

EPSON RC+ 7.0 选件

RC+ API 7.0

Rev.21

SCM231S5557F

翻译版

EPSON RC+ 7.0 选项 RC+ API 7.0 Rev.21

EPSON RC+ 7.0 选件

RC+ API 7.0

Rev.21

©Seiko Epson Corporation 2012-2023

前言

感谢您购买本公司的机器人系统。
本手册记载了正确使用机器人所需的事项。
安装系统之前，请阅读本手册与相关手册。
阅读之后，请妥善保管，以便随时取阅。

本公司的产品均通过严格的测试和检查，以确保机器人系统的性能符合本公司的标准。但是如果在超出本手册所描述的环境中使用本产品，则可能会影响产品的基本性能。

本手册阐述了本公司可以预见的危险和问题。请务必遵守本手册中的安全注意事项，安全正确地使用机器人系统。

商标

Microsoft、Windows、Windows 标识为 Microsoft Corporation 在美国与其他国家的注册商标或商标。其他品牌与产品名称均为各公司的注册商标或商标。

本手册中的商标符号

Microsoft® Windows® 8 operating system

Microsoft® Windows® 10 operating system

Microsoft® Windows® 11 operating system

在本手册中，Windows 8、Windows 10和Windows 11指的是上述各操作系统。在某些情况下，Windows一般是指Windows 8、Windows 10和Windows 11。

注意事项

禁止擅自复印或转载本手册的部分或全部内容。
本手册记载的内容将来可能会随时变更，恕不事先通告。
如果您发现本手册的内容有误或需要改进之处，请不吝斧正。

制造商

SEIKO EPSON CORPORATION

联系方式

有关咨询处的详细内容，请参阅下记手册序言中的“销售商”。

机器人系统 安全手册 请首先阅读本手册

1. 简介	1
1.1 功能	1
2. 安装	2
2.1 逐步说明	2
2.2 安装内容	2
3. 操作入门	4
3.1 Visual Basic 使用入门	4
3.2 Visual C# 使用入门	5
3.3 Visual C++ 使用入门	6
3.4 在 Visual C++ 2017 创建失败 (MSB8036) 的对策	9
4. 环境	11
4.1 开发环境	11
4.1.1 开发启动	11
4.1.2 Spel 类实例初始化	11
4.1.3 Spel 类实例终止	11
4.1.4 开发循环	11
4.2 生产设备	12
4.2.1 运行时打开 EPSON RC+ 7.0	12
4.2.2 使用 EPSON RC+ 7.0 对话框和窗口	12
4.2.3 目标系统上的安装	12
5. 执行方法、程序和任务	13
5.1 执行方法	13
5.1.1 使用多线程	13
5.2 执行 SPEL+ 程序	18
5.3 执行 SPEL+ 任务	18
5.4 中止所有任务	19
6. 事件	20
6.1 概述	20
6.2 系统事件	20
6.3 来自 SPEL+ 的用户事件	20
7. 错误处理	22
7.1 Spel 方法的错误	22
8. 处理暂停与继续	24
8.1 暂停状态	24
8.2 捕获 Pause 事件	24
8.3 执行 Pause	25
8.4 暂停后继续	26
8.5 暂停后中止	26

9. 处理紧急停止	27
9.1 使用系统 EStop 事件	27
10. EPSON RC+ 7.0 窗口和对话框	28
10.1 窗口	28
10.2 对话框	29
11. 播放视频	30
使用多视频显示	31
12. 使用 AsyncMode	33
13. SPELCom_Event	35
14. RCAPINet 参考	36
14.1 Spel 类	36
14.2 Spel 类属性	36
14.3 Spel 类方法	65
14.4 Spel 类事件	359
14.5 SPELVideo 控制	363
14.6 SPELVideo 控制属性	363
14.7 SPELVideo 控制方法	366
14.8 SPELVideo 控制事件	367
14.9 SpelConnectionInfo 类	367
14.10 SpelControllerInfo 类	367
14.11 SpelException 类	368
14.12 SpelOptionInfo 类	369
14.13 SpelPoint 类	369
14.13.1 SpelPoint 属性	371
14.13.2 SpelPoint 方法	372
14.14 SpelRobotInfo 类	373
14.15 SpelTaskInfo 类	373
14.16 枚举	374
14.16.1 SpelArmDefMode 枚举	374
14.16.2 SpelArmDefType 枚举	374
14.16.3 SpelAxis 枚举	374
14.16.4 SpelBaseAlignment 枚举	374
14.16.5 SpelCalPlateType 枚举	374
14.16.6 SpelConnectionType 枚举	374
14.16.7 SpelDialogs 枚举	375
14.16.8 SpelElbow 枚举	375
14.16.9 SpelEvents 枚举	375
14.16.10 SpelForceAxis 枚举	375
14.16.11 SpelForceCompareType 枚举	376
14.16.12 SpelForceProps 枚举	376

14.16.13 SpelHand 枚举	377
14.16.14 SpellOLabelTypes 枚举	377
14.16.15 SpelLocalDefType 枚举.....	377
14.16.16 SpelOperationMode 枚举.....	378
14.16.17 SpelOptions 枚举	378
14.16.18 SpelOptionStatus 枚举.....	378
14.16.19 SpelRobotPosType 枚举	378
14.16.20 SpelRobotType 枚举	379
14.16.21 SpelShutdownMode 枚举.....	379
14.16.22 SpelSimObjectType 枚举	379
14.16.23 SpelSimProps 枚举	379
14.16.24 SpelStopType 枚举	380
14.16.25 SpelTaskState 枚举.....	380
14.16.26 SpelTaskType 枚举	380
14.16.27 SpelToolDefType 枚举.....	380
14.16.28 SpelToolDefType3D 枚举	381
14.16.29 SpelUserRights 枚举.....	381
14.16.30 SpelVDefShowWarning 枚举	381
14.16.31 SpelVisionImageSize 枚举.....	382
14.16.32 SpelVisionObjectTypes 枚举.....	382
14.16.33 SpelVisionProps 枚举	383
14.16.34 SpelWrist 枚举.....	383
14.16.35 SpelWindows 枚举	383
14.17 Spel 错误代码和消息.....	383
15. 32 位和 64 位应用	384
16. 使用 LabVIEW VI Library	385
16.1 概述	385
16.2 安装	385
16.3 工具和控制面板.....	386
16.4 操作入门.....	388
16.5 在 Spel+ 项目中作业	389
16.6 播放视频.....	390
16.7 VI 参考.....	391
17. 在 RCNetLib 使用 LabVIEW	501
17.1 概述	501
17.2 初始化	501
17.2.1 添加 Spel 类的构造函数节点.....	501
17.2.2 初始化 Spel 类实例	502
17.2.3 连接至控制器和设置项目	502
17.3 使用 Spel 属性和方法.....	502
17.4 关闭	502
17.5 使用对话框和窗口	502

18. 如何从一台 PC 控制多个控制器	503
18.1 概述.....	503
18.1.1 系统要求.....	503
18.1.2 PC 和控制器连接	504
18.2 控制多个控制器的限制事项	505
18.2.1 控制器选件的限制事项.....	505
18.2.2 仿真器限制	505
18.3 多个控制器连接的样本程序	506
18.3.1 控制器连接设置.....	506
18.3.2 项目设置.....	506
18.3.3 使用 Visual Basic 的样本程序	507
18.3.4 使用 Visual C# 的样本程序	509

1. 简介

EPSON RC+ 7.0 选件 RC+ API 允许您使用 Microsoft Visual Basic 或支持用以运行机器人应用的 .NET 技术的任何其他语言。这样您便能够创建复杂的用户界面，使用数据库以及与 .NET 配套使用的第三方产品。

也包含 LabVIEW 库。

1.1 功能

RC+ API 包支持以下功能：

- .NET 库和 LabVIEW 库。
- 支持 32 位和 64 位应用。
- 从多个控制器控制多个机器人、I/O 和任务的属性和方法。
- 视觉和力觉*命令的执行方法。
 - * 力觉与力传感器不同。
API 手册中介绍的力觉方法和属性不可用于力传感器。若要使用力传感器的命令，使用 Xpt 方法或执行 SPEL 函数。
API 不支持 EPSON RC+ 选件 Force Guide。
- 支持通过多线程并行执行异步命令。
- .NET 应用可使用多个 EPSON RC+ 7.0 窗口和对话框，包括：
 - 机器人管理器
 - IO 监视器
 - 任务管理器
 - 仿真器
 - 控制器工具对话框

开发期间，EPSON RC+ 7.0 可与 Visual Basic 一起运行。

在生产设备中，EPSON RC+ 7.0 可在后台运行。

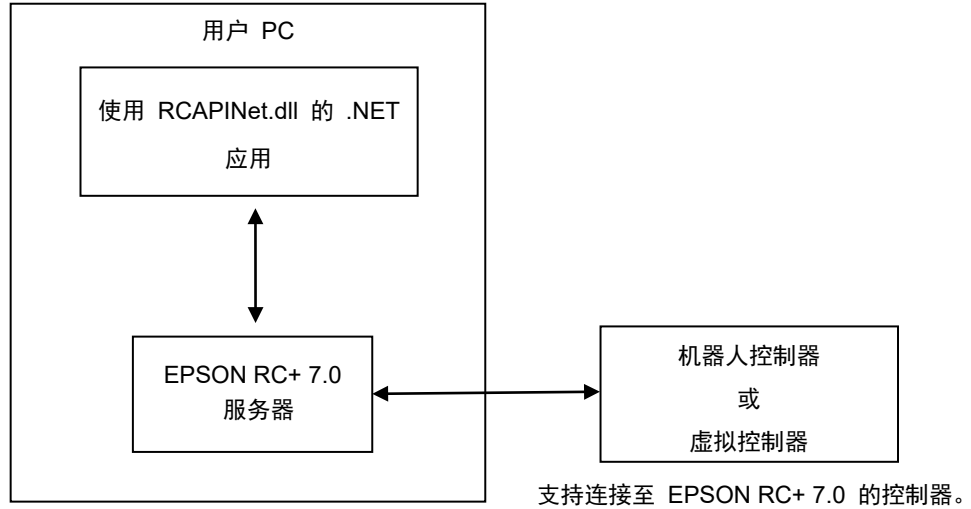


RC+ API 支持“.NET Framework Library”和“LabVIEW VI Library”。

不支持“.NET Core”和“.NET”5 或更高版本。

2. 安装

下图所示为采用 RC+ API 系统的基本结构。



.NET 库的 RC+ API 基本结构

EPSON RC+ 7.0 是 RCAPINet 库的进程外服务器。

每个 RCAPINet Spel 类的实例可以启动一个 EPSON RC+ 7.0 实例。

2. 安装

请遵守本章说明，以助于确保正确安装 RC+ API 软件。

启动前，确保已关闭所有 Windows 应用。

2.1 逐步说明

- (1) 安装以下任意一个：
Visual Studio 2012, 2013, 2015, 2017, 2019
(包括 Enterprise、Professional、Community 或 Express Edition)
LabVIEW 2009 或更高版本
- (2) 安装 EPSON RC+ 7.0。
- (3) 若使用 LabVIEW，安装 LabVIEW VI 库。
- (4) 确保您将要使用的控制器中的 RC+ API 软件密钥已激活。有关激活控制器选项的方式信息，请参阅 EPSON RC+ 7.0 用户指南。

这样便完成了 RC+ API 的安装。

2.2 安装内容

安装期间会在您的 PC 中安装下表所示的目录和文件。

目录和文件	描述
\\EPSONRC70\\API\\VS20xx\\VB\\DEMOS	Visual Basic .NET 演示
\\EPSONRC70\\API\\VS20xx\\VCS\\DEMOS	Visual C# .NET 演示
\\EPSONRC70\\API\\VS20xx\\VC\\DEMOS	Visual C++ .NET 演示
\\EPSONRC70\\API\\LabVIEW	LabVIEW VI 库安装程序

\EPSONRC70\PROJECTS\API_Demos	EPSON RC+ 7.0 演示项目
\EPSONRC70\EXE\RCAPINet.dll	RCAPINet 类库(32 位或 64 位)

3. 操作入门

本章介绍了以下开发环境下的操作入门信息。

- Visual Basic .NET
- Visual C# .NET
- Visual C++ .NET

RC+ API 随附有演示程序。建议您浏览演示程序，以进一步熟悉产品。

LabVIEW 用户此处请参阅章节 16. *使用 LabVIEW VI 库*，了解使用入门和使用库的说明。

首次在 Visual C++ 2017 创建演示程序时，程序创建可能会失败。程序创建失败时，请参阅以下章节：

3.4 在 Visual C++ 2017 创建失败 (MSB8036) 的对策

若在 EPSON RC+7.0 v7.5.0 或更高版本中使用 .NET 程序时，请将 .NET Framework 设置为 v4.5 或以上。

3.1 Visual Basic 使用入门

若要在 Visual Basic .NET 项目中使用 RCAPINet，需定义 Spel 类实例，如下例所示。此时可在您的项目中使用 *g_spel*。

1. 在 Visual Studio .NET 中，选择文件|项目。
2. 创建 Visual Basic 项目作为 Windows 窗体应用。
3. 从项目菜单上选择“添加参考”。
4. 在 NET 组件选项卡中，浏览至 \EpsonRC70\Exe 目录并选择 RCAPINet.dll 文件。
5. 从项目菜单上创建新模块并添加以下代码。

```
Module Module1
    Public WithEvents g_spel As RCAPINet.Spel
    Public Sub InitApp()
        g_spel = New RCAPINet.Spel
        With g_spel
            .Initialize
            .Project = "c:\EpsonRC70\projects\API_Demos\Demol \demol.sprj"
        End With
    End Sub

    Public Sub EventReceived( _
        ByVal sender As Object, _
        ByVal e As RCAPINet.SpelEventArgs) _
        Handles g_spel.EventReceived

        MsgBox("received event " & e.Event)
    End Sub
End Module
```



当应用存在时，需要对每个 Spel 类实例执行 Dispose。这可在主窗体的 FormClosed 事件中完成。如果未执行 Dispose，则应用不会正常关闭。

```
g_spel.Dispose();
```

3.2 Visual C# 使用入门

1. 在 Visual Studio .NET 中，选择文件|项目。
2. 创建 Visual C# 项目作为 Windows 窗体应用。
3. 从项目菜单上选择“添加参考”。
4. 选择浏览选项卡，浏览到 \EpsonRC70\Exe 目录并选择 RCAPINet.dll 文件。
5. 在 Form1 类中，定义 Spel 类变量，如下所示。

```
private RCAPINet.Spel m_spel;
```

6. 在 Form_Load 事件中，添加初始化代码，如下所示。

```
private void Form1_Load(object sender, EventArgs e)
{
    m_spel = new RCAPINet.Spel();
    m_spel.Initialize();

    m_spel.Project =

    "c:\\EpsonRC70\\projects\\API_Demos\\Demo1\\demo1.sprj";

    m_spel.EventReceived += new
        RCAPINet.Spel.EventReceivedEventHandler(m_spel_
            EventReceived);
}
```

7. 添加事件处理程序，如下所示。。

```
public void m_spel_EventReceived(object sender,
    RCAPINet.SpelEventArgs e)
{
}
}
```

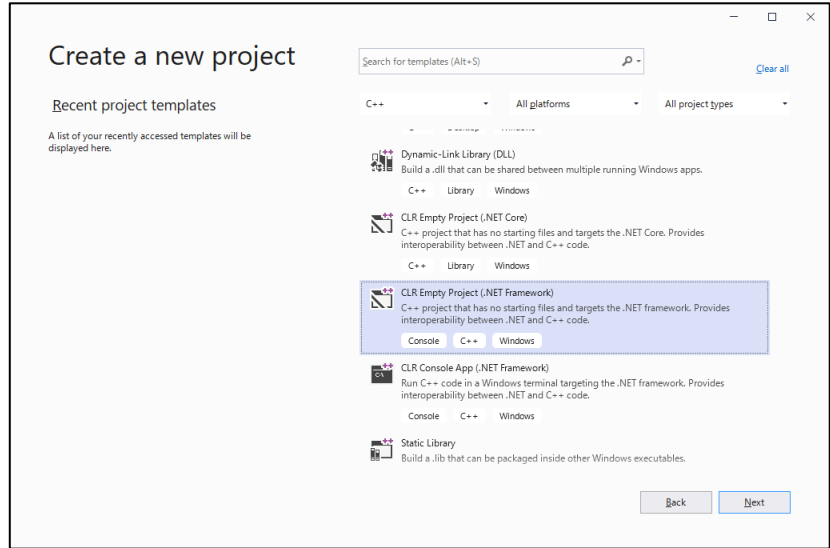


当应用存在时，需要对每个 Spel 类实例执行 Dispose。这可在主窗体的 FormClosed 事件中完成。如果未执行 Dispose，则应用不会正常关闭。

```
m_spel.Dispose();
```

3.3 Visual C++ 使用入门

1. 在 In Visual Studio .NET 中选择[Create a new project]。
2. 选择[Visual C++]-[CLR]-[CLR Empty Project (.NET Framework)]。



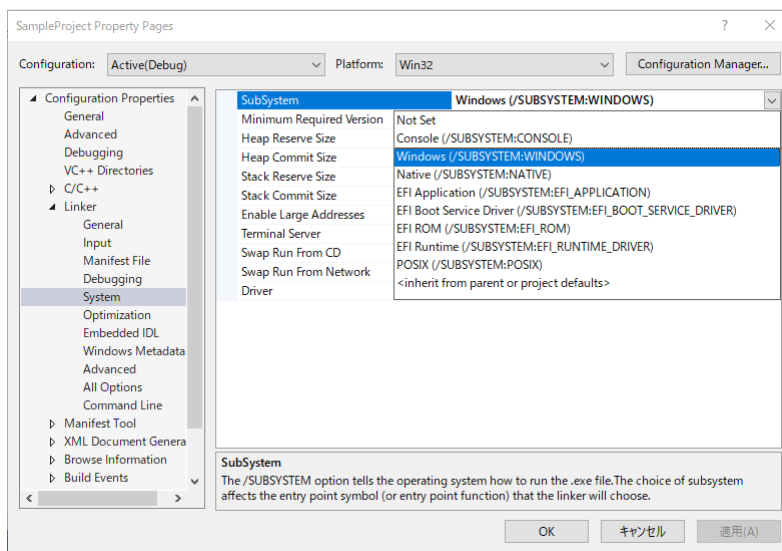
3. 在菜单中选择 [Project]-[Add Reference]。
4. 点击 <Browse> 按钮，浏览到 “/EpsonRC70/Exe”目录，然后选择 “RCAPINet.dll”文件。
5. 在菜单中选择 [Project]-[Add New Item]-[UI]-[Windows Form]。
6. 打开添加的表单中的 cpp 文件(例如: Form1.cpp)，并添加以下源代码。

```
#include "Form1.h"
using namespace SampleProject; ←创建项目的名称
void main() {
    Application::EnableVisualStyles();
    Application::SetCompatibleTextRenderingDefault(false);

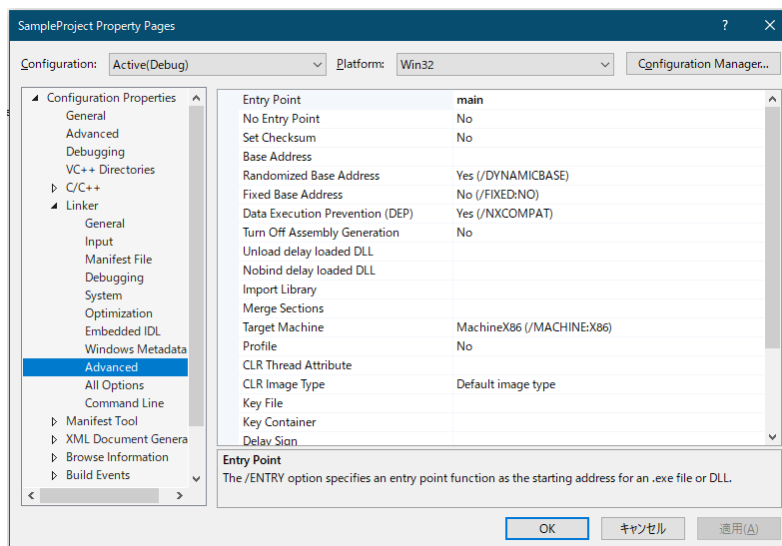
    Form1 frm; ←添加表单的名称
    Application::Run(% frm);
}
```

7. 在菜单中选择 [Project]-[Project Properties]。

8. 在属性页面上选择 [Configuration Properties]-[Linker]-[System]，然后从子系统中选择“Windows (/SUBSYSTEM:WINDOWS)”。



9. 在属性页面上选择 [Configuration Properties]-[Linker]-[Advanced]，然后在“Entry Point”中输入步骤 6 中添加的功能名称。本案例中，输入“main”。



10. 点击<OK>按钮。



设置完成后，执行一次项目创建，检查是否有错误。建议在确认没有错误后，关闭项目，然后重新打开它。

11. 在 Form1 类中，定义 Spel 变量，如下所示。

```
private RCAPINet::Spel^ m_spel;
```

12. 在 `Form_Load` 事件中, 添加初始化代码, 如下所示。

```
private: System::Void Form1_Load(
    System::Object^ sender, System::EventArgs^ e)
{
    m_spel = gcnew RCAPINet::Spel();
    m_spel->Initialize();
    m_spel->Project =
        "c:\\EpsonRC70\\projects\\
API_Demos\\Demo1\\demo1.sprj";
    m_spel->EventReceived += gcnew
        RCAPINet::Spel::EventReceivedEventHandler(
            this, &Form1::m_spel_EventReceived);
}
```

13. 添加事件处理程序, 如下所示。

```
private: System::Void m_spel_EventReceived(
    System::Object^ sender, RCAPINet::SpelEventArgs^ e)
{
    MessageBox::Show(e->Message);
}
```

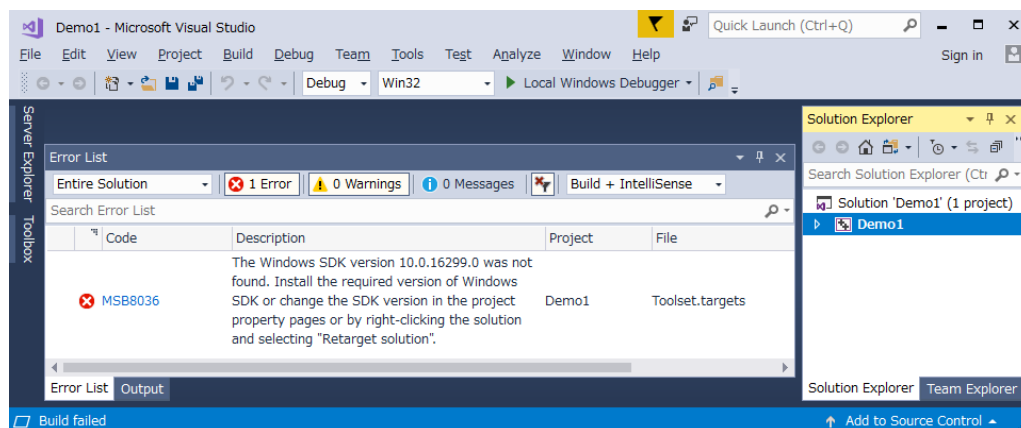


当应用存在时, 如果应用采用堆分配, 则需要删除各 `Spel` 类实例(使用 `gcnew`)。这可在主窗体的 `FormClosed` 事件中完成。如果未删除 `Spel` 类实例, 则应用将不会正常关闭。

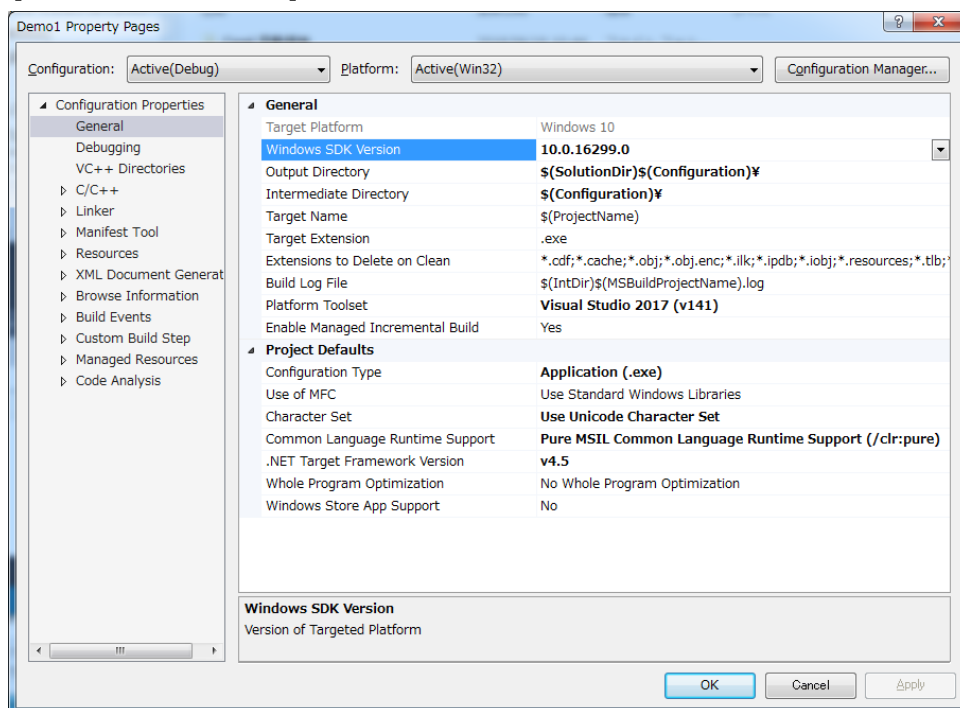
```
delete m_spel;
```


3.4 在 Visual C++ 2017 创建失败 (MSB8036) 的对策

首次在 Visual C++ 2017 创建演示程序时，若程序创建失败的原因是错误：MSB8036，请按照以下步骤解决：

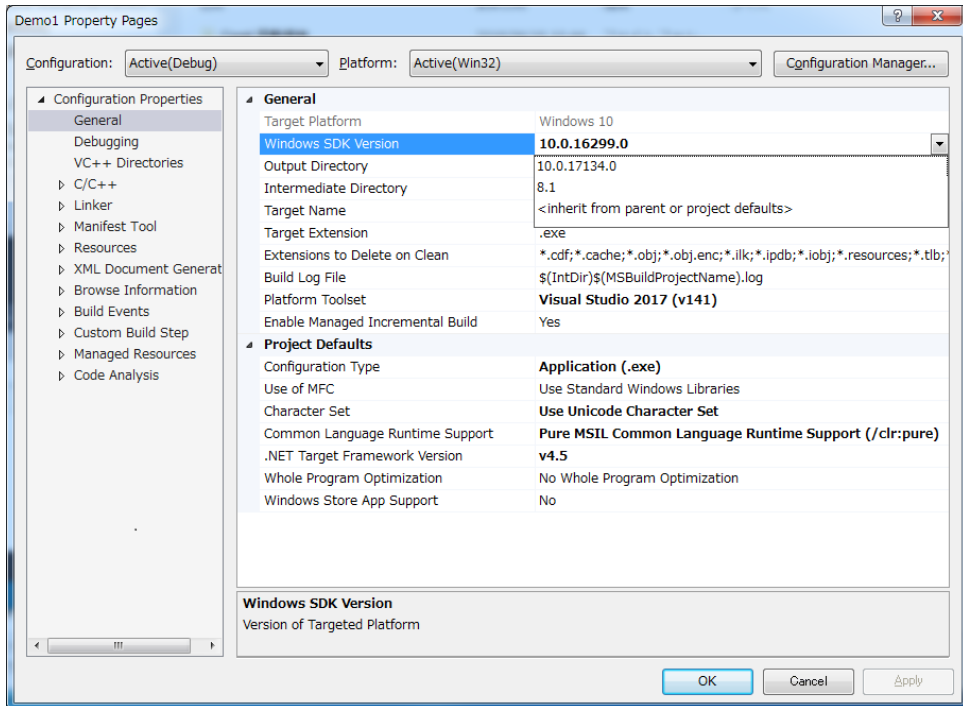


- (1) 选择 Visual Studio C++ 2017-Solution Explorer- “Demo1” 项目。
- (2) 选择菜单-[Project]-[Properties]。
- (3) 在 “Demo1 Property Pages” 上选择[Configuration Properties]-[General]-[Windows SDK Version]。



- (4) 点击 “10.0.16299.0” 右侧的下拉按钮。

(5) 选择开发环境下安装的“Windows SDK Version”。



(6) 点击<OK>按钮。

(7) 重新创建演示程序。

4. 环境

4.1 开发环境

4.1.1 开发启动

通常，您需要执行以下步骤来启动开发：

1. 在 .NET 项目的模块中定义 Spel 类变量。
2. 启动 EPSON RC+ 7.0。
3. 打开所需的 EPSON RC+ 7.0 项目或创建新的 EPSON RC+ 7.0 项目。
4. 构建 EPSON RC+ 7.0 项目。
5. 添加 SPEL 类实例的初始化代码。
6. 运行并调试 .NET 项目。

4.1.2 Spel 类实例初始化

创建新的 Spel 类实例之后，需要对其进行初始化。初始化时，会加载和初始化 EPSON RC+ 7.0 底层模块。通过第一种方法调用或属性访问时，初始化为隐式。您可以通过调用 Initialize 方法对类进行初始化。

```
m_spel.Initialize()
```

4.1.3 Spel 类实例终止

当应用存在时，需要对每个 Spel 类实例执行 Dispose。这可在主窗体的 FormClosed 事件中完成。如果未执行 Dispose，则应用不会正常关闭。

对于 Visual Basic 和 Visual C#，使用 Dispose 方法：

```
m_spel.Dispose()
```

对于 Visual C++，如果采用堆(通过 gnew)创建的 Spel 类实例，则使用 delete：

```
delete m_spel;
```

4.1.4 开发循环

按以下基本步骤编辑和运行 .NET 代码：

1. 停止 .NET 项目。
2. 编辑 .NET 项目。
3. 打开 EPSON RC+ 7.0。
4. 在 EPSON RC+ 7.0 项目中进行更改。
5. 构建 EPSON RC+ 7.0 项目。
6. 关闭 RC+ 7.0。
7. 切换至 Visual Studio。
8. 运行 .NET 项目。

4.2 生产设备

4.2.1 运行时打开 EPSON RC+ 7.0

确定是否允许通过您的应用打开 EPSON RC+ 7.0 环境。这尤其适用于调试。将 `OperationMode` 属性设置为 `Program`，以将 EPSON RC+ 7.0 置于 `Program` 模式并打开 EPSON RC+ 7.0 GUI。

4.2.2 使用 EPSON RC+ 7.0 对话框和窗口

运行时，可在 .NET 应用中打开和隐藏某些 EPSON RC+ 7.0 窗口。还可运行某些 EPSON RC+ 7.0 对话框。

有关详细信息，请参阅 *EPSON RC+ 7.0 窗口和对话框* 章节。

4.2.3 目标系统上的安装

应使用 Visual Studio 安装项目为 .NET 项目编写安装程序。然后按照以下步骤为 .NET 应用安装目标系统：

1. 安装 EPSON RC+ 7.0。
2. 安装 EPSON RC+ 7.0 项目。
3. 安装 .NET 应用。

5. 执行方法、程序和任务

5.1 执行方法

Spel 类中有多种方法。有关可使用方法的介绍，请参阅章节 14.3 *Spel* 类方法。执行一种方法时，将在与控制器通信并执行相关函数的 EPSON RC+ 服务器进程中调用相关内部函数。共有两种类型的方法：立即和异步。对于立即方法：在控制器执行内部函数，并立即返回回复。立即命令包括所有 I/O 命令。对于异步方法，在控制器开始相关函数，然后 Spel 类实例等待从 EPSON RC+ 服务器进程返回表示函数完成的事件。异步方法包括所有机器人动作命令。在等待命令完成过程中，Spel 类实例发送 Windows 事件，使用户 GUI 仍然可响应。例如调用 Go 方法时，机器人正在移动至一点，用户可通过点击按钮使其停止。可以通过将 DisableMsgDispatch 设为 True，在异步方法时禁用 Windows 事件发送。也可以通过将 AsyncMode 设为 True，等待程序中的异步方法完成。

5.1.1 使用多线程

可以在应用中的多线程执行 Spel 方法。以下章节中将介绍各种场景。

在多线程使用一个 Spel 类实例

可以在多线程使用相同 Spel 类实例执行方法，但每次仅可执行一个异步命令。如果试图在一个线程执行异步命令时已经在另一个线程中执行了另一个异步命令，将得到“command in cycle”错误。可以在一个线程中执行立即命令，即使是在另一个线程中执行异步命令期间。

在各线程中使用独立 Spel 类实例

对于各控制器连接，可以有一个或多个 Spel 类实例。各控制器的第一个实例将初始化 EPSON RC+ 7.0 服务器进程，并连接至指定的控制器。若要在其他线程中使用一个或多个另外的实例与控制器连接，必须在 ServerInstance 属性中指定相同值。在使用另外 Spel 类实例前应为第一个实例调用 Initialize。

VB 例:

```
' 为线程 1 初始化 Spel 类实例
m_spel_1 = New Spel
m_spel_1.ServerInstance = 1
m_spel_1.Initialize()
m_spel_1.Project =
"c:\EpsonRC70\Projects\MyProject\MyProject.sprj"
m_spel_1.Connect(1)
```

```
' 为线程 2 初始化 Spel 类实例
' 此线程使用与 m_spel_1 相同的控制器
m_spel_2 = New Spel
m_spel_2.ServerInstance = 1
```

线程 1

```
' 对动作使用实例 m_spel_1
m_spel_1.Robot = 1
Do
    m_spel_1.Go(1)
    m_spel_1.Go(2)
Loop Until m_stop
```

线程 2

```
' 对 I/O 使用实例 m_spel_2

Do
    m_spel_2.On(1)
    m_spel_2.Delay(500)
    m_spel_2.Off(1)
    m_spel_2.Delay(500)
Loop Until m_stop
```

C# 例:

```
// 为线程 1 初始化 Spel 类实例
RCAPINet.Spel m_spel_1 = new RCAPINet.Spel();
m_spel_1.ServerInstance = 1;
m_spel_1.Initialize();
m_spel_1.Project =
@"c:\EpsonRC70\Projects\MyProject\MyProject.sprj";
m_spel_1.Connect(1);
```

```
// 为线程 2 初始化 Spel 类实例
// 此线程使用与 m_spel_1 相同的控制器
RCAPINet.Spel m_spel_2 = new RCAPINet.Spel();
m_spel_2.ServerInstance = 1;
```

线程 1

```
// 对动作使用实例 m_spel_1
m_spel_1.Robot = 1;
do{
    m_spel_1.Go(1);
    m_spel_1.Go(2);
}while(!m_stop);
```

线程 2

```
// 对 I/O 使用实例 m_spel_2
do{
    m_spel_2.On(1);
    m_spel_2.Delay(500);
    m_spel_2.Off(1);
    m_spel_2.Delay(500);
}while(!m_stop);
```

在控制器使用 API 线程

默认为控制器仅支持一个 API 线程。在这种情况下，即使在控制多个机器人时，控制器中异步方法每次执行一个。对于使用一个机器人或使用 SPEL+ 任务执行机器人动作的大部分应用，这已足够，但在例如从同一控制器控制多个机器人等情况下，可以将系统配置为在控制器最多使用 10 个 API 任务，允许 .NET 线程并行处理。

要在控制器中使用多个 API 任务，需要两个基本步骤。

1. 在 EPSON RC+ GUI 中连接控制器，然后打开[Setup]-[System Configuration]-[Controller]-[Preferences]。将“Reserved tasks for API”设为所需的 API 任务数量。请注意，预留的 API 任务越多，SPEL+ 程序中可以使用的任务越少。例如，如果预留 5 个 API 任务，则 SPEL+ 可用 27 个任务(32 - 5)。
2. 在应用中设置 CommandTask 属性，指定要在哪个 API 任务执行方法。

如以下简单示例所示，同一控制器中的每个机器人有一个线程。因为每个线程中使用不同的 `CommandTask`，机器人动作命令将并行执行，对于两个 `Spel` 实例，`ServerInstance` 均设为 1。

VB 例:

' 为线程 1 初始化 Spel 类实例

```
m_spel_1 = New Spel
m_spel_1.ServerInstance = 1
m_spel_1.CommandTask = 1
m_spel_1.Initialize()
m_spel_1.Project =
"c:\EpsonRC70\Projects\MyProject\MyProject.sprj"
m_spel_1.Connect(1)
```

' 为线程 2 初始化 Spel 类实例

' 此线程使用与 `m_spel_1` 相同的控制器

' 使用控制器的第二个 `CommandTask`。

```
m_spel_2 = New Spel
m_spel_2.ServerInstance = 1
m_spel_2.CommandTask = 2
```

线程 1

' 对 Robot 1 动作使用实例 `m_spel_1`

```
m_spel_1.Robot = 1
Do
  m_spel_1.Go(1)
  m_spel_1.Go(2)
Loop Until m_stop
```

线程 2

' 对 Robot 2 动作使用实例 `m_spel_2`

```
m_spel_2.Robot = 2
Do
  m_spel_2.Go(1)
  m_spel_2.Go(2)
Loop Until m_stop
```


C# 例:

```
// 为线程 1 初始化 Spel 类实例
RCAPINet.Spel m_spel_1 = new RCAPINet.Spel();
m_spel_1.ServerInstance = 1;
m_spel_1.CommandTask = 1;
m_spel_1.Initialize();
m_spel_1.Project =
@"c:\EpsonRC70\Projects\MyProject\MyProject.sprj";
m_spel_1.Connect(1);
```

```
// 为线程 2 初始化 Spel 类实例
// 此线程使用与 m_spel_1 相同的控制器
// 使用控制器的第二个 CommandTask。
RCAPINet.Spel m_spel_2 = new RCAPINet.Spel();
m_spel_2.ServerInstance = 1;
m_spel_2.CommandTask = 2;
```

线程 1

```
// 对 Robot 1 动作使用实例 m_spel_1
m_spel_1.Robot = 1;
do{
    m_spel_1.Go(1);
    m_spel_1.Go(2);
}while(!m_stop);
```

线程 2

```
// 对 Robot 2 动作使用实例 m_spel_2
m_spel_2.Robot = 2;
do{
    m_spel_2.Go(1);
    m_spel_2.Go(2);
}while(!m_stop);
```

5.2 执行 SPEL+ 程序

SPEL+ 程序包含一个或多个函数，可通过开始程序的 `main` 函数运行该程序。通过使用 `Spel` 类的 `Start` 方法，可运行当前控制器项目中 64 个内置 `main` 函数中的任意一个。要开始的 `main` 函数必须以 SPEL+ 代码定义。开始一个 `main` 函数时，所有全局变量和模块变量返回默认值。

