

1. 前言

本文可以作为 S7-Hi Graph 编程语言的使用参考资料，希望读者通过对本文的阅读，能够更快地学习 S7-Hi Graph 编程语言。西门子提供了 S7-Hi Graph 编程语言的详尽手册，在安装 S7-Hi Graph 软件包后，通过点击 Windows 菜单 开始->Simatic->Documentation->English 可以阅读到名称为“ S7-Hi Graph - Programming State Graphs” 的 PDF 手册。此手册共分为 11 个章节，其详细地讲解了 S7-Hi Graph 编程语言。一切关于 S7-Hi Graph 使用的问题请以此手册为准。

相对于其它编程语言，S7-Hi Graph 有着它本身的一些特性，一些独特的概念也比较难于理解。但还将在本文对 S7-Hi Graph 手册中的部分内容作出额外强调与讲解，虽然这些讲解与手册有重叠之处，但希望通过这些讲解能够使用户更快地掌握 S7-Hi Graph 的使用。在讲解内容之后，本文又列举出一些用户在实际使用当中经常会遇到的问题，希望能够对用户有所帮助。

相关手册地址连接：

使用 STEP 7 V5.3 编程

<http://support.automation.siemens.com/CN/view/zh/18652056>

S7-HiGraph V5.3

<http://support.automation.siemens.com/CN/view/zh/1137299>

S7-300 和 S7-400 的语句表 (STL) 编程

<http://support.automation.siemens.com/CN/view/zh/18653496>

2. 软件的基本信息

2.1. S7-Hi Graph 简介

S7-Hi Graph 具有以下特点：

- ◇ 通过绘制功能图表来实现异步控制
- ◇ 集成了信号监控及触发功能
- ◇ 非常适合于机械设计工程师，调试及维护工程师
- ◇ 利于自动化工程师与机械工程师相互沟通
- ◇ 适用于 SIMATIC S7-300 (推荐用于CPU314以上CPU), S7-400, C7 and WinAC

S7-Hi Graph 不仅具有 PLC 典型的元素（例如 输入 / 输出，定时器，计数器，符号表），而且还具有图形化编程语言语言的特性，其非常适合于如下任务：

- ◇ 异步控制
- ◇ 自动机械设计

2.2. S7-Hi Graph 与 STL

S7-Hi Graph 程序最终编译成 STL。其代码量相对于 STL 编程有所增加(推荐用于 CPU314 以上 CPU)。

2.3. S7-Hi Graph 的安装与使用

STEP7 各版本均不包括 S7-Hi Graph 软件包及授权,需单独购买。在 S7 程序中,S7-Hi Graph 块可以与其它 STEP7 编程语言生成的块组合互相调用。S7-Hi Graph 生成的块也可以作为库文件被其它语言引用。

2.4. S7-Hi Graph 软件兼容性

不同 S7-Hi Graph 软件版本与 STEP7 及操作系统之间的兼容性:图中的 X 表示兼容,- 表示不兼容

Version	Order Number	STEP 7 V5.3			STEP 7 V5.4				
		Win 2000 SP4	Win XP SP1	Win XP SP2	Win 2000 SP4	Win XP SP1	Win XP SP2	Win 2003	Win 2003 SP1
V5.3	6ES7 811-3CC05-0YA5	X	X	X	X	X	X	-	-
V5.2	6ES7 811-3CC04-0YE0	X	X	(-)	X	X	(-)	-	-
V5.0	6ES7 811-3CC03-0YE0	X ¹⁾	-	-	(-)	-	-	-	-

表 2-1 S7-Hi Graph 软件兼容性

注: 1) 仅仅 S7-Hi Graph V5.0+SP1 或以后版本支持

2.5. S7-Hi Graph 中英文词汇对照关系

由于很多英文的科技专用词汇没有明确统一的中文词汇与之对应,所以在本文的讲解当中,将尽量保持手册中的英文信息,减少使用中文词汇代替英文专用词汇。本文中使用的替代的中英文词汇对照关系如下:

- State 状态
- Transition 转换
- State graph 状态图
- Graph group 图表组
- Action 动作
- Message 消息

3. S7-HiGraph 基本概念

3.1. S7 程序构成

作为 STEP7 的选项包,S7-Hi Graph 软件在安装后,将被集成在 STEP7 中使用。S7-Hi Graph 编程界

面为图形界面，Graphs Group 包含若干个 State Graph。当编译 Graphs Group 时，其生成的块以 FC+DB 的形势出现，此 S7-Hi Graph FC 必须周期调用，例如：OB1，可参考下图。

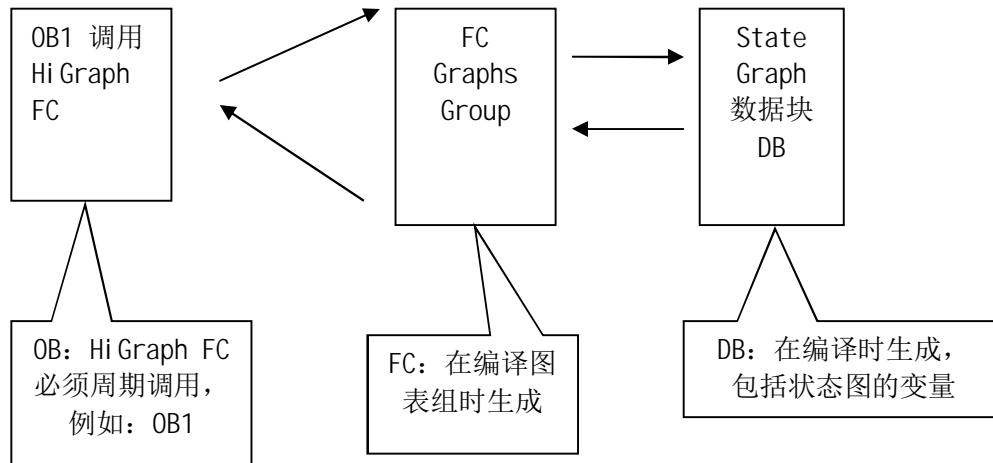


图 3-1: S7 程序构成

备注：S7-Hi Graph 块可以与其它 STEP7 编程语言生成的块组合互相调用，S7-Hi Graph 生成的块也可以作为库文件被其它语言引用，但是如果使用者对 S7-Hi Graph 认识模糊，不清楚 S7-Hi Graph 程序结构，这种操作可能会引起不可预料的后果（如人身伤害，设备损坏等）。

3.2. S7-Hi Graph 程序结构

S7-HiGraph 程序分为 3 级结构，参见下图：

1. 状态+转换条件，若干个这两类元素构成状态图
2. 状态图，多个状态图构成图表组
3. 图表组

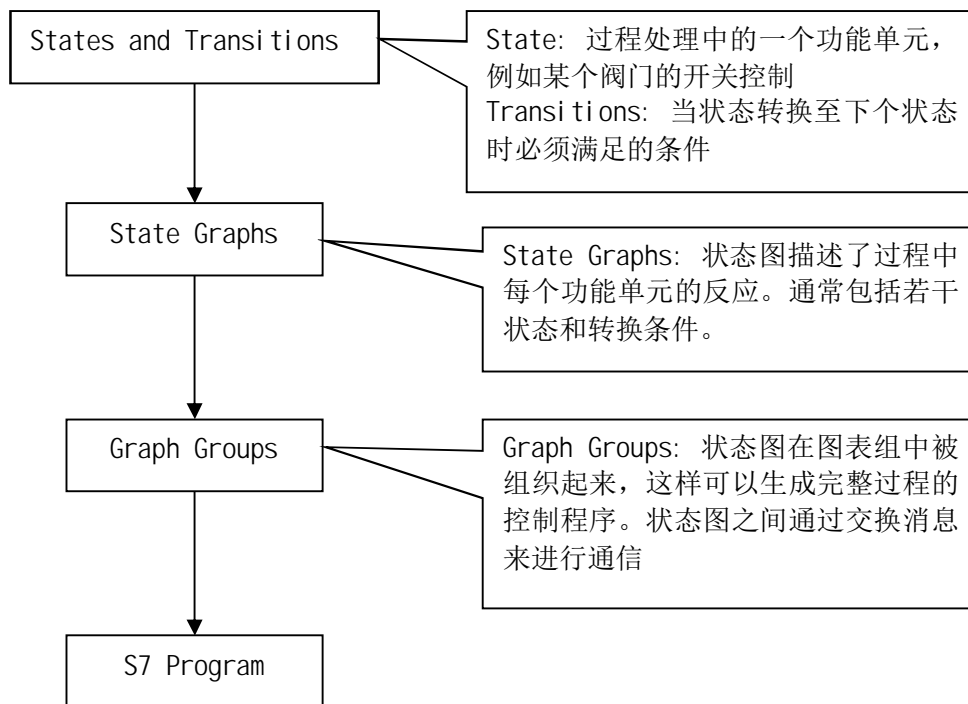


图 3-2: S7-HiGraph 程序结构

由此，我们可以发现 S7-Hi Graph 的一些优点

- 通过图形化的界面，S7-Hi Graph 提高了 STEP7 的编程功能。
- 状态图是控制和镜像 STEP7 控制对象的有效工具
- 过程处理镜像到状态图，程序机构和工艺的描述也变得很容易
- 这样程序结构清晰，对于维护和修改工作都很方便

3.3. S7-Hi Graph 与 S7-GRAPH 的比较:

对于 S7-GRAPH，比较侧重的是顺序控制，可以实现单个/多个的顺控器单独/协调工作。

对于 S7-Hi Graph，在实现顺序控制的基础上，还可以实现如下功能:

- 在 S7-Hi Graph 生成的状态图可以被封装为标准元件，
- 完成封装后的标准元件，可以在图表组中多次重复使用，并分配给不同的实参（即面向对象的标准化编程）
- 标准元件之间可以通过消息或全局变量协同工作。

3.4. S7-Hi Graph 状态图的重复使用

例如：假设某条物流线上共有 100 个传送带，但传送带类型只有 6 种类型。如果使用 S7-Hi Graph 进行编程，编程人员只需要设计 6 种状态图，在这 6 个不同的状态图中分别描述了 6 种不同传送带的特性。在随后的图表组编程中，把这 6 种不同的状态图分别引用若干次，并分配各自的参数，就构成了一条由 100 个传送带构成的物流线。关于图表组的详细应用，将本文中专门讲解。

3.5. S7-Hi Graph 项目完整流程

在手册中利用一个完整 S7-Hi Graph 项目的例子，给出了 S7-Hi Graph 项目完整流程，此流程大致分为 16 步：

1. 定义功能单元及状态图。在程序规划时，可以把完整控制任务分解成若干个功能单元，此单元可以是一个相对独立的设备或功能。
2. 设计状态图。把每个状态图分解为若干个状态及转换条件。确定状态之间转换的顺序。
3. 规划每个状态图的输入/输出信号
4. 新建 S7-Hi Graph 项目
5. 生成符号表
6. 生成状态图
7. 在状态图中定义变量
8. 插入状态和转换条件
9. 输入状态名
10. 输入动作和转换条件
11. 生成图表组
12. 分配参数
13. 编译状态组
14. 编程 OB1
15. 下载用户程序
16. 调试用户程序

4. S7-HiGraph 基本使用

4.1. 用户界面

首先，在 STEP7 当中生成一个新项目，用右键点击 Sourc 文件夹，插入一个新的 State graph, 如下图：

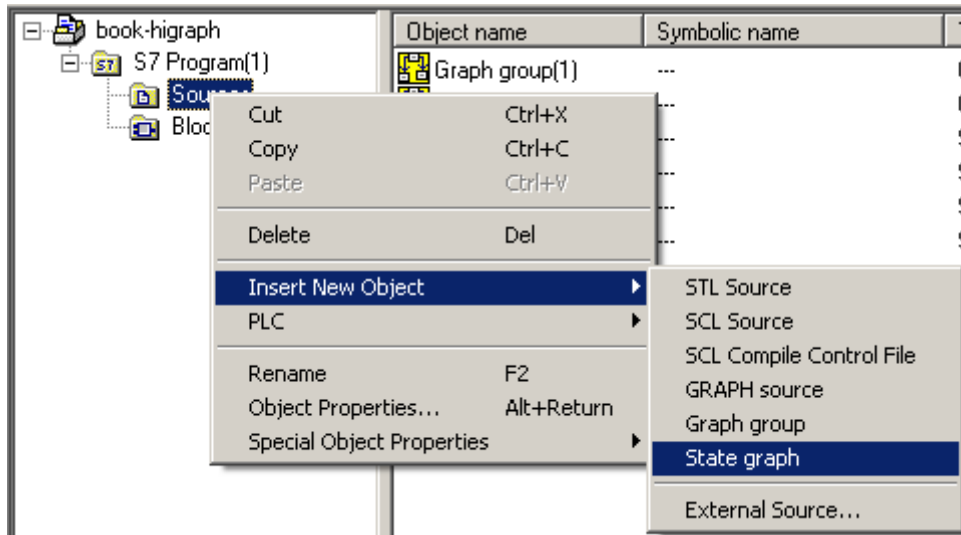


图 4-1: 生成新的 S7-HiGraph 程序

注意: 用户可以修改新插入的 State graph 或 Graph group (一般在 State graph 编辑完成, 整体规划项目时插入) 的名字, 以便于记忆。

双击新生成的 State graph 后, 可以打开用户界面, 如下图:

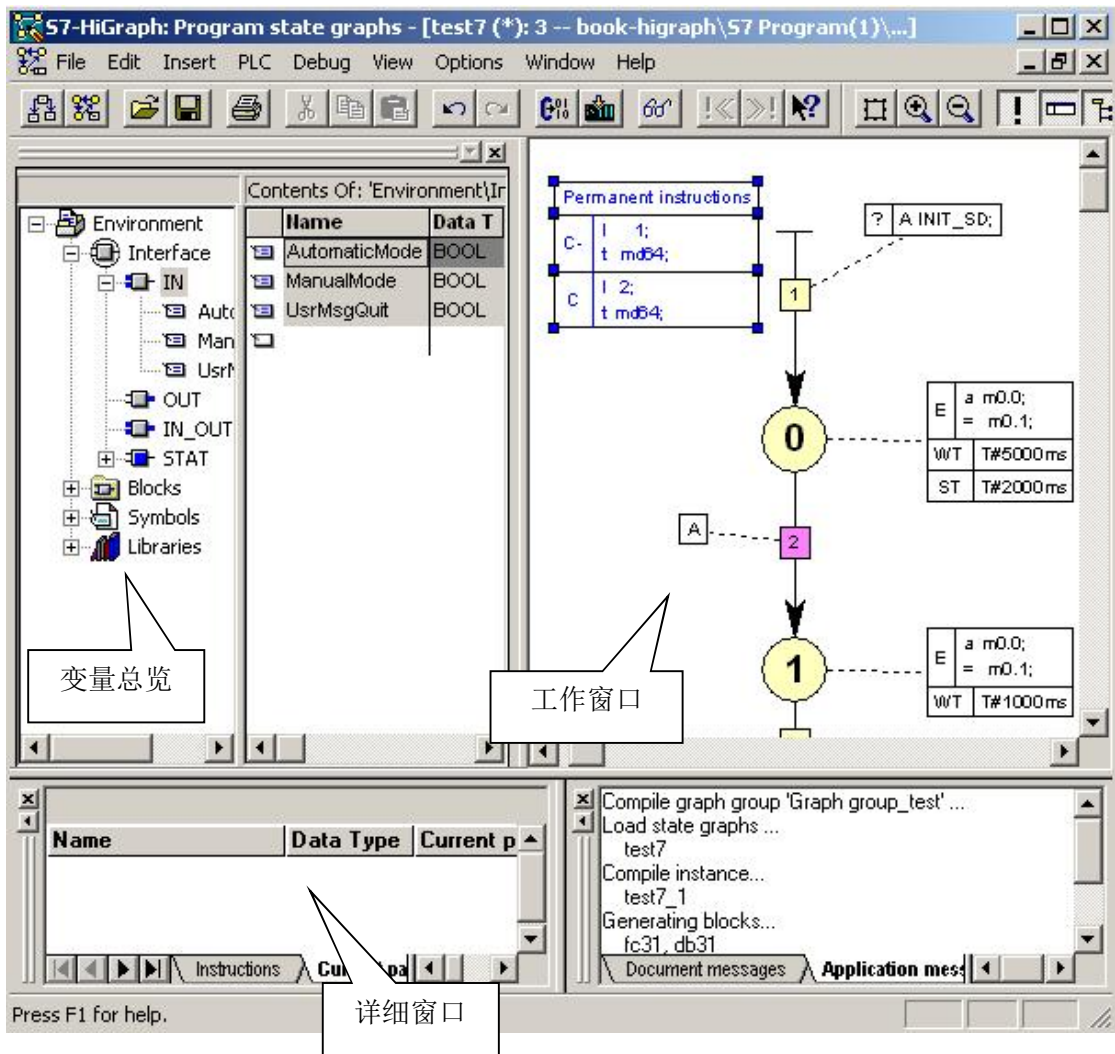


图 4-2: S7-HiGraph 用户界面

4.1.1. 工作窗口

用户在此可以编辑状态图以及图表组，主要的用户程序都在此区域完成。

4.1.2. 变量总览

在此区域可以编辑新生成的状态图的输入/输出等参数

注意：新生成的状态图包括了一些默认的参数，例如：自动模式/手动模式/初始化 (AutomaticMode/Manual Mode/INIT_SD)，对于这些参数系统有特定含义，建议用户不要修改。

