** 的 **title** 区域并输入“Cyclically processed main program”作为示例。

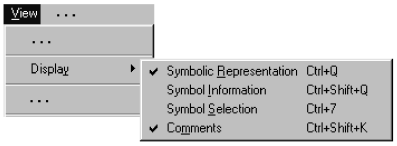


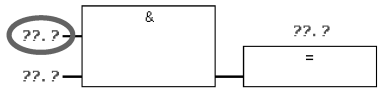
为 AND 功能选择输入区域(在注释(Comment)区域下)。
插入一个 AND 逻辑框(&)和一个赋值(=)。

AND 功能中各元素的地址还未输入。

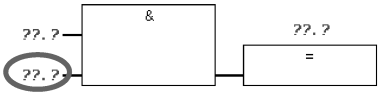


检查符号表达方式是否激活。

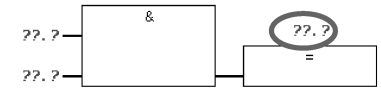




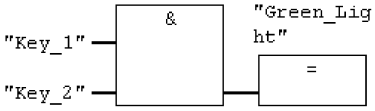
点击**??.?**号并输入符号名“Key_1”(在引号内)。用 **Enter** 确认。



为第二个输入输入符号名“Key_2”。



为赋值输入名称“Green_Light”。



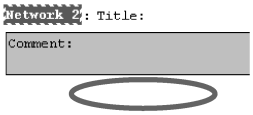
现在你已编程了一个完整的 AND 功能。

如果没有符号显示为红色，则可以保存该块。

如果符号在符号表中不存在或有语法错误则显示为红色。
你还可以直接从符号表中插入符号名。点击??.?号，然后选菜单命令 Insert＞Symbol。滚动下拉列表直至找到相应的名字并选中它。该符号名被自动添加。

用功能块图编程一个 OR 功能

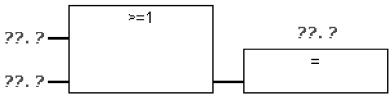
插入一个新段。



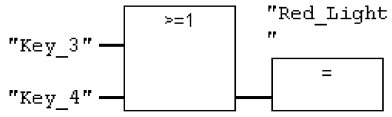
为 OR 功能选择输入区域。

插入一个 OR 逻辑框(≥1)和一个赋值(=)。

OR 功能中还未输入地址。可用与 AND 功能相同的方法进行。



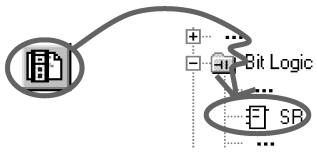
为上面的输入端输入“Key_3”，为下面的输入端输入“Key_4”，为赋值输入“Red_right”。



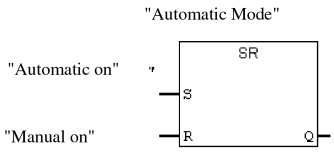
保存该块。

用功能块图编程一个记忆功能

选中第 2 段并插入另一段。选择输入区域(在注释区下面)。



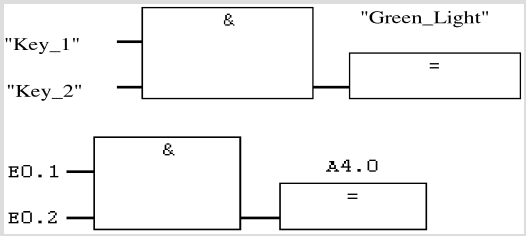
在编程元素目录中的 **Bit Logic** 下浏览，直至找到 **SR**。双击插入该元素。



为 SR 输入下列符号名：
置位“Automatic_On”
复位“Manual_On”
记忆位“Automatic_Mode”。

保存该块并关闭该窗口。

如果你想看一看绝对寻址和符号寻址的区别，释放菜单命令 **View＞Display＞Symbolic Representation**。



示例：
FBD 中的符号寻址

示例：
FBD 中的绝对寻址

你可以在 LAD/STL/FBD 编程窗口中改变符号寻址的行断，用菜单命令 **Options＞Customize**，然后选择“LAD/FBD”标签中的“Address Field Width(地址区域的宽度)。”在这里可以将行断设置为 10 至 24 个字符。

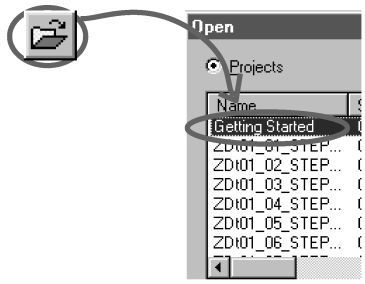
在 **Help＞Contents** 下的主题“编程块”，“创建逻辑块”和“编辑 FBD 指令”中可以找到各多的信息。

5 创建一个有功能块和数据块的程序

5.1 创建并打开功能块(FB)

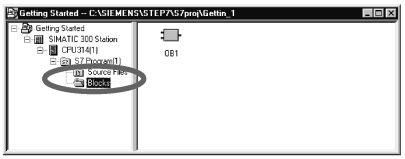
功能块(FB)在程序的分级结构中位于组织块之下。它包含程序的一部分，这部分程序可以在 OB1 中被多次调用。功能块的所有形参和静态数据都存储在一个单独的、被指定给该功能块的数据块(DB)中。

你将在 LAD/STL/FBD 窗口编程一个功能块(FB1，符号名“Engine”；见 3—3 页的符号表)。要这么作，你应该使用和第四章(编程 OB1)相同的编程语言。



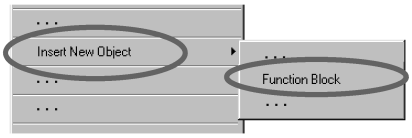
你应该已将符号表拷贝到你的项目“Getting Started”。如果尚未拷贝，可阅读 4—2 页了解如何操作拷贝符号表，然后返回这部分。

如果需要打开“Getting Started”项目。

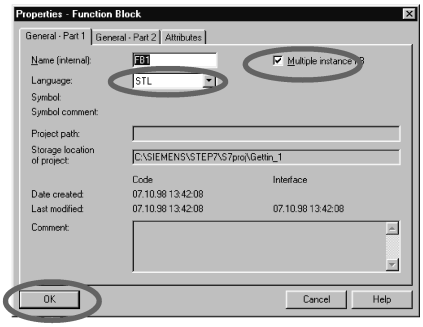


找到 **Blocks** 文件夹并打开它。

用鼠标右键点击右部窗口。

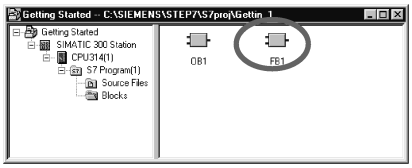


鼠标右键的弹出菜单中包含菜单栏中最重要的命令。插入一个 **Function Block**(功能块)作为新对象。



双击 FB1 打开 LAD/STL/FBD 编程窗口。

在“Properties(特性)—Function Block(功能块)”对话框中，选择你要用以生成块的语言，激活复选框“**Multiple instance FB (多重背景 FB)**”，用 OK 确认其余的设置。



功能块 **FB1** 被插入到 Blocks 文件夹。

根据你所选的编程语言，用梯形逻辑可继续阅读 5.2 部分，用语句表可读 5.3 部分，用功能块图可读 5.4 部分。

在 **Help>Contents** 下的主题“编程块”和“创建块和库”中可以找到更多的信息。

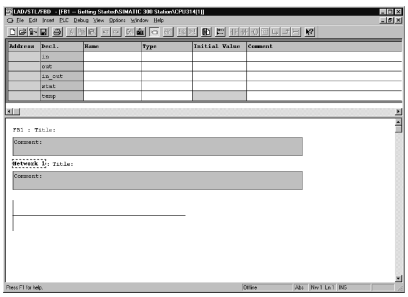
5.2 用梯形逻辑编程 FB1

我们将向您说明如何编程一个功能块，例如，该功能块使用两个不同的数据块可控制和监视一个汽油或柴油发动机。

所有作为块参数从组织块传送给功能块的 “发动机指定的信号”，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”)。

您应该已经了解用 STEP 7 如何输入一个串联电路，一个并联电路和一个记忆功能。

1.填写变量声明表



LAD/STL/FBD 编程窗口已经打开并且选项 **View>LAD**(编程语言)被激活

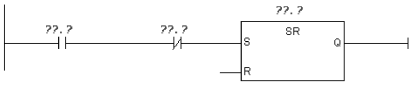
注意，现在 FB1 在标题栏中，因为你是通过双击 FB1 打开的编辑窗口。

在变量声明表中输入以下声明。要这么作，可点击一个单元并使用以下插图中相应的名字和注释。
使用鼠标右键，在弹击菜单命令 **Elementary Types** 中选择类型。当你按 **Enter** 时，光标跳到下一栏或插入新的一行。

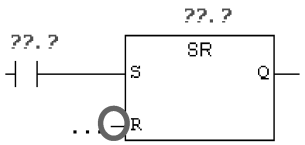
Address	Decl.	Name	Type	Initial Value	Comment
0.0	in	Switch_On	BOOL	FALSE	Switch on engine
0.1	in	Switch_Off	BOOL	FALSE	Switch off engine
0.2	in	Failure	BOOL	FALSE	Engine failure, causes the engine to switch off
2.0	in	Actual_Speed	INT	0	Actual engine speed
4.0	out	Engine_On	BOOL	FALSE	Engine is switched on
4.1	out	Preset_Speed_Reached	BOOL	FALSE	Preset speed reached
	in_out				
6.0	stat	Preset_Speed	INT	1500	Requested engine speed
	temp				

只有字母、数字和下划线是变量声明表中块参数的名字所允许使用的字符。

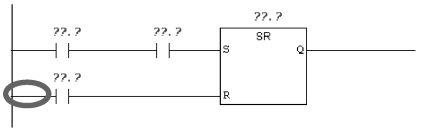
2.编程一个发动机切换其接通和断开



使用工具栏中相应的按扭或编程元素目录在段 1 中顺序插入一个常开触点，一个常闭触点和一个 SR 元素。

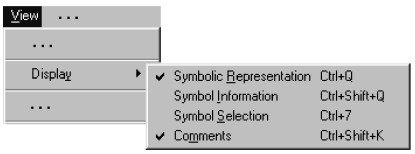


然后在输入 R 之前选择当前分支。

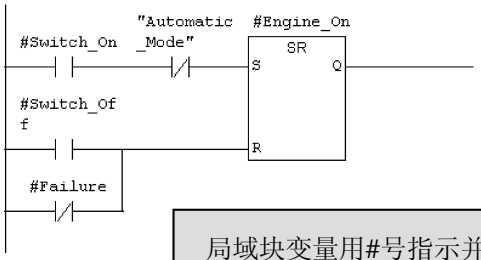


插入另一个常开触点。在该触点前选择当前分支。

插入一个与常开触点并联的常闭触点。
检查符号表达方式是否激活。

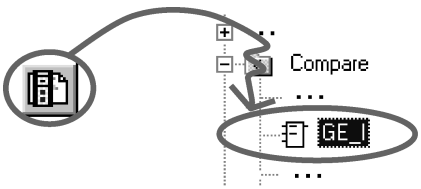


选中问号并输入来自于变量声明表的相应的名称(#号会自动被添加)。
为串联电路中的常闭触点输入符号名“Automatic_Mode”。然后保存你的程序。



局域块变量用#号指示并且只在该块中有效。
全局变量则出现在引号中。这些符号定义在符号表中并且在整个程序中有效。信号状态“Automatic_Mode”在 OB1 中(见 4-7 页段 3)由另一个 SR 元件定义，现在在 FB1 中被查询。

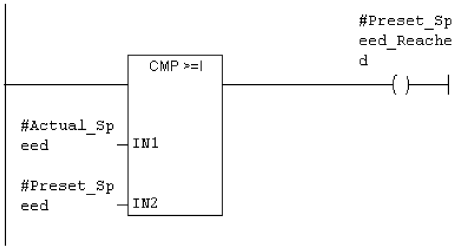
3.编程速度监控



插入一个新段并选择当前分支。然后在编程元素目录中浏览直至找到 **Compare(比较)** 功能并插入一个 **GE_I**。

在当前分支中还插入一个线圈。

选中问号，用变量声明表中的名称标定线圈和比较器。
然后保存你的程序。



发动机何时被切换为接通和关断?

当变量#Switch_On 有信号状态“1”并且变量“Automatic_Mode”有信号状态“0”时，发动机被切换为接通。该功能只有当“Automatic_Mode”被取反时(常闭触点)才能够被使能。

当变量#Switch_Off 有信号状态“1”，变量**#Failure** 有信号状态“0”时，发动机被关断。该功能可通过取反#Failure 再次实现(#Failure 是一个“0 有效”信号，它在常态下为“1”信号，如果出现故障则为“0”。

比较器是如何监控发动机速度的?

比较器比较变量#Actual_Speed 和#Preset_Speed 并把变量的结果赋值给#Preset_Speed_Reached(信号状态“1”)。

在 **Help>Contents** 下主题“编程块”，“创建逻辑块”和“编辑变量声明表”或“编辑 LAD 指令”中可以找到更多的信息。

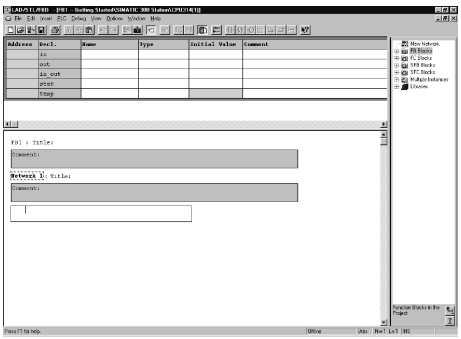
5.3 用语句表编辑 FB1

现在我们将向您说明如何编程一个功能块。例如该功能块使用两个不同的数据块可控制和监视一个汽油或柴油发动机。

所有为块参数从组织块传送给功能块的“发动机指定的”信号都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”)。

您应该已经了解了如何用 STEP 7 输入一个 AND 功能、OR 功能和置位/复位记忆指令。

1.填写变量声明表



LAD/STL/FBD 编程窗口已经打开并且选项 **View>STL**(编程语言)被激活。

注意，现在 FB1 在标题栏中，因为你是通过双击 FB1 打开的编辑窗口。

在变量表中输入以下声明。

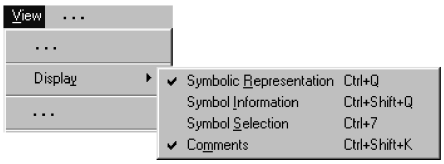
要这么作，可点击一个单元并使用以下插图中相应的名字和注释。

使用鼠标右键，在弹击的菜单命令 **Elementary Types** 中选择类型。按 **Enter** 后，光标跳到下一栏或插入新的一行。

Address	Decl.	Name	Type	Initial Value	Comment
0.0	in	Switch_On	BOOL	FALSE	Switch on engine
0.1	in	Switch_Off	BOOL	FALSE	Switch off engine
0.2	in	Failure	BOOL	FALSE	Engine failure, causes the engine to switch off
2.0	in	Actual_Speed	INT	0	Actual engine speed
4.0	out	Engine_On	BOOL	FALSE	Engine is switched on
4.1	out	Preset_Speed_Reached	BOOL	FALSE	Preset speed reached
	in_out				
6.0	stat	Preset_Speed	INT	1500	Requested engine speed
	temp				

只有字母、数字和下划线是变量声明表中块参数名所允许使用的字符。

2.编程一个发动机切换其接通和断开



检查符号表达方式是否激活

在段 1 中输入相应的指令

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

局域块变量用#号指示并只或在该块中有效。全局变量出现在引号内，它们在符号表中定义并且在整个程序中有效，信号状态“Automatic_Mode”在 OB1 中(见 4—10 页，段 3)由另一个 SR 元素定义，现在在 FB1 中被查询。

3.编程速度监视

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

插入一个新段并输入相应的指令。然后保存程序。

发动机何时被接通和断开?

