

# InTouch<sup>®</sup> HMI 辅助组件指南

**Invensys Systems, Inc.**

修订版 A

最后修订日期：2007 年 8 月 6 日



## 版权声明

© 2007 Invensys Systems, Inc. 版权所有。保留所有权利。

保留所有权利。未经 Invensys Systems, Inc. 事先书面明确同意，不得通过任何手段（电子、机械、影印、录制或其它方式）复制、传输本文档中的任何部分，或是将其存储到检索系统。使用本文档所含信息不需承担任何相关的版权或专利责任。虽然在编制本文档的过程中已采取一切预防措施，但错误或疏漏在所难免，出版商与作者对此概不承担任何责任。对由于使用本文档所含信息而导致的任何损害，亦不承担任何赔偿责任。

本文档中的内容如有变更，恕不另行通知，这些内容亦不代表 Invensys Systems, Inc. 一方的承诺。本文所述软件系在遵守许可协议或保密协议的前提下提供。本软件的使用或复制必须遵守这些协议中的各项条款。

Invensys Systems, Inc.  
26561 Rancho Parkway South  
Lake Forest, CA 92630 U.S.A.  
(949) 727-3200

<http://www.wonderware.com>

对产品文档如有任何意见或建议，请发送电子邮件到 [productdocs@wonderware.com](mailto:productdocs@wonderware.com)。

## 商标

本文所提及且已知为商标或服务标志的所有专用名词均已采用适当的首字母大写形式。Invensys Systems, Inc. 无法证实此类信息的准确性。在本文档中使用某个专用名词不应视为会影响任何商标或服务标志的有效性。

Alarm Logger、ActiveFactory、ArchestrA、Avantis、DBDump、DBLoad、DT Analyst、FactoryFocus、FactoryOffice、FactorySuite、FactorySuite A<sup>2</sup>、InBatch、InControl、IndustrialRAD、IndustrialSQL Server、InTouch、MaintenanceSuite、MuniSuite、QI Analyst、SCADAAlarm、SCADASuite、SuiteLink、SuiteVoyager、WindowMaker、WindowViewer、Wonderware 以及 Wonderware Logger 均为 Invensys plc 及其子公司与附属公司的商标。所有其它品牌可能是其相应所有者的商标。

# 目录

欢迎.....	9
文档惯例.....	9
技术支持.....	10
第 1 章 使用 SPCPro 统计过程控制监控生产质量.....	11
配置 SPCPro 数据库.....	12
配置本地 Microsoft Access 数据库.....	13
通过压缩数据库来节省磁盘空间.....	14
配置分布式 SQL Server 数据库.....	15
配置一组产品的采集与分析选项.....	17
配置产品与关联的过程限制.....	19
使用基于时间或事件的采集方法自动采集 SPC 数据.....	21
配置 SPC 数据库用户.....	22
SPCConnect() 函数.....	23
SPCDisconnect() 函数.....	24
更改数据采集间隔时间.....	24
使用 SPCPro 图表向导显示与手工采集 SPC 数据.....	25
使用 SPC 控制图向导.....	25
使用“SPC 直方图”向导.....	32
使用间接数据集动态更改显示的产品.....	34
分析失控情况的原因.....	35
配置特殊原因代码.....	35

使用“巴累托图向导”以图形化方式 显示特殊原因的分布 .....	36
在运行时动态更改极限值 .....	38
处理 SPCPro 报警 .....	40
配置 SPCPro 报警 .....	40
显示与确认 SPCPro 报警 .....	41
输入修正操作 .....	43
使用 DDE 项目与脚本函数进行运行时控制 .....	45
在运行时添加或删除数据集 .....	55
SPCDatasetDlg() 函数 .....	55
更改图表使用的数据集 .....	55
SPCSelectDataset() 函数 .....	56
更改图表中显示的产品 .....	56
SPCSelectProduct() 函数 .....	57
SPCSetProductDisplayed() 函数 .....	57
滚动图表 .....	58
SPCDisplayData() 函数 .....	59
SPCLocateScooter() 函数 .....	59
SPCMoveScooter() 函数 .....	60
将图表更新到当前时间 .....	60
输入新样本 .....	61
SPCSetMeasurement() 函数 .....	61
SPCSaveSample() 函数 .....	62
检查删除的样本 .....	62
SPCSampleDeleted() 函数 .....	62
更改当前采集的产品 .....	63
SPCSetProductCollected() 函数 .....	64
在运行时创建或删除产品 .....	65
NewProduct DDE 项目 .....	65
SPCCreateNewProduct() 函数 .....	66
SPCDeleteProduct() 函数 .....	66
为各种图表类型输入计算参数 .....	67
SPCSetControlLimits() 函数 .....	67
SPCSetRangeLimits() 函数 .....	68
SPCSetSpecLimits() 函数 .....	68
SPCSetProductControlLimits() 函数 .....	69
SPCSetProductRangeLimits() 函数 .....	69
SPCSetProductSpecLimits() 函数 .....	70
将数据输入属性图表 .....	71
控制配置选项 .....	71
从以前的 SPCPro 版本移植 SPC 数据库 .....	72

## 第 2 章 使用 Recipe Manager ..... 75

Recipe Manager 综述 .....	76
Recipe Manager 实用程序 .....	76
配方模板文件 .....	77
模板定义 .....	77
单元定义 .....	77
配方定义 .....	77
在 Recipe Manager 中编辑配方数据 .....	78
配置 Recipe Manager 编辑网格 .....	78
使用编辑网格 .....	79
定义成分名与数据类型 .....	83
将 InTouch 标记映射到成分 .....	84
为不同配方中的成分定义值 .....	85
在其它应用程序中编辑配方数据 .....	87
使用 Excel 处理配方模板文件 .....	87
使用“记事本”处理配方模板文件 .....	88
通过嵌套配方来创建复杂的结构 .....	89
在 InTouch 中使用配方 .....	90
在配方文件中加载与保存配方数据 .....	91
RecipeLoad() 函数 .....	91
RecipeSave() 函数 .....	92
从配方文件中删除配方 .....	93
RecipeDelete() 函数 .....	93
选择单元（标记成分映射） .....	93
RecipeSelectUnit() 函数 .....	93
从配方文件中选择单独的配方 .....	94
RecipeSelectRecipe() 函数 .....	94
RecipeSelectNextRecipe() 函数 .....	95
RecipeSelectPreviousRecipe() 函数 .....	96
理解配方脚本函数返回的错误消息 .....	97
显示错误码消息 .....	97
RecipeGetMessage() 函数 .....	99
设置配方安全性 .....	100

## 第 3 章 从 InTouch 中使用 SQL 数据库 ..... 101

设置数据源 .....	103
将 InTouch 标记映射到数据库列 .....	104
配置“绑定列表”中的 SQL Server 字符串分隔符 .....	106
定义新表的结构 .....	107

使用数据库应用程序 .....	110
SQL Server 数据库应用程序 .....	110
Microsoft Access 数据库应用程序 .....	111
Oracle 数据库应用程序 .....	112
在 InTouch 中执行常见的 SQL 操作 .....	113
连接与断开数据库 .....	118
SQLConnect() 函数 .....	118
SQLDisconnect() 函数 .....	119
创建新表 .....	119
SQLCreateTable() 函数 .....	119
删除表 .....	120
SQLDropTable() 函数 .....	120
从表中检索数据 .....	121
SQLSelect() 函数 .....	122
SQLGetRecord() 函数 .....	125
SQLNumRows() 函数 .....	125
SQLFirst() 函数 .....	126
SQLNext() 函数 .....	126
SQLPrev() 函数 .....	127
SQLLast() 函数 .....	127
SQLEnd() 函数 .....	128
向表中写入新记录 .....	128
SQLInsert() 函数 .....	129
SQLInsertPrepare() 函数 .....	129
SQLInsertExecute() 函数 .....	130
SQLInsertEnd() 函数 .....	131
更新表中现有的记录 .....	132
SQLUpdate() 函数 .....	132
SQLUpdateCurrent() 函数 .....	133
从表中删除记录 .....	134
SQLClearTable() 函数 .....	134
SQLDelete() 函数 .....	135
执行参数化语句 .....	136
SQLSetStatement() 函数 .....	136
SQLAppendStatement() 函数 .....	137
创建语句或从文件中加载现有的语句 .....	138
SQLLoadStatement() 函数 .....	138
预备语句 .....	139
SQLPrepareStatement() 函数 .....	140
设置语句参数 .....	140
SQLSetParamChar() 函数 .....	140
SQLSetParamDate() 函数 .....	141

SQLSetParamDateTime() 函数.....	141
SQLSetParamDecimal() 函数.....	142
SQLSetParamFloat() 函数.....	143
SQLSetParamInt() 函数.....	143
SQLSetParamLong() 函数.....	144
SQLSetParamNull() 函数.....	144
SQLSetParamTime() 函数.....	146
清除语句参数.....	147
SQLClearParam() 函数.....	147
执行语句.....	147
SQLExecute() 函数.....	147
释放占用的资源.....	150
SQLClearStatement() 函数.....	150
使用事务集.....	151
SQLTransact() 函数.....	151
SQLCommit() 函数.....	152
SQLRollback() 函数.....	153
在运行时打开 ODBC 管理器对话框.....	154
SQLManageDSN() 函数.....	154
理解 SQL 错误消息.....	155
SQLErrorMsg() 函数.....	155
保留字列表.....	159

## 第 4 章 使用 16-Pen Trend 向导..... 163

创建 16-Pen Trend.....	164
配置要在趋势图上显示的标记.....	165
配置趋势的时间跨度与更新速率.....	167
配置趋势的显示选项.....	168
在运行时更改趋势的配置.....	169
使用脚本控制 16-Pen Trend（16 笔趋势）向导.....	170
ptGetTrendType() 函数.....	170
ptLoadTrendCfg() 函数.....	170
ptPanCurrentPen() 函数.....	171
ptPanTime() 函数.....	172
ptPauseTrend() 函数.....	173
ptSaveTrendCfg() 函数.....	173
ptSetCurrentPen() 函数.....	174
ptSetPen() 函数.....	174
ptSetPenEx() 函数.....	175
ptSetTimeAxis() 函数.....	176

ptSetTimeAxisToCurrent() 函数 .....	176
ptSetTrend() 函数 .....	177
ptSetTrendType() 函数 .....	177
ptZoomCurrentPen() 函数 .....	178
ptZoomTime() 函数 .....	179

## 第 5 章 Symbol Factory ..... 181

符号类型 .....	181
图片向导 .....	182
位图向导 .....	182
纹理向导 .....	182
InTouch 对象 .....	183
使用 Symbol Factory .....	183
快速入门 .....	183
在窗口中放置 Symbol Factory 向导 .....	184
配置符号选项 .....	185
给向导设置动画效果 .....	187
编辑符号 .....	188
分解向导以便编辑 .....	189
在网络上共享一个类别的符号 .....	189
使类别只读 .....	189
查看类别属性 .....	190
编辑现有的类别 .....	191
删除类别 .....	191
配置 Symbol Factory .....	192
疑难排解 .....	193

## 索引..... 195



# 欢迎

InTouch 辅助组件包括：

- SPCPro，用于监控与测量生产质量的一套统计分析工具。
- Recipe Manager，用于管理生产配方的工具。
- “SQL 访问管理器”，用于访问、修改、创建及删除数据库表格的工具。
- “16- 笔趋势”，用于创建实时与历史趋势的向导，这些趋势可以显示最多 16 个标记的数据。
- Symbol Factory，4000 多个工业自动化向导的集合。

本文假设您了解如何使用 Microsoft Windows，包括浏览菜单、在应用程序之间切换，以及在屏幕上移动对象。如需有关完成这些任务的帮助，请参阅 Microsoft 文档。

## 文档惯例

本文采用以下惯例：

惯例	用于
首字母大写	路径与文件名。
<b>粗体</b>	菜单、命令、对话框名称以及对话框选项。
等宽字体	代码范例与显示文本。

## 技术支持

Wonderware 的“技术支持”部门提供多种技术支持方案，帮助解答有关 Wonderware 产品及其实施方案的任何疑问。

在与“技术支持”部门联系之前，请参阅本文中相关的章节，以寻求问题的可能解决方案。如果需要联系技术支持以获取帮助，请准备好以下信息：

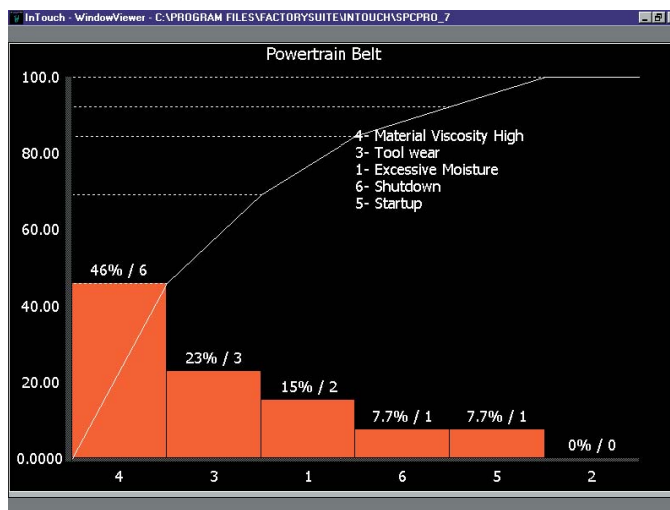
- 使用的操作系统的类型与版本。
- 有关如何重现问题的详细说明。
- 看到的错误消息的准确内容。
- Log Viewer 或任何其它诊断应用程序提供的任何相关输出列表。
- 为解决问题所作的尝试及其结果的详细说明。
- 如果遇到仍然存在的已知问题，请提供指定给该问题的“Wonderware 技术支持”案例号。

# 第 1 章

## 使用 SPCPro 统计过程控制监控生产质量

SPCPro™ 是一个“统计过程控制”（Statistical Process Control，简称 SPC）实用程序，它随 InTouch HMI 一起提供。SPC 是一种监视与控制过程的方法，它收集有关输出特征的数据、分析这些数据，并根据这些数据作出过程管理决策。

SPCPro 分析工具可以监视与测量生产质量。使用 SPCPro 不需要扎实的统计知识。SPCPro 提供图形化的工具与报警确认方式，非常易于使用和理解。



通过 SPCPro 可以：

- 使用统计方法与针对违反规则的报警来实时监视过程质量。
- 获取关于过程的即时反馈，以快速确定各种情况并采取有效的修正操作。
- 访问 SPCPro 数据以进行额外的过程后分析或报告。

Wonderware 还提供了 QI Analyst 产品，它是比 SPCPro 更为灵活的统计解决方案。

SPCPro 独立于 InTouch HMI 进行授权。

## 配置 SPCPro 数据库

在可以使用 SPCPro 之前，必须选择一个数据库以存储 SPC 配置与采集数据。SPCPro 可以同 Microsoft Access 或 Microsoft SQL Server 数据库配合使用。

可以使用哪一种数据库取决于所要创建的 SPCPro 应用程序的类型。

SPCPro 节点类型	支持的数据库
单节点	Microsoft Access 或 SQL Server
多节点	Microsoft SQL Server

在使用 SPCPro 之前，需要先对它进行配置。要使用 SPCPro，必须先安装好 Microsoft ODBC 驱动程序。SPCPro 支持以下 Microsoft ODBC 驱动程序：

- Microsoft Access 驱动程序 4.00.3711.08 或更高版本
- Microsoft SQL Server 3.70.06.23 或更高版本

**重要** 在运行 InTouch 应用程序之前，必须设置新的 SPCPro 数据库，并导入使用较早版本 SPCPro 创建的任何数据集。

## 配置本地 Microsoft Access 数据库

使用 SPCPro 数据库实用程序为单节点应用程序配置 Access 数据库。

**备注** 通过单击**配置 SPC 数据库**对话框中的**修改**按钮来编辑现有的数据库配置时，**ODBC Microsoft Access 安装**对话框也会出现。

### 要配置单节点数据库

- 1 启动 WindowMaker。
- 2 在**特别**菜单上，指向 **SPC**，然后单击**数据库**。此时出现**配置 SPC 数据库**对话框。



**备注** 要关闭“配置 SPC 数据库”而不配置数据库，请按 **ESCAPE** 键。

- 3 在**数据库类型**列表中，单击 **Microsoft Access (Local)**。
- 4 配置 ODBC 数据源。执行以下操作：
  - a 在 **ODBC 数据源**列表中，单击 **< 新 >**。此时出现 **ODBC 数据源管理器**对话框。
  - b 单击**用户 DSN**选项卡，然后从列表中选择 ODBC 数据源，或单击**添加**。此时出现**创建新数据源**对话框。
  - c 从列表中选择 ODBC 驱动程序，然后单击**完成**。此时出现 **ODBC Microsoft Access 安装**对话框。
  - d 在**数据源名**框中，为数据源输入唯一的名称。
  - e 单击**创建**。此时出现**新建数据库**对话框。
  - f 在**数据库名**框中，输入新数据库的名称。
  - g 选择用于存储新数据库的文件夹，然后单击**确定**。出现消息时，单击**确定**。此时再次出现 **ODBC Microsoft Access 安装**对话框。
  - h 选择新创建的数据源。
- 5 单击**确定**。此时再次出现**配置 SPC 数据库**对话框。

- 6 单击**保存**。出现消息时，单击**是**以初始化数据库。出现状态消息时，单击**确定**。
- 7 单击**验证**。如果 ODBC 连接有效，**连接状态**下出现绿灯。
- 8 单击**确定**。

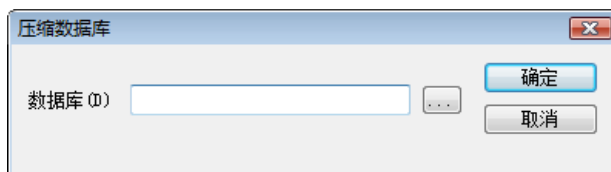
### 通过压缩数据库来节省磁盘空间

spcutil 实用程序可以管理 Access 数据库的大小。Access 数据库增长迅速，应该定期压缩。压缩数据库之后，可以看到数据库文件的大小显著减少。

在开始压缩之前，系统保存旧数据库的一个备份副本。spcutil 实用程序创建一个备份文件，并将它放在数据库文件夹中。确认压缩的数据库有效之后，可以删除备份数据库文件以释放出额外的磁盘空间。

#### 要压缩 Microsoft Access 数据库

- 1 确认 SPCPro 不在运行。
- 2 运行 spcutil.exe。此时出现 **spcutil** 对话框。
- 3 在**文件**菜单上，单击**压缩数据库**。此时出现**压缩数据库**对话框。



- 4 在**数据库**框中，输入 Access 数据库文件的文件夹位置与名称。
- 5 单击**确定**。

## 配置分布式 SQL Server 数据库

要配置 SQL 数据库以用于 SPCPro，必须先创建一个 Microsoft SQL Server 数据库。此外，您还必须拥有 SQL Server 系统管理员权限，才能将数据库用于 SPCPro。

### 要配置 Microsoft SQL Server 数据库

- 1 启动 WindowMaker。
- 2 在**特别**菜单上，指向 **SPC**，然后单击**数据库**。此时出现**配置 SPC 数据库**对话框。



- 3 在**数据库类型**列表中，单击 **Microsoft SQL Server**。
- 4 在**ODBC 数据源**框中，使用以下方法之一来选择 ODBC 数据源：
  - 从列表选择一个现有的 ODBC 数据源。
  - 从列表中选择 **< 新 >**。此时出现 **ODBC 数据源管理器**对话框。完成 ODBC 向导中的各个对话框，以创建一个新的 ODBC 数据源。
- 5 单击**保存**。
- 6 单击**验证**，以确认已经在 ODBC 与数据库之间建立有效的连接。在**连接状态**区域中，指示灯应该是绿色的，并且有一条消息指出 ODBC 连接成功。
- 7 设置其余的数据库选项。
  - a 在**管理员用户 ID**框中，输入 SQL Server 管理员用户帐户的名称。
  - b 在**口令**框中，输入管理员用户帐户的口令。
  - c 在**输出消息级别**区域中，单击设置消息记录级别的选项。**正常**是缺省级别，只将错误消息记录到 Wonderware Log Viewer。**详细**与**跟踪**仅应在进行诊断时选择。此时会有额外的 ODBC 消息写入 Wonderware Log Viewer。选择**详细**或**跟踪**会影响系统性能。

- d 在**数据储存限制**区域中，在**样本保留**框中输入希望保留数据的天数。数据会从当天起再保留指定的天数。超过该天数之后，数据将从数据库中清理掉。
  - e 选择**清理站**复选框。一次只能设置清除一个工作站。如果不定期从数据库中清理数据，应用程序性能会受到影响。缺省条件下，此字段设置为零，因此数据永远不会删除。如果使用缺省值，建议定期清理或归档数据，以防磁盘空间用完。
- 8 单击**确定**以关闭对话框。



## 配置一组产品的采集与分析选项

您必须为 SPC 应用程序配置数据集或间接数据集。在配置 SPC 数据集之前，定义这些 InTouch 标记：

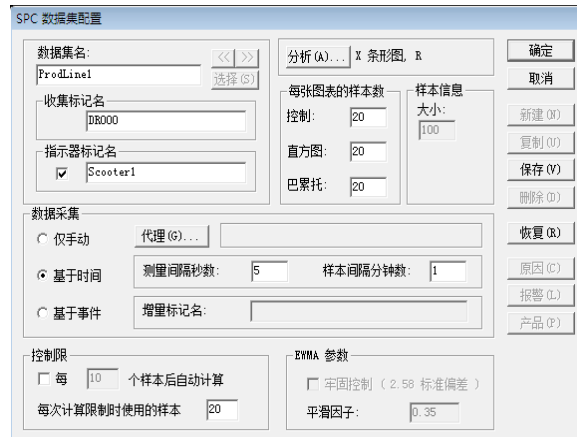
- SPC 采集标记。
- 指示器标记（如果希望在图表中使用指示器）。
- 作为计数器递增的标记（如果打算使用基于事件的采集）。

**备注** WindowViewer 标记使用计数不包括 SPCPro 用作采集标记、指示器标记或触发器标记的任何标记。

对于窗口中出现的或指定为 DDE 项目值的所有值，SPCPro 都使用六位浮点数。SPCPro 图表值如果超过六位精度，则会进行舍入。

### 要配置 SPC 数据集

- 1 启动 WindowMaker。
- 2 在**特别**菜单上，指向**SPC**，然后单击**数据集**。此时出现**SPC 数据集配置**对话框。



- 3 选择数据集。执行以下操作之一：
  - 单击**新建**以创建一个新的数据集。在**数据集名**框中，输入新数据集的名称。此时对话框关闭，所选的数据集名自动插入**数据集名**框。
  - 单击**选择**以选择一个现有的数据集。此时出现**选择数据集**对话框，用于选择要使用的数据集。
- 4 指定要使用的标记。执行以下操作：
  - a 在**收集标记名**框中，输入采集标记的名称；或双击空白框，然后从**选择标记**对话框中选择标记。
  - b 如果打算在 SPC 图表中使用指示器，请选择**指示器标记名**复选框，然后在框中输入模拟（实型或整型）内存标记。

- 5 单击**分析**。此时出现 **SPC 分析选择**对话框。在**分析类型**区域中，选择要给此数据集使用的分析类型，然后单击**确定**。
- 6 在**每张图表的样本数**区域中，输入每种 SPC 图表类型要显示的样本数，或使用缺省值 20。
- 7 在**样本信息**区域的大小框中，输入样本大小。此选项仅当选择 NP 图表时才会有。
- 8 在**数据采集**区域中，配置数据采集方法。执行以下任何一项操作：
  - 单击**仅可手动**，以便让用户通过单击 SPC 图表来输入样本。如果使用 QuickScript 将样本记录到数据集，请选择此方法。通过使用定时脚本，可以实现手工数据采集的自动化。
  - 单击**基于时间**，以便让 SPCPro 按一定的时间间隔来采集数据。通过单击**代理**，然后从 **SPC 用户**对话框中选择一个用户，可以选择代理进行自动数据采集。在**测量间隔秒数**与**样本间隔分钟数**框中，输入两个值。
  - 单击**基于事件**以进行基于事件的数据采集。在**增量标记名**框中，输入某个 InTouch 离散标记的名称。选择一个代理进行基于事件的采集。
- 9 在**控制限**区域中，设置各个控制限选项。执行以下操作：
  - a 选择**每**复选框，让 SPCPro 自动计算控制限。在“个样本后自动计算”框中，输入自动计算控制限的频率。
  - b 在**每一限制计算的样本数**框中，输入计算限制时要使用的样本数。
- 10 如果选择了 EWMA 作为分析类型，请选择 **EWMA 参数**区域中的选项。
  - a 如果希望更严格地控制指数加权移动平均，请选择**牢固控制（2.58 标准偏差）**复选框。
  - b 在**平滑因子**框中，输入希望使用的平滑因子。缺省值是 0.35。
- 11 单击**保存**。此时**产品**、**报警**以及**原因**按钮变为可用状态。

---

**备注** 配置新的数据集之后，必须为它配置至少一种产品，才能关闭 **SPC 数据库配置**对话框。

---

## 配置产品与关联的过程限制

SPC 数据集至少要定义一种产品。您也可以为相同的 SPC 数据集定义其它产品。

如果生产过程使用相同设备来生产多种不同的产品，请使用多种产品。例如，某个 SPC 数据集监视搅拌器的温度。搅拌器的温度设定点与系统响应根据正在生产的产品而各异。

如果使用多个产品名，则每次使用 **ProductCollected** 这个 DDE 项目来更改产品时，都自动更改所有的图表变量。

此项目中发生更改时，**SPCPro** 程序搜索上次运行产品时的文件，并将上次的图表变量用作新数据采集间隔的起点。

**备注** 配置新数据集之后，在关闭 **SPC 数据库配置** 对话框之前，必须为该数据集配置至少一种产品。

### 要为数据集配置产品

- 1 启动 WindowMaker。
- 2 在**特别**菜单上，指向 **SPC**，然后单击**数据集**。此时出现 **SPC 数据集配置**对话框。
- 3 单击**产品**。此时出现**产品**对话框。

- 4 在**名称**框中，输入产品名，最多包含 64 个字符。

- 5 在**中心图表**区域中，设置图表控制限、规格限、中心线及目标的值。执行以下操作：
  - a 在 **UCL** 框中，输入上控制限。
  - b 在 **USL** 框中，输入上规格限。
  - c 在**中心**框中，输入中心值。
  - d 在**目标**框中，输入图表的目标值。
  - e 在 **LCL** 框中，输入下控制限。
  - f 在 **LSL** 框中，输入下规格限。
- 6 在**宽度图表**区域中，设置均值与范围或标准偏差图表的控制限。执行以下操作：
  - a 在 **UCL** 框中，输入上控制限。
  - b 在**均值**框中，输入图表均值。
  - c 在 **LCL** 框中，输入下控制限。

这些新值用作下一个样本的缺省值。在运行时，可以通过 DDE 来更改这些极限。数据集为配置的每种产品单独保存一份图表值。**X 条形图**，**R**、**X 条形图**，**s** 以及 **X 方向移动**，**R 方向移动**图表均提供此选项。
- 7 在**样本信息**区域的**每个样本的测量数**框中，输入用于计算样本点的测量值数。有效值是 2 到 300。**X 条形图**，**R** 或 **X 条形图**，**s** 图表提供此选项。
- 8 在**显示标题**区域中，为产品的每种图表类型输入标题。
- 9 单击**保存**。要添加另一个产品，单击**新建**，回到第 3 步以配置该产品。
- 10 配置好所有产品时，单击**确定**。此时再次出现 **SPC 数据集配置**对话框。
- 11 单击**确定**以关闭 **SPC 数据集配置**对话框。

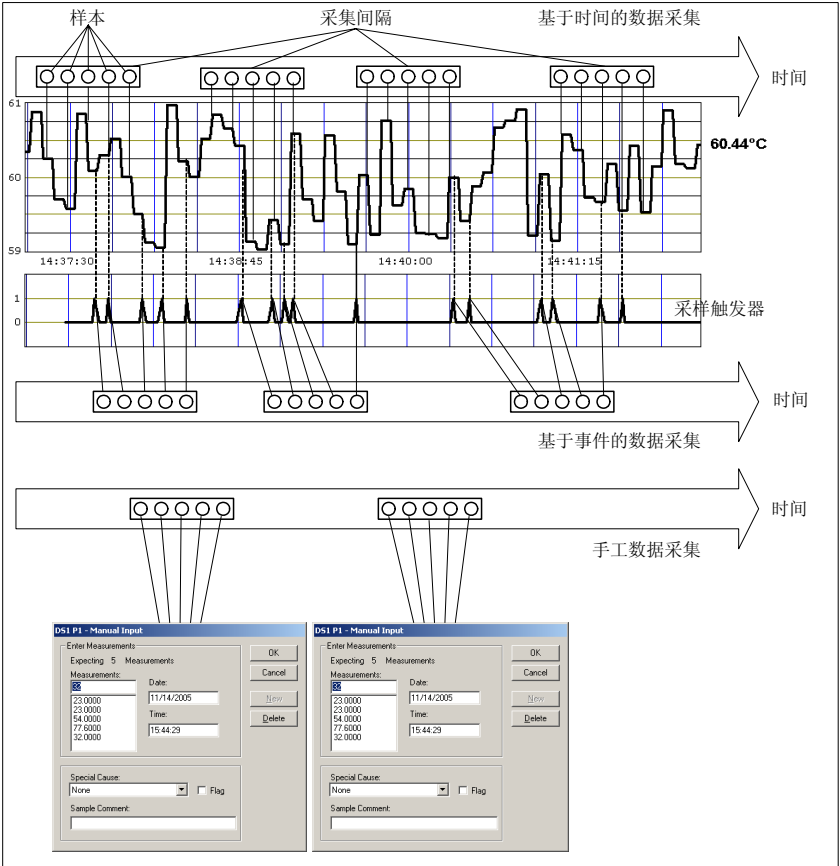
#### 要编辑现有的产品

- 1 启动 WindowMaker。
- 2 在**特别菜单**上，指向 **SPC**，然后单击**数据集**。此时出现 **SPC 数据集配置**对话框。
- 3 单击**产品**。此时出现**产品**对话框。
- 4 单击**选择**。此时出现**选择产品**对话框。
- 5 选择要修改的产品。此时再次出现**产品**对话框，并显示所选产品的配置设置。
- 6 执行所需的修改，然后单击**确定**。

# 使用基于时间或事件的采集方法自动采集 SPC 数据

通常使用基于时间或基于事件的采样方法将 SPCPro 数据采集到数据集。自动数据采集要求有代理或用户连接到存储统计数据的数据库。

下图显示 SPCPro 如何根据基于时间和基于事件的采样来自动采集统计数据。



基于时间的采集按固定的间隔将指定持续时间内的数据保存到数据集。例如，基于时间的数据集可以配置为每 5 分钟采集 10 秒内的样本。

基于事件的数据采集要求使用一个唯一的离散标记，此标记用作开始一个采样周期的触发器。离散标记设置为真时，数据集采集一段时间的样本，随后该标记重置为假。

您也可以使用 SPCPro 向导来手工采集数据。如需有关手工采集数据的详细信息，请参阅第 25 页的“使用 SPCPro 图表向导显示与手工采集 SPC 数据”。

## 配置 SPC 数据库用户

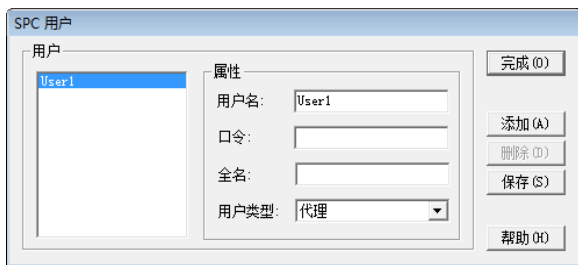
要执行自动数据集采集，必须配置 SPC 数据库用户与口令。不过，如果希望为 SPCPro 手工采集数据，则不必配置 SPC 用户。

**备注** 在设置用户文件之前，必须先配置数据库。

SPC 用户仅用于 SPCPro，不用于 SQL Server。SPCPro 用户不添加到 SQL Server 的用户列表。

### 要配置 SPC 数据库用户

- 1 启动 WindowMaker。
- 2 在**特别**菜单上，指向 **SPC**，然后单击**用户**。此时出现 **SPC 用户** 对话框。



- 3 单击**添加**。此时**属性**区域中的各个选项变为活动状态。
- 4 设置新用户的属性。
  - a 在**用户名**框中，输入用户名。
  - b 在**口令**框中，输入用户口令。此口令用作 SPCCConnect() 函数连接到数据库时的参数。对于不要求提供用户口令的数据库来说，**口令**框可以保持空白。
  - c 作为可选项，在**全名**框中输入完整的用户名。
  - d 在**用户类型**列表中，单击**代理**。
- 5 单击**完成**，或单击**保存**以配置另一个用户。

## SPCConnect() 函数

SPCConnect() 函数可以将代理或用户连接到数据集。

SPCConnect() 函数在自动数据采集数据集开始采集数据之前确定用户的身份。您必须使用 SPCConnect() 函数向 SPCPro 指出此节点是哪个“用户”。

您可以使用 SPCConnect() 在应用程序启动时连接到数据库。通过使用 SPCConnectType DDE 消息标记，可以监视自己是作为客户端还是代理（服务器）进行连接的。

通过运行包含 SPCConnect() 函数的脚本，可以将用户连接到数据库，并使用此用户 ID 启动所有的自动数据采集数据集。只要代理仍处于连接状态，配置了该代理的每个数据集便都会采集数据。

尽管一个代理可以关联到多个数据集，但每个数据集只能指定一个代理。此外，一次只能从每个 InTouch 应用程序中连接一个代理。

当前代理必须使用 SPCTDisconnect() 函数断开，另一个代理才能连接到 SPCPro 引擎。此外，相同的代理也可以从另一个 InTouch 节点进行连接。

类别  
SPC

语法

```
SPCConnect ("User", "Password");
```

参数

*User*

数据库用户名。实际的字符串或消息标记。

*Password*

用户的口令。实际的字符串或消息标记。

---

**备注** SPCConnect() 函数的 *User* 与 *Password* 参数区分大小写。如果在 Wonderware Log Viewer 中没有看到“成功地连接到用户 XXXX”消息，则表明代理未连接，数据集未在采集任何数据。

---

附注

SPCConnect() 将代理或用户连接到数据库，并使用此用户 ID 启动所有的自动数据采集数据集。如果连接数据库时不要求提供口令，请不要给 SPCConnect() 的 *Password* 参数指定任何值。

示例

本例在没有口令的情况下将 User1 连接到 SPCPro 数据库。

```
SPCConnect ("User1", "");
```

另请参阅

SPCTDisconnect()

### SPCDisconnect() 函数

SPCDisconnect() 函数断开代理与 SPCPro 数据库的连接。指定给此代理的所有自动数据集都停止采集数据。

**类别**  
SPC

**语法**

```
SPCDisconnect();
```

**参数**  
无。

**示例**

本例断开代理与 SPCPro 数据库的连接。指定给此代理的所有自动数据集都停止采集数据。

```
SPCDisconnect();
```

## 更改数据采集间隔时间

基于时间的数据集采集周期会自动错开，以提高系统性能。在有多个数据集同时采集数据时，这可以最大限度地缩短周期，从而减轻系统负载。缺省间隔周期是两秒。

如果按 1 分钟的间隔使用基于时间的采集，并且采集 30 个以上的数据集，则应该更改数据集间隔值。缺省的间隔值是两秒。

### 要更改数据采集间隔时间

- 1 如果需要，停止 WindowViewer 中的应用程序。
- 2 编辑 InTouch 应用程序文件夹中的 spc.ini 文件。
- 3 根据采集数据的数据集的数量，给该文件添加新的一行。

#### 30-40 个数据集

```
StaggerValue = 1500
```

#### 41-60 个数据集

```
StaggerValue = 1000
```

间隔值以毫秒为单位。

- 4 将更改保存到 spc.ini 文件。



## 使用 SPCPro 图表向导显示与手工采集 SPC 数据

您可以使用“SPC 图表”向导创建多种类型的统计图表。  
SPCPro 提供可以创建三种类型 SPC 图表的向导：

- 控制图表
- 直方图
- 巴累托图。

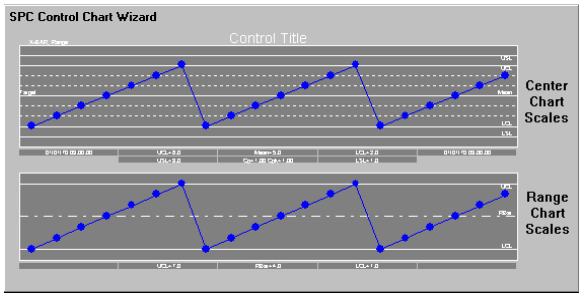
SPC 图表对象是您粘贴到 InTouch 窗口中的向导。然后，您使用这些向导来配置 SPC 图表，并将图表链接到包含统计数据的数据集。

通过配置“SPC 控制图”向导，可以显示以下图表类型：

X 个别	X 条形图 - R 图表	X 条形图 - s 图表
累加 (CUSUM) 图表	X 方向移动, R 方向移动图表	EWMA 图表
C 图表	P 图表	U 图表
NP 图表		

### 使用 SPC 控制图向导

“控制图”向导创建显示每个样本或子组的计算点的 X-Y 图表。这些点用线条连接起来形成控制图，以图形化的方式表示与所监控的过程关联的值。



图中使用一条中心线来显示组中所有点的平均值。上控制限线显示在中心线上方的三个标准偏差处。下控制限线位于中心线下方的三个标准偏差处。

上、下规格限线显示可接受的输出的任意上限与下限。目标线显示过程的期望平均值，它应该与中心线重合。区域线是中心线加上和减去一个或两个标准偏差后的参考线。

如果某个样本落在控制限之外（即违反某条运行规则），则会发生报警，通知用户产生该失控样本的特殊原因。

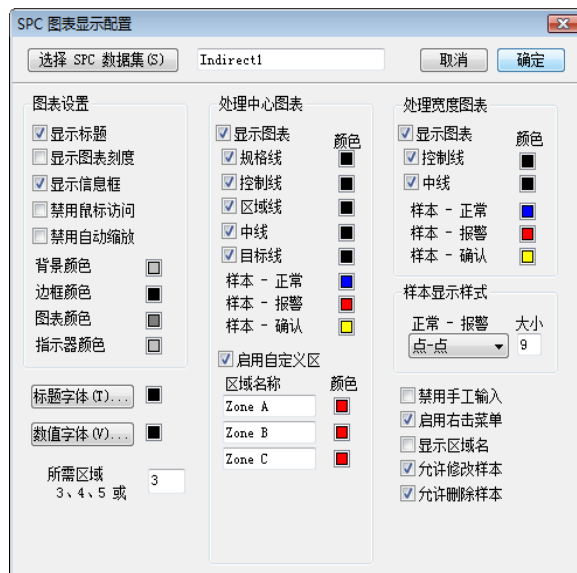
SPCPro 提供三个 SPC 图表向导和一个 SPC 限制向导。要使用这些向导，必须先从 WindowMaker 中将它们放入一个打开的窗口。然后，配置有关属性并将图表向导链接到数据集。