下表所示为 SPEL+ 项目中的程序编号及其相应的函数名称。

程序编号	SPEL+ 函数名称
0	<code>main</code>
1	<code>main1</code>
2	<code>main2</code>
3	<code>main3</code>
...	...
63	<code>main63</code>

以下为“`main`”函数的开始示例：

VB 例:

```
Sub btnStart_Click( _
    ByVal sender As System.Object, _
    ByVal e As System.EventArgs) _
    Handles btnStart.Click

    m_spel.Start(0) ' 开始执行 main 函数
    btnStart.Enabled = False
    btnStop.Enabled = True
End Sub
```

C# 例::

```
void btnStart_Click(object sender, EventArgs e)
{
    m_spel.Start(0); // 开始执行 main 函数
    btnStart.Enabled = false;
    btnStop.Enabled = true;
}
```

5.3 执行 SPEL+ 任务

通过使用 `Xqt` 方法，可以在 SPEL+ 程序中将函数作为正常任务执行。执行任务时，全局变量不会像使用 `Start` 方法那样返回默认值。

若要暂停和恢复任务，使用 `Halt` 和 `Resume` 方法。

若要退出任务，使用 `Quit` 方法。

也可以使用 `StartBGTask` 方法开始控制器后台任务。

5.4 中止所有任务

如果您正在运行任务，并想立刻中止所有任务，则可以使用 Spel 类的 *Stop* 方法。*Stop* 方法有可选参数，用于另行停止所有后台任务。

VB 例:

```
Sub btnStop_Click( _  
    ByVal sender As System.Object, _  
    ByVal e As System.EventArgs) _  
    Handles btnStop.Click  
    m_spel.Stop()  
    btnStop.Enabled = False  
    btnStart.Enabled = True  
End Sub
```

C# 例:

```
void btnStop_Click(object sender, EventArgs e)  
{  
    m_spel.Stop();  
    btnStop.Enabled = false;  
    btnStart.Enabled = true;  
}
```

6. 事件

6.1 概述

Spel 类支持两种类型的事件：系统事件和用户事件。系统事件即系统状态通知。用户定义的事件通过任一 SPEL+ 任务发送到 .NET 应用。

6.2 系统事件

共有多个发送至 .NET 应用的系统事件。每个系统事件表示一种状态变化。事件用于 Pause、Continue、Emergency Stop 等。有关所有系统事件的完整详细信息，请参见 14.4 *Spel 类事件—EventReceived* 的介绍。

使用 Spel 类 EnableEvents 方法控制发送的系统事件。

6.3 来自 SPEL+ 的用户事件

通过 SPEL+ 程序可在 .NET 应用中导致事件发生。例如：您可以通知 .NET 应用有关连续循环回路的信息。这种方法要优于通过 .NET 在控制器中轮询变量值。

若要通过 SPEL+ 触发 .NET 事件，使用 SPEL+ 程序语句中的 SPELCom_Event 命令。例如：

```
SPELCom_Event 1000, cycNum, lotNum, cycTime
```

SPELCom_Event 命令与 Print 命令类似。可指定一段或多段待发送至 .NET 应用的数据。有关 SPELCom_Event 的详细信息，请参阅 13. *SPELCom_Event*。

在您可以接收事件之前，必须用 WithEvents 语句定义 Spel 类变量。

```
Public WithEvents m_spel As RCAPINet.Spel
```

捕获 Spel 类实例的 EventReceived 例程中的事件。若要编辑此例程，需在定义 Spel 类的模块中，从类名称列表选择 “m_spel” 并从程序列表选择 EventReceived。

以下为发生事件时更新一些标签的 EventReceived 例程中的代码示例。

VB 例:

```
Sub m_spel_EventReceived (ByVal sender As Object, _
    ByVal e As RCAPINet.SpelEventArgs) _
    Handles m_spel.EventReceived
    Dim tokens() As String
    Select Case e.Event
        Case 2000
            tokens = e.Message.Split(New [Char]() {" "c}, _
                System.StringSplitOptions.RemoveEmptyEntries)
            lblCycCount.Text = tokens(0)
            lblLotNumber.Text = tokens(1)
            lblCycTime.Text = tokens(2)
    End Select
End Sub
```

C# 例:

```
void m_spel_EventReceived(object sender, SpelEventArgs e)
{
    string[] tokens = new string[3];
    switch(e.Event)
    {
        case 2000:
            tokens = e.Message.Split(' ');
            lblCycCount.Text = tokens(0);
            lblLotNumber.Text = tokens(1);
            lblCycTime.Text = tokens(2);
            break;
        default:
            break;
    }
}
```

7. 错误处理

7.1 Spel 方法的错误

执行 Spel 类方法时，如果发生任何错误，则会抛出异常。

发生错误时，Spel 类实例会将错误抛至调用例程。此时，应使用应用中的错误处理程序捕获此错误。在某些情况下，仅需显示错误消息。

VB 例:

```
Sub btnStart_Click( _  
    ByVal sender As System.Object, _  
    ByVal e As System.EventArgs) _  
    Handles btnStart.Click  
  
    Try  
        m_spel.Start(0)  
    Catch ex As RCAPINet.SpelException  
        MsgBox(ex.Message)  
    End Try  
End Sub
```

您可使用 SpelException 的 ErrorNumber 属性检查与异常相关的错误编号。

```
Try  
    m_spel.Start(0)  
Catch ex As RCAPINet.SpelException  
    MsgBox("SPEL Error: " + ex.ErrorNumber.ToString())  
End Try
```

C# 例:

```
void btnStart_Click(object sender, EventArgs e)
{
    try{
        m_spel.Start(0);
    }
    catch(SpelException ex){
        MessageBox.Show(ex.Message);
    }
}
```

您可使用 `SpelException` 的 `LineNumber` 属性检查与异常相关的错误编号。

```
try {
    m_spel.Start(0);
}
catch(SpelException ex) {
    MessageBox.Show("SPEL Error: " +
ex.LineNumber.ToString());
}
```

8. 处理暂停与继续

8.1 暂停状态

发生暂停时，控制器与 SPEL+ 任务均会进入暂停状态。

任务正在运行期间发生任一以下情形后，控制器会进入暂停状态：

- 执行了 Spel 类 Pause 方法。
- SPEL+ 任务执行了 Pause。
- 打开了安全防护。

8.2 捕获 Pause 事件

Spel 类将向 .NET 应用发送信号，通知发生暂停的情况。

您可以在 Spel 类的 EventReceived 事件中捕获 Pause 事件。

VB 例:

```
Sub m_spel_EventReceived (ByVal sender As Object, ByVal e
As RCAPINet.SpelEventArgs) Handles m_spel.EventReceived
    Select Case e.Event
        Case RCAPINet.SpelEvents.Pause
            btnPause.Enabled = False
            btnContinue.Enabled = True
    End Select
End Sub
```

C# 例:

```
void m_spel_EventReceived(object sender, SpelEventArgs e)
{
    switch(e.Event)
    {
        case SpelEvents.Pause:
            btnPause.Enabled = false;
            btnContinue.Enabled = true;
            break;
        default:
            break;
    }
}
```


8.3 执行 Pause

以下例程所示为使用 *Pause* 方法通过 Visual Basic 发出 PAUSE 的方式。

VB 例:

```
Sub btnPause_Click( _  
    ByVal sender As System.Object, _  
    ByVal e As System.EventArgs) _  
    Handles btnPause.Click  
  
    m_spel.Pause()  
    btnPause.Enabled = False  
    btnContinue.Enabled = True  
End Sub
```

C# 例:

```
void btnPause_Click(object sender, EventArgs e)  
{  
    m_spel.Pause();  
    btnPause.Enabled = false;  
    btnContinue.Enabled = true;  
}
```

8.4 暂停后继续

若要在发生暂停后继续，可使用 *Continue* 方法。

VB 例:

```
Sub btnContinue_Click( _  
    ByVal sender As System.Object, _  
    ByVal e As System.EventArgs) _  
    Handles btnContinue.Click  
  
    m_spel.Continue()  
    btnContinue.Enabled = False  
    btnPause.Enabled = True  
End Sub
```

C# 例:

```
void btnContinue_Click(object sender, EventArgs e)  
{  
    m_spel.Continue();  
    btnContinue.Enabled = false;  
    btnPause.Enabled = true;  
}
```

8.5 暂停后中止

如果您在暂停后不想继续，还可执行 *Stop* 方法。

VB 例:

```
Sub btnStop_Click( _  
    ByVal sender As System.Object, _  
    ByVal e As System.EventArgs) _  
    Handles btnStop.Click  
  
    m_spel.Stop()  
    btnContinue.Enabled = False  
    btnPause.Enabled = False  
End Sub
```

C# 例:

```
void btnStop_Click(object sender, EventArgs e)  
{  
    m_spel.Stop();  
    btnContinue.Enabled = true;  
    btnPause.Enabled = false;  
}
```

9. 处理紧急停止

发生紧急停止时，您可能会想要在程序中执行一些特定动作，如显示对话框或消息框。

Spel 类会针对紧急停止状态发出两个标准事件：EStopOn 和 EStopOff。

9.1 使用系统 EStop 事件

您可以捕获 Visual Basic 应用中 EventReceived 处理程序的系统 EStop 事件。

```
Imports RCAPINet.Spel

Private Sub m_spel_EventReceived(ByVal sender As Object,
ByVal e As SpelEventArgs) Handles m_spel.EventReceived
    Select Case e.Event
        Case RCAPINet.SpelEvens.EstopOn
            MsgBox "E-Stop detected"
            gEStop = True
            lblEStop.BackColor = Color.Red
            lblEStop.Text = "EStop ON"
        Case RCAPINet.SpelEvents.EstopOff
            gEStop = False
            lblEStop.BackColor = Color.Green
            lblEStop.Text = "EStop OFF"
    End Select
End Sub
```

在 C#程序中，您可以将系统 Estop 事件捕获到“EventReceived”中。

```
private void m_spel_EventReceived(object sender,
SpelEventArgs e)
{
    switch(e.Event)
    {
        case SpelEvents.EstopOn:
            MessageBox.Show("E-Stop detected");
            gEStop = true;
            lblEStop.BackColor = Color.Red;
            lblEStop.Text = "EStop ON";
        case SpelEvents.EstopOff:
            gEStop = false;
            lblEStop.BackColor = Color.Green;
            lblEStop.Text = "EStop OFF";
    }
}
```

10. EPSON RC+ 7.0 窗口和对话框

使用 Spel 类的 ShowWindow 和 RunDialog 方法可打开 .NET 应用中的某些 EPSON RC+ 7.0 窗口和对话框。

10.1 窗口

窗口为非模态，这表示这些窗口可在使用 Visual Basic GUI 的其他元素时保持打开。您可以在 Visual Basic 程序中显示和隐藏 EPSON RC+ 7.0 窗口。

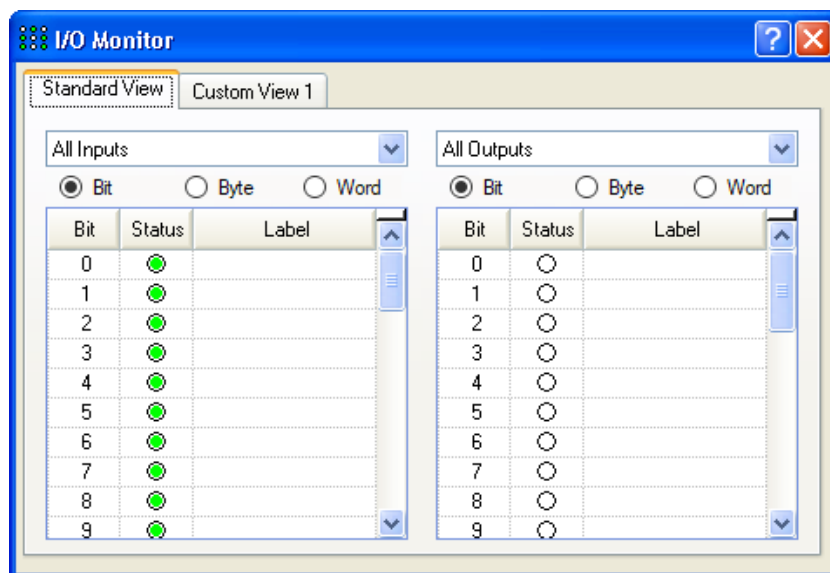
例如，要打开和关闭 I/O Monitor 窗口：

```
m_spel.ShowWindow(RCAPINet.SpelWindows.IOMonitor, Me)
m_spel.HideWindow(RCAPINet.SpelWindows.IOMonitor)
```

使用 C# GUI 的其他元素时，也可以显示 EPSON RC+窗口。

```
m_spel.ShowWindow(RCAPINet.SpelWindows.IOMonitor, this);
m_spel.HideWindow(RCAPINet.SpelWindows.IOMonitor);
```

窗口 ID	窗口
RCAPINet.SpelWindows.IOMonitor	IO Monitor
RCAPINet.SpelWindows.TaskManager	Task Manager
RCAPINet.SpelWindows.ForceMonitor	Force Monitor
RCAPINet.SpelWindows.Simulator	Simulator



I/O Monitor 窗口

10.2 对话框

对话框为模态：一个对话框打开时，将无法使用 .NET GUI 的其他元素，直至关闭该对话框。

例如，要打开机器人管理器对话框：

```
m_spel.RunDialog(RCAPINet.SpelDialogs.RobotManager)
```

一旦对话框打开，则必须由操作员关闭。在程序中无法关闭对话框。这主要是出于安全考虑。

下表所示为可打开的对话框。

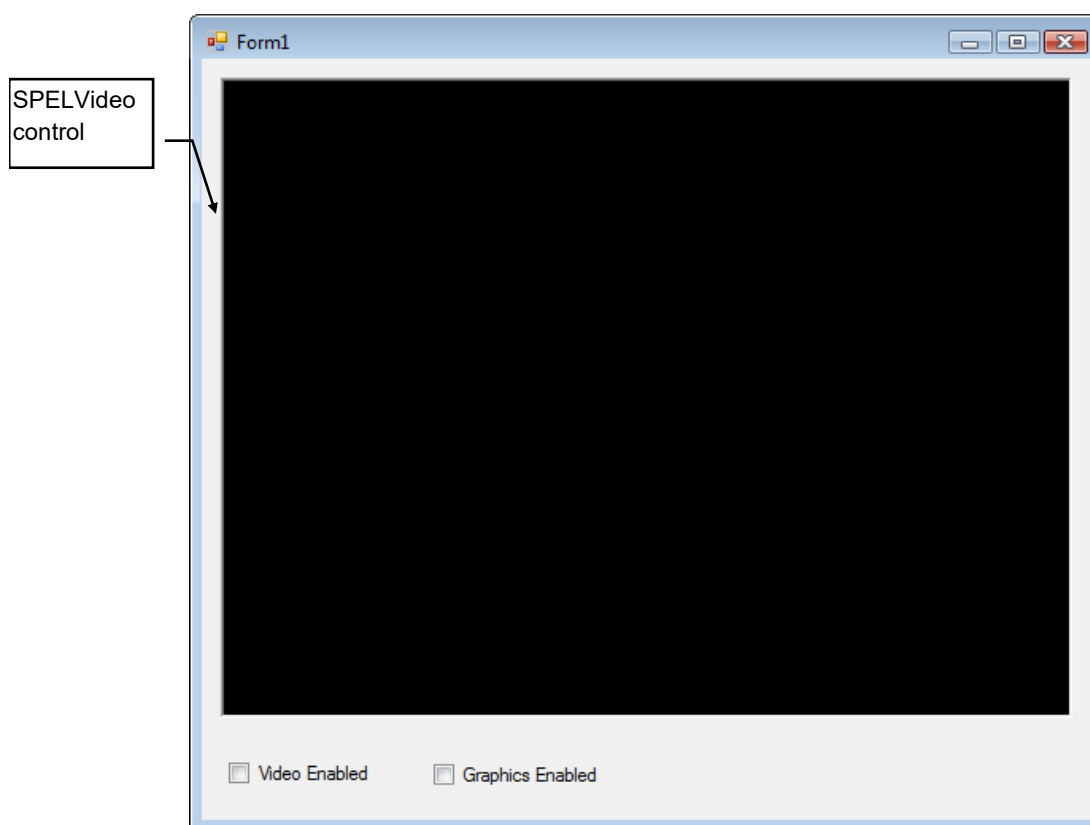
对话框 ID	对话框
RCAPINet.SpelDialogs.RobotManager	Robot Manager
RCAPINet.SpelDialogs.ControllerTools	Controller Tools
RCAPINet.SpelDialogs.VisionGuide	Vision Guide
RCAPINet.SpelDialogs.ForceGuide	Force Guide
RCAPINet.SpelDialogs.PartFeeding	Part Feeding

11. 播放视频

通过使用 SPELVideo 控制，您可以在应用的窗体上播放视频。当您运行视觉顺序时，还可在窗口上显示图形。

执行以下步骤以创建视频显示：

1. 添加 SPELVideo 组件至项目。若要在 Visual Studio .NET 工具箱中添加控制，需右击工具箱并选定“Choose Items”。选择浏览选项卡，浏览到 \EpsonRC70\Exe 目录并选择 RCAPINet.dll 文件。SPELVideo 控制图标将添加至工具箱。
2. 将 SPELVideo 控制放在想要显示视频的窗体上。控制尺寸可更改为全尺寸。
3. 将 VideoEnabled 属性设为 True。
4. 如果您想要显示视觉图形，则将 GraphicsEnabled 属性设为 True。还必须使用 Spel 类 SpelVideoControl 属性将 SPELVideo 控制连接到 Spel 类实例。



放在窗体上的 SPELVideo 控制

当 GraphicsEnabled 属性为 True 且控制连接到 Spel 类实例时，只要在连接至 Spel 类实例的控制器上执行 VRun 方法，就会显示视觉图形。

以下所示为使用 Spel 类实例并放置 SPELVideo 控制的 Visual Basic 窗体上的视频和图形启用方式示例：

```
Private Sub Form_Load(sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    m_spel = New Spel
    m_spel.Initialize()
    m_spel.Project =
"c:\EpsonRC70\projects\test\test.sprj"
    SpelVideo1.VideoEnabled = True
    SpelVideo1.GraphicsEnabled = True
    m_spel.SpelVideoControl = SPELVideo1
End Sub
```

如何在 SPELVideo 控件所在的 C# 表单上，显示视频和图形

```
private void Form_Load(object sender, EventArgs e)
{
    RCAPINet.Spel m_spel = new RCAPINet.Spel();
    m_spel.Initialize();
    m_spel.Project = @"c:\EpsonRC70\projects\test\test.sprj";
    SpelVideo1.VideoEnabled = True;
    SpelVideo1.GraphicsEnabled = True;
    m_spel.SpelVideoControl = SPELVideo1;
}
```

使用多视频显示

启动 EPSON RC+ 7.0 版本 7.3.0 或更高版本，可以在应用中使用多视频显示。可以在每个显示中选择要显示的相机视频。

若要使用多视频显示，必须设置每个显示的 SpelVideoControl 属性。

以下示例所示为包含两个视频显示的初始化。

VB 例:

```
Private Sub Form_Load(sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    m_spel = New Spel
    m_spel.Initialize()
    m_spel.Project =
"c:\EpsonRC70\projects\test\test.sprj"
    SpelVideo1.VideoEnabled = True
    SpelVideo1.GraphicsEnabled = True
    SpelVideo1.Camera = 1
    SpelVideo2.VideoEnabled = True
    SpelVideo2.GraphicsEnabled = True
    SpelVideo2.Camera = 2
    m_spel.SpelVideoControl = SPELVideo1
    m_spel.SpelVideoControl = SPELVideo2
End Sub
```

C# 例:

```
private void Form_Load(object sender, EventArgs e)
{
    RCAPINet.Spel m_spel = new RCAPINet.Spel();
    m_spel.Initialize();
    m_spel.Project =
@"c:\EpsonRC70\project\test\test.sprj";
    SpelVideo1.VideoEnabled = true;
    SpelVideo1.GraphicsEnabled = true;
    SpelVideo1.Camera = 1;
    SpelVideo2.VideoEnabled = true;
    SpelVideo2.GraphicsEnabled = true;
    SpelVideo2.Camera = 2;
    m_spel.SpelVideoControl = SPELVideo1;
    m_spel.SpelVideoControl = SPELVideo2;
}
```


12. 使用 AsyncMode

AsyncMode 允许在执行其他方法时执行 Spel 方法。仅允许异步执行以下 Spel 类方法：

Arc	Jump3
Arc3	Jump3CP
BGo	MCal
BMove	Move
CVMove	Pass
Go	PTran
Home	Pulse
JTran	TGo
Jump	TMove

若要异步执行方法，需将 AsyncMode 属性设为 True，然后执行方法。当 AsyncMode 属性为 true 并执行异步方法时，方法将开始且控制将立即返回至 .NET 应用进行进一步处理。

如果您在之前的方法执行过程中执行另一异步方法，则 SPEL 会等第一个方法完成，然后再开始下一方法并返回至 .NET。

若要等异步方法完成，可使用以下方式之一：

- 执行 WaitCommandComplete 方法。
- 将 AsyncMode 属性设为 False。

若异步命令由于错误无法开始(例如点不存在)，将立即发生异常。但若在命令异步运行期间发生错误，则错误异常将在下次执行异步命令、执行 WaitCommandComplete 或将 AsyncMode 设为 False 时发生。若在下一个命令发生异常，您无法知道哪个语句导致了错误(之前语句或当前语句)。若需要在执行另一个命令前确认异步命令是否已成功完成，则应在下一个命令前调用 WaitCommandComplete。若在之前的异步命令期间发生错误，将发生带有错误编号和消息的 SpelException 异常。请参见以下示例。

VB 例:

```
Try
    m_spel.AsyncMode = True
    m_spel.Go(1)
    ' 此处移动时执行其他操作
    ' 若 Go(1)执行期间有错误, 执行 Go(2)时发生异常
    ' 因此, 无法确定错误事在 Go(1) 还是 Go(2)
    m_spel.Go(2)

    m_spel.Go(3)
    ' 在移动时执行其他操作
    ' 确认 Go(3) 是否正常执行
    m_spel.WaitCommandComplete()
    ' 若 Go(3) 有错误, 将发生异常

    m_spel.Go(4)
Catch ex As SpelException
    ' 处理错误异常

End Try
```

C# 例:

```
try {
    m_spel.AyncMode = true;
    m_spel.Go(1);
    // 此处移动时执行其他操作
    // 若 Go(1)执行期间有错误, 执行 Go(2)时发生异常
    // 因此, 无法确定错误事在 Go(1) 还是 Go(2)
    m_spel.Go(2);

    m_spel.Go(3);
    // 在移动时执行其他操作
    // 确认 Go(3) 是否正常执行
    m_spel.WaitCommandComplete();
    // 若 Go(3) 有错误, 将发生异常

    m_spel.Go(4);
}
catch (RCAPINet.SpelException ex) {
    // 处理错误异常
}
```

13. SPELCom_Event

通过 `Spel` 类实例生成用户事件。

语法

SPELCom_Event *eventNumber* [, *msgArg1*, *msgArg2*, *msgArg3*,...]

参数

eventNumber 数值为 1000-32767 之间的整数表达。
msgArg1, *msgArg2*, *msgArg3*... 可选。各消息参数可为数字、字符串字面量或变量名称。

描述

该指令可轻松地从一个控制器中运行的 `Spel` 任务向应用发送实时信息。例如，可通过发送事件更新工件计数、批号等。

SPELCom_Event 示例

在本例中，SPEL+ 任务利用 RC+ API 向应用发送循环数据。

```
Function RunParts
    Integer cycNum
    String lot$
    Double cycTime

    cycNum = 0
    Do
        TmrReset(0)
        ...
        ...
        cycTime = Tmr(0)
        cycNum = cycNum + 1
        Spelcom_Event 3000, cycNum, lot$, cycTime
        Wait 0.01
    Loop
End
```

14. RCAPINet 参考

14.1 Spel 类

描述

此类允许执行命令并接收 EPSON RC+ 7.0 的事件。

文件名

RCAPINet.dll(64 位和 32 位)

14.2 Spel 类属性

AsyncMode 属性, Spel 类

描述

设置/返回异步执行模式。

语法

Property **AsyncMode** As Boolean

默认值

False

返回值

如果异步模式启用, 则返回 Boolean 值 True, 否则返回 False。

另见

使用 AsyncMode, WaitCommandComplete

AsyncMode 示例

VB 例:

```
With m_spel
    .AsyncMode = True
    .Jump("pick")
    .Delay(500)
    .On(1)
    .WaitCommandComplete()
End With
```

C# 例:

```
m_spel.AsyncMode = true;
m_spel.Jump("pick");
m_spel.Delay(500);
m_spel.On(1);
m_spel.WaitCommandComplete();
```

AvoidSingularity 属性, Spel 类**描述**

设置/返回异点回避模式。

语法

Property **AvoidSingularity** As Boolean

默认值

False

返回值

如果启用异点回避模式, 则返回 Boolean 值 True, 否则返回 False。

另见

Go, Jump, Move

AvoidSingularity 示例**VB 例:**

```
m_spel.AvoidSingularity = True
```

C# 例:

```
m_spel.AvoidSingularity = true;
```

CommandInCycle 属性, Spel 类

描述

返回是否正在执行方法。

语法

ReadOnly Property **CommandInCycle** As Boolean

返回值

如果正在执行方法, 则返回 Boolean 值 True, 否则返回 False。

另见

AsyncMode

CommandInCycle 示例

VB 例:

```
If m_spel.CommandInCycle Then  
    MsgBox "A SPEL command is executing, operation  
    aborted"  
End If
```

C# 例:

```
if (m_spel.CommandInCycle)  
    MessageBox.Show("SPEL command is executing, operation  
    aborted");
```

CommandTask 属性, Spel 类**描述**

为执行机器人命令指定在控制器使用的预留 API 任务。

语法

Property **CommandTask** As Integer

默认值

默认值为 0(不使用预留 API 任务)。

备注

想要在控制器的另一个线程中执行 Spel 机器人命令时, 使用 **CommandTask**。通常 **CommandTask** 在多机器人系统中使用。使用 **CommandTask** 前, 必须首先从 EPSON RC+ 菜单-[Setup]-[System Configuration]-[Controller]-[Preferences]在控制器预留 API 使用的任务。最多可以在控制器预留 16 个 API 任务。

另见

ServerInstance

CommandTask 示例**VB 例:**

```
' Robot1 线程中
m_spel.CommandTask = 1
m_spel.Robot = 1
```

```
' Robot2 线程中
m_spel.CommandTask = 2
m_spel.Robot = 2
```

C# 例:

```
// Robot1 线程中
m_spel.CommandTask = 1;
m_spel.Robot = 1;
```

```
// Robot2 线程中
m_spel.CommandTask = 2;
m_spel.Robot = 2;
```

DisableMsgDispatch 属性, Spel 类

描述

设置/返回是否应在 Spel 方法执行期间处理 Windows 消息。

语法

DisableMsgDispatch

类型

Boolean

默认值

False

备注

通常不得使用该属性。其旨在用于当执行 Spel 方法时不需要键盘或鼠标处理的特殊应用。

ErrorCode 属性, Spel 类**描述**

返回当前控制器错误代码。

语法

ReadOnly Property **ErrorCode** As Integer

返回值

含有错误代码的整数值。

另见

ErrorOn

ErrorCode 示例**VB 例:**

```
If m_spel.ErrorOn Then
    lblErrorCode.Text = m_spel.ErrorCode.ToString()
Else
    lblErrorCode.Text = ""
End If
```

C# 例:

```
if (m_spel.ErrorOn)
    lblErrorCode.Text = m_spel.ErrorCode.ToString();
else
    lblErrorCode.Text = "";
```

ErrorOn 属性, Spel 类

描述

如果控制器内发生重大错误, 则返回 True。

语法

ReadOnly Property **ErrorOn** As Boolean

返回值

如果控制器处于错误状态, 则返回 True, 否则返回 False。

备注

控制器处于错误状态时, ErrorOn 属性返回 True, 此时您可以使用 ErrorCode 属性检索错误代码。

另见

ErrorCode

ErrorOn 示例

VB 例:

```
If m_spel.ErrorOn Then  
    m_spel.Reset  
End If
```

C# 例:

```
if (m_spel.ErrorOn)  
    m_spel.Reset();
```

EStopOn 属性, Spel 类**描述**

返回控制器紧急停止的状态。

语法

ReadOnly Property **EStopOn** As Boolean

返回值

如果紧急停止启动, 则返回 True, 否则返回 False。

EStopOn 示例**VB 例:**

```
If m_spel.EStopOn Then
    lblEStop.Text = "Emergency stop is active"
Else
    lblEStop.Text = ""
EndIf
```

C# 例:

```
if (m_spel.EStopOn)
    lblEStop.Text = "Emergency stop is active";
else
    lblEStop.Text = "";
```

Force_Sensor 属性, Spel 类

描述

设置并返回当前的力传感器编号。

语法

Property **Force_Sensor** As Integer

默认值

1

返回值

当前力传感器编号的整数值

备注

使用任何 **force** 方法之前, 必须使用此属性设置当前的力传感器。

另见

Force_Calibrate, Force_GetForces, Force_SetTrigger

Force_Sensor 示例**VB 例:**

```
' 读取传感器 2 的 z 轴力  
m_spel.Force_Sensor = 2  
f = m_spel.Force_GetForce(3)
```

C# 例:

```
// 读取传感器 2 的 z 轴力  
m_spel.Force_Sensor = 2;  
f = m_spel.Force_GetForce(3);
```

MotorsOn 属性, Spel 类**描述**

设置并返回当前机器人的电机打开或关闭的状态。

语法

Property **MotorsOn** As Boolean

默认值

False

返回值

如果电机打开, 则返回 Boolean 值 True, 否则返回 False。

另见

PowerHigh, Reset, Robot

MotorsOn 示例**VB 例:**

```
If Not m_spel.MotorsOn Then  
    m_spel.MotorsOn = True  
End If
```

C# 例:

```
if (!m_spel.MotorsOn)  
    m_spel.MotorsOn = true;
```

NoProjectSync 属性, Spel 类

描述

设置/返回是否将 PC 中的当前项目与控制器项目同步。

语法

NoProjectSync

类型

Boolean

默认值

False

备注

NoProjectSync 设为 False(默认值)时, Spel 类确保 PC 上的项目与控制器项目同步。

NoProjectSync 设为 True 时, Spel 类不检查 PC 上的任何项目, 不将 PC 项目与控制器同步。这允许您运行控制器程序, 而无需 PC 中的任何项目。

此属性并非永久。若想要将其设为 True, 必须在创建 Spel 类实例后设置。

另见

Start

NoProjectSync 示例**VB 例:**

```
m_spel.Initialize()  
m_spel.NoProjectSync = True
```

C# 例:

```
m_spel.Initialize();  
m_spel.NoProjectSync = true;
```

OperationMode 属性, Spel 类

描述

读取或设置 EPSON RC+ 7.0 的操作模式。

语法

Property **OperationMode** As SpelOperationMode

返回值

SpelOperationMode 值

备注

当 **OperationMode** 设至 Program 时, Spel 类当前实例的 EPSON RC+ 7.0 GUI 会打开且控制器操作模式会设至 Program。如果用户关闭 GUI, **OperationMode** 会设至 Auto。如果从 Visual Basic 中将 **OperationMode** 设至 Auto, 则 GUI 也会关闭。

OperationMode 示例**VB 例:**

```
Sub btnSpelProgramMode_Click _
    ByVal sender As System.Object, _
    ByVal e As System.EventArgs) _
    Handles btnHideIOMonitor.Click

    Try
        m_spel.OperationMode = _
            RCAPINet.SpelOperationMode.Program
        ' 如果想要等用户关闭 RC+ GUI,
        ' 可以在此等待 OperationMode 切换至 Auto
    Do
        Application.DoEvents()
        System.Threading.Thread.Sleep(10)
    Loop Until m_spel.OperationMode = _
        RCAPINet.SpelOperationMode.Auto
    Catch ex As RCAPINet.SpelException
        MsgBox(ex.Message)
    End Try
End If
```

C# 例:

```
void btnSpelProgramMode_Click(object sender, EventArgs e)
{
    try {
        m_spel.OperationMode =RCAPINet.SpelOperationMode.Auto;
        //如果要等待 RC+ GUI 关闭时，可在此待机等待
        Do {
            Application.DoEvents();
            System.Threading.Thread.Sleep(10);
        } while(!m_spel.OperationMode =
RCAPINet.OperationMode.Auto);
    }
    Catch (SpelException ex){
        MessageBox.Show(ex.Message);
    }
}
```


ParentWindowHandle 属性, Spel 类**描述**

设置/返回用于对话框和窗口的父窗口句柄。

语法

Property **ParentWindowHandle** As Integer

返回值

含有窗口句柄的整数值。

备注

使用 **ParentWindowHandle** 在无 .NET 窗体的应用中指定父窗口, 如 LabVIEW。

另见

ShowWindow

ParentWindowHandle 示例**VB 例:**

```
m_spel.ParentWindowHandle = Me.Handle  
m_spel.ShowWindow(RCAPINet.SpelWindows.IOMonitor)
```

C# 例:

```
m_spel.ParentWindowHandle = (int)this.Handle;  
m_spel.ShowWindow(RCAPINet.SpelWindows.IOMonitor);
```

PauseOn 属性, Spel 类

描述

返回控制器暂停状态的状态。

语法

ReadOnly Property **PauseOn** As Boolean

返回值

如果控制器处于暂停状态，则返回 True，否则返回 False。

另见

Continue Pause

PauseOn 示例

VB 例:

```
If m_spel.PauseOn Then  
    btnPause.Enabled = False  
    btnContinue.Enabled = True  
End If
```

C# 例:

```
if (m_spel.PauseOn) {  
    btnPause.Enabled = false;  
    btnContinue.Enabled = true;  
}
```

PowerHigh 属性, Spel 类**描述**

设置并返回当前机器人的功率模式。

语法

Property **PowerHigh** As Boolean

默认值

False

返回值

如果当前机器人的功率模式为 **High**, 则返回 **True**, 否则返回 **False**。

另见

MotorsOn

PowerHigh 示例**VB 例:**

```
If Not m_spel.PowerHigh Then  
    m_spel.PowerHigh = True  
End If
```

C# 例:

```
if (!m_spel.PowerHigh)  
    m_spel.PowerHigh = true;
```

Project 属性, Spel 类

描述

设置/返回当前项目。

语法

Property **Project** As String

默认值

空字符串。

返回值

含有项目路径与文件的字符串。

备注

设置 **Project** 时, 必须提供 EPSON RC+ 7.0 项目制作文件的完整路径与名称。制作文件即带 .SPRJ 扩展名的项目名称。

Project 示例

VB 例:

```
m_spel.Project = "c:\EpsonRC70\projects\myapp\myapp.sprj"
```

C# 例:

```
m_spel.Project = @"c:\EpsonRC70\projects\myapp\myapp.sprj";
```

ProjectBuildComplete 属性, Spel 类**描述**

返回当前项目创建的状态。

语法

ReadOnly Property **ProjectBuildComplete** As Boolean

返回值

如果项目创建已完成, 则返回 **True**, 否则返回 **False**。

另见

BuildProject

ProjectBuildComplete 示例**VB 例:**

```
If m_spel.ProjectBuildComplete Then
    lblBuild.Text = "Project build is Complete"
Else
    lblBuild.Text = "Project build is not Complete"
End If
```

C# 例:

```
if (m_spel.ProjectBuildComplete)
    lblBuild.Text = "Project build is Complete";
else
    lblBuild.Text = "Project build is not Complete";
```

ProjectOverwriteWarningEnabled 属性, Spel 类

描述

设置/返回是否显示项目覆盖的错误信息。

语法

Property **ProjectOverwriteWarningEnabled** As Boolean

默认值

True

返回值

如果显示项目覆盖的错误信息, 则返回 True, 否则返回 False。

另见

BuildProject

备注

默认情况下, 当当前项目与控制器中的项目不同时, 建制程序或向控制器发送信息时, 会显示项目覆盖的错误消息。如果您不需要显示错误消息, 请将 ProjectOverwriteWarningEnabled 设置为 False。此功能多用于当您需要切换控制器的运行项目的应用。

ProjectOverwriteWarningEnabled 示例**VB 例:**

```
' 禁用项目覆盖警告  
m_spel.ProjectOverwriteWarningEnabled = False  
m_spel.Project =  
"c:\EpsonRC70\Projects\Project1\Project1.sprj"
```

C# 例:

```
// 禁用项目覆盖警告  
m_spel.ProjectOverwriteWarningEnabled = false;  
m_spel.Project =  
@"c:\EpsonRC70\Projects\Project1\Project1.sprj";
```

ResetAbortEnabled 属性, Spel 类**描述**

设置/返回是否应启用 ResetAbort 方法。

语法

Property **ResetAbortEnabled** As Boolean

默认值

True

返回值

如果 ResetAbort 已启用, 则返回 True, 否则返回 False。

另见

ResetAbort

ResetAbortEnabled 示例**VB 例:**

```
' 启用 ResetAbort  
m_spel.ResetAbortEnabled = True
```

C# 例:

```
// 启用 ResetAbort  
m_spel.ResetAbortEnabled = true;
```

Robot 属性, Spel 类

描述

设置/返回当前机器人编号。

语法

Property **Robot** As Integer

默认值

如果存在一个或多个机器人，则第一个 Spel 实例的默认值为 1，否则为 0。对于所有其他 Spel 实例，默认值为 0。

返回值

含有当前机器人编号的整数值。

备注

对于使用多个机器人的系统，使用 **Robot** 属性设置机器人执行随后的机器人相关命令，如动作命令。

另见

RobotModel, RobotType

Robot 示例**VB 例:**

```
m_spel.Robot = 2
If Not m_spel.MotorsOn Then
    m_spel.MotorsOn = True
End If
```

C# 例:

```
m_spel.Robot = 2;
if (!m_spel.MotorsOn)
    m_spel.MotorsOn = true;
```


RobotModel 属性, Spel 类

描述

返回当前机器人的型号名称。

语法

ReadOnly Property **RobotModel** As String

返回值

含有当前机器人型号名称的字符串。

另见

Robot, RobotType

RobotModel 示例**VB 例:**

```
lblRobotModel.Text = m_spel.RobotModel
```

C# 例:

```
lblRobotModel.Text = m_spel.RobotModel;
```

RobotType 属性, Spel 类

描述

返回当前机器人的类型。

语法

ReadOnly Property **RobotType** As SpelRobotType

返回值

SpelRobotType 值

另见

Robot, RobotModel

RobotType 示例**VB 例:**

```
Select Case m_spel.RobotType
    Case RCAPINet.SpelRobotType.Scara
        lblRobotType.Text = "Scara"
    Case RCAPINet.SpelRobotType.Cartesian
        lblRobotType.Text = "Cartesian"
End Select
```