当变量#Swith_On 有信号状态“1”并且变量“Automatic_Mode”有信号状态“0”时，发动机被接通。只有当“Automatic_Mode”取反(常闭触点)时该功能才能够被使能。

当变量#Switch_Off 有信号状态“1”或变量#Failure 有信号状态“0”时，发动机被断开。该功能可通过取反#Failure 再次现实(#Fault 是一个“0 有效”信号，它在常态下有信号“1”，若出现故障则为“0”)。

比较器如何监测发动机的速度?

比较器比较变量 #Actual_Speed 和 Preset_Speed 并将变量的结果赋值给 #Presett_Speed_Reached(信号状态“1”)

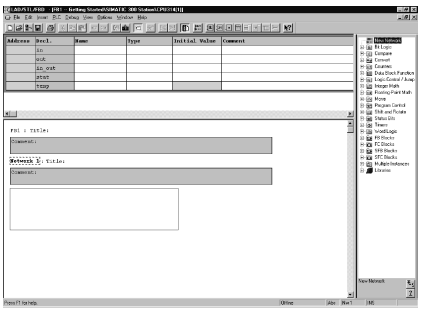
在 **Help>Contents** 下的主题“编程块”，“创建逻辑块”和“编辑变量声明表”或“编辑 STL 指令”中可以找到更多的信息。

5.4 用功能块图编程 FB1

现在我们将向您说明如何编程一个功能块，例如，该功能块使用两个不同的数据块可控制和监视一个汽油或柴油发动机。
所有作为块参数从组织块传送给功能块的“发动机指定的”信号都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”)。

您应该已经了解了如何用 STEP 7 输入一个 AND 功能，一个 OR 功能和置位/复位记忆指令。

1.填写变量声明表



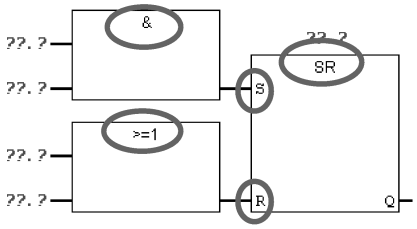
LAD/STL/FBD 编程窗口已经打开并且选项 **View>FBD**(编程语言)被激活。
注意，现在 FBI 在标题栏中，因为你是通过双击 FB1 打开的编辑窗口。

在变量表中输入以下声明。
要这么作，点击一个单元并使用以下插图中相应的名字和注释。
使用鼠标右键，在弹击的菜单命令 **Elementtary Types** 中选择类型。按 **Enter** 后，光标跳到下一栏或插入新的一行。

Address	Decl.	Name	Type	Initial Value	Comment
0.0	in	Switch_On	BOOL	FALSE	Switch on engine
0.1	in	Switch_Off	BOOL	FALSE	Switch off engine
0.2	in	Failure	BOOL	FALSE	Engine failure, causes the engine to switch off
2.0	in	Actual_Speed	INT	0	Actual engine speed
4.0	out	Engine_On	BOOL	FALSE	Engine is switched on
4.1	out	Preset_Speed_Reached	BOOL	FALSE	Preset speed reached
	in_out				
6.0	stat	Preset_Speed	INT	1500	Requested engine speed
	temp				

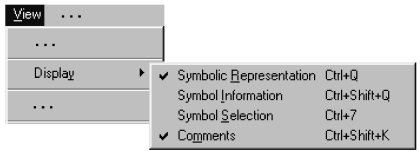
只有字母、数字和下划线是变量声明表中块参数名所允许使用的字符。

2.编辑一个发动机切换其接通和断开



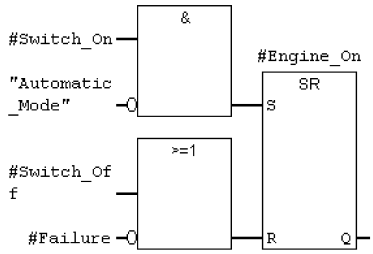
在段 1 用编程元素目录(Bit Logic 文件夹)插入一个 SR 功能。

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



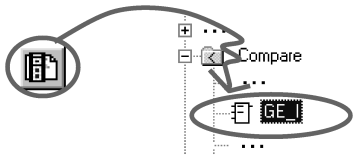
检查符号表达方式是否激活。

点击???号并输入来自声明表的相应的名称(#号被自动添加)。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#Failure 取反。然后保存程序。



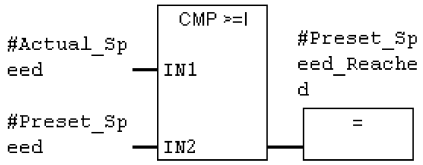
局域块变量用#号指示并且在该块中有效。全局变量出现在引号中。这些符号在符号表中定义并且在整个程序中有效。
信号状态“Automatic_Mode”在 OB1(见 4-14 页，段 3)中由另外一个 SR 元素定义，现在在 FB1 中被查询。

3.编程速度监控



插入一个新段并选择输入区域。
然后在编程元素目录中浏览直至找到 Compare（比较）功能，插入一个 GE_I。

在比较器后面附上一个输出赋值，用来自变量声明表的名称作为输入的地址。
然后，保存程序。



何时发动机被接通和断开?

当变量#Switch_On 有信号状态“1”并且变量“Automatic_Mode”有信号状态“0”时发动机接通。只有当“Automatic_Mode”取反（常闭触点）时，该功能才能够被使能。
当变量#Switch_Off 有信号状态“1”或变量#Failure 有信号状态“0”时发动机被断开。该功能可通过取反#Failure 再次实现（#Failure 是一个“0 有效”信号，它在常态有信号“1”，若出现故障则为“0”）。

比较器如何监控发动机速度?

比较器比较变量#Actual_Speed 和#Preset_Speed 并将变量的结果赋值给#Preset_Speed_Realhed（信号状态“1”）。

在 Help>Contonts 下的主题“编程块”，“创建逻辑块”和“编辑变量声明表”或“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。

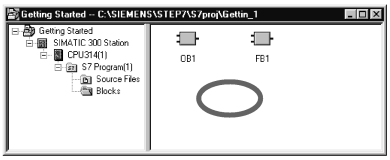
你已经编写了功能块 FB1（“发动机”）并且还在变量声明表中定义了发动机指定的参数。

为使您以后能在 OB1 中编写调用功能块的指令，必须生成相应的数据块。

一个背景数据块（OB）总是被指定给一个功能块。

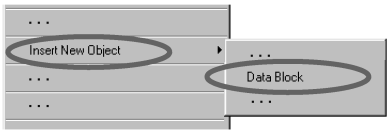
这个功能块用于控制和监视一个汽油或柴油发动机。不同的发动机的预设速度分别存储在两个数据块中，在这些数据块中实际值被修改。

通过只集中编写功能块一次，可以减少相关的编程量。

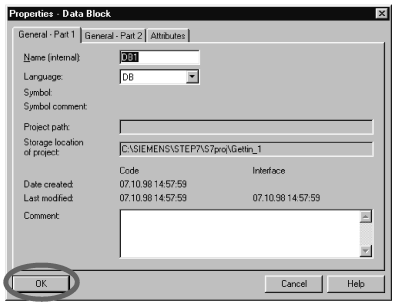


在 SIMATIC 管理器中打开项目“ Getting Started” 。

找到 **Blocks** 文件夹并用鼠标右键点击右半窗口。



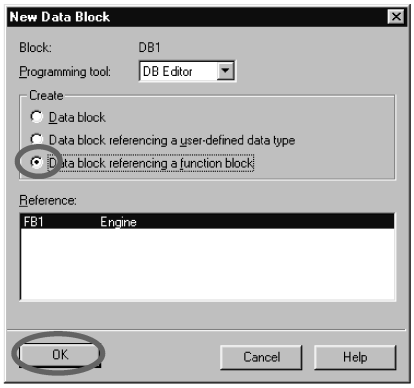
用鼠标右键的弹出菜单插入一个 **data block**（**数据块**）。



用 **OK** 确认“**Properties**”对话框中显示的所有设置。

数据块 **DB1** 被添加到“Getting Started”项目中。

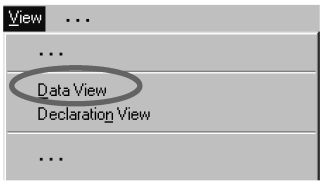
双击打开 **DB1**。



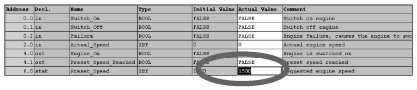
在“New Data **Block**(新数据块)”对话框中激活选项 Data block referencing a function block.

用 **OK** 确认赋值关系“FB1，Engine”.

LAD/STL/FBD 编程窗口打开并显示来自 FB1 变量声明表的数据。

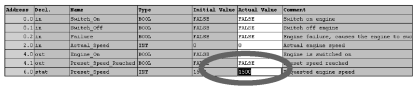


现在 OB1 包含指定给一个汽油发动机的数据。你还要输入这些数据。首先设置为 Data View(数据视窗)。



接着在实际值一栏中为汽油发动机输入数值“1500”（在 Preset_Speed 这一行中）。现在你已为该发动机定义了最大速度。

保存 DB1 并关闭编程窗口。



用与 DB1 相同的方式为 FB1 生成另一个数据块 DB2。

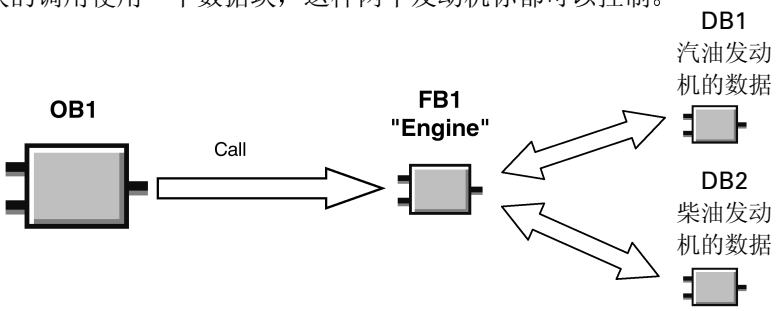
现在为柴油发动机输入实际值“1200”。

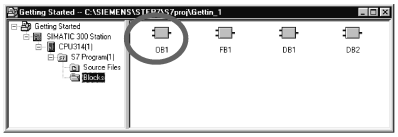
通过修改实际值你已完成用一个功能块控制两个发动机的准备工作，要控制更多的发动机，作所要做的就是生成其它的数据块。
你要做的下一件事情是在 OB1 中编程对功能块的调用。要这么作，根据你使用的编程语言，梯形逻辑可继续阅读 5.6 部分，语句表 5.7 部分，功能块图 5.8 部分。

在 Help>Content 下的主题“编程块”和“创建数据块”中你能够找到更多的信息。

5.6 用梯形逻辑编程块调用

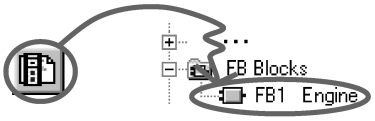
为编程功能块所作的所有事情只有当你在 OB1 中调用该功能块时才有用处。一个功能块的调用使用一个数据块，这样两个发动机你都可以控制。



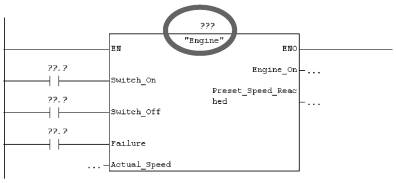


SIMATIC 管理器随着你的项目“Getting Started”被打开。

找到 **Blocks** 文件夹并打开 **OB1**。



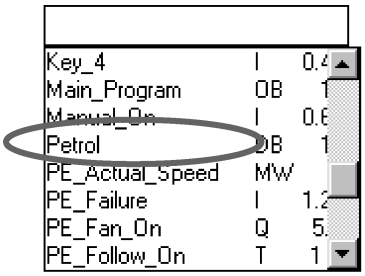
LAD/STL/FBD 编程窗口插入 Network 4。然后在编程元素中查找直到找到 **FB1** 并插入该块。



在以下各项前面插入一个常开触点：Switch_On,Switch_Off 和 Failure。

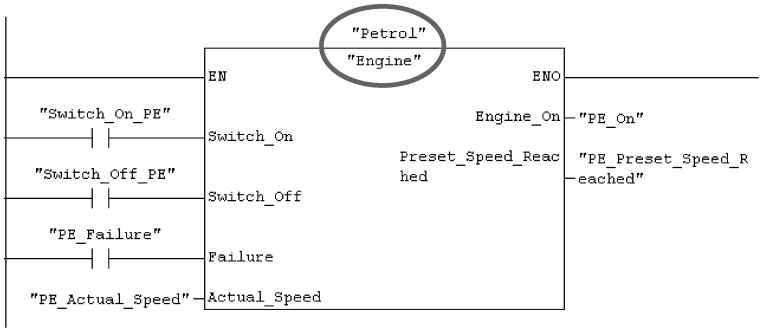
点击“Engine”上的**???**号，然后将光标保持在同一位置用鼠标右键点击输入结构。