如需有关在窗口中放置向导的详细信息，请参阅 *InTouch® HMI 可视化指南* 中的第 5 章“向导”。向导类别是 **SPC 图表**。

使用“SPC 控制图”向导时，必须将“SPC 控制图”配置成链接到 SPC 数据集。

### 要配置控制图

- 1 如果需要，启动 WindowMaker。
- 2 打开包含“SPC 控制图”向导的窗口。
- 3 双击“SPC 控制图”向导。此时出现 **SPC 图表显示配置** 对话框。



- 4 通过使用以下方法之一，输入要用于“SPC 控制图”向导的数据集的名称：
  - 在对话框上部中心附近的输入框中输入数据集名。
  - 单击**选择 SPC 数据集**。此时出现**选择数据集**对话框。从列表选择一个数据集名，然后单击**确定**。
- 5 在**图表设置**区域中，设置确定图表视觉外观的选项：
  - 要在图表上放置一个标题栏，选择**显示标题**复选框。
  - 要在图表旁边显示刻度，选择**显示图表刻度**复选框。
  - 要显示图表信息框，选择**显示信息框**复选框。
  - 要在运行时禁用图表触控功能，选择**禁用鼠标访问**复选框。

如果未选择此选项，则在运行时单击图表中的样本字符时，会出现**样本信息**对话框。

- 要选择图表计算显示范围的方法，选择**禁用自动缩放**复选框。

通常情况下，图表显示范围按可以容纳显示的所有样本、控制限以及规格限来计算。如果选择此选项，则图表范围按以下方式确定：

如果图表配置成显示控制限与规格限，则范围扩展到可以显示所有显示的控制限与规格限值。

如果图表配置成显示控制限而没有规格限，则范围调整到可以显示所有的控制限值。

如果控制限或规格限都没有配置，则图表范围由当前的规格限值确定。

- 6 要设置图表颜色，执行以下操作：
  - a 单击每个颜色选项旁边的颜色框以显示调色板。
  - b 从调色板中选择所需的颜色。
- 7 选择图表使用的字体：
  - a 单击**标题字体**以打开**字体**对话框。选择图表标题的字体、字形以及大小。单击颜色框以便从调色板中选择希望用于标题的颜色。
  - b 单击**数值字体**以打开**字体**对话框。选择图表上显示的值的字体、字形以及大小。单击颜色框，以便从调色板中选择希望用于数值的颜色。
- 8 在**所需区域 3、4、5 或 6** 框中，输入图表中出现的区域的数量。
- 9 在**处理中心图表**区域中，设置确定要在中心图表中显示的线条的选项：
  - a 选择**显示图表**选项，以配置希望在“中心图表”上显示的线条。
  - b 选择希望在图表中出现的图表线条旁边的复选框。
  - c 单击颜色框，以便从调色板中选择希望用于**规格线、控制线、区域线、中线以及目标线**的颜色。
  - d 设置**正常、报警以及确认**样本点的颜色。
  - e 如果希望在“中心图表”中显示各级别的区域，选择**启用自定义区**。
- 10 在**处理宽度图表**区域中，设置用于定义和显示“宽度图表”的选项。您必须选择**显示图表**选项，才能配置要在“宽度图表”上显示的线条。
- 11 在**样本显示样式**区域中，可以更改在“中心图表”与“宽度图表”中使用的样本样式与点字符大小。

**12** 设置确定在运行时用户可以对图表执行什么操作的选项：

- 要防止用户手工输入样本值，选择**禁用手工输入**。
- 要启用带用户选项的快捷操作菜单，选择**启用右击菜单**。

此菜单包含**确认样本**、**删除样本**、**修改样本**、**区域中心**以及**添加 / 删除原因**等选项。

- 要在“中心”图表上显示区域名，选择**显示区域名**。
- 要允许用户通过快捷菜单来删除与修改样本值，选择**允许删除样本 / 允许修改样本**。

**13** 单击**确定**以保存配置值。

任何样本点的详细样本信息都可以获取。注释与特殊原因也可以同任何样本关联。样本信息可以通过 DDE 提供，也可以通过操作员在运行时单击样本所出现的**样本信息**对话框来提供。

## 要访问“样本信息”对话框

- 1 在运行时，单击“SPC 控制图”中当前显示的样本。此时出现**样本信息**对话框。

- **样本号**框显示从“SPC 控制图”中选择的样本的编号。
  - **日期时间**框显示样本的采集日期与时间。
  - 显示所选的“分析类型”的图表值。
  - **报警**窗口显示所显示的样本的所有报警情况。
  - **测量值**列表显示用于计算样本的所有测量值的实际值。
- 2 在**注释**框中，输入 50 个字符以内的样本注释。
  - 3 在**注释文本**框中，输入在图表上出现的备注，不超过 12 个字符。
  - 4 在**特殊原因**框中，输入样本的特殊原因。
  - 5 如果要在图表上给样本做标记，选择**标帜样本**。

- 6 选择**忽略值**以强制重新绘制“SPC 控制图”，而忽略所选样本的自动调整计算结果。

该样本仍会绘制出来，但不会出现在图表显示区内。该样本值在“直方图”中也会被忽略。

**备注** 选择此选项不会在计算控制限的过程中排除该样本点，而只是将它排除在“SPC 控制图”的显示区之外。

- 7 单击**新建**访问**手工输入**对话框，以手工输入样本的测量值。

- 8 单击**修正操作**访问**修正操作**对话框，以便对样本采取修正操作。

- 9 单击**确定**。

通过将“DDE 项目名” **ManualInputDialog** 设置为 1 以 DDE 方式，或是通过访问**手工输入**对话框，都可以在运行时将新的样本添加到数据集。

#### 要使用“手工输入”对话框添加样本

- 1 单击“SPC 控制图”上的某个样本。此时出现**样本信息**对话框。
- 2 单击**新建**。此时出现**手工输入**对话框。

- 3 在**测量值**框中，输入样本的测量值：
  - a 输入为样本采集的测量值。
  - b 按 **Enter** 键。此时该值输入到输入框下的框中。
  - c 重复步骤 a 与 b，直到输入了**测量值**框上方指定的所需数量的测量值。
- 4 要更改日期或时间，请在相应的输入框中输入新信息。  
缺省条件下，**日期**与**时间**框中显示与这个新样本关联的当前日期与时间。
- 5 在**特殊原因**框中，输入样本的特殊原因。否则接收缺省值**无**。

- 6 选择**标识**以便在“SPC 控制图”上给样本做标记。
- 7 在**样本注释**框中, 输入 50 个字符以内的样本注释。
- 8 单击**确定**将测量值添加到数据集并关闭对话框。此时再次出现**样本信息**对话框。
- 9 单击**确定**。

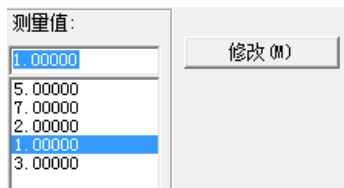
通过使用适当的 DDE 项目, 可以创建手工输入窗口。借助 DDE, 可以使用 InTouch 数据改变或条件脚本来轻松实现手工输入自动化。

“控制图”样本信息可以轻松修改与删除。不过, CuSum、EWMA、“X 方向移动, R 方向移动”数据集类型不提供此选项。

在控制图表配置中, 必须启用此选项。通过在脚本中使用 DDE 项目 SPCAAllowSampleDelMod, 可以在运行时打开 / 关闭它。

#### 要修改样本

- 1 使用鼠标右键单击要修改的点。此时出现操作菜单。
- 2 从菜单中选择**修改样本**。此时出现**样本信息**对话框。

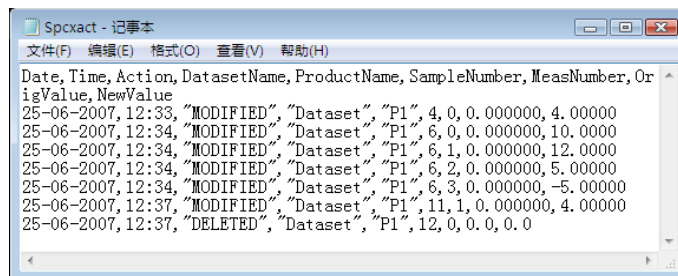


- 3 突出显示要更改的测量值。
- 4 输入新的测量值, 然后单击**修改**。此时所选的测量值显示修改后的值。
- 5 单击**确定**。

### 要删除样本

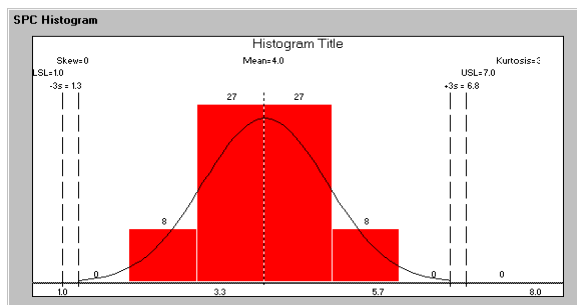
- ◆ 使用鼠标右键单击要删除的点，然后单击**删除样本**。出现消息时，单击**是**。

每次删除或修改样本时，SPCPro 都会将更改记录到文件中。此日志是位于 InTouch 应用程序文件夹中的 spcxact.log 文件。该文件按时间顺序显示对这些样本进行的所有修改。



## 使用“SPC 直方图”向导

直方图根据用于控制图的原始测量数据创建。直方图显示所采集的数据的分布与频率。对于正常的过程，值的分布曲线呈钟形。



使用“SPC 直方图”向导创建图表时，您将向导粘贴到 WindowMaker 窗口，然后配置向导并将它链接到数据集。

**备注** 要配置“SPC 直方图”，必须输入先前定义过的数据集名。

### 要配置“SPC 直方图”向导

- 1 将“SPC 直方图”向导粘贴到窗口中，然后双击它。此时出现 SPC 直方图显示配置对话框。



- 2 通过以下方法之一，指定要用于直方图的数据集的名称：
  - 在输入框中输入数据集名。
  - 单击**选择 SPC 数据集**，然后从出现的**选择数据集**对话框中选择数据集。



- 3 在**区域数**区域中，选择以下选项之一来设置在直方图中显示的区域数：
  - **固定数字**  
创建固定数量的区域，以根据**个区域**框中所输入的数字来限制样本大小。
  - **基于样本大小**  
基于样本大小来创建在“直方图”上显示的区域数。
- 4 在**图表设置**区域中，设置确定直方图视觉外观的选项，具体如下：
  - 要给直方图显示标题，选择**显示标题**。
  - 要在直方图中显示正态分布曲线，选择**显示正态曲线**。
  - 要在直方图中显示规格线，选择**规格行**。
  - 要在直方图中显示控制线，选择**控制线**。
  - 要在直方图中显示中心线，选择**中线**。
- 5 通过执行以下操作来设置直方图的颜色：
  - a 单击每个选项旁边的颜色框以显示调色板。
  - b 从调色板中选择一种颜色。
- 6 通过执行以下操作来选择在直方图中显示的文本的外观：
  - a 单击**标题字体**以打开**字体**对话框。
  - b 选择图表标题的字体、字形以及大小。
  - c 单击颜色框，以便从调色板中选择希望用于“标题”的颜色。
  - d 单击**数值字体**以打开**字体**对话框。
  - e 选择图表上显示的值的字体、字形以及大小。
  - f 单击颜色框，以便从调色板中选择希望用于值的颜色。
- 7 单击**确定**以保存对配置所作的更改。

## 使用间接数据集动态更改显示的产品

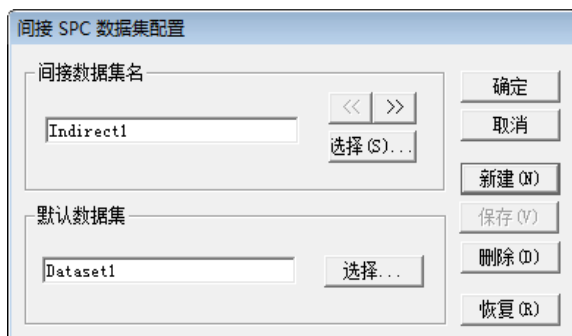
通过使用间接数据集，SPC 图表可以在运行时动态链接到任何数据集。借助间接数据集，可以使用相同的 SPC 图表来显示多个数据集的数据。要在运行时更改链接，可以将 DDE 项目 `DatasetName` 设置为所需的数据集。

配置 SPC 图表时，必须将它链接到 SPC 数据集。如果将 SPC 图表链接到间接数据集，则该图表可以显示任何数据集的数据。将图表链接到间接数据集是通过在运行时更改 DDE 项目 `DatasetName` 来完成的。DDE 项目 `DatasetName` 发生更改时，间接数据集接受自己所链接的数据集的所有属性与项目值。

**备注** 配置间接数据集时，必须输入先前定义过的数据集名。

### 要配置间接数据集

- 1 启动 WindowMaker。
- 2 在**特别菜单**上，指向 **SPC**，然后单击**间接数据集**。此时出现**间接 SPC 数据集配置**对话框。



- 3 在**间接数据集名**框中，为间接数据集输入唯一的名称，最多 31 个字符。
- 4 在**默认数据集**框中，通过以下方法之一来输入希望间接数据集链接的数据集的名称：
  - 在**默认数据集**框中输入间接数据集名。
  - 单击**选择**以显示**选择数据集**对话框。此时列出当前已定义的所有数据集。从列表中选择数据集名。
- 5 单击**确定**。

## 分析失控情况的原因

统计质量控制给普通原因和特殊原因指定距离可接受标准的偏差。普通原因是生产过程中固有的原因，如产品设计不当或湿度过大。特殊原因是局部的、偶发性的问题，如特定机器发生故障或是测量值记录有误。确定并消除特殊原因之后，过程即视为处在统计控制之中。

本节阐述如何使用 SPCPro 来确定生产过程中的特殊原因。本节包含的信息阐述如何：

- 定义特殊原因
- 创建“巴累托图”

### 配置特殊原因代码

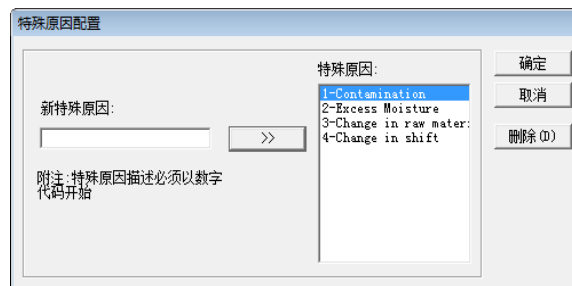
失控的 SPC 样本可以归结为是由于特殊原因所导致的。您可以在 WindowMaker 中通过**特殊原因配置**对话框来定义特殊原因。在 WindowViewer 中，通过 DDE 项目（例如 **CurrentCauseCode**）或**样本信息**对话框，可以将特殊原因附加到任何样本。

**备注** 在运行时，通过使用鼠标右键单击样本，然后选择**添加 / 删除原因**，可以显示**特殊原因配置**对话框。

随后这些原因的摘要可以显示在“巴累托图”上，以确定造成失控样本的主要原因。

#### 要给数据集配置特殊原因

- 1 启动 WindowMaker。
- 2 在**特别**菜单上，指向 **SPC**，然后单击**数据集**。此时出现 **SPC 数据集配置**对话框。
- 3 在 **SPC 数据集配置**对话框，单击**原因**。此时出现**特殊原因配置**对话框。

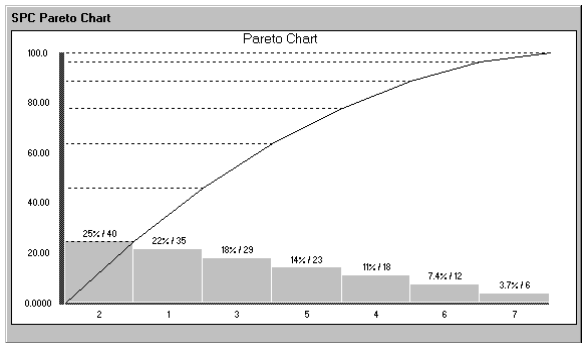


- 4 通过执行以下操作来输入原因代码：
  - a 在**新特殊原因**框中，输入带有代码编号的原因描述。此编号确定“巴累托”图中的各个列。例如，输入 1-Startup

- b 单击 >> 按钮或按 **ENTER** 键，以便将新的特殊原因添加到**特殊原因**列表框。
  - c 根据需要添加任何数量的特殊原因描述。
- 5 单击**确定**。

## 使用“巴累托图向导”以图形化方式显示特殊原因的分布

“巴累托图”是条形图的特殊形式，它显示问题或条件的相对重要性。“巴累托图”以图形化方式呈现发生的特殊原因的数量。由于用户在确认报警时通常会输入特殊原因代号，因此“巴累托图”可以在指定数量的样本上使用这些项，以降序条形图的形式显示它们。



“巴累托图”按最频繁到最不频繁的顺序来排列数据，使您可以确定给定的过程中最重要的因素。尽管导致样本失控的可能原因有很多，但导致产生大部分不合格样本的特殊原因通常只有一两个。

**备注** 对于输入的新样本，“巴累托图”会重新计算控制限。

使用“SPC 巴累托”向导时，必须给它配置将“SPC 巴累托”图链接到 SPC 数据集所需的信息。

**备注** 要配置“SPC 巴累托图”，必须输入先前定义过的数据集名。

### 要配置“巴累托图”

- 1 将“SPC 巴累托”向导粘贴到窗口中，然后双击它。此时出现 **SPC 巴累托图显示配置**对话框。



- 2 按照以下方法之一，指定要用于“巴累托图”的数据集的名称：
  - 在输入框中输入数据集名。
  - 单击**选择 SPC 数据集**，然后从出现的**选择数据集**对话框中选择数据集。
- 3 在**区域数**区域，选择以下选项之一来设置在“巴累托图”中显示的区域数：
  - **固定数字**  
创建固定数量的区域，以根据**个区域**框中所输入的数字来限制样本大小。
  - **全部原因代码**  
如果希望在“巴累托”上显示的“区域数”等于特殊原因代码的数量，请选择此项。
- 4 在**图表设置**区域中，设置确定“巴累托图”视觉外观的选项，具体如下：
  - 要给“巴累托图”显示标题，选择**显示标题**。
  - 要给“巴累托图”显示图例，选择**显示图例**。
- 5 通过执行以下操作来设置“巴累托图”的颜色：
  - a 单击每个选项旁边的颜色框以显示调色板。
  - b 从调色板中选择一种颜色。
- 6 通过执行以下操作来选择“巴累托图”中所显示的文本的外观：
  - a 单击**标题字体**以打开**字体**对话框。
  - b 选择图表标题的字体、字形以及大小。
  - c 单击颜色框，以便从调色板中选择希望用于“标题”的颜色。
  - d 单击**数值字体**以打开**字体**对话框。
  - e 选择图表上显示的值的字体、字形以及大小。
  - f 单击颜色框，以便从调色板中选择希望用于值的颜色。

7 单击**确定**以保存对配置所作的更改。

## 在运行时动态更改极限值

“SPC 限制”向导是一个 SPC 控制面板向导，可用于更新与查看 SPC 规格限和 SPC 控制限。它也可以用于切换数据集、切换产品以及滚动 “SPC 图表”。

XUSL #.####	Update
XUCL #.####	
Target #.####	Current X Sample #.####
MEAN #.####	Current R Sample #.####
XLCL #.####	
XLSL #.####	◀ ▶ Scroll Value #
RUCL #.####	
RBAR #.####	
RLCL #.####	
Dataset #	
Product	
Displayed #	
Collected #	

**备注** 对于输入的第一个新样本，“巴累托图”会重新计算控制限。

使用 “SPC 限制” 向导时，必须给它配置链接到 SPC 数据集所需的信息。

**备注** 要配置 “SPC 限制” 向导，必须输入先前定义过的数据集名。

### 要配置 “SPC 限制” 向导

- 1 将 “SPC 限制” 向导粘贴到窗口中，然后双击它。此时出现 SpcLimit 向导配置对话框。

SpcLimit 向导配置

数据集  
dataset2 数据集(D) 建议(S) 确定 取消

标记

CurrentXUSL	CurrentXLSL	CurrentTarget
CurrentXUSL	CurrentXLSL	CurrentTarget
CurrentXUCL	CurrentXLCL	CurrentSampleBar
CurrentXUCL	CurrentXLCL	CurrentSampleBar
CurrentRUCL	CurrentRLCL	CurrentRBar
CurrentRUCL	CurrentRLCL	CurrentRBar
CurrentSample	CurrentR	CurrentUpdate
CurrentSample	CurrentR	CurrentUpdate
DatasetName	ProductCollected	ProductDisplayed
DatasetName	ProductCollected	ProductDisplayed
LastSampleDisplayed	Scroll	
LastSampleDisplayed	SCROLL	

- 2 在**数据集**区域中，按照以下方法之一指定数据集的名称：
  - 在输入框中输入数据集名。
  - 单击**数据集**，然后从出现的**选择数据集**对话框中选择数据集。
- 3 在**标记**区域中，通过执行以下操作之一来指定要与图表对象关联的标记名：
  - 为各个项目输入已经在“标记名字典”中定义过的标记名。
  - 单击**建议**让向导自动为每个项目建议一个标记。
  - 在空白的**标记**输入框中双击，以显示**选择标记**对话框，该对话框显示所选标记源中已定义标记的列表。从列表中选择要使用的标记，然后单击**确定**。此时标记浏览器关闭，所选的标记自动插入所选的字段。
- 4 单击**确定**进行保存。

## 处理 SPCPro 报警

您可以将 SPCPro 设置为使用 InTouch 标记的数据，以确定某个过程是否处在统计控制之中。本节阐述如何设置 SPCPro 以便：

- 确定报警条件
- 确认报警
- 输入修正操作对报警作出响应

## 配置 SPCPro 报警

您可以使用 SPCPro 来分析报警条件的图表数据。它检查超限条件与七个不同 Western Electric 运行规则。报警记录到 InTouch 数据库，并且通知数据集配置中所指定的标记。

特定的 SPC 报警可以在**样本信息**对话框中查看，或是通过 SPC 报警消息 DDE 项目进行查看。

所有的报警都有指定的优先级。有效的优先级在 1 到 999 之间。此值代表报警的严重程度，1 表示最严重。通过给报警范围指定优先级，可以根据优先级过滤出关键报警。

### 要配置报警条件

- 1 启动 WindowMaker。
- 2 在**特别菜单**上，指向 **SPC**，然后单击**数据集**。此时出现 **SPC 数据集配置**对话框。
- 3 在 **SPC 数据集配置**对话框中，单击**报警**。此时出现 **SPC 报警选择**对话框。

SPC 报警选择		优先级
<b>限制报警</b>		
<input checked="" type="checkbox"/>	超出规格限制的样本	999
<input checked="" type="checkbox"/>	超出控制限制的样本	999
<b>标准偏差报警</b>		
<input checked="" type="checkbox"/>	2 个样本 (3 个最新样本中)，超过 2 标准偏差 (同侧)	999
<input checked="" type="checkbox"/>	4 个样本 (5 个最新样本中)，超过 1 标准偏差 (同侧)	999
<input type="checkbox"/>	1 个样本 (1 个最新样本中)，超过 1 标准偏差	999
<input type="checkbox"/>	1 个样本 (1 个最新样本中)，超过 1 标准偏差 (同侧)	999
<b>连续报警</b>		
<input type="checkbox"/>	在 1 个标准偏差内的连续样本	8 999
<input type="checkbox"/>	超过 1 个标准偏差的连续样本	15 999
<input type="checkbox"/>	递增或递减的连续样本	6 999
<input type="checkbox"/>	上下交替的连续样本	14 999
<input type="checkbox"/>	在中线的一侧的连续样本	8 999

- 4 在**限制报警**区域中，选择根据样本值来限制报警的选项：
  - 要为超出规格限的样本值的数量设置报警，选择**超出规格限制的样本**。



- 要为超出控制限的样本值的数量设置报警，选择**超出控制限制的样本**。
- 5 在**标准偏差报警**区域中，选择一些选项，以便根据超出图表标准偏差极限的样本值的数量来设置报警。
- 要为最新 3 个样本中有 2 个超出 2 个标准偏差的这种情况设置报警，选择**标准偏差报警**区域中相应的选项。
  - 要为最新 5 个样本中有 4 个超出 1 个标准偏差的这种情况设置报警，选择**标准偏差报警**区域中相应的选项。
  - 如果需要，可以通过在**标准偏差报警**区域列出的最后两个选项中，输入超出标准偏差极限的样本数，来创建自定义的标准偏差报警限。
- 第一个选项可以包含图表中心线两侧的样本点。第二个报警选项与此相同，只是它的所有样本都必须在中心线之上或之下。对于这两个选项，头两个字段是整数，最后一个字段是实数。
- 6 在**连续报警**区域中，选择一些选项，以便根据超出图表极限的连续样本值的数量来设置报警。
- a 选择希望用于选择报警的报警选项旁边的复选框。
  - b 为设置报警阈值的每个选项输入连续样本的数量。
- 7 给从 **SPC 报警选择**对话框中选择的每个报警条件指定一个**优先级**。
- 8 单击**确定**。

显示与确认 SPCPro 报警

SPCPro 向 “InTouch 报警管理器” 提供报警数据。您可以在 AlarmViewer ActiveX 控件中显示当前的 SPCPro 报警。通过使用鼠标右键单击 SPCPro 图表中的报警样本，然后从操作菜单中选择**确认样本**，可以确认报警。此时报警在 SPC 图表与 AlarmViewer 中得到确认。此外，报警也可以直接在 AlarmViewer ActiveX 控件中进行确认，该控件随后更新 SPC 样本图表。

时间 /	状态	分类	类型	优先级	名称	组
02/09/2007 11:25:45 上午	UNACK	VALUE	HI	1	ReactLevel	React
02/09/2007 11:25:51 上午	UNACK_RTN	VALUE	HI	1	ReactTemp	React
02/09/2007 11:25:56 上午	UNACK_RTN	VALUE	HI	1	ProdLevel	React

正在显示第 1 到第 3 个报警，共有 默认查询

100 % 完成

SPCPro 在运行时进行初始化的时候，会分析数据集直到计算的上一个控制限，以确保正确分析报警（运行规则）。对于庞大的数据集，这可能要花费较长一段时间。要禁止在初始化期间检查数据集中的报警，请在 SPC.INI 文件的 [General] 部分包含以下这行代码：

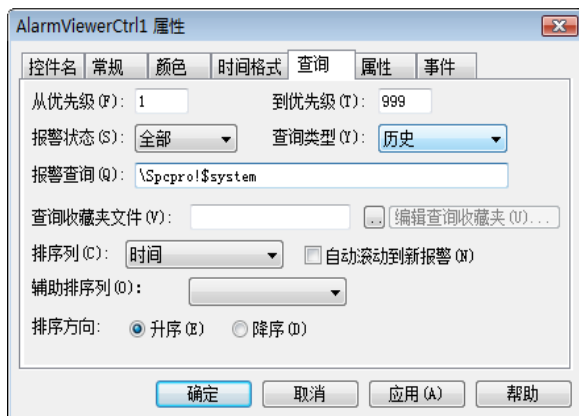
```
AlarmAnalysisOnStartUp=1
```

**备注** SPCPro 是 InTouch 报警供应器，它必须在“报警对象”中配置为报警供应器。

以下操作程序介绍配置 AlarmViewer 控件以显示 SPCPro 报警的基本步骤。如需有关配置所有 AlarmViewer 属性的详细信息，请参阅 *InTouch® HMI 报警与事件指南* 中的第 7 章“查看报警层次结构”中的“配置 Alarm Tree Viewer 控件”。

### 要为 SPCPro 报警配置 Alarm Viewer 控件

- 1 打开包含 Alarm Viewer 控件的窗口。
- 2 双击 Alarm Viewer 控件。此时出现 **AlarmViewerCtrl 属性** 对话框。
- 3 单击**查询**选项卡，以显示该控件的报警属性。



- 4 在**报警查询**框中，根据 SPCPro 的报警供应器输入节点地址。
  - 要显示本地 SPCPro 节点的报警，输入以下内容：  
`\SPCPro!$System`
  - 要显示 SPCPro 服务器节点的所有供应器的报警，输入以下内容：  
`\\NodeName\spcpro!$System`
  - 要显示所有供应器以及本地 SPCPro 节点的报警，输入以下内容，并将空格用作字符串分隔符：  
`\Spcpro!$System \\NodeName\spcpro!$System`

- 5 单击应用。
- 6 设置其它 Alarm Viewer 控件属性。
- 7 设置 Alarm Viewer 控件的其它所有属性之后，单击确定。

## 输入修正操作

操作员执行修正操作或控制移动，以告知 SPCPro 已经对生产过程作出更改或修改。通过执行修正操作，操作员也可以将更改记录在附加到样本号的日志中。

例如，操作员在“控制图”上看到一个报警点。操作员观察到有一个控制阀只打开了一部分，而不是完全打开。操作员跑到生产线上，确认该阀门没有正常工作。因此，操作员决定更换阀门。在完成维修之后，操作员在样本上输入修正操作。

在“SPC 控制图”中，该样本使用符号 [c] 作出标识，指出已经采取修正操作。此外，SPCPro 报警计数器会根据 SPC.INI 文件（位于 InTouch 应用程序文件夹）中的开关设置进行重置。对最后一个样本 (CurrentSampleNumber) 采取“修正操作”也将重置“SPCPro 运行规则计数”。

采取修正操作时，“SPC 报警”计数器自动重置为零。这个重置操作通过 SPC.INI 文件中的开关设置进行控制。缺省设置是只重置那些已执行过修正操作的样本的报警。不过，通过在 SPC.INI 文件中包含以下这行代码，可以重置所有的 SPC 报警计数器：

```
[General]  
ResetAllAlarmCounters=1
```

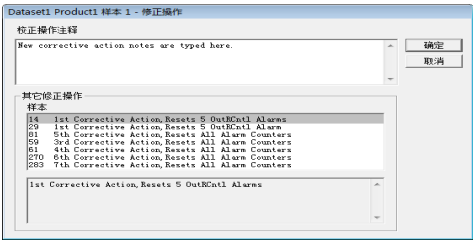
修正操作名通过 SPC.INI 文件中的开关设置进行控制。缺省条件下，使用名称“修正操作”。不过，通过在 SPC.INI 文件中包含以下这行代码，可以将名称更改为“控制移动”：

```
[General]  
CorrectiveAction=0
```

此开关设置将名称更改为“控制移动”。

要输入修正操作

- 1 单击“SPC 控制图”中的报警样本。此时出现**样本信息**对话框。
- 2 单击**修正操作**。此时出现**修正操作**对话框。



- 3 在**修正操作注释**窗口中，输入关于对样本采取的修正操作的备注。

**备注** 其它修正操作窗口列出对链接到“SPC 控制图”的 SPC 数据集中的样本采取的所有修正操作。如果选择列出的某个修正操作，它相应的备注会显示在下方的窗口中。

您可以滚动文本。您也可以选择文本、按 **Ctrl+C** 组合键复制它，然后按 **Ctrl+V** 组合键将该文本粘贴到**修正操作注释**窗口。随后，您可以根据修正操作的需要来修改文本。

- 4 单击**确定**。此时出现一条消息，要求确认对样本执行修正操作。
- 5 单击**是**将修正操作插入 SPC 数据库。此时再次出现**修正操作**对话框。
- 6 单击**确定**。此时再次出现**样本信息**对话框。
- 7 单击**确定**。

## 使用 DDE 项目与脚本函数进行运行时控制

在运行时，可以通过多种方法来控制 SPC 图表。您可以创建由鼠标操作或是键盘输入（激活脚本或 SPC DDE 项目）控制的 SPC 应用程序。

SPC 函数与 DDE 让您可以对图表对象与 SPC 数据进行广泛的控制。本节介绍一些 SPC DDE 项目与 SPC 函数，在运行时可以使用它们来控制 SPC 应用程序。

DDE 项目可用于获取数据集信息与控制图操作。应用程序名是 SPCPro。主题名是数据集名。主题名区分大小写。

“控制”与“显示”DDE 项目用于控制和显示关于主题数据集的信息。“控制”DDE 项目由所有的节点共享。它们是采集的远程数据集产品的数据集值。“显示”DDE 项目是每个节点本地的。它们是本地节点上所显示的产品样本值。

样本修改可应用于采集与显示的本地数据集的产品。通过单击图表显示所作的修改会影响显示的产品。SPC DDE 项目修改所采集的产品。显示的产品与采集的产品各有自己的当前样本，即最近记录的样本。

对于采集与显示的产品，会评估并存储报警。报警仅在运行时针对采集的产品进行报告。

在添加显示与采集的产品之后，许多“SPC DDE 项目”仅应用于采集的产品。在下表中，这些项目通过在 SPC DDE 项目名前加一个星号(\*)做标记。

项目名	DDE 类型	访问权限	描述
AutoCollection	Discrete	可读写	启用 / 禁用自动数据采集。
*CalculateControlLimits	Discrete	可读写	设置为 1 时启动控制限计算。
DatasetName	Message(32)	可读写	设置由“间接数据集”使用的“数据集名”。
HistogramLCL	Real	只读	基于数量显示“直方图”的“下控制限”。
HistogramMean	Real	只读	显示直方图的平均值。
HistogramUCL	Real	只读	基于数量显示“直方图”的“上控制限”。
Kurtosis	Real	只读	“直方图”的分布形状。
LastSampleDisplayed	Integer	可读写	设置数据集显示的最后一个样本号。
*ManualInputDialog	Discrete	可读写	设置为 1 时显示内置的“手工输入”对话框。

项目名	DDE 类型	访问权限	描述
<b>MeasurementsPerSample</b>	Integer	只读	显示配置的“每个样本的测量值数”。
<b>NewProduct</b>	Message(64)	可读写	用于创建新的“产品名”。
<b>NewProductCtrlTitle</b>	Message(32)	可读写	用于设置由 NewProduct 函数创建的新产品的“控制图”标题。
<b>NewProductParetoTitle</b>	Message(32)	可读写	用于设置由 NewProduct 函数创建的新产品的“巴累托图”标题。
<b>NewProductHistTitle</b>	Message(32)	可读写	用于设置由 NewProduct 函数创建的新产品的“直方图”标题。
<b>*ProductCollected</b>	Message(32)	可读写	更改数据集采集的“产品名”。
<b>ProductDisplayed</b>	Message(32)	可读写	更改数据集显示的“产品名”。
<b>SampleSize</b>	Integer	只读	NP 数据集的样本大小。
<b>SamplesControlChart</b>	Integer	可读写	设置“控制图”中显示的样本数。
<b>SamplesPerHistogram</b>	Integer	可读写	设置“直方图”中显示的样本数。
<b>SamplesPerLimitCalc</b>	Integer	可读写	设置计算控制限时使用的样本数。
<b>SamplesPerPareto</b>	Integer	可读写	设置“巴累托图”显示画面中使用的样本数。
<b>SelfSPCOutSpecMsg</b>	Message	只读	“超出规格限制的样本”的报警消息标记。
<b>Skewness</b>	Real	只读	显示与“直方图”上均值的偏差。
<b>SPCAllowSampDelMod</b>	Discrete	可读写	打开或关闭“删除样本”与“修改样本”这两个快捷菜单选项。
<b>SPCConnection</b>	Discrete	只读	如果与服务器失去连接，则设置为 0。
<b>SPCConnectType</b>	Message	只读	显示节点是作为代理（服务器）还是“客户端”来连接的。
<b>SPCLowDBSpace</b>	Discrete	只读	用于监视 Microsoft SQL Server 数据库。仅对 Microsoft SQL Server 数据库有效。数据库空间不足时，它将等于 1。可用于停止 AutoCollection，并提醒操作员分配更多的空间或是释放一些硬盘空间。它将根据 SQL 数据库的状态在 1 与 0 之间自动切换。

项目名	DDE 类型	访问权限	描述
<b>SPCResetRunRules</b>	Discrete	可读写	用于重置给新采集的样本应用的运行规则。这在采集新批次的数据时非常重要；对于新批次的样本，您可能不希望应用那些运行规则。您可能希望重置新批次的规则。
<b>StartCollection</b>	Discrete	可读写	设置为 1 时启动“自动采集”周期。
<p>所有的“当前样本”DDE 项目都与给定数据集中上次采集的样本有关。它们可用于更改同“数据集名”关联的原始数据与极限。要更改当前样本的有关信息，必须将这些信息写入适当的 DDE 项目，然后将 DDE 项目 <b>CurrentUpdate</b> 设置为 1。这会更新样本，并触发任何所需计算的执行。输入样本之后，SPC 程序将 <b>CurrentUpdate</b> 这个“DDE 项目”重置为零。在下一个样本开始一个采集周期之后，当前样本 DDE 项目不能再进行更新。</p> <p>当前样本 DDE 项目在所有的节点之间共享。这些项目值代表为所采集的产品采集的最后一个样本。</p> <p>对于分布式 SPC，最初所有的值都设置为零。SPC 连接到数据库，并且每隔五秒检查一次是否有新的数据。只要发现新的信息，这些项目值便会更新。对当前样本值所作的修改缓冲在本地，直到 <b>CurrentUpdate</b> 项设置为 1。随后，这些值放入当前样本数据包，并发送到远程数据集节点进行分析与储存。</p> <p>在添加显示与采集的产品之后，所有的当前 SPC DDE 项目仅应用于采集的产品。</p>			
项目名	DDE 类型	访问权限	描述
<b>CalculateCpAndCpk</b>	Discrete	可读写	设置为 1 以便在当前样本之后立即计算 Cp 与 Cpk 值。
<b>CurrentCauseCode</b>	Integer	可读写	设置当前样本的特殊原因代码编号。
<b>CurrentCauseString</b>	Message(128)	只读	显示当前样本的特殊原因代码编号的描述。
<b>CurrentComment</b>	Message(50)	可读写	用于读取 / 写入同当前样本关联的任何其它注释。
<b>CurrentCp</b>	Real	只读	显示当前样本的容量。
<b>CurrentCpk</b>	Real	只读	显示当前样本的中心容量。

