```
port_label('input',1,'signal');
port_label('output',1,'gain');
port_label('output',2,'saturation');
port_label('output',3,'sample');
port_label('output',4,'error');
image(imread('dianji.jpg'))
```

第 3 章 Stateflow 建模与应用

Stateflow 是有限状态机〔finite state machine〕的图形工具，它通过开发有限状态机和流程图扩展了 Simulink 的功能。Stateflow 使用自然、可读和易理解的形式，可使复杂的逻辑问题变得清晰与简单，并且还与 MATLAB\Simulink 紧密集成，为包含控制、优先级管理、工作模式逻辑的嵌入式系统设计提供了有效的开发手段，是本书的核心内容之一。读者在 5~8 章将看到 Stateflow 应用于 MCU 器件的嵌入式开发，尤其显得得心应手，一些采用传统方法难于实现的算法，如果利用 Stateflow 建模却非常容易。大到导弹、航空航天器的控制，小到点亮一个发光二极管，Stateflow 都非常称职。Stateflow 状态图模型，还可利用 Stateflow Coder 代码生成工具，直接生成 C 代码。

Stateflow 的主要功能包括：

- 使用层次化、可并行的、有明确执行语义的元素，来描述复杂的逻辑系统。
- 采用流程图定义图形化函数。
- 利用真值表实现表格形式的功能。
- 使用临时逻辑处理状态转移与事件。
- 支持 Mealy 和 Moore 有限状态机。
- 可集成用户自定义的 C 代码。
- 可用动画的形式显示状态图的仿真运行过程，并可记录数据。
- 调试器使用图形化断点进行单步调试，并可观察其中的数据。

本章主要内容：

- Stateflow 工作原理与根本概念
- 建立 Stateflow 状态图与流程图
- Stateflow 的层次结构与并行机制
- Stateflow 应用

3.1Stateflow 根本概念

Stateflow 对象可分为图形对象与非图形对象。

图形对象有状态、历史节点、迁移、默认迁移、连接节点、真值表、图形函数、Embedded MATLAB 函数、盒函数、Simulink 函数；非图形对象有事件、数据、

目标。本节首先介绍常用的对象：状态、迁移、数据、事件的概念和使用，连接节点留待 3.3 节、历史节点留待 3.4 节，其余对象留待 3.5 节说明。

Stateflow 状态机使用一种基于容器的层次结构管理 Stateflow 对象，也就是说，一个 Stateflow 对象可以包含其他 Stateflow 对象。

最高级的对象是 Stateflow 状态机，它包含了所有的 Stateflow 对象，因此也就包含了 Simulink 中的所有 Stateflow 状态图，以及数据、事件、目标对象。

同样地，状态图包含了状态、盒函数、函数、数据、事件、迁移、节点与注释事件〔note events〕。用户可以使用这一系列对象，建立一个 Stateflow 状态图。而具体到一个状态，它也可以包含上述的对象。

图抽象地说明了这样的关系，而图那么具体地说明了 Stateflow 状态机的组成。

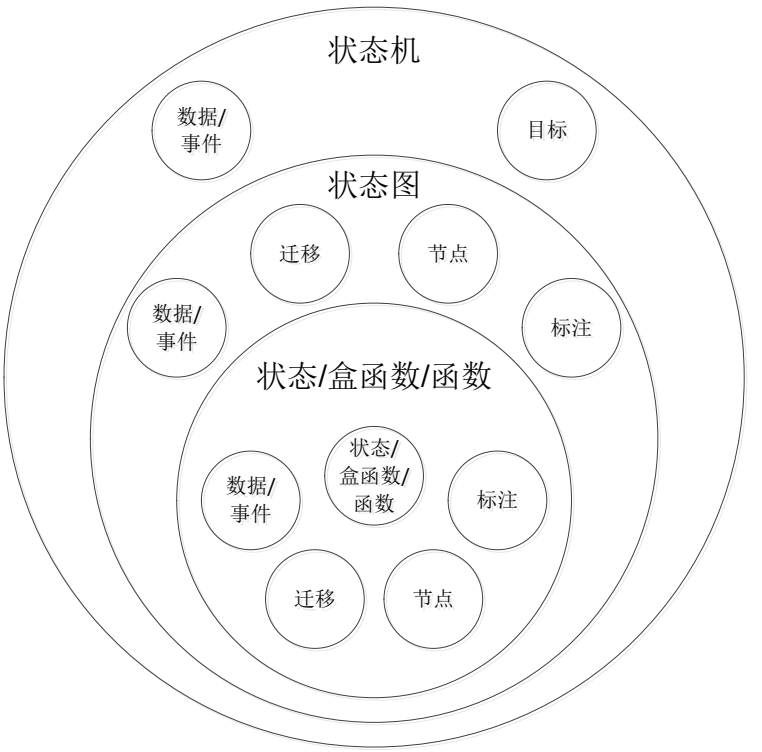

图 3.1.1Stateflow 层次机构〔数据字典〕

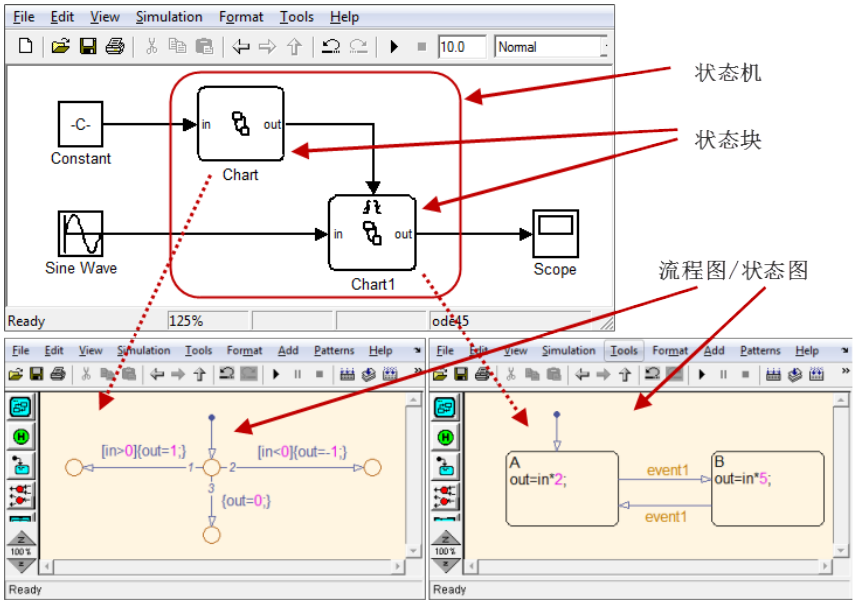

图 3.1.2 Stateflow 状态机的组成

3.1.1 状态图编辑器

在 Simulink 模块库浏览器，找到 Stateflow 模块，如下图 3.1.3，添参加模型窗口，如图 3.1.4 所示。

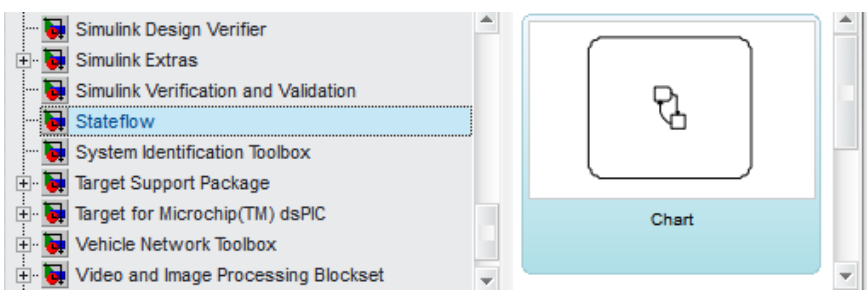

图 3.1.3 Stateflow 模块

用户也可以使用以下命令，建立带有 Stateflow 状态图的 Simulink 模型，如下图 3.1.4，同时翻开 Stateflow 模块库，如图 3.1.5 所示。

```
>>sf
```

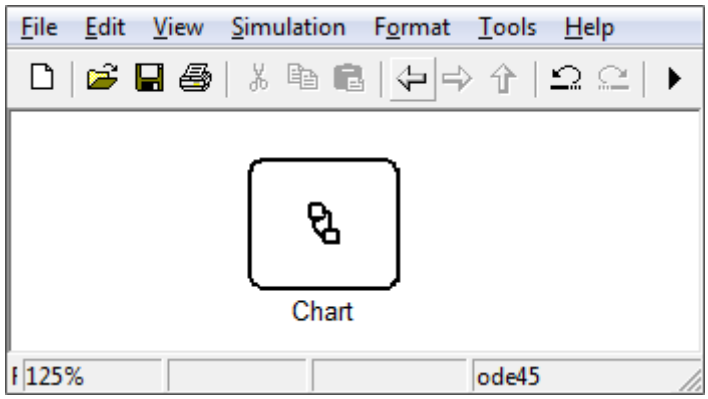

图 3.1.4 带有 Stateflow 状态图的 Simulink 模型

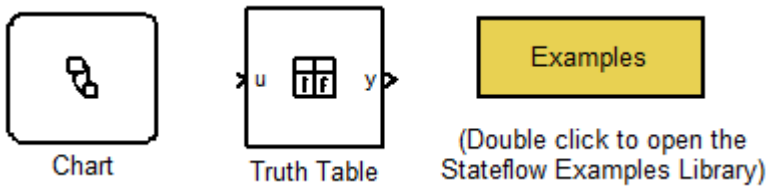

图 3.1.5 Stateflow 模块库

用户还可以直接使用以下命令，快速建立带有 Stateflow 状态图的 Simulink 模型。

```
>>sfnew
```

双击 Chart 模块，翻开 Stateflow 编辑器窗口，如下图，左侧工具栏列出了 Stateflow 图形对象的按钮。

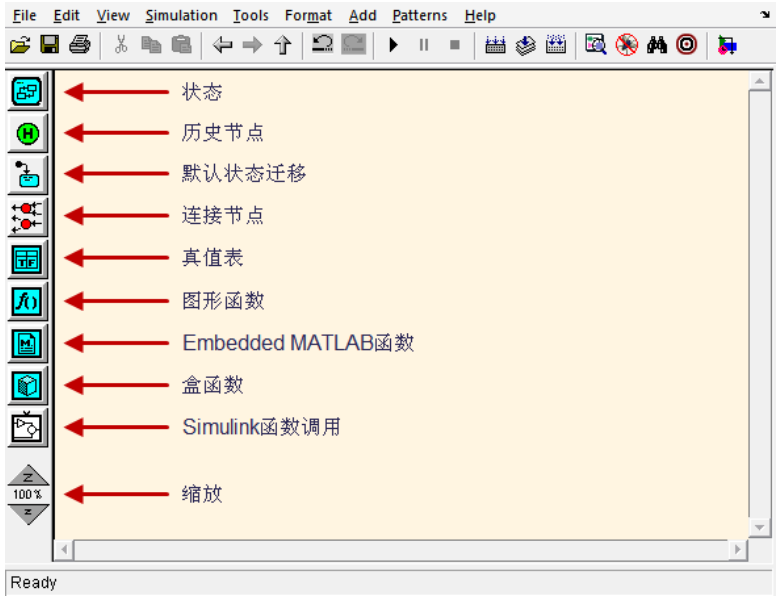

图 3.1.6 Stateflow 编辑器窗口

3.1.2 状态

状态可以理解为事件驱动系统中的模式，可分为激活与非激活状态，而状态是否激活那么是由状态图中的事件与条件决定的，假设没有预先定义的事件或条件发生，状态将一直保持其原先的激活或非激活状态。