另外，系统还列出了其它的一些编程元素：

- 符号表中的符号
- Block 文件夹中已经生成的块
- 函数库

4.1.2.1. 预定义变量：

预定义变量	含义	参数类型	数据类型	是否可被用户赋值
Manual Mode	此变量用来设置为 Manual 操作模式。如果此变量为 1，则仅仅有"Manual"属性的转换条件被检测，此变量不可与 AutomaticMode 同时为 1	IN	BOOL	可以
AutomaticMode	此变量用来设置为 Automatic 操作模式。如果此变量为 1，则仅仅有"Auto"属性的转换条件被检测，此变量不可与 Manual Mode 同时为 1	IN	BOOL	可以
INIT_SD	如果此变量为 1，状态图将被初始化	STAT	BOOL	可以
CurrentState	此变量保存当前状态的序号 *	STAT	WORD	
PreviousState	此变量保存前一个状态的序号 *	STAT	WORD	
StateChange	此变量在状态转换条件满足的周期中为 1，其它周期为 0 *	STAT	BOOL	
ST_Expired	监控时间超出 *	STAT	BOOL	
ST_ExpiredPrev	前一个状态的监控时间超出 *	STAT	BOOL	
ST_Stop	当此变量为 1 时，监控时间被中断	STAT	BOOL	可以
ST_CurrValue	剩余监控时间	STAT	DWORD	
ST_Valid	监控时间激活，此变量仅有内部意义	STAT	BOOL	
WT_Expired	等待时间到达	STAT	BOOL	
WT_Stop	当此变量为 1 时，等待时间被中断	STAT	BOOL	可以
WT_CurrValue	剩余等待时间	STAT	DWORD	
WT_Valid	等待时间激活，此变量仅有内部意义	STAT	BOOL	
DT_Expired	延迟时间超出。此变量只在第一个延迟时间被组态时被生成	STAT	BOOL	
UsrMsgSend	具备消息属性的状态激活。此变量在具备消息属性的状态激活时为 1。此变量仅于使用格式转换器的诊断有关	STAT	BOOL	

UsrMsgQuit	此变量用于错误/消息的确认。此变量仅于使用格式转换器的诊断有关	IN	BOOL	可以
UsrMsgStat	此变量仅有内部意义，仅于使用格式转换器的诊断有关	STAT	WORD	

表 4-1 预定义变量

* 在关于状态图周期执行的讲解中将详细解释这些变量何时被设置

4.1.2.2. 非激活变量

当生成一个新状态图后，如下变量默认是不激活的，用户无法在程序中引用这些变量：

- CurrentState
- PreviousState
- StateChange
- WT_Stop
- ST_Stop
- DT_Expired

为了在用户程序中引用这些变量，可以通过如下操作使能这些变量：

1. 在变量总览窗口中选择变量
2. 选择 Edit > Object Properties
3. 在下一个对话框中，选择 Attributes
4. 将 S7_active 设置为 true

4.1.3. 详细窗口

详细窗口可以显示当前的状态图中编译信息，指令，参数，以及变量等。

4.2. 状态图编程

4.2.1. 状态图元素

各种编程元素以图形格式出现在状态图中，其中包括如下元素：

- States (1)
- Transitions (2)
- Permanent instructions (3)
- states 或 transitions 的指令 (4)

下图为各种编程元素以图形格式出现在状态图中的例子，：

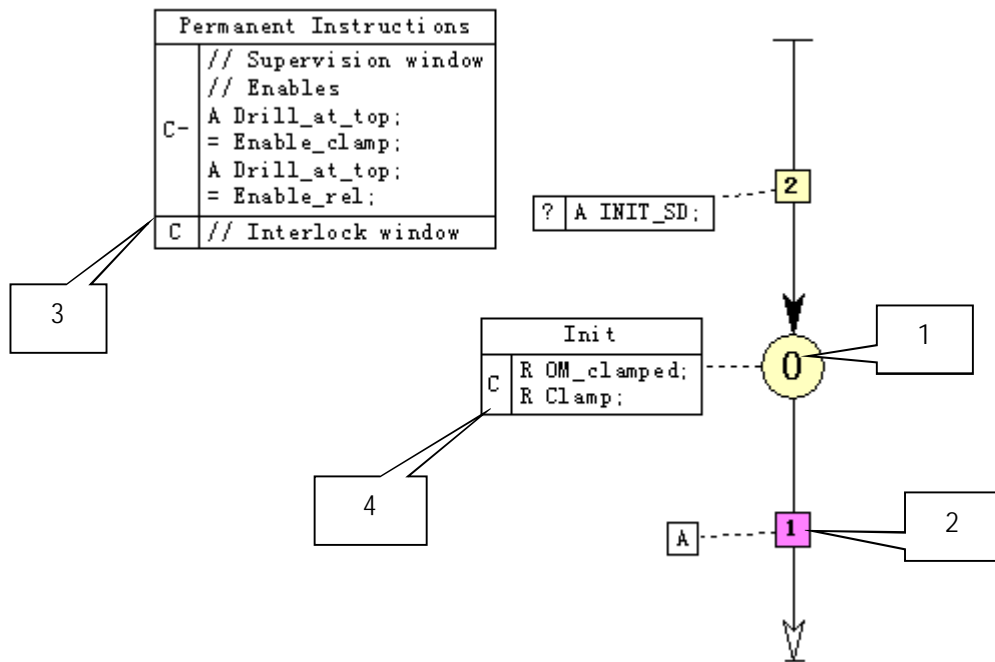


图 4-3: 状态图中的编程元素

4.2.2. 状态图和图表组结构的规则

状态图和图表组的结构必须符合如下规则：

- 一个状态图中最大元素数量：
 - 4090 状态
 - 4090 转换条件
- 一个图表组中最多可以容纳 255 个背景（每个状态图作为背景，可以被单次或多次在图表组中被引用）

4.2.3. State（状态）

4.2.3.1. 状态定义

State（状态）是 state graph（状态图）中的一个编程单元。它可以用来完成预定的控制功能。当程序执行时，状态图中仅有一个状态是激活的。

4.2.3.2. Initial state（初始状态）

初始状态是状态的一种特殊格式，它决定了上电后执行哪一个状态。每个状态图都需要一个初始状态。一个查询预定义变量 INIT_SD 的 Any transition，被认为是一个起始转换条件，它被用来初始化状态图，下图为 Any transition 指向一个初始状态：

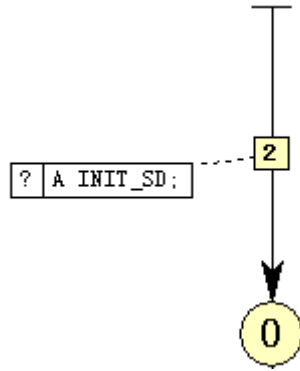


图 4-4: 初始状态

4.2.3.3. 添加 State（状态）

在编程界面的工具栏中，点击下面的图标，通过鼠标在工作区的操作，就可以添加状态。

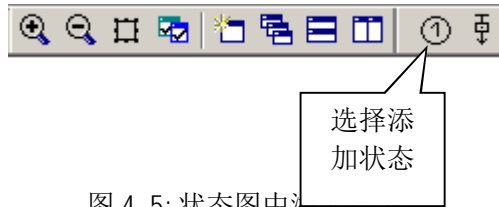


图 4-5: 状态图中添加状态。

4.2.3.4. State（状态）的属性

右键点击状态后，选择 Object Properties, 可以设置状态的属性，

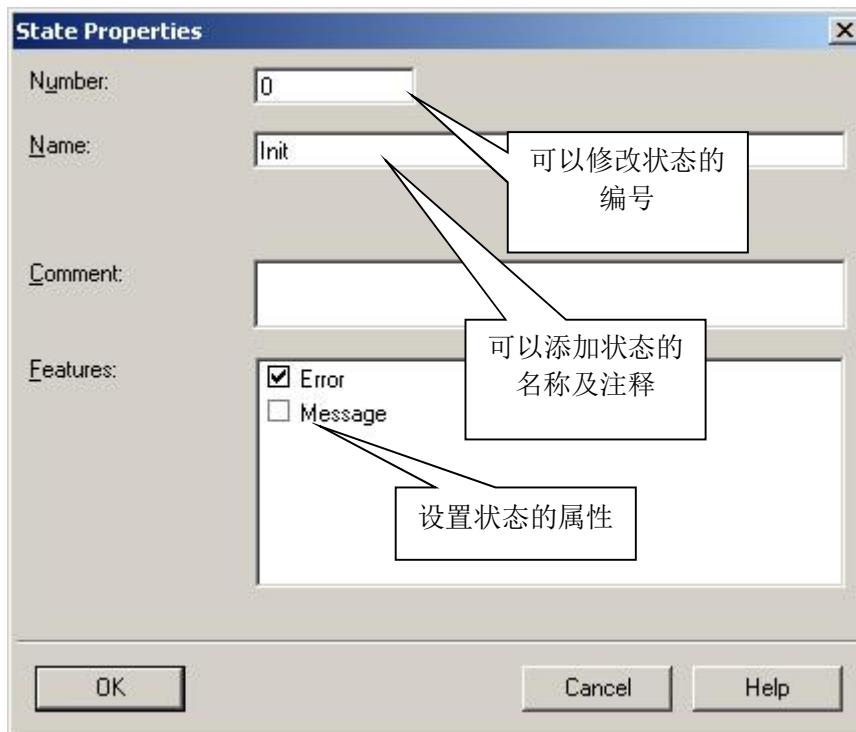


图 4-6: 状态的属性

- Error: 流程执行到此状态时，将输出一个报警消息(alarm message)给诊断程序, 在默认情况下，具备此属性的状态将以红色显示，并添加 E 的标示

- Message: 流程执行到此状态时, 将输出一个状态消息 (status message) 给诊断程序, 在默认情况下, 具备此属性的状态将以亮黄色显示, 并添加 ME 的标示
- 一个状态只能具备 Error 或 Message 中的一项属性, 或都不具备。

State (状态) 的属性的标识见下图:

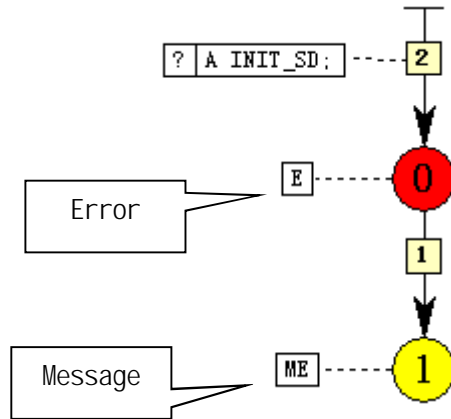


图 4-7: 状态的属性标识

4.2.3.5. 添加 State (状态) 动作

双击状态后, 在用户界面的详细窗口将显示当前状态的指令列表, 默认为没有指令, 右键选择某项, 可以添加一个或多个 action(动作) 或选择是否使用

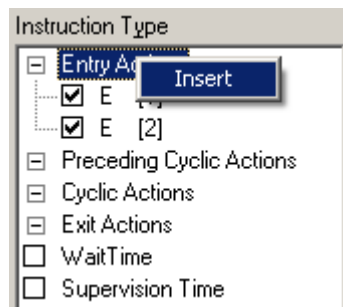


图 4-8: 添加动作

4.2.4. Transition (转换)

4.2.4.1. Transition (转换) 定义

一个转换可以控制两个状态图之间的转换。各个转换将被赋予一些转换条件, 当这些条件满足时, 将触发一个转换状态。

4.2.4.2. Transition (转换) 的优先级

多个转换可以指向同一个状态, 这种情况下可以为转换设置优先级。当多个转换条件都满足时, 优先级最高的转换将被执行。1 为最高的优先级。

4.2.4.3. Transition (转换) 类型

转换可以分为 3 种类型, 参见下表:

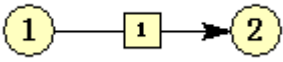
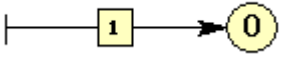
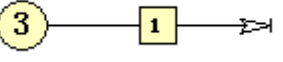
转换类型	功能描述	视图
Normal transition	normal transition 把系统流程从一个状态引导至下一个状态	
Any transition	Any transition 把系统流程从所有其它状态引至此转换条件所指向的目标状态。转换条件被连续处理，而不论当前状态图的状态。它们可以被用于一直需要监控的高优先级的限制条件（如急停）。如果一个状态图中有多个 Any transition，它们会被分配各自的优先级。Any transition 的优先级总是高于 Normal transition 的优先级。一个查询预定义变量 INIT_SD 的 Any transition，被认为是一个起始转换条件，它被用来初始化状态图	
Return transition	Return transition 把系统流程从当前状态引导至前一个激活的状态 Return transitions 优先级低于 normal transitions	

表 4-2: 转换类型

4.2.4.4. 添加 Transition（转换）

在编程界面的工具栏中，点击下面的图标，通过鼠标在工作区的操作，就可以添加转换。

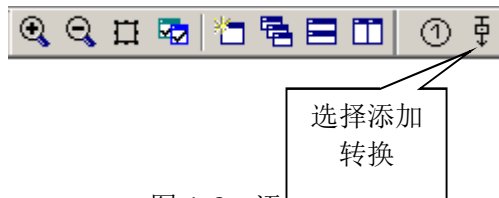


图 4-9: 添加转换

4.2.4.5. 转换的属性

右键点击转换后，选择 Object Properties, 可以设置转换的属性：

- 优先级：1 至 255
- 名称：可以根据需要修改
- 源状态：不可修改，与实际连接关系有关
- 目标状态：不可修改，与实际连接关系有关
- Waiting 当执行转换时，在 source state 中设置的等待时间将有效
- Auto 此转换只在自动模式下执行
- Manual 此转换只在手动模式下执行
- Error 标示此转换为错误转换

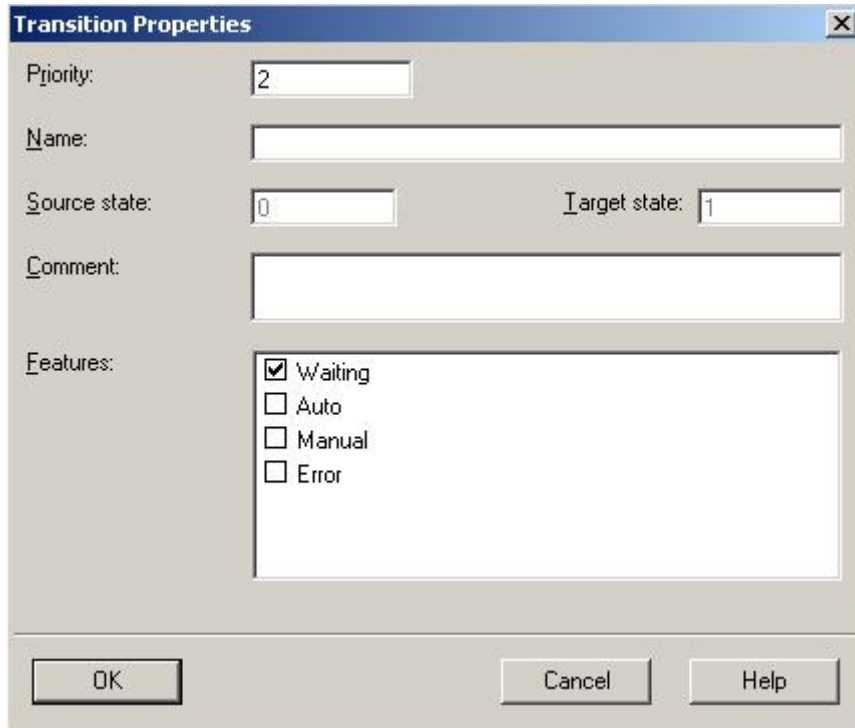


图 4-10: 转换的属性

当设置完毕后，转换的在编程界面下的标识会有所改变：

- Auto 在默认情况下，具备此属性的状态将以粉色显示，并添加 A 的标示
- Manual 在默认情况下，具备此属性的状态将以亮青色显示，并添加 MA 的标示
- 一个状态只能具备 Auto 或 Manual 中的一项属性，或都不具备。