项目名	DDE 类型	访问权限	描述
<b>CurrentDate</b>	Message(10)	可读写	按 DD/MM/YY 或 DD/MM/YYYY 格式设置当前样本的“日期”。如果输入不正确，则缺省使用当前日期。
<b>CurrentFlag</b>	Discrete	可读写	设置当前样本的标识。
<b>CurrentIgnoreValue</b>	Discrete	可读写	设置在“控制图”为“自动缩放”模式时忽略当前样本。
<b>CurrentMx</b>	Real	可读写	设置当前样本的单独测量值。(x=1 到 25)。
<b>CurrentNote</b>	Message(12)	可读写	给控制图表上的当前样本设置备注文本。
<b>CurrentR</b>	Real	只读	显示当前样本的范围。
<b>CurrentRBar</b>	Real	可读写	设置当前样本的平均范围。
<b>CurrentRLCL</b>	Real	可读写	设置范围的“下控制限”。
<b>CurrentRUCL</b>	Real	可读写	设置范围的“上控制限”。
<b>CurrentSample</b>	Real	只读	显示最后一个样本点的值(即 X、C、P)。
<b>CurrentSampleBar</b>	Real	可读写	设置此样本点的当前样本平均值。
<b>CurrentSampleNumber</b>	Integer	只读	显示采集的最后一个样本号。
<b>CurrentTarget</b>	Real	可读写	设置此样本点的目标值。
<b>CurrentTime</b>	Message(8)	可读写	按 HH:MM:SS 格式设置当前样本的时间。如果输入不正确，则缺省使用当前时间。
<b>CurrentUpdate</b>	Discrete	可读写	要更改在任何当前字段中输入的关于样本的信息，将此项目设置为 1。
<b>CurrentXLCL</b>	Real	可读写	设置当前样本的“下控制限”(LCL)。
<b>CurrentXLSL</b>	Real	可读写	设置当前样本的“下规格限”(LSL)。



项目名	DDE 类型	访问权限	描述
<b>CurrentXUCL</b>	Real	可读写	设置当前样本的“上控制限”(UCL)。
<b>CurrentXUSL</b>	Real	可读写	设置当前样本的“上规格限”(USL)。
<b>SPC2L3Out2SD</b>	Integer	只读	“2 个样本 (3 个最新样本中), 超过 2 标准偏差 (同侧)”报警的报警计数器。
<b>SPC2L3Out2SDMsg</b>	Message	只读	“2 个样本 (3 个最新样本中), 超过 2 标准偏差 (同侧)”的报警消息标记。
<b>SPC4L5Out1SD</b>	Integer	只读	“4 个样本 (5 个最新样本中), 超过 1 标准偏差 (同侧)”报警的报警计数器。
<b>SPC4L5Out1SDMsg</b>	Message	只读	“4 个样本 (5 个最新样本中), 超过 1 标准偏差 (同侧)”的报警消息标记。
<b>SPCConSampAltUpDn</b>	Integer	只读	“上下交替的连续样本”报警的报警计数器。
<b>SPCConSampAltUpDnMsg</b>	Message	只读	“上下交替的连续样本”的报警消息标记。
<b>SPCConSampIn1SD</b>	Integer	只读	“在 1 个标准偏差内的连续样本”报警的报警计数器。
<b>SPCConSampIn1SDMsg</b>	Message	只读	“在 1 个标准偏差内的连续样本”的报警消息标记。
<b>SPCConSampIncDec</b>	Integer	只读	“递增或递减的连续样本”报警的报警计数器。
<b>SPCConSampIncDecMsg</b>	Message	只读	“递增或递减的连续样本”的报警消息标记。
<b>SPCConSampOneSideCL</b>	Integer	只读	“在中线的一侧的连续样本”报警的报警计数器。
<b>SPCConSampOneSideCLMsg</b>	Message	只读	“在中线的一侧的连续样本”的报警信息标记。
<b>SPCConSampOut1SD</b>	Integer	只读	“超过 1 个标准偏差的连续样本”报警的报警计数器。
<b>SPCConSampOut1SDMsg</b>	Message	只读	“超过 1 个标准偏差的连续样本”的报警消息标记。

项目名	DDE 类型	访问权限	描述
SPCNLNOOutNSD	Integer	只读	“? 个样本 (? 个最新样本数中), 超出 ? 标准偏差”报警的报警计数器。
SPCNLNOOutNSDMsg	Message	只读	“? 个样本 (? 个最新样本数中), 超出 ? 标准偏差”的报警消息标记。
SPCNLNOOutNSDSS	Integer	只读	“? 个样本 (? 个最新样本数中), 超出 ? 标准偏差 (同侧)”报警的报警计数器。
SPCNLNOOutNSDSSMsg	Message	只读	“? 个样本 (? 个最新样本数中), 超出 ? 标准偏差 (同侧)”的报警消息标记。
SPCOutRCtrl	Integer	只读	“范围超出控制限制”范围图表报警的报警计数器。
SPCOutRCtrlMsg	Message	只读	“范围超出控制限制”范围图表报警的报警消息。
SPCOutXCtrl	Integer	只读	“超出控制限制的样本” X 图表报警的报警计数器。
SPCOutXCtrlMSG	Message	只读	“超出控制限制的样本” X 图表报警的报警消息。
SPCOutSpec	Integer	只读	“超出规格限制的样本”报警的报警计数器。
SPCOutSpecMsg	Message	只读	“超出规格限制的样本”的报警消息标记。
SPCRecalculateCp	Discrete	可读写	<p>设置为 1 时, 则在采集到下一样本时, 引擎重新计算当前数据集的 Cp 与 Cpk 值。采集此样本之后, 此项目的值重置为 0。要再次重新计算 Cp 与 Cpk, 请再次将此值设置为 1。</p> <p>此项目“仅仅”影响 Cp 与 Cpk 值。它不影响控制限或“运行规则”。</p>
SPCResetAlarmCounters	Discrete	可读写	重置所有的报警计数器。

项目名	DDE 类型	访问权限	描述
<b>SPCResetRunRules</b>	Discrete	可读写	用于重置给新采集的样本应用的运行规则。应用于当前采集的产品。打开时，运行规则将会重置，先前的样本将不会用在“4 个样本（5 个最新样本中），超过 1 标准偏差”之类的报警计算中。此操作仅执行一次，运行规则随后恢复正常。要再次执行重置操作，必须重置然后再次打开此项目。
<p>“手工输入” DDE 项目用于创建自定义的手工输入窗口。要使用手工输入项目，请设置适当项目的值，然后将 DDE 项目 <b>MI_Save</b> 设置为 1。这将导致其它 MI 字段中的信息作为新样本进行输入。输入样本之后，SPCPro 程序将 DDE 项目 <b>MI_Save</b> 重置为 0。</p> <p>对于分布式 SPC，手工输入 DDE 项目是每个节点内私有的。这些值在每个节点本地缓冲，直到 DDE 项目 <b>MI_Save</b> 设置为 1。<b>MI_Save</b> 设置为 1 之后，这些值放入手工输入数据包，并发送到远程数据集节点进行分析与储存。</p> <p>在添加显示与采集产品之后，所有的“手工” SPC DDE 项目只应用于采集的产品。</p>			
项目名	DDE 类型	访问权限	描述
<b>MI_CauseCode</b>	Integer	只写	为手工输入的样本设置特殊原因代码编号。
<b>MI_CauseString</b>	Message(127)	只读	显示为样本输入的特殊原因代码编号的描述。
<b>MI_Comment</b>	Message(50)	只写	用于读取 / 写入为样本输入的任何其它注释。
<b>MI_Date</b>	Message(10)	只写	设置当前样本的“日期”。“日期”必须按 DD/MM/YY 或 DD/MM/YYYY 格式输入。如果输入不正确，“日期”将缺省使用当前“日期”。
<b>MI_Flag</b>	Discrete	只写	设置手工输入样本的标识。
<b>MI_IgnoreValue</b>	Discrete	只写	设置在“控制图”为“自动缩放”模式时忽略当前样本。
<b>MI_Mx</b>	Real	只写	为指定的手工输入测量值设置值（ $x = 1$ 到 25）。

项目名	DDE 类型	访问权限	描述
MI_Save	Discrete	只写	<p>将其它 MI 字段中手工输入的信息保存为新样本。</p> <p>MI_Save 项设置为 1 时，所有 MI 项的值会写入相应的“当前”DDE 项目，并且 CurrentSampleNumber 项的索引为 1。</p> <p>再次使用 MI_Save 将产生与上一次使用相同的样本时间。</p>
MI_Time	Message(8)	只写	<p>设置当前样本的“时间”。“时间”必须按 HH:MM:SS 格式输入。如果输入不正确，“时间”将缺省使用当前时间。</p>
<p>各个 Selection “DDE 项目”可用于查看任何样本的有关详细信息。Selection “DDE 项目”用于输入要显示的样本的编号。输入样本号之后，SPCPro 使用该样本的详细信息更新所有其它 Selection 项。</p> <p>旧的数据无法更改。但是通过设置适当的项目，然后将 SelectionUpdate 项设置为 1，便可以添加特殊原因代码、标帜以及注释。这会导致使用新的值来修改所选的样本记录。输入更新的样本之后，SPCPro 将 DDE 项目 SelectionUpdate 重置为零。</p> <p>对于分布式 SPC，所选的样本 DDE 项目是每个节点内部私有的。它们是远程节点为采集的产品的指定样本号记录的样本值。Selection DDE 项目设置为样本号时，可以从远程节点的样本文件中检索样本信息。</p> <p>旧的数据无法更改，但是通过设置适当的 DDE 项目，然后将 SelectionUpdate 项目设置为 1，便可以添加“特殊原因代码”、“标帜”以及“注释”。SelectionUpdate 设置为 1 时，“特殊原因代码”、“注释”、“标帜”以及“忽略值”项目在数据包中发送到远程节点进行储存。</p> <p><b>备注</b> 在添加显示与采集的产品之后，所有的 "Selection" SPC DDE 项目都仅应用于所采集的产品。</p>			

项目名	DDE 类型	访问权限	描述
Selection	Integer	可读写	将此项目设置为样本号时，将使用该样本的适当数据更新所有的选择项目。
SelectionCauseCode	Integer	可读写	为所选的样本设置特殊原因代码编号。

项目名	DDE 类型	访问权限	描述
<b>SelectionCauseString</b>	Message (128)	只读	显示为特殊原因代码输入的描述。
<b>SelectionComment</b>	Message (50)	可读写	用于读取 / 写入为所选的样本输入的任何其它注释。
<b>SelectionCp</b>	Real	只读	显示所选样本的容量。
<b>SelectionCpk</b>	Real	只读	显示所选样本的中心容量。
<b>SelectionDate</b>	Message (10)	只读	显示所选样本的日期。
<b>SelectionFlag</b>	Discrete	可读写	设置所选样本的“标识”。
<b>SelectionIgnoreValue</b>	Discrete	可读写	设置在“控制图”为“自动缩放”模式时忽略所选样本。
<b>SelectionMx</b>	Real	只读	显示组成样本的各个测量值 ( $x=1-25$ ) 的值。
<b>SelectionProduct</b>	Message (32)	只读	显示所选样本的“产品名”。
<b>SelectionRUCL</b>	Real	只读	显示所选样本的范围 UCL。
<b>SelectionRLCL</b>	Real	只读	显示所选样本的范围 LCL。
<b>SelectionR</b>	Real	只读	显示所选样本的范围。
<b>SelectionRBAR</b>	Real	只读	显示所选样本的平均范围。
<b>SelectionSample</b>	Real	只读	显示所选样本点的值。
<b>SelectionSampleBar</b>	Real	只读	显示所选样本点的所选样本平均值。
<b>SelectionTarget</b>	Real	只读	显示所选样本的目标值。
<b>SelectionTime</b>	Message (8)	只读	显示所选样本的时间。
<b>SelectionUpdate</b>	Discrete	可读写	更新各个 Selection 字段中的更改。
<b>SelectionXUSL</b>	Real	只读	显示样本的“上规格限”。
<b>SelectionXLSL</b>	Real	只读	显示样本的“下规格限”。
<b>SelectionXUCL</b>	Real	只读	显示样本的“上控制限”。
<b>SelectionXLCL</b>	Real	只读	显示样本的“下控制限”。

项目名	DDE 类型	访问权限	描述
<b>SeISPC2L3Out2SDMsg</b>	Message	只读	“2 个样本（3 个最新样本中），超过 2 标准偏差（同侧）”的报警消息标记。
<b>SeISPC4L5Out1SDMsg</b>	Message	只读	“4 个样本（5 个最新样本中），超过 1 标准偏差（同侧）”的报警消息标记。
<b>SeISPCConSampAltUpDnMsg</b>	Integer	只读	“上下交替的连续样本”报警的报警消息。
<b>SeISPCConSampIn1SDMsg</b>	Message	只读	“在 1 个标准偏差内的连续样本”的报警消息标记。
<b>SeISPCConSampIncDecMsg</b>	Message	只读	“递增或递减的连续样本”的报警消息标记。
<b>SeISPCConSampOneSideCLMsg</b>	Message	只读	“在中线的一侧的连续样本”的报警信息标记。
<b>SeISPCConSampOut1SDMsg</b>	Message	只读	“超过 1 个标准偏差的连续样本”的报警消息标记。
<b>SeISPCNLNOutNSDMsg</b>	Message	只读	“？ 个样本（？ 个最新样本数中），超出？ 标准偏差”的报警消息标记。
<b>SeISPCNLNOutNSDSSMsg</b>	Message	只读	“？ 个样本（？ 个最新样本数中），超出？ 标准偏差（同侧）”的报警消息标记。
<b>SeISPCOutRCtrlMsg</b>	Message	只读	“范围超出控制限制”范围图表报警的报警消息。
<b>SeISPCOutXCtrlMsg</b>	Message	只读	“超出控制限制的样本”X 图表报警的报警消息。
<b>SeISPCOutSpecMsg</b>	Message	只读	“超出规格限制的样本”的报警消息标记。

相同的实型数据集可以配置与链接多个“间接数据集”。随后，可以将每个“间接数据集”的 **Selection** 值设置为不同的样本号。这样便可以在一个数据集中查看多个样本的详细信息。

## 在运行时添加或删除数据集

您可以创建包含 `SPCDatasetDlg()` 函数的 QuickScript，以添加或删除数据集。

### SPCDatasetDlg() 函数

`SPCDatasetDlg()` 函数在 WindowViewer 中显示 **SPCPro 数据集配置**对话框。从该对话框中，可以添加新的数据集，或是删除现有的数据集。此函数没有参数或返回值。

在 WindowViewer 运行期间，当前数据集与产品信息变灰失效，因此无法更改这些选项。

如果在“自动采集”模式下运行，则会重新开始“自动采集”周期。重新开始“自动采集”周期可能会导致在数据集的初始化阶段丢失数据。

---

**备注** 建议在运行期间使用 **NewProduct** 这个“SPC DDE 项目”来添加“产品”，而不要使用 `SPCDatasetDlg()` 函数。这可以防止重新启动“自动采集”模式。

---

**类别**  
SPC

**语法**

```
SPCDatasetDlg();
```

**附注**

运行包含 `SPCDatasetDlg()` 函数的脚本时，会在 WindowViewer 中打开 **SPCPro 数据集配置**对话框。

**示例**

```
SPCDatasetDlg();
```

**另请参阅**

`SPCSelectDataset()`

## 更改图表使用的数据集

通过 DDE 或 SPC 函数，可以在运行时将“间接数据集”重新指定给不同的数据集。这样便可以创建单个“SPC 图表”来显示任何配置过的数据集。

**要更改图表的数据集**

- 1 创建一个 I/O 消息标记。例如，**Indirect\_DatasetName**。
- 2 通过将 **SPCPro** 用作应用程序名、将配置的有效间接数据集名用作主题名，从而将此标记关联到“SPCPro 访问名”。

---

**备注** 数据集名区分大小写。如果 DSN 的名称是 **Indirect**，并且在“访问名”中输入 **indirect** 作为“主题”，则该“访问名”的任何 I/O 项都将无法正常工作。

---

3 在“标记名字典”的项目框中，输入 SPC DDE  
DatasetName 项目名。

4 创建一个 InTouch QuickScript 来更改数据集名。例如：

```
Indirect_DatasetName = "SPC1";
```

或者，创建以下代码，使用户可以选择数据集名。例如：

```
Indirect_DatasetName = SPCSelectDataset();
```

其中：SPC1 是一个有效的数据集名。

运行此 QuickScript 之后，指定的数据集名（此处是 SPC1）写入 I/O 消息标记 Indirect\_DatasetName。“SPC 图表”显示指定的数据集名的配置设置，包括控制限以及最后采集的样本。最后采集的样本是图表中显示的最后一个样本。

### SPCSelectDataset() 函数

SPCSelectDataset() 函数打开**选择数据集**对话框。用户选择直接数据集之后，此函数将名称指定给 DatasetName 标记。

**类别**

SPC

**语法**

```
DatasetName=SPCSelectDataset()
```

**附注**

选择数据集名之后，此函数将名称指定给 *DatasetName* 标记。这种选择方式也可用于更改“间接数据集”的 *DatasetName*。

**另请参阅**

SPCSelectProduct(), SPCDatasetDlg()

## 更改图表中显示的产品

在运行时，可以动态更改 SPCPro 图表中显示的产品。在事件驱动的 QuickScript 中，SPCSelectProduct() 函数可用于打开一个对话框，供运行时用户选择产品。

**要更改数据集中的产品**

1 创建一个 I/O 消息标记。例如，ProductDisplayed。

2 通过将 SPCPro 用作应用程序名、将配置的有效数据集名用作主题名，从而将此标记关联到“InTouch 访问名”。

3 在“标记名字典”的项目框中，给该 I/O 消息标记输入 ProductDisplayed SPC DDE 项目名。

4 创建一个 InTouch QuickScript 来更改“产品名”。例如：

```
ProductDisplayed = "Product1";
```

在本例中，Product1 是数据集中定义的一个有效的产品名，它指定为“InTouch 访问名”中的主题名。

或者，创建以下脚本使用户可以选择“产品名”。例如：



```
ProductDisplayed = SPCSelectProduct("Dataset");
```

运行此 QuickScript 之后，SPC 图表显示指定的产品。

### SPCSelectProduct() 函数

SPCSelectProduct() 函数打开**选择产品**对话框，以便从给定的数据集中选择产品。

**类别**  
SPC

**语法**

```
ProductName=SPCSelectProduct("Dataset");
```

**参数**

*Dataset*

包含实际的数据集名。实际的字符串或消息标记。

**附注**

选择产品名之后，此函数将它返回给 ProductName 标记。现在此名称可用于更改 *Dataset* 中采集的产品。

**另请参阅**

SPCSelectDataset(), SPCSetProductDisplayed(),  
SPCSetProductCollected()

### SPCSetProductDisplayed() 函数

SPCSetProductDisplayed() 函数更改指定的数据集中正在显示的产品。

**类别**  
SPC

**语法**

```
SPCSetProductDisplayed("Dataset","Product");
```

**参数**

*Dataset*

包含实际的数据集名。实际的字符串或消息标记。

*Product*

包含将显示其数据的产品名。实际的字符串或消息标记。

**示例**

```
SPCSetProductDisplayed("ADatasetName",  
    "AProductName");
```

**另请参阅**

SPCSelectProduct(), SPCSetProductCollected()

## 滚动图表

您可以创建一个按钮并给它附加一个**触动按钮 - 动作脚本**，以便在当前数据集的历史数据中向前或向后滚动。通过在链接到所选对象的脚本中更改 `LastSampleDisplayed` 这个“SPC DDE 项目”的值，可以实现这个目的。另一个“SPC DDE 项目”`SamplePerControlChart` 可用于控制在图表中显示的数据量。

---

**备注** “限制向导”可用于在当前数据集的历史数据中向前或向后滚动。

---

### 要在历史数据中向后滚动

- 1 创建一个图形对象（如按钮）。
- 2 将以下**触动按钮 - 动作脚本**附加到该按钮：

```
LastSampleDisplayed = LastSampleDisplayed -  
SamplesPerControlChart;
```

在运行时，当操作员单击该按钮时，脚本从当前的 `LastSampleDisplayed` 编号中减去 `SamplesPerControlChart` 的值。此时图表向后滚动，以便将产生的样本号显示为图表中的最后一个样本。

例如，如果当前 `LastSampleDisplayed` 是编号 860，`SamplePerControlChart` 项目的值是 20，则图表向后滚动 20 个样本。图表将编号为 821 的样本显示为第一个样本，编号 840 显示为图表的最后一个样本。

### 要在历史数据中向前滚动

- 1 创建一个图形对象（如按钮）。
- 2 将以下**触动按钮 - 动作脚本**附加到该对象：

```
LastSampleDisplayed = LastSampleDisplayed +  
SamplesPerControlChart;
```

在运行时，当操作员单击按钮时，该脚本将 `SamplePerControlChart` 的值加到当前的 `LastSampleDisplayed` 编号上。此时图表向前滚动，以便产生的样本号显示为图表中的最后一个样本。例如，如果当前 `LastSampleDisplayed` 是编号 860，`SamplePerControlChart` 项目的值是 20，则图表向前滚动 20 个样本。图表将编号为 861 的样本显示为第一个样本，编号 880 的样本显示为最后一个样本。

---

**备注** 只要 SPC 图表未显示当前样本号，图表中便会出现“历史”字样。

---

在 QuickScript 中可以使用三个 SPC 函数来选择图表中当前显示的数据。下表列出这些函数。

SPCDisplayData()	将图表滚动到指定的日期与时间。
SPCLocateScooter()	将图表指示器移到任何有效的样本号。
SPCMoveScooter()	按给定的样本数滚动指示器。

SPCDisplayData() 函数

SPCDisplayData() 函数可以在脚本中用于滚动到图表中指定的日期与时间。您可以使用标记来监视 SPC 数据搜索的状态。如果 SPC 找到指定时间段内的数据，则该标记的值指定为 0；如果找不到数据，则指定为 1。

类别  
SPC

语法

```
[Status=]SPCDisplayData("Dataset", "DateString",  
                        "TimeString", RangeInHours);
```

参数

- Dataset*  
实际的数据集名。实际的字符串或消息标记。
- DateString*  
mm/dd/yyyy 格式的日期。实际的字符串或消息标记。
- TimeString*  
hh:mm:ss 格式的时间。实际的字符串或消息标记。
- RangeInHours*  
要在图表中显示的数据的小时数。任何整数或标记。

示例

下例显示自 2007 年 5 月 16 日上午 9:15 开始的当前产品 Dataset1 一个小时的数据。

```
StatusTag = SPCDisplayData("Dataset1",  
                          "05/16/2007", "09:15:00", 1);
```

SPCLocateScooter() 函数

SPCLocateScooter() 函数可以在脚本中用于按指定的样本数移动图表“指示器”。数据集中定义的“指示器”标记使用“X-条形图”样本值进行更新。将 SampleNumber 设置为 0，可以隐藏 / 禁用“指示器”。

类别  
SPC

**语法**

```
SPCLocateScooter( "Dataset", SampleNumber);
```

**参数***Dataset*

实际的数据集名。实际的字符串或消息标记。

*SampleNumber*

有效的样本数。任何数字或整型标记。

**另请参阅**

SPCMoveScooter()

**SPCMoveScooter() 函数**

SPCMoveScooter() 函数可以在脚本用于将“指示器”移到图表中指定的样本号。数据集中定义的“指示器”标记使用“X-条形图”样本值进行更新。

**类别**

SPC

**语法**

```
SPCMoveScooter( "Dataset", IncrementValue);
```

**参数***Dataset*

实际的数据集名。实际的字符串或消息标记。

*IncrementValue*

图表中要滚动的样本数。使用正数向前滚动，使用负数向后滚动。任何数字或整型标记。

**另请参阅**

SPCLocateScooter()

## 将图表更新到当前时间

您可以使操作员能够在运行时将当前样本数据加载到“SPC 控制图”。在 InTouch 应用程序中创建一个可选的对象，并附加一个使用当前数据来更新图表的脚本。

**要使用当前数据填充图表**

**1** 创建一个图形对象（如按钮）。

**2** 将以下**触动按钮 - 动作**脚本附加到该图形对象：

```
LastSampleDisplayed = CurrentSampleNumber;
```

在本例中，CurrentSampleNumber 是数据集的 SPC DDE 项目。

在运行时，当操作员单击该按钮时，该脚本将

LastSampleDisplayed 项目设置为等于

CurrentSampleNumber。此时会根据

SamplesPerControlChart 项目与 CurrentSampleNumber 的值，使用先前的样本来填充图表。例如，如果 CurrentSampleNumber 值是 860，SamplesPerControlChart 的值是 20，则图表将编号为 841 的样本显示为第一个样本，编号为 860 的样本显示为最后一个样本。

#### 要清除图表（不显示任何样本）

您可以将 LastSampleDisplayed 这个“DDE 项目”设置为零，或是设置为小于数据库中第一个样本号的任何数字。这样会清除图表。此外，如果操作员向后滚动到第一个样本之前，则图表也会自动清除。

## 输入新样本

您可以创建一个动作 QuickScript，在应用程序运行期间将数据手工输入到 SPCPro 数据集。此时可以使用两个 SPCPro 函数来设置样本值，然后将它们保存到数据集：

- SPCSetMeasurement() 函数
- SPCSaveSample() 函数

### SPCSetMeasurement() 函数

SPCSetMeasurement() 可以在脚本中用于通过手工或事件驱动的方式将值输入到数据集测量号。

#### 类别

SPC

#### 语法

```
SPCSetMeasurement ("Dataset", Measurement, Value);
```

#### 参数

##### *Dataset*

实际的数据集名。字符串或消息标记。

##### *Measurement*

测量号（1 到 300）。任何数字或整型标记。

##### *Value*

写入指定的测量号的值。任何整型或实型标记。

#### 附注

设置所有的测量值之后，使用 SPCSaveSample() 函数将数据保存到数据库。您必须重新启动 DDE 对话来更新 MI\_Mx 标记的值。

#### 另请参阅

SPCSaveSample()

### SPCSaveSample() 函数

SPCSaveSample() 函数保存手工输入的样本。此函数与 SPCSetMeasurement() 函数配合使用。

类别  
SPC

语法

```
SPCSaveSample ("Dataset");
```

参数

*Dataset*

实际的数据集名。字符串或消息标记。

附注

执行此函数会强制将样本输入到指定的数据集。它将手工输入变量的当前值用作样本中的测量值。输入值可以通过 DDE 标记 MI\_Mx (x = 从 1 到 25 的测量号)，或是使用 SPCSetMeasurement() 函数 (测量值 1 到 300) 进行设置。您必须重新启动 DDE 对话来更新 MI\_Mx 标记的值。

另请参阅

SPCSetMeasurement()

## 检查删除的样本

您可以使用 SPCSampleDeleted() 函数来确定某个样本是否已删除。

### SPCSampleDeleted() 函数

SPCSampleDeleted() 函数从数据集中返回指定样本的状态。

类别  
SPC

语法

```
Result=SPCSampleDeleted ("Dataset", "Product",  
    SampleNum);
```

参数

*Dataset*

实际的数据集名。字符串或消息标记。

*Product*

产品名。字符串或消息标记。

*SampleNum*

样本号。任何数字或整型标记。

**返回值**

-1 = 无效的数据集名、产品名或样本号。

0 = 样本未删除。

1 = 样本已删除。

## 更改当前采集的产品

您可以创建包含 `SPCSetProductCollected()` 函数的 **QuickScript**，以更改数据集中正在采集数据的产品。此外也可以使用数据集的 DDE 项目 `ProductCollected`。

**要更改数据集产品**

- 1 创建一个 I/O 消息标记。例如，`ProductCollected`。
- 2 通过将 `SPCPro` 用作应用程序名、将配置的有效数据集名用作主题名，从而将此标记关联到 “InTouch 访问名”。
- 3 在 “标记名字典” 的 **项目** 框中，为 `ProductCollected` 标记输入 `SPC DDE ProductCollected` 项目名。

- 4 创建一个 InTouch **QuickScript** 来更改 “产品名”。例如：

```
ProductCollected = "Product1";
```

在本例中，`Product1` 是数据集中定义的一个有效 “产品名”，它指定为 “InTouch 访问名” 中的主题名。

或者，创建以下脚本使用户可以选择 “产品名”。例如：

```
ProductCollected = SPCSelectProduct("Dataset");
```

运行此 **QuickScript** 之后，指定的 “产品” 名写入 I/O 消息标记 `ProductCollected`。

### SPCSetProductCollected() 函数

SPCSetProductCollected() 函数更改指定的数据集中正在采集的产品。

类别  
SPC

语法

```
SPCSetProductCollected("Dataset", "Product");
```

参数

*Dataset*

实际的数据集名。字符串或消息标记。

*Product*

要采集其数据的产品名。字符串或消息标记。

附注

SPCSetProductCollected() 函数不更改图表中显示的产品。通过使用此函数来采集数据，同时使用

SPCSetProductDisplayed() 函数来显示采集的数据，可以实现采集一个产品的数据，而显示另一个产品的数据。

示例

```
SPCSetProductCollected("Data5838", "Widgets");
```

另请参阅

SPCSelectProduct(), SPCSetProductDisplayed()



## 在运行时创建或删除产品

SPCPro 程序提供可用于在现有的数据集中创建或删除产品的 DDE 项目与函数。创建新产品名时，也可以为现有产品的新批号创建单独的产品文件。

- NewProduct DDE 项目
- SPCCreateNewProduct() 函数
- SPCDeleteProduct() 函数

### NewProduct DDE 项目

通过设置 SPC DDE 项目 NewProduct，可以创建新的产品。

#### 要在运行时创建新产品

- 1 创建一个 I/O 消息标记。例如，NewProduct。
- 2 通过将 SPCPro 用作应用程序名、将有效的数据集名用作主题名，从而将此标记关联到 “InTouch 访问名”。
- 3 在 “标记名字典” 的项目框中，为 NewProduct 标记输入 NewProduct SPC DDE 项目名。

- 4 创建一个 InTouch QuickScript 来更改 “产品名”。例如：

```
NewProduct= "Product2";
```

在本例中，Product2 不是数据集（在 “InTouch 访问名” 中指定为主题名）中定义过的现有产品名。

- 5 通过使用以下这些 DDE 项目，设置新产品、“控制图”、“巴累托图” 以及 “直方图” 的 “标题”：

```
NewProductCtrlTitle
NewProductParetoTitle
NewProductHistTitle
```

如果在调用 NewProduct 之前设置上述项目，则新创建的产品会使用新的标题名。例如：

```
NewProductCtrlTitle = "Product2";
NewProductParetoTitle= "Product2";
NewProductHistTitle= "Product2";
NewProduct= "Product2";
```

运行此脚本时，指定的产品名写入 I/O 消息标记 NewProduct。SPCPro 自动将 SPC DDE 项目 ProductCollected 的值设置为 NewProduct 的值。

### SPCCreateNewProduct() 函数

SPCCreateNewProduct() 函数创建新的产品。

类别  
SPC

语法

```
SPCCreateNewProduct ("Dataset", "Product",  
    MeasPerSample);
```

参数

*Dataset*

数据集名。实际的字符串或消息标记。

*Product*

要创建的产品的名称。实际的字符串或消息标记。

*MeasPerSample*

产品的每个样本的测量值数。任何数字或整型标记。

另请参阅

SPCDeleteProduct()

### SPCDeleteProduct() 函数

SPCDeleteProduct() 函数从 SPC 数据库中删除所有的产品信息，包括产品的测量值与样本。

类别  
SPC

语法

```
SPCDeleteProduct ("Dataset", "Product");
```

参数

*Dataset*

数据集名。实际的字符串或消息标记。

*Product*

要删除的产品的名称。实际的字符串或消息标记。

附注

如果当前在采集或显示该产品，则无法删除它。此时会有一条消息记录到 Log Viewer。

另请参阅

SPCCreateNewProduct()

## 为各种图表类型输入计算参数

您可以创建 QuickScript 来输入 SPCPro 控制、范围或规格限的计算值。使用这些函数来输入计算参数：

- SPCSetControlLimits() 函数
- SPCSetRangeLimits() 函数
- SPCSetSpecLimits() 函数
- SPCSetProductControlLimits() 函数
- SPCSetProductRangeLimits() 函数
- SPCSetProductSpecLimits() 函数

### SPCSetControlLimits() 函数

SPCSetControlLimits() 可以在脚本中用于通过手工或事件方式为“控制图”输入控制限值。

#### 类别

SPC

#### 语法

```
SPCSetControlLimits ("Dataset", XUCL, XLCL);
```

#### 参数

##### *Dataset*

实际的数据集名。字符串或消息标记。

##### *XUCL*

用于图表的 UCL 的值。数字或实型标记。

##### *XLCL*

用于图表的 LCL 的值。数字或实型标记。

#### 附注

通过执行 SPCSaveSample(), 可以将这些测量值保存到样本中。

#### 另请参阅

SPCSaveSample(), SPCSetRangeLimits(), SPCSetSpecLimits(), SPCSetMeasurement()

### SPCSetRangeLimits() 函数

SPCSetRangeLimits() 函数可以在脚本中用于设置“范围图表”的“控制限”。SPCSetRangeLimits() 函数可用于通过手工或事件驱动的方式为“范围图表”输入“控制限”。

**类别**

SPC

**语法**

```
SPCSetRangeLimits("Dataset", RUCL, RLCL);
```

**参数**

*Dataset*

实际的数据集名。字符串或消息标记。

*RUCL*

用于“范围图表”的 UCL 的值。数字或实型标记。

*RLCL*

用于“范围图表”的 LCL 的值。数字或实型标记。

**另请参阅**

SPCSetControlLimits(), SPCSetSpecLimits()

### SPCSetSpecLimits() 函数

SPCSetSpecLimits() 函数可以在脚本中用于通过手工或事件驱动的方式为“控制图”输入规格限值。

**类别**

SPC

**语法**

```
SPCSetSpecLimits("Dataset", XUSL, XLSL);
```

**参数**

*Dataset*

实际的数据集名。字符串或消息标记。

*XUSL*

用于图表的 USL 的值。数字或实型标记。

*XLSL*

用作图表 LSL 的值。数字或实型标记。

**另请参阅**

SPCSetControlLimits(), SPCSetRangeLimits()

### SPCSetProductControlLimits() 函数

SPCSetProductControlLimits() 函数可以在脚本中用于为空产品的“控制图”输入控制限值。

**类别**  
SPC

**语法**

```
SPCSetProductControlLimits ("Dataset", "Product",  
    XUCL, XLCL);
```

**参数**

*Dataset*

实际的数据集名。字符串或消息标记。

*Product*

要设置其控制限的产品的名称。字符串或消息标记。

*XUCL*

用作图表 UCL 的值。数字或实型标记。

*XLCL*

用作图表 LCL 的值。数字或实型标记。

**附注**

通过执行 SPCSaveSample() 函数，可以将这些测量值保存到样本中。

**另请参阅**

SPCSaveSample(), SPCSetProductRangeLimits(),  
SPCSetProductSpecLimits(),

### SPCSetProductRangeLimits() 函数

SPCSetProductRangeLimits() 函数可以在脚本中用于为空产品的“范围图表”设置“控制限”。

**类别**  
SPC

**语法**

```
SPCSetProductRangeLimits ("Dataset", "Product",  
    UCL, LCL);
```

**参数**

*Dataset*

实际的数据集名。字符串或消息标记。

*Product*

要设置其范围限制的产品的名称。字符串或消息标记。

*RUCL*

用作“范围图表” UCL 的值。数字或实型标记。

***RLCL***

用作“范围图表” LCL 的值。数字或实型标记。

**另请参阅**

SPCSetProductControlLimits(), SPCSetProductSpecLimits()

**SPCSetProductSpecLimits() 函数**

SPCSetProductSpecLimits() 函数可以在脚本中用于为空产品的“控制图”设置规格限值。

**类别**

SPC

**语法**

```
SPCSetProductSpecLimits ("Dataset", "Product", XUSL,  
                          XLSL);
```

**参数*****Dataset***

实际的数据集名。字符串或消息标记。

***Product***

要设置其规格限值的产品的名称。字符串或消息标记。

***XUSL***

用作图表 USL 的值。数字或实型标记。

***XLSL***

用于图表的 LSL 的值。数字或实型标记。

**另请参阅**

SPCSetProductControlLimits(),  
SPCSetProductRangeLimits()

## 将数据输入属性图表

使用以下 DDE 项目将数据输入属性图表：

图表类型	使用的 DDE 项目	注释
C 图表	MI_M1	输入拒绝次数。
P 图表	MI_M1 与 MI_M2	输入拒绝次数与样本大小。 刻度范围是 0.00 到 1.00。 例如， 0.50 = 50% 拒绝。
NP 图表	MI_M1	输入缺陷数目（在“数据集配置”对话框中设置样本大小）。
U 图表	MI_M1 与 MI_M2	在 MI_M1 中输入缺陷数， 在 MI_M2 中输入样本大小。

## 控制配置选项

SPCDatasetDlg() 函数在 WindowViewer 中显示 **SPCPro 数据集配置**对话框。在运行时，可以在 **SPCPro 数据集配置**对话框中添加新的数据集，或是删除现有的数据集。如需有关详细信息，请参阅第 55 页的“SPCDatasetDlg() 函数”。

## 从以前的 SPCPro 版本移植 SPC 数据库

您可以使用 SPCPro Server 实用程序来执行所有的 SPCPro 功能。例如，配置数据库、配置用户，以及创建数据集和“间接数据集”。通过使用 SPCPro Server 实用程序，便不必从 WindowMaker 中配置 SPC。

您必须创建新的数据库来存储转换后的数据集文件。创建的数据库必须为空。您只能将现有的数据集导入空的数据库。在将数据集信息存储到数据库之前，必须在“标记名字典”中定义标记。

### 要导入 SPC 数据集

- 1 如果需要，停止 WindowMaker 与 WindowViewer。
- 2 从命令提示符或 Windows 的“资源管理器”中启动 SPCPro 实用程序。此时出现 SPCPro Server 对话框。
- 3 在数据库菜单上，单击导入。此时出现打开对话框。
- 4 双击要转换其数据集的 SPCPro 应用程序所在的应用程序目录，或是选择该目录并单击打开。此时出现导入数据集对话框。
- 5 在导入选项区域中，选择选择全部数据集选项，以便从所选的应用程序导入所有的数据集；或是选择选择全部产品，以便从所选的应用程序导入所有的产品；或是从左侧的列表中选择数据集与产品。
- 6 在导入选项区域的第二个导入选项组中，选择要导入的样本，具体如下：

不导入样本	仅导入数据集配置。
导入全部样本	导入数据集中的所有样本。
导入样本范围	仅导入指定的起始日期与结束日期之间的样本。

- 7 单击导入。此时导入数据集并将它转换成新的 SPCPro 格式，并且可以在 SPCPro 中查看。

在运行 InTouch 应用程序之前，必须将 SPCPro 6.0 或更早版本所使用的数据集转换成新的格式。

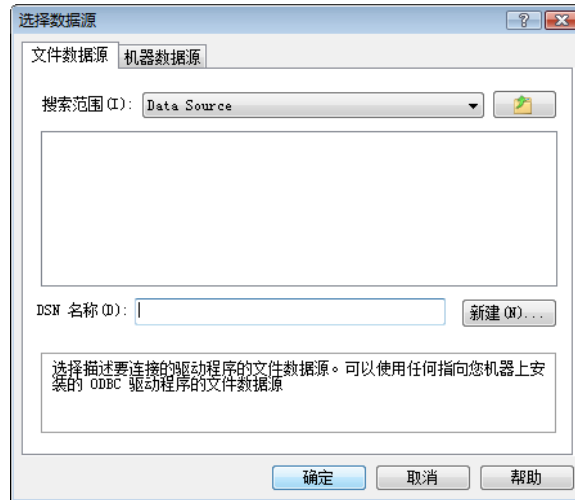
要将 SPCPro 数据库升级到 SPCPro 支持的当前版本，可以使用 SPCPro 实用程序。SPCPro 实用程序与 InTouch 安装在相同的文件夹中。