1. 状态的层次结构

状态可以包含除了目标〔详见第 3.6.6 节〕以外的所有 Stateflow 对象，所以状态内部可以有其他状态，如图 3.1.7 所示，处于外层的 A 称作超状态〔或父状态〕，处于内部的 B 称作子状态。

每一个状态都有其父状态，图 3.1.7 中，状态 A 的父状态就是 Stateflow 状态图本身。

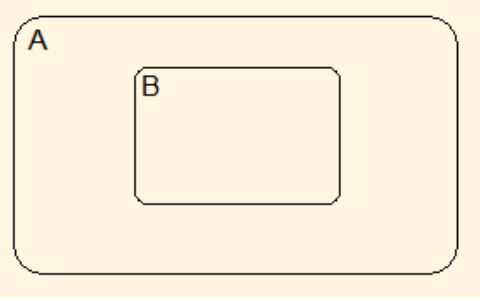

图 3.1.7 超状态与子状态

2. 状态的横向结构

在 Stateflow 状态图的顶层或某一超状态下，通常并存有多个状态，它们之间的关系可分为互斥与并行。

(1) 互斥状态〔OR〕

互斥状态的矩形框边缘显示为实线，同一级的互斥状态，至多允许激活一个状态。如下图 3.1.8 的状态图，状态 A 与状态 B 是互斥的，它们只能有一个处于激活状态；当状态 A 被激活时，同样其子状态 A1 与 A2 也只能有一个处于激活状态。

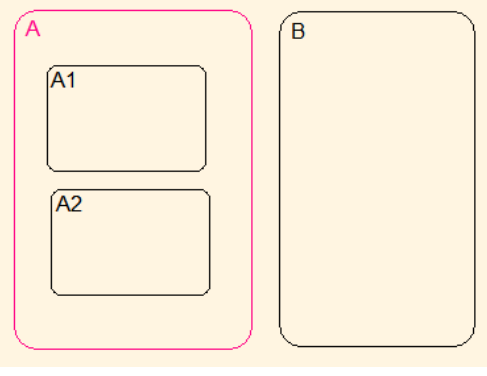

图 3.1.8 互斥状态

(2) 并行状态〔AND〕

并行状态的矩形框边缘显示为虚线，同一级的并行状态，可在同一时刻被激活。如下图 3.1.9 的状态图。状态 A 与状态 B 是并行的，它们可同时处于激活状态；子状态 A1 与 A2 也同时处于激活状态，而子状态 B1 与 B2 只能有一个处于激活状态。

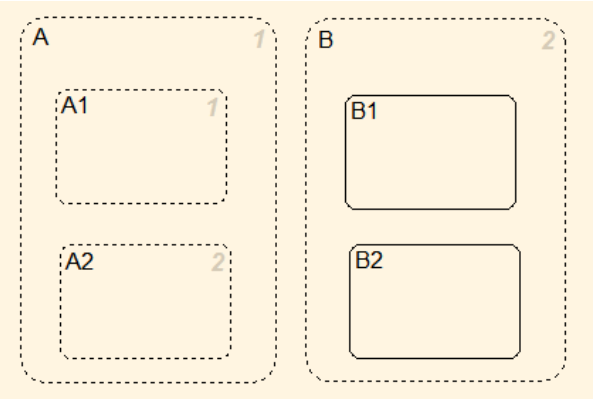

图 3.1.9 并行状态

状态层次结构与并行机制的详细概念与应用，见 3.4 与 3.5 节。

3. 状态标签

状态名仅是状态标签的一局部，完整的标签格式如下，第一行是状态名，以下假设干行是各类动作，用户可以设置全部或局部的状态动作，当然也可以不设置任何动作。

```
name/                                状态名
entry:     entry actions             进入该状态时的动作
during:    during actions            处于该状态时的动作
exit:      exit actions              退出该状态时的动作
on event_name:   on event_name actions   某事件发生时的动作
bind:      events, data              指定需要限制作用范围的事件与数据
```

①状态名

状态名可由字母、数字、下划线组成，如果状态名后跟随的是回车符，那么斜线是可有可无的。根据 Stateflow 的分层结构，同级的各个子状态不允许重名，但不同级的状态那么不受限制。

图 3.1.10 所示的 Stateflow 状态图是有效的，尽管看上去状态 C1、C2 有重名现象，但在 Stateflow 分层结构中，它们的全名分别是：

```
A.On
A.Off
B.On
B.Off
```

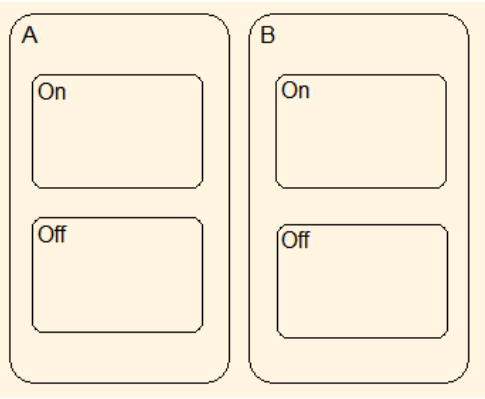

图 3.1.10 状态名

②状态动作

状态动作如表 3.1.1 所列。

表 3.1.1 状态动作类型

动作类型	缩写	说明
entry	en	进入当前状态时的动作
during	du	处于当前状态，并且某事件发生时的动作 Executes when the state isactive and a specific eventoccurs
exit	ex	离开当前状态时的动作
bind	无	约束一个事件或数据，使得仅当前状态及其子状态有权限播送该事件或修改该数据
on event_name	无	当前状态接收 1 次播送事件时的动作
on after(n,event_name)	无	当前状态完整接收 n 次播送事件后的动作
on before(n,event_name)	无	当前状态完整接收 n 次播送事件前的动作
on at(n, event_name)	无	当前状态完整接收 n 次播送事件时的动作
on every(n,event_name)	无	当前状态每接收 n 次播送事件时的动作

每个动作类型，用户可指定多个具体动作，每个动作之间以回车、分号、逗号区隔，动作类型关键词后必须跟随一个半角冒号。

(1) entry 动作

关键词为 entry〔或缩写为 en〕。如果用户在状态名后参加斜线，并直接跟随具体动作，那么该动作默认为进入动作。如下图 3.1.11，进入状态 A 时，y=3，同时又执行 y++，最终的结果 y=4。

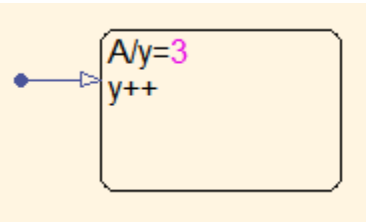

图 3.1.11 entry 动作

(2) during 动作

关键词为 during〔或缩写为 du〕。如下图 3.1.12，进入状态 A 时，y=3，同时不断执行 y++。假设求解器的定点步长取 0.2，仿真时长取 2，那么最终的结果 y=13。

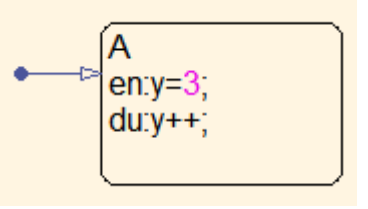

图 3.1.12 during 动作

(3) exit 动作

关键词为 exit〔或缩写为 ex〕。如下图 3.1.13，系统处于状态 A，当 A 的激活时间到达 5 个仿真步长，退出状态 A，进入状态 B，最终的结果 y=4，如图 3.1.14 所示。

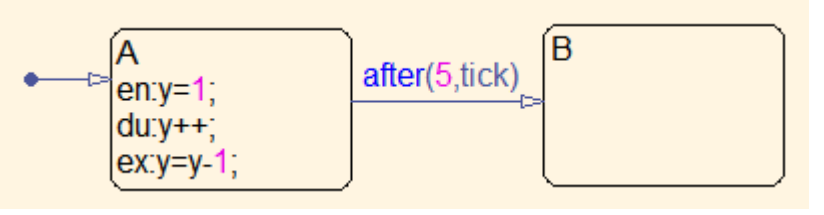

图 3.1.13 exit 动作

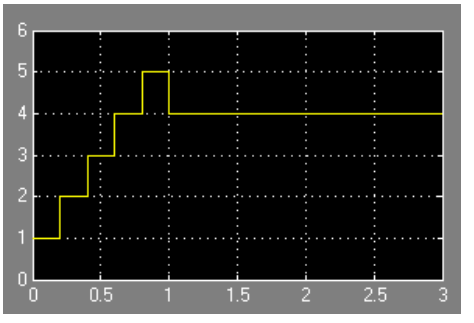

图 3.1.14 输出结果

(4) 播送事件动作

表所列的播送事件动作，能实现各种的事件触发。

以单次播送事件动作为例，关键词为 on event_name，其中 event_name

表示某一播送事件名，事件名应是唯一的。如下图 3.1.15，系统处于状态 A，当检测到事件 stop，立即执行 c()。

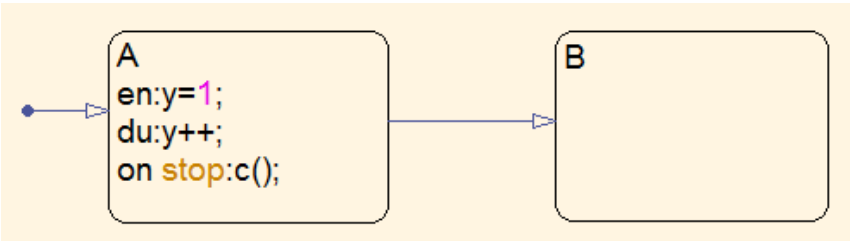

图 3.1.15 播送事件动作

(5) bind 动作

关键词为 bind。如下图 3.1.16，变量 y、事件 start 被绑定在状态 A，这表示仅有 A 状态及其子状态有权限修改变量 y 并播送事件 start，其他状态 B 能够读取变量 y、监听到事件 start，但无权修改变量 y、播送事件 start。