C# 例:

```
switch (m_spel.RobotType)
{
    case SpelRobotType.Scara:
        lblRobotType.Text = "Scara";
        break;
    case SpelRobotType.Cartesian:
        lblRobotType.Text = "Cartesian";
        break;
    default:
        break;
}
```

SafetyOn 属性, Spel 类**描述**

返回控制器安全防护输入的状态。

语法

ReadOnly Property **SafetyOn** As Boolean

返回值

如果安全防护打开, 则返回 True, 否则返回 False。

备注

应用启动时, 使用 SafetyOn 属性获取安全防护状态, 然后使用 SafeguardOpen 和 SafeguardClose 事件更新状态。

SafetyOn 示例**VB 例:**

```
If m_spel.SafetyOn Then  
    lblSafeguard.Text = "Safe guard is active"  
Else  
    lblSafeguard.Text = ""  
End If
```

C# 例:

```
if (m_spel.SafetyOn)  
    lblSafeguard.Text = "Safe guard is active";  
else  
    lblSafeguard.Text = "";
```

ServerInstance 属性, Spel 类

描述

指定要使用的 EPSON RC+ 服务器实例。

语法

Property **ServerInstance** As Integer

默认值

默认值为下一个可用的服务器实例。

备注

API 与 RC+服务器进程通信。**ServerInstance** 指定使用哪个服务器。每个服务器实例对应一个控制器和一个项目。默认情况下, 当您创建新的 **Spel** 类实例时, **ServerInstance** 会自动设置为“1”。

如果您需要为同一个控制器使用多个 **Spel** 类实例 (例如多线程的应用程序), 请为使用同一控制器的每个 **Spel** 类实例设置 **ServerInstance** 属性。

ServerInstance 必须介于 1 到 10 之间, 并且需要预先进行设置, 然后才能对其进行初始化或运行其他方法。

另见

CommandTask, Initialize

ServerInstance 示例**VB 例:**

```
' 控制器 1
spel1 = New Spel
spel1.ServerInstance = 1
spel1.Initialize()
spel1.Connect(1)

' 控制器 2
spel2 = New Spel
spel2.ServerInstance = 2
spel2.Initialize()
spel2.Connect(2)
```

C# 例:

```
// 控制器 1
RCAPINet.Spel spel1 = new RCAPINet.Spel();
spel1.ServerInstance = 1;
spel1.Initialize();
spel1.Connect(1);

// 控制器 2
RCAPINet.Spel spel2 = new RCAPINet.Spel();
spel2.ServerInstance = 2;
spel2.Initialize();
spel2.Connect(2);
```

SPELVideoControl 属性, Spel 类**描述**

用于将 SPELVideo 控制连接至 Spel 类实例, 以显示视频和图形。

语法

Property **SpelVideoControl** As SpelVideo

另见

Graphics Enabled, VideoEnabled, Camera

SpelVideoControl 示例**VB 例:**

```
m_spel.SpelVideoControl = SpelVideo1
```

C# 例:

```
m_spel.SpelVideoControl = SpelVideo1;
```

Version 属性, Spel 类

描述

返回当前的 EPSON RC+ 7.0 软件版本。

语法

ReadOnly Property **Version** As String

返回值

含有当前 EPSON RC+ 7.0 软件版本的字符串。

Version 示例

VB 例:

```
'获取软件版本  
curVer = m_spel.Version
```

C# 例:

```
//获取软件版本  
curVer = m_spel.Version;
```

WarningCode 属性, Spel 类**描述**

返回控制器警告代码。

语法

ReadOnly Property **WarningCode** As Integer

返回值

含有当前控制器警告代码的整数值。

另见

WarningOn

WarningCode 示例**VB 例:**

```
If m_spel.WarningOn Then
    lblWarningCode.Text = m_spel.WarningCode.ToString()
Else
    lblWarningCode.Text = ""
End If
```

C# 例:

```
if (m_spel.WarningOn)
    lblWarningCode.Text = m_spel.WarningCode.ToString();
else
    lblWarningCode.Text = "";
```

WarningOn 属性, Spel 类

描述

返回控制器警告状态的状态。

语法

ReadOnly Property **WarningOn** As Boolean

返回值

如果控制器处于警告状态, 则返回 True, 否则返回 False。

另见

WarningCode

WarningOn 示例**VB 例:**

```
If m_spel.WarningOn Then
    lblWarningStatus.Text = "ON"
Else
    lblWarningStatus.Text = "OFF"
End If
```

C# 例:

```
if (m_spel.WarningOn)
    lblWarningStatus.Text = "ON";
else
    lblWarningStatus.Text = "OFF";
```


14.3 Spel 类方法

Accel 方法, Spel 类

描述

设置 PTP 动作命令 Go、Jump 和 Pulse 的加速度和减速度。

语法

Sub **Accel** (*PointToPointAccel* As Integer, *PointToPointDecel* As Integer, _
 [*JumpDepartAccel* As Integer], [*JumpDepartDecel* As Integer], _
 [*JumpApproAccel* As Integer], [*JumpApproDecel* As Integer])

参数

<i>PointToPointAccel</i>	1-100 之间表示最大加速率百分比的整数表达式。
<i>PointToPointDecel</i>	1-100 之间表示最大减速率百分比的整数表达式。
<i>JumpDepartAccel</i>	1-100 之间表示 Jump 命令 Z 轴向上动作的最大加速率百分比的整数表达式。
<i>JumpDepartDecel</i>	1-100 之间表示 Jump 命令 Z 轴向上动作的最大减速率百分比的整数表达式。
<i>JumpApproAccel</i>	1-100 之间表示 Jump 命令 Z 轴向下动作的最大加速率百分比的整数表达式。
<i>JumpApproDecel</i>	1-100 之间表示 Jump 命令 Z 轴向下动作的最大减速率百分比的整数表达式。

另见

Accels, Speed

Accel 示例

VB 例:

```
m_spel.Accel (50, 50)
m_spel.Go ("pick")
```

C# 例:

```
m_spel.Accel (50, 50);
m_spel.Go ("pick");
```

AccelR 方法, Spel 类

描述

设置工具旋转动作的加速度和减速度。

语法

Sub **AccelR** (*Accel* As Single, [*Decel* As Single])

参数

<i>Accel</i>	0.1 至 5000 deg/sec ² 之间的单一表达式, 用以在动作命令中使用 ROT 时定义工具旋转加速度。如果省略了 <i>Decel</i> , 则此值会用于加速率和减速率。
<i>Decel</i>	可选。0.1 至 5000 deg/sec ² 之间的单一表达式, 用以在动作命令中使用 ROT 时定义工具旋转减速度。

另见

Arc, Arc3, BMove, Jump3CP, Power, SpeedR, TMove

AccelR 示例**VB 例:**

```
Sub MoveToPlace ()
    m_spel.AccelR(100)
    m_spel.Move("place ROT")
End Sub
```

C# 例:

```
void MoveToPlace ()
{
    m_spel.AccelR(100);
    m_spel.Move("place ROT");
}
```

AccelS 方法, Spel 类

描述

设置直线运动和 CP 运动 (Jump3CP、Move、TMove) 的加速度和减速度。

语法

```
Sub AccelS ( Accel As Single, Decel As Single,  
            [JumpDepartAccel As Single], [JumpDepartDecel As Single],_  
            [JumpApproAccel As Single], [JumpApproDecel As Single] )
```

参数

<i>Accel</i>	1-5000 之间以 mm/sec ² 为单位的单一表达式, 用以定义直线和连续路径动作的加速度和减速度值。如果省略了 Decel , 则此值会用于加速率和减速率。
<i>Decel</i>	1-5000 之间以 mm/sec ² 为单位的单一表达式, 用以定义直线和连续路径动作的减速度值。一个用于表示加速率和减速率的参数。
<i>JumpDepartAccel</i>	1-5000 之间表示 Jump3CP 命令 Z 轴向上动作的最大加速率百分比的单一表达式。
<i>JumpDepartDecel</i>	1-5000 之间表示 Jump3CP 命令 Z 轴向上动作的最大减速率百分比的单一表达式。
<i>JumpApproAccel</i>	1-5000 之间表示 Jump3CP 命令 Z 轴向下动作的最大加速率百分比的单一表达式。
<i>JumpApproDecel</i>	1-5000 之间表示 Jump3CP 命令 Z 轴向下动作的最大减速率百分比的单一表达式。

另见

Accel, SpeedS, Jump3CP, Move, TMove

AccelS 示例**VB 例:**

```
Sub MoveToPlace ()  
    m_spel.AccelS(500)  
    m_spel.Move(pick)  
    m_spel.AccelS(500, 300)  
    m_spel.Move(place)  
End Sub
```

C# 例:

```
void MoveToPlace ()  
{  
    m_spel.AccelS(500);  
    m_spel.Move(pick);  
    m_spel.AccelS(500, 300);  
    m_spel.Move(place);  
}
```

Agl 方法, Spel 类

描述

返回选定旋转轴的关节角度, 或选定线性轴的位置。

语法

Function **Agl** (*JointNumber* As Integer) As Single

参数

JointNumber 1-9 之间表示关节编号的整数表达式。

另见

Pls, CX - CT

Agl 示例

VB 例:

```
Dim j1Angle As Single  
j1Angle = m_spel.Agl(1)
```

C# 例:

```
float j1Angle;  
j1Angle = m_spel.Agl(1);
```

AIO_In 方法, Spel 类

描述

从模拟 I/O 选件的输入通道，读取模拟值。

语法

Function **AIO_In** (*Channel* As Integer) As Single

参数

Channel 指定模拟 I/O 的通道编号。

返回值

返回一个实数，该数值是指定编号的模拟 I/O 输入通道的模拟输入值。返回值的范围由模拟 I/O 板卡的输入范围设置。

另见

AIO_InW, AIO_Out, AIO_OutW

AIO_In 示例**VB 例:**

```
Dim val As Single  
val = m_spel.AIO_In(2)
```

C# 例:

```
float val;  
val = m_spel.AIO_In(2);
```

AIO_InW 方法, Spel 类

描述

从模拟 I/O 选件的输入通道，读取模拟值。

语法

Function **AIO_InW** (*Channel As Integer*) As Integer

参数

Channel 指定模拟 I/O 的通道编号。

返回值

返回指定的模拟 I/O 通道的输入状态(0~65535 的整数)。

每个输入通道的输入电压(电流)与返回值，因模拟 I/O 板的输入范围设置而异，其对应关系如下所示。

输入数据		设置输入范围				
16 进制	10 进制	±10.24(V)	±5.12(V)	0-5.12(V)	0-10.24(V)	0-24(mA)
0xFFFF	65535	10.23969	5.11984	5.12000	10.24000	24.00000
0x8001	32769	0.00031	0.00016	2.56008	5.12016	12.00037
0x8000	32768	0.00000	0.00000	2.56000	5.12000	12.00000
0x0000	0	-10.24000	-5.12000	0.00000	0.00000	0.00000

另见

AIO_In, AIO_Out, AIO_OutW

AIO_InW 示例**VB 例:**

```
Dim val As Integer
val = m_spel.AIO_InW(2)
```

C# 例:

```
int val;
val = m_spel.AIO_InW(2);
```

AIO_Out 方法, Spel 类

描述

从模拟 I/O 选件的输出通道，读取或设置模拟量值。

语法

```
Function AIO_Out (Channel As Integer) As Single
Sub AIO_Out (Channel As Integer, Value As Single)
```

参数

Channel 指定模拟 I/O 通道的编号。

Value 一个表达式或数值，用于显示要输出的电压[V]或电流[mA]的实数。

返回值

返回指定的模拟 I/O 通道的电压和电流输出状态的实数值。电压输出时的单位为[V]，电流输出时的单位为[mA]。

Function **AIO_Out** (*Channel* As Integer) As Single 时：

当在指定的通道中输出机器人的速度信息时，也可以通过本方法返回值。

备注

将表示指定电压[V]或电流[mA]的实数值，输出到通道编号指定的模拟输出端口。请通过端口上的开关，设置模拟输出端口的电压输出范围和选择电压、电流输出。如指定的值超出模拟 I/O 板的输出范围时，则会输出范围内的边界值(最大值或最小值)。

Sub **AIO_Out** (*Channel* As Integer, *Value* As Single)时：

当在指定的通道中输出机器人信息速度时，本方法会发生输出设定错误。请在停止输出速度信息后，执行本方法。

另见

AIO_In, AOI_InW, AIO_OutW

AIO_Out 示例**VB 例:**

```
Dim val As Single
val = m_spel.AIO_Out(1)
```

C# 例:

```
float val;
val = m_spel.AIO_Out(1);
```

AIO_OutW 方法, Spel 类

描述

从模拟 I/O 选件的输出通道，读取或设置模拟量值。

语法

Function **AIO_OutW** (*Channel As Integer*) As Integer

Sub **AIO_OutW** (*Channel As Integer, OutputData As Integer*)

参数

Channel 指定模拟 I/O 通道的编号。

OutputData 将输出数据 (0 到 65535 的整数)，指定为表达式或数值。

返回值

返回指定的模拟 I/O 通道的输出状态(0~65535 的整数)。

每个输出通道的输出电压(电流)与返回值，因模拟 I/O 板的输出范围设置而异，其对应关系如下所示。

输出数据		设置输出范围					
16 进制	10 进制	±10(V)	±5(V)	0-5(V)	0-10(V)	4-20(mA)	0-20(mA)
0xFFFF	65535	9.99970	4.99985	5.00000	10.00000	20.00000	20.00000
0x8001	32769	0.00031	0.00015	2.50008	5.00015	12.00024	10.00031
0x8000	32768	0.00000	0.00000	2.50000	5.00000	12.00000	10.00000
0x0000	0	-10.00000	-5.00000	0.00000	0.00000	4.00000	0.00000

当在指定的通道中输出机器人的速度信息时，也可以执行本方法。

另见

AIO_In, AOI_InW, AIO_Out

AIO_OutW 示例**VB 例:**

```
Dim val As Integer
val = m_spel.AIO_OutW(1)
```

C# 例:

```
int val;
val = m_spel.AIO_OutW(1);
```


Arc 方法, Spel 类

描述

Arc 可利用 XY 平面内的圆弧插补将机械臂移至指定点处。

语法

```
Sub Arc (MidPoint As Integer, EndPoint As Integer)
Sub Arc (MidPoint As SpelPoint, EndPoint As SpelPoint)
Sub Arc (MidPoint As String, EndPoint As String)
```

参数

每个语法均具有两个指定弧中点和端点的参数。

MidPoint 使用整数、SpelPoint 或字符串表达式指定中点。

EndPoint 使用整数、SpelPoint 或字符串表达式指定端点。使用字符串表达式时，内容可包括 ROT、CP、SYNC、Till 搜索表达式以及并行处理语句。

另见

AccelR, AccelS, SpeedR, SpeedS
Arc3, CVMove, Go, Jump, Jump3, Jump3CP, Move
BGo, BMove, TGo, TMove
CP, Till

Arc 示例**VB 例:**

```
' 使用 SpelPoint 指定点
Dim midPoint, endPoint As SpelPoint
midPoint = m_spel.GetPoint("P1")
endPoint = m_spel.GetPoint("P2")
m_spel.Arc(midPoint, endPoint)

' 使用字符串表达式指定点
m_spel.Arc("P1", "P2")
m_spel.Arc("P1", "P2 CP")

' 使用并行处理
m_spel.Arc("P1", "P2 !D50; On 1; D90; Off 1!")
```

C# 例:

```
// 使用 SpelPoint 指定点
SpelPoint midPoint, endPoint;
midPoint = m_spel.GetPoint("P1");
endPoint = m_spel.GetPoint("P2");
m_spel.Arc(midPoint, endPoint);

// 使用字符串表达式指定点
m_spel.Arc("P1", "P2");
m_spel.Arc("P1", "P2 CP");

// 使用并行处理
m_spel.Arc("P1", "P2 !D50; On 1; D90; Off 1!");
```

Arc3 方法, Spel 类

描述

Arc3 可利用 3 维空间的圆弧插补将机械臂移至指定点。

语法

```
Sub Arc3 (MidPoint As Integer, EndPoint As Integer)
Sub Arc3 (MidPoint As SpelPoint, EndPoint As SpelPoint)
Sub Arc3(MidPoint As String, EndPoint As String)
```

参数

每个语法均具有两个指定弧中点和端点的参数。

MidPoint 使用整数、SpelPoint 或字符串表达式指定中点。

EndPoint 使用整数、SpelPoint 或字符串表达式指定端点。使用字符串表达式时，内容可包括 ROT、ECP、CP、SYNC、Till 搜索表达式以及并行处理语句。

另见

AccelR, AccelS, SpeedR, SpeedS
Arc, CVMove, Go, Jump, Jump3, Jump3CP, Move
BGo, BMove, TGo, TMove
CP, Till

Arc3 示例**VB 例:**

```
' 使用 SpelPoint 指定点
Dim midPoint, endPoint As SpelPoint
midPoint = m_spel.GetPoint("P1")
endPoint = m_spel.GetPoint("P2")
m_spel.Arc3(midPoint, endPoint)

' 使用表达式指定点
m_spel.Arc3("P1", "P2")
m_spel.Arc3("P1", "P2 CP")

' 使用并行处理
m_spel.Arc3("P1", "P2 !D50; On 1; D90; Off 1!")
```

C# 例:

```
// 使用 SpelPoint 指定点
SpelPoint midPoint, endPoint;
midPoint = m_spel.GetPoint("P1");
endPoint = m_spel.GetPoint("P2");
m_spel.Arc3(midPoint, endPoint);

// 使用字符串表达式指定点
m_spel.Arc3("P1", "P2");
m_spel.Arc3("P1", "P2 CP");

// 使用并行处理
m_spel.Arc3("P1", "P2 !D50; On 1; D90; Off 1!");
```

Arch 方法, Spel 类

描述

定义与 JUMP 指令一同使用的 ARCH 参数(开始水平动作前需移动的 Z 高度)。

语法

Sub **Arch** (*ArchNumber* As Integer, *DepartDist* As Integer, *ApproDist* As Integer)

参数

ArchNumber 待定义的 Arch 编号。有效 Arch 编号为 Arch 表中的 7 个条目(0-6)。

DepartDist 启动水平动作之前, Jump 指令开始时移动的起始距离, 单位为毫米。

ApproDist Jump 指令目标位置上方的结束距离, 单位为毫米。

另见

Jump, Jump3, Jump3CP

Arch 示例**VB 例:**

```
Sub SetArchs()
    With m_spel
        .Arch(1, 30, 30)
        .Arch(2, 60, 60)
        .Jump("P1 C1")
        .Jump("P2 C2")
    End With
End Sub
```

C# 例:

```
void SetArchs()
{
    m_spel.Arch(1, 30, 30);
    m_spel.Arch(2, 60, 60);
    m_spel.Jump("P1 C1");
    m_spel.Jump("P2 C2");
}
```

Arm 方法, Spel 类

描述

选择当前的机器人机械臂。

语法

Sub **Arm** (*ArmNumber* As Integer)

参数

ArmNumber 0-15 之间的整数表达式。用户最多可选择 16 个不同的机械臂。机械臂 0 是标准(默认)机器人机械臂。机械臂 1-15 是 ArmSet 指令定义的辅助机械臂。

另见

ArmSet, GetArm, Tool

Arm 示例

VB 例:

```
m_spel.Arm(1)
```

C# 例:

```
m_spel.Arm(1);
```

ArmClr 方法, Spel 类

描述

清除(取消定义)当前机器人的机械臂。

语法

```
Sub ArmClr (ArmNumber As Integer)
```

参数

ArmNumber 1-15 之间的整数表达式。机械臂 0 是标准(默认)机器人机械臂, 无法清除。机械臂 1-15 是 **ArmSet** 指令定义的辅助机械臂。

另见

ArmSet, **GetArm**, **Tool**

ArmClr 示例**VB 例:**

```
m_spel.ArmClr(1)
```

C# 例:

```
m_spel.ArmClr(1);
```

ArmDef 方法, Spel 类

描述

返回是否定义了机器人机械臂。

语法

Function **ArmDef** (*ArmNumber* As Integer) As Boolean

参数

ArmNumber 1-15 之间的整数表达式。机械臂 0 是标准(默认)机器人机械臂, 始终已定义。机械臂 1-15 是使用 **ArmSet** 方法定义的辅助机械臂。

返回值

如果指定的机械臂已定义, 则返回 **True**, 否则返回 **False**。

另见

ArmSet, GetArm, Tool

ArmDef 示例

VB 例:

```
x = m_spel.ArmDef(1)
```

C# 例:

```
x = m_spel.ArmDef(1);
```

ArmSet 方法, Spel 类

描述

定义辅助机器人机械臂。

语法

```
Sub ArmSet ( ArmNumber As Integer, Param1 As Single, Param2 As Single,
             Param3 As Single, Param4 As Single, Param5 As Single )
```

参数

<i>ArmNumber</i>	整数值: 1-15 的有效范围。
<i>Param1</i>	(对于 SCARA 机器人)肘关节中心线至新定向轴中心线的水平距离。 (即新辅助机械臂的定向轴中心线所在的位置。) (对于 Cartesian 机器人)X 轴方向位置相对于 X 轴原点位置的偏移量, 单位为 mm。
<i>Param2</i>	(对于 SCARA 机器人)正常肘中心线与正常定向轴中心线之间的成线以及新辅助机械臂肘中心线和新定向轴中心线之间的成线之间的偏移量(度)。(这 2 条线应在肘中心线处相交, 所成角度即 <i>Param2</i> 。) (对于 Cartesian 机器人)Y 轴方向位置相对于 Y 轴原点位置的偏移量, 单位为 mm。
<i>Param3</i>	(对于 SCARA 和 Cartesian 机器人)新定向轴中心与旧定向轴中心之间的 Z 高度偏移量。(此为距离量。)
<i>Param4</i>	(对于 SCARA 机器人)新辅助轴肩中心线至肘中心线的距离。 (对于 Cartesian 机器人)这是一个虚拟参数(指定为 0)。
<i>Param5</i>	(对于 SCARA 和 Cartesian 机器人)新定向轴与旧定向轴之间的角度偏移量(度)。

另见

Arm, Tool, TLSet

ArmSet 示例**VB 例:**

```
Sub SetArms ()
    With m_spel
        .ArmSet (1, 1.5, 0, 0, 0, 0)
        .ArmSet (2, 3.2, 0, 0, 0, 0)
    End With
End Sub
```

C# 例:

```
void SetArms ()
{
    m_spel.ArmSet (1, 1.5, 0, 0, 0, 0);
    m_spel.ArmSet (2, 3.2, 0, 0, 0, 0);
}
```

Atan 方法, Spel 类

描述

返回数值表达式的弧正切值。

语法

Function **Atan** (*number* As Double) As Double

参数

Number 表示角度值的正切值的数值表达式。

返回值

指定值的弧正切值

另见

Atan2

Atan 示例

VB 例:

```
Dim angle As Double
```

```
angle = m_spel.Atan(.7)
```

C# 例:

```
double angle;
```

```
angle = m_spel.Atan(.7);
```


Atan2 方法, Spel 类

描述

返回虚线连接点(0, 0)和(X, Y)的角度, 单位为弧度。

语法

Function **Atan2** (*Dx* As Double, *Dy* as Double) As Double

参数

Dx 表示 X 坐标的数值表达式。

Dy 表示 Y 坐标的数值表达式。

返回值

含有角度的 double 值。

另见

Atan

Atan2 示例**VB 例:**

```
Dim angle As Double

angle = m_spel.Atan2(-25, 50)
```

C# 例:

```
double angle;

angle = m_spel.Atan2(-25, 50);
```

ATCLR 方法, Spel 类

描述

清除并初始化关节的有效扭矩。

语法

Sub **ATCLR** ()

另见

ATRQ, PTCLR, PTRQ

ATCLR 示例

VB 例:

```
m_spel.ATCLR()
```

C# 例:

```
m_spel.ATCLR();
```

AtHome 方法, Spel 类**描述**

如果当前机器人位于起始点位置, 则返回 True。

语法

Function **AtHome** () As Boolean

返回值

如果当前机器人位于起始点位置, 则返回 True, 否则返回 False。

另见

Home

AtHome 示例**VB 例:**

```
If m_spel.AtHome () Then
    lblCurPos.Text = "Robot is at home position"
Else
    lblCurPos.Text = "Robot is not at home position"
End If
```

C# 例:

```
if(m_spel.AtHome ())
    lblCurPos.Text = "Robot is at home position";
else
    lblCurPos.Text = "Robot is not at home position";
```

ATRQ 方法, Spel 类

描述

返回指定关节的有效扭矩。

语法

Function **ATRQ** (*JointNumber* As Integer) As Single

参数

JointNumber 表示轴编号的整数值(范围: 1~ 机器人关节数量)

返回值

返回 0~1 的实数值。

另见

ATCLR, PTCLR, PTRQ

ATRQ 范例**VB 例:**

```
Dim val As Single
Dim i As Integer
For i = 1 To 4
    val = m_spel.ATRQ(i)
Next i
```

C# 例:

```
float avgTorque;
for(int i = 1; i <= 4; i++)
    avgTorque = m_spel.ATRQ(i);
```

AvgSpeed 方法, Spel 类

描述

返回指定关节的速度的平均绝对值。

语法

Function AvgSpeed (*JointNumber* As Integer) As Single

参数

JointNumber 表示轴编号的整数值(范围: 1~ 机器人关节数量)

返回值

返回 0~1 的实数值。

备注

本方法可以返回指定关节速度的平均绝对值。本方法可用于确认电机的负载状态。返回的结果时 0~1 之间的实数值。最大平均速度为 1。

在执行本方法前，必须先执行 AvgSpeedClear 方法。

本方法有时间限制。请在执行 AvgSpeedClear 方法后，60 秒内执行本方法。超过 60 秒，则会发生错误 4088。

在虚拟控制器和空转的情况下，将计算命令速度而不是实际速度的平均绝对值。
本方法不支持 PG 附加轴。

另见

AvgSpeedClear, PeakSpeed, PeakSpeedClear

AvgSpeed 示例**VB 例:**

```
Dim val As Single
Dim i As Integer
For i = 1 To 4
    val = m_spel.AvgSpeed(i)
Next i
```

C# 例:

```
float avgSpeed;
for(int i = 0; i <=4; i++)
    avgSpeed = m_spel.AvgSpeed(i);
```

AvgSpeedClear 方法, Spel 类

描述

清除并初始化关节速度的平均绝对值。

语法

Sub AvgSpeedClear ()

备注

本方法可以清除指定关节的速度的平均绝对值。

执行 AvgSpeed 方法前，必须先执行本方法。

本方法不支持 PG 附加轴。

另见

AvgSpeed, PeakSpeed, PeakSpeedClear

AvgSpeedClear 示例

VB 例:

```
m_spel.AvgSpeedClear ()
```

C# 例:

```
m_spel.AvgSpeedClear ();
```

AxisLocked 方法, Spel 类**描述**

如果指定轴处于伺服控制之下, 则返回 True。

语法

Function **AxisLocked** (*AxisNumber* As Integer) As Boolean

参数

AxisNumber 表示轴编号的数值表达式。数值可为 1-9。

返回值

如果指定轴处于伺服控制之下, 则返回 True。

另见

SLock, SFree

AxisLocked 示例**VB 例:**

```
If m_spel.AxisLocked(1) Then
    lblAxis1.Text = "Robot axis #1 is locked"
Else
    lblAxis1.Text = "Robot axis #1 is free"
End If
```

C# 例:

```
if (m_spel.AxisLocked(1))
    lblAxis1.Text = "Robot axis #1 is locked";
else
    lblAxis1.Text = "Robot axis #1 is free";
```

Base 方法, Spel 类

描述

定义基座坐标系。

语法

```
Sub Base (OriginPoint As SpelPoint [, XAxisPoint As SpelPoint] [, YAxisPoint As SpelPoint] [, Alignment As SpelBaseAlignment])
```

参数

OriginPoint 表示基座坐标系原点的 SpelPoint。

XAxisPoint 可选。位于基座坐标系 X 轴任意位置的 SpelPoint。

YAxisPoint 可选。位于基座坐标系 Y 轴任意位置的 SpelPoint。

Alignment 可选。提供 *XAxisPoint* 和 *YAxisPoint* 参数时，使用 *Alignment* 参数指定与基座对齐的轴。

另见

Local

Base 示例**VB 例:**

```
Dim originPoint As New SpelPoint  
originPoint.X = 50  
originPoint.Y = 50  
m_spel.Base(originPoint)
```

C# 例:

```
SpelPoint originPoint = new SpelPoint();  
originPoint.X = 50;  
originPoint.Y = 50;  
m_spel.Base(originPoint);
```


BGo 方法, Spel 类

描述

在选定的本地坐标系中执行 PTP 的相对运动。

语法

```
Sub BGo (PointNumber As Integer)
Sub BGo (Point As SpelPoint)
Sub BGo (Point As SpelPoint, AttribExpr As String)
Sub BGo (PointExpr As String)
```

参数

每个语法具有一个指定机械臂在 BGo 动作期间所移至端点的参数。此参数为 PTP 动作结束时的最终位置。

PointNumber 通过对当前机器人控制器点内存中之前示教的点使用点编号来指定端点。

Point 通过使用 SpelPoint 数据类型指定端点。

AttribExpr 通过使用字符串表达式指定端点属性。

PointExpr 通过使用字符串表达式指定端点。

另见

Accel, Speed
Arc, Arc3, CVMove, Go, Jump, Jump3, Jump3CP, Move
BMove, TGo, TMove
CP, Till

BGo 示例**VB 例:**

```
' 使用点编号
m_spel.Tool(1)
m_spel.BGo(100)

' 使用 SpelPoint
Dim pt As SpelPoint
pt = m_spel.GetPoint("P*")
pt.X = 125.5
m_spel.BGo(pt)

' 使用属性表达式
m_spel.BGo(pt, "ROT")

' 使用点表达式
m_spel.BGo("P0 /L /2")
m_spel.BGo("P1 :Z(-20)")

' 使用并行处理
m_spel.BGo("P1 !D50; On 1; D90; Off 1!")

' 使用点标签
m_spel.BGo("pick")
```

C# 例:

```
// 使用点编号
m_spel.Tool(1);
m_spel.BGo(100);

// 使用 SpelPoint
SpelPoint pt;
pt = m_spel.GetPoint("P*");
pt.X = 125.5;
m_spel.BGo(pt);

// 使用属性表达式
m_spel.BGo(pt, "ROT");

// 使用点表达式
m_spel.BGo("P0 /L /2");
m_spel.BGo("P1 :Z(-20)");

// 使用并行处理
m_spel.BGo("P1 !D50; On 1; D90; Off 1!");

// 使用点标签
m_spel.BGo("pick");
```

BMove 方法, Spel 类

描述

在选定的本地坐标系中执行线性内插相对运动。

语法

```
Sub BMove (PointNumber As Integer)
Sub BMove (Point As SpelPoint)
Sub BMove (Point As SpelPoint, AttribExpr As String)
Sub BMove (PointExpr As String)
```

参数

每个语法具有一个指定机械臂在 BMove 动作期间所移至端点的参数。此参数为线性内插动作结束时的最终位置。

PointNumber 通过对当前机器人控制器点内存中之前示教的点使用点编号来指定端点。

Point 通过使用 SpelPoint 数据类型指定端点。

AttribExpr 通过使用字符串表达式指定端点属性。

PointExpr 通过使用字符串表达式指定端点。

另见

AccelR, AccelS, SpeedR, SpeedS
Arc, Arc3, CVMove, Go, Jump, Jump3, Jump3CP, Move
BGo, TGo, TMove
CP, Till

BMove 示例**VB 例:**

```
' 使用点编号
m_spel.Tool(1)
m_spel.BMove(100)

' 使用 SpelPoint
Dim pt As SpelPoint
pt = m_spel.GetPoint("P*")
pt.X = 125.5
m_spel.BMove(pt)

' 使用点表达式
m_spel.BMove("P0 /L /2")

' 使用并行处理
m_spel.BMove("P1 !D50; On 1; D90; Off 1!")

' 使用点标签
m_spel.BMove("pick")
```

C# 例:

```
// 使用点编号
m_spel.Tool(1);
m_spel.BMove(100);

// 使用 SpelPoint
SpelPoint pt;
pt = m_spel.GetPoint("P*");
pt.X = 125.5;
m_spel.BMove(pt);

// 使用点表达式
m_spel.BMove("P0 /L /2");

// 使用并行处理
m_spel.BMove("P1 !D50; On 1; D90; Off 1!");

// 使用点标签
m_spel.BMove("pick");
```

Box 方法, Spel 类

描述

指定 box 内定义的结束检查区域。

语法

Sub **Box** (*AreaNumber* As Integer, *MinX* As Single, *MaxX* As Single, *MinY* As Single, *MaxY* As Single, *MinZ* As Single, *MaxZ* As Single)

Sub **Box** (*AreaNumber* As Integer, *MinX* As Single, *MaxX* As Single, *MinY* As Single, *MaxY* As Single, *MinZ* As Single, *MaxZ* As Single, *PolarityOn* As Boolean)

参数

<i>AreaNumber</i>	1-15 之间表示 15 个待定义 box 的整数。
<i>MinX</i>	结束检查区域的最小 X 坐标位置。
<i>MaxX</i>	结束检查区域的最大 X 坐标位置。
<i>MinY</i>	结束检查区域的最小 Y 坐标位置。
<i>MaxY</i>	结束检查区域的最大 Y 坐标位置。
<i>MinZ</i>	结束检查区域的最小 Z 坐标位置。
<i>MaxZ</i>	结束检查区域的最大 Z 坐标位置。
<i>PolarityOn</i>	可选。使用对应远程输出时设置远程输出逻辑。若要设置为夹具末端在 box 区域内时 I/O 输出打开, 使用 True。若要设置为夹具末端在 box 区域内时 I/O 输出关闭, 使用 False。

另见

BoxClr, BoxDef, Plane

Box 示例**VB 例:**

```
m_spel.Box(1, -5, 5, -10, 10, -20, 20)
```

C# 例:

```
m_spel.Box(1, -5, 5, -10, 10, -20, 20);
```

BoxClr 方法, Spel 类

描述

清除 box(结束检查区域)的定义。

语法

Sub **BoxClr** (*BoxNumber* As Integer)

参数

BoxNumber 1 至 15 之间表示区域编号的整数表达式。

另见

Box, BoxDef

BoxClr 示例

VB 例:

```
m_spel.BoxClr(1)
```

C# 例:

```
m_spel.BoxClr(1);
```

BoxDef 方法, Spel 类

描述

返回是否已定义 Box。

语法

Function **BoxDef** (*BoxNumber* As Integer) As Boolean

参数

BoxNumber 1 至 15 之间表示区域编号的整数表达式。

返回值

如果已定义指定 box，则返回 True，否则返回 False。

另见

Box, BoxClr

BoxDef 示例**VB 例:**

```
x = m_spel.BoxDef(1)
```

C# 例:

```
x = m_spel.BoxDef(1);
```

Brake 方法, Spel 类

描述

读取或设置指定关节的制动状态。

语法

Sub **Brake** (*JointNumber* As Integer, *State* As Boolean)

Function **Brake** (*JointNumber* As Integer) As Boolean

参数

<i>JointNumber</i>	表示轴编号的整数值(范围: 1~ 机器人关节数量)
<i>State</i>	启动制动器: 使用 On。 解除制动器: 使用 Off。

返回值

0 = 制动器 Off

1 = 制动器 On

备注

本方法可以开启或解除制动, 垂直 6 轴型机器人(包括 N 系列)的一个关节。此方法仅供维护人员使用。

执行本方法, 将会初始化机器人的控制参数。



警告

- 解除制动器时, 需谨慎操作。确保关节是否正确固定。如关节固定不当, 可能会导致机器人关节失重掉落, 从而导致机器人故障或人员受伤。

注意

在执行 Brake Off 命令时, 请务必将紧急停止开关放在手边。

当控制器进入紧急停止状态时, 电机制动器将被锁定。执行 Brake Off 命令, 可能会导致机器人手臂由于自重而下降。

另见

Reset, SFree, SLock

Brake 示例**VB 例:**

```
Dim state As Boolean
state = m_spel.Brake(1)
```

C# 例:

```
bool state;
state = m_spel.Brake(1);
```


BTst 方法, Spel 类

描述

返回数字中 1 位的状态。

语法

Function **BTst** (*Number* As Integer, *BitNumber* As Integer) As Boolean

参数

Number 用表达式或数值指定位测试的数字。

BitNumber 指定待测试的位数(0-31 之间的整数值)。

返回值

如果已设置指定位, 则返回 True, 否则返回 False。

另见

On, Off

BTst 示例**VB 例:**

```
x = m_spel.BTst(data, 2)
```

C# 例:

```
x = m_spel.BTst(data, 2);
```

BuildProject 方法, Spel 类**描述**

构建 Project 属性指定的 EPSON RC+ 7.0 项目。

语法

Sub **BuildProject** ()

另见

Project, ProjectBuildComplete, ProjectOverwriteWarningEnabled

BuildProject 示例**VB 例:**

```
With m_spel
    .Project = "c:\EpsonRC70\projects\myproj\myproj.sprj"
    If Not .ProjectBuildComplete() Then
        .BuildProject()
    End If
End With
```

C# 例:

```
m_spel.Project =
@"c:\EpsonRC70\projects\myproj\myproj.sprj";
if(!m_spel.ProjectBuildComplete())
    m_spel.BuildProject();
```

Call 方法, Spel 类**描述**

调用(执行)可随意返回一个值的 SPEL+ 函数。

语法

Function **Call** (*FuncName* As String [, *Parameters* As String]) As Object

参数

FuncName 当前项目中已定义的函数名称。

Parameters 可选。
指定参数列表。使用以逗号(,)分隔的参数。

返回值

SPEL+ 函数的返回值。数据类型与函数的数据类型匹配。

备注

使用 Call 方法调用 SPEL+ 函数并检索返回值。将 Call 结果分配到变量时，确保使用正确的数据类型，否则会发生类型不匹配的错误。

还可调用通过 Visual Basic 应用在 SPEL+ 代码中定义的 DLL 函数。通过 Call 方式执行的函数无法通过 Stop、Pause、Halt 或 Quit 方式停止/暂停。

若需要停止或暂停，请使用 Xgt 方法。

注意

Call 方法执行的函数，不能被 Stop, Pause, Halt 和 Quit 方法停止或暂停。

如需停止或暂停，请使用 Xqt 方法。

另见

Xqt

Call 示例

' Visual Basic 代码

```
Dim errCode As Integer  
errCode = m_spel.Call("GetPart", ""Test"",2")
```

' C# 代码

```
int errCode;  
errCode = m_spel.Call("GetPart", ""Test"",2");
```

' SPEL+ 函数

```
Function GetPart(Info$ As String, Timeout As Integer) As  
Integer  
    Long errNum  
    OnErr GoTo GPErr  
    Print Info$  
    errNum = 0  
    Jump P1  
    On vacuum  
    Wait Sw(vacOn) = On, Timeout  
    If TW = True Then  
        errNum = VAC_TIMEOUT  
    EndIf  
  
    GetPart = errNum  
    Exit Function  
GPErr:  
    GetPart = Err  
Fend
```