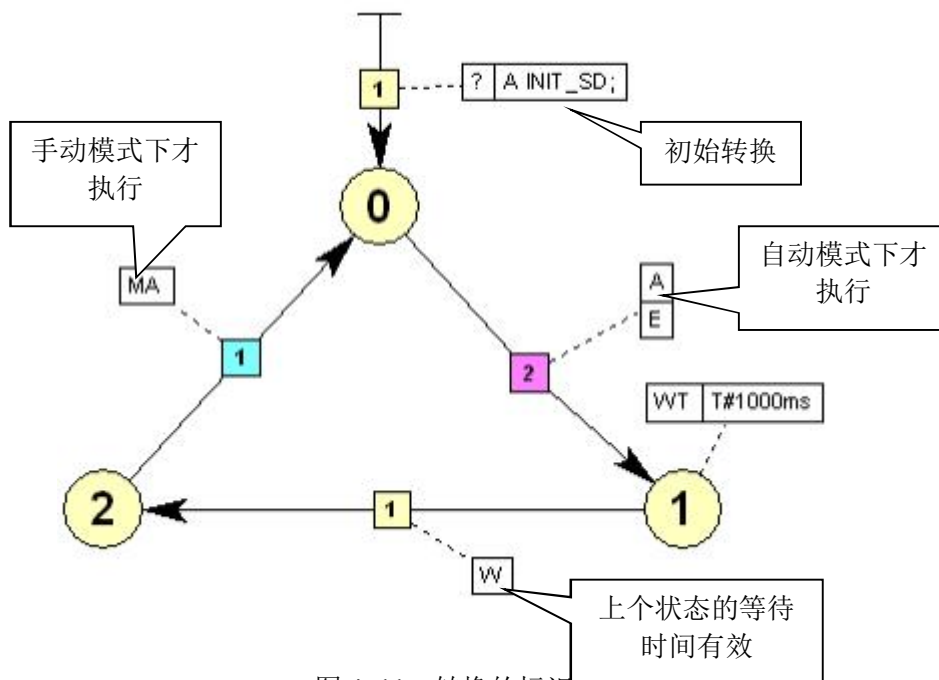


图 4-11: 转换的标识

4.2.5. Permanent instructions

4.2.5.1. Permanent instructions 定义

Permanent instructions 字面含义为永久指令。在每个状态图的执行周期，无论当前状态图执行到了哪一个状态，这些指令将被执行一次。

4.2.5.2. Permanent instructions 类型

指令类型	功能描述	视图
Preceding cyclic actions	在每个周期开始时被执行	C-
Cyclic actions	在每个周期结束时被执行	C

表 4-3: 指令类型

4.2.5.3. 添加 Permanent instructions

在每个新建的状态图中，系统都自动添加如下表格，双击此表格，在详细窗口中添加用户指令即可。

Permanent instructions	
C-	
C	

图 4-12: 添加 Permanent instructions

4.2.5.4. Permanent instructions 用途

Permanent instructions 可以有特殊用途，例如：

- 连续计算在很多点（状态或转换）都关心的过程变量
- 执行系统在任何情况下都必须相应的控制程序（例如安全门控制）

4.3. 指令编程

4.3.1. 程序中的指令

4.3.1.1. 指令定义

指令即过程控制命令，可以调用功能块，控制输入/输出等。

指令可以被指定给状态，转换，当某个状态或转换激活时，这些指令被执行。

Permanent instructions 中的指令在每个状态图的执行周期，无论当前状态图执行到了哪一个状态，这些指令将被执行一次。

4.3.1.2. 指令表

在状态图中，指令都是以指令表的形式存在的。

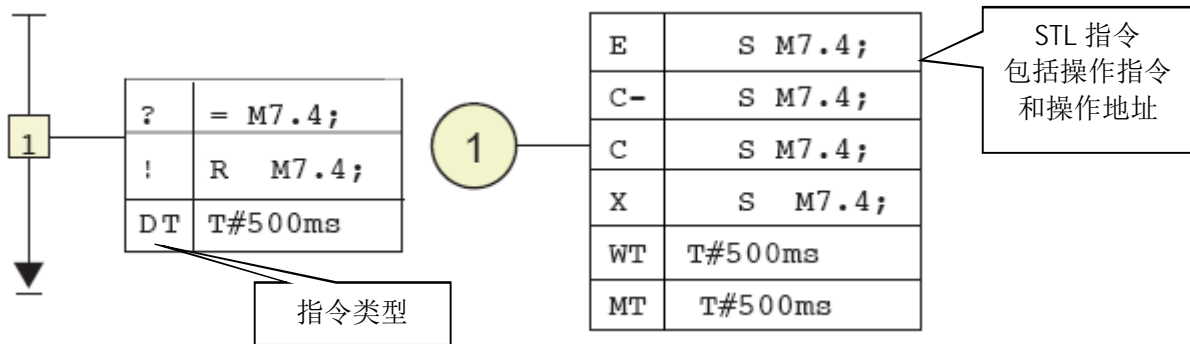


图 4-13: 指令表视图

4.3.2. 指令类型

在状态图中，各种指令可以完成如下功能：

指令类型	标识	功能	使用范围
Entry actions/进入动作	E	当进入一个状态时执行的动作	States
Preceding cyclic actions/周期前动作	C-	在一个周期中，并且在转换条件检查前，需要执行的动作	States/Permanent instructions
Cyclic actions/周期动作	C	在一个状态中需要周期执行的动作	States/Permanent instructions
Exit actions/退出动作	X	当退出一个状态时，执行动作	States
Waiting times/等待时间	WT	定义控制系统在一个状态中至少要停留的时间	States
Monitoring times/监控时间	MT	定义在一个状态中停留的时间是否需要监视	States
Delay times/延迟时间	DT	定义条件转换时的延迟时间	Transitions
Conditions/条件	?	这些指令描述了当状态转换时，必须满足的条件	Transitions
Transition actions/转换动作	!	当转换触发时，这些指令将被执行一次	Transitions

表 4-4: 指令类型

4.3.3. 输入 STL 指令规则

注意：输入 STL 指令规则对于 S7-HiGraph 程序编写非常重要，每个使用者必须牢记并深刻理解，否则将影响程序编制，并有可能造成不可预料的后果（如人身伤害，设备损坏等）

名称	规则
语法	基本语法规则遵守 STL 编程语法规则
行	每条指令必须写到独立的行中
指令块	同一种类型的指令（例如状态进入事件指令）可以分成多个块
分号	指令的每行必须以分号结束
大小写	不区分符号，地址的大小写
地址	为了能够多次使用状态图（作为通用功能使用），应当在状态

	图中尽量通过定义变量来代替绝对地址的使用。当在图表组中以背景方式插入状态图后，再为这些状态图分配实参
RLO	每个指令表都以 RLO=1 开始
嵌套堆栈	编译系统不检查嵌套深度，最多为 7 层
间接寻址	不支持间接寻址

表 4-5: 输入 STL 指令规则

解释 1: RLO 每个指令表都以 RLO=1 开始。在下图中，包括 E 类型指令，C 类型指令被分为了 2 个指令表。如果不考虑指令表的作用，仅考虑 STL 语句，M0.3 的状态应当取决于 M0.0, M0.1, M0.2, 即:

```
A M0.0
A M0.1
A M0.2
= M0.3
```

但是在状态图指令中，RLO 每个指令表都以 RLO=1 开始，如下图中的 M0.3 仅仅与 M0.2 的状态有关，图中的指令 A M0.0 及指令 A M0.1 实际没有什么意义。

不推荐使用绝对地址格式

STL 指令格式，分行，以分号结束，不区分大小写，可分为多块
句符操作比句和操作地址

依然为 C 指令，但被划分为了另一块

Address	Display format	Status value
M 0.0	BOOL	false
M 0.1	BOOL	false
M 0.2	BOOL	true
M 0.3	BOOL	true

图 4-14: RLO 的使用规则

解释 2: 不支持间接寻址。状态图中不支持类似 L DBW [MDO] 的间接寻址格式，如果需要使用间接寻址功能，可以将此类语句编写为一个 FC，再使用 CALL 指令调用此 FC。

状态图中调用 FC, FB 的格式如下，注意分号的位置与参数赋值:

```
CALL FC10 (
  param1 :=I 0.0,
  param2 :=I 0.1);
CALL FB10, DB100 (
  para1 :=I 0.0, para2 :=I 0.1);
```

注意: 在状态图中建议尽量不使用 CALL 指令调用 FC 或 FB，因为这样降低了状态图的通用性。

4.3.4. 绝对地址与符号编程

在状态图中建议尽量不使用绝对地址。为了能够多次使用状态图（作为通用功能使用），应当在状态图中尽量通过定义变量来代替绝对地址的使用。当在图表组中以背景方式插入状态图后，再为这些状态图分配实参，这样状态图可以作为通用背景来使用。

4.4. 等待，监控，延迟时间编程

4.4.1. 等待时间编程

定义：在转换条件检查之前，系统在一个状态中至少停留的时间。此时间可以是一个时间常数，也可以是个变量。

等待时间编程可以分为两步：定义等待时间和在转换条件中使能等待时间。

步骤 1:

选择一个状态，在此状态的详细视图中选择等待时间，如下图

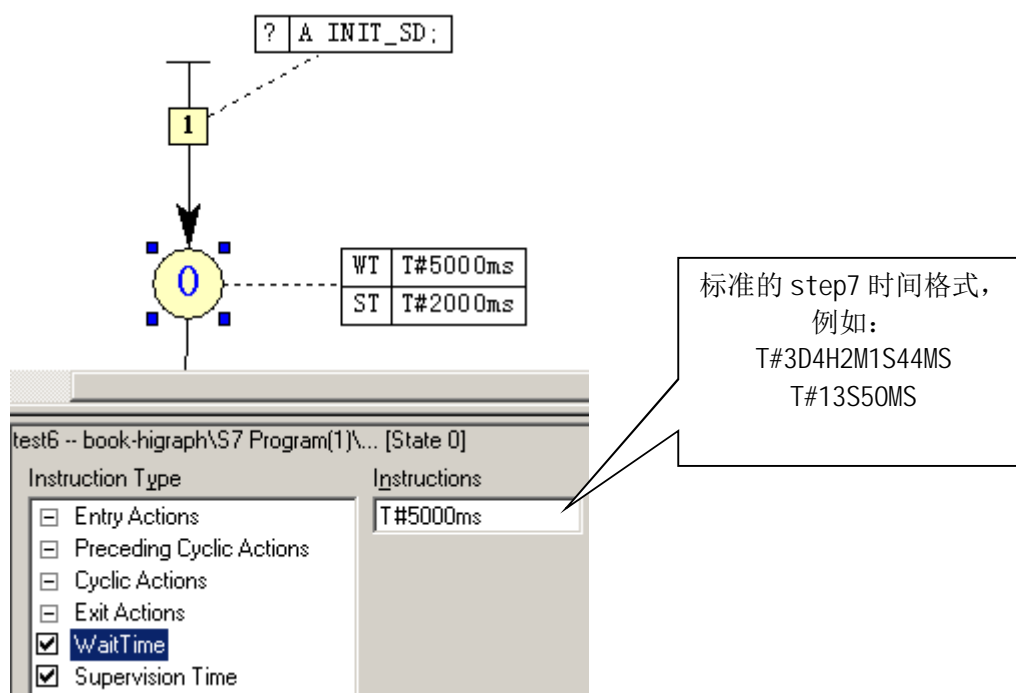


图 4-15: 等待时间编程

步骤 2:

选择与此状态连接的下一个转换，在其属性中使能“Waiting”选项。

4.4.2. 监控时间编程

定义：监控时间可以用来监控系统在状态中停留的时间。此时间可以是一个时间常数，也可以是个变量。

步骤：选择一个状态，在此状态的详细视图中定义监控时间，如下图：

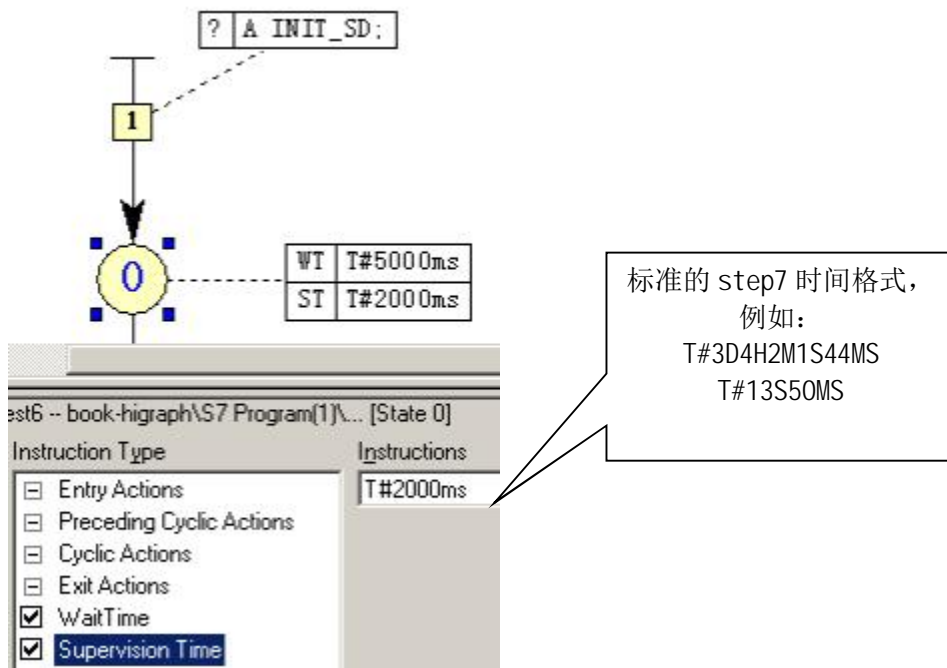


图 4-16: 监控时间编程

当监控时间超出时:

- 默认变量“ ST_Expired” 被置位
- 一个错误消息输出到诊断程序

4.4.3. 延迟时间编程

定义: 延迟时间可以用来防止 PLC 对短脉冲的作出响应。一个有延迟时间的转换在转换条件满足时, 并不进行切换, 只有转换条件满足的时间达到了设定的延迟时间, 转换才进行。此时间可以是一个时间常数, 也可以是个变量。

步骤: 选择一个转换, 在此转换的详细视图中可以定义延迟时间。

注意: 此功能仅仅 S7-Hi Graph V5.3 以上的版本支持。



图 4-17: 延迟时间编程

4.5. 操作模式编程

4.5.1. S7-HiGraph 中系统支持两种操作模式:

- Auto mode 此模式的状态由系统默认参数 AutomaticMode 为 1 时确认
- Manual mode 此模式的状态由系统默认参数 Manual Mode 为 1 时确认

4.5.2. 对于转换条件切换的影响：

1. 如果一个转换具备 Auto 属性，则此转换只在以下情况进行转换：
 - 转换条件 (transition condition) 满足
 - AutomaticMode 数值为 1
2. 如果一个转换具备 Manual 属性，则此转换只在以下情况进行转换：
 - 转换条件 (transition condition) 满足
 - ManualMode 数值为 1
3. 如果一个转换不具备 Manual 或 Auto 属性，则此转换只根据转换条件来决定是否进行转换。

4.6. 图表组编程

4.6.1. 图表组：

定义：图表组拥有一定数量的状态图，这些状态图可以被编译，保存和下载。

功能：状态图可以描述机器设备的独立单元，为了描述一个完整的机器或平台，可以把一定数量的状态图在图表组中组合起来。

4.6.2. 图表组编程步骤：

步骤 1：生成图表组。右键点击项目的 Sources 目录，Insert New Object->Graph group, 输入文件名

步骤 2：选择菜单 Insert->Instance, 这时可以选择当前项目中已经编写完毕的状态图。并且可以多次使用同一个状态图，如下图：状态图 Molding 在图表组中作为背景被应用了 3 次。