选择鼠标右键的弹出菜单中的 **Insert Symbol**。会出现一个下拉列表。如果你是第一次这么作，这个过程会需要一些时间。



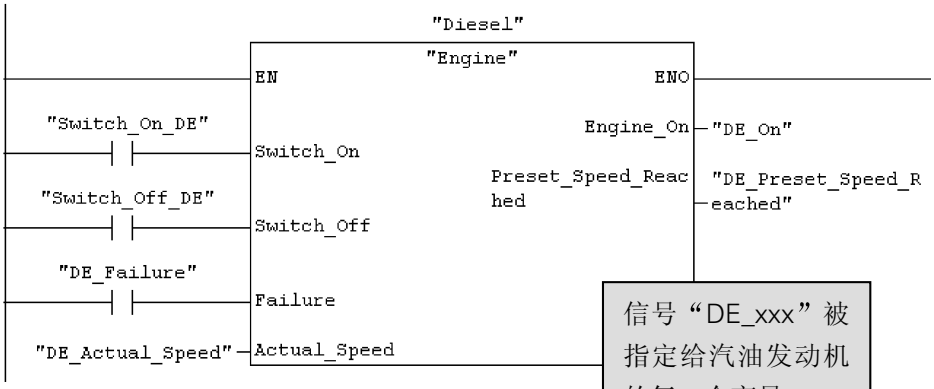
点击数据块 **Petrol**。该块则自动被输入到输入结构中，并被加上引号。

点击问号，使用下拉列表中相应的符号名为功能块的所有其它参数输入地址。



发动机指定的输入和输出变量（声明“in”和“out”）显示在 FB“Engine”中。
名为“PE-xxx”的信号被赋值给汽油发动机的每一个信号。

在一个新段中编写功能块“Engine”(FB1)的调用指令并使用数据块Diesel”(DB2)。使用下拉列表中相应的地址。



信号“DE_xxx”被指定给汽油发动机的每一个变量

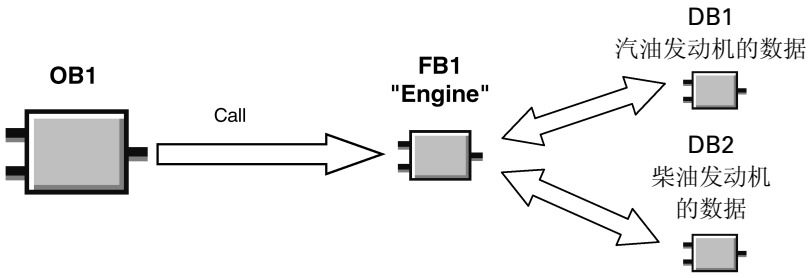
保存程序并关闭块。

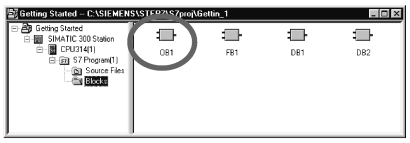
当你创建一个有组织块，功能块和数据块的程序结构时，必须在子程序块（如 FB1）的前一级块中（如，OB1）编写对它们的调用指令。这个过程都是一样的。
你还可以在符号表中输出各个块的符号名（例如，FB1 的名字为“Engine”以及 DB1 名为“Petrol”）。
你可以在任可时候存档或打印出编程的块。相应的功能在 SIMATIC 管理器下的菜单命令 File>Archive 或 File>Print。

在 Help>Contents 下的主题“调用参考帮助”“语言描述:LAD”以及“程序控制指令”中可以找更多的信息。

5.7 用语句表编程块调用

只有当你在 OB1 中调用一个功能块时,为编写该块所作的所有工作才是有意义的.功能块的每一次调用要使用一个数据块,用这种方式,你可以控制两个发动机.





SIMATIC 管理器及你的项目" Getting Started"被打开.

找到 **Blocks** 文件夹并找开 **OB1**

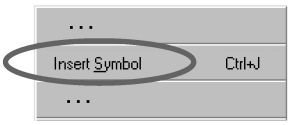
在 LAD/STL/FBD 编辑窗口插入段 4。

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

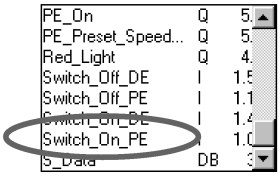
在指令区键入 **CALL** "**Engine**", " **Petrol**" 并按 **Enter**。

该功能块"Petrol"的所有参数都显示出来。

将光标放在"Switch_on"后面的等号上并按鼠标右键。



选择鼠标右键的弹出菜单中的 **Insert Symbol**。出现一个下拉列表。若你第一次作，这一过程需要一些时间。



点击 **Switch_on_PE**，该名字从下拉列表中取出自动添加在引号内。

```
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所有其它参数分配地址。

信号“PE_xxx”被赋值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"
```

在一个新段中编程对功能块“Engine”(FB1)的调用并使用数据块“Diesel”(DB2),按与其他调用相同的方式进行。

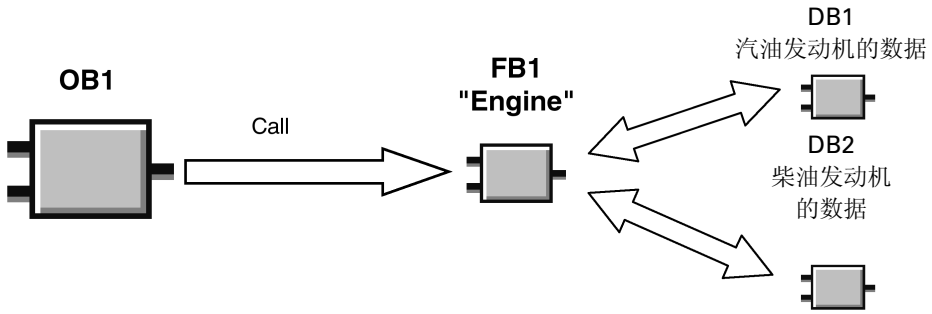
保存程序并关闭块

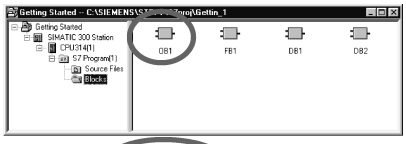
当你创建一个有组织块，功能块和数据块的程序结构时，必须在分级结构中高一级的块中（如 OB1）编写子程序块（如 FB1）的调用。其过程都是一样的。
你还可以在符号表中给出各个块的符合名（例如，FB1 的符号为“Engine”，DB1 的名字为“Petrol”）。
你还可以在任何时候存档或打印编程块。相应的功能可在 SIMATIC Manager 中菜单命令 **File>Archive** 或 **File>Print** 下找到。

在 **Help>Contents** 下的主题“调用参考帮助”，“语言描述：FBD”，“程序控制指令”中能够找到更多的信息。

5.8 用功能块图编程块调用

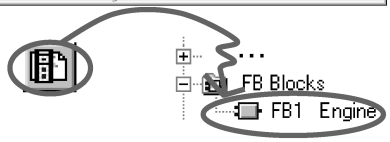
只有在 OB1 中编号调用功能的指令，你为编写该功能块作的所有工作才是有意义的。每一次功能块的调用使用一个数据块，用这种方式，你可以控制两个电机。



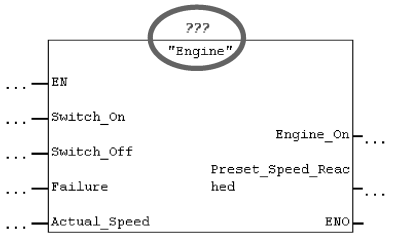


SIMATIC 管理器连同你的项目“Getting Started”被打开。

找到 **Blocks** 文件夹并打开 **OB1**。

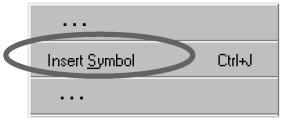


在 LAD/STL/FBD 编程窗口插入段 4。在编程元素目录中查找直到找到 **FB1**，插入该块。

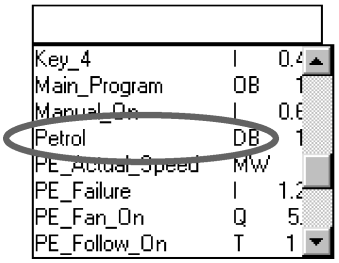


显示所有的发动机指定的输入和输出变量。

点击“Engine”上面的**???**号，然后保持光标在同一位置，用鼠标右键点击输入结构。

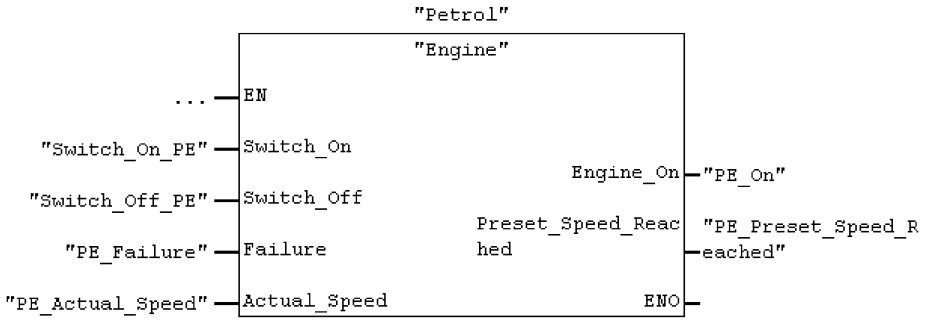


选择鼠标右键的弹出菜单中的 **Insert Symbol**。出现一个下拉列表，若你第一次作，这一过程需要一些时间。



点击数据块 **Petrol**，将它从下拉列表中取出并且自动地被输入到输入结构中，加在引号内。

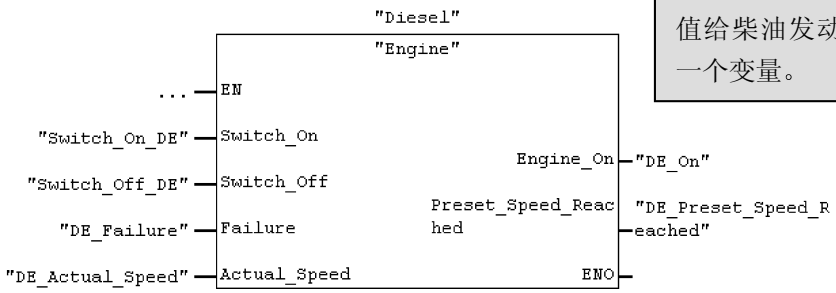
用下拉列表中相应的符号名为功能块的所有其他参数分配地址。



信号“PE_xxx 被赋值给汽油发动机的每个变量。

在一个新段中编程对功能块“Engine”(FB1)的调用并使用数据块“Diesel”(DB2)，使用下拉列表中相应的地址。

信号“DE_xxx”被赋值给柴油发动机的每一个变量。



保存你的程序并关闭该块

当你创建一个有组织块、功能块和数据的程序结构时，必须在分级结构中高一级别的块中（如 OBI）编写子程序块（如 FB1）的调用，其过程都是一样的。
你还可以在符号表中给出各个块的符号名（例如，FB1 的符名为“Engine”，DB1 的名字为“Petrol)”。
你可以在任何时候存档或打印编程的块。相应的功能可在 SIMATIC Manager 中菜单命令 File>Archive 或 File>print 下找到。

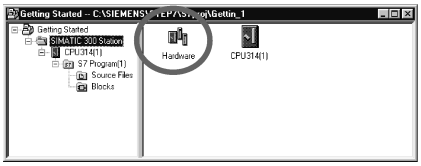
在 Help>Contents 下的主题“调用参考帮助，“语言描述：FBD”，和“程序控制指令”中能够找到更多的帮助。

6 组态中央机架

6.1 组态硬件

一旦你创建了一个有 SIMATIC 站的项目，就可以组态硬件了。在 2.1 部分用 STEP 7 Wizard 创建的项目结构完全能满足这一要求。

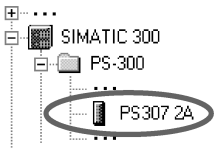
硬件用 STEP 7 组态。这些组态数据以后可以通过下载(downloading)"(见第七章)传送到可编程控制器。



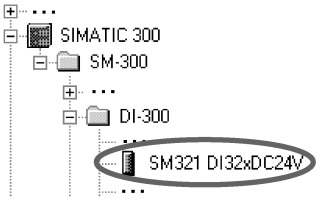
开始处在打开的 SIMATIC Manager 及"Getting Started"项目。

打开 **SIMATIC 300 Station** 文件夹并双击 **Hardware**(硬件)符号。

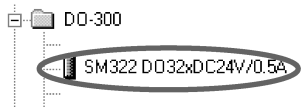
"Hw Config"窗口打开。在创建项目时所选择的 CPU 显示出来。对于"Getting Started"项目，是 CPU 314。



首先你需要一个电源模板。在目录中查找直找到 **PS307 2A**，将该模板拖至 1 号槽。



查找输入模板（DI，数字输入）**SM321 DI32XDC24V**，将它插入到 4 号槽。3 号槽空着。



用同样的方式插入输出模板 **SM322 DO32 DC24V/0.5A** 在 5 号槽。

要在一个项目内修改模板参数（例如，地址），双击该模板。但是，你只能修改这样一些参数，即，你确实了解这些参数的修改会对你的可编程控制器产生什么样的影响。

对于“Getting Started”项目，不需要作任何参数修改。

Slot	Module	Order Number	MPI Address	I Add...	Q...	Comment
1	PS307 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						

用菜单命令 **Save and Compile**（存储并编译)为向 CPU 传送准备好数据。

一旦你关闭“Hw Config ”应用程序，在 Blocks 文件夹中将出现系统数据的符号。

使用菜单命令 **Station>Consistency Check** 还可以检查你的组态错误。对任何可能出现的错误，STEP 7 都为你提供了可能的解决方案。