CalPIs 方法, Spel 类

描述

读取或设置校准时使用的位置姿势脉冲值。

语法

Function CalPIs (JointNumber As Integer) As Integer

Sub CalPIs (J1Pulses As Integer, J2Pulses As Integer, J3Pulses As Integer, J4Pulses As Integer, [J5Pulses As Integer], [J6Pulses As Integer], [J7Pulses As Integer], [J8Pulses As Integer], [J9Pulses As Integer])

参数

J1Pulses – J9Pulses 表示第 1~9 关节脉冲值的整数
J5Pulses – J9Pulses 可以省略

返回值

如果省略脉冲值, 则会显示当前设置的脉冲值。

备注

输入并保存正确的脉冲值以校准点位。

本方法仅供维护时使用。当更换电机, 导致电机的原点偏离机械臂原点等情况下, 可使用本命令。这个对齐原点的过程则成为校准。

在正常状态下, 校准位置的脉冲值与本方法中设置的脉冲值一致, 但是在维护时(例如更换电机后), 两个值会产生差异, 此时则需要校准。

有一种校准的方法, 是将关节移动到特定的位置后, 执行 Calib。执行 Calib 可以将校准位置的脉冲值更改为本方法中指定的脉冲值(用于校准位置的正确的脉冲值)。

必须设置 Hofs 才能进行校准。要自动计算 Hofs 值, 需要将关节移动到要校准的位置, 然后执行 Calib。控制器会根据校准位置的脉冲值, 自动计算 Hofs 的值。

注意**不能通过关闭电源来更改 CalPIs 值**

即使关闭控制器并重启后, CalPIs 的值也不会被初始化。要更改 CalPIs 值, 只能通过执行 Calib 命令来进行。

另见

Hofs

CalPIs 示例**VB 例:**

```
Dim val As Single
Dim i As Integer
For i = 1 To 4
    val = m_spel.CalPIs(i)
Next i
```

C# 例:

```
float val;
for(int i = 1; i <= 4; i++)
    val = m_spel.CalPIs(i);
```

ClearPoints 方法, Spel 类

描述

清除当前机器人内存中的点。

语法

Sub **ClearPoints** ()

另见

LoadPoints, Robot, SavePoints, SetPoint

ClearPoints 示例

VB 例:

```
With m_spel
    .ClearPoints ()
    .SetPoint(1, 100, 200, -20, 0, 0, 0)
    .Jump(1)
End With
```

C# 例:

```
m_spel.ClearPoints ();
m_spel.SetPoint(1, 100, 200, -20, 0, 0, 0);
m_spel.Jump(1);
```

Connect 方法, Spel 类

描述

连接 Spel 类实例和控制器。

语法

```
Sub Connect (ConnectionName As String)
Sub Connect (ConnectionName As String, ConnectionPassword As String)
Sub Connect (ConnectionNumber As Integer)
Sub Connect (ConnectionNumber As Integer, ConnectionPassword As String)
```

参数

ConnectionName 表示连接名称的字符串。

ConnectionNumber 连接编号的整数表达式。
连接编号是 EPSON RC+中[设置] - [电脑与控制器连接]对话框的编号。如果使用上次成功连接的对象，请执行-1。

ConnectionPassword 连接密码的字符串。
若控制器中有连接控制器的密码，并且该密码并未在 EPSON RC+中[设置] - [电脑与控制器连接]对话框中设置，则需要设置控制器连接密码。

备注

Spel 类实例需要与控制器通信时，便会自动连接。如果需要明确连接至控制器，则使用 Connect 方法。

另见

Disconnect, Initialize

Connect 示例**VB 例:**

```
Try
    m_spel.Connect(1)
Catch ex As RCAPINet.SpelException
    MsgBox(ex.Message)
End Try
```

C# 例:

```
try{
    m_spel.Connect(1);
}
catch(RCAPINet.SpelException ex){
    MessageBox.Show(ex.Message);
}
```

Continue 方法, Spel 类

描述

如果发生暂停, 则恢复控制器中的所有任务。

语法

```
Sub Continue ()
```

备注

使用 **Continue** 恢复已被 **Pause** 方法或安全防护打开所暂停的所有任务。

如果安全防护在任务运行时打开, 则机器人会逐渐减速至停止且机器人电机将关闭。安全防护关闭之后, 可使用 **Continue** 恢复循环。

另见

Pause, Start, Stop

Continue 示例**VB 例:**

```
Sub btnContinue_Click( _
    ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnContinue.Click

    btnPause.Enabled = True
    btnContinue.Enabled = False
    Try
        m_spel.Continue()
    Catch ex As RCAPINet.SpelException
        MsgBox(ex.Message)
    End Try
End Sub
```

C# 例:

```
void btnContinue_Click(object sender, EventArgs e)
{
    btnPause.Enabled = true;
    btnContinue.Enabled = true;

    try{
        m_spel.Continue();
    }
    catch(RCAPINet.SpelException ex){
        MessageBox.Show(ex.Message);
    }
}
```


Ctr 方法, Spel 类

描述

返回指定输入计数器的计数器值。

语法

Function **Ctr** (*BitNumber* As Integer) As Integer

参数

BitNumber 设置为计数器的输入位数。
同一时间只能有 16 个计数器处于活动状态。

返回值

返回计数器值。(整数范围为 0 至 65535)

另见

CtReset

Ctr 示例**VB 例:**

```
lblCounter.Text = m_spel.Ctr(1).ToString()
```

C# 例:

```
lblCounter.Text = m_spel.Ctr(1).ToString();
```

CtReset 方法, Spel 类

描述

重置指定输入计数器的计数器值。还将输入定义为计数器输入。

语法

Sub **CtReset** (*BitNumber* As Integer)

参数

BitNumber 设置为计数器的输入位数。
同一时间只能有 16 个计数器处于活动状态。

另见

Ctr

CtReset 示例

VB 例:

```
m_spel.CtReset(2)
```

C# 例:

```
m_spel.CtReset(2);
```

Curve 方法, Spel 类

描述

定义沿曲线路径移动机械臂所需的数据和点。可在路径中定义多个数据点，以提高路径精度。

语法

Sub **Curve** (*FileName* As String, *Closure* As Boolean, *Mode* As Integer, *NumOfAxis* As Integer, *PointList* As String)

参数

FileName 点数据存储文件的路径和名称字符表达式。指定 *fileName* 的结尾附有扩展名 **CRV**，因此用户无需指定扩展名。执行 **Curve** 指令时，将创建 *fileName*。

Closure 指定是否将路径的最后一个点连接至第一个点的 Boolean 表达式。

Mode 指定机械臂是否在 U 轴的正切方向自动内插。

模式设置	正切校正
0	No
2	Yes

NumOfAxis 2-4 之间指定曲线动作期间所控制轴数的整数表达式，如下所示：

2: 在 XY 平面生成曲线，无 Z 轴移动或 U 轴旋转。

3: 在 XYZ 平面生成曲线，无 U 轴旋转。

(Theta 1、Theta2 和 Z)

4: 在 XYZ 平面生成曲线，U 轴旋转。(控制所有 4 个轴)

PointList { 点表达式 | P(*起始点*:*终点*) } [, *输出命令*] ...

该参数实际是一系列点编号以及以逗号分隔的可选输出语句或以冒号分隔的升序排列的点。

通常，这一系列点以逗号分隔，如下所示：

```
Curve MyFile, 0, 0, 4, P1, P2, P3, P4
```

或以冒号指定，如下所示：

```
Curve MyFile, 0, 0, 4, P(1:4)
```

备注

用 **Curve** 定义待用 **CVMove** 方法执行的样条路径。有关更多详细信息，请参阅 **SPEL+** 命令 **Curve**。

另见

Curve (SPEL+ 语句), **CVMove** 方法

Curve 示例**VB 例:**

```
m_spel.Curve("mycurveFile", True, 0, 4, "P(1:3), On 1, P(4:7)")
m_spel.CVMove("mycurveFile")
```

C# 例:

```
m_spel.Curve("mycurveFile", True, 0, 4, "P(1:3), On 1, P(4:7)");
m_spel.CVMove("mycurveFile");
```

CVMove 方法, Spel 类

描述

执行 **Curve** 指令定义的连续样条路径动作。

语法

Sub **CVMove** (*FileName* As String [, *OptionList* As String])

参数

FileName 连续路径动作数据所需文件的路径和名称字符串表达式。此文件必须事先通过 **Curve** 指令创建。

OptionList 可选。含有 **Till** 规范的字符表达式。

备注

使用 **CVMove** 执行用 **Curve** 方法定义的路径。有关更多详细信息，请参阅 **SPEL+** 命令 **CVMove**。

如果需要用 **CP** 执行 **CVMove**，则建议通过 **SPEL+** 任务代替应用来执行 **CVMove**。其原因在于：为正确执行 **CP** 动作，系统需要提前了解下一动作的目标位置。但由于每次执行一条 **RC+** API 命令，系统无法提前了解下一目标位置。

另见

Curve, **CVMove** (**SPEL+** 命令)

CVMove 示例**VB 例:**

```
m_spel.Curve("mycurveFile", True, 0, 4, "P(1:3), On 1,
P(4:7)")
m_spel.CVMove("mycurveFile", "CP Till Sw(1) = 1")
m_spel.CVMove("mycurveFile")
```

C# 例:

```
m_spel.Curve("mycurveFile", True, 0, 4, "P(1:3), On 1,
P(4:7)");
m_spel.CVMove("mycurveFile", "CP Till Sw(1) = 1");
m_spel.CVMove("mycurveFile");
```

CX、CY、CZ、CU、CV、CW、CR、CS、CT 方法, Spel 类

描述

检索点的坐标值

CV 和 CW 用于 6 轴机器人

CS 和 CT 用于附加轴

CR 用于 Joint 7 轴机器人

语法

Function **CX** (*PointExpr* As String) As Single

Function **CY** (*PointExpr* As String) As Single

Function **CZ** (*PointExpr* As String) As Single

Function **CU** (*PointExpr* As String) As Single

Function **CV** (*PointExpr* As String) As Single

Function **CW** (*PointExpr* As String) As Single

Function **CR** (*PointExpr* As String) As Single

Function **CS** (*PointExpr* As String) As Single

Function **CT** (*PointExpr* As String) As Single

参数

PointExpr 指定待检索指定坐标的点的字符串表达式。可使用任一有效的点表达式。P* 还可用于检索当前位置的坐标。

返回值

指定坐标值。

CX、CY、CZ 的返回值: 实际值(mm)

CU、CV、CW 的返回值: 实际值(度)

CR、CS、CT 的返回值: 实际值

另见

GetPoint, SetPoint

CX、CY、CZ、CU、CV、CW、CR、CS、CT 示例**VB 例:**

```
Dim x As Single, y As Single
x = m_spel.CX("P1")
y = m_spel.CY("P*")
```

C# 例:

```
float x, y;
x = m_spel.CX("P1");
y = m_spel.CY("P*");
```

Delay 方法, Spel 类

描述

延迟指定的毫秒数。

语法

Sub **Delay** (*Milliseconds As Integer*)

参数

Milliseconds 含有待延迟毫秒数的整数值。

Delay 示例

VB 例:

```
m_spel.Delay(500)
```

C# 例:

```
m_spel.Delay(500);
```

DegToRad 方法, Spel 类**描述**

将度转换为弧度。

语法

Function **DegToRad** (*degrees* As Double) As Double

参数

Degrees 待转换为弧度的度数。

返回值

含有弧度的双值。

另见

RadToDeg

DegToRad 示例**VB 例:**

```
Dim rad As Double

rad = m_spel.DegToRad(45)
```

C# 例:

```
double rad;

rad = m_spel.DegToRad(45);
```

Disconnect 方法, Spel 类

描述

断开 Spel 类实例的当前连接。

语法

Sub **Disconnect** ()

备注

用 **Disconnect** 断开当前控制器的连接。

另见

Connect, Initialize

Disconnect 示例**VB 例:**

```
Try
    m_spel.Disconnect()
Catch ex As RCAPINet.SpelException
    MsgBox(ex.Message)
End Try
```

C# 例:

```
try{
    m_spel.Disconnect();
}
catch(RCAPINet.SpelException ex){
    MessageBox.Show(ex.Message);
}
```


ECP 方法, Spel 类

描述

选择当前的 ECP 定义。

语法

Sub **ECP** (*ECPNumber* As Integer)

参数

ECPNumber 0-15 之间表示 16 个 ECP 定义中将与下一动作指令一同使用的定义的整数。

另见

ECPSet

ECP 示例**VB 例:**

```
m_spel.ECP(1)
m_spel.Move("P1 ECP")
```

C# 例:

```
m_spel.ECP(1);
m_spel.Move("P1 ECP");
```

ECPClr 方法, Spel 类

描述

清除(取消定义)当前机器人的外部控制点。

语法

Sub **ECPClr** (*ECPNumber* As Integer)

参数

ECPNumber 表示待清除(取消定义)的 15 个外部控制点之一的整数表达式。
(ECP 0 为默认点, 无法清除。)

另见

ECP, ECPDef

ECPClr 示例

VB 例:

```
m_spel.ECPClr(1)
```

C# 例:

```
m_spel.ECPClr(1);
```

ECPDef 方法, Spel 类

描述

返回 ECP 定义状态。

语法

Function **ECPDef** (*ECPNumber* As Integer) As Boolean

参数

ECPNumber 表示返回状态的 ECP 的整数值。

返回值

如果已定义指定 ECP, 则返回 True, 否则返回 False。

另见

ECP, ECPClr

ECPDef 示例**VB 例:**

```
x = m_spel.ECPDef(1)
```

C# 例:

```
x = m_spel.ECPDef(1);
```

ECPSet 方法, Spel 类

描述

定义 ECP(外部控制点)。

语法

Sub **ECPSet** (*ECPNumber* As Integer, *Point* As *SpelPoint*)

Sub **ECPSet** (*ECPNumber* As Integer, *XCoord* as Double, *YCoord* as Double, *ZCoord* as Double, *UCoord* as Double [, *VCoord* As Double] [, *WCoord* as Double])

参数

ECPNumber 1-15 之间表示 15 个外部控制点中待定义点的整数值。

Point 含有点数据的 *SpelPoint*。

XCoord 外部控制点 X 坐标。

YCoord 外部控制点 Y 坐标。

ZCoord 外部控制点 Z 坐标。

UCoord 外部控制点 U 坐标。

VCoord 可选。外部控制点 V 坐标。

WCoord 可选。外部控制点 W 坐标。

另见

ArmSet, ECP, GetECP, TLSet

ECPSet 示例**VB 例:**

```
m_spel.ECPSet(1, 100.5, 99.3, 0, 0)
```

C# 例:

```
m_spel.ECPSet(1, 100.5, 99.3, 0, 0);
```

EnableEvent 方法, Spel 类

描述

启用 EventReceived 事件的某些系统事件。

语法

Sub **EnableEvent** (EventNumber As RCAPINet.SpelEvents, Enabled as Boolean)

参数

Event 待启用或禁用的事件。

Enabled 设为 True 启用事件, 设为 False 禁用事件。

另见

EventReceived

EnableEvent 示例**VB 例:**

```
With m_spel
    .EnableEvent (RCAPINet.SpelEvents.ProjectBuildStatus, True)
    .BuildProject ()
End With
```

C# 例:

```
m_spel.EnableEvent (RCAPINet.SpelEvents.ProjectBuildStatus,
true);
m_spel.BuildProject ();
```

ExecuteCommand 方法, Spel 类

描述

向 EPSON RC+ 7.0 发送命令并等待命令完成。

语法

Sub **ExecuteCommand** (*Command* As String , [ByRef *Reply* As String])

参数

Command 含有 SPEL+ 命令的字符串。

Reply 可选回复。

备注

通常情况下, ExecuteCommand 不是必需的。大多数操作均可通过 Spel 方法执行。然而, 有时候需要执行 SPEL+ 多语句。多语句即含有以分号分隔的多个语句的一行命令。使用 ExecuteCommand 执行多语句。

例如:

```
m_spel.ExecuteCommand("JUMP pick; ON tipvac")  
最大命令行长度为 200 个字符。
```

另见

Pause

ExecuteCommand 示例**VB 例:**

```
m_spel.ExecuteCommand("JUMP P1!D50; ON 1!")
```

C# 例:

```
m_spel.ExecuteCommand("JUMP P1!D50; ON 1!");
```

FBusIO_GetBusStatus 方法, Spel 类**描述**

返回指定的现场总线的状态。

语法

Function FBusIO_GetBusStatus (*BusNumber* As Integer) As Integer

参数

BusNumber 显示现场总线系统编号的整数值。
该数字始终为 16，是连接到控制器 PC 端的现场总线主板的总线的 ID。
显示要定义的外部控制点的表达式，或 1~15 的整数表达式。

返回值

0 – OK
1 – 未连接
2 – 关机

备注**注意**

本方法仅在启用现场总线主站选件时，方可使用。

另见

FBusIO_GetDeviceStatus, FBusIO_SendMsg, IsOptionActive

FBusIO_GetBusStatus 示例**VB 例::**

```
Dim val As Integer  
val = m_spel.FBusIO_GetBusStatus(16)
```

C# 例:

```
int busStatus;  
busStatus = m_spel.FbusIO_GetBusStatus(16);
```

FBusIO_GetDeviceStatus 方法, Spel 类

描述

返回指定的现场总线设备的状态。

语法

Function FBusIO_GetDeviceStatus (*BusNumber* As Integer, *DeviceID* As Integer) As Integer

参数

BusNumber 显示现场总线系统编号的整数值。
该数字始终为 16, 是连接到控制器 PC 端的现场总线主板的总线的 ID。
显示要定义的外部控制点的表达式, 或 1~15 的整数表达式。

DeviceID 显示设备的现场总线 ID 的整数值。

返回值

0 – OK
1 – 未连接
2 – 关机
3 – 同步错误: 设备正在初始化, 或者设备的波特率有误。

备注**注意**

本方法仅在启用现场总线主站选件时, 方可使用。

另见

FBusIO_GetBusStatus, FBusIO_SendMsg, IsOptionActive

FBusIO_GetDeviceStatus 示例**VB 例::**

```
Dim val As Integer
val = m_spel.FBusIO_GetDeviceStatus(16, 10)
```

C# 例:

```
int deviceStatus;
deviceStatus = m_spel.FbusIO_GetDeviceStatus(16, 10);
```


FBusIO_SendMsg 方法, Spel 类

描述

向现场总线 I/O 设备发送消息并返回应答。

语法

```
Sub FBusIO_SendMsg (BusNumber As Integer, DeviceID As Integer, MsgParam As Integer, SendData As Byte(), ByRef RecvData As Byte())
```

参数

- BusNumber* 显示现场总线系统编号的整数值。
该数字始终为 16，是连接到控制器 PC 端的现场总线主板的总线的 ID。
显示要定义的外部控制点的表达式，或 1~15 的整数表达式。
- DeviceID* 显示设备的现场总线 ID 的整数值。
- MsgParam* 显示消息参数的整数表达式
不适用于 DeviceNet。
- SendData* 在作为字节类型的数组中指定发送到设备的数据。该数组的大小必须和发送的字节数的大小相同。如果不需要发送数据，请设置为“0”。
- RecvData* 从设备接受的数据指定为字节类型数组。根据接收到的字节数，此数组自动转换字节数的大小。

备注**注意**

本方法仅在启用现场总线主站选件时，方可使用。

另见

FBusIO_GetBusStatus, FBusIO_GetDeviceStatus, IsOptionActive

FBusIO_SendMsg 示例**VB 例::**

```
' 向 DeviceNet 发送信息
Dim recvData() as Byte
Dim sendData(6) as Byte
Array.Clear(sendData, 0, sendData.Length)
sendData(0) = 14 '命令
sendData(1) = 1 '类
sendData(3) = 1 '示例
sendData(5) = 7 '属性
' DeviceNet 的 MsgParam 为 0
m_spel.FbusIO_SendMsg(16, 1, 0, sendData, recvData)

' 向 Profibus 发送信息
Dim recvData() As Byte;
m_spel.FbusIO_SendMsg(16, 1, 56, Nothing, recvData);
```

C# 例::

```
// 向 DeviceNet 发送信息
byte[] sendData, recvData;
byte[] sendData = new byte[6];
Array.Clear(sendData, 0, sendData.Length);
sendData[0] = 14; //命令
sendData[1] = 1; //类
sendData[3] = 1; //示例
sendData[5] = 7; //属性
// DeviceNet 的 MsgParam 为 0
m_spel.FbusIO_SendMsg(16, 201, 0, sendData, out recvData);

// 向 Profibus 发送信息
byte[] recvData;
m_spel.FbusIO_SendMsg(16, 1, 56, null, out recvData);
```

FGGet 方法, Spel 类

描述

获取力觉引导序列, 或力觉引导对象的结果。

语法

```
Sub FGGet (Sequence As String, [Object As String], Property As SpelForceProps, ByRef Result As Boolean)
```

```
Sub FGGet (Sequence As String, [Object As String], Property As SpelForceProps, ByRef Result As Double)
```

```
Sub FGGet (Sequence As String, [Object As String], Property As SpelForceProps, ByRef Result As Integer)
```

```
Sub FGGet (Sequence As String, [Object As String], Property As SpelForceProps, ByRef Result As String)
```

参数

Sequence 显示力觉引导序列或力觉引导对象的名称的字符串变量。

Object 力觉引导对象的名称, 或显示力觉引导对象名称的字符串变量。
如需获得力觉序列的结果时, 可省略此操作。

Property 获得值的结果的名称

Result 显示一个变量, 代表返回的值。
数字和类型因返回的结果而异。

另见

FGRUN

FGGet 示例**VB 例::**

```
Dim val As Integer
m_spel.MotorsOn = True

m_spel.FGRUN ("Sequence1")
m_spel.FGGet ("Sequence1", "", SpelForceProps.EndStatus, val)
```

C# 例:

```
int errCode;
m_spel.MotorsOn = true;

m_spel.FGRUN ("Sequence1");
m_spel.FGGet ("Sequence1", "", SpelForceProps.EndStatus, val);
```

FGRun 方法, Spel 类

描述

执行力觉引导序列。

语法

Sub FGRun (*Sequence As String*)

参数

Sequence 力觉引导序列的名称, 或显示力觉引导序列名称的字符串变量。

备注

执行指定的力觉引导序列。力觉引导序列从执行 FGRun 语句的位置开始。在执行之前, 请先用 Go 语句或 Move 语句等运动指令, 移动到所需的开始位置。

FGRun 会在指定的力觉引导序列结束后, 进入下一条语句。

如需获得 FGRun 中执行的序列的结果, 请使用 FGGe。

当 CP 参数或 CP 语句中使用了通过运动, 请等待其停止后再执行力觉引导序列。

开始执行时, 如果满足以下任一条件, 则会发生错误。

- 程序中指定的机器人和 RobotNumber 属性中指定的机器人不一致。
请在 Robot 语句中指定正确的机器人。
- 程序中执行的机器人型号和 RobotType 属性中指定的机器人型号不一致。
请在 Robot 语句中指定正确的机器人型号。
- 程序中指定的工具编号和 RobotTool 属性中指定的工具编号不一致。
请在 Tool 语句中指定正确的 Tool 编号。
- 电机已关闭。
请在 Motor 语句中开启电机。
- 正在运行力觉控制功能。
请在 FCEnd 语句中停止力觉控制。
- 正在运行传送带跟踪。
请在 Cnv_AbortTrack 语句中停止传送带跟踪。
- 在扭矩控制模式下。
请在 TC 语句中关闭扭矩控制模式。

FGRun 在执行时会自动覆盖以下属性, 因此不能与以下属性同时适用。

FM 对象

AvgForceClear 属性

PeakForceClear 属性

当正在执行程序时, 无法执行本方法。

另见

FGGet

FGRUN 示例**VB 例::**

```
Dim errCode As Integer
m_spel.MotorsOn = True

m_spel.FGRUN("Sequence1")
errCode = m_spel.FGGet("Sequence1", SpelForceProps.EndStatus, val)
```

C# 例:

```
int errCode;
m_spel.MotorsOn = true;

m_spel.FGRUN("Sequence1");
errCode = m_spel.FGGet("Sequence1", SpelForceProps.EndStatus,
val);
```

Find 方法, Spel 类

描述

设置动作指令期间保存坐标的条件。

语法

Sub **Find** (*Condition* As String)

参数

Condition 指定作为触发器的输入状态。

另见

Go, Jump

Find 示例

VB 例:

```
m_spel.Find("Sw(5) = On")
```

C# 例:

```
m_spel.Find("Sw(5) = On");
```

Fine 方法, Spel 类

描述

指定并显示目标点的定位精度。

语法

```
Sub Fine ( J1MaxErr As Integer, J2MaxErr As Integer, J3MaxErr As Integer,  
          J4MaxErr As Integer, J5MaxErr As Integer, J6MaxErr As Integer  
          [, J7MaxErr As Integer] [, J8MaxErr As Integer] [, J9MaxErr As Integer] )
```

参数

J1MaxErr - *J9MaxErr* 0-32767 之间表示各关节允许定位误差的整数值。
关节 7、8 和 9 的此数值是可选的。

另见

Weight

Fine 示例**VB 例:**

```
m_spel.Fine(1000, 1000, 1000, 1000, 0, 0)
```

C# 例:

```
m_spel.Fine(1000, 1000, 1000, 1000, 0, 0);
```

Force_Calibrate 方法, Spel 类

描述

为当前力传感器的所有轴设置零偏移。

语法

Sub **Force_Calibrate()**

备注

应在应用启动时为每个传感器调用 Force_Calibrate。这样将计入传感器上安装的组件重量。

注意

仅在安装了力感测(ATI 力传感器)时, 可使用本方法。

使用爱普生力觉传感器时, 请使用 FGGet 方法和 FGRun 方法。

另见

Force_Sensor, Force_GetForces, Force_SetTrigger

Force_Calibrate 示例

VB 例:

```
m_spel.ForceSensor = 1  
m_spel.Force_Calibrate()
```

C# 例:

```
m_spel.ForceSensor = 1;  
m_spel.Force_Calibrate();
```

Force_ClearTrigger 方法, Spel 类**描述**

清除当前力传感器的所有触发条件。

语法

```
Sub Force_ClearTrigger()
```

备注

使用 Force_ClearTrigger 清除当前力传感器的所有触发条件。

注意

仅在安装了力感测(ATI 力传感器)时, 可使用本方法。

使用爱普生力觉传感器时, 请使用 FGGet 方法和 FGRun 方法。

另见

Force_Sensor, Force_GetForces, Force_SetTrigger

Force_ClearTrigger 示例**VB 例:**

```
m_spel.ForceSensor = 1  
m_spel.Force_ClearTrigger()
```

C# 例:

```
m_spel.ForceSensor = 1;  
m_spel.Force_ClearTrigger();
```

Force_GetForce 方法, Spel 类

描述

返回指定力传感器轴的力。

语法

Function **Force_GetForce**(*Axis* As SpelForceAxis) As Single

参数

Axis 待检索的轴值，如下所示：

SpelForceAxis	数值
XForce	1
YForce	2
ZForce	3
XTorque	4
YTorque	5
ZTorque	6

备注

使用 Force_GetForce 读取当前为一个轴设置的力。单位由力传感器的配置决定。

注意

仅在安装了力感测(ATI 力传感器)时，可使用本方法。

使用爱普生力觉传感器时，请使用 FGGet 方法和 FGRun 方法。

另见

Force_Sensor, Force_GetForces, Force_SetTrigger

Force_GetForce 示例**VB 例:**

```
m_spel.ForceSensor = 1
zForce = m_spel.Force_GetForce(SpelForceAxis.ZForce)
```

C# 例:

```
m_spel.ForceSensor = 1;
zForce = m_spel.Force_GetForce(SpelForceAxis.ZForce);
```

Force_GetForces 方法, Spel 类**描述**

以数组形式返回所有力传感器轴的力和扭矩。

语法

```
Sub Force_GetForces( Values() As Single)
```

参数

Values 将返回的具有 6 个元素的一维数组。

备注

使用 Force_GetForces 一次读取所有力值和扭矩值。

注意

**仅在安装了力感测(ATI 力传感器)时, 可使用本方法。
使用爱普生力觉传感器时, 请使用 FGGet 方法和 FGRun 方法。**

另见

Force_Sensor, Force_GetForces, Force_SetTrigger

Force_GetForces 示例**VB 例:**

```
Dim values() as Single = Nothing  
m_spel.ForceSensor = 1  
m_spel.Force_GetForces(values)
```

C# 例:

```
float[] values = new float[6];  
  
m_spel.ForceSensor(1);  
m_spel.Force_GetForces(values);
```

Force_SetTrigger 方法, Spel 类

描述

为 Till 命令设置力触发器。

语法

Sub **Force_SetTrigger**(*Axis* As SpelForceAxis, *Threshold* As Single, *CompareType* As SpelForceCompareType)

参数

Axis 用于触发器的轴，如下所示：

SpelForceAxis	数值
XForce	1
YForce	2
ZForce	3
XTorque	4
YTorque	5
ZTorque	6

Threshold 表示阈值的单一表达式。

CompareType LessOrEqual 或 GreaterOrEqual。

备注

若要停止力传感器的动作，必须设置传感器的触发器，然后在动作语句中使用 Till Force。

可使用多个轴设置触发器。然后为每个轴调用 Force_SetTrigger。

若要清除所有触发条件，使用 Force_ClearTrigger。

注意

仅在安装了力感测(ATI 力传感器)时，可使用本方法。

使用爱普生力觉传感器时，请使用 FGGet 方法和 FGRun 方法。

另见

Force_ClearTrigger, Force_Sensor, Till

Force_SetTrigger 示例**VB 例:**

```
m_spel.ForceSensor = 1
m_spel.Force_SetTrigger(SpelForceAxis.ZForce, -2.0, _
    SpelForceCompareType.GreaterOrEqual)
m_spel.Till("Force")
m_spel.Move("P1 Till")
```

C#例:

```
m_spel.ForceSensor = 1;
m_spel.Force_SetTrigger(SpelForceAxis.ZForce, -2.0,
    SpelForceCompareType.GreaterOrEqual);
m_spel.Till("Force");
m_spel.Move("P1 Till");
```

GetAccel 方法, Spel 类

描述

返回指定加速度/减速度值。

语法

Function **GetAccel** (*ParamNumber* As Integer) As Integer

参数

ParamNumber 可含有以下数值的整数表达式:

- 1: 加速度规定值
- 2: 减速度规定值
- 3: Jump 的起始加速度规定值
- 4: Jump 的起始减速度规定值
- 5: Jump 的结束加速度规定值
- 6: Jump 的结束减速度规定值

返回值

含有指定加速度/减速度值的整数。

另见

Accel

GetAccel 示例**VB 例:**

```
Dim x As Integer  
x = m_spel.GetAccel(1)
```

C# 例:

```
int x;  
x = m_spel.GetAccel(1);
```

GetArm 方法, Spel 类

描述

返回当前机器人的当前机械臂编号。

语法

Function **GetArm** () As Integer

返回值

含有当前机械臂编号的整数。

另见

Arm, ArmSet, Robot, Tool

GetArm 示例

VB 例:

```
saveArm = m_spel.GetArm()  
m_spel.Arm(2)
```

C# 例:

```
saveArm = m_spel.GetArm();  
m_spel.Arm(2);
```

GetConnectionInfo 方法, Spel 类**描述**

返回控制器连接的相关信息。

语法

Function **GetConnectionInfo**() As SpelConnectionInfo()

返回值

SpelConnectionInfo 的数组。

另见

GetControllerInfo

备注

GetConnectionInfo 返回 SpelConnectionInfo 的数组。连接信息在 EPSON RC+ 从 [Setup]-[PC to Controller Communication]对话框中配置。

GetConnectionInfo 示例**VB 例:**

```
Dim info() As SpelConnectionInfo  
  
info = m_spel.GetConnectionInfo()
```

C# 例:

```
SpelConnectionInfo[] info = m_spel.GetConnectionInfo();
```

GetControllerInfo 方法, Spel 类

描述

返回当前控制器的相关信息。

语法

Function **GetControllerInfo**() As SpelControllerInfo

返回值

SpelControllerInfo 实例。

另见

GetErrorMessage, GetRobotInfo, GetTaskInfo

备注

GetControllerInfo 返回 SpelControllerInfo 类的新实例，其含有控制器的信息属性。

GetControllerInfo 示例**VB 例:**

```
Dim info As SpelControllerInfo
Dim msg As String

info = m_spel.GetControllerInfo()
msg = "Project Name: " & info.ProjectName & vbCrLf _
    & "Project ID: " & info.ProjectID
MsgBox(msg)
```

C# 例:

```
SpelControllerInfo info;
string msg;

info = m_spel.GetControllerInfo();
msg = "Project Name:" + info.ProjectName + "\r\n"
    + "ProjectID :" +
    "info.ProjectID";
MessageBox.Show(msg);
```


GetCurrentConnectionInfo 方法, Spel 类**描述**

返回当前控制器的连接信息。

语法

Function **GetCurrentConnectionInfo** () As SpelConnectionInfo

返回值

SpelConnectionInfo

另见

GetConnectionInfo, GetControllerInfo

GetCurrentConnectionInfo 示例**VB 例::**

```
Dim info As SpelConnectionInfo

info = m_spel.GetCurrentConnectionInfo()
```

C# 例:

```
SpelConnectionInfo info;

info = m_spel.GetCurrentConnectionInfo();
```

GetCurrentUser 方法, Spel 类

描述

返回当前的 EPSON RC+ 7.0 用户。

语法

Function **GetCurrentUser** () As String

返回值

含有当前用户的字符串变量。

另见

Login

GetCurrentUser 示例

VB 例:

```
Dim currentUser As String  
  
currentUser = m_spel.GetCurrentUser ()
```

C# 例:

```
string currentUser;  
  
currentUser = m_spel.GetCurrentUser ();
```

GetECP 方法, Spel 类**描述**

返回当前机器人的当前 ECP 编号。

语法

Function **GetECP** () As Integer

返回值

含有当前 ECP 编号的整数。

另见

ECP, ECPSet

GetECP 示例**VB 例:**

```
saveECP = m_spel.GetECP()  
m_spel.ECP(2)
```

C# 例:

```
saveECP = m_spel.GetECP();  
m_spel.ECP(2);
```

GetErrorMessage 方法, Spel 类

描述

返回指定错误的错误消息或警告代码。

语法

Function **GetErrorMessage** (*ErrorCode* As Integer) As String

参数

ErrorCode 用以返回相关错误消息的错误代码。

返回值

含有错误消息的字符串。

另见

ErrorCode

GetErrorMessage 示例**VB 例:**

```
Dim msg As String

If m_spel.ErrorOn Then
    msg = m_spel.GetErrorMessage(m_spel.ErrorCode)
    MsgBox(msg)
End If
```

C# 例:

```
string msg;

if (m_spel.ErrorOn) {
    msg = m_spel.GetErrorMessage(m_spel.ErrorCode);
    MessageBox.Show(msg);
}
```

GetIODef 方法, Spel 类

描述

获取输入、输出或内存 I/O 位、字节或字的定义信息。

语法

```
Sub GetIODef(Type As SpellIOLabelTypes, Index As Integer, ByRef Label as String,
ByRef Description As String)
```

参数

Type 指定 I/O 类型，如下所示：

InputBit = 1 InputByte = 2 InputWord = 3
OutputBit = 4 OutputByte = 5 OutputWord = 6
MemoryBit = 7 MemoryByte = 8 MemoryWord = 9
InputReal = 10 OutputReal = 11

Index 指定位或端口号。

Label 返回标签。

Description 返回描述。

返回值

数值通过 Label 和 Description 参数返回。

备注

使用 GetIODef 获得用于当前项目中所有 I/O 的标签和描述。

另见

SetIODef

GetIODef 示例**VB 例:**

```
Dim label As String
Dim desc As String
m_spel.GetIODef(SpellIOLabelTypes.InputBit, 0, label, desc)
```

C# 例:

```
string label, desc;

m_spel.GetIODef(SpellIOLabelTypes.InputBit, 0, out label, out
desc);
```

GetJRange 方法, Spel 类

描述

获取指定关节的允许工作区域脉冲值(范围设定值)。

语法

Function **GetJRange** (*JointNumber* As Integer, *Bound* As Integer) As Integer

参数

JointNumber 表示关节编号的整数值 (范围: 1~ 机器人的关节数量)

Bound 用整数指定以下两个值的其中之一。

1: 指定下限脉冲值

2: 指定上限脉冲值

返回值

返回指定关节的范围设定值(整数值, 单位:脉冲)。

另见

JRange

GetJRange 示例**VB 例::**

```
Dim val1 As Integer
Dim val2 As Integer
val1 = m_spel.GetJRange(1, 1)
val2 = m_spel.GetJRange(1, 2)
```

C# 例:

```
int minRange, maxRange;
minRange = m_spel.GetJRange(1, 1);
maxRange = m_spel.GetJRange(1, 2);
```

GetLimitTorque 方法, Spel 类**描述**

返回当前机器人指定关节的限制扭矩。

语法

Function **GetLimitTorque** (*JointNumber* As Integer) As Integer

参数

JointNumber 所需关节的整数表达式。

返回值

表示指定关节限制扭矩设置的 1 至 9 整数值。

另见

GetRealTorque, GetRobotPos

GetLimitTorque 示例**VB 例:**

```
Dim j1LimitTorque As Integer  
j1LimitTorque = m_spel.GetLimitTorque(1)
```

C# 例:

```
int j1LimitTorque;  
j1LimitTorque = m_spel.GetLimitTorque(1);
```

GetLimZ 方法, Spel 类

描述

返回当前的 LimZ 设置。

语法

Function **GetLimZ** () As Single

返回值

含有 LimZ 值的实际值。

另见

LimZ, Jump

GetLimZ 示例

VB 例:

```
saveLimZ = m_spel.GetLimZ()  
m_spel.LimZ(-22)
```

C# 例:

```
saveLimZ = m_spel.GetLimZ();  
m_spel.LimZ(-22);
```


GetPoint 方法, Spel 类

描述

检索机器人点的坐标数据。

语法

Function **GetPoint** (*PointNumber* As Integer) As SpelPoint
Function **GetPoint** (*PointName* As String) As SpelPoint

参数

PointNumber 当前机器人控制器点内存中的点的整数表达式。

PointName 字符串表达式。可以是点标签, “Pxxx”、“P*”或“*”。

另见

SetPoint

GetPoint 示例**VB 例:**

```
Dim pt As SpelPoint  
pt = m_spel.GetPoint("P*")  
pt.X = 25.0  
m_spel.Go(pt)
```

C# 例:

```
SpelPoint pt;  
pt = m_spel.GetPoint("P0");  
pt.X = 25.0;  
m_spel.Go(pt);
```