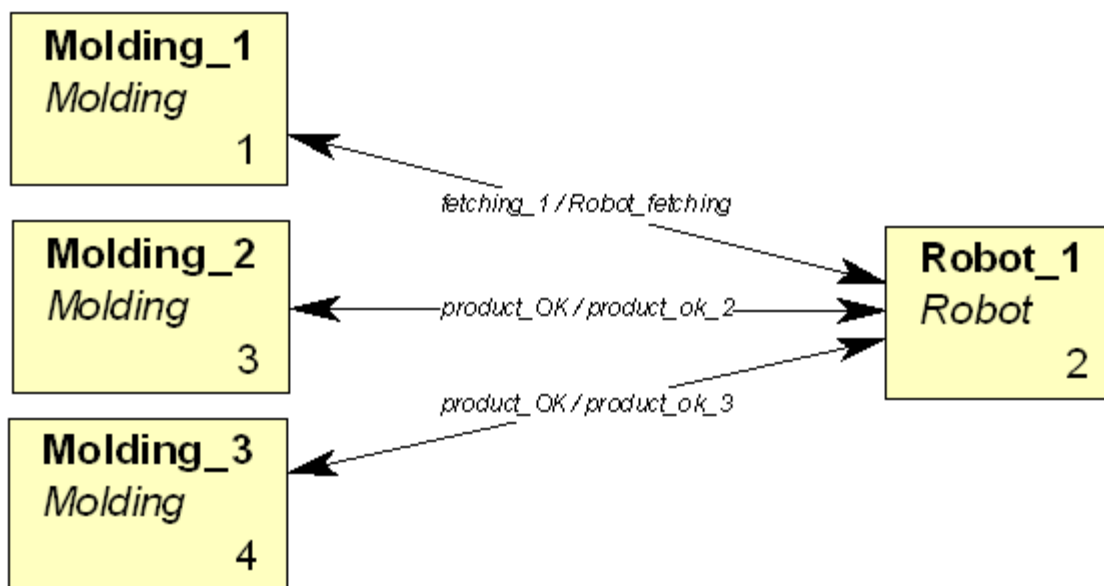


图 4-18: 状态图作为背景被使用

步骤 3: 选择菜单 Edit->Run Sequence, 可以通过上下箭头来调整各个状态图在图表组中的运行顺序

步骤 4: 分配参数。在图表组中, 选择一个背景, 在详细视图的窗口中为其分配参数, 如下图。

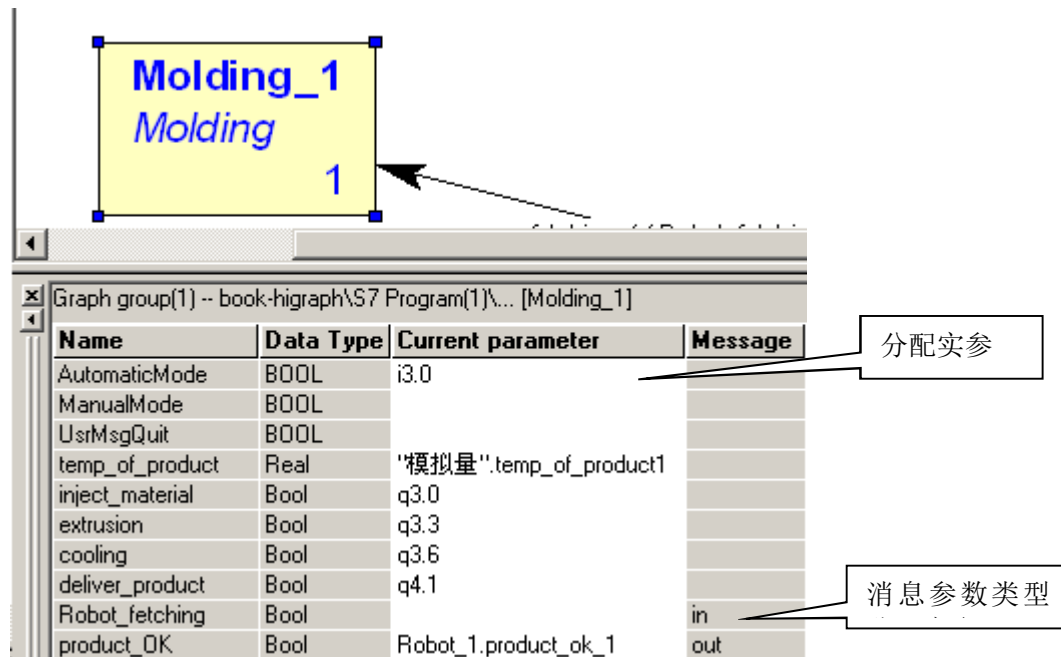


图 4-19: 为状态图背景分配参数

4.7. 状态图消息交换编程

4.7.1. 消息的基本信息

定义: 消息 (Message) 用来在状态图之间用来通信, 一个状态图设置另外一个状态图可以接受的信号, 这样在运行时, 状态图之间可以互相影响。

消息分类: 消息 (Message) 可以分为两类

1. Internal message: 在一个图表组中的多个状态图之间进行通信, 此通信通过 S7-Hi Graph 数据块中的位地址来进行 (消息的数据类型必须为 bit)。
2. External message: 不同的图表组之间或者 S7-Hi Graph 与其它 STEP7 程序之间通信。这种通信也使用位地址, 编程者须自行定义。

消息显示:

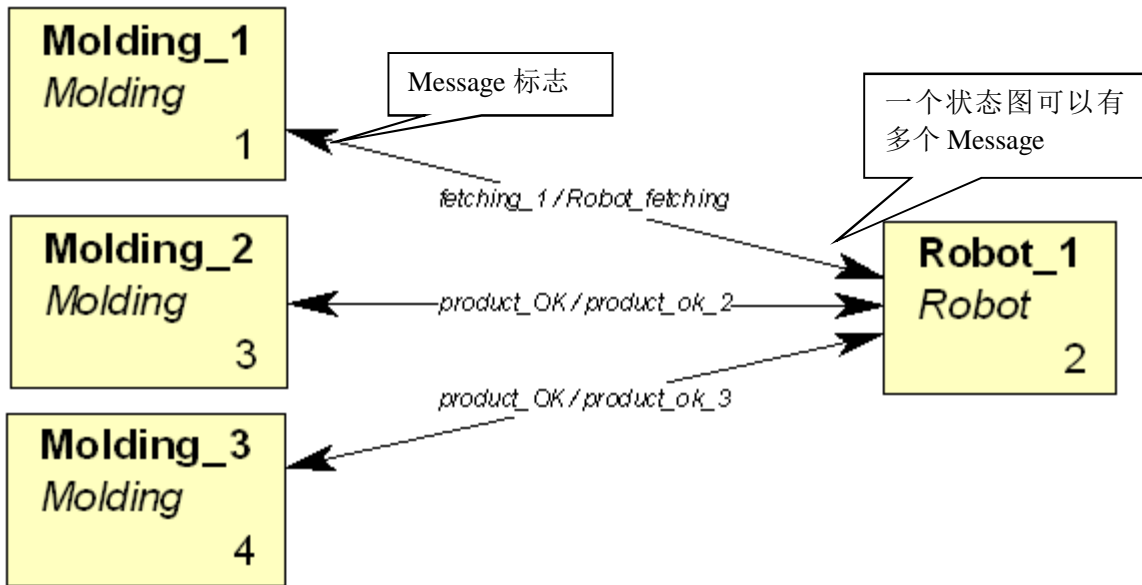


图 4-20: 状态图之间的消息显示

4.7.2. 声明消息变量

发送消息中的 out 变量将把信息发送给接收消息状态图中的 in 变量。

状态图 A ----OUT 消息变量----信息----IN 消息变量----> 状态图 B

消息变量定义步骤:

1. 打开发送消息的状态图
2. 在 IN_OUT 变量表中定义一个 BOOL 变量
3. 在消息类型中选择“ out”
4. 打开希望接收消息的状态图
5. 在 IN_OUT 变量表中定义一个 BOOL 变量
6. 在消息类型中选择“ in”

注意: 这两个状态图中的变量名称可以不同

4.7.3. 连接消息

要求: 为了连接消息变量, 首先要把需要发送及接收消息的状态图以背景方式, 放置到图表组中。

连接内部消息 Internal message

1. 选中需要发送消息的背景 (状态图)
2. 在“ Current parameters” 栏中, 选择发送消息变量
3. 输入接收变量名称, 格式为: 接收状态图名称.接收变量名。如: Robot_1.product_ok_1, 参见下图
4. 对于接收变量, 不需要再次组态, 因为其连接关系在发送变量组态中已经组态完毕。

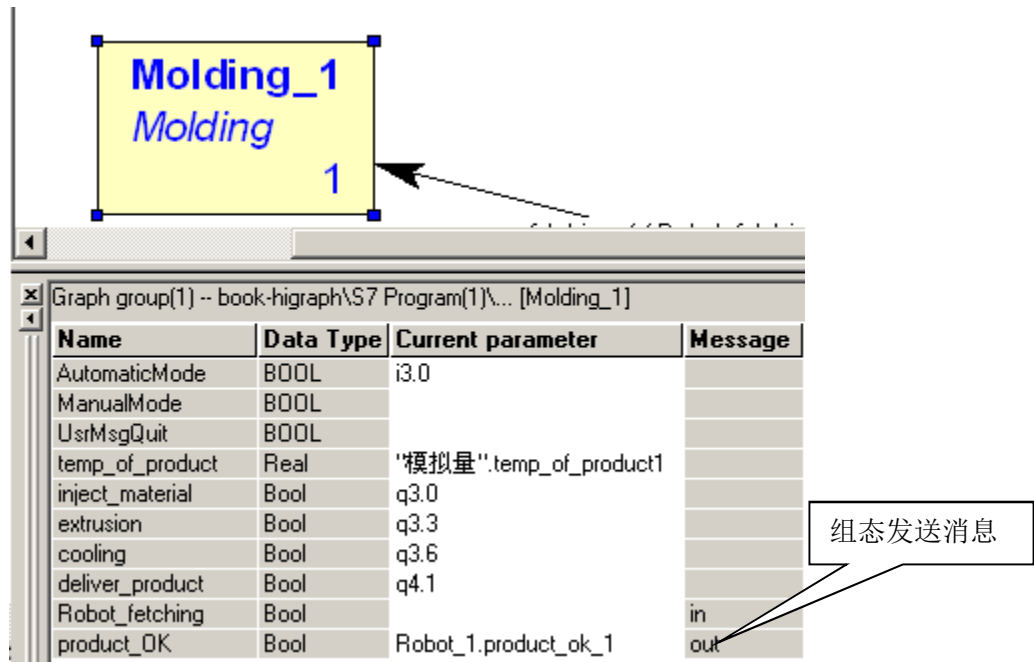


图 4-21: 连接消息变量

连接外部消息 External message:

1. 选中需要发送消息的背景（状态图）
2. 在“ Current parameters” 栏中，选择发送消息变量
3. 输入接收变量名称，如：DB10.DBX5.1,组态即完毕

4.8. 程序编译

4.8.1. 编译选项

在程序编写完毕后，用户需要编译后，才可以生成执行代码及数据块。编译选项中的设置对系统运行结果有着非常重要的影响，用户一定要给予足够的重视。

选择菜单：Options -> Settings for graph groups/state graphs -> Compile, 打开如下图的用户界面。

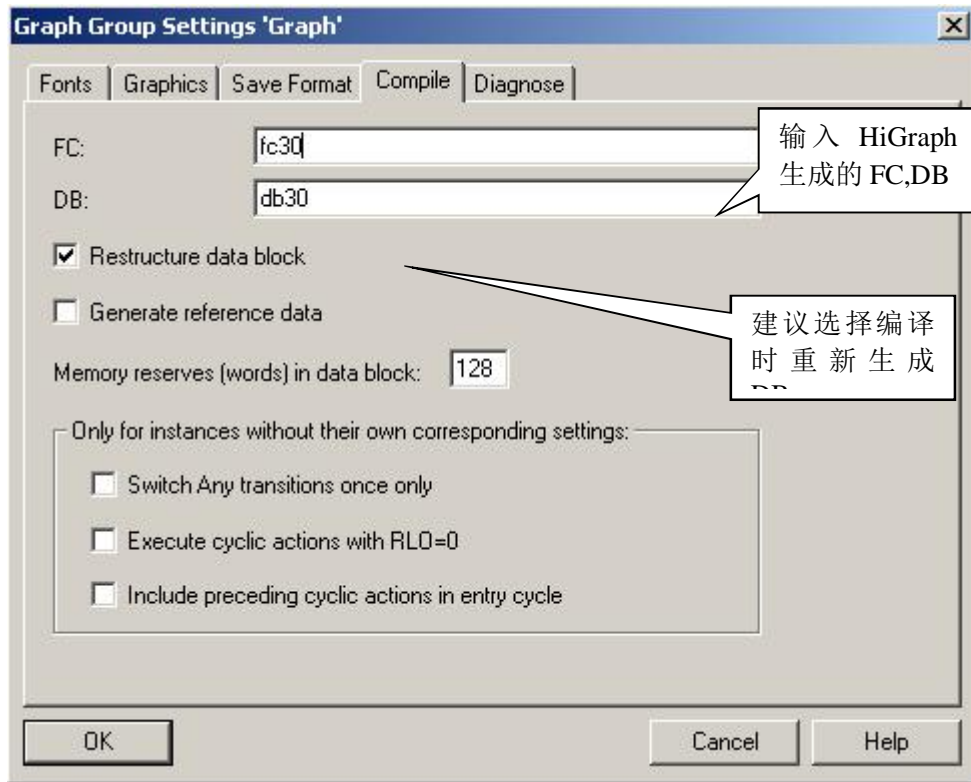


图 4-22: 编译选项设置

4.8.2. Memory reserve (words) in data block:

指定在数据块中为添加状态，消息，用户自定义变量而预留的内存大小。如果希望在线修改下载程序，此设置很重要。

注意：此选项仅在 Restructure data block 选项被选择时，才有效。

额外内存要求如下：

每个 state graph	12 words
每个 message	1 bit

在 STAT 变量表中声明的变量：

数据类型：BOOLEAN	1 bit
数据类型：Word/INT/DATE/S5TIME	2 bytes
数据类型：Dword/DINT/REAL/TIME/TOD	4 bytes

4.8.3. 图表组设置

注意：下面这些设置仅仅在此情况下有效：

图表组中引用的状态图背景没有指定的自身的设置，对于这些状态图背景，将使用图表组统一的设置。

Switch Any transitions only once

选择此项后，当控制系统已经在 Any transition 指向的状态中时，则 Any transition 不再发生

切换。

Cyclic actions with RLO 0

选择此项后，当状态离开时，系统将 RLO 清 0，并再次执行一次 cyclic actions 中的指令。此功能可以用于将状态图中所有置 1 的信号清 0。（此内容将在常见问题中详细解答）

Preceding cyclic actions also in the entry cycle

选择此项后，在进入周期时，(C-)中的指令执行一次。（此内容将在常见问题中详细解答）

4.9. 程序调用/下载/调试

4.9.1. 程序调用

S7-Hi Graph 中生成的 FC, DB, 可以在其它程序中调用，例如：OB1。

```
CALL FC    31      //调用 S7-Hi Graph 中生成的 FC
INIT_SD:=M100.1  //程序的初始化控制
```

4.9.2. 程序下载

用户程序可能包括如下部分：

- S7-Hi Graph FC
- S7-Hi Graph DB
- S7-Hi Graph 诊断 DB
- (OB, FB or FC)
- 程序中需要使用的其它块

如果选择了诊断功能，可能还包括如下部分：

- Hi GraphErrEmitterFB (FB20)，格式转换诊断
- Hi GraphMsgEmitterFC (FC101)，格式转换诊断
- Hi GraphUniversalEmitterFC (FC102)，标准诊断
- Alarm_S (SFC18) and Alarm_SQ (SFC 17)，标准诊断

将这些在 Block 文件夹中的块下载到 PLC 中。

4.9.3. 程序调试

在 S7-Hi Graph 程序下载完毕，并且被调用后，可以在图表组的视图下，选择菜单 Debug -> Monitor 来监视程序的执行情况。也可以双击图表组中的背景状态图，监控状态图的状态。