在 **Help>Contents** 下的主题“组态硬件”和“组态中央机架”中可以找到更多的信息。

7 下载和调式程序

7.1 建立一个在线连接

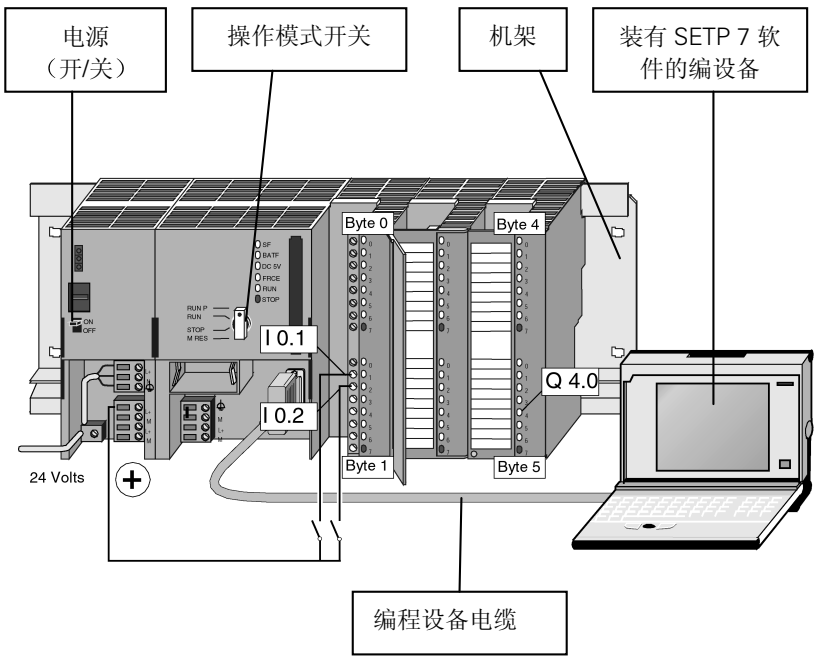
使用提供的项目“GS-LAD Example”或你已创建的项目“Getting Started”以及一个简单的测试组态，我们将向你显示如何下载程序到可编程逻辑控制器（PLC），然后如何调试它。

你应该已经

- 为“Getting Started”项目组态了硬件（见第六章）
- 按照安装手册设置了硬件

一个串联电路（AND 功能）的示例：

只有当键 I 0.1 和键 I 0.2 都按下时，输出 Q 4.0 点亮(数字输出模板上的指示灯 Q 4.0 点亮)。用你的 CPU 的接线建立测试组态。



组态硬件

要将模板组装到导轨，可按以下给出的顺序进行：

- 将模板与总线连接器相连
- 将模板挂在导轨上并向下摆动
- 因定模板位置
- 组装其余模板
- 一旦完成所有模板的组装后将钥匙开关插在 CPU 上

如果你使用的硬件与图示不一样，也仍然可以进行测试。只是要使地址与输入和输出相符。

STEP 7 为您提供各种各样的调试程序的方法；例如，使用程序状态或用变量表的方式。

在手册“S7-300，硬件和安装/模板技术规范”和“S7-400/M7-400-硬件”中可以找到更多的关于组态中央机架的信息。

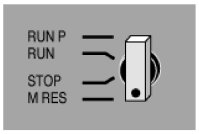
7.2 下载程序到可编程控制器

你应该已建立了一个在线连接，以便下载程序。

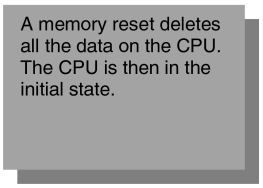
用 ON/OFF 开关接通电源。CPU 上的“DC5V”指示灯点亮。

将操作模式开关转为 STOP 位置（如果尚未处于 STOP）。红色的“STOP”LED 将点亮。

复位 CPU 并切换到 RUN

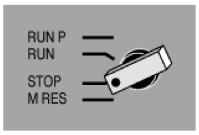


将操作模式开关转到 **MRES** 位置并保持至少 3 秒钟直至红色的“STOP”LED 开始慢闪。

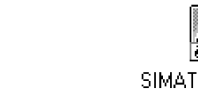


放开开关并且至多在 3 秒之内将开关再转到 **MRES** 位置。当“STOP”LED 快闪时，CPU 已被复位。
如果“STOP”LED 没有开始快闪，重复这一过程。

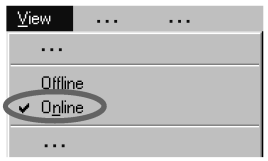
下载程序到 CPU



现在将操作模式开关重新转为“STOP”下载程序。

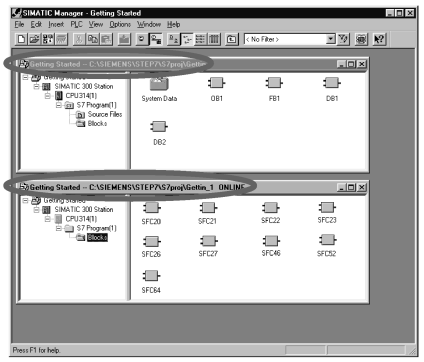


启动 SIMATIC Manager,在“Open”对话框中打开“Getting Started”项目（如果它尚未被打开）。



除了“Getting Started Offline(离线)”窗口外，打开“Getting Started ONLINE(在线)”窗口。在线或离线状态通过不同颜色的标头指示。

在两个窗口中定位到 **Blocks** 文件夹。

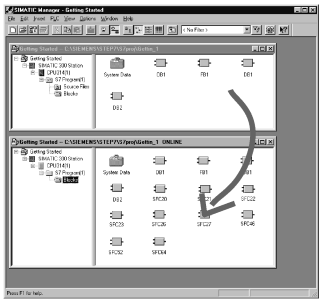


离线窗口显示编程设备上的情形；在线窗口显示 CPU 上的情形。

> 既使执行存储器复位，系统功能（SFC）也会保留在 CPU 中。CPU 提供这些操作系统的功能。它们无须被下载，也不能被删除。

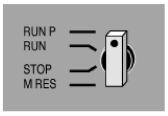
在离线窗口选择 Blocks 文件夹，然后用菜单命令 **PLC > Download** 下载程序到 CPU,用 **OK** 确认提示。

当你完成下载之后,这些程序块就显示在在线窗口。



> 你还可以用工具栏中相应的按钮或鼠标右键的弹出菜单来调用菜单命令 **PLC > Download**。

接通 CPU 并检查操作模式：



将操作模式开关转为 **RUN-P**。绿色的“RUN”LED 点亮而红色“STOP”LED 灭掉。这 CPU 已为操作作好准备。

当绿色 LED 变亮时，你可以开始测试程序。

如果红色 LED 仍亮着，说明有错误出现。你需要评估诊断缓存区以便诊断错误。

下载单个块

在实际当中，为对错误作出快速反应，可以使用拖放功能将块一个一个地传送到 CPU。当你下载块时，CPU 上的操作模式开关必须在“RUN-P”或“STOP”模式。在“RUN-P”模式下载的块立即被启动，因此，你必须记住以下各事项：

- 如果没有错误的块被错误块重写，这将导致系统故障。你可以在下载块之前对它们进行测试从而避免这种情况。
- 如果你没有按照一定的顺序下载块———首先是子程序块，然后是更高一级的块——CPU 将进入“Stop”模式。你可以下载整个程序到 CPU，从而避免这种情况。

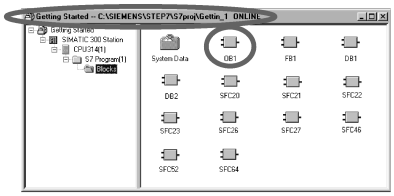
在线编程

在实际当中，为测试目的，你可能需要修改已经下载到 CPU 的块。要这么作，在在线窗口双击所需的块，打开 LAD/STL/FBD 编程窗口。然后象通常一样编程该块。注意，这个编完的块会立即在你的 CPU 中生效。

在 **Help>Contents** 下的主题“建立在线连接并进行 CPU 设置”和“从 PG/PC 下载到可编辑控制器”中可以找到更多的信息。

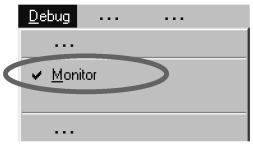
7.3 用程序状态测试程序

使用程序状态（Program status）功能，可以在一个块中测试程序。这一功能的需求是：你已建立了与 CPU 的在线连接，该 CPU 在 RUN 或 RUN-P 模式，并且程序已经下载。



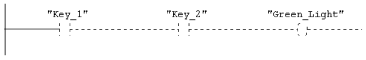
在项目窗口“Getting Started ONLINE”打开 **OB1**。

LAD/STL/FBD 编辑窗口打开。



激活功能 **Debug>Monitor**.

梯形逻辑进行调试

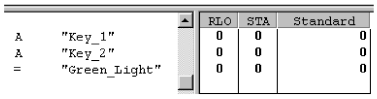


段 1 中的串联电路以梯形逻辑的形式显示。当前支路一直到 Key1(I0.1)表示为一条实线；这表明已为该电路提供电源。

用块能块图进行调试

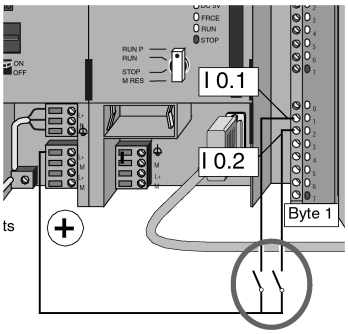
信号状态由“0”“和“1”指示。
点虚线表示没有逻辑操作结果。

用语句表进行调试



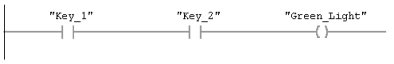
在语句表中，以表格的形式显示以下内容：
- 逻辑操作结果（RLO）
- 状态位（STA）
- 标准状态（STANDARD）

在测试过程中可以使用 **Options>Customize** 改变编程语言的表达方式

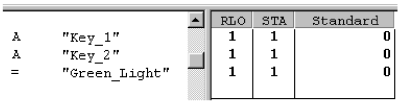


现在，在你的测试组态中将两个开关都按下。

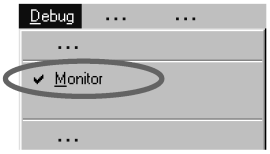
输入模板上输入 I0.1 和 I0.2 的指示灯亮。
输出模板上输出 Q 4.0 的指示灯亮。



在图形编程语言梯形逻辑和功能块图中，你可以通过编程的段中颜色的变化来追踪测试结果。这种颜色变化显示该点逻辑操作结果满足。



使用语句表编程语言，当逻辑操作结果满足时，STA 和 RLO 栏中的显示变化。



释放功能 **Debug>Monitor** 并关闭窗口。

然后关闭 SIMATIC Manage 中的在线窗口。

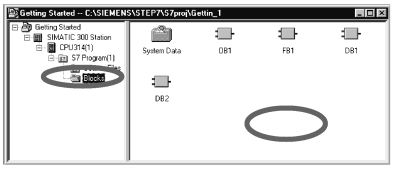
我们建议不要将大规模的程序全部下载到 CPU 中运行，因为，可能的错误源的数量会使故障诊断更困难。所以，为了获得一个更好的概貌，应该将块单个地下载并测试它们。

在 **Help>Contents** 下的主题“调试”和“用程序状态测试”中可以找到更多的信

7.4 用变量表测试程序

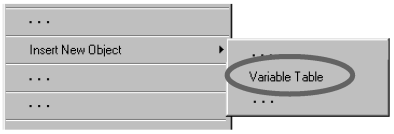
你可以通过监视和修改功能测试各个程序变量。这一功能的要求是你已建立了与 CPU 的在线连接，该 CPU 在 RUN-P 模式并且程序已下载。与用程序状态测试一样，你可以在变量表中监视段 1（串联电路或 AND 功能）中的输入和输出。你还可以通过预置一个实际速度测试 FB1 中用于发动机速度比较的比较器。

创建变量表

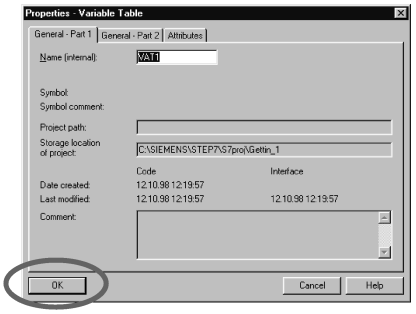


开始处还是在 SIMATIC 管理器中打开的项目窗口“Getting Started Offline”。

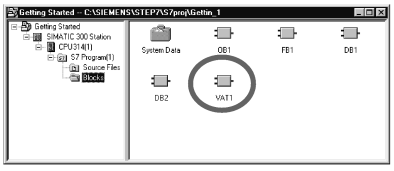
浏览找到 **Blocks** 文件夹并用鼠标右键点击右半窗口。



用鼠标右键的弹出菜单插入一个 **Variable Table**(变量表)。



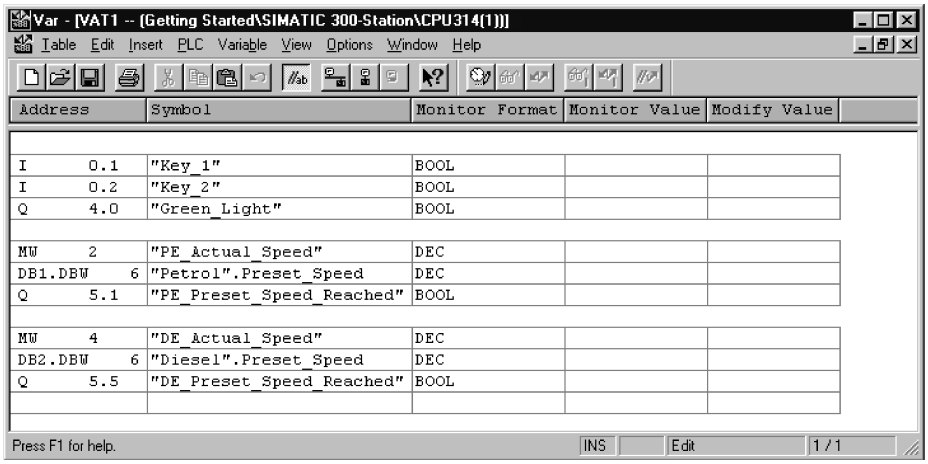
用 **OK** 关闭“Properties(特性)”对话框接受缺省设置。



一个 **VAT 1**（变量表）在 Blocks 文件夹中生成。

双击打开 **VAT 1**；“Monitoring and modifying Variables（监视和修改变量）”窗口将被打开。

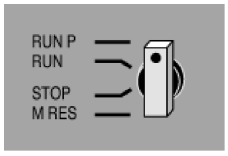
一开始，变量表是空的。按下面插图所示，为“Getting Started”示例输入符号名或地址。当你用 **Enter** 完成输入项时，其余的明细数据会加进来。
将所有的速度值的状态格式改为 DEC（十进制）格式。要这么作，可点击相应单元的首部（光标在状态格式栏上将变为一个箭头），用鼠标右键选择 DEC 格式。