GetRealTorque 方法, Spel 类**描述**

返回当前机器人指定关节的扭矩。

语法

Function **GetRealTorque** (*JointNumber* As Integer) As Double

参数

JointNumber 所需关节的整数表达式。

返回值

表示当前功率模式的指定关节最大扭矩比的 0 至 1 的 Double 值。

另见

GetLimitTorque, GetRobotPos

GetRealTorque 示例**VB 例:**

```
Dim j1Torque As Double  
j1Torque = m_spel.GetRealTorque(1)
```

C# 例:

```
double j1Torque;  
j1Torque = m_spel.GetRealTorque(1);
```

GetRobotInfo 方法, Spel 类

描述

返回机器人信息。

语法

Function **GetRobotInfo** (*RobotNumber* As Integer) As SpelRobotInfo

参数

*RobotNumber*指定机器人编号

返回值

SpelRobotInfo

另见

GetControllerInfo, GetTaskInfo

GetRobotInfo 示例**VB 例::**

```
Dim info As SpelRobotInfo
Dim msg As String

info = m_spel.GetRobotInfo(1)
msg = "Robot Model: " & info.RobotModel & vbCrLf _
      & "Robot Serial: " & info.RobotSerial
MsgBox(msg)
```

C# 例:

```
SpelRobotInfo info;
string msg;

info = m_spel.GetRobotInfo(1);
msg = "Robot Model: " + info.RobotModel +
      "\r\n Robot Serial: " + info.RobotSerial;
MessageBox.Show(msg);
```

GetRobotPos 方法, Spel 类

描述

返回当前的机器人位置。

语法

Function **GetRobotPos**(*PosType* As SpelRobotPosType, *Arm* As Integer, *Tool* As Integer, *Local* As Integer) As **Single**()

参数

<i>PosType</i>	指定待返回的位置数据类型。
<i>Arm</i>	指定机器人机械臂的整数表达式。
<i>Tool</i>	指定机器人工具的整数表达式。
<i>Local</i>	指定机器人本地的整数表达式。

返回值

含有 9 个元素的单数据类型数组。返回的数据取决于指定的 *PosType*。

World X, Y, Z, U, V, W, R, S, T

Joint J1, J2, J3, J4, J5, J6, J7, J8, J9

Pulse Pls1, Pls2, Pls3, Pls4, Pls5, Pls6, Pls7, Pls8, Pls9

另见

GetPoint

GetRobotPos 示例**VB 例:**

```
Dim values() As Single
values = m_spel.GetRobotPos(SpelRobotPosType.World, 0, 0, 0)
```

C# 例:

```
float[] values;
values = m_spel.GetRobotPos(SpelRobotPosType.World, 0, 0, 0);
```

GetSpeed 方法, Spel 类

描述

返回当前机器人的三个速度设置之一。

语法

Function **GetSpeed** (*ParamNumber* As Integer) As Integer

参数

ParamNumber 评估一个以下数值的整数表达式。

1: PTP 动作速度

2: Jump 起始速度

3: Jump 结束速度

返回值

1-100 之间的整数表达式。

另见

Speed

GetSpeed 示例**VB 例:**

```
Dim x As Integer  
x = m_spel.GetSpeed(1)
```

C# 例:

```
int ptpSpeed;  
ptpSpeed = m_spel.GetSpeed(1);
```

GetTaskInfo 方法, Spel 类

描述

返回任务信息。

语法

Function **GetTaskInfo** (*TaskName* As String) As SpelTaskInfo

Function **GetTaskInfo** (*TaskNumber* As Integer) As SpelTaskInfo

参数

TaskName 指定任务名称

TaskNumber 指定任务编号

返回值

SpelTaskInfo

另见

GetControllerInfo, GetRobotInfo

GetTaskInfo 示例**VB 例:**

```
Dim info As SpelTaskInfo
Dim msg As String

info = m_spel.GetTaskInfo(1)
msg = "Task Name: " & info.TaskName & vbCrLf _
      & "Task State: " & info.TaskState
MsgBox(msg)
```

C# 例:

```
SpelTaskInfo info;
string msg;

info = m_spel.GetTaskInfo(1);
msg= "Task Name: " + info.TaskName +
     "\r\n Task State: " + info.TaskState;
MessageBox.Show(msg);
```

GetTool 方法, Spel 类**描述**

返回当前机器人的当前工具编号。

语法

Function **GetTool** () As Integer

返回值

含有当前工具编号的整数。

另见

Arm, TLSet, Tool

GetTool 示例**VB 例:**

```
saveTool = m_spel.GetTool()  
m_spel.Tool(2)
```

C# 例:

```
saveTool = m_spel.GetTool();  
m_spel.Tool(2);
```

GetVar 方法, Spel 类

描述

返回控制器中 SPEL+ 全局保留变量的值。

语法

Function **GetVar**(*VarName* As String) As Object

参数

VarName SPEL+ 全局保留变量的名称。
对于数组, 可返回整个数组或仅返回一个元素。

返回值

返回数据类型由 SPEL+ 变量类型决定的数值。

备注

可使用 **GetVar** 检索控制器当前项目中所有全局保留变量的数值。在能够检索数值之前, 必须已成功构建项目。

如果需要检索整个数组, 则在 *VarName* 中提供数组名称。若要检索数组中的一个元素, 在 *VarName* 中提供下标。

另见

SetVar

GetVar 示例

在 SPEL+ 项目中, 定义变量:

```
Global Preserve Integer g_myIntVar
Global Preserve Real g_myRealArray(10)
Global Preserve String g_myStringVar$
Function main
    ...
End
```

在 Visual Basic 项目中:

由于 **g_myIntVar** 定义为整数, 因此用于检索 **g_myIntVar** 数值的 Visual Basic 变量必须定义为整数。对于 **g_myRealArray**, Visual Basic 变量必须定义为 Single 变量的数组。

```
Dim myIntVar As Integer
Dim myRealArray() As Single
Dim myStringVar As String

myIntVar = m_spel.GetVar("g_myIntVar")
myRealArray = m_spel.GetVar("g_myRealArray")
myStringVar = m_spel.GetVar("g_myStringVar$")
```


在 C# 项目中:

由于 `g_myIntVar` 定义为整数, 因此用于检索 `g_myIntVar` 数值的 C# 变量必须定义为整数。对于 `g_myRealArray`, C# 变量必须定义为 Float 变量的数组。

```
int myIntVar;
float[] myRealArray;
string myStringVar;

myIntVar = m_spel.GetVar("g_myIntVar");
myRealArray = m_spel.GetVar("g_myRealArray");
myStringVar = m_spel.GetVar("g_myStringVar$");
```

Go 方法, Spel 类

描述

以 PTP 的形式将机械臂从当前位置移至指定点或 XY 位置。**GO** 指令可同时移动任意组合的机器人轴。

语法

```
Sub Go (PointNumber As Integer)
Sub Go (Point As SpelPoint)
Sub Go (Point As SpelPoint, AttribExpr As String)
Sub Go (PointExpr As String)
```

参数

每个语法具有一个指定机械臂在 Go 动作期间所移至端点的参数。此参数为 PTP 动作结束时的最终位置。

PointNumber 通过对当前机器人控制器点内存中之前示教的点使用点编号来指定端点。

Point 通过使用 SpelPoint 数据类型指定端点。

AttribExpr 通过使用字符串表达式指定端点属性。

PointExpr 通过使用字符串表达式指定端点。

另见

Accel, Speed
Arc, Arc3, CVMove, Jump, Jump3, Jump3CP, Move
BGo, BMove, TGo, TMove
Arch, CP, Sense, Till

Go 示例**VB 例:**

```
' 使用点编号指定
m_spel.Tool(1)
m_spel.Go(100)

' 使用 SpelPoint 指定
Dim pt As SpelPoint
pt = m_spel.GetPoint("P*")
pt.X = 125.5
m_spel.Go(pt)

' 使用点表达式指定
m_spel.Go("P0 /L /2")
m_spel.Go("P1 :Z(-20)")

' 使用并行处理
m_spel.Go("P1 !D50; On 1; D90; Off 1!")

' 在点标签中指定
m_spel.Go("pick")
```

C# 例:

```
// 使用点编号指定
m_spel.Tool(1);
m_spel.Go(100);

// 使用 SpelPoint 指定
SpelPoint pt;
pt = m_spel.GetPoint("P0");
pt.X = 125.5;
m_spel.Go(pt);

// 使用点表达式指定
m_spel.Go("P0 /L /2");
m_spel.Go("P1 :Z(-20)");

// 使用并行处理
m_spel.Go("P1 !D50; On 1; D90; Off 1!");

// 在点标签中指定
m_spel.Go("pick");
```

Halt 方法, Spel 类

描述

暂停指定任务的执行。

语法

Sub **Halt** (*TaskNumber* As Integer)

Sub **Halt** (*TaskName* As String)

参数

TaskNumber 待暂停任务的任务号。
任务号范围为 1 至 32。

TaskName 含有待暂停任务名称的字符串表达式。

另见

Resume, Xqt

Halt 示例

VB 例:

```
m_spel.Halt(3)
```

C# 例:

```
m_spel.Halt(3);
```

Here 方法, Spel 类

描述

示教当前位置的点。

语法

```
Sub Here (PointNumber As Integer)  
Sub Here (PointName As String)
```

参数

PointNumber 当前机器人点内存中的点的整数表达式。可使用以 0 开头的任何有效点编号。

PointName 点标签的字符串表达式。

另见

SetPoint

Here 示例**VB 例:**

```
m_spel.Here ("P20")
```

C# 例:

```
m_spel.Here ("P20");
```

HideWindow 方法, Spel 类

描述

隐藏之前用 ShowWindow 显示的 EPSON RC+ 7.0 窗口。

语法

Sub **HideWindow** (WindowID As SpelWindows)

参数

WindowID 待隐藏 EPSON RC+ 7.0 窗口的 ID。

另见

RunDialog, ShowWindow

HideWindow 示例**VB 例:**

```
Sub btnHideIOMonitor_Click _  
    ByVal sender As System.Object, _  
    ByVal e As System.EventArgs) _  
    Handles btnHideIOMonitor.Click  
  
    m_spel.HideWindow(RCAPINet.SpelWindows.IOMonitor)  
End Sub
```

C# 例:

```
void btnHideIOMonitor_Click(object sender, EventArgs e)  
{  
    m_spel.HideWindow(RCAPINet.SpelWindows.IOMonitor);  
}
```

Home 方法, Spel 类**描述**

将机器人机械臂移至用户定义的并用 **HomeSet** 方法设置的起始点位置。

语法

```
Sub Home ()
```

另见

HomeSet, MCal

Home 示例**VB 例:**

```
With m_spel  
    .MotorsOn = True  
    .Home ()  
End With
```

C# 例:

```
m_spel.MotorsOn = true;  
m_spel.Home ();
```

Hofs 方法, Spel 类

描述

读取或设置从每个关节的编码器原点到软件原点的校正脉冲值。

语法

```
Function Hofs (JointNumber As Integer) As Integer
Sub Hofs (J1Pulses As Integer, J2Pulses As Integer, J3Pulses As Integer, J4Pulses As Integer, [J5Pulses As Integer], [J6Pulses As Integer], [J7Pulses As Integer], [J8Pulses As Integer], [J9Pulses As Integer])
```

参数

J1Pulses – J9Pulses 以整数显示第 1~9 关节的脉冲值
J5Pulses – J9Pulses 可以省略

返回值

返回指定关节的校正脉冲值 (整数值、单位: 脉冲)。

备注

本方法可用于显示或设置原点校正脉冲值。本方法可设置从编码器原点(Z相)到机械原点的偏移值。

机器人运动控制基于, 安装在各关节上的编码器的原点进行的。但是编码器原点和机器人机械原点不一定完全匹配。为了在软件上将编码器原点调整为与机械原点完全匹配, 则可适用本方法设置校正脉冲值。

注意**如非必须情况, 请勿更改 Hofs 值**

Hofs 的值是在工厂出货前, 进行精密设置的。如随意修改该值, 可能会造成定位错误并使机器人发生意外动作, 非常危险。如非必须情况, 请勿更改该值。

自动计算 Hofs 值

要自动计算 Hofs 的值, 请将机械臂移动到您要校准的位置, 并运行 Calib。这样, 控制器将根据 CalPls 脉冲值和校准位置的脉冲值, 自动计算 Hofs 值。

保存并恢复 Hofs 值

可以通过在菜单中选择[System Configuration]-[Robot]-[Calibration], 然后选择保存和读取功能, 保存或恢复 Hofs 的值。

另见

CalPls, Home, Hordr, Mcal

Hofs 示例**VB 例:**

```
Dim val As Integer
val = m_spel.Hofs(1)
```

C# 例:

```
int val;
val = m_spel.Hofs(1);
```


HomeSet 方法, Spel 类

描述

指定 Home 方法使用的位置。

语法

```
Sub HomeSet ( J1Pulses As Integer, J2Pulses As Integer, J3Pulses As Integer,
              J4Pulses As Integer, J5Pulses As Integer, J6Pulses As Integer
              [, J7Pulses As Integer] [, J8Pulses As Integer] [, J9Pulses As Integer] )
```

参数

J1Pulses - *J9Pulses* 每个关节的 Home 位置编码器脉冲值。
关节 7、8 和 9 是可选的。

另见

Home, MCal

HomeSet 示例**VB 例:**

```
' 将当前位置设为 Home 位置
With m_spel
    .HomeSet(.Pls(1), .Pls(2), .Pls(3), .Pls(4), 0, 0)
End With
```

C# 例:

```
//将当前位置设为 Home 位置
m_spel.HomeSet(m_spel.Pls(1), m_spel.Pls(2), m_spel.Pls(3),
               m_spel.Pls(4), 0 ,0);
```

Hordr 方法, Spel 类

描述

指定所有轴返回至各自 HOME 位置的顺序。

语法

```
Sub Hordr ( Home1 As Integer, Home2 As Integer, Home3 As Integer, Home4 As Integer,  
           Home5 As Integer, Home6 As Integer [, Home7 As Integer]  
           [, Home8 As Integer] [, Home9 As Integer] )
```

参数

Home 1 - 9 通知各轴应在 Home 过程各步骤返回起始点的位模式。
在步骤 1 期间, 0 至全部轴之间任意数量的轴均可在第一步过程中
返回至起始点位置。
在指定了 R、S 或 T 轴时, 可指定 *Home 7 - 9*。

另见

Home, HomeSet, Mcordr

Hordr 示例**VB 例:**

```
m_spel.Hordr(2, 13, 0, 0, 0, 0)
```

C# 例:

```
m_spel.Hordr(2, 13, 0, 0, 0, 0);
```

Hour 方法, Spel 类

描述

返回累计的系统运行小时数。

语法

Function **Hour** () As Single

返回值

表示时间的整数表达式。

Hour 示例

VB 例:

```
Dim hoursRunning As Single  
hoursRunning = m_spel.Hour()
```

C# 例:

```
float hoursRunning;  
hoursRunning = m_spel.Hour();
```

ImportPoints 方法, Spel 类

描述

将点文件导入当前机器人的当前项目。

语法

```
Sub ImportPoints (SourcePath As String, ProjectFileName As String [, RobotNumber  
As Integer])
```

参数

- SourcePath* 含有待导入当前项目的特定路径和文件的字符串表达式。扩展名必须为 .pts。
- ProjectFileName* 含有待导入至当前机器人或在提供 *RobotNumber* 时指定机器人的当前项目的特定文件的字符表达式。扩展名必须为 .pts。
- RobotNumber* 可选。将使用点文件的机器人的整数表达式。指定为 0 可使文件成为公共点文件。

另见

SavePoints

ImportPoints 示例**VB 例:**

```
With m_spel  
    .ImportPoints ("c:\mypoints\model1.pts", "robot1.pts")  
End With
```

C# 例:

```
m_spel.ImportPoints (@"c:\mypoints\model1.pts",  
"robot1.pts");
```

In 方法, Spel 类

描述

返回指定输入端口的状态。每个端口含有 8 个输入位(一个字节)。

语法

```
Function In (PortNumber As Integer) As Integer  
Function In (Label As String) As Integer
```

参数

PortNumber 表示其中一个输入端口的整数表达式。
每个端口含有 8 个输入位(一个字节)。

Label 含有输入字节标签的字符串表达式。

返回值

0 至 255 之间表示输入端口状态的整数。

另见

InBCD, Out, OpBCD, Sw

In 示例**VB 例:**

```
Dim port1Value As Integer  
port1Value = m_spel.In(1)
```

C# 例:

```
int port1Value;  
port1Value = m_spel.In(1);
```

InBCD 方法, Spel 类

描述

使用 BCD 格式返回 8 个输入的输入状态。(二进制编码的十进制)

语法

```
Function InBCD (PortNumber As Integer) As Integer
```

```
Function InBCD (Label As String) As Integer
```

参数

PortNumber 表示其中一个输入端口的整数表达式。

Label 含有输入字节标签的字符串表达式。

返回值

0 至 9 之间表示输入端口状态的整数。

另见

In, Out, OpBCD, Sw

InBCD 示例**VB 例:**

```
Dim port1Value As Integer  
port1Value = m_spel.InBCD(1)
```

C# 例:

```
int port1Value;  
port1Value = m_spel.InBCD(1);
```

Inertia 方法, Spel 类

描述

指定当前机器人的负载惯性和偏心距。

语法

Sub **Inertia** (*LoadInertia* As Single, *Eccentricity* As Single)

参数

<i>LoadInertia</i>	指定包含夹具末端和部件的夹具末端关节中心附近的总装载惯性的实数表达式, 单位为 kgm^2 。
<i>Eccentricity</i>	指定包含夹具末端和部件的夹具末端关节中心附近的偏心距的实数表达式, 单位为 mm。

另见

Weight

Inertia 示例**VB 例:**

```
m_spel.Inertia(0.02, 1.0)
```

C# 例:

```
m_spel.Inertia(0.02, 1.0);
```

Initialize 方法, Spel 类

描述

初始化 Spel 类实例。

语法

Sub **Initialize** ()

备注

通常情况下，当执行第一个方法时会自动初始化 Spel 类实例。EPSON RC+ 7.0 载入内存时，初始化会需要几秒的时间。因此，在一些情况下，可能希望在启动时首先在应用中调用 initialize。

根据 ServerInstance，启动 RC+作为服务器进程。每个 ServerInstance 对应一个控制器和一个项目。若使用 ServerInstance，请在执行初始化之前完成设置。

另见

Connect, Disconnect, ServerInstance

Initialize 示例**VB 例:**

```
m_spel.Initiialize ()
```

C# 例:

```
m_spel.Initialize ();
```


InReal 方法, Spel 类

描述

将 2 个字节(32 位)输入数据, 读取为 32 位浮点数据 (符合 IEEE754 标准)。

语法

Function **InReal** (*PortNumber* As Integer) As Single

参数

PortNumber 显示输入端口的整数

返回值

将输入端口的状态返回为 32 位浮点数据 (符合 IEEE754 标准)。

另见

In, InBCD, InW, Out, OutW, OutReal

InReal 示例**VB 例::**

```
Dim val As Single  
val = m_spel.InReal(32)
```

C# 例:

```
float val;  
val = m_spel.InReal(32);
```

InsideBox 方法, Spel 类

描述

返回结束检查区域的检查状态。

语法

Function **InsideBox** (*BoxNumber* As Integer) As Boolean

参数

BoxNumber 1 至 15 之间表示所返回状态的结束检查区域的整数表达式。

返回值

如果机器人夹具末端位于指定框内，则返回 **True**，否则返回 **False**。

另见

Box, InsidePlane

InsideBox 示例

VB 例:

```
Dim isInside As Boolean  
isInside = m_spel.InsideBox(1)
```

C# 例:

```
bool isInside;  
isInside = m_spel.InsideBox(1);
```

InsidePlane 方法, Spel 类

描述

返回结束检查平面的检查状态。

语法

Function **InsidePlane** (*PlaneNumber* As Integer) As Boolean

参数

PlaneNumber 1 至 15 之间表示所返回状态的结束检查平面的整数表达式。

返回值

如果机器人夹具末端位于指定框内，则返回 **True**，否则返回 **False**。

另见

InsideBox, Plane

InsidePlane 示例**VB 例:**

```
Dim isInside As Boolean  
isInside = m_spel.InsidePlane(1)
```

C# 例:

```
bool isInside;  
isInside = m_spel.InsidePlane(1);
```

InW 方法, Spel 类

描述

返回指定输入字端口的状态。每个字端口含有 16 个输入位。

语法

```
Function InW (PortNumber As Integer) As Integer  
Function InW (Label As String) As Integer
```

参数

PortNumber 表示输入端口的整数。

Label 含有输入字标签的字符串表达式。

返回值

0 至 65535 之间表示输入端口的整数值

另见

In, InBCD, Out, OpBCD, Sw

InW 示例**VB 例:**

```
Dim data As Integer  
data = m_spel.InW(0)
```

C# 例:

```
int data;  
data = m_spel.InW(0);
```

IsOptionActive 方法, Spel 类**描述**

返回软件选件的状态。

语法

Function **IsOptionActive** (*option* As SpelOptions) As Boolean

参数

option 表示选件编号的整数值。

返回值

False – 无效

True – 有效

另见

GetControllerInfo

IsOptionActive 示例**VB 例:**

```
Dim ret As Boolean
ret = m_spel.IsOptionActive(SpelOptions.FieldbusMaster)
```

C# 例:

```
bool ret;
ret = m_spel.IsOptionActive(SpelOptions.FieldbusMaster);
```

JRange 方法, Spel 类

描述

以脉冲形式定义指定机器人关节的允许工作范围。

语法

```
Sub JRange (JointNumber As Integer, LowerLimitPulses As Integer, UpperLimitPulses  
As Integer)
```

参数

JointNumber 1-9 之间表示将指定的 **JRange** 关节的整数值。

LowerLimitPulses 表示指定关节下限范围的编码器脉冲计数位置的整数值。

UpperLimitPulses 表示指定关节上限范围的编码器脉冲计数位置的整数值。

另见

XYLim

JRange 示例**VB 例:**

```
m_spel.JRange(1, -30000, 30000)
```

C# 例:

```
m_spel.JRange(1, -30000, 30000);
```

JS 方法, Spel 类

描述

Jump Sense 会检测机械臂在完成 JUMP 指令(使用 SENSE 输入)之前是否已停止或机械臂是否已完成 JUMP 移动。

语法

Function **JS** () As Boolean

返回值

如果在动作期间检测到 SENSE 输入, 则返回 True, 否则返回 False。

另见

Jump, Jump3, Jump3CP, Sense, Till

JS 示例**VB 例:**

```
With m_spel
    .Sense("Sw(1) = On")
    .Jump("P1 Sense")
    stoppedOnSense = .JS()
End With
```

C# 例:

```
m_spel.Sense("Sw(1) = On");
m_spel.Jump("P1 Sense");
stoppedOnSense = m_spel.JS();
```

JTran 方法, Spel 类

描述

执行相对关节移动。

语法

Sub **JTran** (*JointNumber* As Integer, *Distance* As Single)

参数

JointNumber 待移动的特定关节。

Distance 待移动的距离。对于旋转关节，单位为度；对于线性关节，单位为毫米。

另见

PTran, Pulse

JTran 示例

VB 例:

’ 移动关节 1，正方向 45 度。

```
m_spel.JTran(1, 45.0)
```

C# 例:

// 移动关节 1，正方向 45 度。

```
m_spel.JTran(1, 45.0);
```


Jump 方法, Spel 类

描述

使用 PTP 动作将机械臂从当前位置移至指定点，首先垂直上移，然后水平移动，最后再垂直下移到达最终目标点。

语法

```
Sub Jump (PointNumber As Integer)
Sub Jump (Point As SpelPoint)
Sub Jump (Point As SpelPoint, AttribExpr As String)
Sub Jump (PointExpr As String)
```

参数

每个语法具有一个指定机械臂在 **Jump** 动作期间所移至端点的参数。此参数为 PTP 动作结束时的最终位置。

PointNumber 通过对当前机器人控制器点内存中之前示教的点使用点编号来指定端点。

Point 通过使用 **SpelPoint** 数据类型指定端点。

AttribExpr 通过使用字符串表达式指定端点属性。

PointExpr 通过使用字符串表达式指定端点。

另见

Accel, Speed
Arc, Arc3, CVMove, Go, Jump3, Jump3CP, Move
BGo, BMove, TGo, TMove
Arch, CP, Sense, Till

Jump 示例**VB 例:**

```
' 使用点编号指定
m_spel.Tool(1)
m_spel.Jump(100)

' 使用 SpelPoint 指定
Dim pt As SpelPoint
pt = m_spel.GetPoint("P*")
pt.X = 125.5
m_spel.Jump(pt)

' 使用点表达式指定
m_spel.Jump("P0 /L /2")
m_spel.Jump("P1 :Z(-20)")
m_spel.Jump("P1 C0")
m_spel.Jump("P1 C0 LimZ -10")
m_spel.Jump("P1 C0 Sense Sw(0)=On")

' 使用并行处理
m_spel.Jump("P1 !D50; On 1; D90; Off 1!")

' 在点标签中指定
m_spel.Jump("pick")
```

C# 例:

```
// 使用点编号指定
m_spel.Tool(1);
m_spel.Jump(100);

// 使用 SpelPoint 指定
SpelPoint pt;
pt = m_spel.GetPoint("P0");
pt.X = 125.5;
m_spel.Jump(pt);

// 使用点表达式指定
m_spel.Jump("P0 /L /2");
m_spel.Jump("P1 :Z(-20)");
m_spel.Jump("P1 C0");
m_spel.Jump("P1 C0 LimZ -10");
m_spel.Jump("P1 C0 Sense Sw(0)=On");

// 使用并行处理
m_spel.Jump("P1 !D50; On 1; D90; Off 1!");

// 在点标签中指定
m_spel.Jump("pick");
```

Jump3 方法, Spel 类

描述

使用两个 CP 动作和一个 PTP 动作的结合通过 3 维闸极动作。机器人移至起始点，然后移至结束点，最后移至目标点。

语法

Sub **Jump3** (*DepartPoint* As Integer, *ApproPoint* As Integer, *DestPoint* As Integer)

Sub **Jump3** (*DepartPoint* As SpelPoint, *ApproPoint* As SpelPoint, *DestPoint* As SpelPoint)

Sub **Jump3** (*DepartPoint* As String, *ApproPoint* As String, *DestPoint* As String)

参数

DepartPoint 使用点编号或字符串点表达式的当前位置上方的起始点。

ApproPoint 使用点编号或字符串点表达式的目标位置上方的结束点。

DestPoint 使用点编号或字符串点表达式动作的目标位置。

另见

Accel, AccelR, AccelS, Speed, SpeedR, SpeedS
Arc, Arc3, CVMove, Go, Jump, Jump3CP, Move
BGo, BMove, TGo, TMove
Arch, CP, Sense, Till

Jump3 示例**VB 例:**

‣ 使用点编号指定

```
m_spel.Tool(1)
m_spel.Jump3(1, 2, 3)
```

‣ 使用 SpelPoint 指定

```
Dim pd As SpelPoint
Dim pa As SpelPoint
Dim pt As SpelPoint
pd = m_spel.GetPoint("P*")
pd.Z = 125.5
pa = m_spel.GetPoint("P2")
pa.Z = 125.5
pt = m_spel.GetPoint("P2")
m_spel.Jump3(pd, pa, pt)
```

‣ 使用表达式指定

```
m_spel.Jump3("P1", "P2", "P3 C0")
m_spel.Jump3("P1", "P2", "P3 C0 Sense Sw(0)=On")
m_spel.Jump3("P0 -TLZ(10), P1 -TLZ(10), P1")
```

‣ 使用并行处理

```
m_spel.Jump3("P1", "P2", "P3 !D50; On 1; D90; Off 1!")
```

‣ 在点标签中指定

```
m_spel.Jump3("depart", "approach", "place")
```

C# 例:

```
// 使用点编号指定
m_spel.Tool(1);
m_spel.Jump3(1, 2, 3);

// 使用 SpelPoint 指定
SpelPoint pd, pa, pt;
pd = m_spel.GetPoint("P1");
pd.Z = 125.5;
pa = m_spel.GetPoint("P2");
pa.Z = 125.5;
pt = m_spel.GetPoint("P2");
m_spel.Jump3(pd, pa, pt);

// 使用表达式指定
m_spel.Jump3("P1", "P2", "P3 C0");
m_spel.Jump3("P1", "P2", "P3 C0 Sense Sw(0)=On");
m_spel.Jump3("P0 -TLZ(10), P1 -TLZ(10), P1");

// 使用并行处理
m_spel.Jump3("P1", "P2", "P3 !D50; On 1; D90; Off 1!");

// 在点标签中指定
m_spel.Jump3("depart", "approach", "place");
```

Jump3CP 方法, Spel 类

描述

使用三个 CP 动作的组合通过 3 维闸极动作。

语法

Sub **Jump3CP** (*DepartPoint* As Integer, *ApproPoint* As Integer, *DestPoint* As Integer)

Sub **Jump3CP** (*DepartPoint* As SpelPoint, *ApproPoint* As SpelPoint, *DestPoint* As SpelPoint)

Sub **Jump3CP** (*DepartPoint* As String, *ApproPoint* As String, *DestPoint* As String)

参数

DepartPoint 使用点编号或字符串点表达式的当前位置上方的起始点。

ApproPoint 使用点编号或字符串点表达式的目标位置上方的结束点。

DestPoint 使用点编号或字符串点表达式动作的目标位置。

另见

AccelR, AccelS, SpeedR, SpeedS

Arc, Arc3, CVMove, Go, Jump, Jump3, Move

BGo, BMove, TGo, TMove

Arch, CP, Sense, Till

Jump3CP 示例**VB 例:**

‣ 使用点编号指定

```
m_spel.Tool(1)
m_spel.Jump3CP(1, 2, 3)
```

‣ 使用 SpelPoint 指定

```
Dim pd As SpelPoint
Dim pa As SpelPoint
Dim pt As SpelPoint
pd = m_spel.GetPoint("P*")
pd.Z = 125.5
pa = m_spel.GetPoint("P2")
pa.Z = 125.5
pt = m_spel.GetPoint("P2")
m_spel.Jump3CP(pd, pa, pt)
```

‣ 使用点表达式指定

```
m_spel.Jump3CP("P1", "P2", "P3 C0")
m_spel.Jump3CP("P1", "P2", "P3 C0 Sense Sw(0)=On")
m_spel.Jump3CP("P0 -TLZ(10), P1 -TLZ(10), P1")
```

‣ 使用并行处理

```
m_spel.Jump3CP("P1", "P2", "P3 !D50; On 1; D90; Off 1!")
```

‣ 在点标签中指定

```
m_spel.Jump3CP("depart", "approch", "place")
```

C# 例:

```
// 使用点编号指定
m_spel.Tool(1);
m_spel.Jump3CP(1, 2, 3);

// 使用 SpelPoint 指定
SpelPoint pd, pa, pt;
pd = m_spel.GetPoint("P0");
pd.Z = 125.5;
pa = m_spel.GetPoint("P2");
pa.Z = 125.5;
pt = m_spel.GetPoint("P2");
m_spel.Jump3CP(pd, pa, pt);

// 使用点表达式指定
m_spel.Jump3CP("P1", "P2", "P3 C0");
m_spel.Jump3CP("P1", "P2", "P3 C0 Sense Sw(0)=On");
m_spel.Jump3CP("P0 -TLZ(10), P1 -TLZ(10), P1");

// 使用并行处理
m_spel.Jump3CP("P1", "P2", "P3 !D50; On 1; D90; Off 1!");

// 在点标签中指定
m_spel.Jump3CP("depart", "approch", "place");
```

LimitTorque 方法, Spel 类

描述

设置当前机器人高功率模式下的上限扭矩。

语法

Sub **LimitTorque** (*AllJointsMax* As Integer)

Sub **LimitTorque** (*J1Max* As Integer, *J2Max* As Integer, *J3Max* As Integer, *J4Max* As Integer, *J5Max* As Integer, *J6Max* As Integer)

参数

AllJointsMax 高功率模式下所需的所有关节上限扭矩整数表达式。

J1Max - *J6Max* 高功率模式下所需的每个关节上限扭矩整数表达式。

返回值

表示指定关节限制扭矩设置的 1 至 9 整数值。

另见

GetRealTorque, GetRobotPos

LimitTorque 示例**VB 例:**

```
Dim j1LimitTorque As Integer
j1LimitTorque = m_spel.LimitTorque(1)
```

C# 例:

```
int j1LimitTorque1
j1LimitTorque = m_spel.LimitTorque(1);
```

LimZ 方法, Spel 类

描述

设置 JUMP 命令的 Z 轴高度默认值。

语法

Sub **LimZ** (*ZLimit* As Single)

参数

ZLimit Z 轴可移动范围内的坐标值。

另见

Jump

LimZ 示例

VB 例:

```
saveLimZ = m_spel.GetLimZ()  
m_spel.LimZ(-22)
```

C# 例:

```
saveLimZ = m_spel.GetLimZ();  
m_spel.LimZ(-22);
```


LoadPoints 方法, Spel 类**描述**

将 SPEL+ 点文件载入当前机器人的控制器点内存。

语法

Sub **LoadPoints** (*FileName* As String [, *Merge* As Boolean])

参数

FileName 当前项目中的有效点文件。

Merge 可选。将当前点合并至指定点文件时设置。

另见

ImportPoints, SavePoints

LoadPoints 示例**VB 例:**

```
With m_spel  
    .LoadPoints ("part1.pts")  
End With
```

C# 例:

```
m_spel.LoadPoints("part1.pts");
```

Local 方法, Spel 类

描述

定义本地坐标系。

语法

```
Sub Local (LocalNumber As Integer, OriginPoint As SpelPoint, [XAxisPoint As SpelPoint], [YAxisPoint As SpelPoint])
Sub Local (LocalNumber As Integer, LocalPoint1 As Integer, BasePoint1 As Integer, LocalPoint2 As Integer, BasePoint2 As Integer)
Sub Local (LocalNumber As Integer, LocalPoint1 As String, BasePoint1 As String, LocalPoint2 As String, BasePoint2 As String)
```

参数

<i>LocalNumber</i>	本地坐标系编号。总共可定义 15 个本地坐标系(采用整数值 1 至 15)。
<i>OriginPoint</i>	本地坐标系原点的 SpelPoint 变量。
<i>XAxisPoint</i>	可选。本地坐标系 X 轴上一点的 SpelPoint 变量。
<i>YAxisPoint</i>	可选。本地坐标系 Y 轴上一点的 SpelPoint 变量。
<i>LocalPoint1, LocalPoint2</i>	通过整数或字符串指定, 表示本地坐标系的点数据。
<i>BasePoint1, BasePoint2</i>	通过整数或字符串指定, 表示 Base 坐标系的点数据。

另见

Base

Local 示例**VB 例:**

```
Dim originPoint As New SpelPoint
originPoint.X = 100
originPoint.Y = 50
m_spel.Local(1, originPoint)
```

C# 例:

```
SpelPoint originPoint = new SpelPoint();
originPoint.X = 100;
originPoint.Y = 50;
m_spel.Local(1, originPoint);
```

LocalClr 方法, Spel 类

描述

清除为当前机器人定义的 Local。

语法

```
Sub LocalClr (LocalNumber As Integer)
```

参数

LocalNumber 表示 15 个本地中(采用整数 1 至 15)待清除(取消定义)的整数表达式。

另见

Local, LocalDef

LocalClr 示例**VB 例:**

```
m_spel.LocalClr(1)
```

C# 例:

```
m_spel.LocalClr(1);
```

LocalDef 方法, Spel 类

描述

返回本地定义状态。

语法

Function **LocalDef** (*LocalNumber* As Integer) As Boolean

参数

LocalNumber 表示所返回状态的本地坐标系的整数表达式(1-15)。

返回值

如果已定义指定本地, 则返回 True, 否则返回 False。

另见

Local, LocalClr

LocalDef 示例**VB 例:**

```
Dim localExists As Boolean  
localExists = m_spel.LocalDef(1)
```

C# 例:

```
bool localExists;  
localExists = m_spel.LocalDef(1);
```

Login 方法, Spel 类

描述

作为另一个用户登录到 EPSON RC+ 7.0。

语法

Sub **LogIn** (*LoginID* As String, *Password* As String)

参数

LoginID 含有用户登录 ID 的字符串表达式。

Password 含有用户密码的字符串表达式。

备注

您可在应用中使用 EPSON RC+ 7.0 安全。例如：可显示允许不同用户登录系统的菜单。每类用户具有各自的安全权限。有关安全的更多详细信息，请参阅 *EPSON RC+ 7.0 用户指南*。

如果启用了安全却并未执行 **LogIn**，则会作为访客用户登录您的 Visual Basic 应用。如果 EPSON RC+ 7.0 中的 Auto LogIn 启用，则会作为当前的 Windows 用户自动登录应用(如果 EPSON RC+ 7.0 中已配置此类用户)。

另见

GetCurrentUser

Login 示例**VB 例:**

```
With m_spel
    .Project =
    "c:\EpsonRC70\projects\myproject\myproject.sprj"
    .LogIn("operator", "oprpass")
End With
```

C# 例:

```
m_spel.Project =
@"c:\EpsonRC70\projects\myproject\myproject.sprj";
m_spel.LogIn("operator", "oprpass");
```

MCal 方法, Spel 类

描述

通过增量编码器对机器人执行机器校准。

语法

Sub **MCal** ()

另见

MCalComplete, MotorsOn

MCal 示例

VB 例:

```
If Not m_spel.MCalComplete() Then  
    m_spel.MCal ()  
End If
```

C# 例:

```
if (!m_spel.MCalComplete())  
    m_spel.MCal ();
```

MCalComplete 方法, Spel 类**描述**

如果已成功完成 MCal, 则返回 True。

语法

Function **MCalComplete** () As Boolean

返回值

如果 MCal 已完成, 则返回 True, 否则返回 False。

另见

MCal

MCalComplete 示例**VB 例:**

```
If m_spel.MCalComplete() Then  
    lblStatus.Text = "MCal Complete"  
Else  
    lblStatus.Text = "MCal Not Complete"  
End If
```

C# 例:

```
if (m_spel.MCalComplete())  
    lblStatus.Text = "MCal Complete";  
else  
    lblStatus.Text = "MCal Not Complete";
```

Mcordr 方法, Spel 类

描述

指定机器校准 MCal 所需的移动轴顺序。

语法

```
Sub Mcordr ( Step1 As Integer, Step2 As Integer, Step3 As Integer,  
             Step4 As Integer, Step5 As Integer, Step6 As Integer,  
             [Step7 As Integer], [Step8 As Integer], [Step9 As Integer] )
```

参数

Step 1 - 9 通知各轴应在 MCal 过程各步骤返回起始点的位模式。
0 至全部轴之间或任意数量的轴均可在第一步过程中返回至起始点位置。
Step 7 - 9 为可选，用于具有 7 个轴以上的机器人。