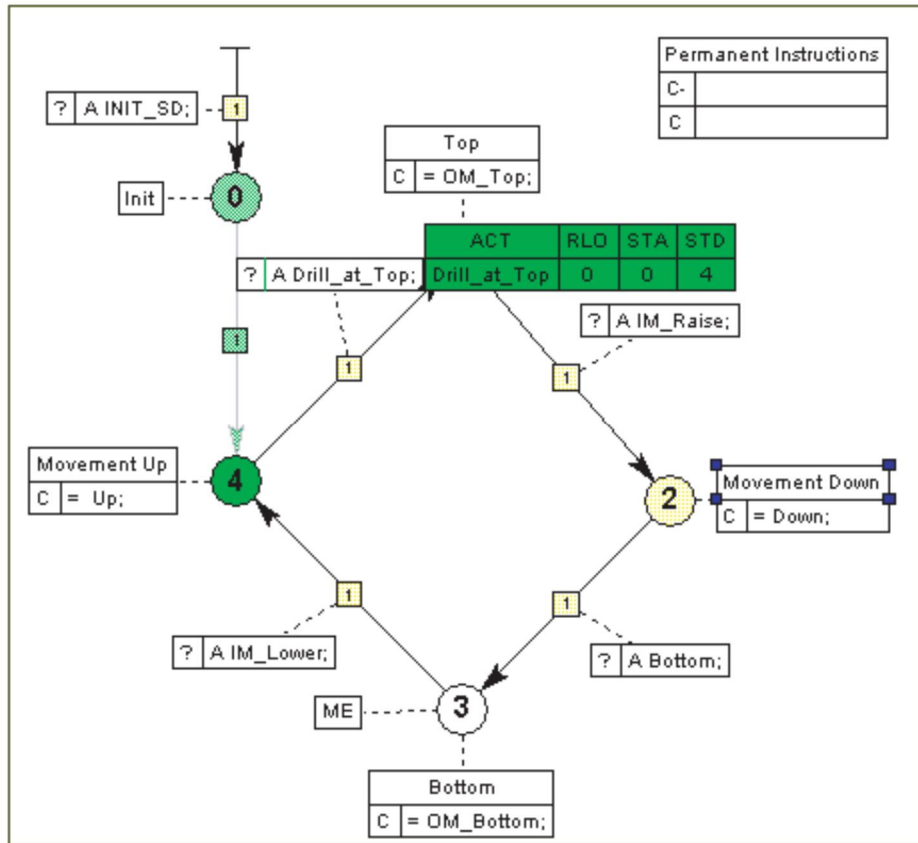


图 4-23: 监控视图

RLD: 逻辑运行结果

STA: 状态位

STD: ACCU1 中的内容

在图表组/状态图的视图下，在菜单 Debug -> Select variable，可以选择关心的变量进行监控。

5. S7-HiGraph 应用于虚拟工程

5.1. 虚拟工程工艺要求

1-3 号成型机各自生产产品，当某台成型机产品就绪时，机械手将产品取走，并放置在传送带上。

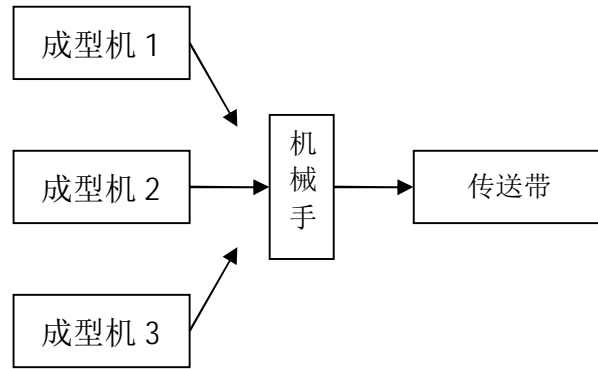


图 5-1: 产品成型系统构成

成型机工艺流程:

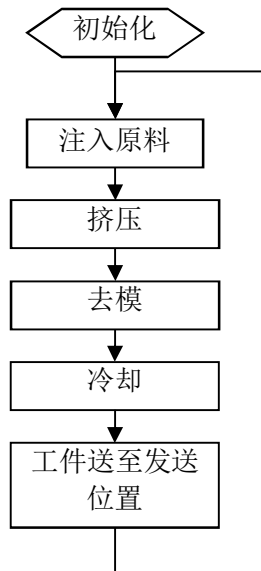


图 5-2: 成型机工艺流程

机械手工艺流程:

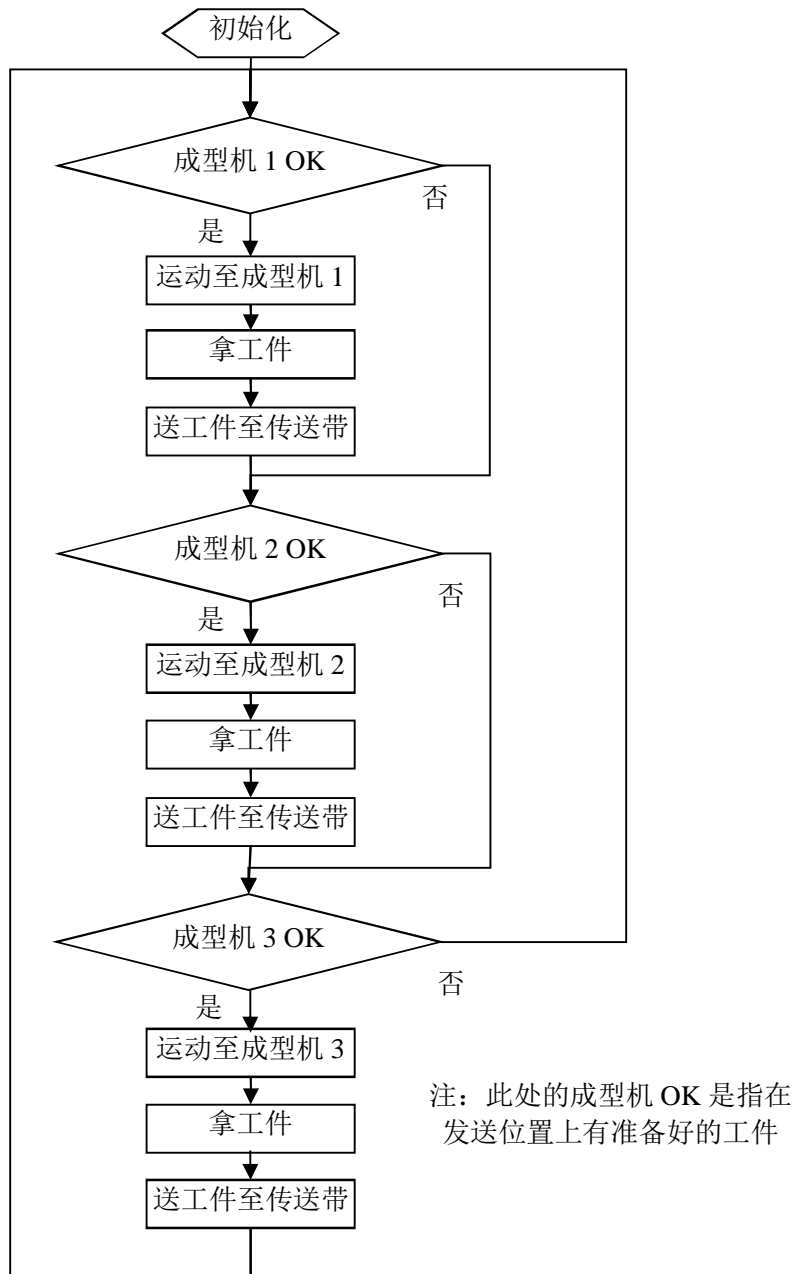


图 5-3: 机械手工艺流程

5.2. S7-Hi Graph 简单示例

在下面的例子中，将编写一个 S7-Hi Graph 用于成型工艺控制的程序：

- ✧ 控制每个成型机
- ✧ 控制机械手
- ✧ 协调成型机及机械手工作

- 1) 右键点击 Sources -> Insert New Object-> State graph, 名字为 mol di ng
- 2) 右键点击 Sources -> Insert New Object-> State graph, 名字为 robot
- 3) 右键点击 Sources -> Insert New Object-> Graph group
- 4) 双击 mol di ng 打开，编辑成型机程序，定义输入参数