存储您的变量表

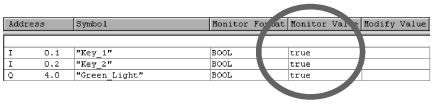
将变量表切换为在线

点击“Monitoring and Modifying Variable”窗口工具栏中的 **ON** 按钮建立与组态的 CPU 之间的连接。“ONLINE”的字样则会出现在状态栏中。



将 CPU 的钥匙开关设置为 **RUN-P**（如果你尚未设置为 RUN-P）。

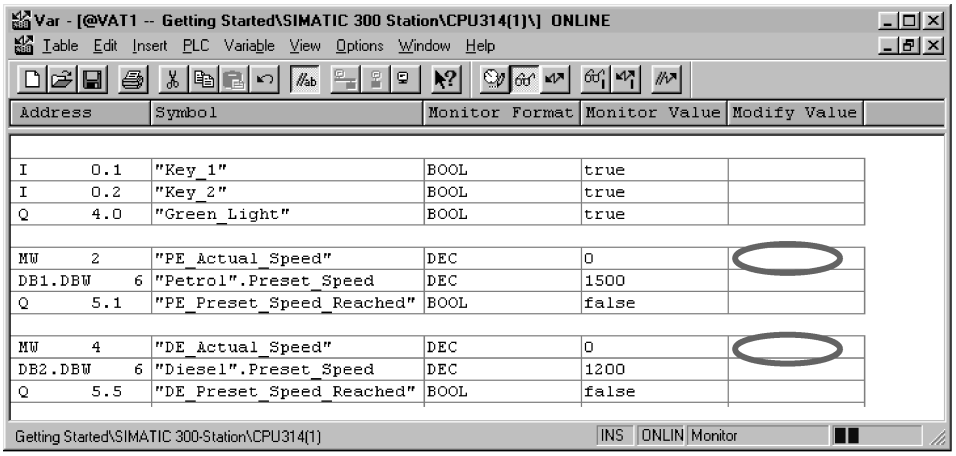
监视变量



点击工具栏中的 **Monitor Variables** (监视变量)。CPU 的操作模式显示在状态栏中。按下你所测试的组态中的 Key1 和 Key2。(键 1 和键 2)，监视变量表有中的结果。变量表中的状态值将由 false 变为 true。

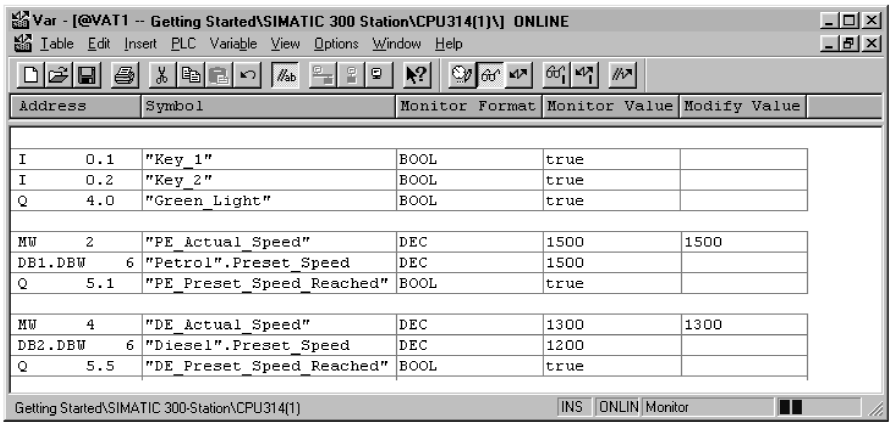
修改变量

在 Modify Value(修改值)这一栏中为地址 MW2 输入数值“1500”，为地址 MW4 输入“1300”。



传送修改值到你的 CPU

在传送之后，这些数值将在你的 CPU 中被处理，可以看到比较的结果。停止监视变量（再次点击工具栏中的按钮）并关闭窗口，对任何提问回答 YES 或 OK。



由于屏幕空间的限制，非常大的变量表经常不能被完全显示。
如果你有庞大的变量表，我们建议你用 STEP 7 为一个 S7 程序生成几个变量表，你可以调整变量表使之正好适合你的测试要求。
你可以为每个变量表分配一个名字，方法与给块起名字一样（例如，OB1Network1 替代 VAT1）。用符号表分配新名字。

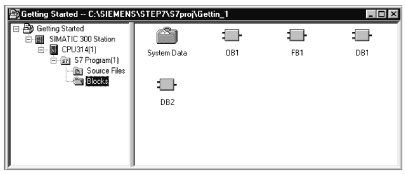
在 Help>Contents 下的主题“调试”和“用变量表测试”中能够找到更多的信息。

7.5 评估诊断缓存区

在一个极端的情况下，如果在处理一个 S7 程序时 CPU 进入 STOP，或者当你下载程序后无法将 CPU 切换为 RUN，你可以从诊断提缓存区的事件列表中判定错误的原因。

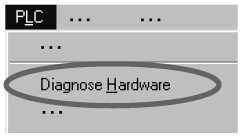
这一功能的要求是你已建立与 CPU 的连接并且 CPU 在 STOP 模式。

首先将 CPU 上的操作模式开关转为 STOP。

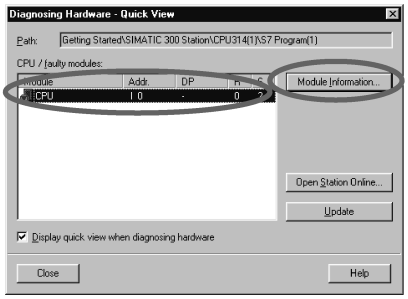


开始点在 STMATIC 管理器中打开的项目窗口“Getting Started Offline”。

选择 **Blocks** 文件夹。



如果你的项目中有几个 CPU，首先判定是哪个 CPU 进入了 STOP。



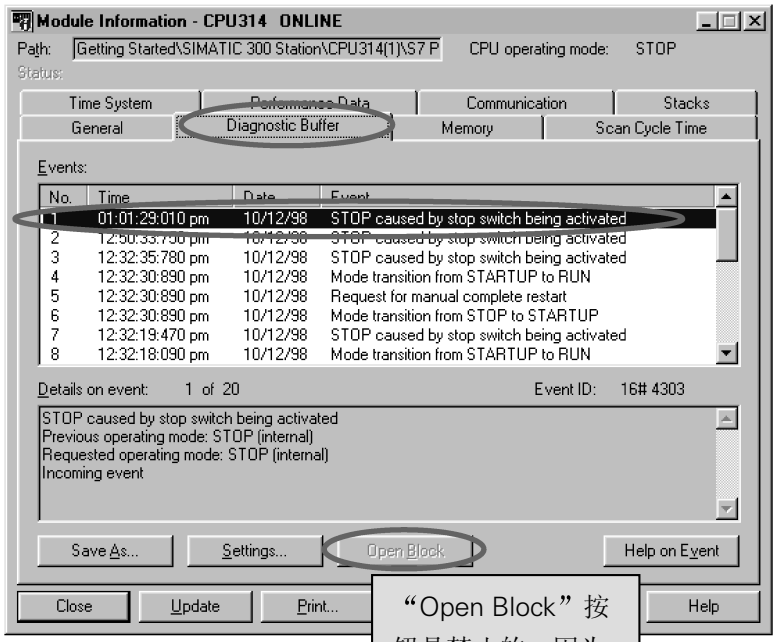
所有可访问的 CPU 都列在“Diagnosing Hardware(诊断硬件)”对话框中。处于 STOP 操作模式的 CPU 被加亮。

在“Getting Started”项目中只显示有一个 CPU。

点击 **Module Information**(模板信息)，对该 CPU 诊断缓存区进行评估。

如果只连接了一个 CPU，你可以用菜单命令 **PLC>Module Information** 直接为该 CPU 查询模板信

“Module Information”窗口为你提供关于你的 CPU 的特性及参数的信息。现在选择“Diagnostic Buffer(诊断缓存区)”标签，判定造成 STOP 的原因。



“Open Block”按钮是禁止的，因为在“Getting Started”项目的块里没有错误。

最近的事件（号码 1）在列表的最上面。显示造成 STOP 状态的原因。关闭除 SIMATIC Manager 之外的所有窗口。

如果是由编程错误造成 CPU 进入 STOP 模式，选择该事件并点击“Open Block”按钮。该块则在你熟悉的 LAD/STL/FBD 编程窗口打开，出错的段被加亮。通过本章你已成功地完成了“Getting Started”样板项目，从创建一个项目到调试完成的程序。在下一章中，通过作选择的练习可以进一步扩展你的知识。

在 **Help>Contents** 的主题“调用模板信息”中可以找到更多的信息。

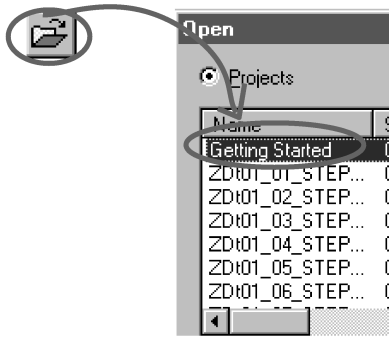
8 编程一个功能

8.1 创建并打开功能（FC）

功能和功能块一样，在程序结构中在组织块的下面。为使一个功能被 CPU 处理，它必须被它的前一级块调用。与功能块不同，功能不需要数据块。

在功能中，在变量声明表中列出参数，但是不允许使用静态局域数据。你可以使用 LAD/STL/FBD 编程窗口编程一个功能，其方法与编程一个功能块一样。

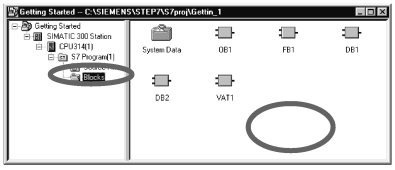
你应该已经熟悉了用梯形逻辑、功能块图或语句表（见第四和第五章）编程，还有符号编程（见第三章）。



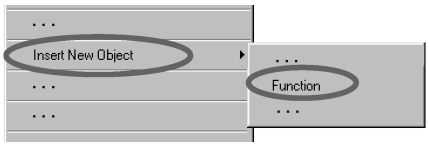
如果你已在第一至第七章中创建了样板项目“Getting Started”,现在打开它。

如果没有，用菜单命令 **File> “New Project”** wizard 在 SIMATIC 管理器中生成一个新项目。要这么作可遵循 2.1 部分的指导，并且将项目重新命名为“Getting Started Function”。

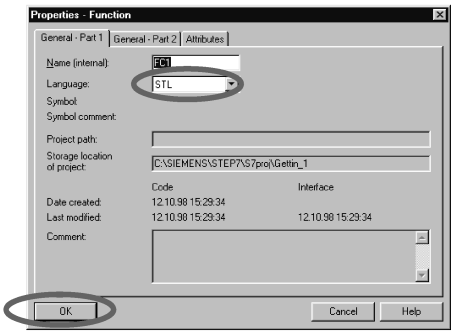
我们将继续使用“Getting Started”项目。而你可以用新项目来完成每一个步骤。



找到 **Blocks** 文件夹并打开它。
用鼠标右键点击右半窗口。

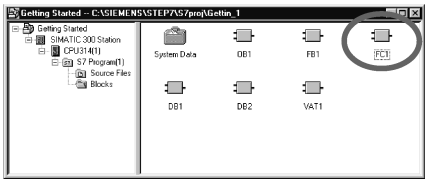


从弹击菜单中插入一个功能（**FC**）。



在“Properties-Function(特性-功能)”对话框中，接受名称 FC1 并选择所需的编程语言。

用 **OK** 确认其余的缺省设置。



功能 FC 被加入到 Blocks 文件夹。

双击打开 **FC1**

与功能块不同，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当该块关闭时仍可保留下来，例如，静态数据可以是用于“Speed(速度)”限值的存储位（见第五章）。
要编程功能，你可以使用符号表中的符号名。

在 **Help>Contents** 下的主题“设计自动化概念”，“设计程序结构的基础”以及“用户程序中的块”中可以找到更多的信息。

8.2 编程功能

在这一部分中，你将在我们的示例中编程一个定时器功能。当发动机接通时该定时器功能使能一个风扇接通（见第五章），然后，在发动机断开后该风扇继续运行 4 秒钟（延迟断开）。

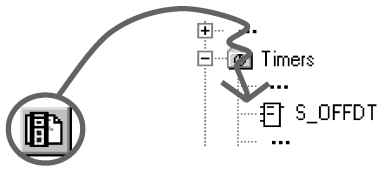
正如在前面提到的，你必须在变量声明表中指定功能的输入和输出参数（“in”和“out”声明）。

LAD/STL/FBD 编程窗口打开，使用该变量声明表的方法与使用功能块的声明表一样（见第五章）。

输入下列声明：

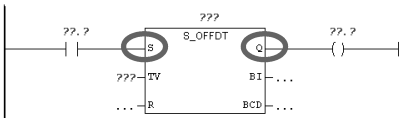
Address	Decl.	Name	Type	Initial Val	Comment
0.0	in	Engine_On	BOOL		Signal for switching on the engine
2.0	in	Timer_Function	TIMER		Timer function used for the switch-off delay
4.0	out	Fan_On	BOOL		Signal for switching on the fan
	in_out				
	temp				

用梯形逻辑编程定时器功能



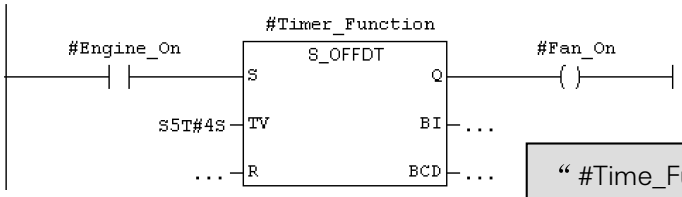
选择当前支路输入梯形图指令。

在编程元素目录中查找直到找到 S_OFFDT（启动延时断开定时器），选择该元素。



在输入 S 之前插入一个常开触点，在输出之后插入一个线圈 Q。

选择问号并输入变量声明表中相应的名字（#号会自动添加）。
在 S_OFFDT 的输入 TV 上设置延迟时间，S5T#4S 是一个定义为数据类型 S5Time#(S5T#)的常数，持续时间 4 秒（4S）。
然后保存该功能并关闭窗口。