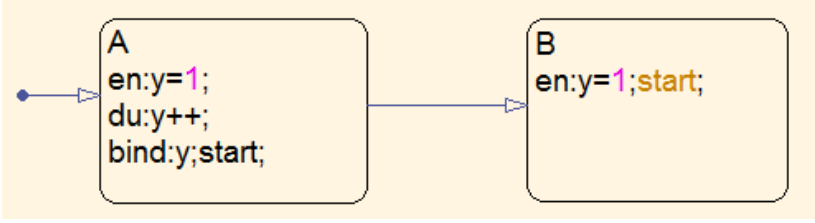

图 3.1.16 bind 动作

假设运行该状态图，系统提示变量 y 仅能由状态 A 及其内部的状态迁移修改，事件 start 仅能在状态 A。

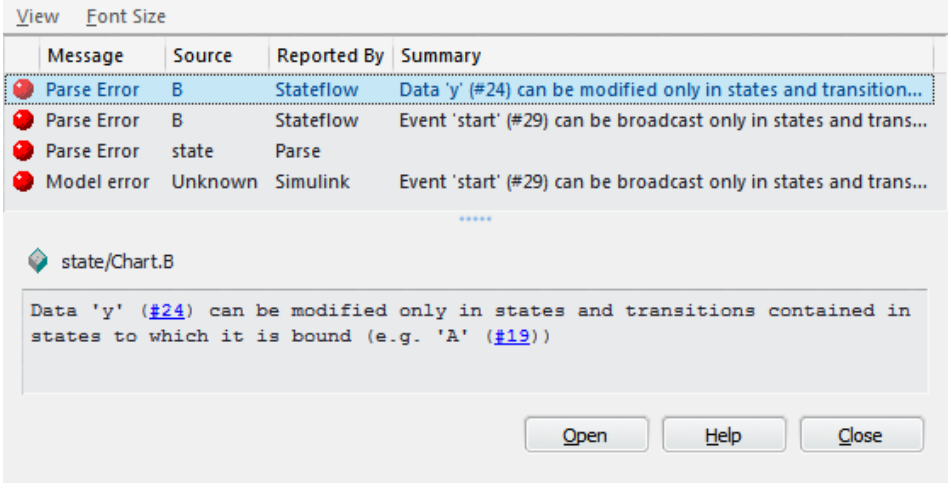

图 3.1.17 错误提示

与其他动作不同，bind 动作不需要判断当前状态是否已激活，也就是说它在整个 Stateflow 状态图范围内都是有效的，因此不同状态不允许约束同一个变量与事件。

如下图 3.1.18，状态 A、B 同时约束了变量 y，系统提示这是不允许的。

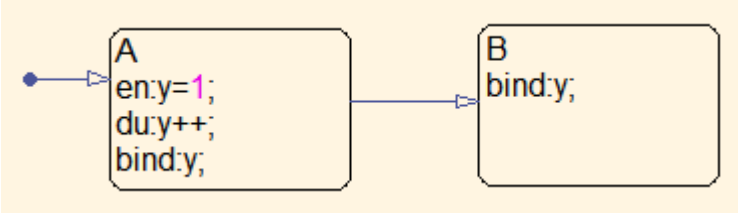

图 3.1.18 无效的 bind 动作

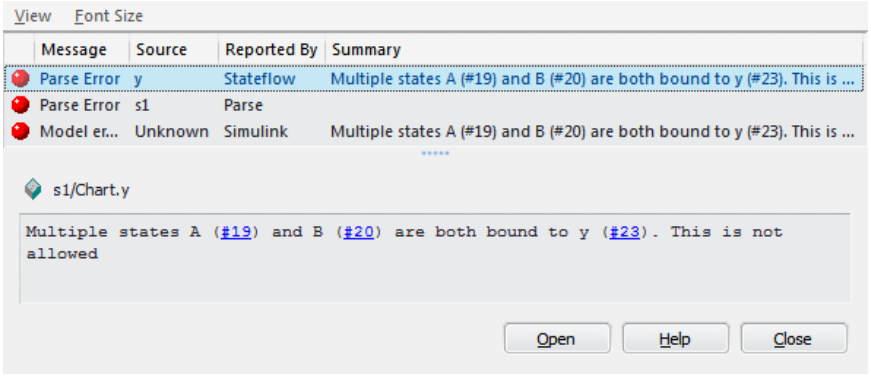

图 3.1.19 错误提示

3.1.3 迁移

1.迁移

Stateflow 状态图使用一条单向箭头曲线表示迁移，它将两个图形对象连接起来，多数情况下，迁移是指系统从源状态向目标状态的转移。

在迁移曲线上加上标签，可以指定系统在何种条件下从源状态向目标状态转移。

如下图 3.1.20，当系统处于状态 A1 时间到达 1 秒，即向状态 A2 迁移。

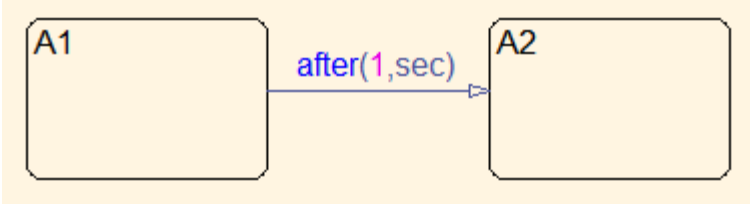

图 3.1.20 状态迁移

2. 默认迁移

默认迁移是一种特殊的迁移形式，它没有源对象。用于指定同一级有多个互斥状态并存时，首先激活的状态。

某些情况下，默认迁移也可以参加标签，限制其所指向目标状态的激活。

如下图 3.1.21，状态 A1 与 A2 是互斥的，当它们的父状态 A 激活时，状态 A1 也同时激活。

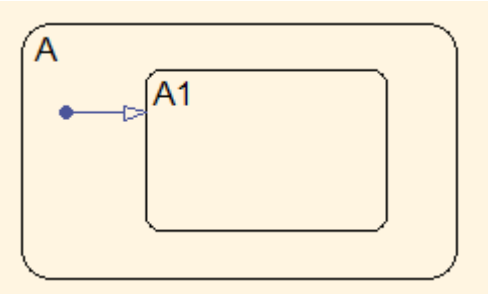

图 3.1.21 默认迁移

3. 迁移标签

迁移标签的完整格式如下，它可用于一般迁移与默认迁移，如下图 3.1.22。

```
event[condition]{condition_action}/transition_action
```

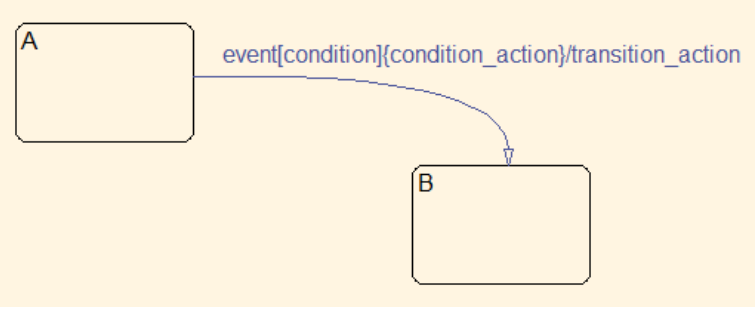

图 3.1.22 完整的迁移标签

各字段的意义如表 3.1.2 所示：

表 3.1.2 迁移标签字段

标签字段	说明
event	引发迁移的事件
[condition]	条件动作与迁移的发生条件
{condition_action}	当条件为真时，执行的动作
/transition_action	发生迁移，进入目标状态前所执行的动作

① 事件

指定迁移的触发事件。如果用户另行指定了触发条件，那么当条件为真，且发生该触发事件时，即发生迁移。这是个可选项，如果用户不指定触发事件，那么任何事件都能够引发该迁移。多个触发事件之间使用逻辑或运算符“|”分隔。

如图，当条件 after(1,sec)为真时，触发了迁移，系统状态从 A1 变成 A2。

②条件

条件是一个布尔表达式，当它为真时，一旦发生指定的触发事件，那么发生迁移。条件表达式的前后必须使用方括号“[ ]”包围。

如下图，当条件[y>=3]为真时，发生迁移。

③条件动作

当条件表达式为真时，立刻执行条件动作。假设事先未指定条件，系统那么假设条件为真，并执行该条件动作。

如下图，当条件[y>=3]为真，条件动作{ y=10 }立刻执行。

④迁移动作

当迁移目标有效时，执行迁移动作。假设迁移标签由多个字段组成，那么当整个标签有效时，执行迁移动作。

如下图，当条件[y>=3]为真，且目标状态 B 有效时，发生迁移，并执行迁移动作 z=20。运行结果如图 4 所示：

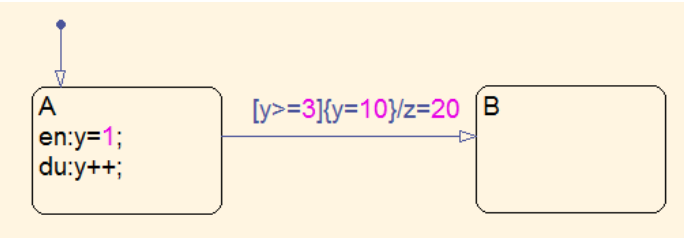

图 3.1.23 迁移条件与动作

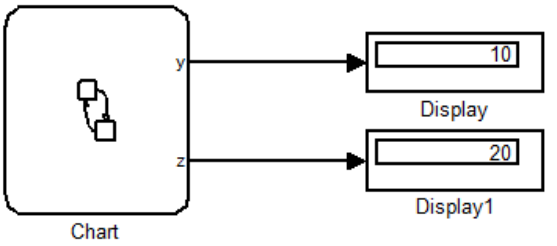

图 3.1.24 输出结果

4. 迁移有效条件

对于非默认的迁移，当源对象处于激活状态的且迁移标签有效时，发生迁移；对于默认迁移，当其父状态被激活时，发生迁移。

表 3.1.3 列出了迁移标签的有效条件，用户可以根据需要，选择性地输入迁移标签的局部或全部字段。

表 3.1.3 迁移标签有效条件

标签内容	标签有效条件
仅事件	该事件发生
事件与条件	该事件发生及条件为真
仅条件	任何事件发生及条件为真
仅行动	任何事件发生
空	任何事件发生

3.1.4 数据与事件

图所示的数据字典中，数据与事件是合并在一个圆圈内的，这说明他们有相似之处，这里合并介绍。

1. 数据

数据是非图形的对象，它有一个很重要的特性：作用域，用户在使用数据时必须明确定义该特性。

根据作用域的不同，数据可分为：

- Stateflow 状态图本地数据〔Local〕；
- 自外部 Simulink 模块输入的数据〔Input from Simulink〕；
- 向外部 Simulink 模块输出的数据〔Output from Simulink〕；
- 临时数据
- 定义在 MATLAB 工作空间的数据；
- 常数〔Constant〕；
- 向 Simulink 模型与 Stateflow 状态图外部的目标〔代码〕导出的数据；
- 自 Simulink 模型与 Stateflow 状态图外部的源代码导入的数据。

数据的简单使用，见 3.2.4 小节。

2. 事件

事件也是非图形的对象，它驱动着整个 Stateflow 状态图的运行。如同数据，事件同样有它的作用域，

根据作用域的不同，事件可分为：

- Stateflow 状态图本地事件；
- 自外部 Simulink 模块输入的事件；
- 向外部 Simulink 模块输出的事件；

事件的简单使用，见 3.2.4；事件的分类，见 3.5 各小节。

3. 动作

Stateflow 状态图支持状态动作、条件动作、迁移动作，已在上文做了简要介绍。这里所说的动作可以是一个函数调用，播送事件，数学运算等等。

例如：

函数调用：ml.log10(x);……

事件播送：Start;Stop;……

数学运算：x=1;y=2;z=x+y;……

3.1.5 对象的命名规那么

以上简要介绍了常用对象的概念，用户可以使用任意的字母、数字与下划线的组合为这些对象命名，但名称不能以数字开头，中间也不能有空格。

由于 Real-Time Workshop 代码生成工具的限制，对象的名称不能超过一定长度，用户可以在模型参数设置对话框的 Real-Time Workshop→Symbols 面板进行修改，默认的长度是 31，最大的长度是 256，如下图。