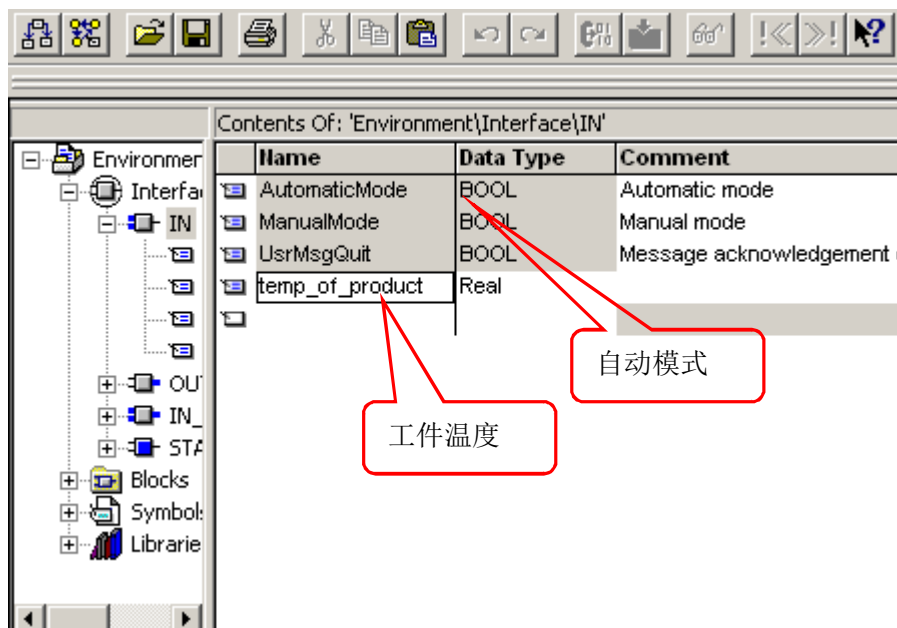


图 5-4: 成型机输入参数

5) 编辑成型机输出参数

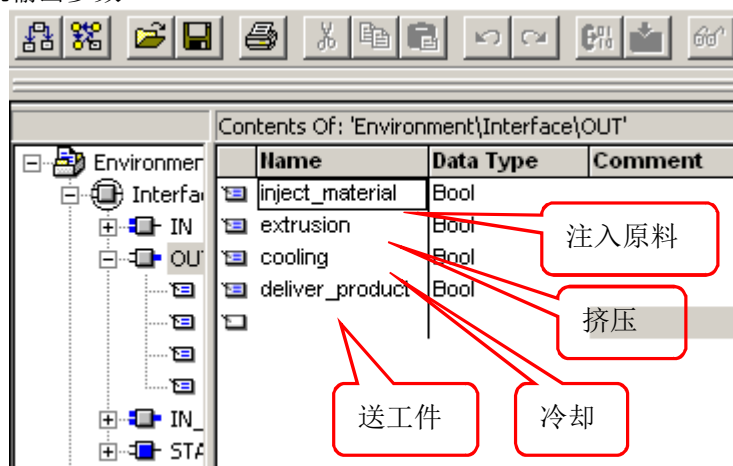


图 5-5: 成型机输出参数

6) 编辑成型机消息

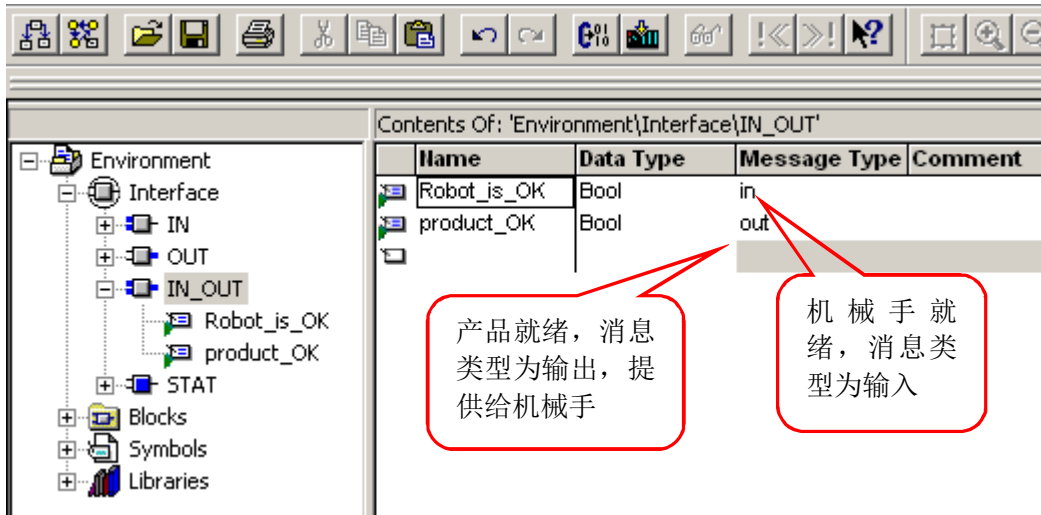


图 5-6: 编辑成型机消息

7) 编辑成型机静态变量参数

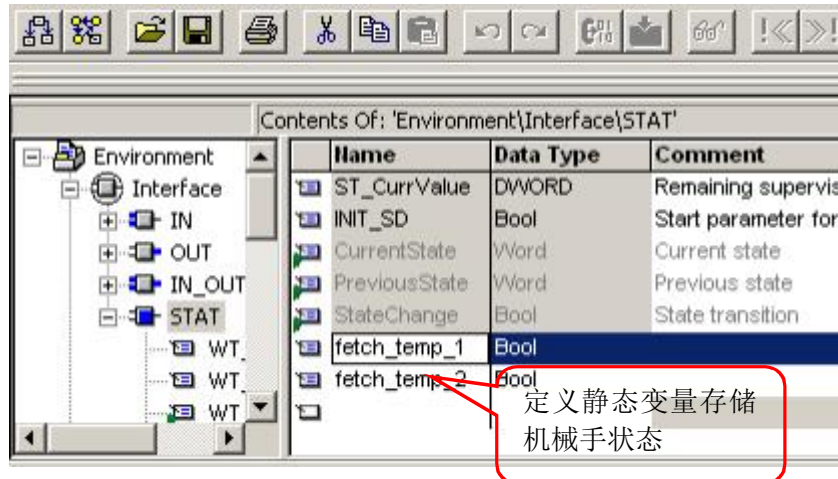


图 5-7: 成型机静态变量参数

8) 添加状态及条件

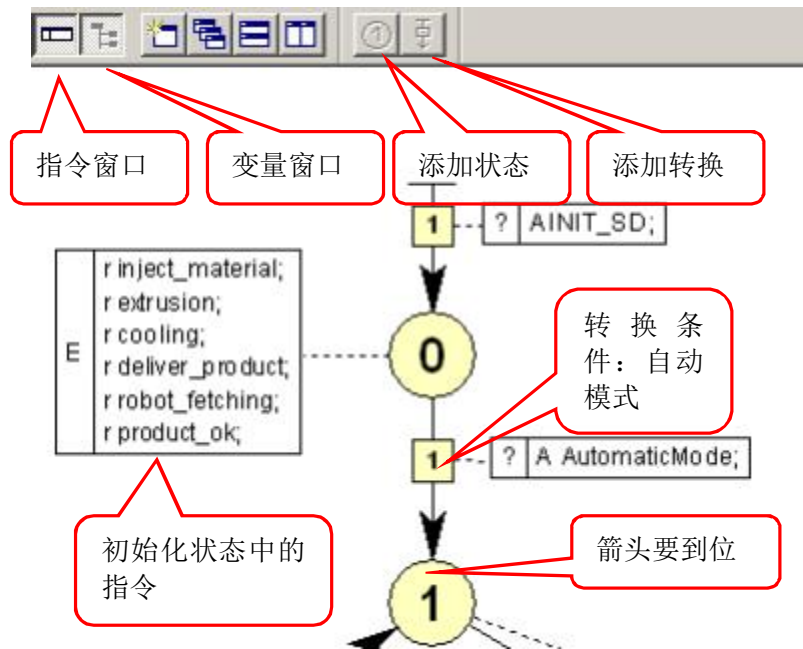


图 5-8: 添加状态及条件

9) 双击状态，添加各种指令

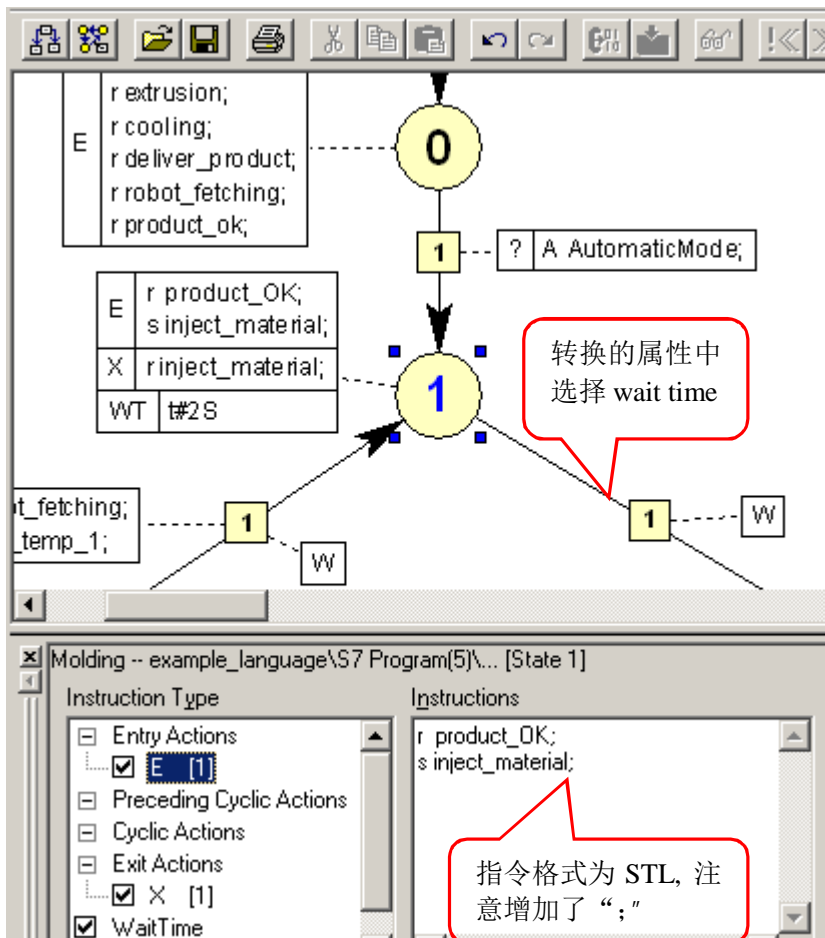


图 5-9: 添加各种状态下的指令

10) 成型机程序概览

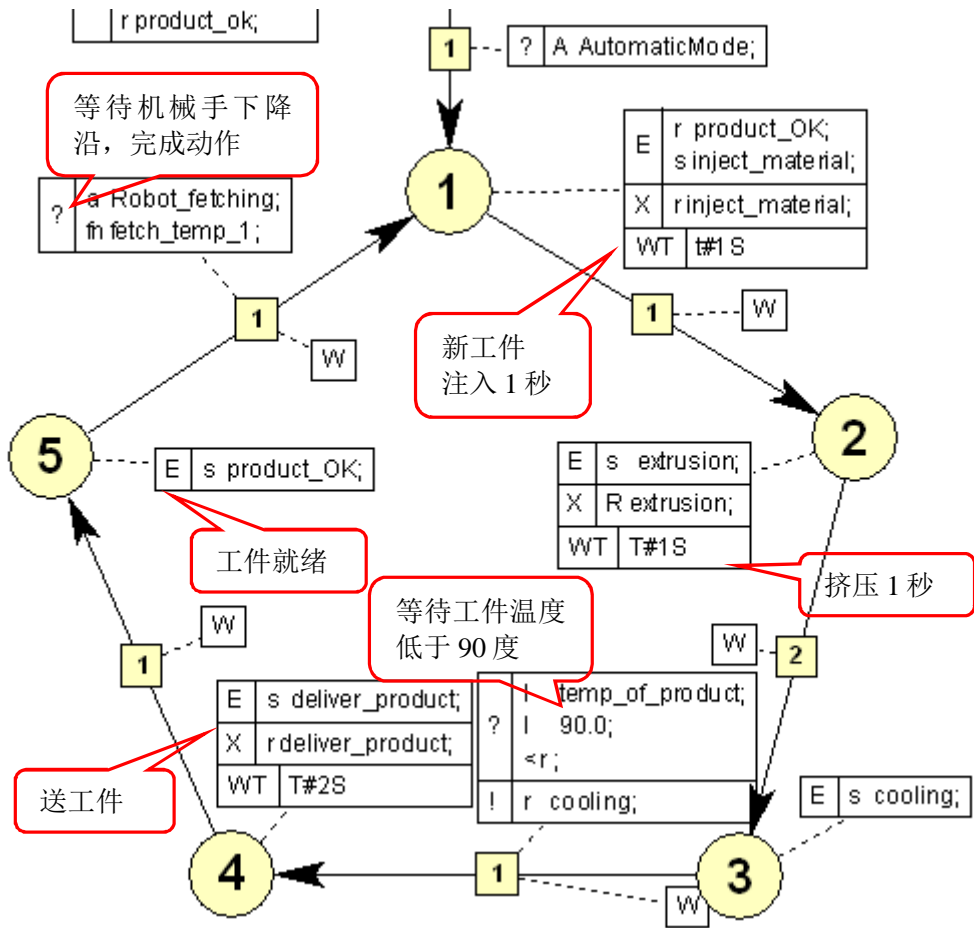


图 5-10: 成型机程序概览

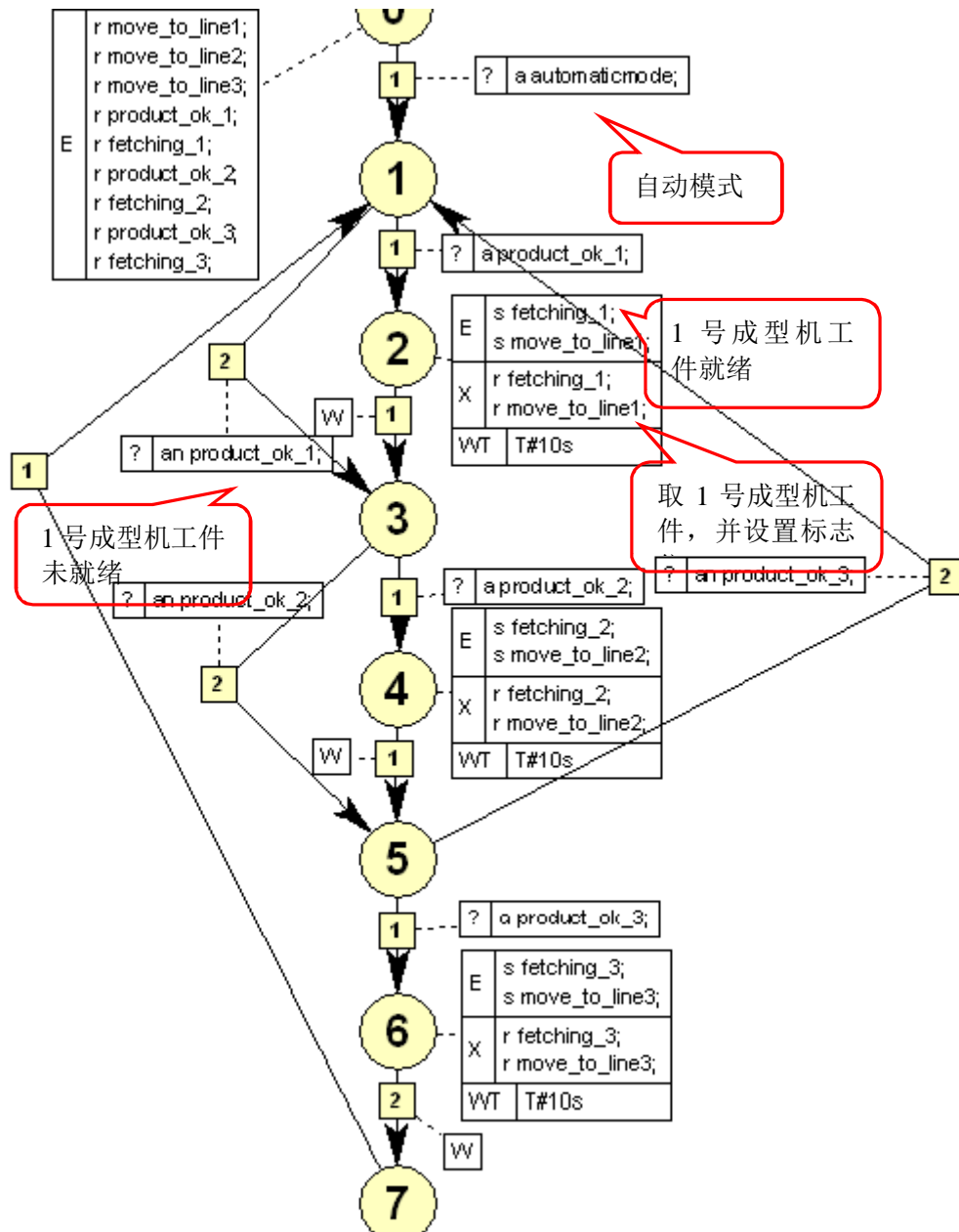


图 5-11: 机械手程序概览

- 12) 双击 Graph group (1) 打开, 右键 Insert instance, 添加项目中的 molding 和 robot
- 13) 也可以右键设置 Run Sequence
- 14) 将 molding 复制并粘贴 2 次
- 15) 选择 molding, 分配参数

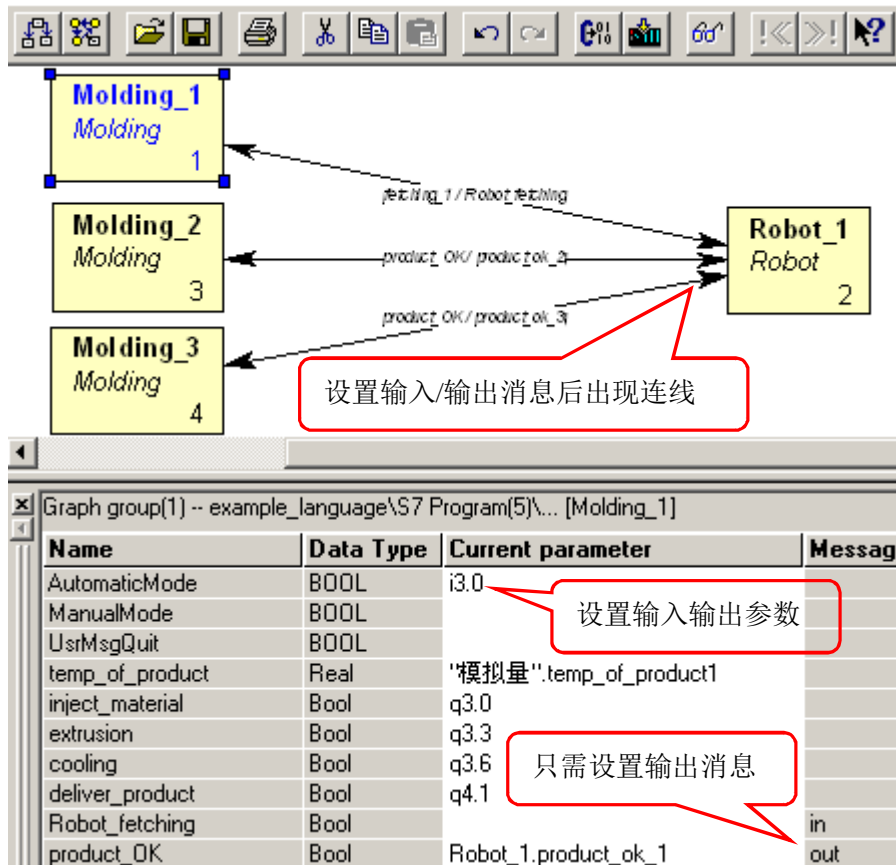


图 5-12: 分配成型机接口参数

16) 依次分配成型机 2, 3 号的接口参数, 在输入/输出消息配置完成后, 模块之间出现连接线

17) 分配机械手接口参数

- 自身的 fetching_1, fetching_2, fetching_3, 分别送给 1-3 号成型机, 来传送是否正在拿某台成型机的工件
- 自身的 Product_ok_1, Product_ok_2, Product_ok_3, 分别接受 1-3 号成型机工件就绪的信号

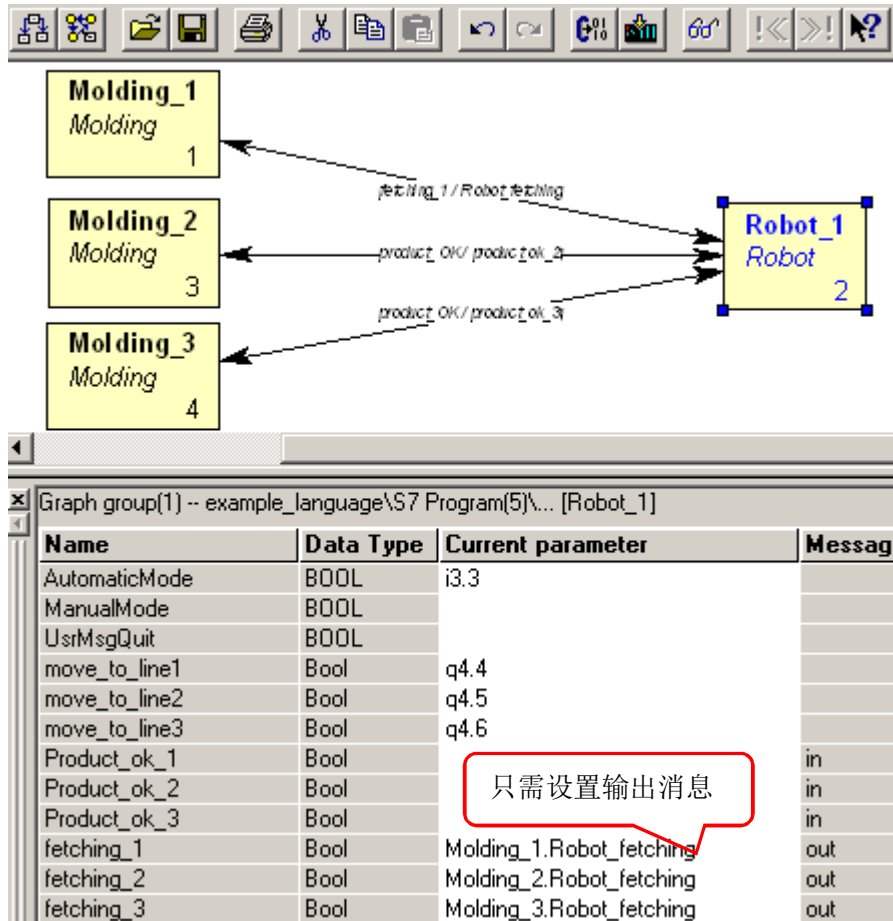


图 5-13: 分配机械手接口参数

18) 在菜单中 Options->Settings for graph groups/state graphs->Compile 设置编译生成的 FC 和 DB 序号。