另见

Home, HomeSet, Hordr, MCal

Mcordr 示例**VB 例:**

```
m_spel.Mcordr(2, 13, 0, 0, 0, 0)
```

C# 例:

```
m_spel.Mcordr(2, 13, 0, 0, 0, 0);
```


MemIn 方法, Spel 类

描述

返回指定内存 I/O 字节端口的状态。每个端口含有 8 个内存 I/O 位。

语法

Function **MemIn** (*PortNumber* As Integer) As Integer
Function **MemIn** (*Label* As String) As Integer

参数

PortNumber 表示其中一个内存 I/O 端口的整数表达式。

Label 含有内存 I/O 字节标签的字符串表达式。

返回值

含有端口值的整数。

另见

In, InBCD, MemOut, MemSw, Sw, Off, On, Oport

MemIn 示例**VB 例:**

```
data = m_spel.MemIn(1)
```

C# 例:

```
data = m_spel.MemIn(1);
```

MemInW 方法, Spel 类

描述

返回指定内存 I/O 字端口的状态。每个字端口含有 16 个内存 I/O 位。

语法

Function **MemInW** (*PortNumber* As Integer) As Integer
Function **MemInW** (*Label* As String) As Integer

参数

PortNumber 表示内存 I/O 字的整数表达式。

Label 含有内存 I/O 字标签的字符串表达式。

返回值

0 至 65535 之间表示输入端口状态的整数表达式。

另见

In, InBCD, MemIn, MemSw, Sw, Off, On, Oport

MemInW 示例**VB 例:**

```
data = m_spel.MemInW(1)
```

C# 例:

```
data = m_spel.MemInW(1);
```

MemOff 方法, Spel 类

描述

关闭 S/W 内存 I/O 的指定位。

语法

Sub **MemOff** (*BitNumber* As Integer)

Sub **MemOff** (*Label* As String)

参数

BitNumber 表示其中一个内存 I/O 位的整数表达式。

Label 含有内存 I/O 位标签的字符串表达式。

另见

In, InBCD, MemOut, MemSw, Sw, Off, On, Oport

MemOff 示例**VB 例:**

```
m_spel.MemOff(500)
```

C# 例:

```
m_spel.MemOff(500);
```

MemOn 方法, Spel 类

描述

打开内存 I/O 的指定位。

语法

Sub **MemOn** (*BitNumber* As Integer)

Sub **MemOn** (*Label* As String)

参数

BitNumber 表示其中一个内存 I/O 位的整数表达式。

Label 含有内存 I/O 位标签的字符串表达式。

另见

In, InBCD, MemOut, MemSw, Sw, Off, On, Oport

MemOn 示例

VB 例:

```
m_spel.MemOn(500)
```

C# 例:

```
m_spel.MemOn(500);
```

MemOut 方法, Spel 类

描述

根据用户指定的 8 位值同时设置 8 个内存 I/O 位。

语法

```
Sub MemOut (PortNumber As Integer, Value As Integer)  
Sub MemOut (Label As String, Value As Integer)
```

参数

PortNumber 表示其中一个内存 I/O 字节的整数表达式。
Label 含有内存 I/O 字节标签的字符串表达式。
Value 含有指定字节输出模式的整数表达式。有效值范围为 0-255。

另见

In, InBCD, MemIn, MemSw, Sw, Off, On, Oport

MemOut 示例**VB 例:**

```
m_spel.MemOut (2, 25)
```

C# 例:

```
m_spel.MemOut (2, 25);
```

MemOutW 方法, Spel 类

描述

根据用户指定的 16 位值同时设置 16 个内存 I/O 位。

语法

Sub **MemOutW** (*PortNumber* As Integer, *Value* As Integer)

Sub **MemOutW** (*Label* As String, *Value* As Integer)

参数

PortNumber 表示其中一个内存 I/O 字的整数表达式。

Label 含有内存 I/O 字标签的字符串表达式。

Value 使用表达式或数值指定输出数据(整数范围为 0 至 65535)。

另见

In, InBCD, MemIn, MemSw, Sw, Off, On, Oport

MemOutW 示例**VB 例:**

```
m_spel.MemOutW(2, 25)
```

C# 例:

```
m_spel.MemOutW(2, 25);
```

MemSw 方法, Spel 类

描述

返回指定的内存 I/O 位状态。

语法

Function **MemSw** (*BitNumber* As Integer) As Boolean
Function **MemSw** (*Label* As String) As Boolean

参数

BitNumber 表示其中一个内存 I/O 位的整数表达式。
Label 含有内存 I/O 位标签的字符串表达式。

返回值

如果指定的内存 I/O 打开, 则返回 True, 否则返回 False。

另见

In, InBCD, MemIn, Sw, Off, On, Oport

MemSw 示例**VB 例:**

```
If m_spel.MemSw(10) Then  
    m_spel.On(2)  
End If
```

C# 例:

```
if (m_spel.MemSw(10))  
    m_spel.On(2);
```

Move 方法, Spel 类

描述

使用 CP 运动(直线运动), 将机械臂从当前位置移至指定点。

语法

```
Sub Move (PointNumber As Integer)
Sub Move (Point As SpelPoint)
Sub Move (Point As SpelPoint, AttribExpr As String)
Sub Move (PointExpr As String)
```

参数

每个语法具有一个指定机械臂在 Move 动作期间所移至端点的参数。此参数为线性内插动作结束时的最终位置。

PointNumber 通过对当前机器人控制器点内存中之前示教的点使用点编号来指定端点。

Point 通过使用 SpelPoint 数据类型指定端点。

AttribExpr 通过使用字符串表达式指定端点属性。

PointExpr 通过使用字符串表达式指定端点。

另见

AccelR, AccelS, SpeedR, SpeedS
Arc, Arc3, CVMove, Go, Jump, Jump3, Jump3CP
BGo, BMove, TGo, TMove
Arch, CP, Till

Move 示例**VB 例:**

```
' 使用点编号指定
m_spel.Tool(1)
m_spel.Move(100)

' 使用 SpelPoint 指定
Dim pt As SpelPoint
pt = m_spel.GetPoint("P*")
pt.X = 125.5
m_spel.Move(pt)

' 使用点表达式指定
m_spel.Move("P0 /L /2 ROT")
m_spel.Move("P1 :Z(-20)")

' 使用并行处理
m_spel.Move("P1 !D50; On 1; D90; Off 1!")

' 在点标签中指定
m_spel.Move("pick")
```


C# 例:

```
// 使用点编号指定
m_spel.Tool(1);
m_spel.Move(100);

// 使用 SpelPoint 指定
SpelPoint pt;
pt = m_spel.GetPoint("P0");
pt.X = 125.5;
m_spel.Move(pt);

// 使用点表达式指定
m_spel.Move("P0 /L /2 ROT");
m_spel.Move("P1 :Z(-20)");

// 使用并行处理
m_spel.Move("P1 !D50; On 1; D90; Off 1!");

// 在点标签中指定
m_spel.Move("pick");
```

Off 方法, Spel 类

描述

关闭指定输出。

语法

Sub **Off** (*BitNumber* As Integer)

Sub **Off** (*Label* As String)

参数

BitNumber 表示其中一个标准或扩展输出的整数表达式。用以通知 **Off** 指令待关闭的输出。

Label 含有输出位标签的字符串表达式。

另见

On, Oport, Out, OutW

Off 示例

VB 例:

```
m_spel.Off(1)
```

C# 例

```
m_spel.Off(1);
```

OLRate 方法, Spel 类

描述

返回指定关节的过负载率。

语法

Function **OLRate** (*JointNumber* As Integer) As Single

参数

JointNumber 显示关节编号的整数值 (范围: 1~ 机器人的关节数量)

返回值

返回指定关节的过负载率。返回的值在 0.0~2.0 的范围内。

备注

OLRate 会检查循环是否造成关节过载。在高负载的循环中应用中, 温度和电流会导致伺服错误。使用 OLRate 可帮助检查机器人是否会出现伺服错误。

在循环操作期间执行令一个任务以监控 OLRate。如果有的关节 OLRate 超过 1.0, 则会发生伺服错误。

超负载时极有可能发生伺服错误。您可以在测试循环时使用 OLRate, 确认速度和加减速, 以采取有效错误防止运行时产生伺服错误。

要获得有效值, 请在机器人运行时使用本方法。

正常负载状态下, 无法使用本方法。

OLRate 示例**VB 例:**

```
Dim data As Single  
data = m_spel.OLRate(1)
```

C# 例:

```
float data;  
data = m_spel.OLRate(1);
```

On 方法, Spel 类

描述

打开指定输出。

语法

Sub **On** (*BitNumber* As Integer)

Sub **On** (*Label* As String)

参数

BitNumber 表示其中一个标准或扩展输出的整数表达式。

Label 含有输出位标签的字符串表达式。

另见

Off, Oport, Out, OutW

On 示例

VB 例:

```
m_spel.On(1)
```

C# 例:

```
m_spel.On(1);
```

OpBCD 方法, Spel 类

描述

使用 BCD(二进制编码的十进制)格式同时设置 8 个输出位。

语法

OpBCD (*PortNumber* As Integer, *Value* As Integer)

OpBCD (*Label* As String, *Value* As Integer)

参数

PortNumber 表示其中一个端口的整数。
每个端口含有 8 个输出位(一个字节)。

Value 0-99 之间表示指定端口输出模式的整数。
第二位(称为个位)表示端口中的 4 个低位输出, 第一位(称为十位)表示端口中的 4 个高位输出。

另见

Off, Out, Sw

OpBCD 示例**VB 例:**

```
m_spel.OpBCD(1, 25)
```

C# 例:

```
m_spel.OpBCD(1, 25);
```

Oport 方法, Spel 类

描述

返回指定输出位的状态。

语法

Function **Oport** (*BitNumber* As Integer) As Boolean
Function **Oport** (*Label* As String) As Boolean

参数

BitNumber 表示其中一个标准或扩展输出的整数表达式。

Label 含有输出字节标签的字符串表达式。

返回值

如果指定的输出位打开, 则返回 **True**, 否则返回 **False**。

另见

Off, On, OpBCD, Out, Sw

Oport 示例**VB 例:**

```
If m_spel.Oport(1) Then  
    m_spel.On(2)  
End If
```

C# 例:

```
if (m_spel.Oport(1))  
    m_spel.On(2);
```

Out 方法, Spel 类

描述

同时读取或设置 8 个输出位(一个字节)。

语法

```
Sub Out (PortNumber As Integer, Value As Integer)
Sub Out (Label As String, Value As Integer)
Function Out (PortNumber As Integer) As Integer
Function Out (Label As String) As Integer
```

参数

PortNumber 表示其中一个输出端口的整数。

Label 含有输出字节标签的字符串表达式。

Value 0-255 之间表示输出端口输出模式的整数。如果以十六进制形式表示, 则范围为 &H0 至 &HFF。

返回值

0-255 之间含有端口值的整数。

另见

InBCD, OpBCD, Oport, OutW, Sw

Out 示例**VB 例:**

```
m_spel.Out(1, 240)
```

C# 例:

```
m_spel.Out(1, 240);
```

OutReal 方法, Spel 类

描述

将输出端口的状态, 取得或设置为 32 位浮点数据 (符合 IEEE754 标准)。

语法

```
Function OutReal (WordPortNumber As Integer) As Single  
Sub OutReal (WordPortNumber As Integer, Value As Single)
```

参数

WordPortNumber 显示输出端口的整数
Value 显示输出数据的实数值

返回值

将输出端口的状态返回为 32 位浮点数据 (符合 IEEE754 标准)。

另见

In, InBCD, InReaql, InW, Out, OutW

OutReal 示例**VB 例::**

```
Dim val As Single  
val = m_spel.OutReal(32)
```

C# 例:

```
float val;  
val = m_spel.OutReal(32);
```


OutW 方法, Spel 类

描述

同时读取或设置 16 个输出位(一个字)。

语法

```
Sub OutW (PortNumber As Integer, Value As Integer)
Sub OutW (Label As String, Value As Integer)
Function OutW (PortNumber As Integer) As Integer
Function OutW (Label As String) As Integer
```

参数

PortNumber 表示其中一个输出端口的整数。

Label 含有输出字标签的字符串表达式。

Value 0-65535 之间表示输出端口输出模式的整数。如果以十六进制形式表示, 则范围为 &H0 至 HFFFF。

返回值

0-65535 之间含有端口值的整数。

另见

InBCD, OpBCD, Oport, Out, Sw

OutW 示例**VB 例:**

```
m_spel.OutW(1, 240)
```

C# 例:

```
m_spel.OutW(1, 240);
```

PAgl 方法, Spel 类

描述

返回指定关节的选定旋转轴的关节角度, 或选定线性轴的位置。

语法

Function **PAgl** (*PointNumber* As Integer, *JointNumber* As Integer) As Single

Function **PAgl** (*Point* As SpelPoint, *JointNumber* As Integer) As Single

Function **PAgl** (*Label* As String, *JointNumber* As Integer) As Single

参数

PointNumber 表示当前机器人点内存中点编号的整数表达式。

Point 之前初始化的 SpelPoint。

Label 含有当前机器人点内存中点标签的字符串表达式。

JointNumber 表示所需关节编号的整数表达式。
数值可为 1~9。

返回值

含有以度或毫米为单位的指定关节角度的单值。

另见

Agl, Pls, CX - CT

PAgl 示例**VB 例:**

```
Dim t1Angle As Single  
t1Angle = m_spel.PAgl(1, 1)
```

C# 例:

```
float t1Angle;  
t1Angle = m_spel.PAgl(1, 1);
```

Pallet 方法, Spel 类

描述

定义托盘。

语法

```
Sub Pallet ( PalletNumber As Integer, Point1 As String, Point2 As String, Point3 As String
            [, Point4 As String] , rows As Integer, columns As Integer )
```

参数

PalletNumber 以 0 至 15 的整数表示的托盘编号。

Point1 定义第一个托盘位置的点变量。

Point2 定义第二个托盘位置的点变量。

Point3 定义第三个托盘位置的点变量。

Point4 可选。定义第四个托盘位置的点变量。

Rows 托盘横向侧的点数。每个数为 1 至 32767 之间的整数。

Columns 托盘纵向侧的点数。每个数为 1 至 32767 之间的整数。

另见

Jump, Go, SetPoint

Pallet 示例**VB 例:**

```
m_spel.Pallet(1, 1, 2, 3, 4, 3, 4)
```

C# 例:

```
m_spel.Pallet(1, 1, 2, 3, 4, 3, 4);
```

Pass 方法, Spel 类

描述

指定 PTP 动作通过指定点附近, 而没有停止动作。

语法

Sub **Pass**(PointNumber As Integer)

Sub **Pass**(PassExpr As String)

参数

PointNumber 使用控制器中保存的当前机器人点内存中的示教点指定一个点。

PassExpr 使用字符串表达式指定一个点。

Point specification [, {On | Off | MemOn | MemOff} bit number [,point specification ...]] [LJM [Orientation flag]]

Point specification 指定点编号、P(表达式)或点标签。若点数据完整并以升序或降序排列, 两个点编号可使用冒号组合, 指定为 P(1:5)。

Bit number 使用整数或输出标签指定要打开/关闭的 I/O 输出位或内存 I/O 位。

LJM 可选。使用 LJM 函数转换起始坐标、结束坐标和目标坐标。

Orientation flag 可选。指定 LJM 函数的定向标记参数。

另见

Accel, Go, Jump, Speed

Pass 示例**VB 例:**

```
m_spel.Jump(1)
m_spel.Pass(2) ' 将机械臂 2 移近 P2, 并在到达 P2 之前执行以下命令
m_spel.On(2)
m_spel.Pass(3)
m_spel.Pass(4)
m_spel.Off(0)
m_spel.Pass(5)
```

C# 例:

```
m_spel.Jump(1);
m_spel.Pass(2); // 将机械臂 2 移近 P2, 并在到达 P2 之前执行以下命令
m_spel.On(2);
m_spel.Pass(3);
m_spel.Pass(4);
m_spel.Off(0);
m_spel.Pass(5);
```

Pause 方法, Spel 类**描述**

暂停控制器中所有正常的 SPEL+ 任务。如果机器人正在移动, 则其会自动减速直至停止。

语法

```
Sub Pause ()
```

另见

Continue, EventReceived, Stop

Pause 示例**VB 例:**

```
Sub btnPause_Click() _  
    ByVal sender As System.Object, _  
    ByVal e As System.EventArgs) _  
    Handles btnPause.Click  
  
    m_spel.Pause ()  
    btnPause.Enabled = False  
    btnContinue.Enabled = True  
End Sub
```

C# 例:

```
void btnPause_Click(object sender, EventArgs e)  
{  
    m_spel.Pause ();  
    btnPause.Enabled = false;  
    btnContinue.Enabled = true;  
}
```

PDef 方法, Spel 类

描述

返回指定点的定义状态。

语法

Function **PDef** (*PointNumber* As Integer) As Boolean

参数

PointNumber 当前机器人点内存中的点编号的整数表达式。

返回值

如果指定点已定义, 则返回 **True**, 否则返回 **False**。

另见

PDel

PDef 示例

VB 例:

```
Dim p1Defined As Boolean  
p1Defined = m_spel.PDef(1)
```

C# 例:

```
bool p1Defined;  
p1Defined = m_spel.PDef(1);
```

PDel 方法, Spel 类

描述

删除指定的位置数据。

语法

Sub **PDel** (*FirstPointNumber* As Integer [, *LastPointNumber* As Integer])

参数

FirstPointNumber 指定删除范围内第一个点的整数表达式。

LastPointNumber 可选。指定删除范围内最后一个点的整数表达式。如果忽略, 则只会删除 *FirstPointNumber* 中指定的点。

另见

PDef, LoadPoints, Clear, SavePoints

PDel 示例**VB 例:**

```
m_spel.PDel(1, 10)
m_spel.SavePoints("modell.pts")
```

C# 例:

```
m_spel.PDel(1, 10);
m_spel.SavePoints("modell.pts");
```

PeakSpeed 方法, Spel 类

描述

返回指定关节的峰值速度。

语法

Function **PeakSpeed** (*JointNumber* As Integer) As Single

参数

JointNumber 显示关节编号的整数值 (范围: 1~ 机器人关节数量)

返回值

返回-1~1 范围中的实数值。

备注

本方法会返回一个带符号的值, 该值代表关节最大速度的绝对值。最大速度为 1。峰值速度表示为-1~1 范围中的实数值。

请先执行 PeakSpeedClear 方法, 然后再执行本方法以显示关节的速度峰值。

使用虚拟控制器和空转时, 峰值速度是根据命令速度计算得出的, 而不是实际速度。

本方法不支持 PG 附加轴。

另见

AvgSpeed, AvgSpeedClear, PeakSpeedClear

PeakSpeed 示例**VB 例::**

```
Dim val As Single  
val = m_spel.PeakSpeed(1)
```

C# 例:

```
float val;  
val = m_spel.PeakSpeed(1);
```


PeakSpeedClear 方法, Spel 类**描述**

清除并初始化关节的峰值速度。

语法

```
Sub PeakSpeedClear ()
```

备注

本方法可清除关节的峰值速度值。

执行 PeakSpeed 方法前, 必须先执行本方法。

本方法不支持 PG 附加轴。

另见

AvgSpeed, AvgSpeedClear, PeakSpeed

PeakSpeedClear 示例**VB 例::**

```
m_spel.PeanSpeedClear ()
```

C# 例:

```
m_spel.PeanSpeedClear ();
```

PF_Abort 方法, Spel 类

描述

强制结束指定零件的 Part Feeding 进程操作。

语法

Sub **PF_Abort** (*PartID* As Integer)

参数

PartID 显示零件 ID 的整数值(1~16)

备注

立即中断指定零件的 Part Feeding 进程。

和 PF_Stop 方法不同, 执行中的 Callback 函数会被中断。

如 Part Feeding 操作进程尚未开始, 则不会有任何影响。

PF_Abort 示例

VB 例::

```
m_spel.PF_Abort(1)
```

C# 例:

```
m_spel.PF_Abort(1);
```

PF_Backlight 方法, Spel 类

描述

开启或关闭供料器中的内置背光灯。

语法

Sub **PF_Backlight** (*FeederNumber* As Integer, *State* As Boolean)

参数

<i>FeederNumber</i>	显示供料器编号的整数值
<i>State</i>	指定 On (True) / Off (False)

备注

当系统正在自动执行视觉处理时，背光灯将会自动开启或关闭。

当使用 PF_Vision 调用函数时，可使用本方法开启或关闭内置背光灯。

PF_Backlight 示例**VB 例::**

```
m_spel.PF_Backlight(1, True)
```

C# 例:

```
m_spel.PF_Backlight(1, true);
```

PF_BacklightBrightness 方法, Spel 类

描述

设置供料器内置背光灯的亮度。

语法

Sub **PF_Backlightness** (*FeederNumber* As Integer, *Brightness* As Integer)

参数

FeederNumber 显示供料器编号的整数

SBrightness 用 0%~100%的值指定背光灯的亮度

备注

通常可以在[Part Feeding Configuration]对话框中设定内置背光灯的亮度。当程序运行中需要修改亮度时，则可以使用本方法。

PF_BacklightBrightness 示例**VB 例::**

```
m_spel.PFBacklightness(1, 80)
```

C# 例:

```
m_spel.PF_BacklightBrightness(1, 80);
```

PF_Name 方法, Spel 类

描述

从零件 ID 获取零件名称。

语法

Function **PF_Name** (*PartID* As Integer) As String

参数

PartID 显示零件 ID 的整数值(1~16)

返回值

用字符串返回指定零件 ID 的名称。

备注

如果指定的零件 ID 无效, 则返回 "" (空字符串)。

PF_Name 示例**VB 例::**

```
Dim part1Name As String
Part1Name = m_spel.PF_Name(1)
```

C# 例:

```
string part1Name;
part1Name = m_spel.PF_Name(1);
```

PF_Number 方法, Spel 类

描述

从零件名称获取零件 ID。

语法

Function **PF_Number** (*PartName* As String) As Integer

参数

PartName 显示零件名称的字符串

返回值

返回指定零件名称的零件 ID (整数 1~16)。

备注

如果零件名称不存在, 则返回-1。

如果零件名称重复, 则返回 ID 最小的零件。

PF_Number 示例**VB 例::**

```
Dim part1ID As Integer  
Part1ID = m_spel.PF_Number("Part1")
```

C# 例:

```
int part1ID;  
part1ID = m_spel.PF_Number("Part1");
```

PF_Start 方法, Spel 类

描述

开始指定零件的 Part Feeding 进程。

语法

Sub PF_Start (PartID1 As Integer, [PartID2 As Integer], [PartID3 As Integer], [PartID4 As Integer])

参数

<i>PartID1</i>	显示住零件 ID 的整数值(1~16)
<i>PartID2</i>	显示附属零件 ID 的整数值 (1~16)。可省略
<i>PartID3</i>	显示附属零件 ID 的整数值 (1~16)。可省略
<i>PartID4</i>	显示附属零件 ID 的整数值 (1~16)。可省略

备注

开始本方法之前, 请限制性以下操作。

- 选择要使用的机器人
- 开启电机
- 执行 PF_InitLog 以输出日志

执行此方法将创建一个新任务, 并将控制权返回给调用命令。

如在以下条件中执行 Status 调用函。

Part Feeding 进程则不会启动。

条件	Status 调用函数的参数 Status 的值
零件 ID 无效	PF_STATUS_BAD_ID
零件参数设定无效 (Enabled 未选中)	PF_STATUS_BAD_PARAMETER
供料器未完成校准	PF_STATUS_CAL_NOT_COMPLETE
发生错误	PF_STATUS_ERROR

本方法无法同时执行多次。如果执行, 则已经执行的处理将继续。不会发生错误。
请在普通任务中执行 PF_Start。如果在后台任务中执行, 则会报错。

注意

在本方法中, 去过指定不存在的零件 ID, 则会产生 7600 错误。

PF_Start 示例**VB 例::**

```
m_spel.PF_Start(1)
```

C# 例:

```
m_spel.PF_Start(1);
```

PF_Stop 方法, Spel 类

描述

要求 Part Feeding 进程结束处理。
如有正在执行中的调用函数，则会等待其完成。
然后执行 PF_CycleStop 调用函数，结束进程。

语法

Sub **PF_Stop** (*PartID* As Integer)

参数

PartID 显示零件 ID 的整数 (1~16)

备注

停止 Part Feeding 进程。
和 PFAbort 方法不同，本方法会等待正在执行中的调用函数完成。
调用函数结束后，执行 PF_CycleStop 调用函数。
如果 Part Feeding 进程尚未开始，则不会有任何影响。

PF_Stop 示例**VB 例::**

```
m_spel.PF_Stop(1)
```

C# 例:

```
m_spel.PF_Stop(1);
```


PLabel 方法, Spel 类

描述

获取或设置指定点编号定义的点标签。

语法

```
Function PLabel (PointNumber As Integer) As String  
Sub PLabel (PointNumber As Integer, PointName As String)
```

参数

PointNumber 显示点编号的整数值
PointName 指定点数据中使用的点标签的字符串

返回值

返回与指定点编号相应的标签。

另见

PDef

PLabel 示例**VB 例::**

```
Dim pt1Label As String  
Pt1Label = m_spel.PLabel(1)
```

C# 例:

```
string pt1Label;  
pt1Label = m_spel.PLabel(1);
```

Plane 方法, Spel 类

描述

定义平面。

语法

Sub **Plane** (*PlaneNumber* As Integer, *Point* As SpelPoint)

Sub **Plane** (*PlaneNumber* As Integer, *X* As Single, *Y* As Single, *Z* As Single, *U* As Single, *V* As Single, *W* As Single)

参数

PlaneNumber 1-15 之间表示 15 个平面中待定义平面的整数。

Point 表示结束检查平面坐标数据的点数据。

X 表示结束检查平面坐标数据的点的 X 坐标。

Y 表示结束检查平面坐标数据的点的 Y 坐标。

Z 表示结束检查平面坐标数据的点的 Z 坐标。

U 表示结束检查平面坐标数据的点的 U 坐标。

V 表示结束检查平面坐标数据的点的 V 坐标。

W 表示结束检查平面坐标数据的点的 W 坐标。

另见

PlaneClr, PlaneDef

Plane 示例**VB 例:**

```
m_spel.Plane(1, -5, 5, -10, 10, -20, 20)
```

C# 例:

```
m_spel.Plane(1, -5, 5, -10, 10, -20, 20);
```

PlaneClr 方法, Spel 类

描述

清除(取消定义)平面。

语法

Sub **PlaneClr** (*PlaneNumber* As Integer)

参数

PlaneNumber 1-15 之间表示 15 个平面中待清除平面的整数。

另见

Plane, PlaneDef

PlaneClr 示例**VB 例:**

```
m_spel.PlaneClr(1)
```

C# 例:

```
m_spel.PlaneClr(1);
```

PlaneDef 方法, Spel 类

描述

返回是否已定义平面。

语法

Function **PlaneDef** (*PlaneNumber* As Integer) As Boolean

参数

PlaneNumber 1 至 15 之间表示平面编号的整数表达式。

返回值

如果指定的平面已定义, 则返回 **True**, 否则返回 **False**。

另见

Plane, PlaneClr

PlaneDef 示例

VB 例:

```
x = m_spel.PlaneDef(1)
```

C# 例:

```
x = m_spel.PlaneDef(1);
```

Pls 方法, Spel 类

描述

返回每个轴在当前位置的当前编码器脉冲数。

语法

Function **Pls** (*JointNumber* As Integer) As Integer

参数

JointNumber 获取当前编码器脉冲数的特定轴。(1 至 9)

返回值

含有指定关节当前脉冲数的整数。

另见

Agl, Pulse

Pls 示例**VB 例:**

```
j1Pulses = m_spel.Pls(1)
```

C# 例:

```
j1Pulses = m_spel.Pls(1);
```

PTCLR 方法, Spel 类

描述

清除并初始化关节的扭矩峰值。

语法

Sub **PTCLR** ()

备注

使用本方法可以清除指定关节的扭矩峰值。
执行 PTRQ 方法前, 请务必先执行本方法。

另见

ATCLR, ATRQ, PTRQ

PTCLR 示例

VB 例::

```
m_spel.PTCLR ()
```

C# 例:

```
m_spel.PTCLR ();
```

PTPBoost 方法, Spel 类

描述

设置短距离 PTP(点到点)动作的增量参数。

语法

Sub **PTPBoost** (*BoostValue* As Integer [, *DepartBoost* As Integer] [, *ApproBoost* As Integer])

参数

BoostValue 0-100 之间的整数表达式。

DepartBoost 可选。Jump 起始增量值。0-100 之间的整数表达式。

ApproBoost 可选。Jump 结束增量值。0-100 之间的整数表达式。

另见

PTPBoostOK

PTPBoost 示例**VB 例:**

```
m_spel.PTPBoost(50)  
m_spel.PTPBoost(50, 30, 30)
```

C# 例:

```
m_spel.PTPBoost(50);  
m_spel.PTPBoost(50, 30, 30);
```

PTPBoostOK 方法, Spel 类

描述

返回从当前位置到目标位置的 PTP(点到点)动作是否为较短的行进距离。

语法

Function **PTPBoostOK** (*PointNumber* As Integer) As Boolean

Function **PTPBoostOK** (*Point* As SpelPoint) As Boolean

Function **PTPBoostOK** (*PointExpr* As String) As Boolean

参数

每个语句具有一个指定待检查目标点的参数。

PointNumber 通过使用当前机器人控制器点内存中之前示教点的点编号来指定目标点。

Point 通过使用 SpelPoint 数据类型指定目标点。

PointExpr 通过使用字符串表达式指定目标点。

返回值

如果将使用 PTPBoost, 则返回 True, 否则返回 False。

另见

PTPBoost

PTPBoostOK 示例**VB 例:**

```
If m_spel.PTPBoostOK(1) Then
    m_spel.Go(1)
End If
```

C# 例:

```
if (m_spel.PTPBoostOK(1))
    m_spel.Go(1);
```


PTran 方法, Spel 类

描述

通过脉冲执行相对关节移动。

语法

Sub **PTran** (*JointNumber* As Integer, *Pulses* As Integer)

参数

JointNumber 待移动的特定关节。

Pulses 待移动的脉冲数。

另见

JTran, Pulse

PTran 示例**VB 例:**

```
' 移动关节 1, 正方向 5000 个脉冲。  
m_spel.PTran(1, 5000)
```

C# 例:

```
// 移动关节 1, 正方向 5000 个脉冲。  
m_spel.PTran(1, 5000);
```

PTRQ 方法, Spel 类

描述

返回指定关节的扭矩峰值。

语法

Function PTRQ (*JointNumber* As Integer) As Single

参数

JointNumber 显示关节编号的整数值(范围: 1~ 机器人的关节数量)

返回值

返回 0~1 范围内的实数值。

另见

ATCLR, ATRQ, PTCLR

PTRQ 示例

VB 例::

```
Dim peakTorque As Single  
peakTorque = m_spel.PTRQ(1)
```

C# 例:

```
float peakTorque;  
peakTorque = m_spel.PTRQ(1);
```

Pulse 方法, Spel 类

描述

通过 PTP 控制将机器人机械臂移至所有机器人关节脉冲值指定的点。

语法

```
Sub Pulse (J1Pulses As Integer, J2Pulses As Integer, J3Pulses As Integer,
           J4Pulses As Integer [, J5Pulses As Integer] [, J6Pulses As Integer]
           [, J7Pulses As Integer] [, J8Pulses As Integer] [, J9Pulses As Integer])
```

参数

J1Pulses - *J9Pulses* 含有关节 1-9 的脉冲值的整数表达式。
关节 5-9 为可选。

NOTE: 脉冲值必须介于各关节指定的范围内。

另见

Go, Move, Jump

Pulse 示例**VB 例:**

```
m_spel.Pulse(5000, 1000, 0, 0)
```

C# 例:

```
m_spel.Pulse(5000, 1000, 0, 0);
```

Quit 方法, Spel 类

描述

终止指定任务的执行。

语法

Sub **Quit** (*TaskNumber* As Integer)

Sub **Quit** (*TaskName* As String)

参数

TaskNumber 待中断任务的任务号。
任务号范围为 1 至 32。

TaskName 含有任务名称的字符串表达式。

另见

Halt, Resume, Xqt

Quit 示例**VB 例:**

```
m_spel.Quit(3)
```

C# 例:

```
m_spel.Quit(3);
```

RadToDeg 方法, Spel 类

描述

将弧度转换为度。

语法

Function **RadToDeg** (*Radians* As Double) As Double

参数

Radians 含有待转换为度的弧度的双值表达式。

返回值

含有以度为单位的转换值的双值。

另见

DegToRad

RadToDeg 示例**VB 例:**

```
Dim deg As Double

deg = m_spel.RadToDeg(1)
```

C# 例:

```
double deg;
deg = m_spel.RadToDeg(1);
```

RebootController 方法, Spel 类

描述

对当前连接中的控制器进行重启。

语法

Sub **RebootController** (*ShowStatusDialog* As Boolean)

参数

ShowStatusDialog 设置是否在重启完成之前显示状态画面。

True=显示画面、False=不显示画面

备注

使用 ShowStatusDialog 显示带有进度条的对话框。您可以从对话框或 Abort 方法中止操作。

另见

Abort

RebootController 示例**VB 例::**

```
m_spel.RebootController(True)
```

C# 例:

```
m_spel.RebootController(true);
```

RebuildProject 方法, Spel 类**描述**

完全重新构建 Project 属性中指定的当前项目。

语法

```
Sub RebuildProject ()
```

另见

BuildProject, EnableEvent, EventReceived, Project, ProjectBuildComplete

RebuildProject 示例**VB 例:**

```
With m_spel  
    .Project =  
    "c:\EpsonRC70\projects\myproject\myproject.sprj"  
    .RebuildProject()  
End With
```

C# 例:

```
m_spel.Project =  
@"c:\EpsonRC70\projects\myproject\myproject.sprj";  
m_spel.RebuildProject();
```

Recover 方法, Spel 类

描述

Recover 会将机器人移回至安全防护打开时所在的位置。

语法

Function **Recover** () As Boolean

备注

Recover 方法可在安全防护关闭之后用以打开机器人电机，并将机器人缓慢移回至安全防护打开时所在的位置。Recover 成功完成后，可执行 Cont 方法继续循环。如果 Recover 已成功完成，则会返回 True。如果在恢复动作期间发生暂停、中止或安全防护打开，则 Recover 会返回 False。

返回值

如果恢复动作已完成，则返回 True，否则返回 False。

另见

Continue, Pause

Recover 示例**VB 例:**

此示例首先执行 **recover**，然后 **continue**

```
Sub btnCont_Click( _
    ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnCont.Click
    Dim sts As Boolean
    Dim answer As Integer

    sts = m_spel.Recover()
    If sts = False Then
        Exit Sub
    End If
    answer = MsgBox("Ready to continue?", vbYesNo)
    If answer = vbYes Then
        m_spel.Continue()
    EndIf
End sub
```

此示例所示为在按下按钮时使用按钮执行 **recover** 的方法。如果在恢复动作期间释放按钮，则会 **pause** 且恢复中止。如果按住按钮直至恢复完成，则会显示一条消息。

```
Sub btnRecover_MouseDown( _
    ByVal sender As System.Object, _
    ByVal e As System.Windows.Forms.MouseEventArgs) _
    Handles btnRecover.MouseDown
    Dim sts As Boolean

    sts = m_spel.Recover()
    If sts = True Then
        MsgBox("Recover complete")
    EndIf
End Sub

Sub btnRecover_MouseUp( _
    ByVal sender As System.Object, _
    ByVal e As System.Windows.Forms.MouseEventArgs) _
    Handles btnRecover.MouseUp

    m_spel.Pause()
End Sub
```

C#例:

此示例首先执行 `recover`，然后 `continue`

```
void btnCont_Click(object sender, EventArgs e)
{
    bool sts;
    DialogResult answer;

    sts = m_spel.Recover();
    if (sts == true){
        answer = MessageBox.Show("Continue?", "",
            MessageBoxButtons.YesNo);
        If (answer == DialogResult.Yes)
            m_spel.Continue();
    }
}
```

此示例所示为在按下按钮时使用按钮执行 `recover` 的方法。如果在恢复动作期间释放按钮，则会 `pause` 且恢复中止。如果按住按钮直至恢复完成，则会显示一条消息。

```
void btnCont_Click(object sender, EventArgs e)
{
    bool sts;

    sts = m_spel.Recover();
    if (sts == true)
        MessageBox.Show("Recover complete");
}

void btnRecover_MouseUp(object sender, EventArgs e)
{
    m_spel.Pause();
}
```

Reset 方法, Spel 类**描述**

将控制器设为初始化状态。

语法

```
Sub Reset ()
```

另见

ResetAbort

Reset 示例**VB 例:**

```
m_spel.Reset ()
```

C# 例:

```
m_spel.Reset ();
```

ResetAbort 方法, Spel 类**描述**

重置使用 Stop 方法设置的中止标志。

语法

```
Sub ResetAbort ()
```

备注

执行 Stop 方法且循环中不存在任何其他 Spel 方法时, 则下一 Spel 方法将会生成用户中止错误。这样便会使得无论何时发出 Stop, 正在执行 Spel 方法的例程都会收到错误。此时可使用 **ResetAbort** 清除这种条件。

NOTE: ResetAbortEnabled 属性必须设为 True, 以使 ResetAbort 功能工作。

另见

Abort, Reset, ResetAbortEnabled

ResetAbort 示例**VB 例:**

```
Sub btnMcal_Click() Handles btnMcal.Click  
    m_spel.ResetAbort()  
    m_spel.MCal()  
End Sub
```

C# 例:

```
void btnMCal_Click(object sender, EventArgs e)  
{  
    m_spel.ResetAbort();  
    m_spel.MCal();  
}
```

Resume 方法, Spel 类

描述

恢复被 Halt 方法暂停的任务。

语法

```
Sub Resume (TaskNumber As Integer)  
Sub Resume (TaskName As String)
```

参数

TaskNumber 已中断任务的任务号。任务号范围为 1 至 32。
TaskName 含有任务名称的字符串表达式。

另见

Quit, Xqt

Resume 示例**VB 例:**

```
m_spel.Resume (2)
```

C# 例:

```
m_spel.Resume (2);
```

RunDialog 方法, Spel 类

描述

运行 EPSON RC+ 7.0 对话框。

语法

Sub **RunDialog** (*DialogID* As SpelDialogs [, *Parent* As Form])

参数

DialogID 待运行 EPSON RC+ 7.0 对话框的 ID。

Parent 可选。 .NET 窗体将作为父窗口。

另见

ShowWindow

RunDialog 示例**VB 例:**

```
Sub btnRobotManager_Click( _  
    ByVal sender As System.Object, _  
    ByVal e As System.EventArgs) _  
    Handles btnRobotManager.Click  
  
    m_spel.RunDialog(SpelDialogs.RobotManager)  
End Sub
```

C# 例:

```
void btnRobotManager_Click(object sender, EventArgs e)  
{  
    m_spel.RunDialog(SpelDialogs.RobotManager);  
}
```