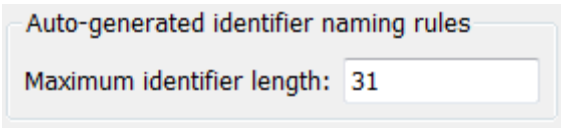

图 3.1.25 设置对象名称的长度

表列出了一些关键字，它们是 Stateflow 动作语言的组成局部，因此是不能用来为对象命名的。

表 3.1.4 关键字

关键字	在 Stateflow 中的用途
hasChanged,hasChangedFrom,hasChangedTo	变更监测
complex, imag, real	复数数据
boolean, double, int8,int16, int32, single,uint8, uint16, uint32	数据类型
cast, fixdt, type	数据类型操作
send	明确事件
change, chg, tick, wakeup	隐含事件
false, inf, true, t	标志位
matlab, ml	MATLAB 函数与数据
bind, du, during, en,entry, ex, exit, on	状态动作
in	状态激活
after, at, before, every,sec, temporalCount	时间逻辑

3.2Stateflow 状态图

长跑比赛时，通常要用到以圈计时的方法，它的意思是：计时器初次开启时，2 组数码管皆清零；运发动出发时，按下 Start 按钮开始计时，数码管 1 显示实时时间；

第一次回到起点，表示跑完一圈，这时按下 LAP 按钮，数码管 2 显示当前的时间值，表示一圈所花费的时间，但比赛仍在进行，因此计时器仍然在计时；再次按下 Start 按钮，2 组数码管同时显示最后的时间；第三次按下 Start 按钮，2 组数码管清零，回到初始状态。本节以此为例，说明 Stateflow 状态图的建立过程。

3.2.1 状态

1.添加状态

新建一个空白的 Stateflow 模型，单击状态按钮，并在 Stateflow 窗口的适当位置再次单击，参加一个状态，如下图 3.2.1。

在参加之前，用户可随时按下键盘的 ESC 键，或再次单击按钮，取消添加。

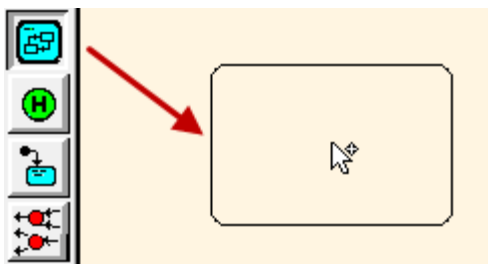

图 3.2.1 添加状态

2.状态命名

在状态矩形框左上角的编辑提示符后，输入状态的名称，如 stop，如下图 3.2.2。假设需要修改状态名，可将鼠标移至名称附近，待光标变成编辑样式时，再单击修改，如图 3.2.3 所示。

图 3.2.2 状态命名

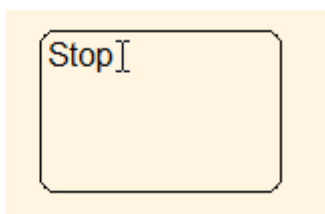

图 3.2.3 状态名修改

3.添加子状态

将鼠标移至状态矩形框 4 个角落的任意一个，调整其大小，如下图 3.2.4。

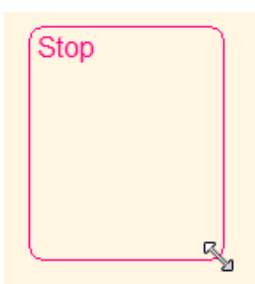

图 3.2.4 调整状态框

再按步骤①、②，添加状态 Reset、Finished，放置在状态 Stop 的矩形框内，这时 Stop 为超状态，Reset、Finished 为子状态，如下图 3.2.5。

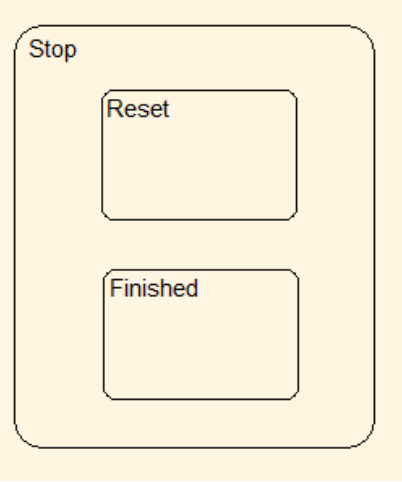

图 3.2.5 父状态与子状态

3.2.2 迁移

1. 添加迁移

将鼠标移至源状态矩形框的边缘，当光标变成十字时，如下图 3.2.6，按下左键并拖向目标状态的边缘，然后释放，如图 3.2.7 所示，即添加了一个迁移。

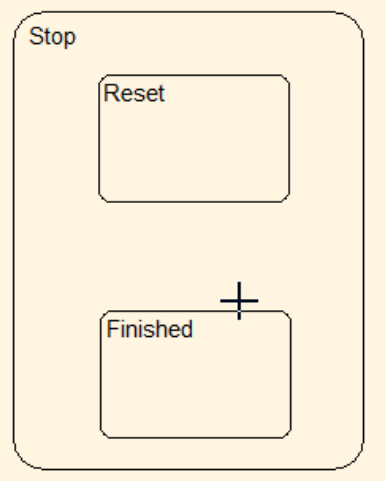

图 3.2.6 迁移起点

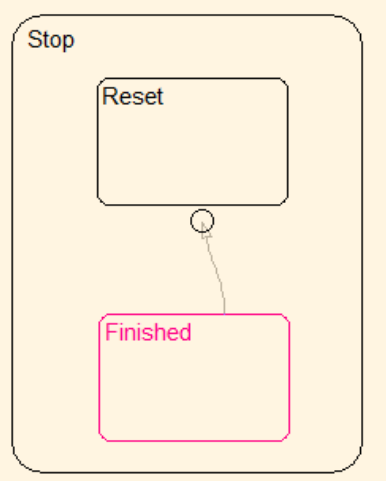

图 3.2.7 迁移终点

2. 添加默认迁移

单击  ，将鼠标移至默认状态矩形框的水平或垂直边缘，如下图 3.2.8。

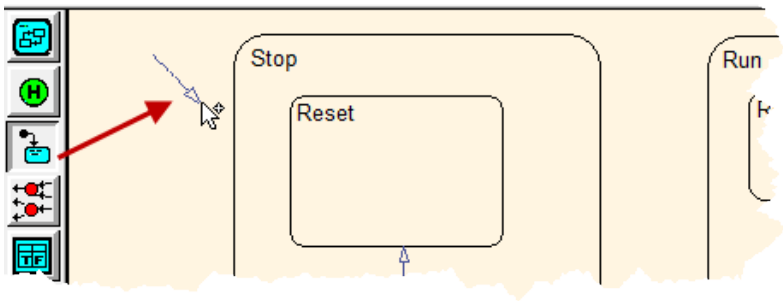

图 3.2.8 选择默认迁移

再次单击，即添加了一个默认迁移，如下图 3.2.9。

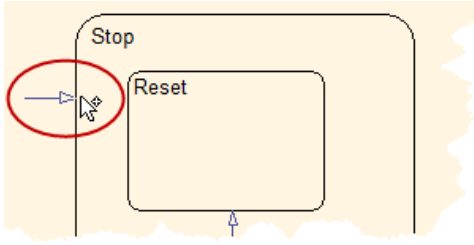

图 3.2.9 添加默认迁移

由于 Stop 是父状态，还需要针对其中的子状态，设置默认迁移，如下图 3.2.10，关于 Stateflow 的层次结构，详见 3.4 节。

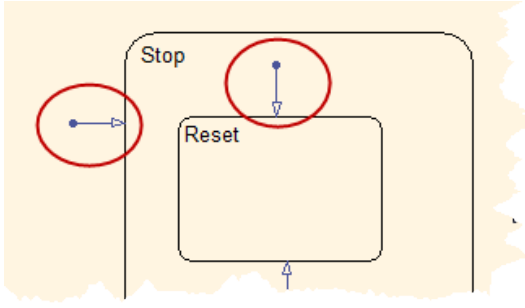

图 3.2.10 添加子状态默认迁移

3. 迁移变更

鼠标放置在迁移的起点或终点，当光标变成圆圈时〔图 3.2.11〕，按住鼠标左键，可将该端点移至其他状态，如图 3.2.12 所示。

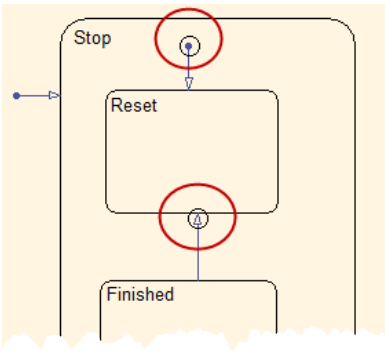

图 3.2.11　开始迁移变更

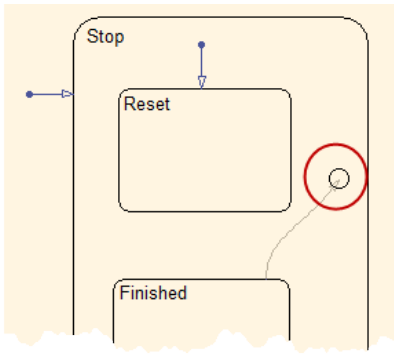

图 3.2.12 完成迁移变更

将默认迁移的起点移至某一状态，即转换为一般迁移；将一般迁移的起点悬空，即转化为默认迁移，如下图 3.2.13，假设迁移终点悬空，那么该迁移无效，如图 3.2.14 所示。