升级数据库时，SPCPro 实用程序会创建数据库的备份。此备份作为文件保存在与数据库所在的相同文件夹。确认升级后的数据库正常工作之后，可以删除数据库备份文件。



### 要将 SPCPro 数据库升级到更新的版本

- 1 确认 SPCPro 不在运行。
- 2 从 Windows 的“资源管理器”或是 Windows 命令提示符中运行 spcutil.exe 命令。此时出现 spcutil 对话框。
- 3 在文件菜单上，单击**转换架构**。此时出现**选择数据源**对话框。



- 4 通过以下方法之一来选择数据源名：
  - 在 **DSN 名称** 框中，输入指向要转换的数据库的数据源名，然后单击**确定**。
  - 单击**机器数据源**选项卡，从列表中选择数据源名，然后单击**确定**。

此时出现 **SQL Server 登录**对话框。



- 5 登录到 SQL Server。
  - a 在**登录 ID** 框中，输入 SQL Server 管理员的登录 ID。
  - b 在**密码**框中，输入管理员的密码。
  - c 单击**确定**。

此时 SPCPro 开始升级数据库。随后出现一条消息，指出数据库升级完毕。

- 6 单击**确定**。

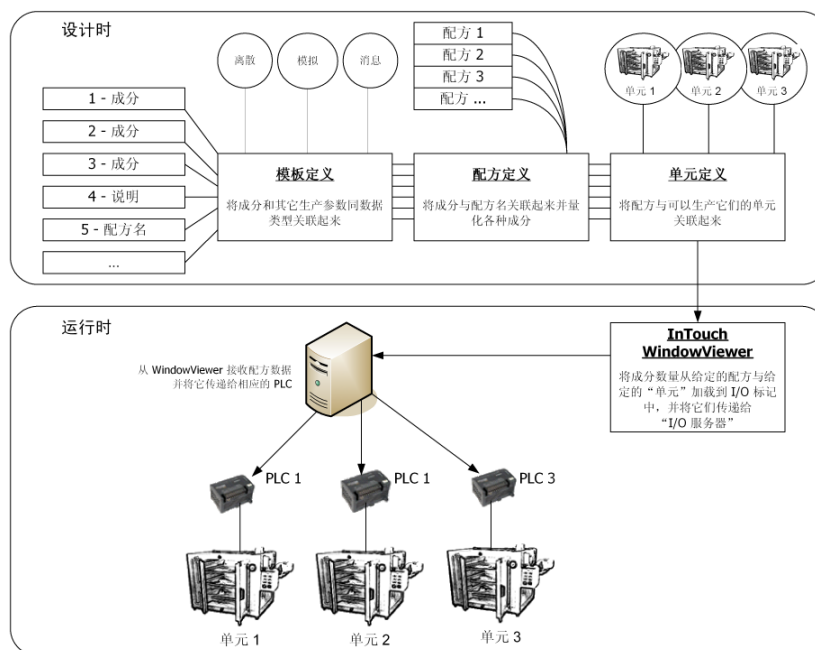


## 第 2 章

# 使用 Recipe Manager

制造业根据使用标准数量原材料的可重复过程来生产产品。从本质上说，产品是根据配方进行生产的。配方描述原材料、原材料数量以及它们如何组合起来，从而生产出最终的产品。最直观的情况是，面包房根据列出所有成分与程序化步骤的基本配方来制作饼干。

Recipe Manager 是 InTouch HMI 的辅助组件，可用于简化创建生产配方的过程。下图概要介绍 Recipe Manager 如何从配方模板中获取信息，以便管理产品生产的过程。



## Recipe Manager 综述

Recipe Manager 可以作为可选组件随 InTouch 一起安装。

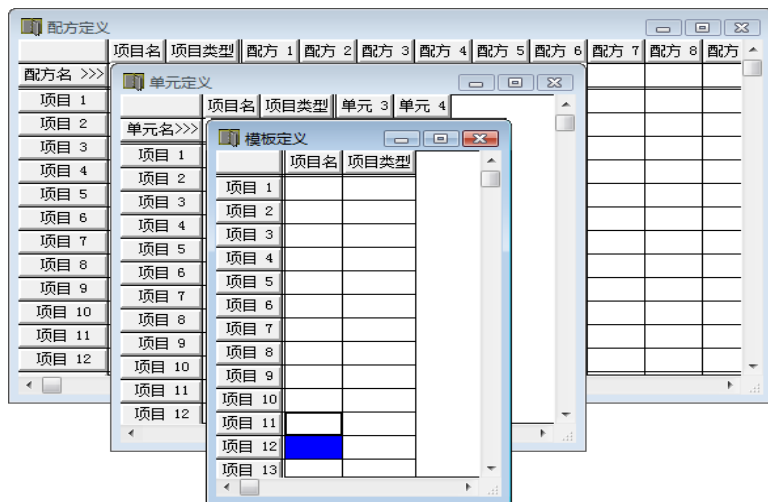
Recipe Manager 由 Recipe Manager 实用程序和一套 InTouch 配方脚本函数组成。

Recipe Manager 实用程序可以从 WindowMaker，或是从 Windows 的“开始”菜单访问。Recipe Manager 实用程序包含一个可用于创建与编辑配方模板的界面。Recipe Manager 将模板保存在配方文件中。

通常，与生产过程关联的标记使用 QuickScript 来访问配方模板文件中的数据。Recipe Manager 包含一套 QuickScript 函数，可用于选择、加载、修改、创建及删除模板文件中包含的生产配方。

## Recipe Manager 实用程序

Recipe Manager 实用程序提供一个类似于电子表格的用户界面，可用于创建与维护配方模板文件。一个文件由三个模板组成。通过在每个模板的电子表格的单元格中添加或修改数据，可以创建与编辑这些模板。



这些模板保存在“逗号分隔值”（Comma Separated Value，简称 CSV）文件中。使用任何支持 .csv 文件格式的程序（如“记事本”或 Excel），都可以创建与编辑配方模板定义。不过，Recipe Manager 提供了一些预先设置好格式的电子表格和一套编辑工具，可以轻松、可靠地创建与维护这些模板。

## 配方模板文件

Recipe Manager 模板文件包含以下信息：

- 配方中使用的成分的名称及其数据类型。
- 将 InTouch 标记与配方成分值关联起来的单元名。
- 配方名，它包含配方实例中使用的每种成分的数量或值。

### 模板定义

“模板定义”模板定义所有的配方成分。每个配方成分都有数据类型与之关联。成分数据类型可以是模拟、离散或消息。成分名不要求是 InTouch 标记。

### 单元定义

“单元定义”模板将 InTouch 标记与配方成分关联起来。您可以创建许多种不同的加载定义。这些定义称为单元。通过使用 RecipeLoad() 函数，可以将配方的特定实例加载到关联的 InTouch 标记。“单元定义”可以由文件中定义的所有成分，或者仅是其中的一部分成分组成。

---

**备注** 单元标记可以是能够在 InTouch 窗口中进行查看与编辑的内存型，或是能够直接加载到 PLC 的 I/O 标记。

---

### 配方定义

“配方定义”模板指定每个配方的名称及其使用的各种成分的数量。配方实例可以在运行时通过配方函数进行修改、创建或删除。

## 在 Recipe Manager 中编辑配方数据

通过完成一系列的任务，可以创建生产配方。下面的列表给出创建配方需要完成的 Recipe Manager 任务以及应该遵循的顺序：

- 配置 Recipe Manager 编辑网格。
- 编辑模板中的数据。
- 给“模板定义”模板指定成分名与单元类型。
- 在“单元定义”模板中，将 InTouch 标记映射到成分。
- 在“配方定义”模板中，将值指定给配方成分。

### 配置 Recipe Manager 编辑网格

在创建生产配方之前，应该配置 Recipe Manager。配置 Recipe Manager 编辑功能有两项任务：

- 设置模板项目数的最大限制。
- 设置 ENTER 键滚动功能。

在创建配方之前，需要配置可以输入到配方模板中的最大项目数。您必须为项目、单元以及配方名指定一组最大限制。

模板最多可以包含 9999 个项目、单元以及配方名。不过，很大的最大限制可能会影响系统性能。此外，如果设置的最大限制所需的内存超出计算机的可用内存，则可能会看到错误消息。

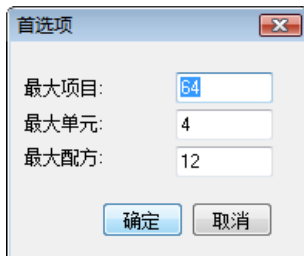
#### 要配置配方模板的最大限制

**1** 通过以下方法之一启动 Recipe Manager：

- 启动 WindowMaker。在工具视图中，展开**应用程序**，然后选择 **Recipe Manager**。
- 从 Windows 的桌面单击**开始**。依次单击**程序**、**Wonderware**，**InTouch**，最后单击 **Recipe Manager**。

此时出现 **Recipe Manager** 对话框。

**2** 在**选项**菜单上，单击**首选项**。此时出现**首选项**对话框。



- 3** 在**最大项目**框中，输入**模板定义**模板中允许的最大项目名数量。
- 4** 在**最大单元**框中，输入**单元定义**模板中允许的最大单元数量。

5 在**最大配方**框中，输入**配方定义**模板中允许的最大配方数量。

**注意** 在**首选项**对话框中设置的值应用于创建的所有配方模板文件。修改这些值时，也会修改现有的所有配方模板文件。


6 单击**确定**。  
Recipe Manager 包含一个可以简化在配方模板中输入数据的选项。选择**按回车键自动向下**选项时，按 ENTER 键可以将光标向下移动到列中的下一个单元格。

要设置 ENTER 键模板滚动功能

- 1 打开 Recipe Manager。  
缺省条件下，Recipe Manager 不自动滚动到模板电子表格中的下一个单元格。
- 2 在**选项**菜单上，单击**按回车键自动向下**以设置单元格滚动功能。
- 3 如果希望关闭单元格滚动功能，请再次单击**按回车键自动向下**。

使用编辑网格

Recipe Manager 包含一组编辑命令，可用于在模板中添加、修改或删除数据。通常，您在模板中选择要编辑的数据，然后执行编辑操作。下表介绍配方模板用于输入与选择模板数据的常用功能。

功能	描述
输入框	文本输入框用于为所选的模板单元格输入数据。选择模板单元格时，其内容显示在 Recipe Manager 对话框顶部附近的文本输入框中。
选择所有单元格	单击模板左上角的单元格可以选择所有的单元格。 <div></div>
选择整行	单击模板的行名可以选择该行中的所有单元格。
选择整列	单击模板的列标题可以选择该列中的所有单元格。
自动调整所有列的大小	双击模板可以按各列中最长输入项的宽度来自动调整模板中各个列的大小。

功能	描述
自动调整一列的大小	双击标题可以按照列中最长输入项的宽度来自动调整该列的大小。
	<b>备注</b> 模板定义模板中的项目类型列的大小无法自动调整。

编辑模板时，可以执行以下操作：

- 清除一定范围单元格中的数据。
- 将一定范围单元格中的数据复制到邻近所选范围的单元格。
- 在模板定义模板中插入一行。
- 在模板中插入一列。
- 从模板定义模板中删除一行。
- 从模板中删除一列。

要清除一定范围单元格的数据

- 1 从模板中选择一定范围的单元格。

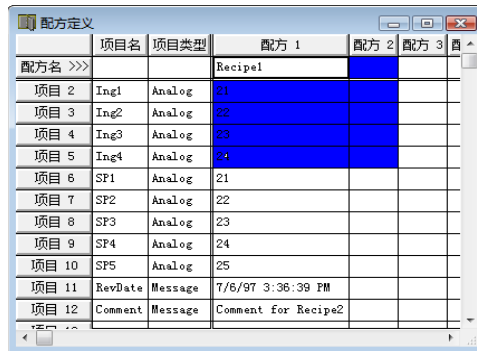


- 2 在编辑菜单上，单击清除。此时出现一条消息，要求确认是否应清除所选范围的单元格。
- 3 单击是。此时模板清除所选范围单元格的数据。



要将一定范围的单元格复制到邻近的所选范围

- 1 选择要复制的单元格或一定范围的单元格。
- 2 选择要将数据复制到的邻近单元格范围。



所选的范围必须大小相同，可以位于原始所选单元格范围的上方、下方、左侧或右侧。

- 3 在**编辑**菜单上，选择适当的填充命令。此时数据复制到所选范围的单元格。

**备注** 如果数据复制到其中的新列不足以容纳最长的输入项，请双击列标题以根据最长的输入项来调整其宽度。

要在“模板定义”模板中插入一行

- 1 选择模板定义模板。
- 2 在模板定义模板中，通过单击项目 # 来选择一行，以便在其上方插入新的行。

您无法直接在配方定义或单元定义模板中插入一些行。相反，配方定义与单元定义模板会自动继承对模板定义所作的全部修改。

- 3 在**编辑**菜单上，单击**插入**。此时新行正好插入所选行的上方，同时所有后续的行自动重新编号。



**备注** 如果已经达到给 Recipe Manager 首选项配置的最大值，则插入命令处于非活动状态。您必须增加指定的数值，才能给配方模板添加“项目 / 单元 / 配方”。

修改首选项时，所作的更改应用于现有的全部配方模板文件。

要插入一列

- 1 单击**单元 #** 或**配方 #** 以选择某个列，此列将位于要插入的列的右侧。  
您无法在**配方定义**或**单元定义**模板中插入一些列。
- 2 在**编辑**菜单上，单击**插入**。此时新列插入到所选列的左侧。

单元名>>>	项目名	项目类型	单元 1	单元 2	单元 3	单元 4
项目 1			Review		Mixer2	
项目 2	Ing1	Analog	Ing1		M2Ing1	
项目 3	Ing2	Analog	Ing2		M2Ing2	
项目 4	Ing3	Analog	Ing3		M2Ing3	
项目 5	Ing4	Analog	Ing4		M2Ing4	
项目 6	SF1	Analog	SF1		M2Ing4	
项目 7	SF2	Analog	SF2		M2SP2	
项目 8	SF3	Analog	SF3		M2SP3	
项目 9	SF4	Analog	SF4		M2SP4	
项目 10	SF5	Analog	SF5		M2SP5	
项目 11	RevDate	Message	Date			
项目 12	Comment	Message	Comment			

在本例中，**Mixer2** 数据移到**单元 3** 列中，新的列作为**单元 2** 插入。

要删除一列

- 1 单击**单元 #** 或**配方 #** 列标题，以便选择要删除的列。  
您无法从**配方定义**或**单元定义**模板中删除一些列。
- 2 在**编辑**菜单上，单击**删除**。此时出现一个确认消息对话框，要求确认是否删除。
- 3 单击**是**。此时该列从模板中删除，剩余的列重新编号。

要删除一行

- 1 选择**模板定义**模板。  
您可以从**模板定义**模板中删除一些行，但无法从**配方定义**或**单元定义**模板中进行删除。
- 2 单击**项目 #** 行标题，以便选择要删除的行。
- 3 在**编辑**菜单上，单击**删除**。此时出现一个确认消息对话框，要求确认是否删除。
- 4 单击**是**。此时该行从模板中删除。

## 定义成分名与数据类型

“模板定义”模板列出配方的各种成分以及与每种成分关联的单元类型。您必须先完成“模板定义”模板，然后才能将数据添加到其它配方模板。

### 要定义“模板定义”模板

- 1 启动 Recipe Manager。
- 2 在文件菜单上，单击**新建**。此时出现三个 Recipe Manager 模板。
- 3 单击**模板定义**标题栏以选择模板窗口。
- 4 在**项目名**列单元格中，输入为配方成分选择的名称。  
每个单元格只能输入一种成分。
- 5 在**项目类型**列单元格中，为相应的配方成分输入有效的项目类型。



有效的项目类型是：模拟、离散或消息。输入 A 代表 analog（模拟），D 代表 discrete（离散），M 代表 message（消息）。按 ENTER 键时，Recipe Manager 自动完成项目类型的剩余字符。

## 将 InTouch 标记映射到成分

单元定义模板将 InTouch 标记与给定单元的配方成分关联起来。如下图所示，单元定义模板的头两列会列出来自模板定义模板的“项目名”与“项目类型”。



单元名>>>	项目名	项目类型	单元 1	单元 2	单元 3	单元 4
项目 1	H2SO4	Analog				
项目 2	Benzene	Analog				
项目 3	DistillateB	Analog				
项目 4	Naphtelene	Analog				
项目 5						
项目 6						
项目 7						
项目 8						
项目 9						
项目 10						
项目 11						
项目 12						

给单元定义的标记可以是内存标记，或是从“I/O 服务器”获取 PLC 数据的远程标记。

在 InTouch QuickScript 中使用 RecipeLoad() 函数时，必须指定“单元名”。随后在运行 QuickScript 时，该“配方名”定义中包含的值会加载到在“单元名”中指定的标记。

### 要定义“单元定义”模板

- 1 单击单元定义模板的标题栏，以选择该模板窗口。



单元名>>>	项目名	项目类型	单元 1	单元 2	单元 3
项目 1	Ing1	Analog	Ing1	M1Ing1	M2Ing1
项目 2	Ing2	Analog	Ing2	M1Ing2	M2Ing2
项目 3	Ing3	Analog	Ing3	M1Ing3	M2Ing3
项目 4	Ing4	Analog	Ing4	M1Ing4	M2Ing4
项目 5	SP1	Analog	SP1	M1SP1	M2SP1
项目 6	SP2	Analog	SP2	M1SP2	M2SP2
项目 7	SP3	Analog	SP3	M1SP3	M2SP3
项目 8	SP4	Analog	SP4	M1SP4	M2SP4
项目 9	SP5	Analog	SP5	M1SP5	M2SP5
项目 10	RevDate	Message	Date		
项目 11	Comment	Message	Comment		

- 2 在单元名行中，输入要定义每个单元的名称。
- 3 在单元 # 列单元格中，使用以下方法之一，为每个相应的配方成分输入 InTouch 标记的名称。
  - 输入标记名。
  - 如果 WindowMaker 正在运行，请双击单元格以显示选择标记对话框。然后，双击所需的标记将它插入到单元格中，或是选择它并单击确定。
- 4 对于每个“单元 / 配方”组合，重复此操作程序。

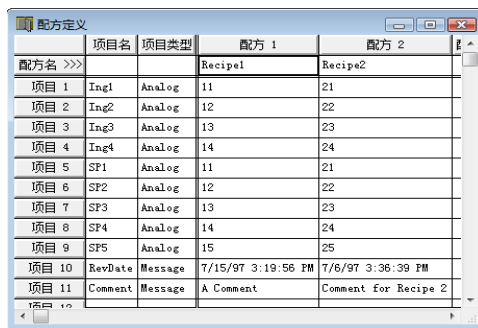
## 为不同配方中的成分定义值

“配方定义”模板指定每个配方的名称及其使用的各种成分的数量。“配方定义”模板显示先前定义的“模板定义”模板中的“项目名”与“项目类型”信息。

在 InTouch QuickScript 中执行 **RecipeLoad()** 函数时，成分值加载到 InTouch 标记中。

### 要定义“配方定义”模板

- 1 单击**配方定义**模板的标题栏，以选择该模板窗口。
- 2 在**配方名**行中，输入要定义的几个配方的名称。

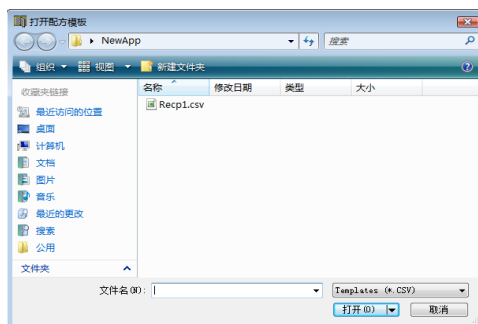


配方名 >>>	项目名	项目类型	配方 1	配方 2
项目 1	Ing1	Analog	11	21
项目 2	Ing2	Analog	12	22
项目 3	Ing3	Analog	13	23
项目 4	Ing4	Analog	14	24
项目 5	SP1	Analog	11	21
项目 6	SP2	Analog	12	22
项目 7	SP3	Analog	13	23
项目 8	SP4	Analog	14	24
项目 9	SP5	Analog	15	25
项目 10	RevDate	Message	7/15/97 3:19:56 PM	7/6/97 3:36:39 PM
项目 11	Comment	Message	A Comment	Comment for Recipe 2

- 3 在**配方 #**列单元格中，为**项目名列**中的每个相应配方成分输入值。
- 4 在**文件**菜单上，单击**保存**以保存配方模板文件。

### 要打开现有的配方模板文件

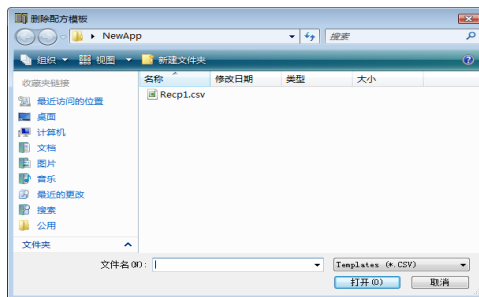
- 1 打开 Recipe Manager。
- 2 在**文件**菜单上，单击**打开**。此时出现**打开配方模板**对话框。



- 3 找到并选择“配方”文件，然后单击**打开**。文件中的三个配方模板出现在 Recipe Manager 中，此时可以进行编辑。

### 要删除配方模板文件

- 1 在文件菜单上，单击**删除**。此时出现**删除配方模板**对话框。



- 2 找到并选择配方文件，然后单击**打开**，或双击文件名。此时出现一个消息框，要求确认是否删除。

---

**备注** 打开的配方模板文件无法删除。

---

- 3 单击**是**以删除该文件。

## 在其它应用程序中编辑配方数据

使用任何支持逗号分隔数据的程序，都可以创建与编辑配方模板定义。您可以使用 Microsoft Excel 或“记事本”来创建与编辑 Recipe Manager 模板文件。

### 使用 Excel 处理配方模板文件

如果不希望使用 Recipe Manager 实用程序，则可以使用 Excel 来创建或编辑配方模板。使用 Excel 创建或编辑的 Recipe Manager 模板必须保存到文件扩展名为 .csv 的文件。

要在 Microsoft Excel 中打开现有的配方模板文件

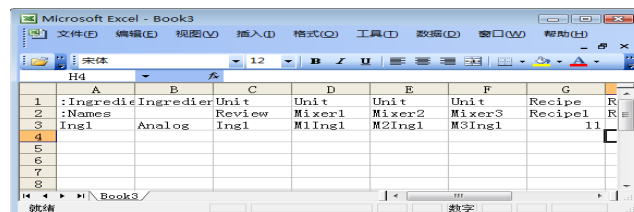
- 1 启动 Excel。
- 2 在文件菜单上，单击**打开**。此时出现**打开**对话框。
- 3 找到并选择 .csv 文件，然后单击**打开**，或双击文件名。此时 Excel 显示该文件的内容。

Microsoft Excel - Recp1.csv									
文件(F)	编辑(E)	视图(V)	插入(I)	格式(O)	工具(T)	数据(W)	窗口(W)	帮助(H)	...
地址: 本地				: 12		B 1		=	
A1				:IngredientName					
1	:Ingredient	Ingredient	Unit	Unit	Recipe	Recipe			
2	:Names	Review	Mixer1	Mixer2	Recipe1	Recipe1			
3									
4	Ing1	Analog	Ing1	M1Ing1	M2Ing1	21	21		
5	Ing2	Analog	Ing2	M1Ing2	M2Ing2	22	22		
6	Ing3	Analog	Ing3	M1Ing3	M2Ing3	23	23		
7	Ing4	Analog	Ing4	M1Ing4	M2Ing4	24	24		
8	SP1	Analog	SP1	M1SP1	M2Ing4	21			
9	SP2	Analog	SP2	M1SP2	M2SP2	22			
10	SP3	Analog	SP3	M1SP3	M2SP3	23			
11	SP4	Analog	SP4	M1SP4	M2SP4	24			
12	SP5	Analog	SP5	M1SP5	M2SP5	25			
13	RevDate	Message	Date			7/6/97 3:36:39 PM			
14	Comment	Message	Comment			Comment for Recipe2			
H:\Recp1.csv									
地址				数字					

- 4 编辑配方文件的内容并保存更改。

要在 Excel 中创建新的配方模板文件

- 1 启动 Excel。
- 2 创建新的工作簿。
- 3 在电子表格中输入配方数据，如下图所示。



H4	A	B	C	D	E	F	G
1	:Ingredient	Ingredient	Unit	Unit	Unit	Recipe	R
2	:Names	Review	Mixer1	Mixer2	Mixer3	Recipe1	R
3	Ing1	Analog	Ing1	M1Ing1	M2Ing1	M3Ing1	11
4							
5							
6							
7							
8							

各个项目必须按照图中所示的顺序进行输入。“单元名”必须在包含“配方名”的列左侧的列中定义。

- 4 使用 .csv 文件扩展名保存电子表格。

## 使用“记事本”处理配方模板文件

如果不希望使用 Recipe Manager 实用程序，则可以使用“记事本”来创建或编辑配方模板。使用“记事本”创建或编辑的 Recipe Manager 模板必须保存到文件扩展名为 .csv 的文件。

### 要在“记事本”中打开现有的配方模板文件

- 1 启动“记事本”。
- 2 在文件菜单上，单击**打开**。此时出现**打开**对话框。
- 3 找到并选择配方文件，然后单击**打开**，或双击文件名。
- 4 编辑配方文件的内容并保存更改。

### 要在“记事本”中创建新的配方模板文件

- 1 启动“记事本”。
- 2 在文件菜单上，单击**新建**。
- 3 按下面这种格式输入以下数据：  

```
:IngredientName,IngredientType[,Unit]...[,Recipe]..  
.  
:Names,,[,UnitName]...[,RecipeName]...  
IngredientName,{Analog,Discrete,Message},[,tag]...[,value]
```

---

**备注** 所有的“单元名”必须先于“配方名”在文件中进行定义。

---

- 4 使用 .csv 文件扩展名保存文件。



## 通过嵌套配方来创建复杂的结构

使用 InTouch QuickScript 可以将多个配方模板文件互相链接起来，从而创建复杂的应用。通过定义一个与“单元名”模板中的消息标记关联的成分名，便可以链接配方模板文件。然后，您使用配方的名称来加载该消息标记。

通过链接配方模板文件，可以创建主配方模板文件，这些主文件定义不同配方文件中的各种配方所使用的机器配置参数。将这类信息保存在一个中心文件中，可以大幅减少数据发生更改时进行维护与更新所需的时间。

在下图中，“项目名” **Setup** 标记定义为消息型，并且单元中包含此项目的 **Setup** 消息标记。每个配方都包含在另一个配方文件（选择配方时，此文件加载到 **Setup** 标记）中定义的第二个配方名。

	A	B	C	D	E	F	G	H	I
1	:IngredientName	IngredientType	Unit	Unit	Unit	Unit	Recipe 1	Recipe 2	Recipe 3
2	:Names		Review	Mixer 1	Mixer 2	Mixer 3	Recipe 1	Recipe 2	Recipe 3
3	Ing1	Analog	Ing1	M1Ing1	M2Ing1	M3Ing1	11	21	31
4	Ing2	Analog	Ing2	M1Ing2	M2Ing2	M3Ing2	12	22	32
5	Ing3	Analog	Ing3	M1Ing3	M2Ing3	M3Ing3	13	23	33
6	Ing4	Analog	Ing4	M1Ing4	M2Ing4	M3Ing4	14	24	34
7	SP1	Analog	SP1	M1SP1	M2SP1	M3SP1	11	21	31
8	SP2	Analog	SP2	M1SP2	M2SP2	M3SP2	12	22	32
9	SP3	Analog	SP3	M1SP3	M2SP3	M3SP3	13	23	33
10	SP4	Analog	SP4	M1SP4	M2SP4	M3SP4	14	24	34
11	SP5	Analog	SP5	M1SP5	M2SP5	M3SP5	15	25	35
12	Setup	Message	Setup	LinkFile	LinkFile	LinkFile	Setup2A	Setup3A	Setup1A

为此，将输入以下脚本：

```
RecipeName="Recipe2";
```

```
RecipeLoad("c:\recipe\recfilea.csv", "Review", RecipeName);
```

此脚本运行时，**Setup** 标记的值变为 **Setup3A**，并加载到 **Review** 单元。随后，通过运行以下脚本来执行下一次配方加载操作，以便将机器设置参数加载到为 **PLC1** 单元定义的标记，并将 **Setup** 标记的值用作“配方名”。

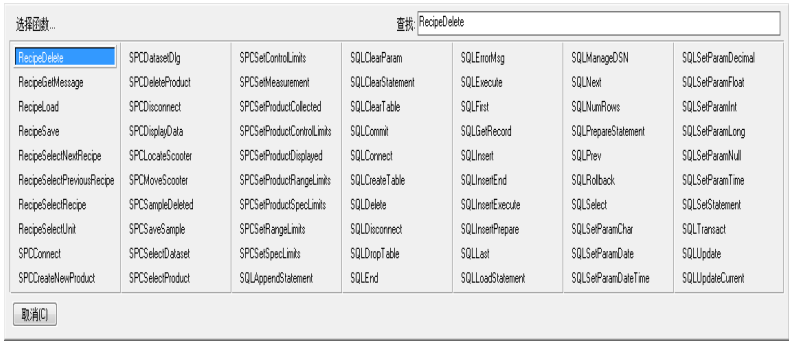
```
RecipeLoad("c:\recipe\machine.csv", "PLC1", Setup);
```

	A	B	C	D	E	F	G	H
1	:ItemName	ItemType	Unit	Recipe	Recipe	Recipe		
2	:Names		PLC1	Setup1A	Setup2A	Setup3A		
3	PARM1	Analog	PARM1	11	21	31		
4	PARM2	Analog	PARM2	12	22	32		
5	PARM3	Analog	PARM3	13	23	33		
6	PARM4	Analog	PARM4	14	24	34		
7	PARM5	Analog	PARM5	11	21	31		
8	PARM6	Analog	PARM6	12	22	32		
9	PARM7	Analog	PARM7	13	23	33		
10	PARM8	Analog	PARM8	14	24	34		
11	PARM9	Analog	PARM9	15	25	35		

## 在 InTouch 中使用配方

您可以使用 InTouch QuickScript 与配方模板文件进行交互。Recipe Manager 包含一套可插入 QuickScript 中的脚本函数。通过使用包含这些函数的脚本，可以在配方模板文件中选择、修改、插入或删除记录。

使用 InTouch 脚本编辑器可以将配方函数添加到任何类型的脚本。下图显示列出了配方函数的 InTouch 脚本编辑器窗口。所有的配方函数都通过将前缀 Recipe 用作函数名的一部分来进行标识。



InTouch 配方函数直接读取和写入配方模板文件。因此，Recipe Manager 程序不需要处在运行状态，配方函数便可以在 InTouch QuickScript 中正常运行。

如果配方模板文件正由 InTouch HMI 使用，则创建的任何新配方，或是对现有配方所作的任何更改，都无法写入配方模板文件。Recipe Manager 仅用于创建配方模板文件。创建“配方模板”文件之后，关闭 Recipe Manager。

### 要将配方函数自动插入到脚本中

- 1 打开 InTouch 脚本编辑器。
- 2 将光标放到脚本中要插入配方函数的位置。
- 3 在函数下，单击附件。此时出现选择函数对话框。
- 4 单击要插入到 QuickScript 中的配方函数。此时对话框关闭，该函数插入到光标位置。

## 在配方文件中加载与保存配方数据

Recipe Manager 提供单独的 InTouch QuickScript 函数在配方文件中加载与保存配方数据。

### RecipeLoad() 函数

RecipeLoad() 函数将配方数据加载到 InTouch 应用程序中特定标记的单元。

#### 类别

配方

#### 语法

```
RecipeLoad("Filename", "UnitName", "RecipeName");
```

#### 参数

##### *Filename*

配方模板文件的名称。与 *Filename* 关联的值可以是字符串常数，也可以是包含配方模板文件名的消息标记。

##### *UnitName*

指定的配方模板文件中特定单元的名称。RecipeSelectUnit() 函数将值返回给此参数。与 *Filename* 关联的值可以是字符串常数，也可以是包含该单元名的消息标记。

##### *RecipeName*

指定的配方模板文件中特定配方的名称。与 *Filename* 关联的值可以是字符串常数，也可以是包含配方名的消息标记。

#### 示例

下面的语句将 Recipe1 配方的值加载到为 Unit1 定义的标记中：

```
RecipeLoad("c:\recipe\recfile.csv", "Unit1",  
"Recipe1");
```

## RecipeSave() 函数

RecipeSave() 函数将新配方或是对现有配方所作的更改保存到指定的配方模板文件。

### 类别

配方

### 语法

```
RecipeSave ("Filename", "UnitName", "RecipeName");
```

### 参数

#### *FileName*

配方模板文件的名称。与 *FileName* 关联的值可以是字符串常数，也可以是包含配方模板文件名的消息标记。

#### *UnitName*

指定的配方模板文件中此函数将要使用的特定单元的名称。

RecipeSelectUnit() 函数将值返回给此参数。与 *FileName* 关联的值可以是字符串常数，也可以是包含该单元名的消息标记。

#### *RecipeName*

指定的配方模板文件中特定配方的名称。与 *FileName* 关联的值可以是字符串常数，也可以是包含配方名的消息标记。

### 示例

下例将对 Recipe3 配方所作的更改保存到 recfile.csv 文件。如果 recfile.csv 文件中当前不存在 Recipe3，则会创建它。这些值用于设置为 Unit2 定义的标记的值：

```
RecipeSave ("c:\recipe\recfile.csv", "Unit2",  
            "Recipe3");
```

## 从配方文件中删除配方

使用 `RecipeDelete` 函数，可以从指定的配方模板文件中删除配方。

### `RecipeDelete()` 函数

`RecipeDelete` 函数从指定的配方模板文件中删除配方。

类别

配方

语法

```
RecipeDelete ("Filename", "RecipeName");
```

参数

*FileName*

配方模板文件的名称。与 *FileName* 关联的值可以是字符串常数，也可以是包含配方模板文件名的消息标记。

*RecipeName*

指定的配方模板文件中特定配方的名称。与 *FileName* 关联的值可以是字符串常数，也可以是包含配方名的消息标记。

示例

下面的语句从 `recfile.csv` 文件中删除 `Distlt1` 配方：

```
RecipeDelete ("c:\recipe\recfile.csv", "Distlt1");
```

## 选择单元（标记成分映射）

使用 `RecipeSelectUnit()` 函数，可以选择要加载当前配方值的标记所对应的单元。

### `RecipeSelectUnit()` 函数

`RecipeSelectUnit()` 函数打开**选择单元**对话框，供运行时用户选择单元。所选的单元名返回到消息标记。`RecipeSelectRecipe()` 与 `RecipeSelectUnit()` 函数都与 `RecipeLoad()` 函数配合使用。

类别

配方

语法

```
RecipeSelectUnit ("Filename", UnitName, Number);
```

参数

*FileName*

配方模板文件的名称。*FileName* 参数可以是字符串常量，也可以是包含配方模板文件名的消息标记。

*UnitName*

写入所选单元名称的消息标记。不带英文引号的实际消息标记，或是字符串。

### *Number*

返回给参数的最大字符串长度。在 InTouch 中，字符串（消息）标记的最大长度是 131 个字符。除非已缩短 InTouch 标记的最大字符串长度，否则此参数使用 131。数字或整型标记。

### 示例

下面的语句打开**选择单元**对话框：

```
RecipeSelectUnit("c:\recipe\recfile.csv",  
    UnitName, 131);
```

选择“单元”之后，其名称返回给 UnitName 标记。

## 从配方文件中选择单独的配方

Recipe Manager 包括一套可以从配方文件中选择单独配方的函数。在脚本中使用这些函数时，可以根据名称从文件中选择配方，也可以根据文件中的顺序选择上一个或下一个配方。

### RecipeSelectRecipe() 函数

RecipeSelectRecipe() 函数打开**选择配方**对话框，供运行时用户选择配方。所选的配方名返回到消息标记。

### 类别

配方

### 语法

```
RecipeSelectRecipe("Filename", RecipeName,  
    Number);
```

### 参数

#### *FileName*

配方模板文件的名称。*FileName* 参数可以是字符串常量，也可以是包含配方模板文件名的消息标记。

#### *RecipeName*

写入所选配方名称的消息标记。不带英文引号的实际消息标记，或是字符串。

#### *Number*

返回给参数的最大字符串长度。InTouch 消息标记的最大长度是 131 个字符。除非已经缩短 InTouch 标记的最大字符串长度，否则此参数使用 131。数字或整型标记。

### 示例

下面的语句会打开**选择配方**对话框：

```
RecipeSelectRecipe("c:\recipe\recfile.csv",  
    RecipeName, 131);
```

从对话框中选择配方之后，其名称返回给 RecipeName 标记。

## RecipeSelectNextRecipe() 函数

RecipeSelectNextRecipe() 函数选择配方模板文件中的下一个配方。

### 类别

配方

### 语法

```
RecipeSelectNextRecipe (" Filename", RecipeName,  
                        Number) ;
```

### 参数

#### *FileName*

配方模板文件的名称。 *FileName* 参数可以是字符串常量，也可以是包含配方模板文件名的消息标记。

#### *RecipeName*

消息标记，包含用作起点的配方名（执行函数之前）与所选的配方名（执行函数之后）。不带英文引号的实际消息标记，或是字符串。

#### *Number*

返回给参数的最大字符串长度。在 InTouch 中，字符串（消息）标记的最大长度是 131 个字符。除非已经缩短 InTouch 标记的最大字符串长度，否则此参数使用 131。数字或整型标记。

### 示例

下面的语句读取 *RecipeName* 标记的当前值并返回文件中的下一个配方。如果 *RecipeName* 的值为空或是无法找到，则返回文件中的第一个配方。如果 *RecipeName* 当前包含文件中的最后一个“配方名”，则原封不动地返回该配方名。配方按照它们的创建顺序保存在配方模板文件中。

```
RecipeSelectNextRecipe ("c:\recipe\recfile.csv",  
                        RecipeName, 131);
```

### RecipeSelectPreviousRecipe() 函数

RecipeSelectPreviousRecipe() 函数选择配方模板文件中定义的上一个配方。

类别

配方

语法

```
RecipeSelectPreviousRecipe("Filename", RecipeName, Number);
```

参数

*FileName*

配方模板文件的名称。*FileName* 参数可以是字符串常量，也可以是包含配方模板文件名的消息标记。

*RecipeName*

消息标记，包含用作起点的配方名（执行函数之前）与所选的配方名（执行函数之后）。不带英文引号的实际消息标记，或是字符串。

*Number*

返回给此参数的最大字符串长度。在 InTouch 中，“消息”标记的最大长度是 131 个字符。除非已经缩短 InTouch 标记的最大字符串长度，否则此参数使用 131。数字或整型标记。