“#Time_Function”由输入参数“#Engine_On”启动，当该功能在 OB1 中调用之后，它将被赋予汽油发动机的参数一次，柴油发动机的参数一次（例如，T1 “PE_Follow_on”）。你将在以后在符号表输入这些参数的符号名。

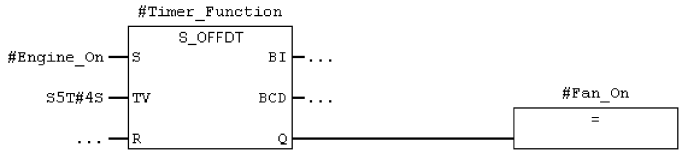
用语句表编程定时器功能

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

如果你用语句表编程，选择段下面的输入区域，按所示输入语句。
然后保存该功能并关闭窗口。

用功能块图编程定时的功能。

如果你用功能块图编程，选择段下面的输入区域，为定时器功能输入下面的 FBD 程序。
然后保存该功能并关闭窗口。



为使定时器功能得到处理，你要在块的分级结构中更高一级的块中调用该功能（在我们的示例中是在 OB1 中）。

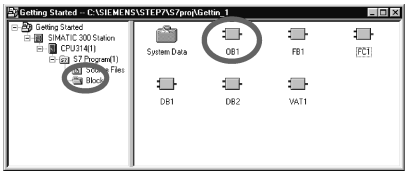
在 **Help>Contents** 下的主题“调用参考帮助”，“STL、FBD 或 LAD 语言描述”和“定时器指令”中可以找到更多的信息。

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 中用汽油发动机或柴油发动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。地址可以是绝对的或符号的。



SIMATIC Manager 连同项目“Getting Started”或你的新项目一同打开。

找到 **Blocks** 文件夹并打开 **OB1**，LAD/STL/FBD 编程窗口打开。

如果你已在第四章中从样板项目（GS-LAD_Example,GS-STL_Example, 或 GS-FBD_Example）中复制符号表到你的“Getting Started”项目，现在你就无须增加任何符号了。

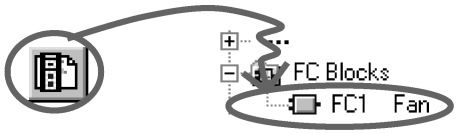
在以后增加符合名

用菜单命令 **Options>Symbol Table** 从 LAD/STL/FBD 编程窗口打开符号表，使用窗口右边的滚动条滚动到符号表底部。

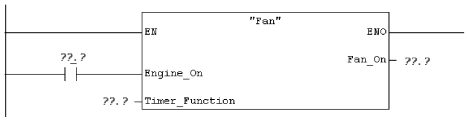
现在将下列符号加入到符号表：

Symbol	Address	Data Type	Comment
DE_Follow_On	T 2	TIMER	Follow-on time for diesel engine fan
PE_Follow_On	T 1	TIMER	Follow-on time for petrol engine fan
Fan	FC 1	FC 1	Fan control
PE_Fan_On	Q 5.2	BOOL	Command for switching on petrol engine fan
DE_Fan_On	Q 5.6	BOOL	Command for switching on diesel engine fan

用梯形逻辑编辑调用



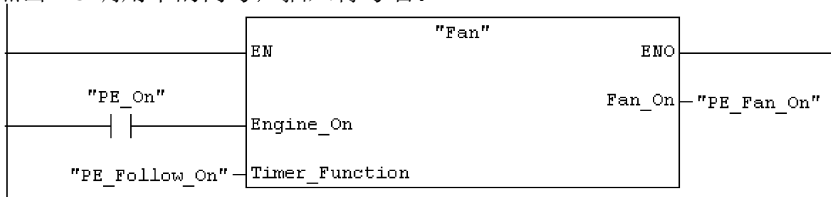
你在 LAD 视窗中。插入一个新段（第六段）。然后在编程元素目录中查找直到找到 FC1，插入该功能。



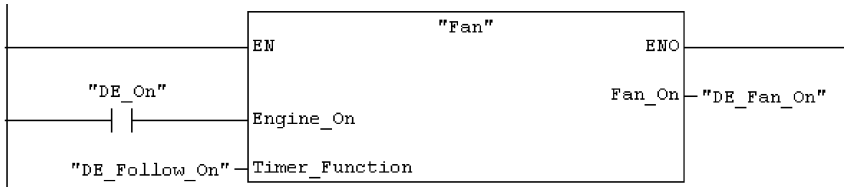
在“Engine_On”之前插入一个常开触点。

使用菜单命令 View>Display>Symbolic Representation 可以在符号地址和绝对地址之间切换。

点击 FC 调用中的问号，插入符号名。



在第七段中编辑对功能 FC1 的调用并使用柴油发动机制的地址。
方法与前一段一样（你已经将柴油发动机的地址加入到符号表中）。



保存该块然后关闭窗口

激活菜单命令 View>Display>Symbol Information 显示每个段中各个地址的信息。

要在屏幕上显示几个段，释放菜单命令 View>Dispaly>Comment。

使用菜单命令 View>Zoom Factor,可以改变显示的段的大小。

用语句表编程调用

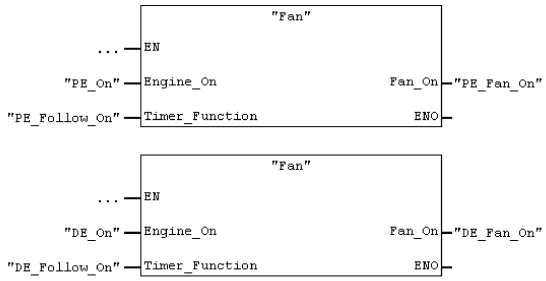
```
Network 6 : Controlling the Fan for the Petrol Engine
   CALL  "Fan"
    Engine_On     :="PE_On"
    Timer_Function:="PE_Follow_On"
    Fan_On        :="PE_Fan_On"

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"
```

如果你使用语句表编程，在一个新段的下面选择输入区域，输入这里所示的 STL 语句。
然后存储该调用并关闭窗口。

用功能块图编程调用。

如果使用功能块图编程，在一个新段下面选择输入区域并输入下面所示的 FBD 指令。
然后存储该调用并关闭窗口。



在我们的示例中所编程的功能的调用是一个无条件调用；即，该功能总会被处理。
根据你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，一个输入或一个前面的逻辑操作结果。框图中的输入 EN 和输出 ENO 就是用于程序的条件调用。

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

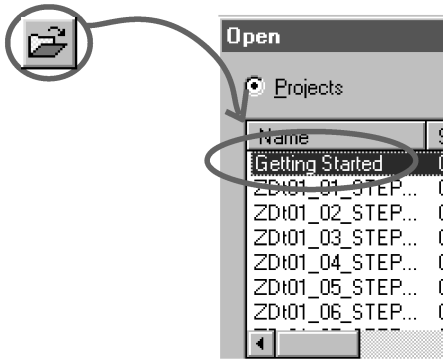
9 编程一个共享数据块

9.1 创建并打开共享数据块

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。

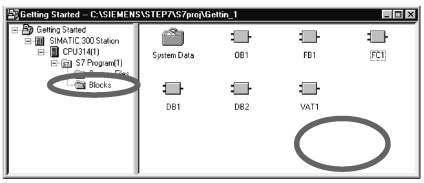
存储在共享数据块中的数据可以被其它的任意一个块使用。而一个背景数据块被指定给一个特定的功能块，它的数据只在这个功能块内局域有效(见 5.5 部分)。

你应该已经熟悉了用梯形逻辑、功能块图或语句表编程(见第四章和第五章)以及符号编程(见第三章)。



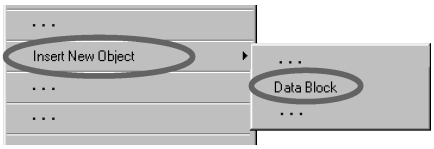
如果你已在第一至第七章中创建并使用了样本项目“Getting Started”，现在打开它。

如果没有，用菜单命令 **File> “New Project ” Wizard** 在 SIMATIC Manager 中创建一个新的项目。要这么做可遵循 2.1 部分中的指导并重新命令该项目为“Getting Started Function”。

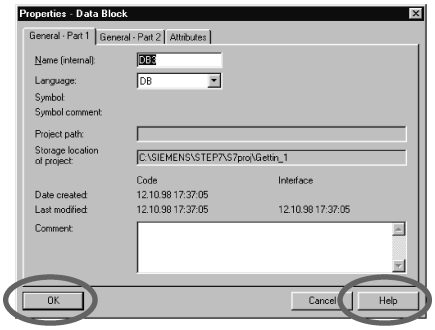


我们将继续使用“Getting Started”项目。而你使用一个新项目也可以实现每一步。

定位到 **Blocks** 文件夹并打开它。用鼠标右键点击右半窗口。



从弹出菜单中插入一个 **Data Block**(DB)。

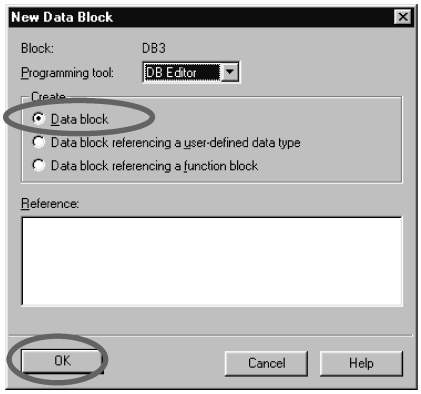


在“Properties-Data Block”对话框中用 **OK** 接受所有的缺省设置。

需要更进一步的信息使用“Help”按钮。

数据块 DB3 已被加入到 **Blocks** 文件夹中。

双击打开 **DB3**



在随后出现的“New Data Block”对话框中激活选项 **Data block**(数据块)。

用 **OK** 关闭对话框。

回忆：在 5.5 部分中你通过激活选项“Data block referencing a function block(参考一个功能块的数据块)”生成一个背景数据块。与之相反，用“Data block”可以生成一个共享的数据块。

在数据块中编辑变量

Address	Name	Type	Initial Value	Comment
0.0		STRUCT		
=0.0		END_STRUCT		

在 Name(名字)栏中输入 #“PE_Actual_Speed”。

点击鼠标右键选择类型，使用弹出菜单中的菜单命令 **Elementary Types>INT.**

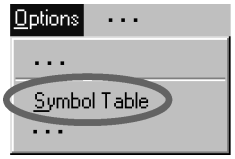
下面的示例中，DB3 中定义了三个共享数据，在变量声明表中按照这些数据输入。

Address	Name	Type	Initial Value	Comment
0.0		STRUCT		
+0.0	PE_Actual_Speed	INT	0	Actual speed for petrol engine
+2.0	DE_Actual_Speed	INT	0	Actual speed for diesel engine
+4.0	Preset_Speed_Reached	BOOL	FALSE	Both engines have reached the preset speed
=6.0		END_STRUCT		

对于数据块中用于实际速度的变量“PE_Actual_Speed”和“DE_Actual_Speed”的处理与存储字 MW2(PE_Actual_Speed)和 MW4(DE_Actual_Speed)相同。这将在下一章中看到

保存该共享数据块。

分配符号



你也可以给数据块分配一个符号名。

打开 **Symbol Table**(符号表)为数据块 DB3 输入符号名“S_Data”。

如果你在第四章中从样本项目(zEn01_02_STEP 7_STL_1-10，zEn01_06_STEP 7_LAD_1-10 或 zEn01_04_STEP 7_FBD_1-10)中拷贝符号表到你的“Getting Started”项目，现在你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“Symbol Editor(符号编辑器)”窗口。

也关闭共享数据块的变量声明表。

变量声明表中的共享数据块：

使用菜单命令 **View>Data View** 可为共享数据块修改表中数据类型 INT 的实际值(见 5.5 部分)。

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 **Help>Contents** 下的主题“编程块”和“创建数据块”中可以找到更多的信息。

10 编程一个多重背景

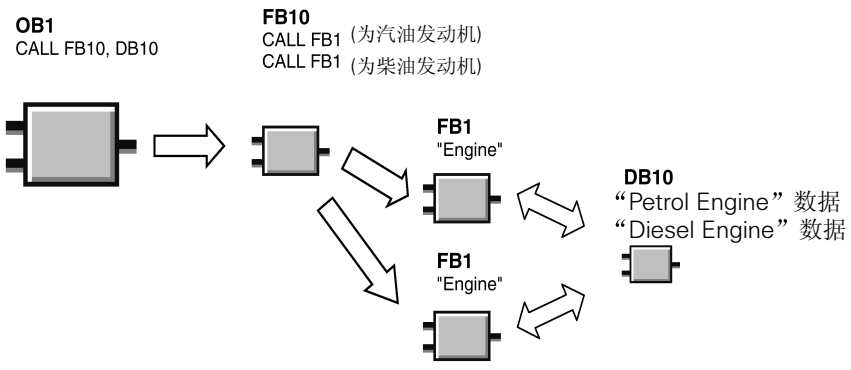
10.1 创建并打开较高一级的功能块

在第五章中你创建了一个程序，用功能块“Engine”(FB1)控制一个发动机。当功能块 FB1 在组织块 OB1 中调用时，使用了数据块“Petrol”

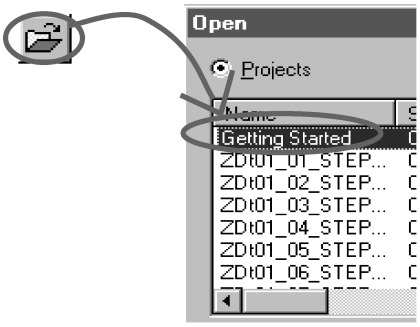
(DB1)和“Diesel”(DB2)。每个数据块包含发动机不同的数据(例如，#Preset_Speed)。现在想像一下，你的自动化设备还需要其它的程序控制发动机；例如，一个用于菜籽油发动机的控制程序，或一个氢发动机，等等。按照目前你已学过的步骤，现在你要为一个附加的发动机控制程序使用 FB1，并且为这个发动的每一次的数据分配一个新的数据块；例如，FB1 和 DB3 用于控制菜籽油发动机，FB1 和 DB4 用于氢发动机，等等，当你创建新的发动机控制程序时，块的数量的增加是相当可观的。

另一方面，通过使用多重背景则可以减少块的数量。要这么作，你要创建一个新的，较高一级的功能块(在我们的示例中是 FB10)，并在其中调用未作任何修改的 FB1 作为一个“局域背景”。对每一个调用，子程序 FB1 将它的数据存储在较高一级 FB10 的数据块 DB10 中。这意味着你无须给 FB1 分配任何数据块。所有的功能块都指向一个数据块(这里是 DB10)。