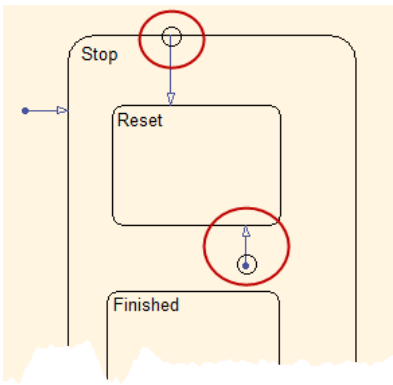

图 3.2.13 迁移起点变更

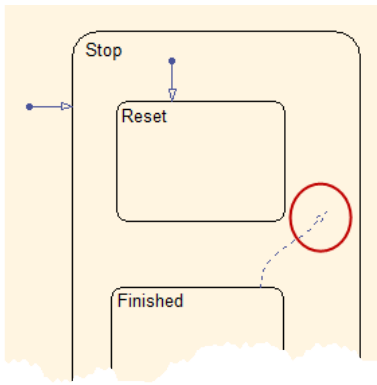

图 3.2.14 迁移终点变更

4. 迁移标签

新建的迁移标签不包含任何文字信息，用户单击迁移曲线一次，曲线上方显示“？”如下图 3.2.15：

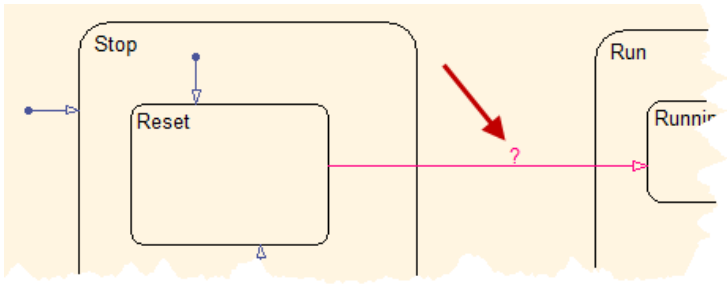

图 3.2.15 添加迁移标签

将鼠标移至“？”附近，再次单击，当显示编辑光标时，可编辑迁移标签，如下图 3.2.16。

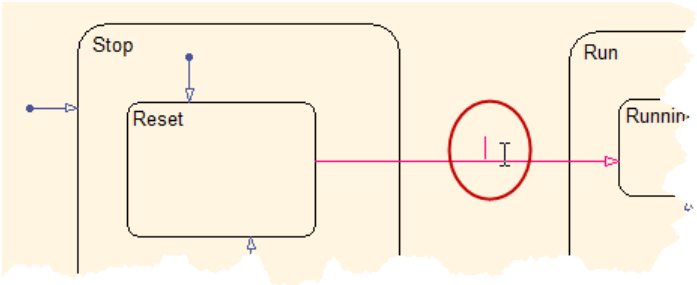

图 3.2.16 编辑迁移标签

完成编辑后，将鼠标放在标签的任意位置，按住左键并拖动，调整标签的位置，如下图 3.2.17。

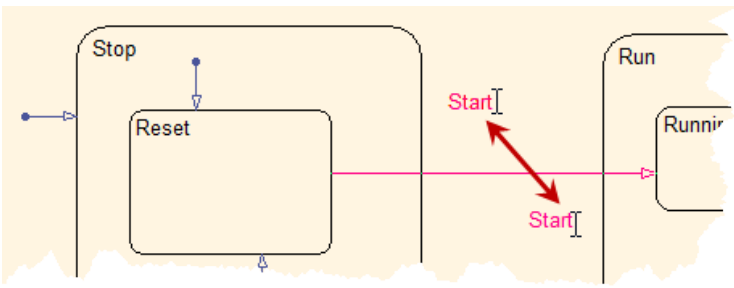

图 3.2.17 移动迁移标签

3.2.3 计时器状态图

根据以圈计时的特点，整个系统可分为 2 个父状态：停止与运行。停止状态包含 2 个子状态：计时器清零 Reset、计时器停止 Finished；运行状态也包含 2 个子状态：计时器运行 Running、以圈计时 LAP。如下图：

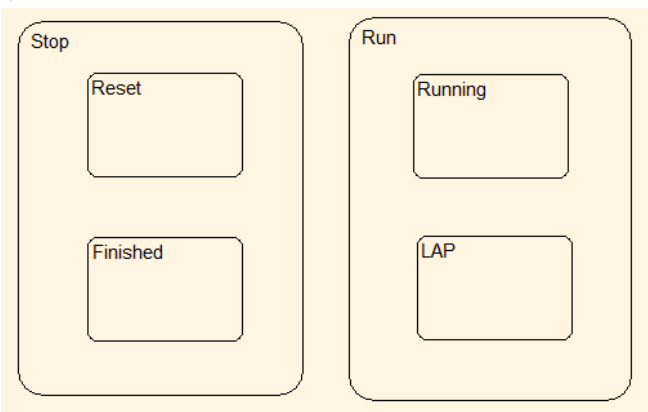

图 3.2.18 添加 4 个状态

再根据各状态之间的联系，添加默认迁移、迁移以及迁移标签，如下图 3.2.19。图中的迁移标签 start 表示按下 start 按钮这个事件，而 LAP 那么表示按下 LAP 按钮。

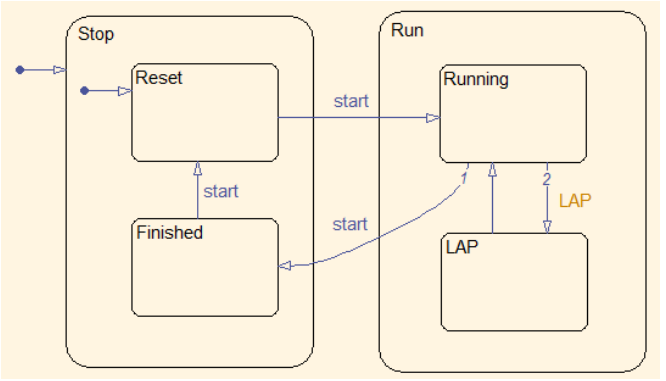

图 3.2.19 添加迁移及迁移标签

3.2.4 数据与事件

以圈计时需要 2 组数码管显示当前以及记录的分、秒、百分秒，另有两个按钮，为此需要添加 6 个数据与 2 个事件。数据是向外输出的，而事件是自外输入的。

添加数据或事件的方法有两种：使用菜单项 Add 或使用模型浏览器〔Model Explorer〕。前者的优势是添加方便，但菜单项仅提供了添加功能，无法通过菜单删除已添加的数据或事件，因此我们推荐用户使用模型浏览器。

为了照顾读者的不同需求，本文仍旧介绍两种添加方式。

1. 菜单项

在 Stateflow 编辑器窗口，选择菜单项 Add→Data→Output to Simulink，如下图 3.2.20。

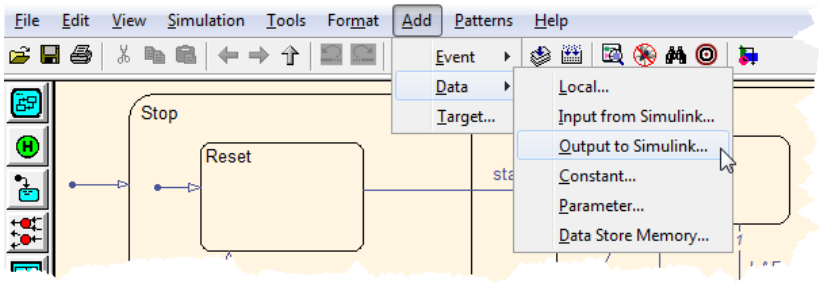

图 3.2.20 添加输出数据

在 Name 栏填入输出变量名 min，另外用户在 scope 栏还可以再次决定变量的作用域，如下图 3.2.21。

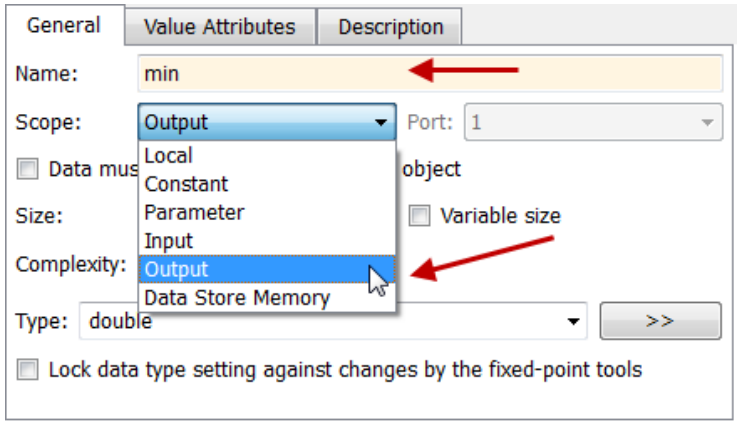

图 3.2.21 修改数据名及作用范围

2. 模型浏览器

在 Stateflow 状态图的顶层〔即不选中任何图形对象〕，选择菜单项 Tools→Explore，或直接按下 Stateflow 编辑器窗口的工具栏按钮，翻开模型浏览器，并确认已选中左侧模型结构图中的 Chart 节点，如下图 3.2.22。

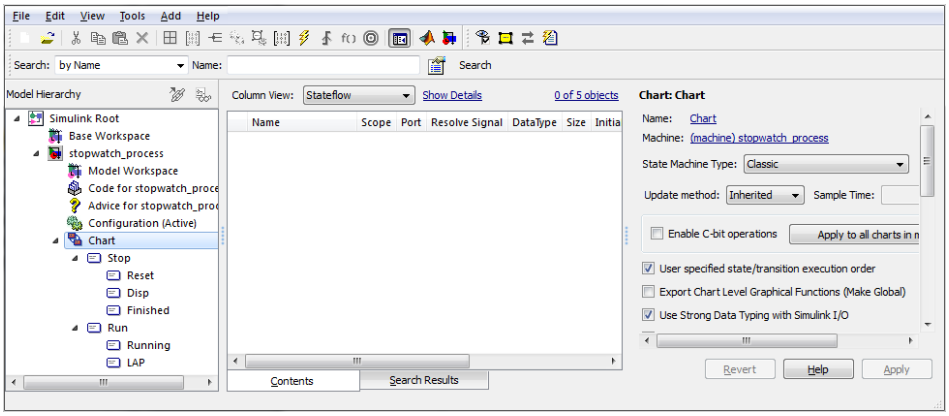

图 3.2.22 模型浏览器

在浏览器窗口的工具栏找到按钮、与，添加一个数据/事件或删除对应项。在中部窗口选中数据/事件的条目，右侧窗口即显示它的属性，如下图 3.2.23。与菜单项方法不同的是，使用浏览器添加的数据/事件，默认的作用域是本地〔Local〕，用户需要手动修改为外部输入或外部输出。

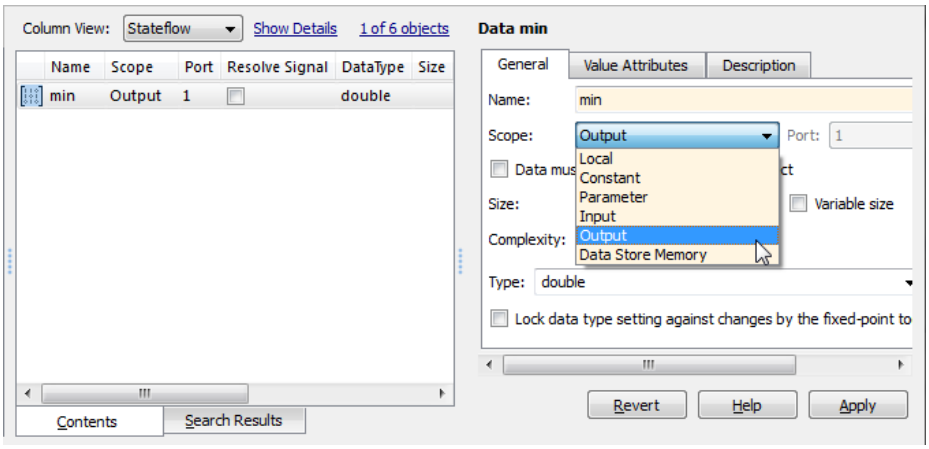

图 3.2.23 利用模型浏览器修改数据

对于事件，用户还需指定它的触发方式，Rising 表示上升沿、Failing 表示下降沿，而 Either 表示上升或下降沿皆可触发，本例的两个输入事件 start 与 LAP 皆选用 Either 方式触发，如下图 3.2.24。