示例

下面的语句会导致系统读取 *RecipeName* 标记的当前值，并返回文件中的上一个“配方名”。返回的这个字符串将存储在 *RecipeName* 中，并且将覆盖掉当前值。如果 *RecipeName* 的值为空或是无法找到，则返回文件中的最后一个配方。如果 *RecipeName* 当前包含文件中的第一个“配方名”，则原封不动地返回该配方名。（配方按照它们的创建顺序进行保存）。

```
RecipeSelectPreviousRecipe("c:\recipe\recfile.csv", RecipeName, 131);
```



## 理解配方脚本函数返回的错误消息

通过使用配方函数返回的诊断错误码，可以排解配方应用程序的错误。本节包含配方函数错误码列表，并介绍如何使用 `RecipeGetMessage()` 函数来显示与错误码关联的消息。

如果 `RecipeLoad()` 函数成功执行，它会将 `ErrorCode` 模拟标记的值设置为 0。如果 `RecipeLoad()` 失败，它会将 `ErrorCode` 标记设置为特定错误条件的编号。

要检索配方函数的错误码，必须将它赋给某个 InTouch 模拟标记。下例显示一个可以返回配方函数错误码的脚本语句：

```
ErrorCode = RecipeLoad(FileName, UnitName, RecipeName);
```

### 显示错误码消息

每个配方函数都会返回一个代表该函数的错误条件的编号。通过在 InTouch 的“数据改变”脚本中使用 `RecipeGetMessage()` 函数，可以将相应的错误码写入一个模拟标记，并将关联的错误码消息写入一个消息标记。

下面的代码示例显示一个“数据改变”脚本。

```
RecipeGetMessage(ErrorCode, ErrorMessage, 131);
```

只要 `ErrorCode` 标记的值发生改变，此脚本便会自动运行。此脚本运行时，`RecipeGetMessage()` 函数读取 `ErrorCode` 标记的当前数值，并将与该值关联的消息返回给 `ErrorMessage` 标记。

下表列出可能的错误码及其对应的错误消息与描述：

值	错误消息	描述
0	成功	调用的配方函数成功执行。
-1	无此配方模板	指定的配方模板文件不存在。
-2	View 没有激活	由于 WindowViewer 不在运行，另一程序所调用的配方函数无法执行。
-3	内存不足	内存不足，无法完成当前的活动。
-4	在配方模板文件中的行太长	配方模板文件中的某一行超出最大允许长度。
-5	配方模板文件中的行被截断	配方模板文件中的某一行被截断。
-6	不是一个有效的配方模板文件	指定的文件不是一个有效的配方模板文件。
-7	期望“单元”或“配方”	配方模板文件中遗失单元名或配方名。

值	错误消息	描述
-8	配方模板文件中未定义单元	配方文件的“单元定义”模板中尚未定义任何单元。
-9	配方模板文件中未找到配方名	指定的配方在配方模板文件中未定义。
-10	配方模板文件中未找到单元名	指定的单元名在单元定义模板文件中未定义。
-12	期望是“模拟”、“离散”或“消息”	给配方模板文件中的某个项目输入的类型不正确。有效的类型是 <b>Analog</b> （模拟）、 <b>Discrete</b> （离散）或 <b>Message</b> （消息）。
-13	标记类型与“模拟”、“离散”或“消息”不匹配	指定的标记与项目类型不匹配。例如，将某个配方项目定义为模拟类型，同时又已将某个消息标记定义为它的单元。
-14	无效的离散值，期望是“0”或“1”	给配方模板文件中的某个离散标记输入了不正确的值。离散标记的有效值仅限 0 或 1。
-15	无法打开临时文件	临时文件无法打开，可能是由于空闲磁盘空间不足。
-16	保存配方模板文件时发生写入错误	保存配方模板文件期间发生错误。
-17	没有选定用户	用户在 <b>选择配方</b> 对话框中没有选择配方名，而是选择了 <b>取消</b> 。
-19	其它应用程序使用中的配方模板	指定的配方模板文件已经打开，因此 <b>WindowViewer</b> 无法访问它。

## RecipeGetMessage() 函数

RecipeGetMessage() 函数接受错误号（由其它一些配方函数返回），并返回该错误号的纯文本错误消息。

### 类别

配方

### 语法

```
RecipeGetMessage (Analog_Tag, Message_Tag, Number) ;
```

### 参数

#### Analog\_Tag

要获取其错误消息的错误号。

#### Message\_Tag

不带英文引号的实际消息标记，或是字符串。

#### Number

*Number* 参数设置由 *Message\_Tag* 参数返回的字符串的最大长度。缺省条件下，InTouch 消息标记设置为最大长度，即 131 个字符。除非已经缩短 “InTouch 标记名字典” 中 *Message\_Tag* 标记的最大字符串长度，否则此参数使用 131。*Number* 参数可以是常数，也可以是包含数字的整型标记。

### 示例

通过在 InTouch 的 “数据改变” QuickScript 中使用 RecipeGetMessage() 函数，可以将错误码写入 ErrorCode 标记，将关联的错误码消息写入 ErrorMessage 标记：

```
Data Change Script Tagname[.field]:ErrorCode
Script body:RecipeGetMessage(ErrorCode,
    ErrorMessage,131);
```

只要 ErrorCode 标记的值发生改变，此 QuickScript 便自动运行。此 QuickScript 运行时，RecipeGetMessage() 函数读取 ErrorCode 标记的当前数值，并将与该值关联的消息返回给 ErrorMessageTag。

```
ErrorCode = RecipeLoad
    ("c:\App\recipe.csv","Unit1","cookies");
RecipeGetMessage(ErrorCode, ErrorMessageTag,
    131);
```

## 设置配方安全性

通过在配方模板文件中定义一个“项目名”，由它设置加载、保存或删除配方时所需的最低安全访问级别，可以控制对配方的访问权限。

在下面的文件示例中，SecurityLevel 项目名定义为模拟标记。Review 单元包含此项目的 SecurityLevel 模拟标记。每个配方定义一个值，在该配方加载到 Review 单元时，这个值加载到 SecurityLevel 标记中。

MACHINE.CSV							
1	Item Name	Item Type	Unit	Unit	Recipe	Recipe	Recipe
2	Names		Review	PLC1	Setup1A	Setup2A	Setup3A
3	PARM1	Analog		PARM1	11	21	31
4	PARM2	Analog		PARM2	12	22	99
5	PARM3	Analog		PARM3	13	23	66
6	PARM4	Analog		PARM4	14	24	34
7	PARM5	Analog		PARM5	11	21	31
8	PARM6	Analog		PARM6	12	22	32
9	PARM7	Analog		PARM7	13	23	33
10	PARM8	Analog		PARM8	14	24	34
11	PARM9	Analog		PARM9	15	25	35
12	SecurityLevel	Analog	SecurityLevel		2000	5000	7000

您可以创建一个包含“访问被拒绝”消息的窗口；只要用户的安全访问级别对所选的配方而言无效，便显示该消息。为此，所选的配方必须加载到仅包含一个模拟标记的单元，而所选配方的安全级别值将加载到此模拟标记中进行验证。

例如：

```
RecipeSelectRecipe("c:\recipe\machine.csv",  
    MyRecipe, "131");
```

此时出现**选择配方**对话框。选择“配方名”之后，它返回给 RecipeName 标记，同时脚本继续运行。

```
RecipeLoad( "c:\Recipe\Machine.csv", "Review", MyRecipe );  
IF SecurityLevel <= $AccessLevel THEN  
    Status =RecipeLoad( "c:\Recipe\Machine.csv", "PLC1",  
        MyRecipe );  
    ELSE Show "Access Denied";  
ENDIF;
```

此脚本运行时，如果访问级别大于或等于 7000，则所选配方的值加载到 PLC1 单元的标记中。否则出现**访问被拒绝**窗口，并且配方不会加载到 PLC1 中。

## 第 3 章

# 从 InTouch 中使用 SQL 数据库

数据库在共享共同的属性或字段的表中存储信息。“结构化查询语言”（Structured Query Language，简称 SQL）是用来以查询的形式访问该类信息的语言。“SQL 访问管理器”让您可以使用查询来访问、修改、创建以及删除数据库表。

“SQL 访问管理器”是可以随 InTouch 一起安装的可选程序。

“SQL 访问管理器”可用于：

- 创建与运行复杂查询。这些查询可以动态构建，也可以保存在外部文件中。此外，这些查询可以包含要在运行时传递给查询的参数。
- 运行数据库支持的 SQL 语句并从查询中检索结果。您还可以给“SQL 访问管理器”使用存储过程，尽管并非完全支持所有的存储过程。

通过在 QuickScript 编辑器对话框中单击**附件**按钮，可以自动将 SQL 函数插入 InTouch QuickScript。SQL 函数自动插入到脚本中的当前光标位置。

您可以使用“SQL 访问管理器”将数据（如批次配方）从 SQL 数据库传输到 InTouch 应用程序。“SQL 访问管理器”还可以用来将运行时数据、报警状态或历史数据从 InTouch 传输到数据库。例如，在完成一个机器周期之后，公司希望保存几组数据，每组针对一个不同的应用程序。SQL 数据库提供了在一个或多个第三方应用程序之间轻松传输信息的功能。“SQL 访问管理器”使这些数据可以在任何 InTouch 应用程序中进行访问与显示。

“SQL 访问管理器”由一个应用程序和一套 SQL 函数组成。

“SQL 访问管理器”程序创建数据库列，并将它们与 InTouch 标记关联起来。将数据库列与 InTouch 数据库标记关联起来的过程称为绑定。通过将 InTouch 数据库标记绑定到数据库列，

“SQL 访问管理器”可以直接操纵数据库中存储的 InTouch 数据。

“SQL 访问管理器”在逗号分隔变量文件 SQL.DEF 中保存数据库字段名及其关联的对象。此文件位于 InTouch 应用程序文件夹，可以使用“SQL 访问管理器”或任何文本编辑器（如“记事本”）进行查看或修改。“SQL 访问管理器”还可以创建“表模板”来定义 InTouch 使用的数据库的结构与格式。

SQL 函数可以使用在脚本中，根据操作员的输入或标记值的变化情况，或是在存在一组特定的条件时自动运行。您可以使用这些函数在选择访问的表中选择、修改、插入或删除记录。例如，如果存在某个报警条件，则应用程序可以运行包含 SQLInsert() 或 SQLUpdate() 函数的脚本，以便保存所有适当的数据点以及报警的状态。

## 设置数据源

“SQL 访问管理器”是一个与 ODBC 兼容的应用程序，可以同支持可用 ODBC 驱动程序或 OLE DB 供应器的任何数据库进行通讯。

配置数据库连接字符串有多种方法：

- 使用 “Microsoft ODBC 管理器” 程序来配置要给 “SQL 访问管理器” 使用的 ODBC 驱动程序。
- 运行 SQLConnect() 函数并使用参数值来指定 OLE DB 供应器。如需有关详细信息，请参阅第 110 页的 “SQL Server 数据库应用程序”。
- 使用外部 UDL 文件来设置数据库连接字符串。

### 要配置 ODBC 驱动程序

- 1 运行 “Microsoft ODBC 管理器” 程序。
- 2 选择驱动程序或数据源，然后单击**添加新名称、设置缺省值或配置**。此时出现 **ODBC 驱动程序设置**对话框。

选项	描述
数据源名	用于确定数据源的用户自定义名称。
描述	数据源的用户自定义描述。
数据库目录	确定包含数据库文件的文件夹。如果未指定，则使用当前工作文件夹。

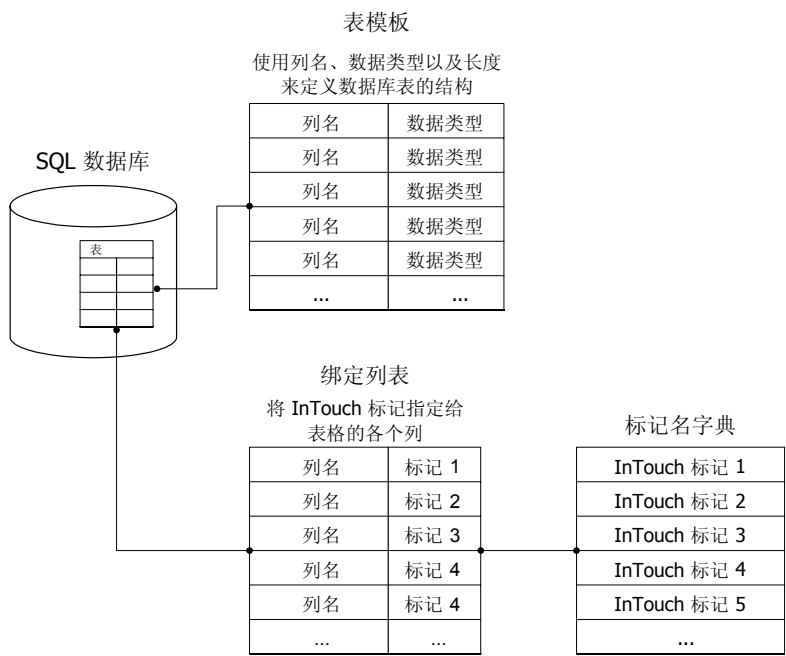
- 3 单击**确定**。

**备注** 创建 “ODBC 数据源” 时，会在 Windows 目录中创建一个 ODBC.INI 文件。您可以手工编辑 ODBC.INI 文件。

## 将 InTouch 标记映射到数据库列

您可以将 “InTouch 标记” 映射到数据库列。这是通过 “绑定列表” 来完成的。大多数 SQL Access 函数都通过 “绑定列表” 来使得 InTouch 标记可以访问 SQL 数据库表中的数据。

“绑定列表” 将数据库表列与 InTouch 的 “标记名字典” 中的标记关联起来。“绑定列表” 还包含一个描述数据库表格式的 “表模板”。



运行包含 SQLInsert()、SQLSelect() 或 SQLUpdate() 函数的脚本时，“绑定列表” 更新为指定使用了哪些 InTouch 标记，以及同这些标记关联的是哪些表格列。

### 要创建新的 “绑定列表”

- 1 在特别菜单上，指向 SQL 访问管理器，然后单击绑定列表。此时出现一条消息，要求确认是否创建 SQL.DEF 文件。
- 2 单击是以创建 SQL.DEF 文件。此时出现选择绑定列表对话框。



3 单击**新建**。此时出现**绑定列表配置**对话框。



4 在**绑定列表名**框中，输入“绑定列表名”。

“绑定列表名”的最大长度是 32 个字符。

5 要为该“绑定列表”定义标记，执行以下操作之一：

- 在**标记名.域名**框中，输入 InTouch 标记名。您也可以按 **tag\_name.dotfield\_name** 的形式添加可选的标记点域。
- 双击**标记名**以选择一个现有的标记。此时出现**选择标记**对话框。从列表选择一个标记。

**备注** WindowViewer 一启动，应用程序中未使用但在“SQL 访问绑定列表”中指定的 I/O 型标记便会立即激活（提示 DAServer）。

6 通过执行以下操作之一选择要附加到标记的点域：

- 在**标记名.域名**框中，在标记名后面输入一个英文句点和点域名。
- 单击**域名**。此时出现**选择域名**对话框。单击要附加到标记的点域。

7 在**列名**框中，输入列的名称。

列名的最大长度是 30 个字符。如果列名包含空格，则在“绑定列表”与脚本中使用时，请使用方括号将列名括起来。例如：

```
WHERE EXPR= "[Valve ID] = " + text (tagname,"#");
```

8 通过执行以下操作之一调整标记在“绑定列表”中的位置：

- 单击**向上移动**将所选的标记在列表中上移一级。
- 单击**向下移动**将所选的标记在列表中下移一级。

9 单击**添加项**将新的**标记名.域名**与**列名**添加到“绑定列表”。

10 单击**确定**以保存新的“绑定列表”配置并关闭对话框。

## 配置“绑定列表”中的 SQL Server 字符串分隔符

SQLInsert() 与 SQLUpdate() 函数使用缺省格式，即使用英文单引号将消息字符串括起来。有些 SQL 数据库期望收到由其它类型的分隔符括起来的消息字符串。例如，Oracle 8.0 期望收到由方括号括起来的日期字符串。发生这种情况时，必须使用 Delim() 函数。

在绑定列表配置对话框的列名字段中，在列名后面使用 delim() 函数。输入关键字 "delim" 时，后面必须跟着：

- 英文左圆括号
- 左分隔符
- 系统区域设置中定义的列表分隔符
- 右分隔符
- 英文右圆括号

英文系统的示例：datestring delim ('')

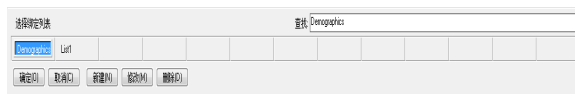
德文系统的示例：datestring delim (';')

要将相同的字符用作左、右分隔符，请在圆括号中指定分隔符（不要带分隔符），如下例所示：

datestring delim ('')

### 要修改“绑定列表”

- 1 在特别菜单上，指向 SQL 访问管理器，然后单击绑定列表。此时出现选择绑定列表对话框。



- 2 选择要更改的“绑定列表”名称，然后单击修改。此时出现绑定列表配置对话框。
- 3 修改所需的项目。
- 4 单击确定以保存更改并关闭对话框。

### 要使用 Excel 修改“绑定列表”

“SQL 访问管理器”将“绑定列表”与“表模板”的配置信息保存到 SQL.DEF 文件。此文件是采用“逗号分隔值”（Comma Separated Value，简称 CSV）格式的文件。

SQL.DEF 文件可以使用任何支持“逗号分隔值”文件的产品（如 Excel）来修改。

数据出现在该文件中的形式如下：

```
:BindListName, BindListName
```

*Tagname1.FieldName,ColumnName1*  
*Tagname2.FieldName,ColumnName2*  
*Tagname3.FieldName,ColumnName3*  
*:TableTemplateName, TableTemplateName*  
*ColumnName1,ColumnType,[ColumnLength],Null,Index*  
*ColumnName2,ColumnType,[ColumnLength],Null,Index*  
*ColumnName3,ColumnType,[ColumnLength],Null,Index*

### 要删除“绑定列表”

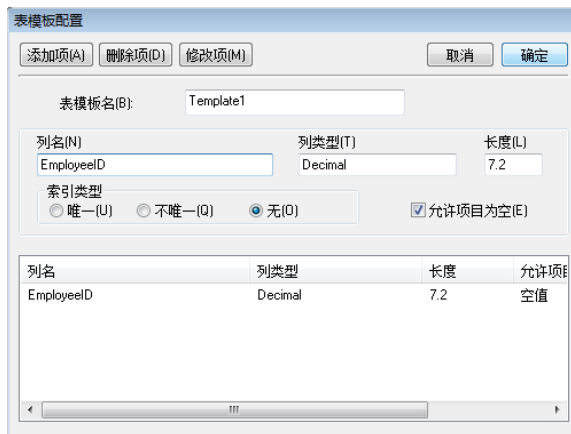
- 1 在**特别菜单**上，指向 **SQL 访问管理器**，然后单击**绑定列表**。此时出现**选择绑定列表**对话框。
- 2 选择要删除的“绑定列表”名称。
- 3 单击**删除**。此时出现一条消息，要求确认是否删除“绑定列表”。
- 4 单击**确定**以删除所选的“绑定列表”。此时再次出现**绑定列表配置**对话框。
- 5 单击**确定**以关闭对话框。

## 定义新表的结构

“表格模板”定义数据库中创建的新表的结构与格式。“表模板”存储在 SQL.DEF 文件中。

### 要创建新的“表模板”

- 1 在**特别菜单**上，指向 **SQL 访问管理器**，然后单击**表模板**。
- 2 单击**新建**。此时出现**表模板配置**对话框。



- 3 在**表模板名**框中，输入“表模板”的名称。  
没有索引时，“表模板名”的最大长度是 32 个字符。如果要创建索引（不论是否唯一），则“表模板名”不得超过 24 个字符。

- 4 在**列名**框中，输入“表模板”的列名。  
列名的最大长度是 30 个字符。
- 5 在**列类型**框中，输入列的数据类型。根据使用的数据库的不同，数据类型选项会有所不同。
- 6 在**索引类型**区域中，选择以下选项之一：
  - **唯一**：每个列值必须是唯一的。
  - **不唯一**：每个列值不要求是唯一的。
  - **无**：无索引。

**备注** 运行包含 `SQLCreateTable()` 函数的脚本时，自动创建一个索引文件。

- 7 选择**允许项目为空**以允许在此列中输入空数据。

**备注** InTouch 不支持空数据。

插入数据时，如果尚未给标记输入值，则根据标记类型指定空值。

数据类型	值
离散	0
整型	0
消息	没有字符的字符串

- 8 单击**添加项**，以便将新的列名、列类型、长度以及索引类型添加到“表模板”。
- 9 单击**确定**，以便保存新的“表模板”配置并关闭对话框。

**要修改“表模板”**

- 1 在**特别菜单**上，指向 **SQL 访问管理器**，然后单击**表模板**。此时出现**选择表模板**对话框。



- 2 选择要修改的“表模板”名，然后单击**修改**。此时出现**表模板配置**对话框。
- 3 修改所需的项目。
- 4 单击**确定**以保存更改并关闭对话框。

### 要删除“表模板”

- 1 在**特别**菜单上，指向 **SQL 访问管理器**，然后单击**表模板**。此时出现**选择表模板**对话框。
- 2 选择要删除的“表模板”名称
- 3 单击**删除**。此时出现一条消息，要求确认是否删除“表模板”。
- 4 单击**是**。此时再次出现**表模板配置**对话框。
- 5 单击**确定**以关闭对话框。

## 使用数据库应用程序

“SQL 访问管理器”支持 Oracle、Microsoft SQL Server 以及 Microsoft Access 数据库。每种数据库都有一些独特的要求。本节包含几个单独的小节，分别介绍如何配置每种数据库与“SQL 访问管理器”之间的连接。

### SQL Server 数据库应用程序

您可以在 InTouch QuickScript 中使用 SQLConnect() 函数来连接 Microsoft SQL Server 数据库。SQLConnect() 函数将用户登录到 SQL Server 数据库并打开一个连接。SQLConnect() 函数使用的连接字符串的格式如下：

```
(SQLConnect(ConnectionId,"<attribute>=<value>;  
<attribute>=<value>;...");
```

*ConnectionID* 参数是包含会话编号的整型标记。几乎每个其它 SQL Access 函数都要使用这个会话编号来引用与 SQL Server 数据库的连接。每调用一次 SQLConnect() 函数，会话编号递增 1。

下表介绍 Microsoft SQL Server 使用的 SQLConnect() 函数属性：

属性	值
Provider	SQLOLEDB
Data Source	安装数据库的服务器名
Initial Catalog	数据库名
User ID	登录 ID，区分大小写
Password	口令，区分大小写

```
"Provider=SQLOLEDB.1;User ID=UserIDStr;  
Password=PasswordStr;Initial Catalog=DatabaseName;Data  
Source=ServerName;"
```

“SQL 访问管理器”将四种类型的 InTouch 标记（离散、整型、实型以及消息）同其它 SQL Server 数据库数据类型关联起来。

数据类型	长度	范围	标记类型
char	8000 个字 符	1 到 131	消息

数据类型	长度	范围	标记类型
int		-2147483648 到 2147483647	整型
float	15 位	-1.79E+308 到 1.79E+308	实型

char 数据类型包含固定长度的字符串。InTouch 消息标记要求使用 char 数据类型。字段长度必须指定。Microsoft SQL Server 数据库支持最大长度为 8000 个字符的 char 字段。不过，InTouch 消息标记限制为最多 131 个字符。如果消息标记值包含的字符超过给数据库字段指定的长度，则在将该字符串插入数据库时会截断它。

int 数据类型代表 InTouch 整型标记。如果未指定字段长度，则长度设置为数据库的缺省值。如果指定了长度，则它的形式是“宽度”。“宽度”确定列的最大位数。

float 数据类型代表 InTouch 实型标记。字段长度设置是由数据库固定下来的。此数据类型不要求指定字段长度。

## Microsoft Access 数据库应用程序

要与 Microsoft Access 通讯，必须通过在 InTouch QuickScript 中执行 SQLConnect() 函数来连接它。

SQLConnect() 函数用于连接到 Microsoft Access 数据库。通过运行包含 SQLConnect() 函数的脚本，可以登录到数据库服务器，并打开一个连接，以便运行其它 SQL 函数。SQLConnect() 使用的连接字符串的格式如下：

```
SQLConnect(ConnectionId,"<attribute>=<value>;
<attribute>=<value>;...");
```

DSN 是 Microsoft Access 使用的一个独特属性，用于确定在“Microsoft ODBC 管理器”中配置的数据源的名称。

```
SQLConnect(ConnectionId,"DSN=MSACC");
```

“SQL 访问管理器”支持的有效数据类型取决于所使用的 ODBC 驱动程序的版本。

数据类型	长度	缺省值	范围	标记类型
text	255 个字符	--	--	消息
number	--	--	--	整型
number	--	--	--	实型

text 数据类型包含固定长度的字符串，并且用于 InTouch “消息” 标记。长度必须指定。Microsoft Access 数据库支持最大长度为 255 个字符的 text 字段。InTouch “消息” 标记限制为 131 个字符。如果消息变量包含的字符超过给数据库字段指定的长度，则在将该字符串插入数据库时会截断它。Microsoft Access ODBC 驱动程序支持每个列名最多 17 个字符。使用 SQLSetStatement( Select Col1, Col2, ...) 时，支持的最大列数为 40。

## Oracle 数据库应用程序

要在 SQL Access 与 Oracle 数据库之间进行通讯，必须通过运行包含 SQLConnect() 函数的脚本来连接它。

### 要与 Oracle 8.0 数据库进行通讯

- 1 确认运行 InTouch 的计算机上是否安装了 Oracle OLEDB Provider (MSDAORA.DLL) 文件。此文件是由随 InTouch 一起安装的 MDAC 来安装的。
- 2 通过在 InTouch 动作脚本中执行 SQLConnect() 函数连接到 Oracle。

SQLConnect() 函数使用的连接字符串的格式如下：

```
SQLConnect(ConnectionId,"<attribute>=<value>;  
<attribute>=<value>;...");
```

下表介绍 Oracle 使用的函数属性：

属性	值
Provider	MSDAORA
User ID	用户名
Password	口令
Data Source	Oracle Server 机器名

```
SQLConnect(ConnectionId, "Provider=MSDAORA; Data  
Source=OracleServer; User ID=SCOTT; Password=TIGER;");
```

下表列出 “SQL 访问管理器” 针对 Oracle 数据库所支持的有效数据类型。

数据类型	长度	缺省值	范围	标记类型
char	2000 个字 符	1 个字符		消息
number	38 位数	38 位数		整型



要将日期与时间记录到 Oracle 8.0 日期字段中，必须使用 `delim()` 函数配置绑定列表。

#### 要将日期与时间记录到 Oracle 日期字段

- 1 在“应用程序浏览器”下的 **SQL 访问管理器**中，双击**绑定列表**。此时出现**绑定列表配置**对话框。
- 2 在**标记名.域名**框中，输入要使用的标记。例如，`DATE_TIME_TAG`。
- 3 在**列名**框中，输入 Oracle 日期字段的名称。如果使用 Oracle 8.0，请使用 `delim()` 函数指定任何分隔符。如果使用 Oracle 9.2 或更高版本，则不需要 `delim()` 函数。
- 4 在 InTouch 应用程序中，创建一个 **QuickScript**，以准备根据当前日期与时间得到的输入数据。例如：

```
DATE_TIME_TAG = "TO_DATE(' " + $DateString + " " +
StringMid($TimeString,1,8) + "', 'mm/dd/yy
hh24:mi:ss')";
```

此 **QuickScript** 在 **WindowViewer** 中运行之后，日期按以下格式出现：

```
TO_DATE('08/22/06 23:32:18' , 'mm/dd/yy hh24:mi:ss')
```

## 在 InTouch 中执行常见的 SQL 操作

InTouch 使用 **SQL Access** 函数与数据库中存储的信息进行交互。通过使用这些 **SQL Access** 函数，可以编写脚本来选择、修改、插入或删除数据库记录。

**SQL** 动作是同步型的。从 InTouch 应用程序中运行数据库 **QuickScript** 时，只有在函数请求的数据库操作完成之后，控制权才返回给 InTouch。

**SQL Access** 函数遵循一些标点符号标准，用以描述与函数关联的参数的类型。使用英文双引号括起的脚本字符串来输入参数 ("Arg1") 时，会完全按照原样使用该字符串。如果未使用英文双引号，则将参数值视位标记名，同时将标记的当前值与该参数关联起来。

大多数 **SQL** 函数都会返回结果码。如果结果码不为零，则函数失败，此时应该采取其它措施。结果码可以由 `SQLErrorMsg()` 函数使用。

通过使用 InTouch **QuickScript** 编辑器，可以将 **SQL** 函数插入到 **QuickScript** 中。将 **SQL** 函数插入到脚本中的一般操作程序包括以下步骤：

要将 SQL 函数添加到脚本中

- 1 启动 InTouch WindowMaker。
- 2 使用 QuickScript 编辑器打开 QuickScript。
- 3 将光标放到脚本中要插入 SQL 函数的位置。
- 4 在函数区域中，单击附件以显示选择函数对话框。
- 5 单击要插入到 QuickScript 中的 SQL 函数。此时该脚本更新并显示所插入的 SQL 函数。

与 SQL Access 函数关联的参数包括：

- *BindList*

对应于 SQL.DEF 文件中定义的“绑定列表”名。

- *ConnectionID*

*ConnectionID* 参数引用某个内存整型标记的名称，该标记存放由 SQLConnect() 函数指定给每个数据库连接的编号 (ID)。

- *ConnectString*

*ConnectString* 确定数据库系统以及任何附加的登录信息。它按照以下格式输入：

```
"DSN=data source name[;attribute=value
[;attribute=value]..."
```

#### Microsoft SQL Server 连接字符串

- **Microsoft OLE DB Provider for SQL Server (建议使用)。**

```
"Provider=SQLOLEDB.1;User ID=sa;
Password=;Initial Catalog=MyDB;Data
Source=MyServer;"
```

OLE DB Provider for SQL Server 是 sqloledb。

- **Microsoft OLE DB Provider for SQL Server (建议使用)**

```
"Provider=SQLOLEDB.1;uid=sa;pwd=;Database=MyDB"
```

- **Microsoft OLE DB Provider for ODBC (使用缺省供应器 MSDASQL for SQL Server)：**

```
"DSN=Pubs;UID=sa;PWD=;"
```

- **Microsoft OLE DB Provider for ODBC (使用缺省供应器 MSDASQL for SQL Server)：**

```
"Data Source=Pubs;User ID=sa;Password=;"
```

#### Oracle 连接字符串

- **Microsoft OLE DB Provider for Oracle (建议使用)**

```
"Provider=MSDAORA;Data Source=ServerName;User
ID=UserIDStr; Password=PasswordStr;"
```

#### Microsoft Access 连接字符串

Microsoft OLE DB Provider for Microsoft Jet (建议使用)。Microsoft.Jet.OLEDB.4.0 是本地 OLE DB Provider for Microsoft Jet (Microsoft Access Database 引擎)。

```
"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=d:\DBName.mdb;User
ID=UserIDStr;Password=PasswordStr;"
```

Microsoft OLE DB Provider for ODBC （使用缺省供应器  
MSDASQL for MS Access）：

```
"Provider=MSDASQL;DSN=DSNStr;UID=UserName;
PWD=PasswordStr;"
```

- *ErrorMsg*

包含对错误的文本描述的消息变量。

- *FileName*

包含信息的文件的名称。

- *MaxLen*

与参数关联的列的最大长度。此参数确定数据是可变字符型还是长可变字符型。如果 *MaxLen* 小于或等于数据库允许的最大字符串长度，则数据是可变字符型。如果大于该数值，则数据是长可变字符型。

- *OrderByExpression*

定义各个列以及是按升序还是降序进行排列。只有列名可用于排序。表达式必须采用以下格式：

**ColumnName [ASC|DESC]**

要按列名的升序方式给所选的表排序：

```
"manager ASC"
```

要按多个列进行排序，则表达式的格式为：

**ColumnName [ASC|DESC],**

**ColumnName [ASC|DESC]**

要按某个列名（如 temperature）的升序、另一个列名（如 time）的降序给所选的表排序：

```
"temperature ASC, time DESC"
```

- *ParameterNumber*

语句中的实际参数编号。

- *ParameterType*

指定的参数的数据类型。有效值是：

类型	描述
Char	用空格填充的固定长度字符串
Var Char	可变长度字符串
Decimal	BCD 数
Integer	4 字节带符号整数
Small integer	2 字节带符号整数
Float	4 字节浮点数
Double Precision Float	8 字节的浮点数
DateTime	8 字节日期时间值
Date	4 字节日期时间值
Time	4 字节日期时间值
No Type	无数据类型

- *ParameterValue*

要设置的实际值。

- *Precision*

是十进制值的精度、字符的最大个数，或日期时间值的字节长度。

- *RecordNumber*

要检索的实际记录号。

- *ResultCode*

大多数 SQL 函数都会返回的整型变量。如果函数执行成功，则返回的 *ResultCode* 是零 (0)；如果失败，则返回负整数。

- *Scale*

是十进制值的小数位数。仅当适用于设置为空的参数时，才需要使用此值。

- *StatementID*

使用高级功能语句时，SQL 返回供内部使用的 *StatementID*。

- *SQLStatement*

实际的语句，例如：

```
ResultCode=SQLSetStatement(ConnectionID, °±Select  
LotNo, LotName from LotInfo°±);
```

- *TableName*

*TableName* 参数包含要在数据库中访问或创建的表的名称。

- *TemplateName*

*TemplateName* 参数是 SQL.DEF 文件中定义表的模板的名称。

- *WhereExpr*

为表中的任一行定义的可以为真或为假的条件。函数仅提取表中条件为真的那些行。表达式必须采用以下格式：

*ColumnName comparison\_operator expression*

---

**备注** 如果列的数据类型是字符，则表达式必须使用英文单引号括起来。

---

下例选择 Name 列包含 EmployeeID 值的所有行：

```
Name='EmployeeID'
```

下例选择包含部件号 100 到 199 的所有行：

```
partno>=100 and partno<200
```

下例选择 temperature 列包含的值大于 350 的所有记录：

```
temperature>350
```

## 连接与断开数据库

通过在脚本中使用 `SQLConnect()` 与 `SQLDisconnect()` 函数，可以连接到 SQL 数据库，以及断开与 SQL 数据库的连接。

### SQLConnect() 函数

您可以在 InTouch QuickScript 中使用 `SQLConnect()` 函数来连接到 *ConnectionString* 参数指定的数据库。

`SQLConnect()` 返回一个值给 *ConnectionID* 参数，后续的所有 SQL 函数都将它用作参数。在脚本中使用 `SQLConnect` 函数之前，必须先在应用程序文件夹中定义“绑定列表”。

#### 类别

SQL

#### 语法

```
[ResultCode=] SQLConnect (ConnectionID,  
    "ConnectionString");
```

#### 参数

##### *ConnectionID*

内存整型标记的名称，用于存放 `SQLConnect()` 函数指定给每个数据库连接的编号 (ID)。

##### *ConnectionString*

确定 `SQLConnect()` 函数中使用的数据库以及任何附加登录信息的字符串。

#### 附注

应用程序文件夹中必须存在“绑定列表” (SQL.DEF 文件)。否则此函数无法正常工作。

如果在 win.ini 文件的 [InTouch] 部分定义了 `SQLTrace=1`，则每次成功执行 `SQLConnect` 时，都将 ADO、供应器以及数据库系统的版本信息记录到 Wonderware Log Viewer。

#### 示例

下面的语句连接到 IBM OS/2 Database Manager 与名称为 SAMPLE 的数据库：

```
[ResultCode=] SQLConnect (ConnectionID, "DSN=OS2DM;  
    DB=SAMPLE");
```

此函数返回一个值给 *ConnectionID* 变量，所有后续的“SQL 函数”都要将此变量用作参数。

```
"DSN=data source name[;attribute=value  
    [;attribute=value]..."
```

### SQLDisconnect() 函数

SQLDisconnect() 函数断开与数据库的连接，并清理为 SQLPrepareStatement() 与 SQLInsertPrepare() 函数获取且尚未释放的所有资源。

类别

SQL

语法

```
[ResultCode=] SQLDisconnect (ConnectionID) ;
```

参数

*ConnectionID*

内存整型标记的名称，用于存放 SQLConnect() 函数指定给每个数据库连接的编号 (ID)。

另请参阅

SQLConnect()

## 创建新表

通过在 InTouch QuickScript 中使用 SQLCreateTable() 函数，可以使用指定的“表模板”中的参数在数据库中创建表。

### SQLCreateTable() 函数

通过在 InTouch QuickScript 中使用 SQLCreateTable() 函数，可以使用指定的“表模板”中的参数在数据库中创建表。“表模板”在 SQL.DEF 文件中定义，该文件包含数据库表的结构。

类别

SQL

语法

```
[ResultCode=] SQLCreateTable ( ConnectionID,  
                                TableName, TemplateName ) ;
```

参数

*ConnectionID*

内存整型标记的名称，用于存放 SQLConnect() 函数指定给每个数据库连接的编号 (ID)。

*TableName*

要创建的数据库表的名称。

*TemplateName*

要使用的模板定义的名称。

示例

下面的 SQLCreatTable() 函数示例使用 OutputVal 模板中定义的列名与数据类型来创建 BATCH1 表：

```
ResultCode=SQLCreateTable(ConnectionID,"BATCH1",  
"OutputVal") ;
```

另请参阅  
SQLConnect()

## 删除表

您可以在 InTouch QuickScript 中使用 SQLDropTable() 函数从数据库中删除某个表。

### SQLDropTable() 函数

您可以在 InTouch QuickScript 中使用 SQLDropTable() 函数从数据库中删除某个表。包含 SQLDropTable() 函数的 QuickScript 执行完毕之后，将不再能够识别该表，它也不会对任何 SQL 语句作出响应。

类别  
SQL

#### 语法

```
[ResultCode=] SQLDropTable ( ConnectionID, TableName );
```

#### 参数

##### *ConnectionID*

内存整型标记的名称，用于存放 SQLConnect() 函数指定给每个数据库连接的编号 (ID)。

##### *TableName*

要从数据库中删除的表的名称。

#### 示例

下面的 SQLDropTable() 函数示例从数据库中删除 BATCH1 表：

```
ResultCode=SQLDropTable(ConnectionID,"BATCH1");
```

另请参阅  
SQLConnect()



## 从表中检索数据

您可以在脚本中使用一组 SQL 函数从数据库中检索数据，并将值写入 InTouch 标记。

- **SQLSelect()** 函数从表中检索信息，并将这些信息以记录的形式放入内存中创建的临时“结果表”。
- **SQLGetRecord()** 函数从当前的选择缓冲区中检索 *RecordNumber* 所指定的记录。
- **SQLNumRows()** 函数返回表中满足前面的 **SQLSelect()** 函数所指定的准则的行数。
- **SQLFirst()** 函数检索由上一个 **SQLSelect()** 函数创建的“结果表”的第一条记录。
- **SQLNext()** 函数检索由上一个 **SQLSelect()** 函数创建的“结果表”的下一条记录。
- **SQLPrev()** 函数检索逻辑表中上一行的数据，获取该行的值并将其放入 InTouch 标记。
- **SQLLast()** 函数检索逻辑表的最后一行，获取该行的值并将其放入 InTouch 标记。
- **SQLEnd()** 函数释放内存，该块内存存储同 **ConnectionId** 关联的“结果表”的内容。

**SQLFirst()**、**SQLPrev()**、**SQLNext()**、**SQLLast()** 以及 **SQLGetRecord()** 函数从逻辑表中指定的行检索数据，并将它保存为 InTouch 标记值。如果某个字段为 **NULL**，则根据标记是模拟型还是消息型，将关联的 InTouch 标记的值设置为零或是零长字符串。

如果数据库中字符串的长度超过 131 个字符，则只有头 131 个字符从数据库中复制到关联的 InTouch 消息标记。

### SQLSelect() 函数

SQLSelect() 函数从表中检索记录。包含 SQLSelect() 函数的脚本处理完毕之后，检索到的记录放入内存中的临时“结果表”。这些记录可以使用 SQLFirst()、SQLLast()、SQLNext() 以及 SQLPrev() 函数进行浏览。

---

**重要** 在包含 SQLSelect() 函数的脚本结束之后，务必调用 SQLEnd() 函数以释放“结果表”占用的内存。

---

类别  
SQL

语法

```
[ResultCode=] SQLSelect ( ConnectionID, TableName,  
                        BindList, WhereExpr, OrderByExpression ) ;
```

参数

*ConnectionID*

内存整型标记的名称，用于存放 SQLConnect() 函数指定给每个数据库连接的编号 (ID)。

*TableName*

要访问的数据库表的名称。

*BindList*

定义所使用的 InTouch 标记以及这些标记关联的数据库列。

*WhereExpr*

为表中的任一行定义的可以为真或为假的条件。SQLSelect() 函数仅从 *WhereExpr* 条件为真的那些行中提取数据。表达式必须采用以下格式：

ColumnName *comparison\_operator* expression.