数据块 DB1 和 DB2 被集成在 DB10 中，要这么作，必须在 FB10 的静态局域数中声明 FB1。

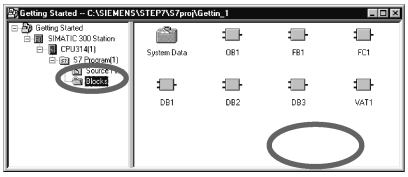


你应该已经熟悉了用梯形逻辑、功能块图或语句表编程(见表四和第五章)以及符号编程(见第三章)。



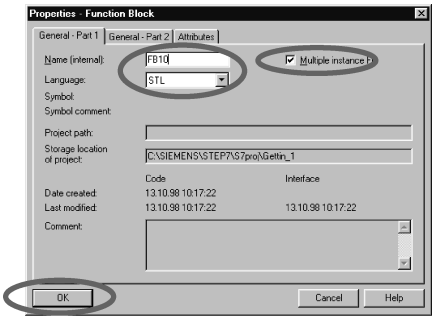
如何你已在第一至第七章中创建并使用了“Getting Started”示例，打开“Getting Started”项目。

如果没有，可在 SIMATIC Manager 中打开以下项目之一：
zEn01_06_STEP 7_**LAD**_1-9 用于梯形逻辑，
zEn01_02_STEP 7_**STL**_1-9 用于语句表
zEn01_04_STEP 7_**FBD**_1-9 用于功能块图。



定位到 **Blocks** 文件夹并打开它。

用鼠标右键点击右半窗口，用弹出菜单插入一个功能块。



将块名改为 FB10 并选择所需的编程语言。

激活 **Multiple instance FB**(如果有必要)用 **OK** 确认其余的缺省设置。

FB10 被加入到 Blocks 文件夹。双击打开 **FB10**。

你可以为任意功能块创建多重背景，例如，为阀门控制程序。如果你要使用多重背景，注意调用块和被调用块都必须有多重背景的能力。

在 **Help>Contents** 下的主题“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“局域背景(local instance)”调用，则需为每一个 FB1 的调用声明一个有不同名字的静态变量。这里，数据类型是 FB1(“Engine”)

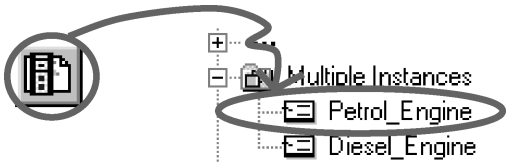
填写变量声明表

LAD/STL/FBD 编程窗口打开。为 FB1 的调用声明以下变量：

Address	Decl.	Name	Type	Initial Val	Comment
	in				
0.0	out	Preset_Speed_Reached	BOOL	FALSE	Both engines have reached the preset speed
	in_out				
2.0	stat	Petrol_Engine	"Engine"		First local instance of FB1 "Engine"
10.0	stat	Diesel_Engine	"Engine"		Second local instance of FB1 "Engine"
0.0	temp	PE_Preset_Speed_Reached	BOOL		Preset speed reached (petrol engine)
0.1	temp	DE_Preset_Speed_Reached	BOOL		Preset speed reached (diesel engine)

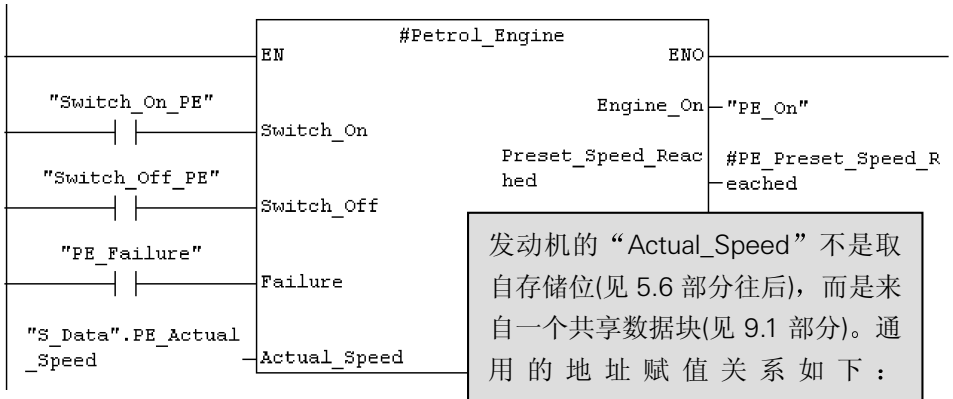
声明的局域背景将出现在编程元素目录的“Multiple Instance (多重背景)下面。

用梯形逻辑编程 FB10



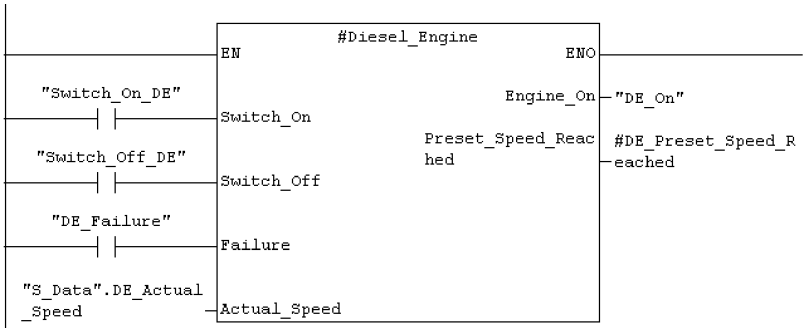
在第一段中插入调用“Petrol_Engine”作为多重背景块“Petrol_Engine”。

然后插入所需的常开触点并用符号名完成调用。

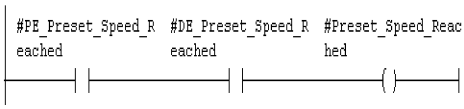


发动机的“Actual_Speed”不是取自存储位(见 5.6 部分往后)，而是来自一个共享数据块(见 9.1 部分)。通用的地址赋值关系如下：“Data_Block” Addres, 例如：“S_Data”. PE_Actual_Speed。

插入一个新段并为柴油发动机编程调用。按照与第 1 段相同的方法进行。



插入一个新段并用相应的地址编程一个串联电路。然后保存你的程序并关闭该块。



临时变量(“PE_Preset_speed_ Reached”和“DE_ Preset_Speed _Reached”)被提供给输出参数“Preset_Speed _Reached”，它将在 OB1 中被进一步处置。