图 5-14: 设置 GRAPH 参数

19) 编译，在 OB1 中调用 FC30，并下载

20) 监控 graph group

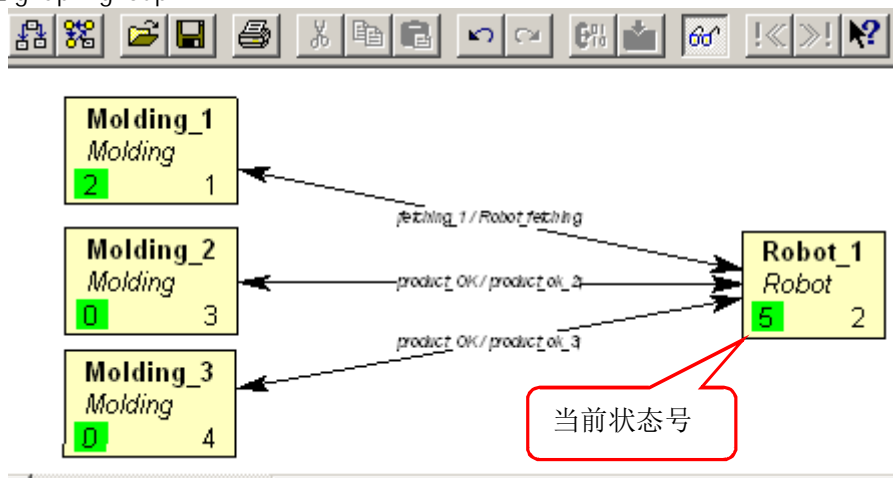


图 5-15: 监控 graph group

21) 双击 mol di ng_1, 可以监控每个状态图

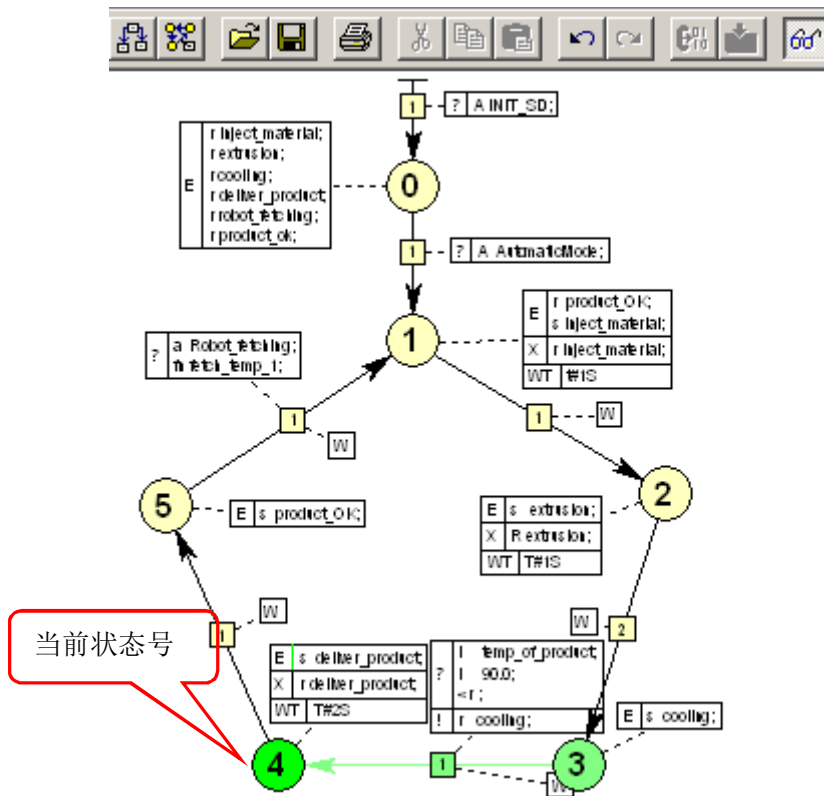


图 5-16: 监控 state graph

22) 在菜单中的 Debug -> Select variable, 选择关心的变量

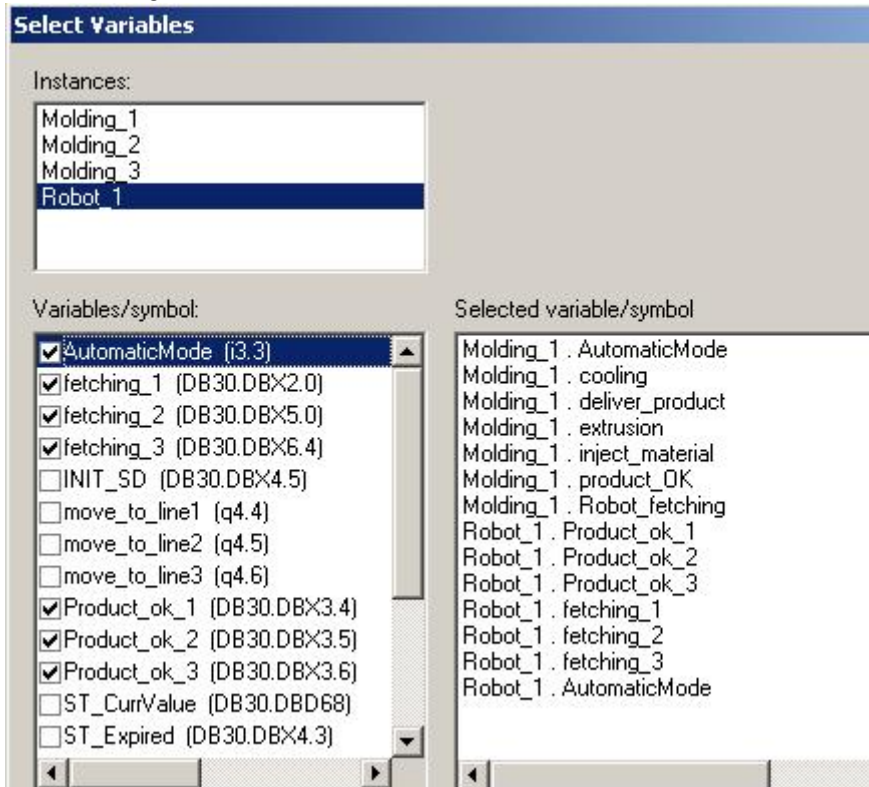


图 5-17: 选择监控变量

23) 确认后, 系统使用变量表进行监控

	Address	Name	Symbol	Display format	Status value
1	I 3.0	AutomaticMode	"molding1_auto"	BOOL	true
2	Q 3.6	cooling	"cooling1"	BOOL	false
3	Q 4.1	deliver_product	"deliver_product1"	BOOL	false
4	Q 3.3	extrusion	"extrusion1"	BOOL	false
5	Q 3.0	inject_material	"inject_material1"	BOOL	true
5	DB30.DBX 3.4	Product_ok_1		BOOL	false
7	DB30.DBX 2.0	fetching_1		BOOL	false
8	DB30.DBX 3.4	Product_ok_1		BOOL	false
9	DB30.DBX 3.5	Product_ok_2		BOOL	false
10	DB30.DBX 3.6	Product_ok_3		BOOL	false
11	DB30.DBX 2.0	fetching_1		BOOL	false
12	DB30.DBX 5.0	fetching_2		BOOL	false
13	DB30.DBX 6.4	fetching_3		BOOL	false
14	I 3.3	AutomaticMode	"robot_auto"	BOOL	true
15	DB10.DBD 8		"模拟量".temp_of_pr	FLO...	0.0

图 5-18: 监控变量

至此，一个简单的 S7-Hi Graph 程序示例就结束了，限于篇幅，本文对例子程序作了简化处理，并没有真正体现 S7-Hi Graph 优势所在。下图为 S7-Hi Graph 手册中的一个状态图，是一个典型应用例子：

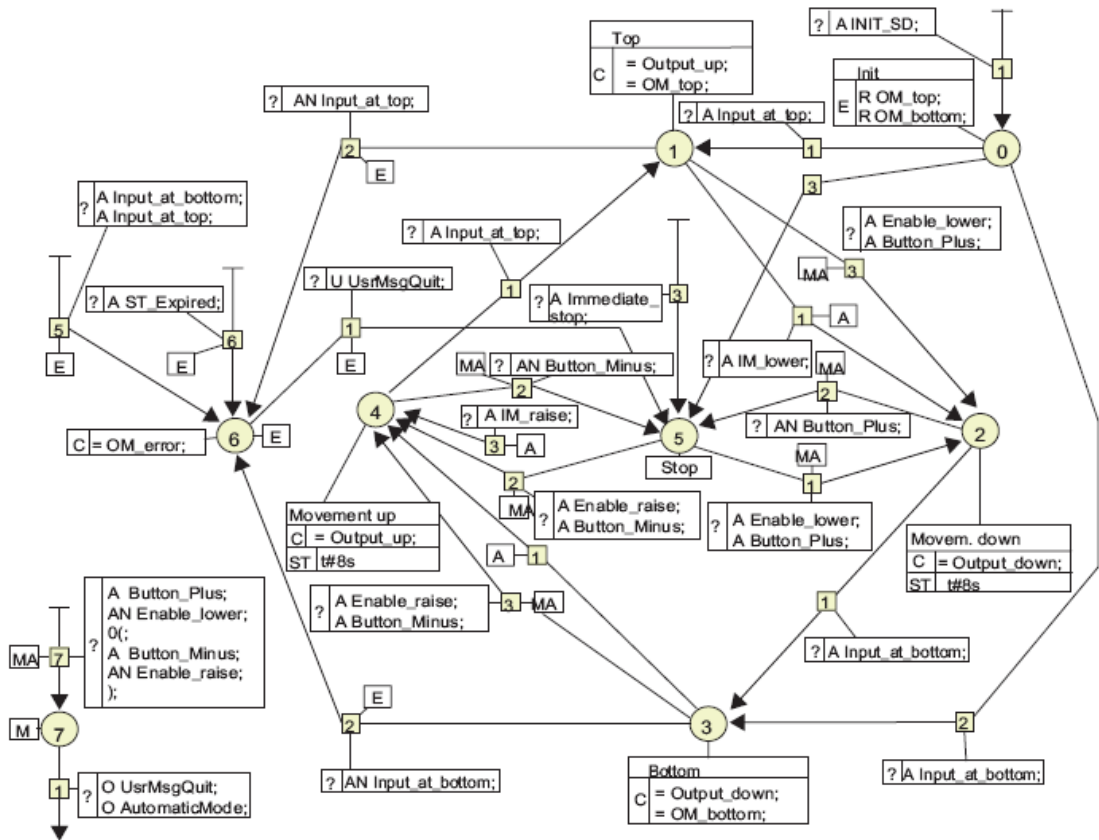


图 5-19: S7-Hi Graph 应用例子

任何编程语言都有其复杂性，并非一朝一夕就可掌握，关于 S7-Hi Graph 的具体使用，请按照

本文提供的地址连接下载 S7-Hi Graph 手册。

5.3. 重要提示:

- ◇ 本文的虚拟工程与真实工程实例有重大差别，示例中并未遵循规范的工程设计流程进行编程，请读者切勿将其与工程实例相混淆。
- ◇ 由于此例子是免费的，任何用户可以免费复制或传播此程序例子。程序的作者对此程序不承担任何功能性或兼容性的责任，使用者风险自负。
- ◇ 西门子不提供此程序例子的错误更改或者热线支持。

6. 常见问题

6.1. 与监控调试相关问题

6.1.1. 问题：S7-Hi Graph 状态图中的各种动作的执行顺序是什么样的

问题：S7-Hi Graph 状态图中的各种动作的执行顺序是什么样的？

解答：此问题在 S7-Hi Graph 手册的 7.2 章节中有详细描述，这里提供一个简单的例子，目的是让用户更直观的理解 S7-Hi Graph 状态图中的各种动作的执行顺序。

下面的例子中，程序在完成初始化后，将从状态 0 执行到状态 3。在此期间，下列动作将被执行：

- 状态 1 的 E, C-, C, X 动作
- 转换动作 !
- 状态 2 的 E, C-, C, X 动作
- 状态 3 的 E, 动作

在此特别提醒用户的是：动作执行顺序不是按照上面顺序进行的！

为了记录动作执行的顺序，每个动作执行完毕后，都将向 MW0 开始的内存区域写当前动作的标识。此功能由辅助功能 FC1 来完成。

OB1 中的代码：

```
CALL FC 31  
INIT_SD: =M100.1
```

用户在变量监控表中将 M100.1 由 FALSE 改写为 TURE，完成初始化，再改写为 FALSE，状态图完成一次流程。

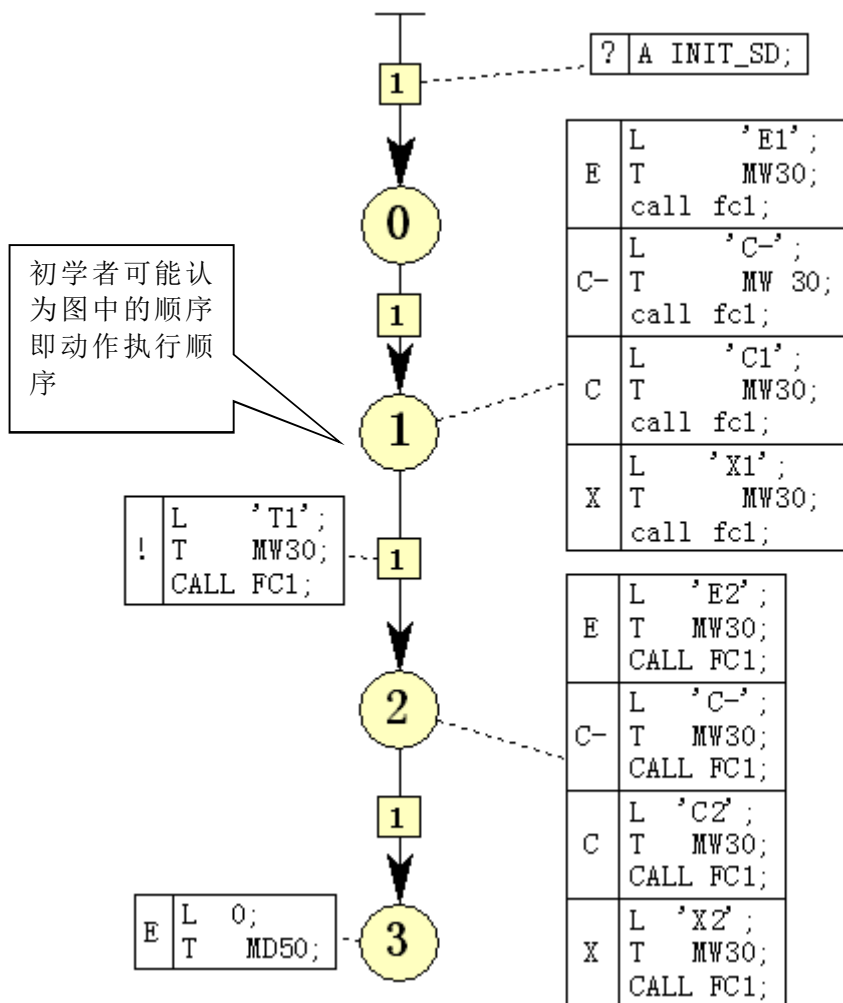


图 6-1:

情况 1: 编译选项中没有选择任何附加设置, 程序执行结果见下图:

Address	Display format	Status value
1 MD 50	HEX	DW#16#00000000
2 MW 30	CHARACTER	'X2'
3		
4 MW 0	CHARACTER	'E1'
5 MW 2	CHARACTER	'C1'
6 MW 4	CHARACTER	'C-'
7 MW 6	CHARACTER	'T1'
8 MW 8	CHARACTER	'X1'
9 MW 10	CHARACTER	'E2'
10 MW 12	CHARACTER	'C2'
11 MW 14	CHARACTER	'C-'
12 MW 16	CHARACTER	'X2'
13 MW 18	CHARACTER	W#16#0000
14 M 100.1	BOOL	false

注意: T1 在 X1 前面

图 6-2: 程序执行结果(a)

情况 2: 状态图编译选项中选择了:

Do not use graph group settings, use instead (不使用图表组设置, 而使用替代设置)

Include preceding cyclic actions in entry cycle (在进入周期时执行 preceding cyclic actions)

程序执行结果见下图:

	Address	Display format	Status value
1	MD 50	HEX	DW#16#00000000
2	MW 30	CHARACTER	'X2'
3			
4	MW 0	CHARACTER	'E1'
5	MW 2	CHARACTER	'C-'
6	MW 4	CHARACTER	'C1'
7	MW 6	CHARACTER	'C-'
8	MW 8	CHARACTER	'T1'
9	MW 10	CHARACTER	'X1'
10	MW 12	CHARACTER	'E2'
11	MW 14	CHARACTER	'C-'
12	MW 16	CHARACTER	'C2'
13	MW 18	CHARACTER	'C-'
14	MW 20	CHARACTER	'X2'
15	M 100.1	BOOL	false

注意: 每个步的 C-都在 E 后多执行了一次

注意: 每个步的 C-都在 E 后多执行了一次

图 6-3: 程序执行结果(b)

情况 3: 状态图编译选项中选择了: (S7-Hi Graph V5.3 及以上版本不需要考虑此情况)

Do not use graph group settings, use instead (不使用图表组设置, 而使用替代设置)

Execute cyclic action with RLO=0 (RLO 设置为 0, 并执行 cyclic actions)

程序执行结果见下图:

	Address	Display format	Status value
1	MD 50	HEX	DW#16#00000000
2	MW 30	CHARACTER	'X2'
3			
4	MW 0	CHARACTER	'E1'
5	MW 2	CHARACTER	'C1'
6	MW 4	CHARACTER	'C-'
7	MW 6	CHARACTER	'T1'
8	MW 8	CHARACTER	'X1'
9	MW 10	CHARACTER	'E2'
10	MW 12	CHARACTER	'C2'
11	MW 14	CHARACTER	'C-'
12	MW 16	CHARACTER	'X2'
13	MW 18	CHARACTER	W#16#0000
14	M 100.1	BOOL	false

注意：在每个 T 动作后都将多执行一次 C 动作，此处应为 C1（此图中为 V5.3 的监控结果，不受影响）

图 6-4: 程序执行结果(c)

6.2. 与使用技巧相关问题

6.2.1. 问题：S7-Hi Graph 状态图中的支持间接寻址吗

问题：S7-Hi Graph 状态图中的支持间接寻址吗？

解答：S7-Hi Graph 状态图中的不完全支持间接寻址（尽管支持 POINTER 数据类型）。对于跳转指令，标号，同样不支持，相关详细内容可以参考手册的第 8 章，如果用户希望使用这些功能，可以把这些指令编写在 FB 或 FC 中，再在 S7-Hi Graph 调用的动作中调用。如上例中的 FC1 的使用。

6.2.2. 问题：为什么无法设置 Delay time

问题：为什么无法设置 Delay time ？

解答：Delay time 功能需要 S7-Hi Graph V5.3 及以上版本支持，此前的 V4, V5, V5.2 都没有此功能。

6.2.3. 问题：S7-HiGraph 状态图中的支持 ARRAY 数据类型吗？

问题：S7-Hi Graph 状态图中的支持 ARRAY 数据类型吗？

解答：S7-Hi Graph 状态图中的不支持 ARRAY 数据类型，详细内容可以参考手册的第 8 章

6.2.4. 问题：如何通过 IN、OUT 和 IN-OUT 变量实现 UDT(用户自定义数据类型)?