SavePoints 方法, Spel 类**描述**

将当前机器人的点保存到文件中。

语法

Sub **SavePoints** (*FileName* As String)

参数

FileName 保存当前项目中的点的文件名称。

另见

LoadPoints

SavePoints 示例**VB 例:**

```
With m_spel  
    .SavePoints ("part1.pts")  
End With
```

C# 例:

```
m_spel.SavePoints ("part1.pts");
```

Sense 方法, Spel 类

描述

指定输入条件, 若此条件满足, 则会在目标位置上方停止机器人, 进而完成正在进行的 Jump。

语法

Sub **Sense** (*Condition* As String) As Boolean

参数

Condition 指定 I/O 条件。
有关详细信息, 请参阅 *SPEL+ 语言参考手册* 中的 *Sense* 语句。

另见

Jump, JS

Sense 示例**VB 例:**

```
With m_spel
    .Sense("Sw(1) = On")
    .Jump("P1 SENSE")
    stoppedOnSense = .JS()
End With
```

C# 例:

```
m_spel.Sense("Sw(1) = On");
m_spel.Jump("P1 SENSE");
stoppedOnSense = m_spel.JS();
```


SetIODef 方法, Spel 类

描述

设置输入、输出或内存 I/O 位、字节或字的 I/O 标签和描述。

语法

Sub **SetIODef** (*Type* As SpelLabelTypes, *Index* As Integer, *Label* As String, *Description* As String)

参数

<i>Type</i>	指定 I/O 类型, 如下所示: InputBit = 1 InputByte = 2 InputWord = 3 OutputBit = 4 OutputByte = 5 OutputWord = 6 MemoryBit = 7 MemoryByte = 8 MemoryWord = 9 InputReal = 10 OutputReal = 11
<i>Index</i>	指定位或端口号。
<i>Label</i>	指定新标签。
<i>Description</i>	指定新描述。

备注

使用 SetIODef 定义所有 I/O 点的标签和描述。

另见

GetIODef

SetIODef 示例**VB 例:**

```
Dim label, desc As String
label = "StartCycle"
desc = "Starts the robot cycle"
m_spel.SetIODef(SpelLabelTypes.InputBit, 0, label, desc)
```

C# 例:

```
string label, desc;
label = "StartCycle";
desc = "Starts the robot cycle";
m_spel.SetIODef(SpelLabelTypes.InputBit, 0, label, desc);
```

SetPoint 方法, Spel 类

描述

设置当前机器人点的坐标数据。

语法

Sub **SetPoint**(*PointNumber* As Integer, *Point* As SpelPoint)

Sub **SetPoint**(*PointLabel* As String, *Point* As SpelPoint)

Sub **SetPoint**(*PointNumber* As Integer, *X* As Single, *Y* As Single, *Z* As Single, *U* As Single)

Sub **SetPoint**(*PointLabel* As String, *X* As Single, *Y* As Single, *Z* As Single, *U* As Single)

Sub **SetPoint**(*PointNumber* As Integer, *X* As Single, *Y* As Single, *Z* As Single, *U* As Single,

Local As Integer, *Hand* As SpelHand)

Sub **SetPoint**(*PointLabel* As String, *X* As Single, *Y* As Single, *Z* As Single, *U* As Single, *Local* As Integer, *Hand* As SpelHand)

Sub **SetPoint**(*PointNumber* As Integer, *X* As Single, *Y* As Single, *Z* As Single, *U* As Single,

V As Single, *W* As Single)

Sub **SetPoint**(*PointLabel* As String, *X* As Single, *Y* As Single, *Z* As Single, *U* As Single, *V* As Single, *W* As Single)

Sub **SetPoint**(*PointNumber* As Integer, *X* As Single, *Y* As Single, *Z* As Single, *U* As Single,

V As Single, *W* As Single, *Local* As Integer, *Hand* As SpelHand, *Elbow* As SpelElbow, *Wrist* As SpelWrist, *J4Flag* As Integer, *J6Flag* As Integer)

Sub **SetPoint**(*PointLabel* As String, *X* As Single, *Y* As Single, *Z* As Single, *U* As Single, *V* As Single, *W* As Single, *Local* As Integer, *Hand* As SpelHand, *Elbow* As SpelElbow, *Wrist* As SpelWrist, *J4Flag* As Integer, *J6Flag* As Integer)

Sub **SetPoint**(*PointNumber* As Integer, *X* As Single, *Y* As Single, *Z* As Single, *U* As Single,

V As Single, *W* As Single, *S* As Single, *T* As Single)

Sub **SetPoint**(*PointLabel* As String, *X* As Single, *Y* As Single, *Z* As Single, *U* As Single, *V* As Single, *W* As Single, *S* As Single, *T* As Single)

Sub **SetPoint**(*PointNumber* As Integer, *PointExpr* As String)

Sub **SetPoint**(*PointLabel* As String, *PointExpr* As String)

参数

<i>PointNumber</i>	指定当前机器人点内存中点编号的整数表达式。
<i>X</i>	指定点的 X 坐标。
<i>Y</i>	指定点的 Y 坐标。
<i>Z</i>	指定点的 Z 坐标。
<i>U</i>	指定点的 U 坐标。
<i>V</i>	指定点的 V 坐标。
<i>W</i>	指定点的 W 坐标。
<i>S</i>	指定点的 S 坐标。
<i>T</i>	指定点的 T 坐标。
<i>Local</i>	指定点的本地编号。没有本地时使用 0。
<i>Hand</i>	指定点的手方向。
<i>Elbow</i>	指定点的肘方向。
<i>Wrist</i>	指定点的手腕方向。
<i>PointExpr</i>	通过使用字符串表达式指定点。

NOTE

请勿在 X、Y、Z、U、V、W、S 和 T 参数中输入整数值。使用 Single 变量或直接输入 Single 类型值。

另见

GetPoint, LoadPoints, SavePoints

SetPoint 示例**VB 例:**

```
Dim pt As SpelPoint
' 获取 P1 坐标
pt = m_spel.GetPoint(1)
' 变更坐标
pt.U = pt.U - 10.5
m_spel.SetPoint(1, pt)
```

C# 例:

```
SpelPoint pt;
// 获取 P1 坐标
pt = m_spel.GetPoint(1);
// 变更坐标
pt.U = pt.U - 10.5;
m_spel.SetPoint(1, pt);
```

SetVar 方法, Spel 类

描述

设置控制器中 SPEL+ 全局保留变量的值。

语法

Sub **SetVar** (*VarName* As String, *Value* As Object)

参数

VarName SPEL+ 全局保留变量的名称。

Value 新值。

备注

可使用 SetVar 设置单变量和数组变量的值。请参见以下示例。

另见

GetVar

SetVar 示例**VB 例:**

```
m_spel.SetVar("g_myIntVar", 123)

Dim i, myArray(10) As Integer
For i = 1 To 10
    myArray(i) = i
Next i
m_spel.SetVar("g_myIntArray", myArray)

m_spel.SetVar("g_myIntArray(1)", myArray(1))
```

C# 例:

```
m_spel.SetVar("g_myIntVar", 123);

int[] myArray = new int[10];
for(int i = 1; i < 10; i++)
    myArray[i] = i;

m_spel.SetVar("g_myIntArray", myArray);

m_spel.SetVar("g_myIntArray[1]", myArray[1]);
```

SFree 方法, Spel 类

描述

释放伺服控制的指定机器人轴。

语法

```
Sub SFree ()  
Sub SFree (ParamArray Axes() As Integer)
```

参数

Axes 含有一个用以释放每个机器人轴的元素整数参数数组。
可指定编号为 1 - 9 之间的任一轴。

另见

SLock

SFree 示例**VB 例:**

```
' Sfree 第 1 关节和第 2 关节  
m_spel.SFree (1, 2)
```

C# 例:

```
// Sfree 第 1 关节和第 2 关节  
m_spel.SFree (1, 2);
```

ShowWindow 方法, Spel 类

描述

显示 EPSON RC+ 7.0 窗口。

语法

```
Sub ShowWindow (WindowID As SpelWindows [, Parent As Form])
```

参数

WindowID 待显示 EPSON RC+ 7.0 窗口的 ID。

Parent 可选。 .NET 窗体将作为父窗口。

备注

可使用 *Parent* 参数为窗口指定 .NET 父窗体。如果无法使用 .NET 父窗体，则必须忽略 *Parent* 参数并使用 *ParentWindowHandle* 属性设置父窗体的句柄。

另见

HideWindow, ParentWindowHandle, RunDialog, ServerOutOfProcess

ShowWindow 示例**VB 例:**

```
Sub btnShowIOMonitor_Click( _  
    ByVal sender As System.Object, _  
    ByVal e As System.EventArgs) _  
    Handles btnShowIOMonitor.Click  
  
    m_spel.ShowWindow(RCAPINet.SpelWindows.IOMonitor, Me)  
End Sub
```

C# 例:

```
void btnShowIOMonitor_Click(object sender, EventArgs e)  
{  
    m_spel.ShowWindow(RCAPINet.SpelWindows.IOMonitor, this);  
}
```

SimGet 方法, Spel 类

描述

获取模拟器中各种对象的属性设置。

语法

```
Sub SimGet (Object As String, Property As SpelSimProps, ByRef Value As Boolean)
Sub SimGet (RobotName As String, HandName As String, Property As SpelSimProps,
ByRef Value As Boolean)
Sub SimGet (Object As String, Property As SpelSimProps, ByRef Value As Double)
Sub SimGet (RobotName As String, HandName As String, Property As SpelSimProps,
ByRef Value As Double)
Sub SimGet (Object As String, Property As SpelSimProps, ByRef Value As Integer)
Sub SimGet (RobotName As String, HandName As String, Property As SpelSimProps,
ByRef Value As Integer)
Sub SimGet (Object As String, Property As SpelSimProps, ByRef Value As Boolean)
Sub SimGet (RobotName As String, HandName As String, Property As SpelSimProps,
ByRef Value As String)
```

参数

<i>Object</i>	显示获取属性值的对象名称的字符串变量。
<i>RobotName</i>	显示安装了“Hand”中指定的夹爪的机器人的名称的字符串变量。
<i>HandName</i>	显示获取属性值的夹爪名称的字符串变量。
<i>Property</i>	获取值的属性的名称。
<i>Value</i>	显示返回值的变量。

备注

本方法用于获取模拟器中各种对象的属性设置。

有关各属性的详细资讯，请参阅以下手册
EPSON RC+ 7.0 SPEL+ 语言参考 SimGet

另见

SimSet

SimGet 示例**VB 例:**

```
Dim posX As Double
m_spel.SimGet("SBox_1", SpelSimProps.PositionX, posX)
```

C# 例:

```
double posX;
SimGet("SBox_1", SpelSimProps.PositionX, out posX);
```

SimResetCollision 方法, Spel 类

描述

重置碰撞检测。

语法

Sub **SimResetCollision** ()

备注

执行本方法，如果机器人与对象之间没有发生碰撞，则将解除碰撞状态，并更新模拟器的 3D 显示。如果机器人与对象之间产生碰撞，则无法解除碰撞状态，模拟器的 3D 显示也不会更新。

有关详细咨询，请参阅以下手册。

EPSON RC+ 7.0 SPEL+ 语言参考 SimSet

另见

SimSet

SimResetCollision 示例**VB 例:**

```
m_spel.SimResetCollision ()
```

C# 例:

```
m_spel.SimResetCollision ();
```


SimSet 方法, Spel 类

描述

在模拟器中设置各种对象的属性。还能进行机器人动作和对象操作，以及模拟器设置等操作。

语法

Sub **SimSet** (*Object* As String, *Property* As SpelSimProps, *Value* As Boolean)

Sub **SimSet** (*RobotName* As String, *HandName* As String, *Property* As SpelSimProps, *Value* As Boolean)

Sub **SimSet** (*Object* As String, *Property* As SpelSimProps, *Value* As Integer)

Sub **SimSet** (*RobotName* As String, *HandName* As String, *Property* As SpelSimProps, *Value* As Integer)

Sub **SimSet** (*Object* As String, *Property* As SpelSimProps, *Value* As Double)

Sub **SimSet** (*RobotName* As String, *HandName* As String, *Property* As SpelSimProps, *Value* As Double)

Sub **SimSet** (*Object* As String, *Property* As SpelSimProps, *Value* As String)

Sub **SimSet** (*RobotName* As String, *HandName* As String, *Property* As SpelSimProps, *Value* As String)

参数

Object 显示取得属性值的对象名称的字符串变量

RobotName 显示安装了“Hand”中指定的夹爪的机器人的名称的字符串变量

HandName 显示获取属性值的夹爪名称的字符串变量

Property 要为其设置新值的属性名称

Value 新数值的表达式

备注

本方法用于更改模拟器中各种对象的属性设置和操作，机器人操作和模拟器设置。



无法使用属性指定 SpelSimProps.Type。

有关各属性的详细资讯，请参阅以下手册。

EPSON RC+ 7.0 SPEL+ 语言参考 SimSet

另见

SimGet

SimSet 示例**VB 例::**

```
m_spel.SimSet ("SBox_1", SpelSimProps.PositionX, 100.0)
```

C# 例:

```
m_spel.SimSet ("SBox_1", SpelSimProps.PositionX, 100.0);
```

SimSetParent 方法, Spel 类

描述

进行对象的操作设置。

语法

Sub **SimSetParent** (*Object* As String)

Sub **SimSetParent** (*Object* As String, *ParentObject* As String)

参数

Object 显示设置了父级对象的对象名称的字符串变量

ParentObject 显示父级对象名称的字符串变量

备注

对于“Object”中指定的对象，将“ParentObject”中指定的对象设置未父级对象。“ParentObject”可以省略。如省略，则将“Object”中指定的对象作为父级对象。例如，在“Object”中指定的对象是某个对象的子级对象，则将取消子级对象的设置。另外，如在“Object”中指定的是零件或机械臂安装设备的对象，则无法指定父级对象。

有关 SetParent 中可以指定的对象，请参阅以下手册。

EPSON RC+ 7.0 SPEL+ 语言参考 SimSet

如果是相机对象，只有设置为固定相机的对象可以使用 SetParent。

另见

SimSet

SimSetParent 示例**VB 例::**

```
m_spel.SimSetParent ("SBox_1")
```

C# 例:

```
m_spel.SimSetParent ("SBox_1");
```

SimSetPick 方法, Spel 类

描述

对指定的机器人, 执行对象的抓取操作。

语法

```
Sub SimSetPick (RobotName As String, Object As String)
```

```
Sub SimSetPick (RobotName As String, Object As String, ToolNumber As Integer)
```

参数

RobotName 显示 Pick 机器人名称的字符串变量

Object 显示被 Pick 的对象名称的字符串变量

ToolNumber 显示 Pick 时使用的 Tool 编号的表达式

备注

“Robot”中指定的机器人对“Object”中指定的对象, 执行抓取操作。抓取的对象可注册为机器人的一部分。另外, 在“Tool”中指定任意的工具编号, 则可以使用指定的工具进行抓取。如果省略“Tool”的设置, 则会使用 Tool0 进行抓取动作。

所有注册为机器人一部分的对象, 或是机械臂安装设备的对象, 都无法被抓取。而且也无法抓取相机。

有关详细资讯, 请参阅以下手册。

EPSON RC+ 7.0 SPEL+ 语言参考 SimSet

另见

SimGet, SimSet, SimSetPlace

SimSetPick 示例**VB 例:**

```
m_spel.SimSetPick ("Robot1", "SBox_1", 1)
```

C# 例:

```
m_spel.SimSetPick ("Robot1", "SBox_1", 1);
```

SimSetPlace 方法, Spel 类

描述

指定的机器人执行对象的放置动作。

语法

Sub **SimSetPlace** (*RobotName* As String, *Object* As String)

参数

RobotName 显示要 Place 机器人名称的字符串变量

Object 显示被 Place 对象名称的字符串变量

备注

“Robot”中指定的机器人，对“Object”中指定的对象，进行放置动作。被放置的对象将被取消作为机器人一部分的注册。

所有被取消机器人一部分的注册的对象，都可以被放置。

有关详细资讯，请参阅以下手册。

EPSON RC+ 7.0 SPEL+ 语言参考 SimSet

另见

SimGet, SimSet, SimSetPick

SimSetPlace 示例**VB 例::**

```
m_spel.SimSetPlace ("Robot1", "SBox_1")
```

C# 例:

```
m_spel.SimSetPlace ("Robot1", "SBox_1");
```

Shutdown 方法, Spel 类

描述

关闭或重启 Windows。

语法

Sub **Shutdown** (*Mode* As SpelShutdownMode)

参数

Mode 0 = 关闭 Windows。
 1 = 重启 Windows。

另见

Reset

Shutdown 示例

VB 例:

```
' 重启 Windows  
m_spel.Shutdown(1)
```

C# 例:

```
// 重启 Windows  
m_spel.Shutdown(1);
```

SLock 方法, Spel 类

描述

将指定的轴返回至伺服控制。

语法

```
Sub SLock ()  
Sub SLock (ParamArray Axes() As Integer)
```

参数

Axes 含有一个用以锁定每个机器人轴的元素整数参数数组。
可指定编号为 1 - 9 之间的任一轴。

另见

SFree

SLock 示例**VB 例:**

```
' 恢复轴 1 和轴 2 的伺服控制  
m_spel.SLock (1, 2)
```

C# 例:

```
// 恢复轴 1 和轴 2 的伺服控制  
m_spel.SLock (1, 2);
```

Speed 方法, Spel 类

描述

指定用于 PTP 指令 Go、Jump 和 Pulse 的机械臂速度。

语法

```
Sub Speed ( PointToPointSpeed As Integer [, JumpDepartSpeed As Integer ]  
            [, JumpApproSpeed As Integer] )
```

参数

PointToPointSpeed 指定用于 PTP 指令 Go、Jump 和 Pulse 的机械臂速度。

JumpDepartSpeed 1-100 之间表示 Jump 指令 Z 轴向上动作速度的整数。

JumpApproSpeed 1-100 之间表示 Jump 指令 Z 轴向下动作速度的整数。

另见

Accel, Jump, Go

Speed 示例**VB 例:**

```
m_spel.Speed(50)
```

C# 例:

```
m_spel.Speed(50);
```

SpeedR 方法, Spel 类

描述

指定使用 ROT 时的工具旋转速度。

语法

Sub **SpeedR** (*RotationSpeed* As Single)

参数

RotationSpeed 指定以度/秒为单位的工具旋转速度。

另见

Arc, Arc3, BMove, Jump3CP, Power, TMove

SpeedR 示例

VB 例:

```
m_spel.SpeedR(100)
```

C# 例:

```
m_spel.SpeedR(100);
```


SpeedS 方法, Spel 类

描述

指定用于连续路径指令 Jump3CP、Move、Arc 和 CVMove 的机械臂速度。

语法

Sub **SpeedS** (*LinearSpeed* As Single [, *JumpDepartSpeed* As Single] [, *JumpApproSpeed* As Single])

参数

LinearSpeed 指定用于连续路径指令 Jump3CP、Move、Arc 和 CVMove 的机械臂速度。

JumpDepartSpeed 表示 Jump3CP 指令 Z 轴向上动作速度的单一表达式。

JumpApproSpeed 表示 Jump3CP 指令 Z 轴向下动作速度的单一表达式。

另见

AccelS, Jump3CP, Move, TMove

SpeedS 示例**VB 例:**

```
m_spel.SpeedS(500)
```

C# 例:

```
m_spel.SpeedS(500);
```

Start 方法, Spel 类

描述

启动一个 SPEL+ 程序。

语法

Sub **Start** (*ProgramNumber* As Integer)

参数

ProgramNumber 启动程序编号，对应于下表所示 SPEL+ 中的 64 个 main 函数。范围为 0-63。

程序编号	SPEL+ 函数名称
0	main
1	main1
2	main2
3	main3
4	main4
5	main5
...	...
63	main63

备注

执行 **Start** 时，控制将立即返回至调用程序。但不能启动正在运行的程序。请注意，**Start** 会导致控制器中的全局变量被清除并加载默认的机器人点。

另见

Continue, Pause, Stop, Xqt

Start 示例**VB 例:**

```
Sub btnStart_Click( _
    ByVal sender As System.Object, _
    ByVal e As System.EventArgs) _
    Handles btnStart.Click

    m_spel.Start(0)
End Sub
```

C# 例:

```
void btnStart_Click(object sender, EventArgs e)
{
    m_spel.Start(0);
}
```

StartBGTask 方法, Spel 类

描述

启动一个 SPEL+ 任务作为后台任务。

语法

```
Sub StartBGTask (FuncName As String)
```

参数

FuncName 待执行函数的名称。

备注

使用 StartBGTask 启动控制器中的 Spel+ 后台任务。后台任务必须在控制器中启用。

请注意, BGMain 会在控制器切至自动模式时自动启动, 因此通常情况下, StartBGTask 不是必需的。如果需要停止所有任务, 然后再启动后台任务, 则需要使用 StartBGTask。

另见

Call, Start, Stop, Xqt

StartBGTask 示例**VB 例:**

```
' 停止所有任务, 包括后台任务
m_spel.Stop(SpelStopType.StopAllTasks)
...
m_spel.RebuildProject()

' 开始主后台任务
m_spel.StartBGTask("BGMain")
```

C# 例:

```
// 停止所有任务, 包括后台任务
m_spel.Stop(SpelStopType.StopAllTasks);
...
m_spel.RebuildProject();

// 开始主后台任务
m_spel.StartBGTask("BGMain");
```

Stat 方法, Spel 类

描述

返回控制器状态。

语法

Function **Stat** (*Address* As Integer) As Integer

参数

Address 指定表示控制器状态的地址。
(整数范围为 0 至 2)

返回值

返回表示控制器状态的 4 字节值。(请参见下表。)

地址	位		位打开时的控制器状态
0	0-15	&H1-&H8000	任务 1 至 16 正在执行(Xqt)或 Halt 状态
	16	&H10000	任务正在执行
	17	&H20000	暂停状态
	18	&H40000	错误状态
	19	&H80000	TEACH 模式
	20	&H100000	紧急停止状态
	21	&H200000	低功率模式
	22	&H400000	安全防护打开
	23	&H800000	Enable 开关打开
	24	&H1000000	未定义
	25	&H2000000	未定义
	26	&H4000000	测试模式
	27	&H8000000	T2 模式状态
	28-31		未定义
1	0	&H1	Jump...Sense 语句的条件满足后在目标位置上方停止的日志。(此日志在执行另一个 Jump 语句时被消除)。
	1	&H2	Go/Jump/Move...Till 语句的条件满足后在中间行进位置停止的日志。(此日志在执行另一个 Go/Jump/Move...Till 语句时被消除)
	2	&H4	未定义
	3	&H8	检测到 Trap 语句时停止正在进行动作的日志。
	4	&H10	Motor On 状态
	5	&H20	起始点位置
	6	&H40	低功率模式
	7	&H80	未定义
	8	&H100	关节 #4 被磁化。
	9	&H200	关节 #3 被磁化。
	10	&H400	关节 #2 被磁化。
	11	&H800	关节 #1 被磁化。
	12	&H1000	关节 #6 被磁化。
	13	&H2000	关节 #5 被磁化。
	14	&H4000	T 轴被磁化。
	15	&H8000	S 轴被磁化。
	16	&H10000	关节 #7 被磁化。
17-31		未定义	
2	0-15	&H1-&H8000	任务 17 至 32 正在执行(Xqt)或在 Halt 状态

另见

EStopOn, PauseOn, SafetyOn

Stat 示例**VB 例:**

```
Dim ctr_stat As Integer  
ctr_stat = m_spel.Stat(0)
```

C# 例:

```
int ctr_stat;  
ctr_stat = m_spel.Stat(0);
```

Stop 方法, Spel 类

描述

停止控制器中正在运行的所有正常 SPEL⁺ 任务, 以及可选停止所有后台任务。

语法

```
Sub Stop ()
Sub Stop (SpelStopType StopType)
```

参数

StopType 可选。指定仅停止正常任务(StopNormalTasks)还是停止所有任务(StopAllTasks)。
如果忽略, 则会指定 StopNormalTasks。

NOTE: 若在 ResetAbortEnabled 为 True 时执行 Stop 方法, 执行 Start 或 Reset 方法时将发生错误 10101。

若要消除错误, 在执行 Stop 方法后执行 ResetAbort 方法。

另见

Continue, Pause, ResetAbort, ResetAbortEnabled, Start, SpelStopType

Stop 示例**VB 例:**

```
Sub btnStop_Click( _
    ByVal sender As System.Object, _
    ByVal e As System.EventArgs) _
    Handles btnStop.Click

    m_spel.Stop()
End Sub
```

C# 例:

```
void btnStop_Click(object sender, EventArgs e)
{
    m_spel.Stop();
}
```

Sw 方法, Spel 类

描述

返回选定的输入位状态。

语法

Function **Sw** (*BitNumber* As Integer) As Boolean
Function **Sw** (*Label* As String) As Boolean

参数

BitNumber 表示其中一个标准或扩展输入的整数表达式。

Label 含有输入位标签的字符串表达式。

返回值

如果指定的输入位打开, 则返回 True, 否则返回 False。

另见

In, InBCD, MemSw, Off, On, Oport

Sw 示例**VB 例:**

```
If m_spel.Sw(1) Then  
    m_spel.On(2)  
End If
```

C# 例:

```
if (m_spel.Sw(1))  
    m_spel.On(2);
```

TargetOK 方法, Spel 类

描述

返回表示是否能够通过 PTP(点到点)动作从当前位置移至目标位置的状态。

语法

Function **TargetOK** (*PointNumber* As Integer) As Boolean

Function **TargetOK** (*Point* As SpelPoint) As Boolean

Function **TargetOK** (*PointExpr* As String) As Boolean

参数

每个语句具有一个指定待检查目标点的参数。

PointNumber 通过使用当前机器人控制器点内存中之前示教点的点编号来指定目标点。

Point 通过使用 SpelPoint 数据类型指定目标点。

PointExpr 通过使用字符串表达式指定目标点。

返回值

如果能够从当前位置移至目标位置, 则返回 True, 否则返回 False。

另见

Go, Jump, Move, TGo, TMove

TargetOK 示例**VB 例:**

```
If m_spel.TargetOK("P1 /F") Then  
    m_spel.Go("P1 /F")  
End If
```

C# 例:

```
if (m_spel.TargetOK("P1 /F"))  
    m_spel.Go("P1 /F");
```


TasksExecuting 方法, Spel 类**描述**

如有任何 SPEL+ 任务正在执行, 则返回 True。

语法

Function **TasksExecuting** () As Boolean

返回值

如有任何 SPEL+ 任务正在执行, 则返回 True, 否则返回 False。

另见

TaskState, Xqt

TasksExecuting 示例**VB 例:**

```
tasksRunning = m_spel.TasksExecuting()
```

C# 例:

```
tasksRunning = m_spel.TasksExecuting();
```

TaskState 方法, Spel 类

描述

返回任务的状态。

语法

```
Function TaskState (TaskNumber As Integer) As SpelTaskState  
Function TaskState (TaskName As String) As SpelTaskState
```

参数

TaskNumber 待返回执行状态的任务编号。

TaskName 含有任务名称的字符串表达式。

返回值

SpelTaskState 值。

另见

TasksExecuting, Xqt

TaskState 示例**VB 例:**

```
Dim taskState As SpelTaskState  
taskState = m_spel.TaskState(2)
```

C# 例:

```
SpelTaskState taskState;  
taskState = m_spel.TaskState(2);
```

TeachPoint 方法, Spel 类

描述

运行允许操作员步进并示教一个点的对话框。

语法

Function **TeachPoint** (*PointFile* As String, *PointNumber* As Integer, *Prompt* As String)
As Boolean

Function **TeachPoint** (*PointFile* As String, *PointName* As String, *Prompt* As String) As
Boolean

Function **TeachPoint** (*PointFile* As String, *PointNumber* As Integer, *Prompt* As String,
Parent As Form) As Boolean

Function **TeachPoint** (*PointFile* As String, *PointName* As String, *Prompt* As String,
Parent As Form) As Boolean

参数

PointFile 含有点文件名称的字符串。

PointNumber 待示教的点编号。

PointName 显示点标签的字符串

Prompt 含有示教对话框底部显示的指示性文本的字符串。

Parent .NET 表单成为窗口的父级 (可省略)

返回值

如果操作员点击了示教按钮，则返回 **True**；如果操作员点击了取消，则返回 **False**。

备注

使用 **TeachPoints** 可允许操作员示教控制器中的一个机器人点。执行 **TeachPoints** 时，将从控制器加载点文件。点击 **Teach** 按钮时，将在控制器中示教点并将点文件保存至控制器。

TeachPoint 示例**VB 例:**

```
Sub btnTeachPick_Click( _  
    ByVal sender As System.Object, _  
    ByVal e As System.EventArgs) _  
    Handles btnTeachPick.Click  
  
    Dim sts As Boolean  
    Dim prompt As String  
  
    prompt = "Jog to Pick position and click Teach"  
    sts = m_spel.TeachPoint("points.pts", 1, prompt)  
  
End Sub
```

C# 例:

```
void btnTeachPick_Click(object sender, EventArgs e)  
{  
    bool sts;  
    string prompt;  
  
    prompt = "Jog to Pick position and click Teach";  
    sts = m_spel.TeachPoint("points.pts", 1, prompt);  
  
}
```

Till 方法, Spel 类

描述

指定事件条件, 如果此条件满足, 则会在中间位置减速并停止机器人, 以完成正在进行的动作命令(Jump、Go、Move 等)。

语法

Sub **Till** (*Condition* As String) As Boolean

参数

Condition 指定 I/O 条件。有关详细信息, 请参阅 *SPEL+ 语言参考手册* 中的 *Till* 语句。

另见

Go, Jump, JS, Sense, TillOn

Till 示例**VB 例:**

```
With m_spel
    .Till("Sw(1) = On")
    .Go("P1 TILL")
End With
```

C# 例:

```
m_spel.Till("Sw(1) = On");
m_spel.Go("P1 TILL");
```

TillOn 方法, Spel 类

描述

如果在最后的 Go/Jump/Move 语句期间因 till 条件而发生停止, 则返回 True。

语法

Function **TillOn** () As Boolean

返回值

如果机器人因 Till 条件而停止, 则返回 True, 否则返回 False。

备注

使用 TillOn 可检查 Till 条件是否在使用 Till 的最后一个动作命令期间打开。

TillOn 等同于 ((Stat(1) And 2) <> 0)

另见

Jump, Till

TillOn 示例**VB 例:**

```
If m_spel.TillOn() Then  
    m_spel.Jump(2)  
End If
```

C# 例:

```
if (m_spel.TillOn())  
    m_spel.Jump(2);
```

TGo 方法, Spel 类

描述

在当前工具坐标系中执行 PTP 的相对运动。

语法

```
Sub TGo (PointNumber As Integer)
Sub TGo (Point As SpelPoint)
Sub TGo (Point As SpelPoint, AttribExpr As String)
Sub TGo (PointExpr As String)
```

参数

每个语法具有一个指定机械臂在 TGo 动作期间所移至端点的参数。此参数为 PTP 动作结束时的最终位置。

PointNumber 通过对当前机器人控制器点内存中之前示教的点使用点编号来指定端点。

Point 通过使用 SpelPoint 数据类型指定端点。

AttribExpr 通过使用字符串表达式指定端点属性。

PointExpr 通过使用字符串表达式指定端点。

另见

Accel, Speed
Arc, Arc3, CVMove, Go, Jump, Jump3, Jump3CP, Move
BGo, BMove, TMove
CP, Till

TGo 示例**VB 例:**

```
' 使用点编号指定
m_spel.Tool(1)
m_spel.TGo(100)

' 使用 SpelPoint 指定
Dim pt As SpelPoint
pt = m_spel.GetPoint("P*")
pt.X = 125.5
m_spel.TGo(pt)

' 使用点表达式指定
m_spel.TGo("P0 /L /2")
m_spel.TGo("P1 :Z(-20)")

' 使用并行处理
m_spel.TGo("P1 !D50; On 1; D90; Off 1!")

' 在点标签中指定
m_spel.TGo("pick")
```

C# 例:

```
// 使用点编号指定
m_spel.Tool(1);
m_spel.TGo(100);

// 使用 SpelPoint 指定
SpelPoint pt;
pt = m_spel.GetPoint("P0");
pt.X = 125.5;
m_spel.TGo(pt);

// 使用点表达式指定
m_spel.TGo("P0 /L /2");
m_spel.TGo("P1 :Z(-20)");

// 使用并行处理
m_spel.TGo("P1 !D50; On 1; D90; Off 1!");

// 在点标签中指定
m_spel.TGo("pick");
```


TLClr 方法, Spel 类

描述

清除(取消定义)工具坐标系。

语法

Sub **TLClr** (*ToolNumber* As Integer)

参数

ToolNumber 表示待清除(取消定义)工具的整数表达式。
(工具 0 为默认工具, 无法清除。)

另见

Tool, TLDef

TLClr 示例**VB 例:**

```
m_spel. TLClr(1)
```

C# 例:

```
m_spel. TLClr(1);
```

TLDef 方法, Spel 类

描述

返回工具定义状态。

语法

Function **TLDef** (*ToolNumber* As Integer) As Boolean

参数

ToolNumber 表示返回状态的工具的整数表达式。

返回值

如果指定的工具已定义, 则返回 True, 否则返回 False。

另见

Tool, TLClr

TLDef 示例

VB 例:

```
m_spel.TLDef(1)
```

C# 例:

```
m_spel.TLDef(1);
```

TLSet 方法, Spel 类

描述

定义工具坐标系。

语法

```
Sub TLset (ToolNumber As Integer , Point As SpelPoint)
Sub TLset ( ToolNumber As Integer, XCoord As Single, YCoord As Single, ZCoord As
Single,
UCoord As Single, VCoord As Single, WCoord As Single )
```

参数

<i>ToolNumber</i>	1-15 之间表示 15 个待定义工具的整数表达式。 (Tool 0 为默认工具, 无法更改。)
<i>Point</i>	含有点数据的 SpelPoint。
<i>XCoord</i>	工具坐标系原点 X 坐标。
<i>YCoord</i>	工具坐标系原点 Y 坐标。
<i>ZCoord</i>	工具坐标系原点 Z 坐标。
<i>UCoord</i>	工具坐标系绕着 Z 轴旋转。
<i>VCoord</i>	工具坐标系绕着 Y 轴旋转。
<i>WCoord</i>	工具坐标系绕着 X 轴旋转。

另见

Arm, Armset, GetTool, Tool

TLSet 示例**VB 例:**

```
m_spel.TLSet(1, .5, 4.3, 0, 0, 0, 0)
```

C# 例:

```
m_spel.TLSet(1, .5, 4.3, 0, 0, 0, 0);
```

TMove 方法, Spel 类

描述

在当前工具坐标系中执行线性内插相对运动。

语法

```
Sub TMove (PointNumber As Integer)
Sub TMove (Point As SpelPoint)
Sub TMove (Point As SpelPoint, AttribExpr As String)
Sub TMove (PointExpr As String)
```

参数

每个语法具有一个指定机械臂在 TMove 动作期间所移至端点的参数。此参数为线性内插动作结束时的最终位置。

PointNumber 通过对当前机器人控制器点内存中之前示教的点使用点编号来指定端点。

Point 通过使用 SpelPoint 数据类型指定端点。

AttribExpr 通过使用字符串表达式指定端点属性。

PointExpr 通过使用字符串表达式指定端点。

另见

AccelR, AccelS, SpeedR, SpeedS
Arc, Arc3, CVMove, Go, Jump, Jump3, Jump3CP, Move
BGo, BMove, TGo
CP, Till

TMove 示例**VB 例:**

```
' 使用点编号指定
m_spel.Tool(1)
m_spel.TMove(100)

' 使用 SpelPoint 指定
Dim pt As SpelPoint
pt = m_spel.GetPoint("P*")
pt.X = 125.5
m_spel.TMove(pt)

' 使用点表达式指定
m_spel.TMove ("P0")
m_spel.TMove ("XY(0, 0, -20, 0)")

' 使用并行处理
m_spel.TMove ("P1 !D50; On 1; D90; Off 1!")

' 在点标签中指定
m_spel.TMove ("pick")
```

C# 例:

```
// 使用点编号指定
m_spel.Tool(1);
m_spel.TMove(100);

// 使用 SpelPoint 指定
SpelPoint pt;
pt = m_spel.GetPoint("P0");
pt.X = 125.5;
m_spel.TMove(pt);

// 使用点表达式指定
m_spel.TMove("P0");
m_spel.TMove("XY(0, 0, -20, 0)");

// 使用并行处理
m_spel.TMove("P1 !D50; On 1; D90; Off 1!");

// 在点标签中指定
m_spel.TMove("pick");
```

Tool 方法, Spel 类

描述

选择当前机器人的工具。

语法

Sub **Tool** (*ToolNumber* As Integer)

参数

ToolNumber 0-15 之间表示 16 个工具定义中将与随后的动作指令一同使用的定义的整数。

另见

TLSet, Arm, TGo, TMove

Tool 示例

VB 例:

```
m_spel.Tool(1)  
m_spel.TGo(100)
```

C# 例:

```
m_spel.Tool(1);  
m_spel.TGo(100);
```

TrapStop 方法, Spel 类

描述

如果当前机器人被之前动作命令中的 Trap 条件而停止, 则返回 True。

语法

Function **TrapStop** () As Boolean

返回值

如果机器人被 Trap 条件而停止, 则返回 True, 否则返回 False。

另见

EStopOn, ErrorOn

TrapStop 示例**VB 例:**

```
If m_spel.TrapStop() Then  
    MsgBox "Robot stopped by Trap"  
End If
```

C# 例:

```
if (m_spel.TrapStop())  
    MessageBox.Show("Robot stopped by trap");
```

TW 方法, Spel 类

描述

返回 WAIT 条件和 WAIT 定时器间隔的状态。

语法

Function **TW** () As Boolean

返回值

如果发生超时, 则返回 True, 否则返回 False。

另见

WaitMem, WaitSw

TW 示例**VB 例:**

```
Const PartPresent = 1
m_spel.WaitSw(PartPresent, True, 5)
If m_spel.TW() Then
    MsgBox "Part present time out occurred"
End If
```

C# 例:

```
const int PartPresent = 1;
m_spel.WaitSw(PartPresent, True, 5);
if (m_spel.TW())
    MessageBox.Show("Part present time out occurred");
```


UserHasRight 方法, Spel 类**描述**

返回当前登录的用户是否具有指定权限。

语法

Function **UserHasRight** (SpelUserRights *Right*) As Boolean

参数

Right 希望检查当前登录用户的权限。

返回值

如果用户具有指定的权限，则返回 **True**，否则返回 **False**。

另见

Login, GetCurrentUser

UserHasRight 示例**VB 例:**

```
Dim hasRight As Boolean  
hasRight = m_spel.UserHasRight (SpelUserRights.EditPoints)
```