---

**备注** 如果使用字符表达式进行比较，则表达式必须使用英文单引号括起来。

---

下例选择 **name** 列包含 **EmployeeID** 值的所有行：

```
name='EmployeeID'
```

下例选择包含部件号 100 到 199 的所有行：

```
partno>=100 and partno<200
```

下例选择 **temperature** 列包含的值大于 350 的所有行：

```
temperature>350
```

### WhereExpr - 内存消息标记

```
OrderByExpr - 内存消息标记
Speed_Input - 内存实型 - 用户输入模拟
Serial_Input - 内存消息 - 用户输入字符串
```

模拟示例

```
WhereExpr = "Speed = " + text
(Speed_Input, "#.##");
```

**Speed\_Input** 是数字，因此必须将它转换为文本以便同 **WhereExpr** 字符串串联。

字符串示例

```
WhereExpr = "Ser_No = " +
Serial_input + "'";
```

**Serial\_Input** 是字符串，因此必须用英文单引号括起它的值，例如：**WhereExpr = "Ser\_No=' 125gh' "**；

使用 **like** 语句的字符串示例

```
WhereExpr = "Ser_No like '.'" "
```

使用 **Like** 比较运算符时，可以将 % 字符用作通配符。

使用布尔 **AND** 运算符的字符串与模拟示例

```
WhereExpr = "Ser_No = " + Serial_input + "' " +
" and " + "Speed = " +
text(Speed_Input, "#.##"); OrderByExpr = "";
```

如果顺序无关紧要，请使用如上所示的空字符串。

使用 **WhereExpr** 标记的 **SQLSelect**

```
ResultCode = SQLSelect(Connect_Id, TableName,
BindList,
WhereExpr, OrderByExpr);
ErrorMsg = SQLErrorMsg( ResultCode );
```

内置 **WhereExpr** 的 **SQLSelect** 函数

```
ResultCode = SQLSelect(Connect_Id, TableName,
BindList,
```

```
"Ser_No = '" + Serial_input + "' ",
OrderByExpr);
Error_msg = SQLErrorMsg( ResultCode );
```

### *OrderByExpr*

定义表列中数据的排序方向。只有列名可用于排序，并且表达式必须采用以下格式：

*ColumnName* [ASC|DESC]

下例按照 **manager** 列中数据的升序对表进行排序：

```
"manager ASC"
```

您也可以按多个列进行排序，此时表达式采用以下格式：

*ColumnName* [ASC|DESC],

*ColumnName* [ASC|DESC]

下例按 **temperature** 列的升序与 **time** 列的降序给所选的表排序：

```
"temperature ASC,time DESC"
```

### 示例

下面的语句使用绑定列表 **List1** 从 **BATCH** 表中选择列名类型包含 **cookie** 这个值的记录。它将按 **amount** 列的升序与 **sugar** 列的降序来显示信息：

```
ResultCode=SQLSelect(ConnectionID,"BATCH",
    "List1","type='cookie'", "amount ASC, sugar
    DESC");
```

下面的语句未指定 *WhereExpr* 与 *OrderByExpr* 的值，因此会选择数据库中的所有数据：

```
ResultCode=SQLSelect(ConnectionID,"BATCH",
    "List1", "", "");
```

### 另请参阅

SQLFirst(), SQLConnect(), SQLLast(), SQLNext(),  
SQLPrev(), SQLEnd(), SQLSelect()

## SQLGetRecord() 函数

SQLGetRecord() 函数从当前的选择缓冲区中检索 *RecordNumber* 参数所指定的记录。

类别  
SQL

语法

```
[ResultCode=] SQLGetRecord ( ConnectionID,  
                             RecordNumber ) ;
```

参数

*ConnectionID*

内存整型标记的名称，用于存放 SQLConnect() 函数指定给每个数据库连接的编号 (ID)。

*RecordNumber*

要检索的实际记录号。

示例

```
ResultCode=SQLGetRecord(ConnectionID, 3) ;
```

另请参阅

SQLConnect()

## SQLNumRows() 函数

SQLNumRows() 指出有多少行满足在上一个 SQLSelect() 函数中指定的准则。例如，如果使用 *WhereExpr* 参数来选择所有列名为 AGE（其中 AGE 等于 45）的行，则返回的行数可能是 40 或 4000。这可以确定接下来要处理的函数。

类别  
SQL

语法

```
SQLNumRows ( ConnectionID ) ;
```

参数

*ConnectionID*

内存整型标记的名称，用于存放 SQLConnect() 函数指定给每个数据库连接的编号 (ID)。

示例

下面的语句将选择的行数返回给 NumRows 整型标记：

```
NumRows=SQLNumRows (ConnectionID) ;
```

另请参阅

SQLConnect()

### SQLFirst() 函数

SQLFirst() 函数选择上一个 SQLSelect() 函数创建的“结果表”中的第一条记录。

类别

SQL

语法

```
[ResultCode=] SQLFirst ( ConnectionID ) ;
```

参数

*ConnectionID*

内存整型标记的名称，用于存放 SQLConnect() 函数指定给每个数据库连接的编号 (ID)。

另请参阅

SQLConnect(), SQLSelect()

### SQLNext() 函数

SQLNext() 函数选择上一个 SQLSelect() 函数创建的“结果表”序列中的下一条记录。在脚本中运行 SQLNext() 函数之前，必须先处理 SQLSelect() 函数。

类别

SQL

语法

```
[ResultCode=] SQLNext ( ConnectionID ) ;
```

参数

*ConnectionID*

内存整型标记的名称，用于存放 SQLConnect() 函数指定给每个数据库连接的编号 (ID)。

示例

```
ResultCode=SQLNext (ConnectionID) ;
```

另请参阅

SQLConnect(), SQLSelect()

### SQLPrev() 函数

SQLPrev() 函数选择上一个 SQLSelect() 函数创建的“结果表”中的上一条记录。

类别

SQL

语法

```
[ResultCode=] SQLPrev ( ConnectionID ) ;
```

参数

*ConnectionID*

内存整型标记的名称，用于存放 SQLConnect() 函数指定给每个数据库连接的编号 (ID)。

附注

使用此命令之前，必须先处理 SQLSelect() 函数。

示例

```
ResultCode=SQLPrev (ConnectionID) ;
```

另请参阅

SQLConnect(), SQLSelect()

### SQLLast() 函数

SQLLast() 函数选择上一个 SQLSelect() 函数创建的“结果表”中的最后一条记录。

类别

SQL

语法

```
[ResultCode=] SQLLast ( ConnectionID ) ;
```

参数

*ConnectionID*

内存整型标记的名称，用于存放 SQLConnect() 函数指定给每个数据库连接的编号 (ID)。

示例

```
ResultCode=SQLNext (ConnectionID) ;
```

另请参阅

SQLConnect(), SQLSelect()

### SQLEnd() 函数

SQLEnd() 函数在 SQLSelect() 函数之后运行，以释放用于存储“结果表”内容的内存。

类别

SQL

语法

```
[ResultCode=] SQLEnd ( ConnectionID );
```

参数

*ConnectionID*

内存整型标记的名称，用于存放 SQLConnect() 函数指定给每个数据库连接的编号 (ID)。

另请参阅

SQLConnect(), SQLSelect()

## 向表中写入新记录

通过使用 SQLInsert() 函数，可以将新记录插入到数据库中。

SQLInsert() 函数使用 InTouch 标记的当前值向表中插入一条记录。SQLInsert() 函数在一个步骤的操作中完成语句得准备、插入及结束。

如果与 InTouch 消息标记关联的字符串长度超过为表中相应得文本字段定义的大小，则按照为该字段定义的大小使用消息标记中的字符数。

---

**备注** InTouch 标记不得为 NULL。如果“绑定列表”包含相应的字段，则无法使用这些函数更新数据库中的 NULL 值，或是将 NULL 值插入数据库。通过在不包含某个字段的 INSERT 语句（此语句应该已经定义为允许使用 NULL 值）上使用 SQLExecute，可以将 NULL 值插入该字段。

---

SQL Access 提供另外三个其它函数，可以分别完成预备、插入以及记录插入后的清理。同时使用这些函数时，可以编写包含单个预备和结束语句的脚本，然后根据需要添加任意数量的记录插入语句。如果使用单独的函数而不是 SQLInsert() 函数来插入数据，则可以减少计算机上的资源占用量。



## SQLInsert() 函数

SQLInsert() 函数使用所提供的“绑定列表”中标记的值，向引用的表中插入一条新记录。*BindList* 参数定义要使用哪些 InTouch 标记，以及同这些标记关联的是哪些数据库列。

使用 SQLInsert() 函数来执行预备与插入操作，然后再结束语句。

类别  
SQL

语法

```
[ResultCode=]SQLInsert ( ConnectionID, TableName,  
                        BindList );
```

参数

*ConnectionID*

内存整型标记的名称，用于存放 SQLConnect() 函数指定给每个数据库连接的编号 (ID)。

*TableName*

要访问的数据库表的名称。

*BindList*

定义要使用哪些 InTouch 标记，以及同这些标记关联的是哪些数据库列。

示例

下面的语句使用 List1 中指定的标记值向 ORG 表中插入一条新记录：

```
ResultCode=SQLInsert (ConnectionID,"ORG","List1");
```

## SQLInsertPrepare() 函数

SQLInsertPrepare() 函数在每次运行时创建并预备一条 Insert 语句。此时不处理该 Insert 语句。*StatementID* 参数是一个整型标记，在处理该语句之后，它将包含一个值。

类别  
SQL

语法

```
[ResultCode=]SQLInsertPrepare  
    ( ConnectionID, TableName, BindList, StatementID );
```

参数

*ConnectionID*

内存整型标记的名称，用于存放 SQLConnect() 函数指定给每个数据库连接的编号 (ID)。

*TableName*

要访问的数据库表的名称。

***BindList***

定义要使用哪些 InTouch 标记，以及同这些标记关联的是哪些数据库列。

***StatementID***

使用 SQLPrepareStatement() 函数时 SQL 返回的整数值。

**另请参阅**

SQLConnect(), SQLPrepareStatement()

**SQLInsertExecute() 函数**

SQLInsertExecute() 函数运行先前预备好的 Insert 语句，此语句由 SQLInsertPrepare() 函数指定。

SQLInsertExecute() 函数使用 InTouch 标记的当前值向上一个 SQLInsertPrepare() 函数所确定的表中插入一行。如果

*BindList* 参数包含 MS SQL Server 表的 Identity 关键字段，则必须在运行 SQLInsertExecute() 之前设置 IDENTITY\_INSERT 选项。

*StatementID* 参数包含运行脚本中的上一个 SQLInsertPrepare() 函数时 SQL 返回的整数值。

**类别**

SQL

**语法**

```
[ResultCode=] SQLInsertExecute ( ConnectionID,  
                                BindList, StatementID );
```

**参数*****ConnectionID***

由用户创建的内存整型标记，用于存放 SQLConnect() 函数指定给每个数据库连接的编号 (ID)。

***BindList***

定义要使用哪些 InTouch 标记，以及同这些标记关联的是哪些数据库列。

***StatementID***

使用 SQLPrepareStatement() 函数时 SQL 返回的整数值。

**另请参阅**

SQLConnect(), SQLPrepareStatement()

## SQLInsertEnd() 函数

SQLInsertEnd 函数清理与 SQLInsertPrepare 所创建的 *StatementID* 函数关联的资源。

下例显示在脚本中应该如何指定多个插入函数。

```
ResultCode = SQLSetStatement(ConnectionId, "SET  
IDENTITY_INSERT Products ON");  
ResultCode = SQLExecute(ConnectionId, "", 0);  
ResultCode = SQLInsertPrepare(ConnectionId, TableName, Bindlist,  
StatementID);  
ResultCode = SQLInsertExecute(ConnectionId, Bindlist, StatementID);  
ResultCode = SQLInsertEnd(ConnectionId, StatementID);
```

类别

SQL

语法

```
[ResultCode=] SQLInsertEnd ( ConnectionID,  
                             StatementID );
```

参数

*ConnectionID*

内存整型标记的名称，用于存放 SQLConnect() 函数指定给每个数据库连接的编号 (ID)。

*StatementID*

使用 SQLPrepareStatement() 函数时 SQL 返回的整数值。

另请参阅

SQLConnect(), SQLPrepareStatement()

## 更新表中现有的记录

SQL Access 提供两个函数来使用 InTouch 标记的值更新表记录：

- SQLUpdate()
- SQLUpdateCurrent()

### SQLUpdate() 函数

SQLUpdate() 函数使用 InTouch 标记的当前值来更新表中与 *WhereExpr* 参数设置的条件相匹配的所有行。

类别

SQL

语法

```
[ResultCode=] SQLUpdate ( ConnectionID, TableName,  
                          BindList, WhereExpr );
```

参数

*ConnectionID*

内存整型标记的名称，用于存放 SQLConnect() 函数指定给每个数据库连接的编号 (ID)。

*TableName*

要访问的数据库表的名称。

*BindList*

定义要使用哪些 InTouch 标记，以及同这些标记关联的是哪些数据库列。

*WhereExpr*

为表中的任一行定义的可以为真或为假的条件。此函数仅更新表中条件为真的那些行。表达式必须采用以下格式：

*ColumnName comparison\_operator expression.*

---

**备注** 如果列的数据类型是字符，则表达式必须用英文单引号括起来。

---

下例选择 **name** 列包含 **EmployeeID** 值的所有行：

```
name='EmployeeID'
```

下例选择包含部件号 100 到 199 的所有行：

```
partno>=100 and partno<200
```

下例选择 **temperature** 列包含的值大于 350 的所有行：

```
temperature>350
```

示例

下面的语句将 BATCH 表中批号为 65 的所有记录更新为 *BindList* "List1" 中所指定的标记的当前值：

```
ResultCode=SQLUpdate (ConnectionID, "BATCH",
    "List1", "lotno=65");
```

**备注** 务必确保所有记录都是唯一的。如果表中存在完全相同的记录，则所有相似的记录都会更新。

另请参阅

SQLConnect()

### SQLUpdateCurrent() 函数

SQLUpdateCurrent() 函数使用映射到表字段的 InTouch 标记来更新逻辑表中的当前行，映射由 SQLSelect() 或 SQLExecute() 函数语句中指定的“绑定列表”来完成。如果有其它行与当前行完全相同，则所有这些行都会更新。

一次可以更新最多 54 条完全相同的记录。如果有太多完全相同的行要在 SQL Access 中更新，SQLUpdateCurrent() 函数会返回错误。错误消息类似于：“Microsoft Cursor Engine: Key column information is insufficient or incorrect. Too many rows were affected by update”。

要避免此错误，请在表中创建一个唯一的键字段，使得每行都是唯一的。强烈建议让 SQL Access 使用的所有的表都有唯一的键。对于没有键的表，建议将类型为 AutoNumber (Access) 的字段，或是将用作行标识 (SQL Server) 的整型字段用作主键，以便 SQLUpdateCurrent() 函数一次只更新一行。这个主键字段不必包含在“绑定列表”中。

类别

SQL

语法

```
[ResultCode=] SQLUpdateCurrent ( ConnectionID );
```

参数

*ConnectionID*

内存整型标记的名称，用于存放 SQLConnect() 函数指定给每个数据库连接的编号 (ID)。

示例

```
ResultCode=SQLUpdateCurrent (ConnectionID);
```

另请参阅

SQLConnect()

## 从表中删除记录

您可以使用两个 SQL 函数从数据库表中删除记录。

SQL Access 提供两个函数来删除表记录：

- SQLClearTable() 从表中删除记录
- SQLDelete() 从表中删除与指定的条件匹配的记录

### SQLClearTable() 函数

SQLClearTable() 函数从表中删除所有的记录。它不从数据库中删除表。

**类别**

SQL

**语法**

```
[ResultCode=] SQLClearTable ( ConnectionID,  
    " TableName" ) ;
```

**参数**

*ConnectionID*

内存整型标记的名称，用于存放 SQLConnect() 函数指定给每个数据库连接的编号 (ID)。

*TableName*

要清除其所有记录的表的名称。

**示例**

在下例中 SQLClearTable() 函数从 BATCH1 表中清除所有的记录。

```
ResultCode=SQLClearTable (ConnectionID, "BATCH1") ;
```

**另请参阅**

SQLConnect(), SQLClearStatement()

## SQLDelete() 函数

SQLDelete() 函数从表中删除与 *WhereExpr* 参数指定的条件匹配的所有记录。

类别

SQL

语法

```
[ResultCode=] SQLDelete ( ConnectionID, TableName,  
                          WhereExpr ) ;
```

参数

*ConnectionID*

内存整型标记的名称，用于存放 SQLConnect() 函数指定给每个数据库连接的编号 (ID)。

*TableName*

要清除其中的一些记录的表的名称，这些记录满足 *WhereExpr* 参数指定的条件。

*WhereExpr*

为表中的任一行定义的可以为真或为假的条件。SQLDelete() 函数仅删除 *WhereExpr* 条件为真的那些记录行。表达式必须采用以下格式：

*ColumnName comparison\_operator expression*

---

**备注** SQLDelete() 函数不得包含空的 *WhereExpr* 参数。

---

示例

下面的语句删除 BATCH1 表中批号等于 65 的所有记录：

```
ResultCode=SQLDelete (ConnectionID, "BATCH1",  
                      "lotno=65") ;
```

---

**备注** 如果列的数据类型为字符，则必须使用英文单引号将表达式括起来，如 "MachineID='AG\_LX7\_2'"。

---

另请参阅

SQLConnect()

## 执行参数化语句

使用 `SQLSetStatement()` 与 `SQLAppendStatement()` 函数可以建立动态查询。`SQLSetStatement()` 函数开始一个新的 SQL 语句。它可以是任何有效的 SQL 语句，包括存储过程的名称。

`SQLAppendStatement()` 函数使用字符串得内容来追加 SQL 语句。

### SQLSetStatement() 函数

`SQLSetStatement()` 函数在已建立的连接 *ConnectionID* 上，使用 *SQLStatement* 的内容创建一个 SQL 语句缓冲区。每个 *ConnectionID* 可以有一个 SQL 语句缓冲区。错误在函数返回值中返回。

类别  
SQL

语法

```
[ResultCode=] SQLSetStatement ( ConnectionID,  
                                SQLStatement );
```

参数

*ConnectionID*

内存整型标记的名称，用于存放 `SQLConnect()` 函数指定给每个数据库连接的编号 (ID)。

*SQLStatement*

实际的 SQL 语句，请参阅以下示例。

示例

```
ResultCode=SQLSetStatement (ConnectionID, "Select  
    LotNo, LotName from LotInfo");
```

在下例中，`SQLHandle` 设置为零，因此在运行语句之前，语句不必调用 `SQLPrepare(Connect_Id, StatementID)`。由于 `SQLPrepare` 没有创建 `StatementID` 来正确结束这个 select 语句，因此请使用 `SQLEnd()` 函数而不是 `SQLClearStatement()` 函数。

```
SQLSetStatement( Connect_Id, "Select  Speed,  
    Ser_No from tablename where Ser_No = ' " +  
    Serial_input  + "' ");
```

```
SQLExecute(Connect_Id, 0);
```

在下例中，`StatementID` 由 `SQLPrepareStatement()` 函数创建，并用在 `SQLExecute()` 函数中。要结束这个 `SELECT` 语句，请使用 `SQLClearStatement()` 函数释放资源与句柄。

```
SQLSetStatement( Connect_Id, "Select  Speed,  
    Ser_No from tablename where Ser_No = °0" +  
    Serial_input  + " °0");
```

```
SQLPrepareStatement (Connect_Id, StatementID);
```



```

SQLExecute (Connect_Id, StatementID);
SQLSetStatement ( Connect_Id, "Select  Speed,
    Ser_No from tablename where Ser_No =°Ø" +
    Serial_input  + "°Ø");
SQLPrepareStatement (Connect_Id, StatementID);
SQLExecute (Connect_Id, StatementID);

```

另请参阅

SQLConnect()

### SQLAppendStatement() 函数

SQLAppendStatement() 函数使用字符串内容来追加 SQL 语句。返回值指出在函数调用过程中是否发生了错误。

InTouch 标记可以支持的最大字符串长度是 131 个字符。通常使用 SQLAppendStatement() 函数将附加的字符串串联到语句。

类别

SQL

语法

```

[ResultCode=] SQLAppendStatement ( ConnectionID,
    "SQLStatement" );

```

参数

*ConnectionID*

内存整型标记的名称，用于存放 SQLConnect() 函数指定给每个数据库连接的编号 (ID)。

*SQLStatement*

要追加的实际语句。

示例

```

ResultCode=SQLAppendStatement (ConnectionID,
    "where tablename.columnname=TR-773-01");

```

另请参阅

SQLConnect(), SQLClearStatement()

## 创建语句或从文件中加载现有的语句

您可以使用其它第三方数据库工具来创建查询，然后使用 SQL Access 运行该查询。首先，您必须从第三方数据库工具创建的 .SQL 查询文件中加载 SQL 语句。

```
ResultCode = SQLLoadStatement (ConnectionID,
    "c:\myappdir\lotquery.sql");
```

您使用 SQLLoadStatement() 函数加载 SQL 查询。现在便可以运行该语句了。

### SQLLoadStatement() 函数

SQLLoadStatement() 函数从文件中读取 SQL 语句。

每个文件只能包含一条语句。不过，如果尚未调用 SQLPrepareStatement() 或 SQLExecute() 函数，则可以使用 SQLAppendStatement() 函数给语句追加内容。

**类别**  
SQL

**语法**

```
[ResultCode=] SQLLoadStatement ( ConnectionID,
    FileName );
```

**参数**

*ConnectionID*

内存整型标记的名称，用于存放 SQLConnect() 函数指定给每个数据库连接的编号 (ID)。

*FileName*

包含 SQL 语句的文件的名称。

**附注**

加载语句并获取语句句柄之后，使用 SQLPrepareStatement() 函数准备执行语句。

**示例**

SQL.txt 文件包含以下 SQL 语句：

```
Select ColumnName from TableName where
    ColumnName>100;
```

SQLLoadStatement() 函数从文件中加载该语句。

```
ResultCode=SQLLoadStatement (ConnectionID,
    "C:\SQL.txt")
```

**另请参阅**

SQLConnect(), SQLAppendStatement(), SQLExecute(), SQLPrepareStatement

## 预备语句

使用以下函数，您可以创建任何需要的参数化语句，然后逐个动态填充参数。例如，您可以将常规语句保存到文件、使用 `SQLLoadStatement()` 函数加载它、使用 `QLPrepareStatement()` 函数准备它以获取语句标识，然后使用以下函数填充语句中的各个参数：

- `SQLPrepareStatement()`
- `SQLSetParamChar()`
- `SQLSetParamDate()`
- `SQLSetParamDateTime()`
- `SQLSetParamDecimal()`
- `SQLSetParamFloat()`
- `SQLSetParamInt()`
- `SQLSetParamLong()`
- `SQLSetParamNull()`
- `SQLSetParamTime()`
- `SQLClearParam()`
- `SQLClearStatement()`

要在 SQL 语句上执行参数替换，请在 SQL 语句中要指定后续参数的位置放置一个 "?" 号。此时会预备该语句，在语句中设置参数，然后再运行该语句。

### SQLPrepareStatement() 函数

SQLPrepareStatement() 函数预备要运行的 SQL 语句。它不运行该语句，而只是激活该语句以便设置参数值。

类别

SQL

语法

SQLPrepareStatement(*ConnectionId*, *StatementID*)

参数

*ConnectionID*

内存整型标记的名称，用于存放 SQLConnect() 函数指定给每个数据库连接的编号 (ID)。

附注

预备缺省语句，并返回 *StatementID* (1、2、3 等)。对于需要使用 SQLSetParam{*Type*} 函数来设置参数的语句，这个预备操作非常有用。

## 设置语句参数

“SQL 访问管理器”提供一组函数来修改给 SQL 语句中包含的某个参数指定的值。

### SQLSetParamChar() 函数

在脚本中，SQLSetParmChar() 函数可以用于将指定参数的值设置为指定的字符串。此函数可以在执行之前调用多次，导致参数值设置为发送的所有值的串联结果。长度 0（零）会被忽略。

类别

SQL

语法

SQLSetParamChar(*StatementID*, *ParameterNumber*, *Parameter Value*,  
*MaxLength*);

参数

*StatementID*

使用 SQLPrepareStatement() 函数时 SQL 返回的整数值。

*ParameterNumber*

语句中的参数编号。

*ParameterValue*

要设置为参数值的值。

*MaxLength*

与此参数关联的列的最大宽度。此设置确定参数是可变字符类型还是长可变字符类型。如果 *MaxLength* 小于或等于数据库所允许的最大字符串长度，则此参数是可变字符类型。如果大于，则是长可变字符类型。

另请参阅

SQLPrepareStatement()

### SQLSetParamDate() 函数

SQLSetParamDate() 函数将某个参数的值设置为指定的日期。

类别

SQL

语法

```
SQLSetParamDate(StatementID, ParameterNumber, "Value");
```

参数

*StatementID*

确定查询中的 SQL 语句的整数值。

*ParameterNumber*

整数值，用于确定 *StatementID* 参数所确定的 SQL 语句中的参数。

*Value*

指定给参数的日期（用英文双引号括起来的文字形式），或是值为日期的标记的名称。指定给日期的时间是上午 12:00:00。

示例

本例将第三个语句的第二个参数设置为与 NewDate 标记关联的日期。

```
SQLSetParamDate(3, 2, NewDate);
```

另请参阅

SQLPrepareStatement()

### SQLSetParamDateTime() 函数

SQLSetParamDateTime() 函数将参数的值设置为指定的日期与时间。

类别

SQL

语法

```
SQLSetParamDateTime(StatementID, ParameterNumber, Value,  
                    Precision);
```

参数

*StatementID*

确定查询中的 SQL 语句的整数值。

*ParameterNumber*

整数值，用于确定 *StatementID* 参数所确定的 SQL 语句中的参数。

*Value*

指定给 *ParameterNumber* 参数所确定的参数的日期与时间。

*Precision*

整数，用于指定日期时间值的字符数，此日期时间值指定为参数的值。

**另请参阅**

SQLPrepareStatement()

**SQLSetParamDecimal() 函数**

SQLSetParamDecimal() 函数将参数的值设置为十进制数。

**类别**

SQL

**语法**

```
SQLSetParamDecimal(StatementID, ParameterNumber, Value,  
                   Precision,Scale);
```

**参数***StatementID*

确定查询中的 SQL 语句的整数值。

*ParameterNumber*

整数值，用于确定 *StatementID* 参数所确定的 SQL 语句中的参数。

*Value*

*Value* 可以是代表十进制数 (123.456) 的字符串或 InTouch 消息标记，或是 InTouch 内存实型标记。

建议使用消息标记而不要使用实型标记，以确保参数的精度。不过，如果 *Value* 必须是浮点数（例如，从 DAServer 收到的实数值），则函数仍然起作用。但是由于浮点表示法的限制，可能无法保证很高的精度。

*Precision*

指定数字中总计位数的整数。

*Scale*

指定小数点后面的位数的整数。

**示例**

本例将第三条 SQL 语句的第二个参数设置为 123.456。精度是六位，范围到小数点右侧三位。

```
SQLSetParamFloat(3, 2, 123.456, 6, 3)
```

**另请参阅**

SQLPrepareStatement()

### SQLSetParamFloat() 函数

SQLSetParamFloat() 函数将参数的值设置为 64 位有符号浮点值。

类别

SQL

语法

```
SQLSetParamFloat(StatementID, ParameterNumber, Value);
```

参数

*StatementID*

确定查询中的 SQL 语句的整数值。

*ParameterNumber*

整数值，用于确定 *StatementID* 参数所确定的 SQL 语句中的参数。

*Value*

64 位带符号浮点数，要指定为所指定的参数的值。

示例

本例将第三条 SQL 语句的第二个参数设置为 -5。

```
SQLSetParamFloat(3, 2, -5)
```

另请参阅

SQLPrepareStatement()

### SQLSetParamInt() 函数

SQLSetParamInt() 函数将参数的值设置为 16 位有符号整数。

类别

SQL

语法

```
SQLSetParamInt(StatementID, ParameterNumber, Value);
```

参数

*StatementID*

确定查询中的 SQL 语句的整数值。

*ParameterNumber*

整数值，用于确定 *StatementID* 参数所确定的 SQL 语句中的参数。

*Value*

16 位带符号整数，要指定为所指定的参数的值。

示例

本例将第三条 SQL 语句的第二个参数设置为 -5。

```
SQLSetParamInt(3, 2, -5)
```

另请参阅

SQLPrepareStatement()

### SQLSetParamLong() 函数

SQLSetParamLong() 函数将参数的值设置为 32 位有符号模拟数字。

类别  
SQL

语法

```
SQLSetParamLong(StatementID, ParameterNumber, Value);
```

参数

*StatementID*

确定查询中的 SQL 语句的整数值。

*ParameterNumber*

整数值，用于确定 *StatementID* 参数所确定的 SQL 语句中的参数。

*Value*

32 位带符号模拟数字，要指定为所指定的参数的值。

示例

本例将第一条语句的第三个参数设置为 4.5e12。

```
SQLSetParamLong(1, 3, 4.5e12);
```

另请参阅

SQLPrepareStatement()

### SQLSetParamNull() 函数

SQLSetParamNull() 函数将 SQL 语句中指定的参数设置为 NULL。

类别  
SQL

语法

```
SQLSetParamNull(StatementID, ParameterNumber, ParameterType,  
                Precision, Scale)
```

参数

*StatementID*

确定查询中的 SQL 语句的整数值。

*ParameterNumber*

整数值，用于确定 *StatementID* 参数所确定的 SQL 语句中的参数。



*ParameterType*

整数值，用于指定与 *ParameterNumber* 参数所指定的参数关联的数据的类型。*ParameterType* 参数可以赋以下这些值：

- 0: 字符串
- 1: 日期 / 时间
- 2: 整数
- 3: 浮点数
- 4: 十进制数

*Precision*

与参数的数据类型相关联的数据的精度。

*Scale*

十进制值的小数位数。仅当适用于要设置为空的参数时，才需要使用此值。

**附注**

与 NULL 值的比较由 SQL Server 中的 ANSI\_NULLS 选项控制。在 SQL Server 7.0 中，此选项在创建对象（而不是执行查询）时解析。在 SQL Server 7.0 中创建存储过程时，此选项缺省条件下为 ON，因此 "WHERE MyField = NULL" 之类的子句总是返回 NULL (FALSE)，并且使用此子句的 SELECT 语句不会返回任何行。

要让 = 或 <> 比较操作返回 TRUE 或 FALSE，在创建存储过程时，必须将该选项设置为 OFF。如果 ANSI\_NULLS 没有设置为 OFF，则 SQLSetParamNull() 无法按预期执行。在这种情况下，与 NULL 值的比较操作应该使用以下语法："WHERE MyField IS NULL" 或 "WHERE MyField IS NOT NULL"。

**示例**

此事务集返回 Products 表中 ProductName 不为 NULL 的所有行。

```
SET ANSI_NULLS OFF
GO
CREATE PROCEDURE sp_TestNotNull @ProductParam varchar(255)
AS SELECT * FROM Products WHERE ProductName <>
    @ProductParam
GO
SET ANSI_NULLS ON
GO
```

InTouch 可以运行以下 SQL Access 脚本。

```
ResultCode = SQLSetStatement(ConnectionId, "sp_TestNotNull");
ResultCode = SQLPrepareStatement(ConnectionId, StatementID);
ResultCode = SQLSetParamNull(StatementID, 1, 0, 0, 0);
ResultCode = SQLExecute(ConnectionId, BindList, StatementID);
```

```
ResultCode = SQLFirst(ConnectionId);
```

```
ResultCode = SQLClearStatement(ConnectionId, StatementID);
```

另请参阅

SQLPrepareStatement()

### SQLSetParamTime() 函数

SQLSetParamTime() 函数将指定的时间参数值设置为指定的字符串。

类别

SQL

语法

```
SQLSetParamTime(StatementID, ParameterNumber, Value)
```

参数

*StatementID*

确定查询中的 SQL 语句的整数值。

*ParameterNumber*

*StatementID* 参数所确定的 SQL 语句中的实际参数编号。

*Value*

要设置的实际值。将 *ParameterNumber* 参数所指定的参数设置为时间值。运行该函数的计算机的当前日期同指定的时间包含在一起。

示例

本例将第四条 SQL 语句的第二个参数设置为上午 10:00。

```
ResultCode=SQLSetParamTime( 1, 3, "10:00:00 AM" );
```

另请参阅

SQLPrepareStatement()

## 清除语句参数

SQLClearParam() 函数清除所指定的参数的值。

### SQLClearParam() 函数

SQLClearParam() 函数清除所指定的参数的值。在调用 SQLExecute() 函数以运行查询之前，必须再次调用 SQLSetParam()xxx() 函数中的一个以重新加载参数。

类别

SQL

语法

[ResultCode=]SQLClearParam(*StatementID*,*ParameterNumber*);

参数

*StatementID*

SQLPrepareStatement() 函数运行时返回的整数值。

*ParameterNumber*

*ParameterNumber* 参数确定 SQL 语句中要修改的实际参数。将与 *StatementID* 关联的 *ParameterNumber* 的值设置为零或零长字符串，具体取决于该参数是数值还是字符串。

另请参阅

SQLPrepareStatement(), SQLExecute()

## 执行语句

在 InTouch 脚本中，SQLExecute() 函数可以在运行时用于运行 SQL 查询。

### SQLExecute() 函数

SQLExecute 函数在脚本中运行 SQL 查询。如果语句包含 SELECT，则 *BindList* 参数指定用于绑定数据库列与 InTouch 标记的“绑定列表”的名称。如果“绑定列表”为 NULL，则不进行任何关联标记的操作。

类别

SQL

语法

SQLExecute(*ConnectionID*,*BindList*,*StatementID*);

参数

*ConnectionID*

内存整型标记的名称，用于存放 SQLConnect() 函数指定给每个数据库连接的编号 (ID)。

### *BindList*

*BindList* 参数可以是零长字符串。如果 *StatementID* 与返回行的查询关联，则使用 `SQLExecute()` 的结果更新逻辑表。如果指定实型“绑定列表”，则结果与 *BindList* 参数关联。预先知道 *StatementID* 不与返回行的查询关联时，零长“绑定列表”非常有用。

### *StatementID*

使用 `SQLPrepareStatement()` 函数时 SQL 返回的整数值。

### 附注

错误在函数的返回值中返回。如果语句已经预备好，则应该传递 `prepare` 语句返回的语句句柄。如果语句尚未预备好，则该语句的句柄应该是零。

---

**备注** 对于尚未预备好的语句，`SQLExecute()` 函数只能调用一次。如果语句已经预备好，则可以调用它多次。

---

缺省语句与连接 ID 关联。它可以是文本型 SQL 语句（`SELECT`、`INSERT`、`DELETE` 或 `UPDATE`）、查询（带或不带参数）的名称（对于 MS Access），或存储过程（带或不带参数）的名称（对于 MS SQL Server）。

缺省语句由 `SQLLoadStatement()`、`SQLSetStatement()` 以及 `SQLAppendStatement()` 函数修改。只要指定 `StatementID = 0`，`SQLExecute()` 便使用缺省语句。

### 示例

本例从 `lotquery.sql` 文件中加载 SQL 语句，并将 `SELECT` 语句的结果放入“绑定列表”指定的 InTouch 标记。

```
ResultCode = SQLLoadStatement (ConnectionID,
    "c:\myappdir\lotquery.sql");
ResultCode = SQLExecute(ConnectionID, "BindList", 0);
ResultCode = SQLNext (ConnectionID);
```

对于复杂查询以及超过 131 个字符的字符串表达式，必须使用这个 `SQLSetStatement()` 函数。字符串表达式超过 131 个字符时，请使用 `SQLAppend()` 函数。

```
SQLSetStatement(ConnectionID, "Select Speed, Ser_No from
    tablename where Ser_No = ' " + Serial_input + "' ");
SQLExecute (ConnectionID, "BindList", 0);
```

在上例中，`StatementID` 参数设置为零；因此在执行语句之前不必调用 `SQLPrepareStatement(Connection_Id, StatementID)`。