用语句编程 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
```

如果你使用语句表编程，在一个新段下面选择输入区域，输入此处所示的 STL 指令。

然后保存你的程序并关闭该块。

用功能块图编程 FB10

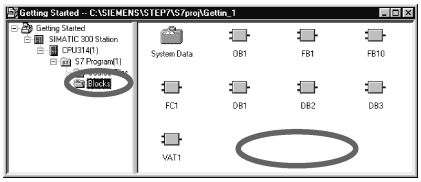
如果你用功能块图编程，在一个新段下选择输入区域并输入下面的 FBD 指令。然后保存你的程序并关闭该块。

要在 FB10 中编辑对 FB1 的两次调用，FB10 本身必须被调用。
多重背景只能为功能块编制。为功能(FC)创建多重背景是不可能的。

在 **Help＞Contents** 下的主题“编程块”，“创建逻辑块”，和“变量声明表中的多重背景”中可找到更多的信息。

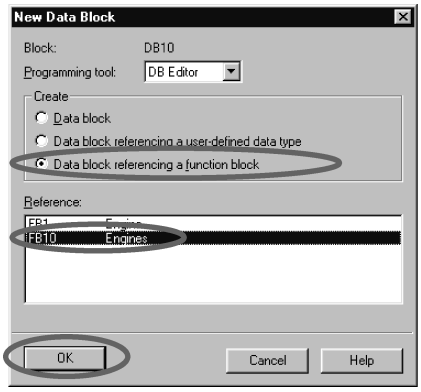
10.3 生成 DB 10 并调整实际值

新的数据块 DB10 将替代数据块 DB1 和 DB2。用于汽油发动机和柴油发动机的数据存储在 DB10 中，后面在 OB1 中调用 FB1 时将会用到(见从 5.6 部分往后的“OB1 中调用 FB1”)。



用弹出菜单在 SIMATIC Manager 中，在项目“Getting Started”的 **Blocks** 文件夹中创建数据块 DB10。

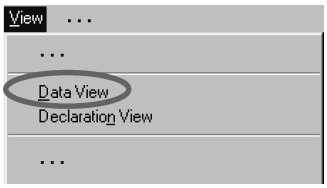
要这么作，在出现的对话框中修改数据块的名字为 DB10，用 **OK** 确认其余的设置。



数据块 DB10 已被加入，打开该块显示“New Data Block”dialog box。

激活选项 **Data block referencing a function block**(参考一个功能块的数据块)并选择 FB10。

用 **OK** 确认设置。



数据块 DB10 被打开。选择菜单命令 View＞Data View。

数据视窗显示 DB10 中的每一个变量，包括 FB1 的两次调用的内部变量(“局域背景”)。声明视窗中变量的显示和它们在 FB10 中声明的一样。

将柴油发动机的实际值改为“1300”，存储该块然后关闭它。

Address	Decl.	Name	Type	Initial Value	Actual Value	Comment
0.0	out	Preset_Speed_Reached	BOOL	FALSE	FALSE	Both engines have reached
2.0	stat:in	Petrol_Engine.Switch_On	BOOL	FALSE	FALSE	Switch on engine
2.1	stat:in	Petrol_Engine.Switch_Off	BOOL	FALSE	FALSE	Switch off engine
2.2	stat:in	Petrol_Engine.Failure	BOOL	FALSE	FALSE	Engine failure, causes the
4.0	stat:in	Petrol_Engine.Actual_Speed	INT	0	0	Actual engine speed
6.0	stat:out	Petrol_Engine.Engine_On	BOOL	FALSE	FALSE	Engine is switched on
6.1	stat:out	Petrol_Engine.Preset_Speed_Reached	BOOL	FALSE	FALSE	Preset speed reached
8.0	stat	Petrol_Engine.Preset_Speed	INT	1500	1500	Requested engine speed
10.0	stat:in	Diesel_Engine.Switch_On	BOOL	FALSE	FALSE	Switch on engine
10.1	stat:in	Diesel_Engine.Switch_Off	BOOL	FALSE	FALSE	Switch off engine
10.2	stat:in	Diesel_Engine.Failure	BOOL	FALSE	FALSE	Engine failure, causes the
12.0	stat:in	Diesel_Engine.Actual_Speed	INT	0	0	Actual engine speed
14.0	stat:out	Diesel_Engine.Engine_On	BOOL	FALSE	FALSE	Engine is switched on
14.1	stat:out	Diesel_Engine.Preset_Speed_Reached	BOOL	FALSE	FALSE	Preset speed reached
16.0	stat	Diesel_Engine.Preset_Speed	INT	1500	1300	Requested engine speed

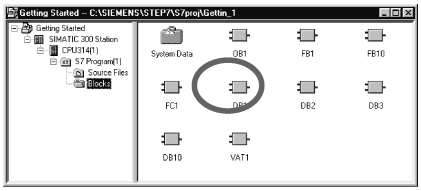
现在所有的变量都包含在 DB10 的变量声明表中。在前一半中，你可以看到调用功能块“Petrol_Engine”的变量，后一半中则是调用功能块“Diesel_Engine”的变量(见 5.5 部分)。

FB1 的“内部”变量仍保持它们的符号名；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 **Help＞Contents** 下的主题“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 FB 10

在我们的示例中，在 OB1 中调用 FB10。这个调用与你已经学过的在 OB1 中对 FB1 的调用具有相同的功能(见 5.6 部分往后)。使用多重背景，你可以替换 5.6 部分中编好的段 4 和段 5。



打开项目中的 **OB1**，你刚在该项目中编过 FB10。

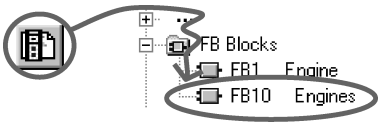
> 如果你已在第四章中从样本项目(zEn01_02_STEP7_STL_1-10, zEn01_06_STEP7_LAD_1-10 或 zEn01_04_STEP7_FBD_1-10)中拷贝了符号表到你的“Getting Started”项目，现在你就不需要增加任何符号了。

定义符号名

LAD/STL/FBD 编程窗口打开。用菜单命令 **Options＞Symbol Table** 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。然后保存符号表并关闭该窗口。

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
...	...	...	...

用梯形逻辑编程调用

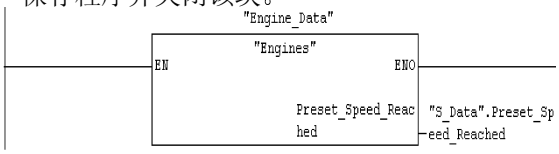


在 OB1 的结尾处插入一个新段并增加对 FB10(“Engines”)的调用。

用相应的符号名完成下面的调用。

删除 OB1 中对 FB1 的调用(来自 5.6 部分中的段 4 和段 5)，因为我们现在通过 FB10 来调用 FB1。

保存程序并关闭该块。



FB10(“Engines”)的输出信号“Preset_Speed_Reached”被传送给共享数据块的变量。

用语句表编程调用

如果你使用语句表编程，在一个新段下面选择输入区域并输入下面的 STL 指令。要这么作，使用编程元素目录中的 FB BLocks＞FB10 Engines。
删除 OB1 中对 FB1 的调用(5.6 部分的段 4 和段 5)，因为我们现在通过 FB10 调用 FB1。
然后保存程序并关闭该块。

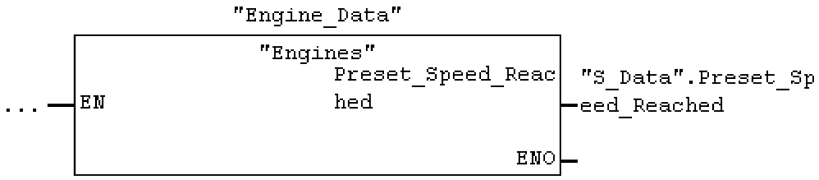
```
CALL  "Engines" , "Engine_Data"
 Preset_Speed_Reached:="S_Data".Preset_Speed_Reached
```

用功能块图编程调用

如果你使用功能块图编程，在一个新段下选择输入区域并输入下面的 FBD 指令。要这么作可使用编程元素目录中的“**FB Blocks＞FB10 Engines**。

删除 OB1 中对 FB1 的调用(5.6 部分的段 4 和段 5)，因为我们现在通过 FB10 来调用 FB1。

然后保存程序并关闭该块。



如果你的自动化任务还需要更多的发动机控制程序；例如，为汽油发动机、氢发动机等等……，你可用相同的方法将它们编作多重背景并在 FB10 中调用它们。要这么作，象 FB10 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)。然后，你可以定义新的符号名；例如，在符号表中为接通(Switch_on)和断开(Switch_off)过程定义符号。

在 **Help＞Contents** 下的主题“调用参考帮助”，“STL、FBD 或 LAD 语言描述”和“程序控制指令”中可以找到更多的信息

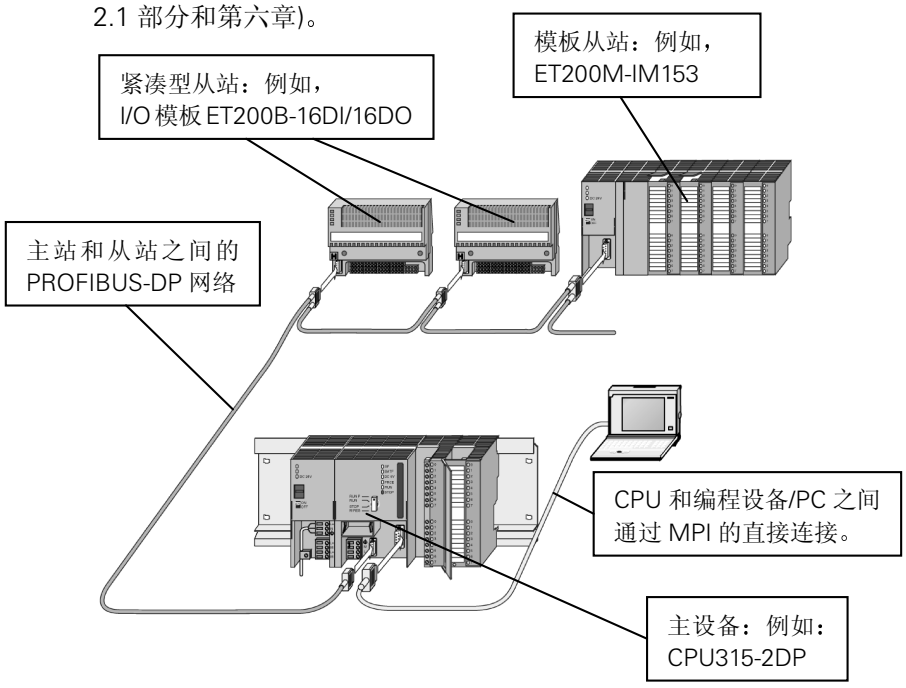
11 组成分布式 I/O

11.1 用 PROFIBUS DP 组态分布式 I/O

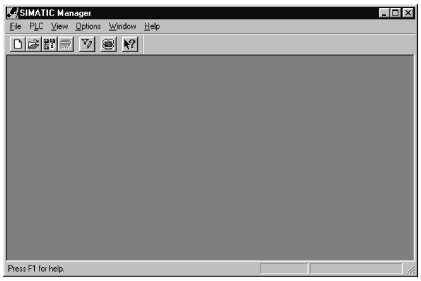
具有常规组态的自动化系统用电缆来连接传感器和执行器，这些电缆直接插到中央可编程控制器的 I/O 模板上。这通常意味着需要大量的接线。

使用分布式组态，通过将输入输出模板放置到离传感器和执行器很近的地方，可以减少大量的接线。你可以使用 PROFIBUS DP 来建立可编程逻辑控制器、I/O 模板和现有设备之间的连接。在第六章中你能够找到如何编程一个常规组态。创建一个中央组态和创建一个分布式组态是没有区别的。你可以从硬件目录中选择要用的模板并把它们安置在机架上，按照你的要求修改它们的特性。

在读这一章之前，最好你已熟悉了创建项目和编程一个中央组态(见 2.1 部分和第六章)。

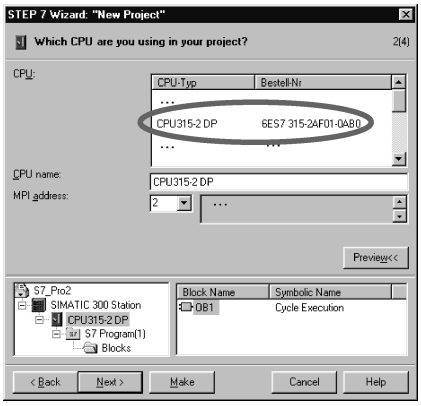


创建一个新项目



起始点在 SIMATIC Manager。为使这一过程进行的顺利，关闭所有打开的项目。

创建一个新项目

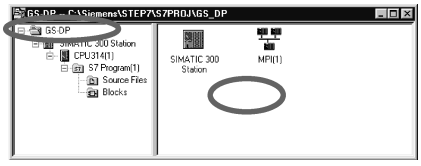


在相应的对话框中选择 **CPU 315-2DP**(带有 PROFIBUS-DP 网的 CPU)。

现在按照 2.1 部分中相同的方法进行，给该项目分配一个名字“GS-DP”(Getting Started Distributed I/O)。

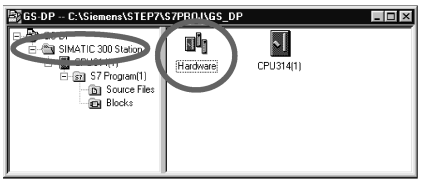
如果你想在这里生成你自己的组态现在可以指定你的 CPU，注意你的 CPU 必须支持分布式 I/O。

插入 PROFIBUS 网

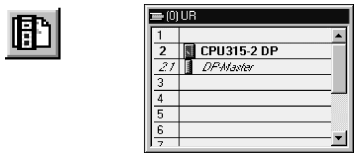


选择 **GS-DP** 文件夹并用鼠标右键点击右半窗口插入 **PROFIBUS** 网。

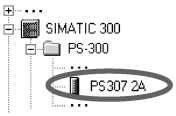
组态站



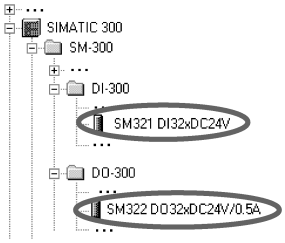
选择 SIMATIC 300 Station 文件夹并双击 Hardware(硬件)，“HW Config（硬件组态)”窗口打开(见 6.1 部分)



CPU315-2DP 已经出现在机架中。如果有必要，用菜单命令 View＞Hardware Catalog 或工具栏中相应的按钮打开硬件目录。



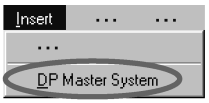
用拖放功能将电源模板
PS 307 2A 拖至 1 号槽。



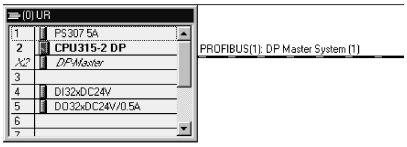
用同样的方式，在 4 号槽和 5 号槽插入 I/O 模板
DI 32×DC 24V 以及 DO 32×DC24V/0.5A。

除了支持分布式 I/O 的 CPU 之外，你也可以在同一机架上放置其它的 CPU(这里不讨论)。

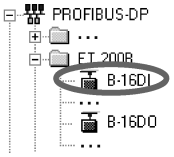
组态 DP 主站系统



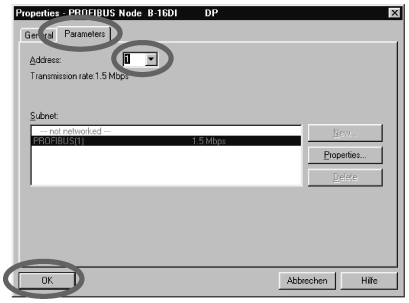
选择槽 2.1 中的 DP 主站，并插入 **DP Master system**.(DP 主站系统)



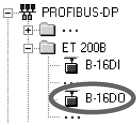
你可以通过按住鼠标左键拖动来移动你放在主站系统中的任意对象。



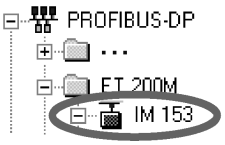
在硬件目录中查找到模板 **B-16DI**，将该模板插入到主站系统(拖动对象到主站系统直到光标变为一个“+”号，放开该对象)。



你可以在“Properties(特性)”对话框的“Network Connection(网络连接)”标签中修改你已插入的模板的站地址。
用 **OK** 确认推荐的地址 **1**。

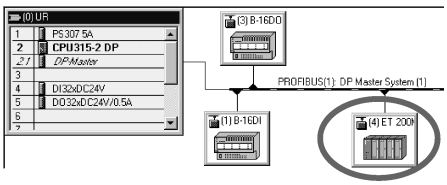


用同样的方式可将模板 **B-16DO** 拖放到主站系统。
在对话框中站地址会自动调整，用 **OK** 确认该输入项。

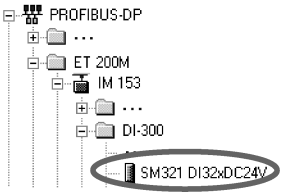


将接口模板 **IM153** 拖至主站系统并用 **OK** 再次确认站地址。

在我们的示例中，我们使用缺省站地址。但是，你可以在任何时候按照你的要求修改这些地址。



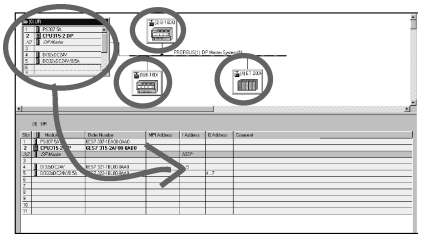
在网络中选择 **ET 200M**。ET 200M 的空槽显示在组态表的下部。这里选择 **4** 号槽。



ET200M 自身可以有附加的 I/O 模板。例如，为 4 号槽选择 **DI32×DC 24V**，双击插入该模板。

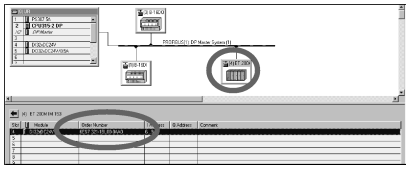
在使用硬件目录时要确认你是在正确的文件夹中。例如，为 ET 200M 选择模板应在 ET 200M 文件夹中查找。

修改站地址

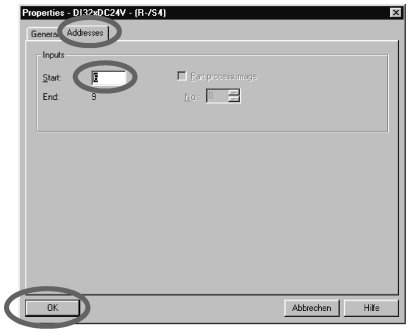


在我们的示例中，我们不需要修改站地址。然而在实际当中却经常需要这样做。

一个一个地选择其它站，检查输入和输出地址。“组态硬件”应用程序已调整了所有地址，所以不会有双重赋值。



让我们试想一下，你要修改 ET 200M 的地址：
选择 **ET 200M** 并双击
DO32×DC 24V/0.4A(槽 4)。

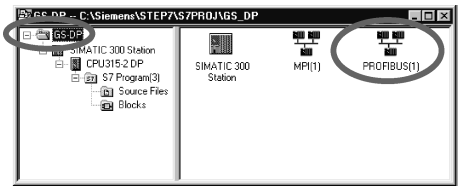


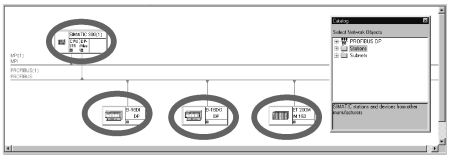
现在修改“Properties”对话框的“Addresses(地址)”标签中的输入地址，从 6 改为 **12**。用 **OK** 关闭对话框。

最后，**保存并编译**(save and compile)分布式 I/O 组态。关闭窗口。

菜单命令 Save and Compile 意味着自动对组态作一致性检查。如果没错，生成系统数据，该系统数据可下载到 CPU。
使用 Save，你可以保存有错的组态。但是，你无法将该组态下载到可编程控制器。

可选软件：组成网络





你也可以使用可选软件包：“Configuring Networks(组态网络)”来组态分布式 I/O。

在 SIMATIC Manager 中双击网络 **PROFIBUS(1)**。

“NETPRO”窗口打开。你可以从网络对象的目录中将其它的 DP 从站拖放到 PROFIBUS DP。双击任意组件，对它进行组态。“Configuring Hardware(组态硬件)”窗口打开。

使用菜单命令 **Station＞Consistency Check**("Configuring Hardware”窗口)以及 **Network＞Consistency Check**(“Configuring Networks”窗口)，可以在存储之前对组态查错。所有错误都会显示出来，STEP7 将推荐可能的解决方案。

在 **Help＞Contents** 下的主题“组态硬件”和“组态分布式 I/O”中可以找到更多的信息。

祝贺你!

你已完成了入门手册并且学习了 STEP 7 最重要的术语、方法步骤和功能。现在你可以开始你的第一个项目了。

在将来的项目工作中，如果要查找特定的功能或者忘记了任何 STEP 7 的任何操作指令，你都可以使用我们关于 STEP 7 的综合帮助。如果你想进一步地扩展 STEP 7 的知识，我们有许多专门的培训课程，你当地的西门子代表处将会非常高兴地为您提供帮助。我们希望你的项目获得成功!

Siemens AG.

附录 A

入门手册中的样本项目概述

- zEn01_02_STEP 7_STL_1-10:
  用 STL 编程语言编程的第一至第十章，包括符号表。
- zEn01_01_STEP 7_STL_1-9:
  用 STL 编程语言编程的第一至九章，包括符号表。
- zEn01_06_STEP 7_LAD_1-10:
  用 LAD 编程语言编程的第一至第十章，包括符号表。
- zEn01_05_STEP 7_LAD_1-9:
  用 LAD 编程语言编程的第一至第九章，包括符号表。
- zEn01_04_STEP 7_FBD_1-10:
  用 FBD 编程语言编程的第一至第十章，包括符号表。
- zEn01_03_STEP 7_STL_1-9:
  用 FBD 编程语言编程的第一至第九章，包括符号表。
- zEn01_07_STEP 7_Dist_IO：
  编程的有分布式 I/O 的第十一章。

索引