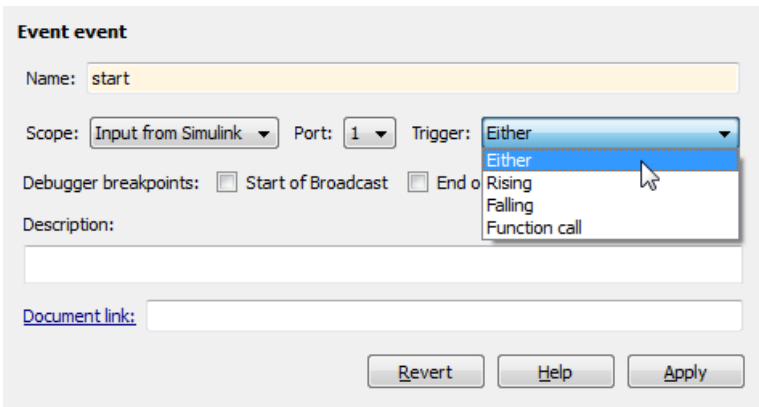

图 3.2.24 利用模型浏览器修改事件

有多个数据或事件时，用户还可以指定它们的端口号，合理地排列这些端口，将有利于以后的 Simulink 模块连线。如下图 3.2.25，是完整的数据与事件列表。

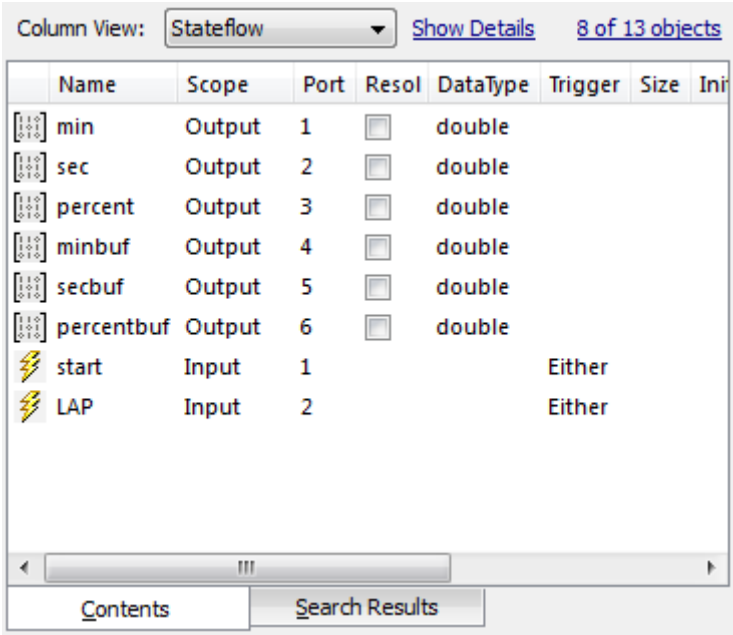

图 3.2.25 数据与事件列表

3.2.5 动作

上文提到，显示时间值可以定义为状态动作，也可以定义为迁移动作。为不失一般性，本小节分别说明这两种动作的定义方法。

计时器复位时，2 组数码管皆应清零，因此设置子状态 Reset 的进入动作为：

```
Reset
entry:
min=0;sec=0;percent=0;
minbuf=0;secbuf=0;percentbuf=0;
```

按下 LAP 按钮或再次按下 start 时，数码管 2 都必须显示当前时刻，因此设置子状态 LAP 与 Finished 的进入动作为：

```
Finished
entry:
minbuf = min;
secbuf = sec;
percentbuf = percent;
```

添加了动作的状态图如下图 3.2.26：

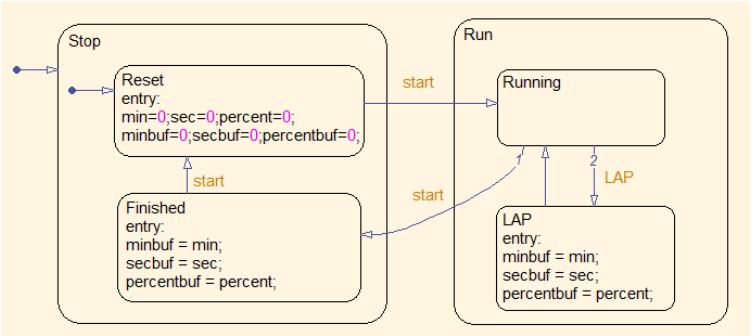

图 3.2.26 动作定义在状态

当然用户可以将显示时间值定义为迁移动作。为此按图 3.2.27 调整状态图，添加节点与迁移动作。对照图 3.2.26，用户应很容易理解图 3.2.27 的意义。

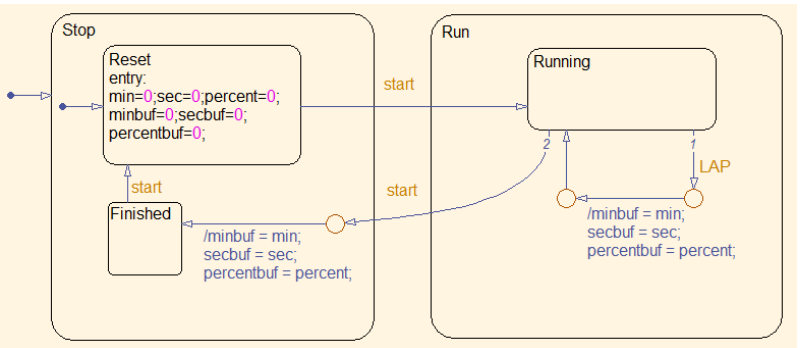

图 3.2.27 动作定义在迁移

不过目前的 Stateflow 状态图还缺少最关键的时钟程序，该过程详见 3.3 节。

3.2.6 自动创立对象

实际建模时，随着思路的不断扩展与成熟，状态、迁移、数据、事件、动作等等各种图形与非图形对象，总是交替着添加，很少按照上述过程逐步进行。这就难免发生遗漏，尤其是大型、多层次的 Stateflow 状态图。

当用户完成状态图的编辑，按下仿真按钮时，系统首先要进行语法检查，如果发现错误，那么给出提示。如果是某些数据或事件未定义，用户可使用随后出现的 Symbol Autocreation Wizard 向导，自动创立缺失的对象。

例如，用户可以在图 3.2.26 中，任意添加一个状态动作 y=1，任意添加一个迁移事件 stop，如图 3.2.28 所示。

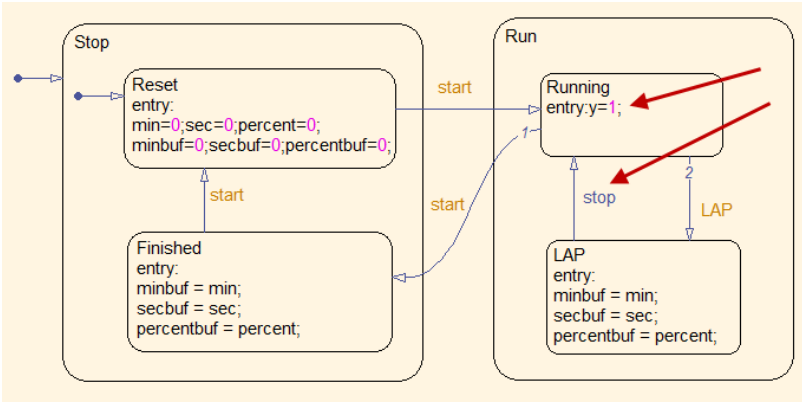

图 3.2.28 添加一个状态动作

按下仿真按钮，系统首先给出语法错误提示，如下图 3.2.29：

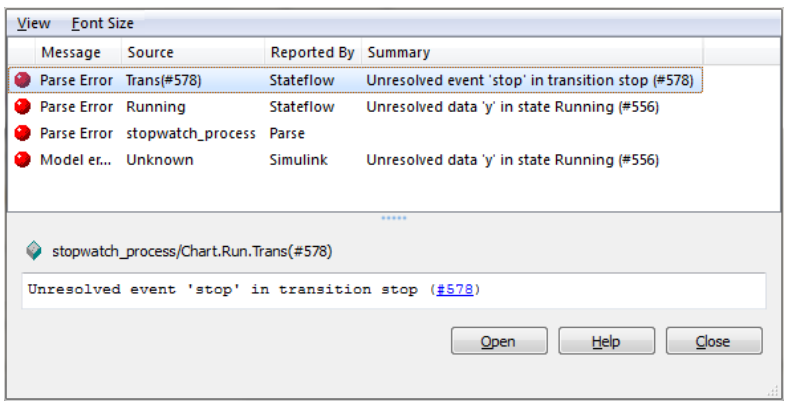

图 3.2.29 错误提示

系统进一步分析该语法错误，是由于状态图中存在无法处理的符号，因此弹出 Symbol Autocreation Wizard 向导与向导使用说明，如下图 3.2.30，假设用户已能够熟练使用，可径自忽略该说明。