由于 `SQLPrepare` 语句没有创建 `StatementID` 来正确结束此 `SELECT` 语句，因此请使用 `SQLEnd()` 函数而不是 `SQLClearStatement()` 函数。

```
SQLSetStatement(Connection_Id, "Select Speed, Ser_No from
tablename where Ser_No =' " + Serial_input + "' ");
SQLPrepareStatement(Connection_Id, StatementID);
SQLExecute(Connection_Id, StatementID);
```

在上例中，StatementID 由 SQLPrepareStatement 函数调用创建，并由 SQLExecute 函数使用。要结束此 SELECT 语句，请在脚本中使用 SQLClearStatement() 函数调用，以释放资源与 StatementID。

SQLExecute() 函数支持某些存储过程。例如，假设在数据库服务器 "LotInfoProc" 上创建了一个存储过程，它包含以下 select 语句："Select LotNo, LotName from LotInfo"。

您编写 InTouch QuickScript 根据正在使用的数据库类型运行该存储过程。下例显示运行 SQL Server 数据库的存储过程的脚本语句。

```
ResultCode = SQLSetStatement (ConnectionID,"LotInfoProc");
ResultCode = SQLExecute(ConnectionID, "BindList", 0);
ResultCode = SQLNext (ConnectionID);
{Get results of Select}
```

下例显示运行 Oracle 数据库的存储过程的脚本语句。

```
ResultCode = SQLSetStatement (ConnectionID, "{CALL LotInfoProc}");
ResultCode = SQLExecute(ConnectionID, "BindList", 0);
ResultCode = SQLNext (ConnectionID);
{Get results of Select}
```

**另请参阅**

SQLConnect(), SQLPrepareStatement()

## 释放占用的资源

SQLClearStatement 函数释放与 *StatementID* 所指定的语句关联的数据库资源。

### SQLClearStatement() 函数

SQLClearStatement() 函数释放与 *StatementID* 参数指定的语句关联的数据库资源。

**类别**

SQL

**语法**

```
[ResultCode=]SQLClearStatement(ConnectionID, StatementID);
```

**参数**

*ConnectionID*

内存整型标记的名称，用于存放 SQLConnect() 函数指定给每个数据库连接的编号 (ID)。

*StatementID*

使用 SQLPrepareStatement() 函数时 SQL 返回的整数值。

**另请参阅**

SQLConnect(), SQLPrepareStatement()

## 使用事务集

SQL Access 包含一系列可从数据库中更改、插入、更新或删除记录的事务函数。通常，这些事务在脚本中按事务集的形式分组。事务集同时提交。

### SQLTransact() 函数

SQLTransact() 函数定义一组称为事务集的 SQL 语句的开始。事务集的处理方式与单个事务一样。SQLTransact() 函数运行之后，在 SQLCommit() 函数成功运行之前，所有的后续操作都不会提交给数据库。

#### 类别

SQL

#### 语法

```
[ResultCode=]SQLTransact(ConnectionID)
```

#### 参数

##### *ConnectionID*

内存整型标记的名称，用于存放 SQLConnect() 函数指定给每个数据库连接的编号 (ID)。

#### 示例

此示例事务集包含三个 insert 语句。

```
ResultCode = SQLTransact( ConnectionID);  
ResultCode = SQLInsertPrepare( ConnectionID, TableName, BindList,  
    StatementID );  
ResultCode = SQLInsertExecute( ConnectionID, BindList,  
    StatementID );  
ResultCode = SQLInsertExecute( ConnectionID, BindList,  
    StatementID );  
ResultCode = SQLInsertExecute( ConnectionID, BindList,  
    StatementID );  
ResultCode = SQLInsertEnd( ConnectionID, StatementID );  
ResultCode = SQLCommit( ConnectionID);
```

#### 另请参阅

SQLCommit(), SQLRollback()

### SQLCommit() 函数

SQLCommit() 函数定义事务集的结束。SQLTransact() 函数运行之后，在 SQLCommit() 函数成功运行之前，事务集中所有的所有后续 SQL 语句都不会提交给数据库。

---

**备注** 编写包含 SQLCommit() 函数的 QuickScript 时，请务必小心。处理时间随着事务集中 SQL 语句数量的增加而增加。

---

类别

SQL

语法

[ResultCode=]SQLCommit(*ConnectionID*)

参数

*ConnectionID*

内存整型标记的名称，用于存放 SQLConnect() 函数指定给每个数据库连接的编号 (ID)。

示例

此示例脚本包含执行三次数据库插入操作的事务集。

```
ResultCode = SQLTransact( ConnectionID);  
ResultCode = SQLInsertPrepare( ConnectionID, TableName, BindList,  
    StatementID );  
ResultCode = SQLInsertExecute( ConnectionID, BindList,  
    StatementID );  
ResultCode = SQLInsertExecute( ConnectionID, BindList,  
    StatementID );  
ResultCode = SQLInsertExecute( ConnectionID, BindList,  
    StatementID );  
ResultCode = SQLInsertEnd( ConnectionID, StatementID );  
ResultCode = SQLCommit( ConnectionID);
```

另请参阅



SQLRollback(), SQLTransact(), SQLCommit()

### SQLRollback() 函数

SQLRollback() 函数反转或回滚最新的事务集。事务集是在 SQLTransact() 与 SQLCommit() 函数之间发出的一组命令。

事务集的处理方式与单个事务一样。SQLTransact() 函数运行之后，所有的后续操作都不会提交给数据库。SQLCommit() 函数运行之后，才发生对数据库的查询更改。如果在 SQLCommit() 函数之前运行 SQLRollback() 函数，则它可以回滚事务集。

#### 类别

SQL

#### 语法

[ResultCode=]SQLRollback(*ConnectionID*)

#### 参数

*ConnectionID*

内存整型标记的名称，用于存放 SQLConnect() 函数指定给每个数据库连接的编号 (ID)。

#### 示例

本例回滚脚本中 SQLTransact 函数之前的数据库值。

```
ResultCode =SQLTransact( ConnectionID);  
ResultCode = SQLInsertPrepare( ConnectionID, TableName, BindList,  
    StatementID );  
ResultCode = SQLInsertExecute( ConnectionID, BindList,  
    StatementID );  
ResultCode = SQLInsertEnd( ConnectionID, StatementID );  
ResultCode =SQLRollback( ConnectionID);
```

#### 另请参阅

SQLCommit(), SQLTransact()

## 在运行时打开 ODBC 管理器对话框

使用 `SQLManageDSN()` 函数在 InTouch 应用程序运行期间运行 “Microsoft ODBC 管理器”。

### SQLManageDSN() 函数

`SQLManageDSN` 函数在 InTouch 应用程序运行期间运行 “Microsoft ODBC 管理器” 设置程序。在脚本中，`SQLManageDSN()` 可用于添加、删除及修改 SQL Server 或 Access 数据库现有的数据源名。

#### 类别

SQL

#### 语法

`SQLManageDSN(ConnectionId)`

#### 参数

*ConnectionId*

*ConnectionId* 不使用。保留它是为了与旧版 SQL Access 保持后向兼容性。因此可以给函数传递任何数字。在运行此函数以打开 “ODBC 数据源管理器” 之前，不需要建立数据库连接。

#### 示例

```
SQLManageDSN(0);
```

## 理解 SQL 错误消息

本节介绍如何排解使用 SQL Access 函数的 InTouch 应用程序的疑难问题。第一部分介绍 `SQLErrorMsg()` 函数；并且包含一个表格，列出 SQL 结果码及对应的错误消息。第二部分包含一些列出具体数据库错误消息的表格。

### SQLErrorMsg() 函数

所有的 SQL 函数都会返回一个可用于排解疑难的结果码。  
`SQLErrorMsg()` 函数返回与结果码关联的错误消息，并将它指定为 InTouch 消息标记的值。

类别  
SQL

语法  
`ErrorMsg=SQLErrorMsg(ResultCode);`

参数  
*ResultCode*  
上一个 SQL 函数返回的整数值。`SQLErrorMsg()` 函数将 InTouch 消息标记的值设置为同结果码关联的消息。如需有关同结果码关联的错误消息的详细信息，请参阅第 155 页的“理解 SQL 错误消息”。

附注  
对于此处未记录的结果码，请参阅数据库的文档。此外也可以到 Wonderware Log Viewer 中浏览任何额外的错误消息。  
对于调试 SQL Access 脚本，`win.ini` 文件 [InTouch] 部分所定义的 `SQLTrace=1` 标帜非常有用。

示例  
本例将同“SQL 访问管理器”结果码关联的错误消息指定给 `ErrorMsg` 消息标记。

`ErrorMsg=SQLErrorMsg(ResultCode)`

另请参阅  
`SQLConnect()`

SQL 访问管理器结果码与消息  
下表列出一些常见的 SQL Access 结果码及相应的错误消息和描述：

结果码	错误消息	描述
0	没有发生错误	SQL 函数成功运行，没有发生错误。
-1	< 来自数据库供应器的消息 >	来自供应商数据库的特定错误消息。

结果码	错误消息	描述
-2	无法执行空语句	SQLExecute(ConnectionId, BindList, 0) 运行之前未调用含非空语句的 SQLSetStatement 或 SQLLoadStatement。
-4	返回值为空	从数据库中读取的整数或实数值为空。这仅仅是发送到 Wonderware Log Viewer 的警告消息。
-5	已无更多的行可供获取	已经到达表中的最后一条记录。
-7	参数标识无效	使用有效的参数标识调用 SQLSetParamChar()、SQLSetParamDate()、SQLSetParamDateTime()、SQLSetParamDecimal()、SQLSetParamFloat()、SQLSetParamInt()、SQLSetParamLong()、SQLSetParamNull() 或 SQLSetParamTime() 函数。
-8	参数列表无效	无效参数列表的示例：1, 2, 3, 5（缺 4）。
-9	NULL 参数类型无效	调用了 SQLSetParamNull 函数，但 Type 参数值无效。
-10	不允许更改参数的数据类型	使用不同类型的现有参数调用 SQLSetParamChar()、SQLSetParamDate()、SQLSetParamDateTime()、SQLSetParamDecimal()、SQLSetParamFloat()、SQLSetParamInt()、SQLSetParamLong()、SQLSetParamNull() 或 SQLSetParamTime() 函数。
-11	语句成功执行后不允许再添加参数。	在语句成功运行之后，针对新参数标识调用 SQLSetParamChar()、SQLSetParamDate()、SQLSetParamDateTime()、SQLSetParamDecimal()、SQLSetParamFloat()、SQLSetParamInt()、SQLSetParamLong()、SQLSetParamNull() 或 SQLSetParamTime() 函数。
-12	日期时间格式无效	遇到无效的日期时间格式，例如在执行 SQLSetParamTime()、SQLInsertExecute() 或 SQLUpdateCurrent() 时。
-13	小数格式无效	遇到无效的小数格式，例如在执行 SQLSetParamDecimal()、SQLInsertExecute() 或 SQLUpdateCurrent() 时。

结果码	错误消息	描述
-14	货币格式无效	遇到无效的货币格式，例如在执行 <code>SQLInsertExecute()</code> 或 <code>SQLUpdateCurrent()</code> 时。
-15	此操作的语句类型无效	使用 <code>SQLPrepareStatement()</code> 创建的语句标识调用 <code>SQLInsertEnd</code> ，或是使用 <code>SQLInsertPrepare()</code> 创建的语句标识调用 <code>SQLClearStatement()</code> 。
-1001	内存不足	没有足够的内存来运行此函数。
-1002	无效连接	传递给 <i>ConnectionId</i> 参数的值无效。
-1003	未找到绑定列表	指定的“绑定列表”名不存在。
-1004	未找到模板	指定的“表模板”名不存在。
-1005	内部错误	发生内部错误。致电“技术支持”。
-1006	字符串为空	警告 - 从数据库中读取的字符串为空。这仅仅是发送到 Wonderware Log Viewer 的警告消息。
-1007	字符串被截断	警告 - 从数据库中读取的字符串超过 131 个字符，在选择时被截断。此警告发送到 Wonderware Log Viewer。
-1008	无位置子句	Delete 语句中没有 Where 子句。
-1009	连接失败	如需有关数据库连接失败的详细信息，请查看 Wonderware Log Viewer。
-1010	在连接字符串的 DB= 部分上指定的数据库不存在	指定的数据库不存在。
-1011	没有选定行	在没有先运行 <code>SQLSelect()</code> 或 <code>SQLExecute()</code> 函数的情况下，便调用 <code>SQLNumRows()</code> 、 <code>SQLFirst()</code> 、 <code>SQLNext()</code> 、 <code>SQLLast()</code> 或 <code>SQLPrev()</code> 函数。
-1013	无法找到要加载的文件	调用了 <code>SQLLoadStatement()</code> 函数，但文件名无法找到。

来自供应商数据库的错误消息返回的 *ResultCode* 是 -1。SQL Access 函数的 *ResultCode* 总是 -1，但消息则原封不动地从数据库供应器中复制而来。

对于使用 Oracle 数据库时发生的错误消息，请参阅 Oracle Server 文档以了解特定的错误消息与解决方案。

下表列出使用 Microsoft SQL Server 或 Access 数据库时可能发生的常见错误消息。

错误消息	解决方案
一次不能有多条语句处于活动状态	在调用 <code>SQLSelect()</code> 函数之后，试图运行 SQL 命令。运行 <code>SQLEnd()</code> 以释放 <code>SQLSelect()</code> 占用的系统资源，或是给第二条语句使用单独的 <code>ConnectionId</code> 。
没有足够的内存来处理命令	尝试重新启动客户端工作站。
对象名 [tablename] 无效	正在使用的数据库中没有该表名。尝试使用 DB= 数据库名。

查看 Microsoft SQL Server 文档以了解特定的错误消息与解决方案。

## 保留字列表

本节列出一些关键字，这些关键字不得用于 SQL Access 的“绑定列表”、“表模板”以及 ODBC 接口。

如果将某个保留关键字用作“绑定列表”或“表模板”中的“列名”，则 Wonderware Log Viewer 会记录一条错误消息。错误的类型取决于所用 ODBC 驱动程序以及该关键字出现的位置。例如，最常见的错误之一是在“绑定列表”或“表模板”中将 DATE 与 TIME 用作“列名”。为避免此错误，请使用略有不同的名称，例如 "aDATE" 与 "aTIME"。

保留关键字定义 InTouch SQL Access 所用的“结构化查询语言”（Structured Query Language，简称 SQL）。正在使用的特定 ODBC 驱动程序也会识别出这些关键字。如果无法正确解释 SQL 命令，则“SQL 访问管理器”会生成一条错误消息，这可以在 Wonderware Log Viewer 中查看到。

下面按字母顺序的列表显示 SQL Access 与 ODBC 的保留关键字：

ABSOLUTE	ADA	ADD
ALL	ALLOCATE	ALTER
AND	ANY	ARE
AS	ASC	ASSERTION
AT	AUTHORIZATION	AVG
BEGIN	BETWEEN	BIT
BIT_LENGTH	BY	CASCADE
CASCADED	CASE	CAST
CATALOG	CHAR	CHAR_LENGTH
CHARACTER	CHARACTER_LENGTH	CHECK
CLOSE COALESCE	COBOL	COLLATE
COLLATION	COLUMN	COMMIT
CONNECT	CONNECTION	CONSTRAINT
CONSTRAINTS	CONTINUE	CONVERT
CORRESPONDING	COUNT	CREATE
CURRENT	CURRENT_DATE	CURRENT_TIME
CURRENT_TIMESTAMP	CURSOR	DATE
DAY	DEALLOCATE	DEC
DECIMAL	DECLARE	DEFERRABLE
DEFERRED	DELETE	DESC
DESCRIBE	DESCRIPTOR	DIAGNOSTICS
DICTIONARY	DISCONNECT	DISPLACEMENT
DISTINCT	DOMAIN	DOUBLE

DROP	ELSE	END
ESCAPE	EXCEPT	EXCEPTION
EXEC	EXECUTE	EXISTS
EXTERNAL	EXTRACT	FALSE
FETCH	FIRST	FLOAT
FOR FOREIGN	FORTRAN	FOUND
FROM FULL	GET	GLOBAL
GO	GOTO	GRANT
GROUP	HAVING	HOURL
IDENTITY	IGNORE	IMMEDIATE
IN	INCLUDE	INDEX
INDICATOR	INITIALLY	INNER
INPUT	INSENSITIVE	INSERT
INTEGER	INTERSECT	INTERVALL
INTO	IS	ISOLATION
JOIN	KEY	LANGUAGE
LAST	LEFT	LEVEL
LIKE	LOCAL	LOWER
MATCH	MAX	MIN
MINUTE	MODULE	MONTH
MUMPS	NAMES	NATIONAL
NCHAR	NEXT	NONE
NOT	NULL	NULLIF
NUMERIC	OCTET_LENGTH	OF
OFF	ON	ONLY
OPEN	OPRN	OPTION
OR	ORDER	OUTER
OUTPUT	OVERLAPS	PARTIAL
PASCAL	PLI	POSITION
PRECISION	PREPARE	PRESERVE
PRIMARY	PRIOR	PRIVILEGES
PROCEDURE	PUBLIC	RESTRICT
REVOKE	RIGHT	ROLLBACK
ROWS	SCHEMA	SCROLL
SECOND	SECTION	SELECT
SEQUENCE	SET	SIZE
SMALLINT	SOME	SQL
SQLCA	SQLCODE	SQLERROR
SQLSTATE	SQLWARNING	SUBSTRING



SUM	SYSTEM	TABLE
TEMPORARY	THEN	TIME
TIMESTAMP	TIMEZONE_HOUR	TIMEZONE_MINU
TO	TRANSACTION	TRANSLATE
TRANSLATION	TRUE	UNION
UNIQUE	UNKNOWN	UPDATE
UPPER	USAGE	USING
VALUE	VALUES	VARCHAR
VARING	VIEW	WHEN
WHENEVER	WHERE	WITH
WORK	YEAR	



## 第 4 章

# 使用 16-Pen Trend 向导

您可以使用 InTouch 向导来创建实时与历史趋势，这些趋势最多可显示 16 个标记的数据。16-Pen Trend（16 笔趋势）是可以在 InTouch 安装期间安装的辅助组件。


16-Pen Trend（16 笔趋势）向导的配置同其它 InTouch 图表向导的非常类似。16-Pen Trend（16 笔趋势）向导允许配置以下趋势属性：

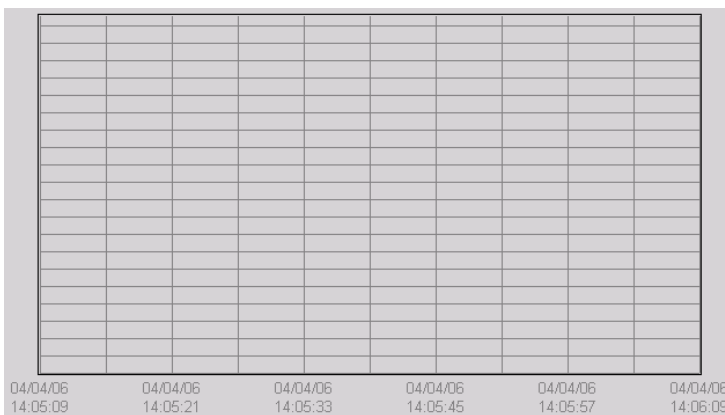
- 指定给每个趋势笔的标记
- 趋势线的宽度与颜色
- 历史趋势的起始和结束日期与时间
- 实时趋势的更新速率与时间跨度
- 指定给趋势标记的最小与最大工程单位
- 主、副趋势时间刻度
- 主、副趋势值刻度

## 创建 16-Pen Trend

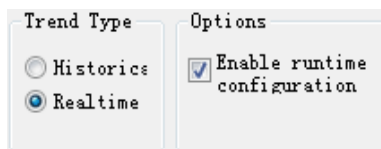
通过从 WindowMaker 中选择 16-Pen Trend（16 笔趋势）向导，可以创建趋势。

### 要创建“16 笔”实时或历史趋势

- 1 从 WindowMaker 中打开一个窗口以放置 16-Pen Trend（16 笔趋势）。
- 2 单击向导工具栏中的向导工具。此时出现 **向导选择** 对话框。
- 3 从向导列表中选择 **Trends**（趋势）。此时 **向导选择** 对话框的右侧面板显示一组趋势向导图标。
- 4  选择 16-Pen Trend（16 笔趋势）向导，然后单击 **确定**。此时 **向导选择** 对话框关闭，您的窗口再次出现。
- 5 单击窗口内部以放置 16-Pen Trend（16 笔趋势）。  
此时向导在窗口中放置一个 16-Pen Trend（16 笔趋势）模板。



- 6 双击该 16-Pen Trend（16 笔趋势）模板，以打开 **PenTrend Control**（笔趋势控件）对话框。
- 7 在 **Trend Type**（趋势类型）区域中，将 **Historical**（历史）或 **Realtime**（实时）选作要创建的趋势的类型。



**PenTrend Control**（笔趋势控件）对话框会根据所选的趋势类型自动显示适当的时间与更新选项。

- 8 在 **Options**（选项）区域中，选择或清除 **Enable runtime configuration**（允许运行时配置）选项。

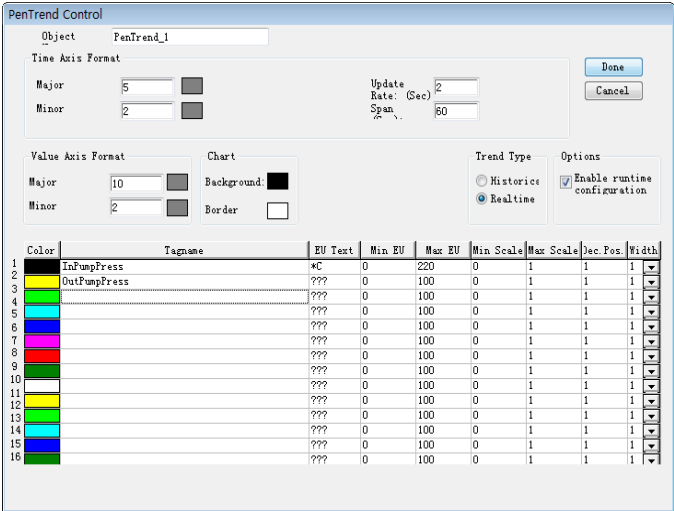
选择此选项时，操作员可以在 16-Pen Trend（16 笔趋势）运行期间修改它的某些属性。

## 配置要在趋势图上显示的标记

您可以使用 16-Pen Trend（16 笔趋势）向导将标记指定给趋势笔。16-Pen Trend（16 笔趋势）向导包括一些列，可以指定要在趋势上显示的标记属性。这些列使用指定给标记的缺省属性值（来自“标记名字典”）。通过在配置趋势时指定其它值，可以覆盖指定的这些标记值。

### 要配置 16-Pen Trend（16 笔趋势）标记

- 1 如果需要，打开包含 16-Pen Trend（16 笔趋势）模板的窗口。
- 2 双击 16-Pen Trend（16 笔趋势）。此时出现 **PenTrend Control**（笔趋势控件）对话框，底部有一块网格区域，可用于指定与趋势笔关联的标记。



- 3 在 **Object Name**（对象名）框中，给 16-Pen Trend（16 笔趋势）指定一个名称。  
缺省名称是 PenTrend\_1，每创建一个新的趋势，名称中的数字便递增 1。
- 4 在 **Tagname**（标记名）框中，输入要与网格左侧列出的笔号关联的标记名。  
双击 **Tagname**（标记名）下的单元格内部时，显示**选择标记**对话框。通过从**选择标记**对话框中选择标记，可以将标记指定给笔。

**备注** 通过选择包含标记名的 **Tagname**（标记名）框，然后按键盘上的空格键，可以删除标记。

- 5 在 **Color**（颜色）列中，单击每个颜色框以打开调色板。给笔选择一种颜色。
- 6 在 **EU Text**（工程单位文本）列中输入文本，在运行时此文本将用作相应的每支笔的笔轴的初始标题文字。

此文本是笔设置为活动时的轴文本。最初 **EU Text**（工程单位文本）列指定为标记的工程单位（来自“标记名字典”）。您可以覆盖 **16-Pen Trend**（16 笔趋势）的这些缺省工程单位。

- 7 在 **Min EU**（最小工程单位）列中，输入给笔指定的最小工程单位值。

最初 **Min EU**（最小工程单位）列显示标记的最小工程单位值（来自“标记名字典”）。您可以指定另一个仅应用于 **16-Pen Trend**（16 笔趋势）的最小工程单位值。

- 8 在 **Max EU**（最大工程单位）列中，输入给笔指定的最大工程单位。

最初 **Max EU**（最大工程单位）列显示标记的最大工程单位值（来自“标记名字典”）。您可以指定另一个仅应用于 **16-Pen Trend**（16 笔趋势）的最大工程单位值。

---

**备注** “最小 / 最大工程单位”对于显示历史趋势数据非常重要。历史趋势显示的工程单位刻度范围是 **0-100%**。

---

- 9 在 **Min Scale**（最小刻度值）列中输入百分比；在运行时，此百分比将用于计算相应的工程单位刻度初始的最小笔轴网格。
- 10 在 **Max Scale**（最大刻度值）列中输入百分比；在运行时，此百分比将用于计算相应的工程单位刻度初始的最大笔轴网格。
- 11 在 **Dec.Pos**（小数位数）列中输入小数位数；在运行时，笔轴网格标签将它用作初始的小数位数。
- 12 在 **Width**（宽度）列中，选择绘制趋势上显示的数据值时使用的笔画宽度（以像素单位）。
- 13 继续执行下一个操作程序，以更新 **16-Pen Trend**（16 笔趋势）的趋势时间与更新速率。

## 配置趋势的时间跨度与更新速率

根据正在创建实时还是历史 16-Pen Trend（16 笔趋势），**Pen Trend Control**（笔趋势控件）对话框显示不同的选项。对于历史趋势，可以设置时间跨度；对于实时趋势，可以设置更新速率。

### 要配置“16 笔”历史趋势的时间跨度

- 1 双击窗口中的“16 笔”历史趋势。此时出现 **Pen Trend Control**（笔趋势控件）对话框，它包含一些选项，可用于设置趋势的起始和结束日期与时间。

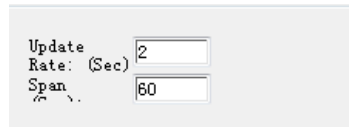


The screenshot shows a dialog box with two text input fields. The first field is labeled 'Start' and contains the text '2006/4/4 2:07:47 PM'. The second field is labeled 'End' and contains the text '2006/4/4 2:08:47 PM'.

- 2 设置历史趋势的起始和结束日期与时间。  
起始和结束日期与时间都采用以下格式：  
MM/DD/YY HH:MM:SS AM/PM

### 要配置“16 笔”实时趋势的更新速率

- 1 双击窗口中的“16 笔”历史趋势对象。此时出现 **Pen Trend Control**（笔趋势控件）对话框，它提供一些选项，可用于设置实时趋势的更新速率与跨度。



The screenshot shows a dialog box with two text input fields. The first field is labeled 'Update Rate: (Sec)' and contains the text '2'. The second field is labeled 'Span' and contains the text '60'.

- 2 在 **Update Rate**（更新速率）框中，输入历史趋势每次刷新间隔的秒数。
- 3 在 **Span**（跨度）框中，输入趋势中显示的实时间隔的秒数。

## 配置趋势的显示选项

通过使用 16-Pen Trend（16 笔趋势）向导，可以配置趋势的外观效果。

### 要配置 16-Pen Trend（16 笔趋势）的显示选项

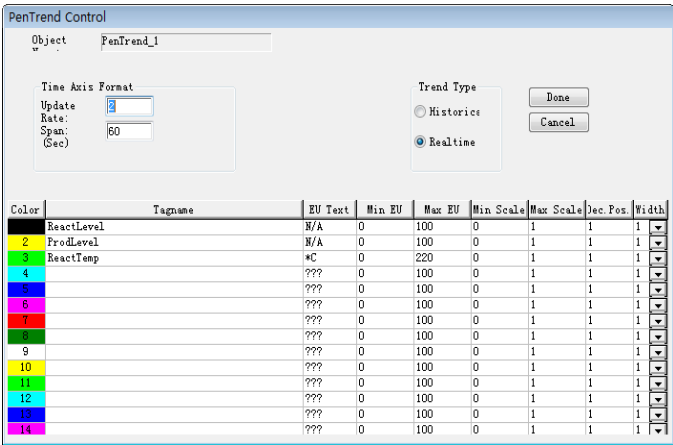
- 1 双击 WindowMaker 中的 16-Pen Trend（16 笔趋势）。此时出现 **Pen Trend Control**（笔趋势控件）对话框。
- 2 在 **Time Axis Format**（时间轴格式）区域的 **Major Divisions**（主刻度）中，输入主时间刻度数。此选项设置趋势横坐标轴的主时间刻度数。
- 3 如果要为主时间刻度线指定另一种颜色，请单击 **Major Divisions**（主刻度）右侧的颜色框，以打开调色板并选择一种颜色。否则会跳过此步骤并接受给主时间刻度线指定的缺省颜色。
- 4 在 **Minor Divisions**（副刻度）框中，输入趋势横坐标轴上显示的副时间刻度数。
- 5 给副时间刻度线选择一种颜色。
- 6 在 **Value Axis Format**（值轴格式）区域中的 **Major Divisions**（主刻度）中，输入趋势的主时间刻度数。此选项设置纵数值轴上显示的主刻度数。
- 7 设置主值刻度的颜色。
- 8 在 **Minor Divisions**（副刻度）框中，输入趋势纵坐标轴上显示的副数值刻度数。
- 9 设置副数值刻度的颜色。
- 10 在 **Chart**（图表）区域中，选择趋势的背景与边框颜色。
- 11 单击 **Done**（完成）将对配置所作的更改保存到 16-Pen Trend（16 笔趋势）中。



## 在运行时更改趋势的配置

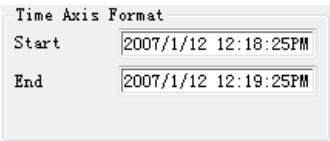
如果选择了 **PenTrend Control**（笔趋势控件）对话框中的 **Enable runtime configuration**（允许运行时配置）选项，则操作员可以在应用程序运行期间更改 16-Pen Trend（16 笔趋势）的一些特征。

在运行时对 16-Pen Trend（16 笔趋势）所作的更改不是永久性的。如果操作员关闭 WindowViewer，然后再次启动应用程序窗口，则 16-Pen Trend（16 笔趋势）将保留原来在 WindowMaker 中定义的配置。下图显示单击正在运行的 16-Pen Trend（16 笔趋势）时出现的 **PenTrend Control**（笔趋势控件）对话框。

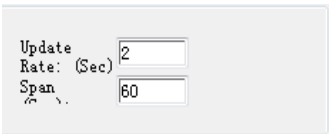


在运行时可以更改以下内容：

- 给趋势笔指定的标记或表达式
- 趋势标记或表达式的特征
- 趋势的类型（“历史”或“实时”）
- 历史趋势的日期与时间范围。



- 实时趋势的更新速率与频率



单击 **Done**（完成）之后，趋势会在当前 WindowViewer 应用程序会话期间保留所作的配置更改。

## 使用脚本控制 16-Pen Trend（16 笔趋势）向导

您可以在 QuickScript 中使用一组函数在运行时控制 16-Pen Trend（16 笔趋势）对象。例如，可以将笔连接到图表上、将新的事件添加到图表、删除或重新绘制网格，以及删除或重新绘制指示器。

### ptGetTrendType() 函数

ptGetTrendType() 函数可以在脚本中用于返回一个值，此值指出 16-Pen Trend（16 笔趋势）的当前模式是显示历史还是实时数据。

**类别**  
笔趋势

**语法**

```
ptGetTrendType(TrendName);
```

**参数**

*TrendName*

趋势的名称。 *TrendName* 必须是字符串常量或消息标记。

**返回值**

返回趋势类型：

0 = 历史趋势

1 = 实时非滚动趋势

2 = 实时趋势

**示例**

下例返回一个值，指出 PumpPress 趋势是显示历史还是实时数据。

```
ptGetTrendType("PumpPress");
```

### ptLoadTrendCfg() 函数

在脚本中， ptLoadTrendCfg() 函数可用于从文件中加载趋势配置值。

**类别**  
笔趋势

**语法**

```
ptLoadTrendCfg(TrendName,FileName);
```

**参数**

*TrendName*

趋势对象的名称。指定给 *TrendName* 的值必须是字符串常量或消息标记。

***FileName***

配置文件的名称。配置文件的文件夹路径必须包含在 *Filename* 参数中。

**示例**

TankFarm 趋势使用 c:\TrendCfg.txt 文件中的值进行配置。

```
ptLoadTrendCfg("TankFarm","C:\TrendCfg.txt");
```

## ptPanCurrentPen() 函数

在脚本中，ptPanCurrentPen() 函数可用于沿纵数值轴向上或向下滚动 16-Pen Trend（16 笔趋势）的笔。垂直滚动由主趋势单位与副趋势单位数（作为参数值指定）确定。

**类别**

笔趋势

**语法**

```
ptPanCurrentPen(TrendName,MajorUnits, MinorUnits);
```

**参数*****TrendName***

趋势对象的名称。*TrendName* 必须是字符串常量或消息标记。

***MajorUnits***

系数，确定按主刻度线定义的单位滚动的次数。负数表示沿纵轴向下滚动。

***MinorUnits***

系数，确定按副刻度线定义的单位额外滚动的次数。负数表示沿纵轴向下滚动。

**示例**

本例将笔向上滚动 1 个主刻度线。

```
ptPanCurrentPen("TrendName", 1, 0);
```

本例将笔向上滚动 0.5 个副趋势刻度。

```
ptPanCurrentPen("TrendName", 0, 0.5);
```

本例将笔向下滚动 2 个主刻度线加上 0.5 个副刻度线。

```
ptPanCurrentPen("TrendName", -2, -.5);
```

本例向上滚动 1 个主刻度线并向下滚动 2 个副刻度线。

```
ptPanCurrentPen("TrendName", 1, -2);
```

## ptPanTime() 函数

在脚本中，`ptPanTime()` 函数可用于根据指定的主或副趋势单位数，沿时间横轴向左或向右滚动 16-Pen Trend（16 笔趋势）的笔。

### 类别

笔趋势

### 语法

```
ptPanTime(TrendName, MajorUnits, MinorUnits);
```

### 参数

#### *TrendName*

趋势对象的名称。*TrendName* 必须是字符串常量或消息标记。

#### *MajorUnits*

系数，确定按横向的主刻度线滚动的次数。负数表示将趋势向左平移。

#### *MinorUnits*

系数，确定按副刻度线定义的单位额外滚动的次数。负数表示将趋势向左平移。

### 附注

开发期间在 **PenTrend Control**（笔趋势控件）中指定的 **Major Division**（主刻度）与 **Minor Division**（副刻度）的设置是计算滚动量的基础。对于时间跨度为 120 秒、主刻度值为 10、副刻度值为 2 的趋势，产生的趋势是每 12 秒有一条主刻度线、每 6 秒有一条副刻度线。`ptPanTime("TrendName",1,0.5)` 函数将时间轴滚动  $1*12 + 0.5*6 = 15$  秒。

### 示例

本例将笔沿趋势的横坐标轴向右滚动 1 个主刻度。

```
ptPanTime("TrendName", 1, 0);
```

本例将笔沿趋势的横坐标轴向右滚动 0.5 个副刻度。

```
ptPanTime("TrendName", 0, 0.5);
```

本例将笔沿趋势的横坐标轴向左滚动 2.5 个主刻度。

```
ptPanTime("TrendName", -2, -0.5);
```

本例将笔沿趋势的横坐标轴向右滚动 1 个主刻度并向左滚动 2 个副刻度。

```
ptPanTime("TrendName", 1, -2);
```

## ptPauseTrend() 函数

在脚本中，ptPauseTrend() 函数可用于使 16-Pen Trend（16 笔趋势）暂时停止更新图形。趋势保持停止状态，直到使用 0 值再次调用 ptPauseTrend。

### 类别

笔趋势

### 语法

```
ptPauseTrend(TrendName, Value);
```

### 参数

#### *TrendName*

趋势对象的名称。*TrendName* 必须是字符串常量或消息标记。

#### *Value*

值为 1 时暂停更新趋势。值为 0 时恢复更新趋势。

### 示例

本例在 *Value* 参数为 1 期间，暂停对 16-Pen Trend（16 笔趋势）的任何进一步更新。

```
ptPauseTrend ("TrendName",1);
```

## ptSaveTrendCfg() 函数

在脚本中，ptSaveTrendCfg() 函数可用于将趋势的当前配置值保存到文件中。

### 类别

笔趋势

### 语法

```
ptSaveTrendCfg(TrendName,FileName);
```

### 参数

#### *TrendName*

趋势对象的名称。必须是字符串常量或消息标记。

#### *FileName*

要用于保存趋势配置值的文件的名称。配置文件的路径可以使用 *Filename* 参数指定。

### 示例

ptSaveTrendCfg() 函数将 PumpTrend 这个 16-Pen Trend（16 笔趋势）的值保存到 c:\Config.txt 文件中。

```
ptSaveTrendCfg ("PumpTrend", "C:\Config.txt")
```

## ptSetCurrentPen() 函数

在脚本中，ptSetCurrentPen() 函数可用于通过指定的编号来选择笔以控制笔轴。

### 类别

笔趋势

### 语法

```
ptSetCurrentPen(TrendName, PenNum);
```

### 参数

#### *TrendName*

趋势的名称。必须是字符串常量或消息标记。

#### *PenNum*

要指定为当前趋势笔的笔的编号 (1-16)。

### 示例

ptSetCurrentPen() 函数将 2 号笔指定为 PumpPress 趋势当前的笔。

```
ptSetCurrentPen("PumpPress",2);
```

## ptSetPen() 函数

在脚本中，ptSetPen() 函数可用于将标记指定给趋势笔。

### 类别

笔趋势

### 语法

```
ptSetPen(TrendName,PenNum,TagName);
```

### 参数

#### *TrendName*

趋势对象的名称。必须是字符串常量或消息标记。

#### *PenNum*

要指定为当前趋势笔的笔号。

#### *TagName*

要指定给趋势笔的标记的名称。

### 示例

ptSetPen() 函数将 PumpInP 标记指定给 PumpPress 趋势的 2 号笔。

```
ptSetPen ("PumpPress",2,"PumpInP");
```

## ptSetPenEx() 函数

在脚本中，ptSetPenEx() 可用于将标记指定给特定的趋势笔，并覆盖在“标记名字典”中给标记指定的配置值。

### 类别

笔趋势

### 语法

```
ptSetPenEx(TrendName, PenNum, TagName, minEu, maxEU,  
           minPercent, maxPercent, Decimal, EU);
```

### 参数

#### *TrendName*

趋势对象的名称。必须是字符串常量或消息标记。

#### *PenNum*

要指定为当前趋势笔的笔号。

#### *TagName*

要指定给趋势笔的标记的名称。

#### *minEU*

指定的标记的最小工程单位值。

#### *maxEU*

指定的标记的最大工程单位值。

#### *minPercent*

在运行时最初用于计算相应工程单位刻度的最小笔轴网格的百分比。

#### *maxPercent*

在运行时最初用于计算相应工程单位刻度的最大笔轴网格的百分比。

#### *Decimal*

趋势中标记值的小数精度。

#### *EU*

标记工程单位的标签。

### 示例

ptSetPenEx() 函数将 PumpInP 标记指定给 PumpPress 趋势的 2 号笔。标记工程单位范围设置为 0 到 1500，其单位是 PSI。网格的百分比范围是 0 到 1，小数精度设置为 2。

```
ptSetPenEx("PumpPress", 2, "PumpInP", 0, 1500, 0, 1, 2, "PSI");
```