问题：如何在 S7-Hi Graph 中通过 IN、OUT 和 IN-OUT 变量实现 UDT(用户自定义数据类型)？

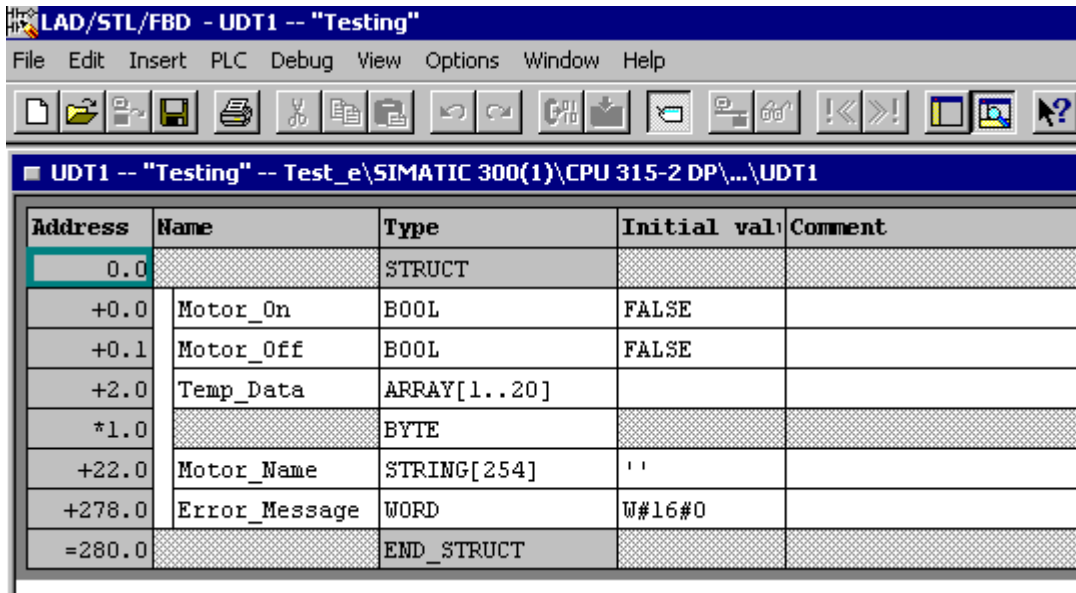
解答：在 S7-Hi Graph 中只能声明静态“UDT”类型的变量。如果想定义“UDT”类型的 IN、OUT 和

IN-OUT 变量，请按照下列表格中的步骤进行。

步骤 1: 在 SIMATIC Manager 中点击块文件夹，在 STEP 7 项目中插入一个 UDT，菜单路径为

- "Insert > S7 block > Data type"

在 UDT 的属性中为其分别符号名 (例如为 UDT1 分配 "Testing") 随后点击 "OK" 保存。打开 UDT " Testing" 并在表中输入不同的变量/数据类型。



The screenshot shows the SIMATIC Manager interface for defining a UDT. The title bar reads "LAD/STL/FBD - UDT1 -- 'Testing'". The menu bar includes File, Edit, Insert, PLC, Debug, View, Options, Window, and Help. The toolbar contains various icons for file operations and navigation. The main window title is "UDT1 -- 'Testing' -- Test_e\SIMATIC 300(1)\CPU 315-2 DP\...\UDT1". Below the title bar is a table defining the structure of the UDT.

Address	Name	Type	Initial val	Comment
0.0		STRUCT		
+0.0	Motor_On	BOOL	FALSE	
+0.1	Motor_Off	BOOL	FALSE	
+2.0	Temp_Data	ARRAY[1..20]		
*1.0		BYTE		
+22.0	Motor_Name	STRING[254]	''	
+278.0	Error_Message	WORD	W#16#0	
=280.0		END_STRUCT		

图 6-5: 定义 UDT

步骤 2: 随后生成一个 UDT-类型的数据块，菜单路径为

- "Insert > S7 block > Data type"

并且为其分配一个符号名，例如 "Testing_Data"。如图所示，对于 DB1，从列表中选择 "DB of type" 和 "UDT1"。

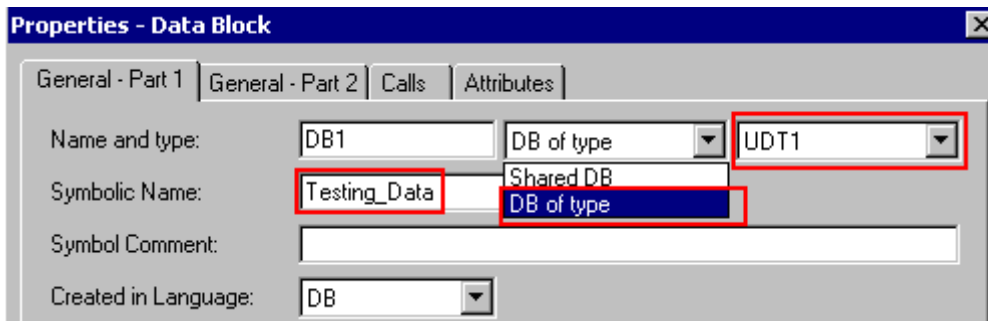


图 6-6: 根据 UDT 生成数据块

步骤 3: 在 Source 文件夹中, 打开图表组然后打开相关的状态图。在接口中输入即将通过数据类型“UDT”定义的变量。举例来说, 对于 IN 变量“ Mot_1” 和“ Temp_1”, 数据类型(BOOL 和字节)必须与 UDT1 中变量的数据类型匹配。

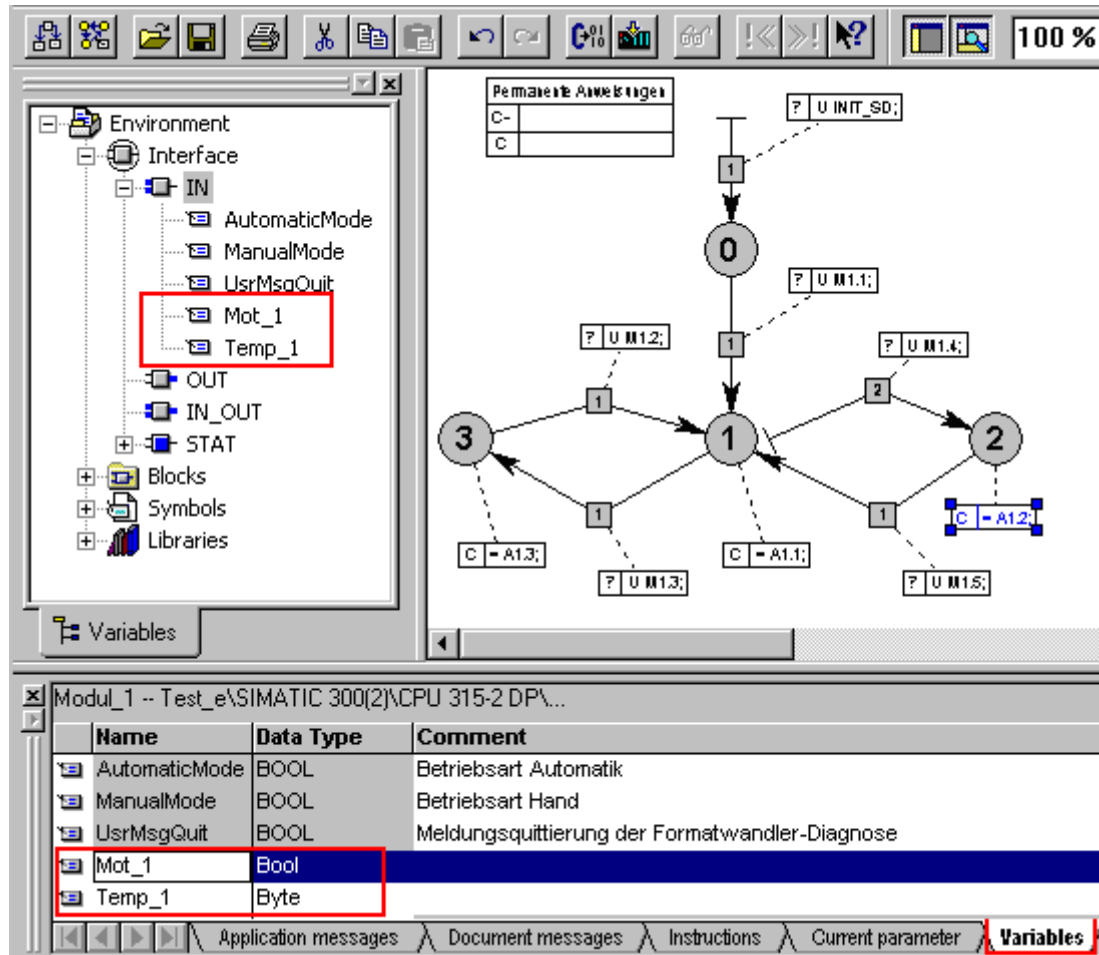


图 6-7: 声明变量

声明 IN 变量之后, 保存状态图表。

步骤 4: 返回到图表组并选择状态图(在本例中为 Modul 1_1)。在 Details 中选择标签“Current parameter” 并向 IN 变量分配 UDT 参数。然后保存并编译图表组。

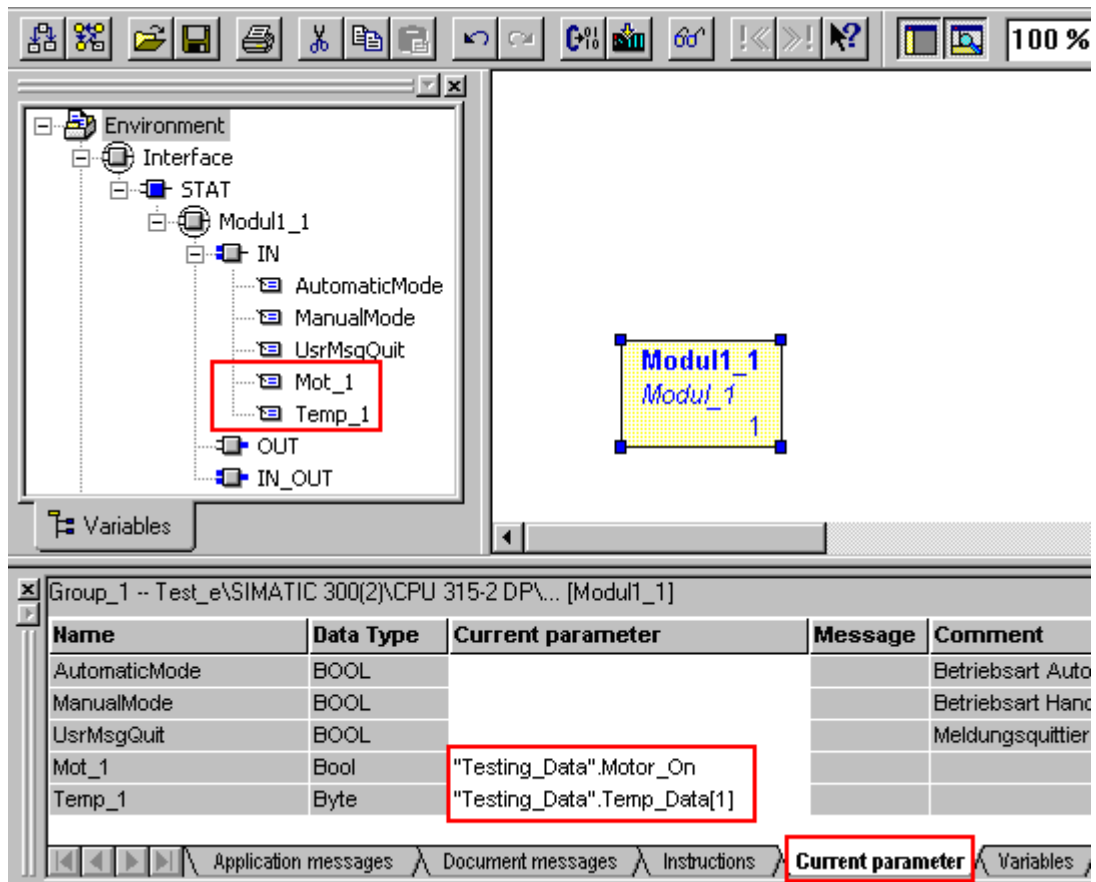


图 6-8: 定义参数

步骤 5: 此时在转换中就可以使用已分配 UDT 参数的 IN 变量进行编程。如下图所示, 在转换中使用 IN 变量 "Mot_1" 进行与操作, 该变量被分配了属于数据块 DB1 (符号名为 "Testing_Data") 数据类型 UDT1 (符号名为 "Testing") 的参数 "Testing_Data.Motor_On"。

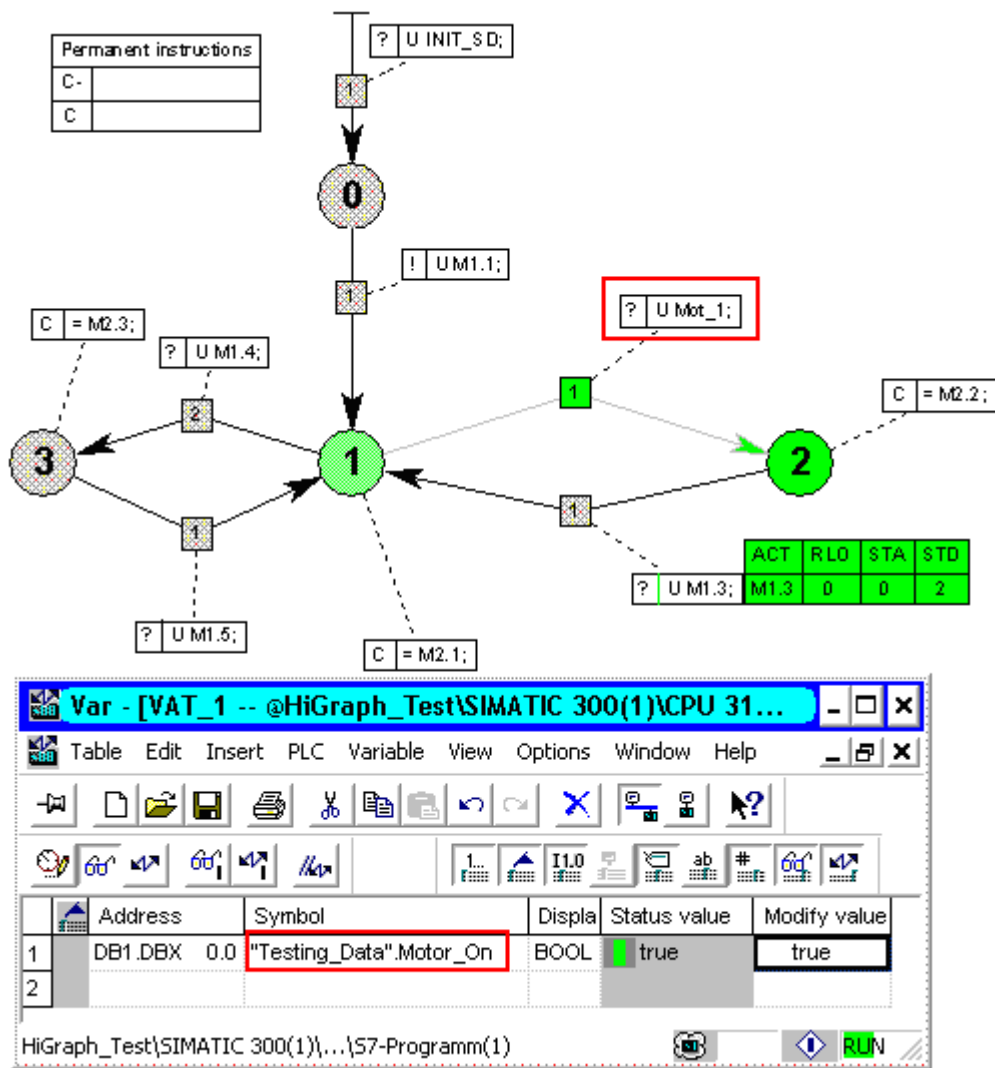


图 6-9: 程序运行监视结果

6.2.5. 问题：状态图与状态图之间的消息都支持哪些数据类型？

问题：状态图与状态图之间的消息都支持哪些数据类型？

解答：目前 S7-Hi Graph 仅支持布尔数据类型的消息格式，如果状态图之间要交换其它数据类型，可以使用全局变量。