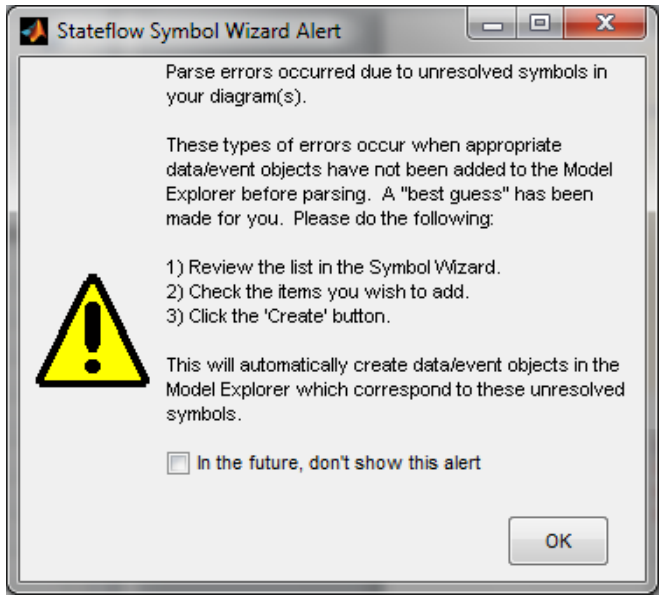

图 3.2.30 向导使用说明

Symbol Autocreation Wizard 向导建议，应另行创立数据 y 与事件 stop，如下图 3.2.31。

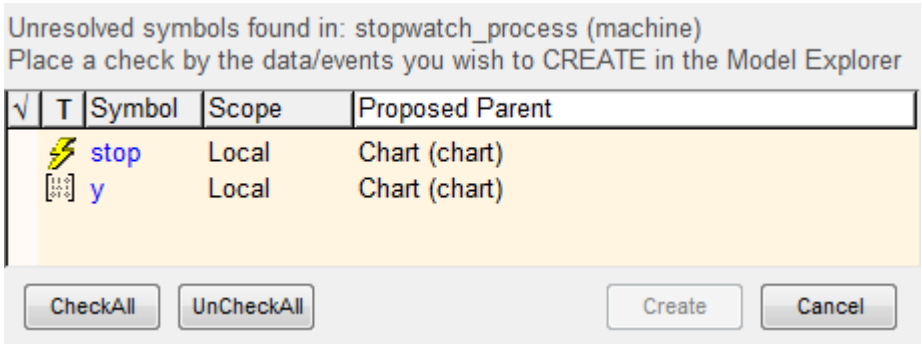

图 3.2.31 自动创立对象向导

用户可以不断单击 Scpoe 与 Proposed Parent 的内容，修改数据/事件的作用域与父对象，如下图 3.2.32。

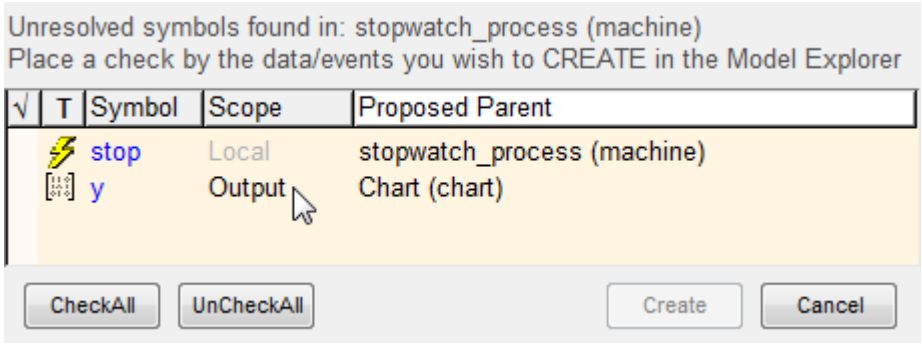

图 3.2.32 修改作用域与父对象

用户假设接受向导的建议，那么单击数据/事件前的空白处，选中该条目，之后按下 Create，创立对象，如下图 3.2.33。

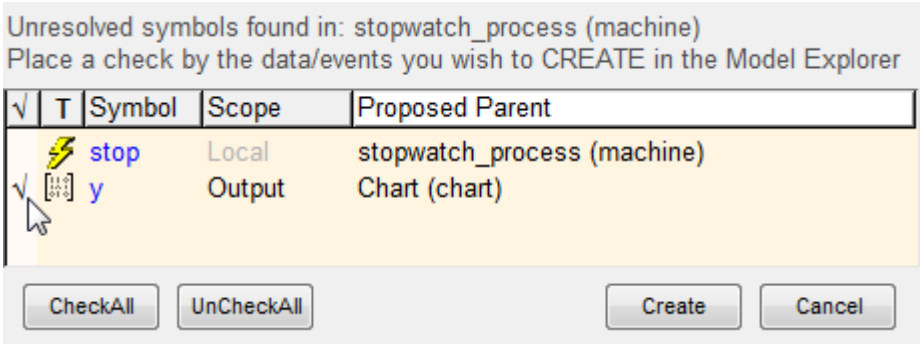

图 3.2.33 创立对象

3.3 Stateflow 流程图

3.3.1 流程图与节点

3.1 与 3.2 节介绍了 Stateflow 状态图的根本概念与创立过程。状态图的一个特点是，在进入下一个仿真步长前，它会记录下当前的本地数据与各状态的激活情况，供下一步长使用。而流程图只是一种使用节点与迁移来表示条件、循环、多路选择等逻辑的图形，它不包含任何的状态。

由于迁移〔除了默认迁移〕总是从一个状态到另一个状态，节点之间的迁移只能是一个迁移段。因此流程图可以看作是有假设干个中间支路的一个迁移，一旦开始执行，就必须执行到终节点〔没有任何输出迁移的节点〕，不能停留在某个中间节点，也就是说必须完成一次完整的迁移。

从另一个角度来看，节点可以认为是系统的一个判决点或集合点，它将一个完整的迁移分成了假设干个迁移段。因此可以将几个相同的迁移段合并为一个，用一个迁移表示多个可能发生的迁移，简化状态图，由此生成的代码也更加有效

对于以下情况，用户应首先考虑使用节点：

- if-else 判断结构、自循环结构、for 循环结构；
- 单源状态到多目标状态的迁移；
- 多源状态到单目标状态的迁移；
- 基于同一事件的迁移；

注意：事件无法触发从节点到状态的迁移。

建议：用户可以把流程图封装成一个图形函数〔详见 3.6.2 节〕，便于在 Stateflow 的任意位置调用。

3.3.2 建立流程图

1. 手动建立

建立流程图的过程与建立状态图的过程相似，以一段简单的代码为例，手动建立流程图。

```
if percent==100
    {percent=0;
    sec=sec+1;}
else if sec==60
       {sec=0;
       min=min+1;}
    end
end
```

① 起始节点

单击  按钮，添加起始节点。如下图 3.3.1：

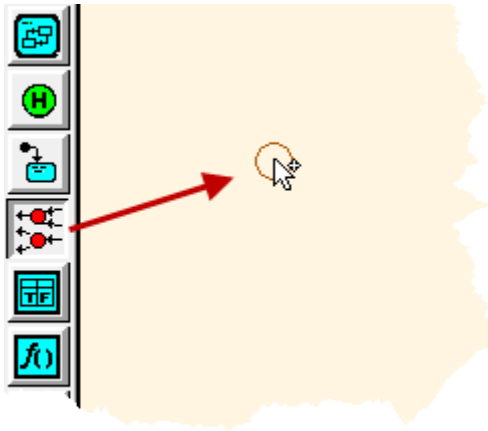

图 3.3.1 添加节点

② 条件节点与终节点

根据代码的执行过程，逐一添加条件节点 A1、B1、C1，终节点 A2、B2，以及节点间的迁移与迁移标签，如下图 3.3.2。

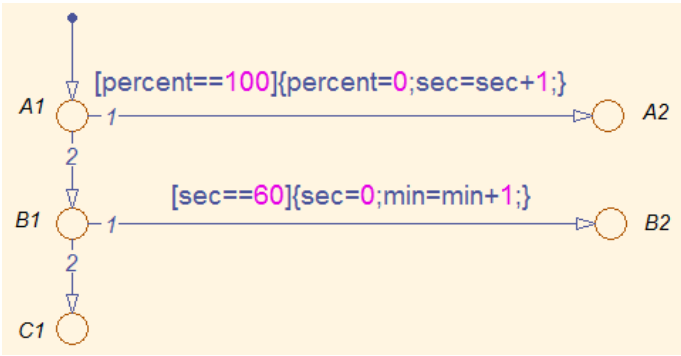

图 3.3.2 流程图

流程图运行过程如下：

1. 系统默认迁移进入节点 A1，如果条件[percent==100]为真，执行{percent=0;sec=sec+1;}，并向终节点 A2 迁移；

2. 如果条件[percent==100]不为真，向 B1 节点迁移，继续判断如果条件[sec==60]为真，执行{sec=0;min=min+1;}，并向终节点 B2 迁移；

3. 如果不满足任何条件，那么向终节点 C1 迁移。

③ 节点与箭头大小

对于某些重要的节点或迁移，用户可以调整其节点大小与迁移箭头的大小，突出其地位。例如，选择节点 C1 的右键菜单项 Junction Size→16，如下图 3.3.3，放大节点；选择节点 A1 的右键菜单项 Arrowhead Size → 20，放大指向该节点的所有迁移箭头，如图 3.3.4 所示。

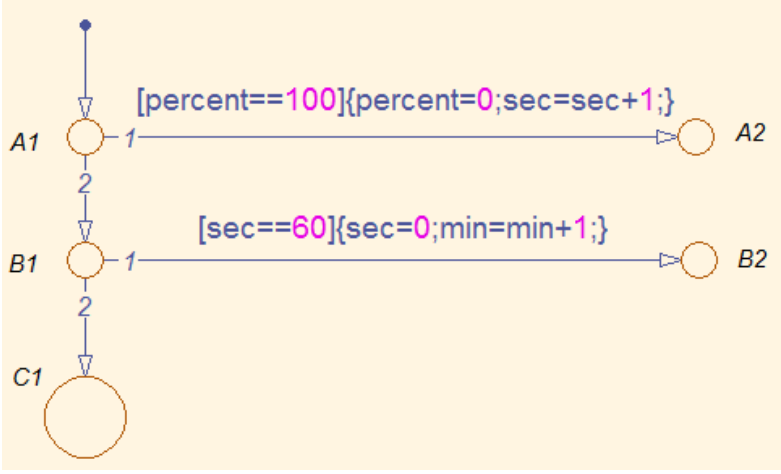

图 3.3.3 节点大小

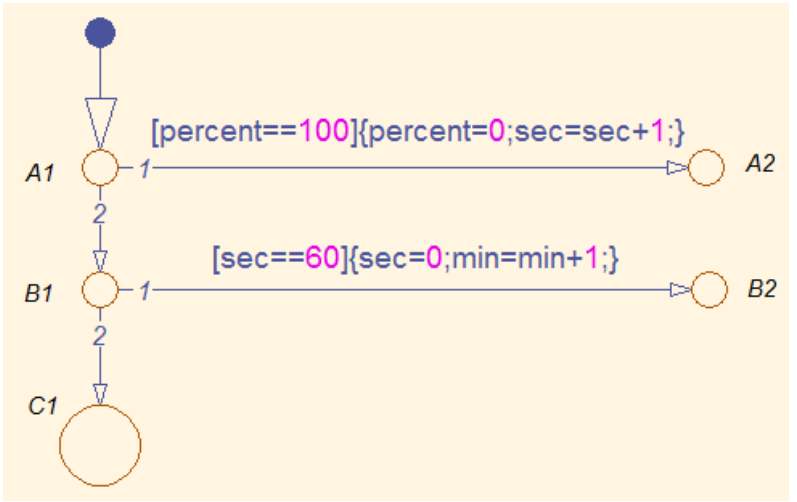

图 3.3.4 箭头大小

④ 优先级

两个判断节点 A1、B1，均有两条输出迁移，分别标记了数字 1、2，这表示迁移的优先级。默认情况下，Stateflow 状态图使用显性优先级模式，用户可以自行修改各个迁移优先级。

例如，选择迁移曲线的右键菜单项 Execution Order，将优先级由 1 降低为 2，如下图 3.3.5。修改了某一输出迁移的优先级，系统会自动调整同一节点另一迁移的优先级。

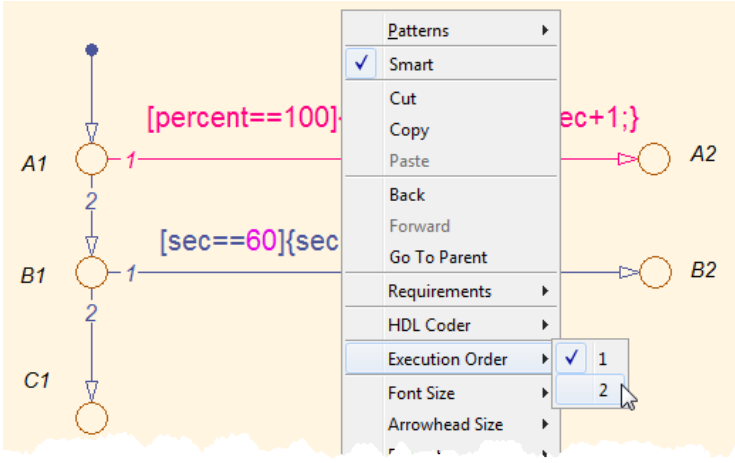

图 3.3.5 迁移优先级

为防止用户错误地设置优先级，Stateflow 提供了另一种模式：隐性优先级。选择编辑器菜单项 File → Chart Properties，取消 User specified state/transition execution order 前的复选框，启用隐性模式，如下图 3.3.6。