## ptSetTimeAxis() 函数

在脚本中，ptSetTimeAxis() 函数可用于设置趋势的起始和结束日期与时间。

### 类别

笔趋势

### 语法

```
ptSetTimeAxis(TrendName,StartDateTime,EndDateTime);
```

### 参数

#### *TrendName*

趋势对象的名称。必须是字符串常量或消息标记。

#### *StartDateTime*

趋势起始的日期与时间。起始日期与时间的格式为：  
mm/dd/yyyy hh:mm:ss AM/PM

#### *EndDateTime*

趋势结束的日期与时间。结束日期与时间的格式为：  
mm/dd/yyyy hh:mm:ss AM/PM。

### 示例

ptSetTimeAxis() 函数将趋势的起始和结束日期与时间设置为从 2007 年 5 月 22 日 8:30 开始，时间跨度是 25 小时。

```
ptSetTimeAxis ("PumpPress", "05/22/2007 08:30:00 AM", "05/23/2007  
09:30:00 AM");
```

## ptSetTimeAxisToCurrent() 函数

在脚本中，ptSetTimeAxisToCurrent() 可用于计算图表当前的时间跨度以及图表的结束时间。

### 类别

笔趋势

### 语法

```
ptSetTimeAxisToCurrent(TrendName);
```

### 参数

#### *TrendName*

趋势对象的名称。*TrendName* 必须是字符串常量或消息标记。

### 示例

ptSetTimeAxisToCurrent() 函数将 PumpPress 趋势的结束日期与时间设置为当前日期与时间。

```
ptSetTimeAxisToCurrent("PumpPress");
```



## ptSetTrend() 函数

在脚本中，ptSetTrend() 函数可用于暂停或重新开始更新 16-Pen Trend（16 笔趋势）。

### 类别

笔趋势

### 语法

```
ptSetTrend(TrendName, EnableUpdates);
```

### 参数

#### *TrendName*

趋势对象的名称。必须是字符串常量或消息标记。

#### *EnableUpdates*

值为 1 时开始更新趋势。值为 0 时停止更新趋势。

### 示例

ptSetTrend() 函数更新 PumpPress 趋势。

```
ptSetTrend("PumpPress",1);
```

## ptSetTrendType() 函数

在脚本中，ptSetTrendType() 函数可用于指定趋势是显示历史还是实时数据。

### 类别

笔趋势

### 语法

```
ptSetTrendType(TrendName, TrendType);
```

### 参数

#### *TrendName*

趋势对象的名称。必须是字符串常量或消息标记。

#### *TrendType*

值为 1 时指定历史趋势。值为 2 时指定实时趋势。

### 示例

ptSetTrendtype() 函数指定 PumpPress 趋势显示实时数据。

```
ptSetTrendType("PumpPress",2);
```

## ptZoomCurrentPen() 函数

在脚本中，ptZoomCurrentPen() 函数可用于更改趋势 Y 轴上显示值的范围。趋势纵数值轴的范围可以按指定的缩放比率增加或减少。

### 类别

笔趋势

### 语法

```
ptZoomCurrentPen(TrendName,ZoomFactor);
```

### 参数

#### *TrendName*

趋势对象的名称。必须是字符串常量或消息标记。

#### *ZoomFactor*

指定大于 1.0 的数字时，可以增加趋势值的范围，具体是将当前的范围限制乘以缩放因子。指定小于 1.0 的缩放因子时，可以减少趋势纵坐标轴上显示值的范围。

### 附注

缩放比率应用于当前笔 Y 轴范围的现有跨度。例如，如果趋势 Y 轴范围开始时为 -50 到 50，则按 2.0 的比率缩放之后，新的范围是 -100 到 100。如果再次按 2.0 的比率进行缩放，则新的范围是 -200 到 200。缩放比率应用于当前生效的范围，而不是原始的 Y 轴范围。

在运行时，每个趋势笔的缩放比率会持久保持。使用

ptSetCurrentPen() 函数从一支笔切换到另一支时，Y 轴值范围会反映所选笔的当前刻度范围。

### 示例

ptZoomCurrentPen 函数将 "PumpPress" 趋势中当前标记的 Y 轴范围加倍。

```
ptZoomCurrentPen("PumpPress",2);
```

## ptZoomTime() 函数

在脚本中，ptZoomTime() 函数可用于更改趋势横轴上显示的时间跨度。

### 类别

笔趋势

### 语法

```
ptZoomTime(TrendName,Zoom);
```

### 参数

#### *TrendName*

趋势对象的名称。必须是字符串常量或消息标记。

#### *Zoom*

指定大于 1.0 的数字时，可以增加趋势横轴上显示的时间段。

指定小于 1.0 的数字时，可以减少横轴上显示的时间段。

### 示例

ptZoomTime() 函数将按 17% 增加趋势横轴上显示的时间段。

```
ptZoomTime("PenTrend_1",1.17);
```

ptZoomTime() 函数按 50% 减少趋势横轴上显示的时间段。例如，如果原始的时间范围设置为 1 小时，则 ptZoomTime() 函数将趋势的时间段减少到 30 分钟。

```
ptZoomTime("PenTrend_1", 0.5);
```



## 第 5 章

# Symbol Factory

Symbol Factory 汇集了 4000 多个工业符号，这些符号可用作 InTouch 应用程序窗口中的可视化元素。Symbol Factory 是可以在 InTouch HMI 安装期间安装的辅助组件。

---

**备注** 通过使用“Archestra 符号编辑器”，可以为与 Application Server 交互的 InTouch 应用程序创建可视化的元素。您还可以使用“符号编辑器”为应用程序创建独立于 InTouch HMI 的可视化智能元素。如需有关“Archestra 符号编辑器”的详细信息，请参阅 Application Server 文档。

---

Wonderware 对于此产品的任何部分都不提供任何形式的担保。您可以将问题报告给“Wonderware 技术支持”。我们强烈建议，在安装或使用任何新的实用程序或应用程序之前，务必备份应用程序与数据。

## 符号类型

Symbol Factory 包含四种类型的向导：

- 图片向导
- 位图向导
- 纹理向导
- InTouch 对象

## 图片向导

Symbol Factory 图片向导是流程图或设备的向量图像。创建应用程序时，可以通过执行以下操作来修改图片向导图像：

- 给图像指定动画
- 水平或垂直翻转图像
- 更改图像的水平与垂直透视图
- 绕轴旋转图像
- 更改图像的填充颜色与图案
- 更改图像线条的尺寸、样式及颜色

## 位图向导

位图向导是位图图像，如 **windows** 图标或文本块。创建应用程序时，可以通过执行以下操作来修改图片位图向导图像：

- 给位图指定动画
- 水平或垂直翻转图像
- 更改位图的水平与垂直长度
- 放置边框或是在位图边框周围放置阴影
- 以 90 度为增量绕轴旋转位图图像
- 定义透明色
- 使用其它颜色替换位图中的最多三种颜色

## 纹理向导

纹理向导类似于位图向导，只是它的大小可以调整，以便形成连续图案。纹理向导通常用于创建窗口的背景，或是用作图形对象的填充。纹理向导从 Symbol Factory 的“纹理”类别中选择。

## InTouch 对象

InTouch 对象是指“按照原样”存储在 Symbol Factory 中的 InTouch 单元或向导。

将 InTouch 对象从 Symbol Factory 中粘贴到 WindowMaker 之后，便不能再在 Symbol Factory 中编辑它。

双击 WindowMaker 中的对象时，如果该对象是单元，则出现**替换标记名**对话框；如果该对象是单独的图形对象，则出现动画链接选择对话框。

## 使用 Symbol Factory

使用 Symbol Factory 向导与使用其它向导非常类似。选择向导，将它放置到窗口中，然后配置它。

### 快速入门

如果熟悉 Symbol Factory，请重温这些提示以快速完成入门：

- 要配置向导的选项，请在窗口中双击它，然后单击 **Symbol Factory by Reichard Software**（Reichard Software 出品的 Symbol Factory）对话框中的 **Options**（选项）。
- 要将向导复制到另一个类别，请将它的略图拖到 **Categories**（类别）窗口中，并放到另一个类别上。要进行移动，请按住 **SHIFT** 键。
- 要编辑向导的描述，请使用鼠标右键单击向导略图。要编辑类别的描述，请使用鼠标右键单击类别。
- 要删除向导，请使用鼠标右键单击向导略图，然后单击 **Delete Symbol**（删除符号）。
- 通过配置 Symbol Factory，一组开发人员可以通过网络使用与充实相同的向导库。
- 特定类别中的所有向导都存储在使用 .cat 扩展名的文件中。Symbol Factory 类别文件通常存放在 c:\program files\wonderware\intouch\symfac 文件夹中。您可以将该文件复制到另一台计算机的 c:\program files\wonderware\intouch\symfac 文件夹中。

## 在窗口中放置 Symbol Factory 向导

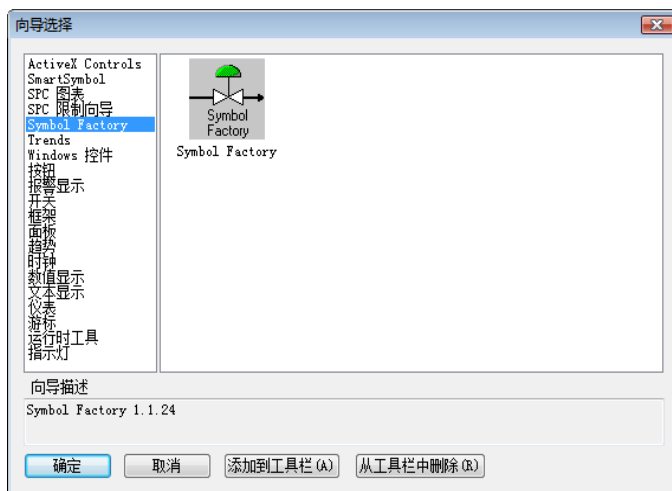
在窗口中放置 Symbol Factory 向导与放置其它向导类似。

要将 Symbol Factory 向导放置到窗口中

1 在 WindowMaker 中打开一个应用程序。



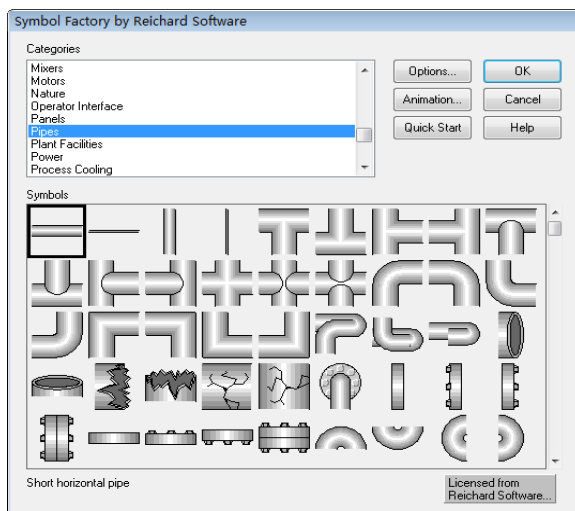
2 单击“向导 /ActiveX 工具栏”中的“向导”图标。此时出现向导选择对话框。



3 在左侧面板显示的向导列表中，单击 **Symbol Factory**。

4 选择显示区域中的 Symbol Factory 向导，然后单击**确定**。

5 在窗口中单击以放置向导。此时出现 **Symbol Factory by Reichard Software**（Reichard Software 出品的 Symbol Factory）对话框。





- 6 在 **Categories**（类别）列表中，选择一个类别。此时 **Symbol**（符号）窗口显示所选类别中的各个向导。
- 7 选择要放置的向导，然后单击 **OK**（确定）。

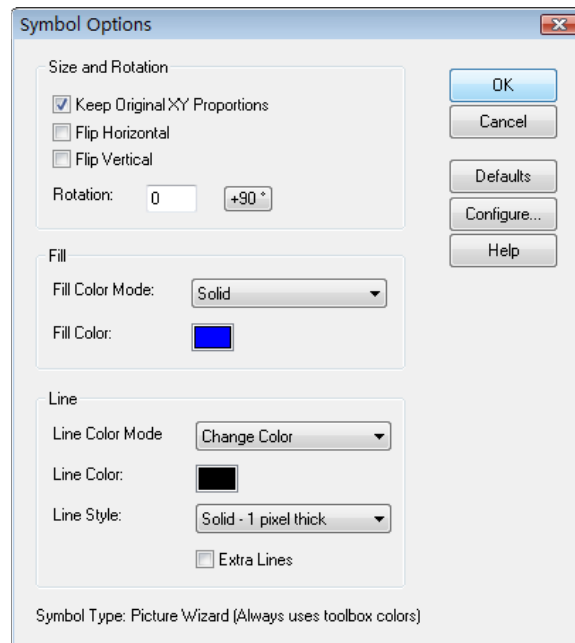
## 配置符号选项

向导的选项随向导的不同而有所不同。由于更改颜色时，每个像素都必须经过扫描并且有可能发生更改，因此将影响位图与纹理的绘制速度。

### 要配置向导选项

- 1 打开包含 **Symbol Factory** 向导的窗口。
- 2 双击该向导。此时出现 **Symbol Factory by Reichard Software**（Reichard Software 出品的 Symbol Factory）对话框。
- 3 保持选中该向导，然后单击 **Options**（选项）。此时出现 **Symbol Options**（符号选项）对话框。

**Symbol Options**（符号选项）对话框上显示的图像属性根据选择要编辑的向导类型而有所不同。下例显示选择图片向导符号时提供的选项。



**提示** 如果选择了 **Configure Symbol Factory**（配置 Symbol Factory）对话框中的 **Enable alternatives to right mouse button**（启用鼠标右键的替代按钮）选项，则会显示 **Edit Symbol**（编辑符号）按钮，单击它可以配置所选的向导。

- 4 在 **Size and Rotation**（大小与旋转）区域中，执行以下任何一项操作：
  - 选择 **Keep Original XY Proportions**（保持原始的 XY 比例）复选框，以保持向导的原始纵横比。
  - 选择 **Flip Horizontal**（水平翻转）复选框，以水平翻转向导。
  - 选择 **Flip Vertical**（垂直翻转）复选框，以垂直翻转向导。
  - 在 **Rotation**（旋转）框中，输入要旋转向导的度数。图片向导可以旋转到任何角度。位图与纹理向导仅能以 90 度为增量（0、90、180 或 270）进行旋转。单击按钮可以自动将旋转角度增加 90 度。
- 5 如果正在配置图片向导，在 **Line**（线条）与 **Fill**（填充）区域中，执行以下任何一项操作：
  - 在 **Fill Color Mode**（填充颜色模式）列表中，单击一种填充类型。单击 **Fill Color**（填充颜色）框以打开调色板。
  - 在 **Line Color Mode**（线条颜色模式）列表中，单击一种线条颜色。单击 **Line Color**（线条颜色）框以访问调色板。
  - 在 **Line Style**（线条样式）列表中，单击一种线条样式。
  - 选择 **Extra Lines**（额外线条）复选框，以便在向导中的渐变边框处添加线条。
- 6 如果正在配置位图或纹理向导，在 **Effects**（效果）与 **Change Colors**（更改颜色）区域中，执行以下任何一项操作：
  - 选择 **Include Border**（包含边框）复选框，以便在向导周围创建黑色边框。
  - 选择 **Include Shadow**（包含阴影）复选框，以便在向导后面创建暗灰色的阴影。
  - 单击每个颜色框，以便打开调色板来更改向导中的各种颜色。
- 7 单击 **OK**（确定）。

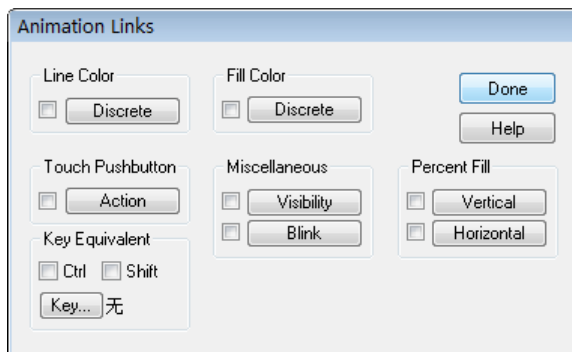
## 给向导设置动画效果

您可以给任何 Symbol Factory 向导设置动画效果。通过 Symbol Factory，可以访问最常用的动画链接。

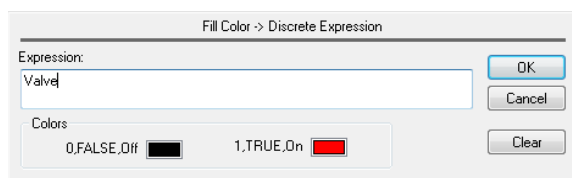
如果需要其它类型的动画链接，则必须分解向导，然后使用标准的 InTouch 动画链接给它设置动画效果。

### 要给向导设置动画效果

- 1 在 Symbol Factory 中选择一个向导；如果已经将向导粘贴到窗口中，则双击该向导。此时出现 **Symbol Factory by Reichard Software**（Reichard Software 出品的 Symbol Factory）对话框。
- 2 单击 **Animation**（动画）。此时出现 **Animation Links**（动画链接）对话框。



- 3 单击要应用于所选向导的每种类型的动画链接所对应的按钮。此时出现表达式对话框。



- 4 在 **Expression**（表达式）窗口中，输入表达式。

在窗口中双击以打开**选择标记**对话框。如果使用现有的标记，则可以双击表达式中的标记打开**标记名字典**，以查看该标记的定义。

如果使用未定义的标记，则关闭表达式对话框时，程序会提示您定义它。

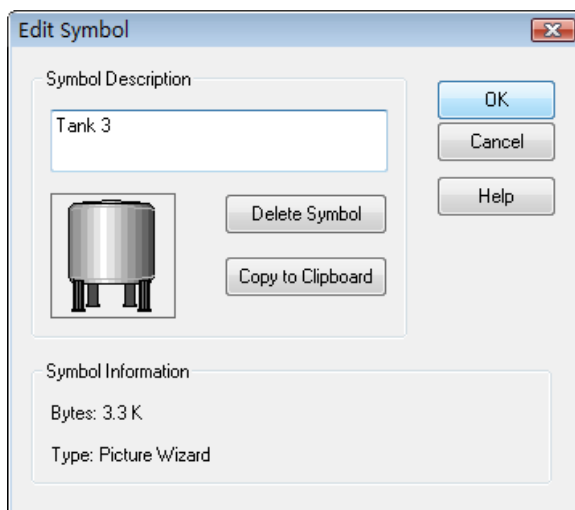
- 5 配置动画链接类型的详细资料。
- 6 单击 **OK**（确定）。

## 编辑符号

您可以更改向导工具提示中的描述、删除向导，或是将向导复制到 Windows 剪贴板。

### 要编辑向导

- 1 在 **Symbol Factory by Reichard Software**（Reichard Software 出品的 Symbol Factory）对话框中，选择包含所需向导的类别。
- 2 使用鼠标右键单击向导。此时出现 **Edit Symbol**（编辑符号）对话框。



- 3 编辑向导。执行以下任何一项操作：
  - 在 **Symbol Description**（符号描述）框中，输入工具提示文本。描述的最大长度是 80 个字符。
  - 单击 **Delete Symbol**（删除符号）以删除向导。
  - 单击 **Copy to Clipboard**（复制到剪贴板）以便将向导复制到 Windows 剪贴板。如果该向导是图片向导，则将作为 Windows 元文件复制。如果该向导是位图向导或纹理向导，则将作为 Windows 位图复制。
- 4 单击 **OK**（确定）。

## 分解向导以便编辑

您可以分解 Symbol Factory 向导以便单独编辑各项。不过在分解向导之后，它的向导属性将会丢失。如果不小心分解了向导，可以使用“撤销”工具重新组装它。

### 要分解向导

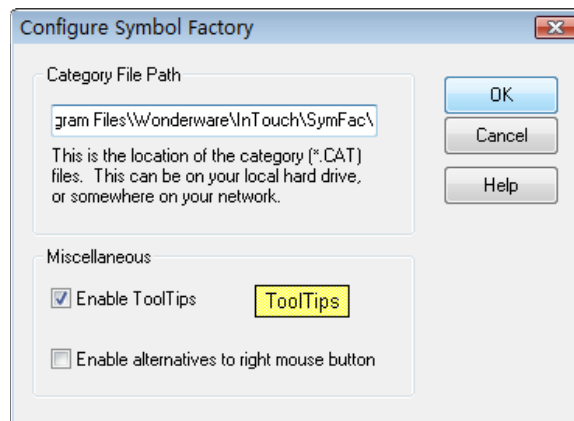
- ◆ 在 WindowMaker 的排列菜单上，单击分解单元。

## 在网络上共享一个类别的符号

通过配置 Symbol Factory，可以允许多个开发人员通过网络使用与充实向导的类别文件。

### 要将类别文件移到网络文件夹中

- 1 在 **Symbol Factory by Reichard Software**（Reichard Software 出品的 Symbol Factory）对话框中，单击 **Options**（选项）。此时出现 **Symbol Options**（符号选项）对话框。
- 2 单击 **Configure**（配置）。此时出现 **Configure Symbol Factory**（配置 Symbol Factory）对话框。



- 3 在 **Category File Path**（类别文件路径）框中，输入要用于保存类别文件的网络文件夹的完整路径。
- 4 单击 **OK**（确定）。

## 使类别只读

将向导文件存储到网络文件夹上时，可能会希望使类别成为只读的，以防止其他用户移动或重命名向导。

### 要使类别只读

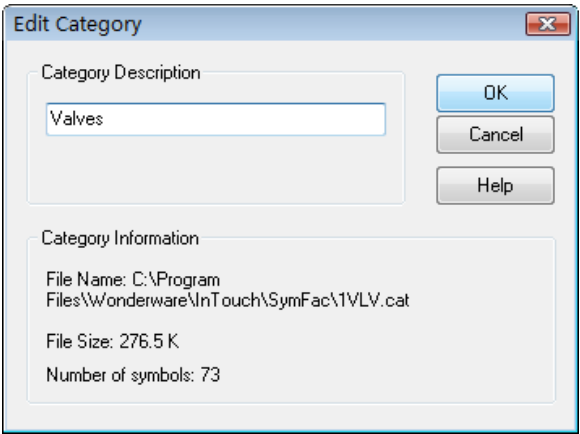
- ◆ 在 Windows 的“资源管理器”中，将文件设置为只读。

## 查看类别属性

您可以查看类别路径、文件大小以及向导个数。

### 要查看类别的属性

- 1 在 **Symbol Factory** 对话框中，使用鼠标右键单击 **Categories**（类别）列表中的类别。此时出现 **Edit Category**（编辑类别）对话框。



- 2 在 **Category Description**（类别描述）框中，输入类别的新描述，然后单击 **OK**（确定）。描述的最大长度是 40 个字符。
- 3 在 **Category Information**（类别信息）区域中，查看属性。

类别信息	描述
Filename (文件名)	类别 (.cat) 文件路径。缺省条件下，此路径是 c:\program files\wonderware\intouch\symfac。
File Size (文件大小)	类别文件的大小，以 KB 为单位。
Number of Symbols (符号个数)	类别中包含的向导总数。最多为 32767。

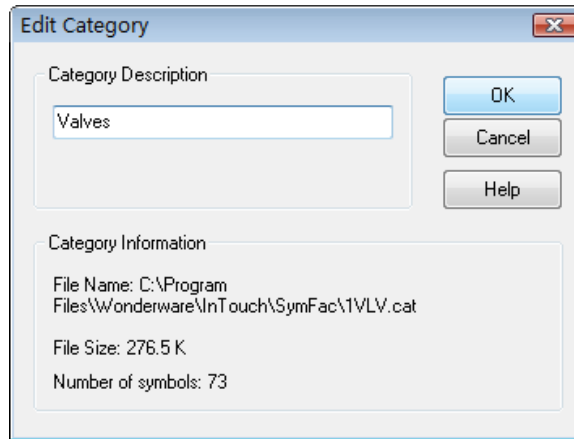
- 4 单击 **OK**（确定）。

## 编辑现有的类别

您只能编辑类别名。

### 要编辑现有的类别

- 1 在 **Symbol Factory** 对话框中，使用鼠标右键单击 **Categories**（类别）列表中的类别。此时出现 **Edit Category**（编辑类别）对话框。



- 2 在 **Category Description**（类别描述）框中，输入类别的新描述，然后单击 **OK**（确定）。描述的最大长度是 40 个字符。
- 3 单击**确定**。

## 删除类别

使用 Windows 的“资源管理器”，可以通过指定类别的文件名来删除类别 (.cat) 文件。

---

**提示** 您可以在 **Edit Category**（编辑类别）对话框中验证该类别的文件名。

---

## 配置 Symbol Factory

配置 Symbol Factory 时，可以指定：

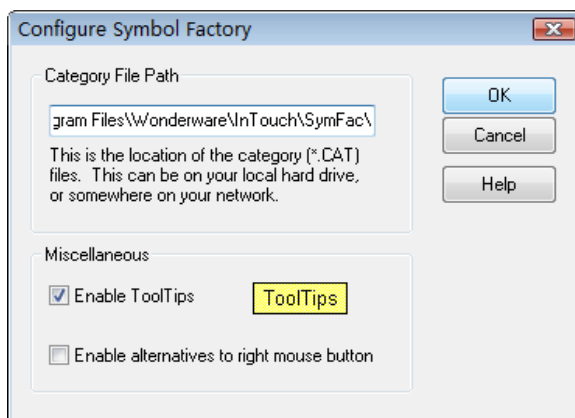
- 选择向导时是否显示工具提示。
- 在 **Symbol Factory by Reichard Software**（Reichard Software 出品的 Symbol Factory）对话框中是否应该出现额外选项。缺省条件下，要编辑类别与向导描述，必须使用鼠标右键单击该项目。
- 类别 (.cat) 文件的位置。每个类别中的所有向导的全部数据都存储在一个类别文件中。由于性能方面的原因，此路径应该仅包含 .cat 文件。要与其他开发人员共享向导，请将此路径设置为网络文件夹。请参阅第 189 页的“在网络上共享一个类别的符号”。

**注意** 请勿将类别文件放置在本地 InTouch 应用程序文件夹中。相反，要将类别文件保存到：

C:\Program Files\Wonderware\intouch\symfac

### 要配置 Symbol Factory

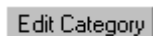
- 1 在 **Symbol Factory by Reichard Software**（Reichard Software 出品的 Symbol Factory）对话框中，单击 **Options**（选项）。此时出现 **Symbol Options**（符号选项）对话框。
- 2 单击 **Configure**（配置）。此时出现 **Configure Symbol Factory**（配置 Symbol Factory）对话框。



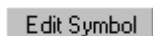
- 3 配置 Symbol Factory。执行以下操作：
  - 在 **Category File Path**（类别文件路径）框中，输入要用于保存 Symbol Factory 类别 (.cat) 文件的位置。
  - 如果要在 **Symbol Factory by Reichard Software**（Reichard Software 出品的 Symbol Factory）对话框中显示向导的工具提示，选择 **Enable ToolTips**（启用工具提示）复选框。



- 如果要添加按钮到 **Symbol Factory by Reichard Software**（Reichard Software 出品的 Symbol Factory）对话框中，选择 **Enable alternatives to right mouse button**（启用鼠标右键的替代按钮）复选框。这些按钮可用于代替鼠标右键进行类别与向导的编辑：

 Edit Category

显示所选类别的 **Edit Category**（编辑类别）对话框。

 Edit Symbol

显示所选向导的 **Edit Symbol**（编辑符号）对话框。

#### 4 单击确定。

## 疑难排解

如果不小心卸载了 Symbol Factory 向导，您需要再次安装它。如需有关安装向导的详细信息，请参阅 *InTouch® HMI 可视化指南* 中的第 5 章“向导”。

如果不小心删除了向导并希望恢复它，则必须检索它。

### 要从类别中检索已删除的向导

- 1 将文件 ~cat.bak 重命名为 temp.cat。
- 2 运行 Symbol Factory 并检查已删除的向导是否已经恢复。将它移到其原始类别中，然后删除 temp.cat 文件。
- 3 如果前一个步骤不起作用，请在按住 CTRL 键的同时使用鼠标右键单击包含已删除的向导的类别。这会压缩类别文件，并创建一个新的备份 ~cat.bak。

执行前面的步骤，直到找到已删除的向导。



# 索引

## 数字

### 16 笔趋势

创建 164

描述 163

配置趋势的时间跨度与更新速率 167

配置趋势的显示选项 168

配置要在图形上显示的标记 165–166

使用脚本进行控制 170–179

在运行时更改配置 169

## H

### 函数

ptGetTrendType() 函数 170

ptLoadTrendCfg() 函数 170

ptPanCurrentPen() 函数 171

ptPanTime() 函数 172

ptPauseTrend() 函数 173

ptSaveTrendCfg() 函数 173

ptSetCurrentPen() 函数 174

ptSetPen() 函数 174

ptSetPenEx() 函数 175

ptSetTimeAxis() 函数 176

ptSetTimeAxisToCurrent() 函数 176

ptSetTrend() 函数 177

ptSetTrendType() 函数 177

ptZoomCurrentPen() 函数 178

ptZoomTime() 函数 179

RecipeDelete() 函数 93

RecipeGetMessage() 函数 99

RecipeLoad() 函数 91

RecipeSave() 函数 92

RecipeSelectNextRecipe() 函数 95

RecipeSelectPreviousRecipe() 函数 96

RecipeSelectRecipe() 函数 94

RecipeSelectUnit() 函数 93

SPCConnect() 函数 23

SPCCreateNewProduct() 函数 66

SPCDatasetDlg() 函数 55

SPCDeleteProduct() 函数 66

SPCDisconnect() 函数 24

SPCDisplayData() 函数 59

SPCLocateScooter() 函数 59

SPCMoveScooter() 函数 60

SPCSampleDeleted() 函数 62

SPCSaveSample() 函数 62

SPCSelectDataset() 函数 56

SPCSelectProduct() 函数 57

SPCSetControlLimits() 函数 67

SPCSetMeasurement() 函数 61

SPCSetProductCollected() 函数 64

SPCSetProductControlLimits() 函数 69

SPCSetProductDisplayed() 函数 57

SPCSetProductRangeLimits() 函数 69

SPCSetProductSpecLimits() 函数 70

SPCSetRangeLimits() 函数 68

SPCSetSpecLimits() 函数 68

SQLAppendStatement() 137

SQLClearParam() 函数 147

SQLClearStatement() 函数 150

SQLClearTable() 函数 134

SQLCommit() 函数 152  
SQLConnect() 函数 118  
SQLCreateTable() 函数 119  
SQLDelete() 函数 135  
SQLDisconnect() 函数 119  
SQLDropTable() 函数 120  
SQLEnd() 函数 128  
SQLErrorMsg() 函数 155  
SQLExecute() 函数 147  
SQLFirst() 函数 126  
SQLGetRecord() 函数 125  
SQLInsert() 函数 129  
SQLInsertEnd() 函数 131  
SQLInsertExecute() 函数 130  
SQLInsertPrepare() 函数 129  
SQLLast() 函数 127  
SQLLoadStatement() 函数 138  
SQLManageDSN() 函数 154  
SQLNext() 函数 126  
SQLNumRows() 函数 125  
SQLPrepareStatement() 函数 140  
SQLPrev() 函数 127  
SQLRollback() 函数 153  
SQLSelect() 函数 122  
SQLSetParamChar() 函数 140  
SQLSetParamDate() 函数 141  
SQLSetParamDateTime() 函数 141  
SQLSetParamDecimal() 函数 142  
SQLSetParamFloat() 函数 143  
SQLSetParamInt() 函数 143  
SQLSetParamLong() 函数 144  
SQLSetParamNull() 函数 144  
SQLSetParamTime() 函数 146  
SQLSetStatement() 函数 136  
SQLTransact() 函数 151  
SQLUpdate() 函数 132  
SQLUpdateCurrent() 函数 133

## L

联系技术支持 10

## P

ptGetTrendType() 函数 170  
ptLoadTrendCfg() 函数 170  
ptPanCurrentPen() 函数 171  
ptPanTime() 函数 172  
ptPauseTrend() 函数 173  
ptSaveTrendCfg() 函数 173  
ptSetCurrentPen() 函数 174  
ptSetPen() 函数 174  
ptSetPenEx() 函数 175  
ptSetTimeAxis() 函数 176  
ptSetTimeAxisToCurrent() 函数 176  
ptSetTrend() 函数 177  
ptSetTrendType() 函数 177  
ptZoomCurrentPen() 函数 178  
ptZoomTime() 函数 179

## R

### Recipe Manager

编辑配方模板文件 85  
编辑配方数据 78  
定义成分名与数据类型 83  
定义成分值 85  
复制一定范围的单元格 81  
将 InTouch 标记映射到成分 84  
描述 75–76  
模板文件 77  
配置编辑网格 78–79  
嵌套配方 89  
清除一定范围单元格的数据 80  
删除模板列 82  
删除模板行 82  
删除配方模板文件 86  
使用 Excel 编辑配方模板文件 87  
使用编辑网格 79  
使用“记事本”编辑配方模板文件 88  
用户界面 76  
在 InTouch 应用程序中使用配方 90  
在模板中插入一列 82  
在“模板定义”模板中插入一行 81

RecipeDelete() 函数 93

RecipeGetMessage() 函数 99

RecipeLoad() 函数 91

RecipeSave() 函数 92  
RecipeSelectNextRecipe() 函数 95  
RecipeSelectPreviousRecipe() 函数 96  
RecipeSelectRecipe() 函数 94  
RecipeSelectUnit() 函数 93

## S

SPCConnect() 函数 23  
SPCCreateNewProduct() 函数 66  
SPCDatasetDlg() 函数 55  
SPCDeleteProduct() 函数 66  
SPCDisconnect() 函数 24  
SPCDisplayData() 函数 59  
SPCLocateScooter() 函数 59  
SPCMoveScooter() 函数 60  
SPCPro  
    从以前的 SPCPro 版本移植 SPC 数据库 72–73  
    更改当前采集的产品 63–64  
    更改数据采集间隔时间 24  
    更改图表使用的数据集 55–56  
    更改图表中显示的产品 56–57  
    滚动图表 58–59  
    将数据输入属性图表 71  
    将图表更新到当前时间 60–61  
    描述 11  
    配置 Access 数据库 13  
    配置 SPC 数据库用户 22  
    配置 SQL Server 数据库 15  
    配置采集与分析选项 17  
    配置数据库 12  
    使用图表向导手工采集数据 25  
    输入计算参数 67–70  
    输入新样本 61–62  
    为数据集配置产品 19  
    在运行时创建新产品 65  
    自动采集 SPC 数据 21  
SPCSampleDeleted() 函数 62  
SPCSaveSample() 函数 62  
SPCSelectDataset() 函数 56  
SPCSelectProduct() 函数 57  
SPCSetControlLimits() 函数 67  
SPCSetMeasurement() 函数 61  
SPCSetProductCollected() 函数 64  
SPCSetProductControlLimits() 函数 69  
SPCSetProductDisplayed() 函数 57  
SPCSetProductRangeLimits() 函数 69  
SPCSetProductSpecLimits() 函数 70  
SPCSetRangeLimits() 函数 68  
SPCSetSpecLimits() 函数 68  
SQL 访问管理器  
    保留字列表 159–161  
    创建新表 119–120  
    创建语句或从文件中加载语句 138  
    从表中检索数据 121–128  
    从表中删除记录 134–135  
    定义新表的结构 107–109  
    更新表中现有的记录 132–133  
    将 InTouch 标记映射到数据库列 104–107  
    理解 SQL 错误消息 155–158  
    连接与断开数据库 118–119  
    描述 101–102  
    清除语句参数 147  
    删除表 120  
    设置 ODBC 数据源 103  
    设置语句参数 140–146  
    释放占用的资源 150  
    使用 Microsoft Access 应用程序 111–112  
    使用 Oracle 应用程序 112–113  
    使用 SQL Server 应用程序 110–111  
    使用事务集 151–153  
    向表中写入新记录 128–131  
    预备语句 139–140  
    在 InTouch 中执行常见的 SQL 操作 113–117  
    在运行时打开“ODBC 管理器”对话框 154  
    执行参数化语句 136–137  
    执行语句 147–149  
SQLAppendStatement() 函数 137  
SQLClearParam() 函数 147  
SQLClearStatement() 函数 150  
SQLClearTable() 函数 134  
SQLCommit() 函数 152  
SQLConnect() 函数 118  
SQLCreateTable() 函数 119  
SQLDelete() 函数 135

SQLDisconnect() 函数 119  
SQLDropTable() 函数 120  
SQLEnd() 函数 128  
SQLErrorMsg() 函数 155  
SQLExecute() 函数 147  
SQLFirst() 函数 126  
SQLGetRecord() 函数 125  
SQLInsert() 函数 129  
SQLInsertEnd() 函数 131  
SQLInsertExecute() 函数 130  
SQLInsertPrepare() 函数 129  
SQLLast() 函数 127  
SQLLoadStatement() 函数 138  
SQLManageDSN() 函数 154  
SQLNext() 函数 126  
SQLNumRows() 函数 125  
SQLPrepareStatement() 函数 140  
SQLPrev() 函数 127  
SQLRollback() 函数 153  
SQLSelect() 函数 122  
SQLSetParamChar() 函数 140  
SQLSetParamDate() 函数 141  
SQLSetParamDateTime() 函数 141  
SQLSetParamDecimal() 函数 142  
SQLSetParamFloat() 函数 143  
SQLSetParamInt() 函数 143  
SQLSetParamLong() 函数 144  
SQLSetParamNull() 函数 144

SQLSetParamTime() 函数 146  
SQLSetStatement() 函数 136  
SQLTransact() 函数 151  
SQLUpdate() 函数 132  
SQLUpdateCurrent() 函数 133  
Symbol Factory  
    编辑符号 188  
    configuring 192–193  
    查看类别属性 190  
    deleting a category 191  
    editing an existing category 191  
    分解向导以便编辑 189  
    符号类型 181–183  
    给向导设置动画效果 187  
    快速入门 183  
    配置符号选项 185–186  
    使类别只读 189  
    troubleshooting 193–??  
    疑难排解 193  
    在窗口中放置向导 184–185  
    在网络上共享一个类别的符号 189

## W

文档惯例 9

## Y

疑难排解

    Symbol Factory 193