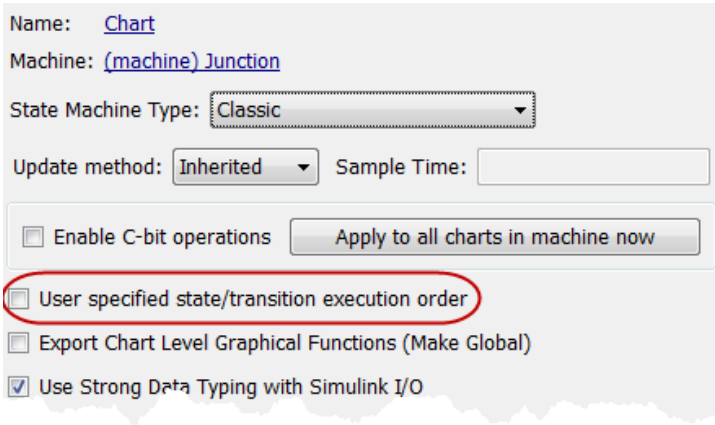

图 3.3.6 自动设置迁移优先级

使用这种模式时，系统根据以下规那么，自动设置迁移优先级，从高到低排列为：

1. 既有事件又有条件的迁移
2. 仅有事件的迁移
3. 仅有条件的迁移
4. 不含任何限制的迁移

注意：同一个 Stateflow 状态图，只能选用一种优先级模式，但对于有多个状态图的 Simulink 模型，那么不受此限制。

2. 自动建立

对于简单的流程图，手动建立难度不大，而对于稍复杂的逻辑，用户难免会感到无从下手。Stateflow 提供了快速建立流程图的向导，它可以生成 3 类根本逻辑：判断、循环、多条件。本小节使用向导，重建图 x 的流程图。

① 单击编辑器菜单项 Patterns→Add ...，选择流程图的类型，如下图 3.3.7。

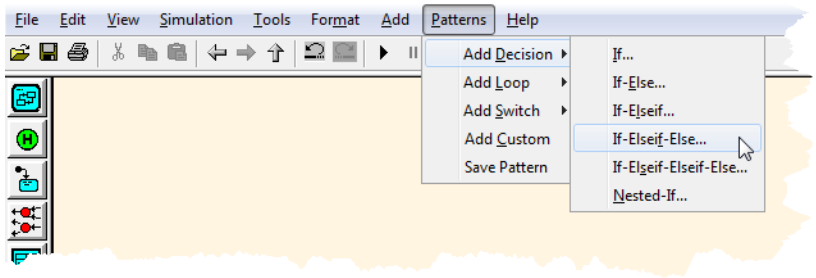

图 3.3.7 流程图向导菜单

② 这里选择 Patterns → Add Decision → If-Elseif-Else...，在随后翻开的对话框中输入判断条件与对应的动作，如下图 3.3.8。

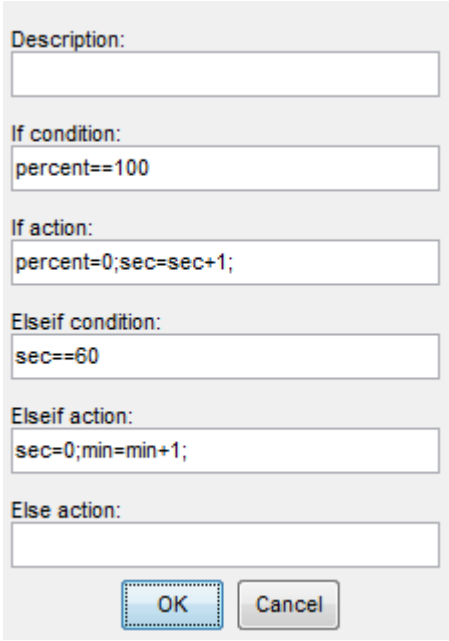

图 3.3.8 新建流程图对话框

① 生成的流程图如下图 3.3.9：

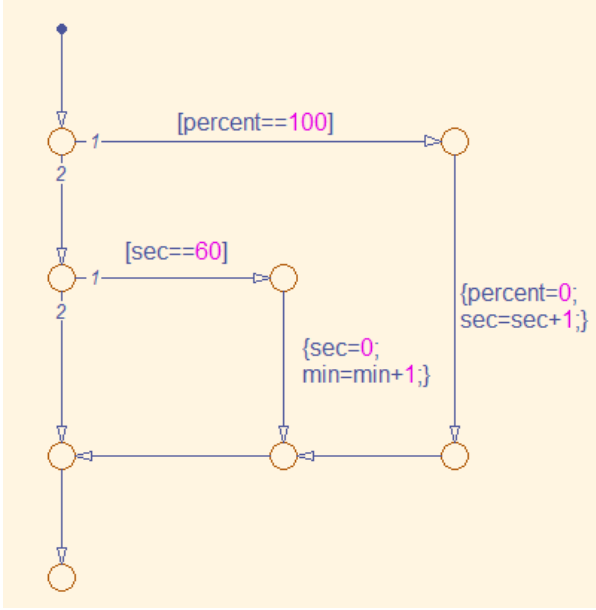

图 3.3.9 流程图

3. 两种方式的比照

尽管用户可以手动建立流程图，但使用流程图向导的优势也是显而易见的：

- 任何一种流程图都可归结为判断、循环、多条件，或者三者的组合，因此皆可以使用向导自动生成；
- 使用向导生成的流程图符合 MAAB〔MathWorksAutomotive Advisory Board

〕规那么，这有利于后期模型检查；

- 各种流程图的外观根本一致；
- 将设计好的流程图，另存为模板，便于重用。

3.4 层次结构

层次的概念

Stateflow 的对象具有层次性，一个 Stateflow 对象可以包含其他 Stateflow 对象，例如状态内假设包含其他状态，那么形成父状态，其内部状态称为子状态。当状态具有第二个层次时，状态就构成了层次。

状态具有了层次，迁移自然也具有了层次，Stateflow 允许在不同层次状态之间存在转移。如果迁移穿越了父状态的边界直接到达了低层次的子状态，那么被称之为超迁移。

在状态图中使用层次有如下几个目的：

- 使用层次，可以将相关的对象组合在一起，构成族群；
- 可以将一些通用的迁移路径或者动作组合成为一个迁移动作或路径，简化模型；
- 适当地使用层次，可以有效地缩减生成代码的大小，也能够提高程序执行的效率和可读性。

3.4.2 迁移的层次

1 内部迁移

内部迁移是指从父状态边缘内部出发，终止于子状态外边缘的迁移，迁移始终处于父状态的内部，不会退出源状态。

在交通灯系统中，同一个父状态 PowerOn 存在红黄绿三个子状态，它们需要不停地转换，但除非发生 PowerOff 事件，不会退出父状态，对于这样的逻辑过程，读者可能习惯使用节点将三种状态的迁移联系起来，如下图。

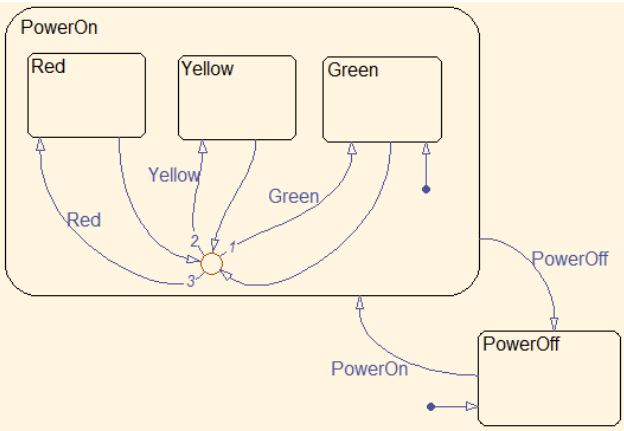

图 3.4.1 带有节点的迁移

使用了内部迁移，可直接从父状态激活相应的子状态，不必经过节点，大大简化状态图，如下图 3.4.2。

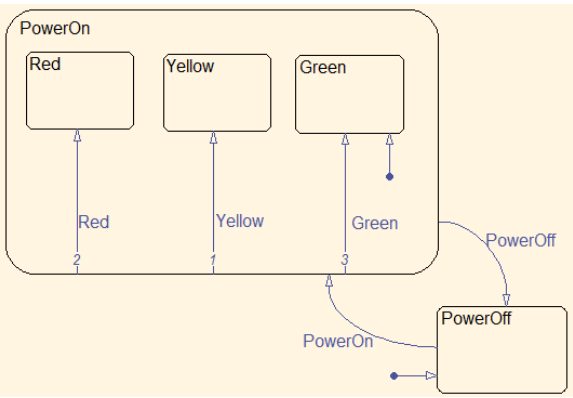

图 3.4.2 内部迁移状态图

2 层次化迁移的优先级

与状态类似，迁移也具有层次性，迁移所属的层次是由其父状态、源状态和目标状态决定的。因此，当多个迁移同时有效时，Stateflow 需要有一个层次化迁移优先级机制来判断迁移顺序。

层次化迁移的优先级规那么为：从高层次到低层次检测；从外部迁移到内部迁移检测；同一层次内，超转移优先。

如下图 3.4.3，图表激活时，默认迁移激活状态 A，继而状态 A.a1 被次级默认迁移激活，这时按以下优先级检测迁移是否有效：

① 检测高层 A,B 状态的外部迁移是否有效〔event1，event2〕；

② 检测高层 A,B 状态的内部迁移是否有效〔event3〕；

③ 检测低层 a1,a2,b1 状态的超迁移是否有效〔event4〕；

④ 检测父状态 A 内部子状态间的迁移是否有效〔event5，event6〕。

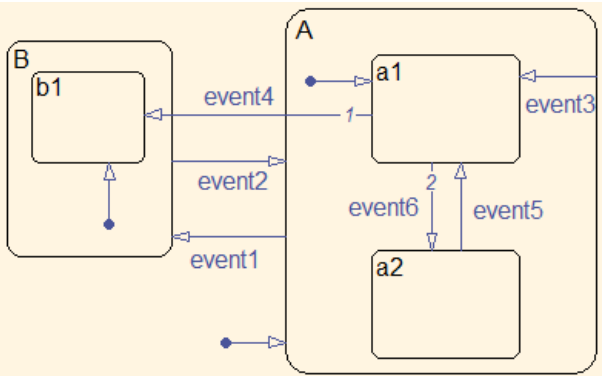

图 层次化的迁移

3.4.3 历史节点

在状态图的顶层或一个父状态里或放置一个历史节点，它便能能记录退出父状态时，正处于激活状态的子状态，当再次进入父状态时，那么默认激活上一次所记录的子状态，而不是激活默认迁移的状态。

历史节点的作用域仅限于它所存在的层级。

如图，父状态 A2 中参加了历史节点，因此当第 1 次激活 A2 状态时子状态 C1 被激活，满足迁移条件时 C2 被激活，但此后 A2 状态向 A1 状态的迁移将优先发生，C2 状态不再向 C1 状态迁移。于是第 1 次激活 A2 状态时，被激活的子状态是 C2，而不是 C1。读者可以从图所示的输出看出上述的迁移过程。

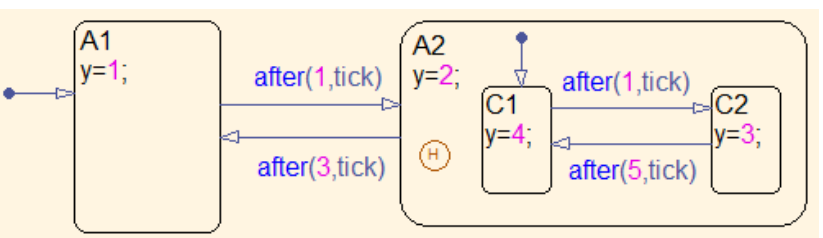

图 历史节点