C# 例:

```
bool hasRight;  
hasRight = m_spel.UserHasRight (SpelUserRights.EditPoints);
```

VCal 方法, Spel 类

描述

该指令允许您执行视觉校准循环。

语法

```
Sub VCal (CalibName As String)
Sub VCal (CalibName As String, ByRef Status As Integer)
Sub VCal (CalibName As String, Parent As Form)
Sub VCal (CalibName As String, Parent As Form, ByRef Status As Integer)
```

参数

<i>CalibName</i>	评估当前项目中校准方案名称的字符串表达式。
<i>Status</i>	可选。接收校准状态的整数变量。 0: 失败, 1: 成功
<i>Parent</i>	可选。.NET 父窗体。

备注

执行 **VCal** 方法时, 机器人将移动。因此, 在执行 **VCal** 之前应确认操作员已准备就绪。

VCal 仅执行校准循环。不允许示教点。需使用 **VCalPoints** 示教点。此外, 必须首先在 EPSON RC+ 7.0 中设置校准。有关详细信息, 请参阅 **Vision Guide** 手册。

使用 **Status** 参数确认校准是否成功。若校准属性 **ShowConfirmation** 为 **True**, 会显示确认对话框。操作员点击<OK>按钮时, **Status** 返回 1: 成功。

另见

VCalPoints

VCal 示例**VB 例:**

```
Dim status As Integer
m_spel.VCal("CAMCAL1", status)
```

C# 例:

```
int status;
m_spel.VCal("CAMCAL1", status);
```

VCalPoints 方法, Spel 类

描述

该命令允许示教视觉校准点。

语法

Sub **VCalPoints** (*CalibName* As String)

Sub **VCalPoints** (*CalibName* As String, *ByRef Status* As Integer)

Sub **VCalPoints** (*CalibName* As String, *Parent* As Form)

Sub **VCalPoints** (*CalibName* As String, *ByRef Status* As Integer, *Parent* As Form)

参数

CalibName 评估当前项目中校准方案名称的字符串表达式。

Status 可选。接收点示教状态的整数变量。

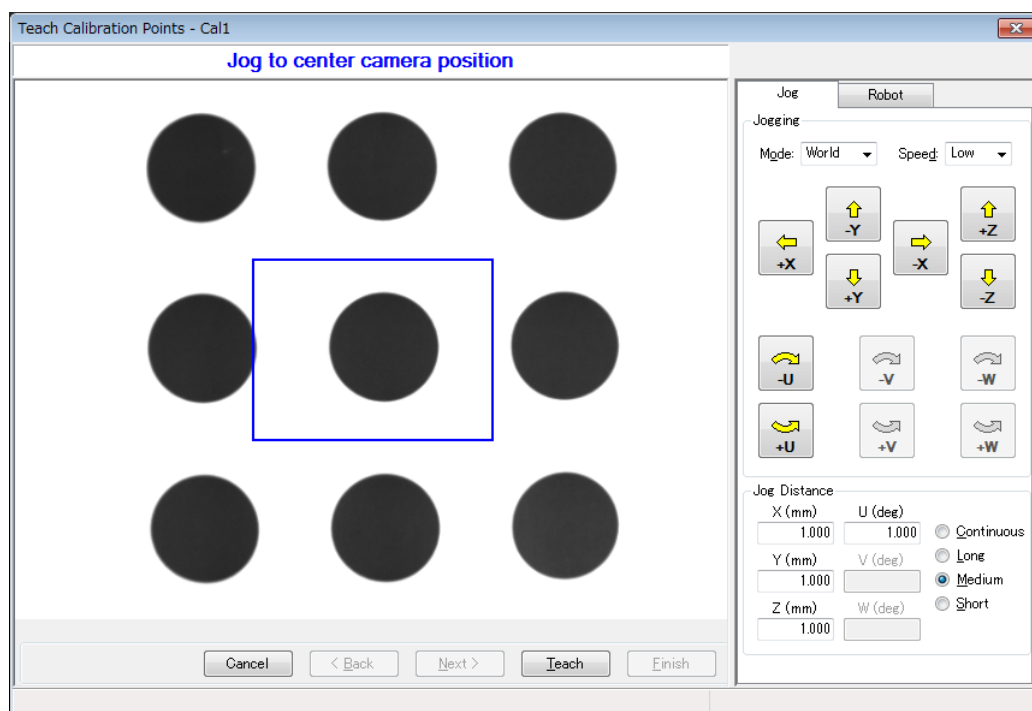
0: 失败, 1: 成功

Parent 可选。.NET 父窗体。

备注

执行 **VCalPoints** 命令时, 将打开示教校准点对话框。点击<Finish>按钮后, 将自动保存校准数据。

但必须已在 EPSON RC+ 7.0 中创建校准方案。



使用 *Status* 参数确认点示教是否成功。所有点已示教并点击了<Finish>按钮时, *Status* 返回 1: 成功。

另见

VCal

VCalPoints 示例

VB 例:

```
Dim status As Integer  
m_spel.VCalPoints("CAMCAL1", status)
```

C# 例:

```
int status;  
m_spel.VCalPoints("CAMCAL1", out status);
```

VClS 方法, Spel 类

描述

清除视觉图形。

语法

Sub VClS ()

备注

使用 VClS 方法清除视觉屏幕。

另见

VRun

VClS 示例**VB 例:**

```
m_spel.VClS ()
```

C# 例:

```
m_spel.VClS ();
```

VCreateCalibration 方法, Spel 类

描述

在当前项目中创建新的视觉校准。

语法

Sub **VCreateCalibration** (*CameraNumber* As Integer, *CalibName* As String)
Sub **VCreateCalibration** (*CameraNumber* As Integer, *CalibName* As String,
CopyCalibName As String)

参数

CameraNumber 含有待校准相机编号的整数表达式。

CalibName 含有待创建视觉校准名称的字符串表达式。

CopyCalibName 可选。含有待复制视觉校准名称的字符串表达式。

另见

VCreateObject, VCreateSequence, VDeleteCalibration

VCreateCalibration 示例**VB 例:**

```
m_spel.VCreateCalibration(1, "mycal")
```

C# 例:

```
m_spel.VCreateCalibration(1, "mycal");
```

VCreateObject 方法, Spel 类

描述

在当前项目中创建视觉对象。

语法

Sub **VCreateObject** (*Sequence* As String, *ObjectName* As String, *ObjectType* As SpelVisionObjectTypes)

参数

Sequence 含有当前项目中视觉序列名称的字符串表达式。

ObjectName 含有将在序列 *Sequence* 中创建的对象名称的字符串表达式。

ObjectType 指定视觉对象类型的 SpelVisionObjectTypes。
(也可使用以下所示的常量)

对象类型	常量	数值
Correlation	Correlation	1
Blob	Blob	2
Edge	Edge	3
Polar	Polar	4
Line	Line	5
Point	Point	6
Frame	Frame	7
ImageOp	ImageOp	8
Ocr	Ocr	9
CodeReader	CodeReader	10
Geometric	Geometric	11
Color Match	ColorMatch	14
Line Finder	LineFinder	15
Arc Finder	ArcFinder	16
Defect Finder	DefectFinder	17
Line Inspector	LineInspector	18
Arc Inspector	ArcInspector	19
Box Finder	BoxFinder	20
Corner Finder	CornerFinder	21
Contour	Contour	22
Text	Text	23
Decision	Decision	26
Coordinates	Coordinates	27

另见

VCreateSequence, VDeleteObject, VDeleteSequence

VCreateObject 示例**VB 例:**

```
m_spel.VCreateObject("myseq", "myblob",
SpelVisionObjectTypes.Blob)
```

C# 例:

```
m_spel.VCreateObject("myseq", "myblob",
SpelVisionObjectTypes.Blob);
```

VCreateSequence 方法, Spel 类

描述

在当前项目中创建新的视觉序列。

语法

```
Sub VCreateSequence (CameraNumber As Integer, SequenceName As String)
Sub VCreateSequence (CameraNumber As Integer, SequenceName As String,
    CopySequenceName As String)
```

参数

CameraNumber 含有待使用相机编号的整数表达式。

SequenceName 含有待创建视觉序列名称的字符串表达式。

CopySequenceName 可选。含有待复制视觉序列名称的字符串表达式。

另见

VCreateObject, VDeleteObject, VDeleteSequence

VCreateSequence 示例**VB 例:**

```
m_spel.VCreateSequence(1, "myseq")
```

C# 例:

```
m_spel.VCreateSequence(1, "myseq");
```


VDefArm 方法, Spel 类

描述

使用可由视觉系统检测的特征点计算移动相机的机械臂设置值。

NOTE:

机器人基于目标的检测结果自动操作。请注意机器人和外围设备之间的互相干扰。此外, 使用时避免各轴伸长至奇点附近的姿势, 以免机械臂设置时发生错误。

语法

```
Sub VDefArm (ArmNumber As Integer, ArmDefType As SpelArmDefType, ArmDefMode As SpelArmDefMode, Sequence As String, Rotation As Double, TargetTolerance As Double)
```

```
Sub VDefArm (ArmNumber As Integer, ArmDefType As SpelArmDefType, ArmDefMode As SpelArmDefMode, Sequence As String, Rotation As Double, TargetTolerance As Double, Parent As Form)
```

```
Sub VDefArm (ArmNumber As Integer, ArmDefType As SpelArmDefType, ArmDefMode As SpelArmDefMode, Sequence As String, Rotation As Double, TargetTolerance As Double, RobotSpeed As Integer, RobotAccel As Integer, ShowWarning As SpelVDefShowWarning)
```

```
Sub VDefArm (ArmNumber As Integer, ArmDefType As SpelArmDefType, ArmDefMode As SpelArmDefMode, Sequence As String, Rotation As Double, TargetTolerance As Double, RobotSpeed As Integer, RobotAccel As Integer, ShowWarning As SpelVDefShowWarning, Parent As Form)
```

参数

<i>ArmNumber</i>	含有要执行机械臂设置的机械臂编号的整数表达式(1 至 15)。
<i>ArmDefType</i>	含有机械臂类型的整数表达式。 J2Camera: 计算 J2 移动相机的图像中心。
<i>ArmDefMode</i>	含有机械臂设置模式的整数表达式。 Rough: 运行粗略机械臂设置的模式。 机器人将 1 mm 设置精度作为目标移动。 机器人动作较小。 Fine: 运行精细机械臂设置的模式。 机器人大幅移动, 并随机械臂方向改变, 提供更高精度的机械臂设置。
<i>Sequence</i>	含有当前项目中视觉序列名称的字符串表达式。
<i>Rotation</i>	含有粗略械臂设置时旋转角度(度)的实数表达式。 值的范围: 0 - 45
<i>TargetTolerance</i>	含有将视觉检测结果视为与目标位置一致的像素距离实数表达式。 值的范围: 0 - 3 像素
<i>Parent</i>	可选。窗口的 .NET 父窗体。
<i>RobotSpeed</i>	可选。将包含机器人速度(%)的整数变量。 值的范围: 0 - 100 如果省略, 则设为“5”。

<i>RobotAccel</i>	可选。将包含机器人加速度(%)的整数变量。 值的范围: 0 - 99 如果省略, 则设为“5”。
<i>ShowWarning</i>	可选。 <i>ArmSetMode</i> 为 Fine 时决定是否显示警告的整数变量。 Always :始终显示 DependsOnSpeed : <i>RobotSpeed</i> 或 <i>RobotAccel</i> 大于 5 时显示。 None :不显示 如果忽略, 则会设为“DependsOnSpeed”。

另见

VDefGetMotionRange, VDefLocal, VDefSetMotionRange, VDefTool, VGoCenter

VDefArm 示例

VB 例:

```
m_spel.VDefArm(1, SpelArmDefType.J2Camera,  
SpelArmDefMode.Rough, "myseq", 5, 1)
```

C# 例:

```
m_spel.VDefArm(1, SpelArmDefType.J2Camera,  
SpelArmDefMode.Rough, "myseq", 5, 1);
```

VDefGetMotionRange 方法, Spel 类

描述

获取由 VDefTool、VDefArm、VDefLocal 和 VGoCenter 限制的动作范围值。

语法

```
Sub VDefGetMotionRange(ByRef MaxMoveDist As Double, ByRef MaxPoseDiffAngle As Double, ByRef LjmMode As Integer)
```

参数

<i>MaxMoveDist</i>	表示移动最大距离的实数变量。若指定 0，则范围不受限制。(0 - 500, 默认值: 200) VDeopfTool、VDefArm、VDefLocal 和 VGoCenter 用于限制范围。
<i>MaxPoseDiffAngle</i>	表示工具方向(UVW)最大位移角度(度)的实数变量。若指定 0，则角度不受限制。仅影响 VDefLocal。(0 - 180, 默认值: 45 度)
<i>LjmMode</i>	表示 LJM 模式的整数变量。

另见

VDefTool, VDefArm, VDefLocal, VGoCenter, VDefSetMotionRange

VDefGetMotionRange 示例**VB 例:**

```
Dim maxMoveDist As Double
Dim maxPoseDiffAngle As Double
Dim ljmMode As Integer
m_spel.VDefGetMotionRange(maxMoveDist, maxPoseDiffAngle, ljmMode)
```

C# 例:

```
double maxMoveDist, maxPoseDiffAngle;
int ljmMode;
m_spel.VDefGetMotionRange(out maxMoveDist, out maxPoseDiffAngle, out ljmMode);
```

VDefLocal 方法, Spel 类

描述

通过移动相机检测放置于工作平面的校准板，并定义平行于工作平面的本地坐标。此外也通过固定相机检测工具末端的用户工件，并定义平行于固定相机传感器的本地平面。

NOTE:

机器人基于目标的检测结果自动操作。请注意机器人和外围设备之间的互相干扰。此外，使用时避免各轴伸长至奇点附近的姿势，以免本地坐标设置时发生错误。

语法

```
Sub VDefLocal(LocalNumber As Integer, LocalDefType As SpelLocalDefType,
  CalPlateType As SpelCalPlateType, Sequence As String, TargetTolerance As Double,
  CameraTool As Integer, RefPoint As SpelPoint)
```

```
Sub VDefLocal(LocalNumber As Integer, LocalDefType As SpelLocalDefType,
  CalPlateType As SpelCalPlateType, Sequence As String, TargetTolerance As Double,
  CameraTool As Integer, RefPoint As SpelPoint, Parent As Form)
```

```
Sub VDefLocal(LocalNumber As Integer, LocalDefType As SpelLocalDefType,
  CalPlateType As SpelCalPlateType, Sequence As String, TargetTolerance As Double,
  CameraTool As Integer, RefPoint As SpelPoint, RobotSpeed As Integer, RobotAccel As
  Integer)
```

```
Sub VDefLocal(LocalNumber As Integer, LocalDefType As SpelLocalDefType,
  CalPlateType As SpelCalPlateType, Sequence As String, TargetTolerance As Double,
  CameraTool As Integer, RefPoint As SpelPoint, RobotSpeed As Integer, RobotAccel As
  Integer, Parent As Form)
```

参数

<i>LocalNumber</i>	表示要设置本地坐标的工具编号的整数。(1 - 15)
<i>LocalDefType</i>	表示本地类型的整数。 J5Camera: 使用 J5 移动相机指定平行于校准板的本地坐标。 J6Camera: 使用 J6 移动相机指定平行于校准板的本地坐标。 FixedUpwardCamera: 使用向上固定相机指定平行于图像传感器的本地坐标。 FixedDownwardCamera: 使用向下固定相机指定平行于图像传感器的本地坐标。
<i>CalPlateType</i>	表示校准板类型的整数。 Large : 大校准板 Medium : 中校准板 Small : 小校准板 XSmall : 极小校准板
<i>Sequence</i>	表示当前项目中视觉序列名称的字符串表达式。 使用移动相机时，该项为拍摄校准板照片的视觉序列。 使用固定相机时，该项为检测用户工件等工具末端特征点的视觉序列。
<i>TargetTolerance</i>	表示用于判断尺寸一致的阈值的实数值。

<i>CameraTool</i>	固定相机：指定保持检测目标工具偏移的工具编号。若要执行自动校准，指定 -1。 J6 移动相机：若已执行自动校准，指定移动相机的工具编号。若要执行自动校准，指定 -1。 J5 移动相机：会将该可选设置忽略。
<i>RefPoint</i>	平行于工作平面的本地平面经过的点编号。 此点用于指定本地平面高度。
<i>Parent</i>	可选。窗口的 .NET 父窗体。
<i>RobotSpeed</i>	可选。将包含机器人速度(%)的整数变量。 值的范围：0 - 100 如果省略，则设为“5”。
<i>RobotAccel</i>	可选。将包含机器人加速度(%)的整数变量。 值的范围：0 - 99 如果省略，则设为“5”。

另见

VDefArm, VDefGetMotionRange, VDefSetMotionRange, VDefTool, VGoCenter

VDefLocal 示例**VB 例:**

```
Dim p2 = m_spel.GetPoint("P2")
m_spel.VDefLocal(1, SpelLocalDefType.J6Camera,
SpelCalPlateType.Large, "myseq", 1.0, 1, p2)
```

C# 例:

```
SpelPoint p2;
p2 = m_spel.GetPoint("P2");
m_spel.VDefLocal(1, SpelLocalDefType.J6Camera,
SpelCalPlateType.Large, "myseq", 1.0, 1, p2);
```

VDefSetMotionRange 方法, Spel 类

描述

由 VDefTool、VDefArm、VDefLocal 和 VGoCenter 限制动作范围。

语法

Sub **VDefSetMotionRange**(MaxMoveDist As Double, MaxPoseDiffAngle As Double, LjmMode As Integer)

参数

<i>MaxMoveDist</i>	表示移动最大距离的实数值。 若指定 0, 则范围不受限制。(0 至 500, 默认值: 200) VDefTool、VDefArm、VDefLocal 和 VGoCenter 用于限制范围。
<i>MaxPoseDiffAngle</i>	表示工具方向(UVW)最大位移角度(度)的实数值。 若指定 0, 则角度不受限制。 仅影响 VDefLocal。(0 - 180, 默认值: 45 度)
<i>LjmMode</i>	表示 LJM 模式的整数。

另见

VDefTool, VDefArm, VDefLocal, VGoCenter, VDefGetMotionRange

VDefSetMotionRange 示例**VB 例:**

```
m_spel.VDefSetMotionRange(100, 30, 1)
```

C# 例:

```
m_spel.VDefSetMotionRange(100, 30, 1);
```

VDefTool 方法, Spel 类

描述

使用视觉检测，计算 TPC 和移动相机位置的工具偏移值。

NOTE:

在除 FixedCameraWithCal 以外的工具类型时，机器人基于目标的检测结果自动操作。请注意机器人和外围设备之间的互相干扰。此外，使用时避免各轴伸长至奇点附近的姿势，以免工具设置时发生错误。

语法

Sub **VDefTool**(ToolNumber As Integer, ToolDefType As SpelToolDefType, Sequence As String, Object As String)

Sub **VDefTool**(ToolNumber As Integer, ToolDefType As SpelToolDefType, Sequence As String, Object As String, Parent As Form)

Sub **VDefTool**(ToolNumber As Integer, ToolDefType As SpelToolDefType, Sequence As String, FinalAngle As Double, InitAngle As Double, TargetTolerance As Double)

Sub **VDefTool**(ToolNumber As Integer, ToolDefType As SpelToolDefType, Sequence As String, FinalAngle As Double, InitAngle As Double, TargetTolerance As Double, Parent As Form)

Sub **VDefTool**(ToolNumber As Integer, ToolDefType As SpelToolDefType, Sequence As String, FinalAngle As Double, InitAngle As Double, TargetTolerance As Double, RobotSpeed As Integer, RobotAccel As Integer)

Sub **VDefTool**(ToolNumber As Integer, ToolDefType As SpelToolDefType, Sequence As String, FinalAngle As Double, InitAngle As Double, TargetTolerance As Double, RobotSpeed As Integer, RobotAccel As Integer, Parent As Form)

参数

<i>ToolNumber</i>	表示执行工具设置的工具编号的整数(1-15)
<i>ToolDefType</i>	表示工具类型的整数。 FixedCamera: 使用未校准的固定相机设置工具。 J4Camera: 计算 J4 移动相机的图像中心。 J6Camera: 计算 J6 移动相机的图像中心。 FixedCameraWithCal: 使用已校准的固定相机设置工具。
<i>Sequence</i>	表示当前项目中视觉序列名称的字符串表达式。
<i>Object</i>	表示指定序列中视觉对象的字符串表达式。该参数在 <i>ToolDefType</i> 为 FixedCameraWithCal 时需要。 <i>ToolDefType</i> 不为 FixedCameraWithCal 时，对象应为空字符串。
<i>FinalAngle</i>	表示工具或相机工具旋转角度(度)的实数值。 值的范围: 0, 5 - 180, -5 - -180 如果省略，则设为“90”。
<i>InitAngle</i>	表示临时工具设置时工具或相机工具旋转角度(度)的实数值。 该值必须小于 <i>FinalAngle</i> 。 值的范围: -10 - 10 如果省略，则设为“5”。
<i>TargetTolerance</i>	表示将视觉检测结果视为与目标位置一致的像素距离的实数值。 值的范围: 0 - 3 pixels 如果省略，则设为“1”。

<i>Parent</i>	可选。窗口的 .NET 父窗体。
<i>RobotSpeed</i>	可选。将包含机器人速度(%)的整数变量。 值的范围：0 - 100 如果省略，则设为“5”。
<i>RobotAccel</i>	可选。将包含机器人加速度(%)的整数变量。 值的范围：0 - 99 如果省略，则设为“5”。

另见

VDefArm, VDefGetMotionRange, VDefLocal, VDefSetMotionRange, VGoCenter

VDefTool 示例

VB 例:

```
m_spel.VDefTool(1, SpelToolDefType.J6Camera, "myseq", 45, 5, 3.0)
m_spel.VDefTool(1, SpelToolDefType.FixedCameraWithCal, "myseq", "myobj")
```

C# 例:

```
m_spel.VDefTool(1, SpelToolDefType.J6Camera, "myseq", 45, 5, 3.0);
m_spel.VDefTool(1, SpelToolDefType.FixedCameraWithCal, "myseq", "myobj");
```


VDefToolXYZ 方法, Spel 类

描述

VDefToolXYZ 可以使用视觉检测来计算工具偏移值 (XYZ)。

注意:

机器人会根据目标的检测结果自动运行。请注意机器人与周边设备的干涉。为避免工具偏移导致的错误, 请避免通过各关节延伸时的奇点附近。

语法

```
Sub VDefToolXYZ(ToolNumber As Integer, LocalNumber As Integer, PointNumber1 As Integer, PointNumber2 As Integer, Sequence1 As String, Sequence2 As String, FinalAngle As Double, InitAngle As Double, TargetTolerance As Double, RobotSpeed As Integer, RobotAccel As Integer)
```

```
Sub VDefToolXYZ(ToolNumber As Integer, LocalNumber As Integer, PointNumber1 As Integer, PointNumber2 As Integer, Sequence1 As String, Sequence2 As String, FinalAngle As Double, InitAngle As Double, TargetTolerance As Double, RobotSpeed As Integer, RobotAccel As Integer, Parent As Form)
```

参数

<i>ToolNumber</i>	表示执行工具设置的工具编号的整数(1-15)
<i>LocalNumber</i>	表示要移动机器人的本地坐标编号的整数。工具会移动到指定的本地坐标 XY 平面。
<i>PointNumber1</i>	表示 Point Number 第一姿态的整数
<i>PointNumber2</i>	表示 Point Number 第二姿态的整数
<i>Sequence1</i>	表示当前项目中第一姿态的视觉序列名称的字串表达式
<i>Sequence2</i>	表示当前项目中第二姿态的视觉序列名称的字串表达式
<i>FinalAngle</i>	表示工具和相机工具旋转角度(度)的实数值 值的范围: 5 ~ 180, □5 ~ □180
<i>InitAngle</i>	表示临时工具的旋转角度(度)的实数值 该值必须小于 FinalAngle 值的范围: 0.01 ~ 10, □0.01 ~ □10
<i>TargetTolerance</i>	表示像素距离的实数, 该像素是检测结果与目标位置匹配的阈值像素。 值的范围: 0.1 ~ 3.0 pixel
<i>RobotSpeed</i>	表示机器人速度的 (%)整数 值的范围: 1 ~ 100
<i>RobotAccel</i>	表示机器人加速度的整数变量 (%) 值的范围: 1 ~ 99
<i>Parent</i>	.NET 表单作为窗口的父级(可省略)

另见

VDefTool, VDefToolXYZUVW

VDefToolXYZ 示例**VB 例:**

```
m_spel.VDefToolXYZ(1, 0, 1, 2, "seq01", "seq02", 5, 30, 1, 5, 5)
m_spel.VDefToolXYZ(2, 0, 3, 4, "seq03", "seq04", 5, 30, 1, 5, 5)
m_spel.VDefToolXYZ(3, 0, 5, 6, "seq05", "seq06", 5, 30, 1, 5, 5)
m_spel.VDefToolXYZUVW(1, 2, 3, SpelToolDefType3D.Bar)
```

C# 例:

```
m_spel.VDefToolXYZ(1, 0, 1, 2, "seq01", "seq02", 5, 30, 1, 5, 5);
m_spel.VDefToolXYZ(2, 0, 3, 4, "seq03", "seq04", 5, 30, 1, 5, 5);
m_spel.VDefToolXYZ(3, 0, 5, 6, "seq05", "seq06", 5, 30, 1, 5, 5);
m_spel.VDefToolXYZUVW(1, 2, 3, SpelToolDefType3D.Bar);
```

VDefToolXYZUVW 方法, Spel 类

描述

用 3 个工具定义来计算工具 UVW 偏移值。

注意:

计算出来的 U, V, W 工具偏移值, 会被设定给指定的工具 1。工具 1 的 X, Y, Z 偏移值不变。

语法

Sub **VDefToolXYZUVW**(ToolNumber1 As Integer, ToolNumber2 As Integer, ToolNumber3 As Integer, ToolDefType3D As SpelToolDefType3D)

参数

- ToolNumber1* 表示第 1 个工具定义的工具编号的整数变量 (1 ~ 15)
在条状类型中, 指定表示工具前端的工具编号。
在平面类型中, 指定表示工具中心的工具编号。
- ToolNumber2* 表示第 2 个工具定义的工具编号的整数变量(1 ~ 15)
在条状类型中, 指定表示工具中央的编号。
在平面类型中, 指定表示工具中心以外的, 与 ToolNumber3 不同的编号。
- ToolNumber3* 表示第 3 个工具定义的工具编号的整数变量(1 ~ 15)
在条状类型中, 指定表示工具末端的工具编号。
在平面类型中, 指定表示工具中心以外的, 与 ToolNumber2 不同的编号。
- ToolDefType3D* 表示工具定义的工具类型的整数变量
Bar: 条状类型
Plane: 平面类型
- ToolNumber* 表示执行工具设置的工具编号的整数(1-15)

另见

VDefToolXYZ

VDefToolXYZUVW 示例**VB 例:**

```
m_spel.VDefToolXYZ (1, 0, 1, 2, "seq01", "seq02", 5, 30, 1, 5, 5)
m_spel.VDefToolXYZ (2, 0, 3, 4, "seq03", "seq04", 5, 30, 1, 5, 5)
m_spel.VDefToolXYZ (3, 0, 5, 6, "seq05", "seq06", 5, 30, 1, 5, 5)
m_spel.VDefToolXYZUVW (1, 2, 3, SpelToolDefType3D.Bar)
```

C# 例:

```
m_spel.VDefToolXYZ (1, 0, 1, 2, "seq01", "seq02", 5, 30, 1, 5, 5);
m_spel.VDefToolXYZ (2, 0, 3, 4, "seq03", "seq04", 5, 30, 1, 5, 5);
m_spel.VDefToolXYZ (3, 0, 5, 6, "seq05", "seq06", 5, 30, 1, 5, 5);
m_spel.VDefToolXYZUVW (1, 2, 3, SpelToolDefType3D.Bar);
```

VDeleteCalibration 方法, Spel 类

描述

删除当前项目中的视觉校准。

语法

Sub **VDeleteCalibration** (*CalibName* As String)

参数

CalibName 含有当前项目中视觉校准名称的字符串表达式。

另见

VCreateCalibration, VDeleteObject, VDeleteSequence

VDeleteCalibration 示例

VB 例:

```
m_spel.VDeleteCalibration("mycal")
```

C# 例:

```
m_spel.VDeleteCalibration("mycal");
```

VDeleteObject 方法, Spel 类

描述

删除当前项目中的视觉对象。

语法

Sub **VDeleteObject** (*Sequence* As String, *ObjectName* As String)

参数

Sequence 含有当前项目中视觉序列名称的字符串表达式。

ObjectName 含有当前项目中视觉对象名称的字符串表达式。

另见

VCreateObject, VCreateSequence, VDeleteSequence

VDeleteObject 示例**VB 例:**

```
m_spel.VDeleteObject("myseq", "myobj")
```

C# 例:

```
m_spel.VDeleteObject("myseq", "myobj");
```

VDeleteSequence 方法, Spel 类

描述

删除当前项目中的视觉序列。

语法

Sub **VDeleteSequence** (*Sequence* As String)

参数

Sequence 含有当前项目中视觉序列名称的字符串表达式。

另见

VCreateObject, VCreateSequence, VDeleteObject

VDeleteSequence 示例

VB 例:

```
m_spel.VDeleteSequence("myseq")
```

C# 例:

```
m_spel.VDeleteSequence("myseq");
```

VEditWindow 方法, Spel 类

描述

编辑搜索窗口中的不要紧的像素。有关详细信息，请参阅 Vision Guide Properties and Results Reference 手册中的“EditWindow 属性”。

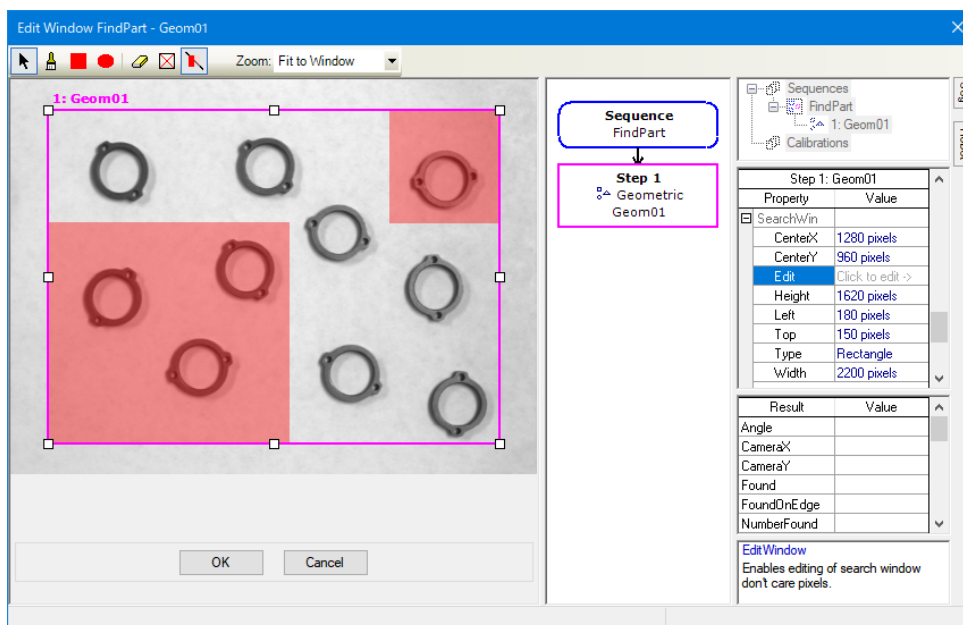
语法

Sub **VEditWindow** (Sequence As String, Object As String)

参数

Sequence 表示当前程序的视觉序列名称的字符串表达式

ObjectName 表示当前程序的视觉对象名称的字符串表达式

**另见**

VSave

VEditWindow 示例**VB 例:**

```
m_spel.VEditWindow("myseq", "myobj")
```

C# 例:

```
m_spel.VEditWindow("myseq", "myobj");
```

VGet 方法, Spel 类

描述

获取视觉序列或对象属性或结果的值。

语法

```
Sub VGet (Sequence As String, PropCode As SpelVisionProps, ByRef Value As Integer)
Sub VGet (Sequence As String, PropCode As SpelVisionProps, ByRef Value As Boolean)
Sub VGet (Sequence As String, PropCode As SpelVisionProps, ByRef Value As Double)
Sub VGet (Sequence As String, PropCode As SpelVisionProps, ByRef Value As Single)
Sub VGet (Sequence As String, PropCode As SpelVisionProps, ByRef Value As String)
Sub VGet ( Sequence As String, Object As String, PropCode As SpelVisionProps,
    ByRef Value As Integer )
Sub VGet (Sequence As String, Object As String, PropCode As SpelVisionProps,
    ByRef Value As Boolean)
Sub VGet (Sequence As String, Object As String, PropCode As SpelVisionProps,
    ByRef Value As Color)
Sub VGet (Sequence As String, Object As String, PropCode As SpelVisionProps,
    ByRef Value As Double)
Sub VGet (Sequence As String, Object As String, PropCode As SpelVisionProps,
    ByRef Value As Single)
Sub VGet (Sequence As String, Object As String, PropCode As SpelVisionProps,
    ByRef Value As String)
Sub VGet (Sequence As String, Object As String, PropCode As SpelVisionProps,
    Result As Integer, ByRef Value As Integer)
Sub VGet (Sequence As String, Object As String, PropCode As SpelVisionProps,
    Result As Integer, ByRef Value As Boolean)
Sub VGet (Sequence As String, Object As String, PropCode As SpelVisionProps,
    Result As Integer, ByRef Value As Double)
Sub VGet (Sequence As String, Object As String, PropCode As SpelVisionProps,
    Result As Integer, ByRef Value As Single)
Sub VGet (Sequence As String, Object As String, PropCode As SpelVisionProps,
    Result As Integer, ByRef Value As String)
```

参数

<i>Sequence</i>	含有当前项目中视觉序列名称的字符串表达式。
<i>Object</i>	含有序列 <i>Sequence</i> 中对象名称的字符串表达式。如果属性用于序列，则此字符串必须为空。
<i>PropCode</i>	指定属性代码的 SpelVisionProps 值。
<i>Result</i>	表示结果编号的整数表达式。
<i>Value</i>	含有属性或结果值的变量。变量类型必须与属性或结果类型匹配。

另见

VSet, VRun

VGet 示例**VB 例:**

```
Dim i As Integer
Redim score(10) As Integer

m_spel.VRun("testSeq")
For i = 1 to 10
    m_spel.VGet("testSeq", "corr" & Format$(i, "00"), _
        SpelVisionProps.Score, score(i))
Next i
```

C# 例:

```
int[] score = new int[11];
for(int i = 1; i <= 10; i++)
{
    m_spel.VGet("testSeq", string.Format("Corr 0{0}", i),
        SpelVisionProps.Score, score(i));
}
```

VGetCameraXYU 方法, Spel 类

描述

检索任一对象的相机 X、Y 和 U 物理坐标。

语法

Sub **VGetCameraXYU** (*Sequence* As String, *Object* As String, *Result* As Integer, ByRef *Found* As Boolean, ByRef *X* As Single, ByRef *Y* As Single, ByRef *U* As Single)

参数

<i>Sequence</i>	含有当前项目中视觉序列名称的字符串表达式。
<i>Object</i>	含有序列 <i>Sequence</i> 中对象名称的字符串表达式。
<i>Result</i>	表示结果编号的整数表达式。
<i>Found</i>	将含有是否找到对象的 Boolean 变量。
<i>X</i>	将包含以毫米为单位的 x 坐标的实数变量。
<i>Y</i>	将包含以毫米为单位的 y 坐标的实数变量。
<i>U</i>	将包含以度为单位的角度的实数变量。

另见

VGetPixelXYU, VGetRobotXYU

VGetCameraXYU 示例**VB 例:**

```
Dim found As Boolean
Dim x As Single, y As Single, u As Single
Dim seq As String, blob As String

seq = "testSeq"
blob = "blob01"
m_spel.VRun(seq)
m_spel.VGetCameraXYU(seq, blob, 1, found, x, y, u)
```

C# 例:

```
bool found;
float x, y, u;
string seq, blob;

seq = "testSeq";
blob = "blob01";
m_spel.VRun(seq);
m_spel.VGetCameraXYU(seq, blob, 1, out found, out x, out y, out u);
```

VGetEdgeCameraXYU 方法, Spel 类

描述

检索 Line Finder 和 Arc Finder 搜索中各边的相机 X、Y 和 U 物理坐标。

语法

Sub **VGetEdgeCameraXYU** (*Sequence* As String, *Object* As String, *EdgeResultIndex* As Integer, *ByRef Found* As Boolean, *ByRef X* As Single, *ByRef Y* As Single, *ByRef U* As Single)

参数

<i>Sequence</i>	含有当前项目中视觉序列名称的字符串表达式。
<i>Object</i>	含有序列 <i>Sequence</i> 中对象名称的字符串表达式。
<i>EdgeResultIndex</i>	表示 Edge 结果索引的整数表达式。
<i>Found</i>	将含有是否找到对象的 Boolean 变量。
<i>X</i>	将包含以毫米为单位的 x 坐标的实数变量。
<i>Y</i>	将包含以毫米为单位的 y 坐标的实数变量。
<i>U</i>	将包含以度为单位的角度的实数变量。

另见

VGetEdgePixelXYU, VGetEdgeRobotXYU, VGetPixelXYU, VGetRobotXYU

VGetEdgeCameraXYU 示例**VB 例:**

```
Dim found(10) As Boolean
Dim x(10) As Single, y(10) As Single, u(10) As Single
Dim seq As String, lineFinder As String

seq = "testSeq"
lineFinder = "LineFind01"
m_spel.VRun(seq)
' LineFinder 的 NumberOfEdges 为 10
For i = 1 To 10
    m_spel.VGetEdgeCameraXYU(seq, lineFinder, i, found(i),
    x(i),
        y(i), u(i))
Next i
```

C# 例:

```
bool[] found = new bool[11];
float[] x = new float[11];
float[] y = new float[11];
float[] u = new float[11];
string seq, lineFinder;
seq = "testSeq";
lineFinder = "LineFind01";
m_spel.VRun(seq);
// LineFinder 的 NumberOfEdges 为 10
for(int i = 1; i <= 10; i++)
    m_spel.VGetEdgeCameraXYU(seq, lineFinder, i, out found[i],
        out x[i], out y[i], out u[i]);
```

VGetEdgePixelXYU 方法, Spel 类

描述

检索 Line Finder 和 Arc Finder 搜索中各边的 X、Y 和 U 像素坐标。

语法

Sub **VGetEdgePixelXYU** (*Sequence* As String, *Object* As String, *EdgeResultIndex* As Integer, *ByRef Found* As Boolean, *ByRef X* As Single, *ByRef Y* As Single, *ByRef U* As Single)

参数

<i>Sequence</i>	含有当前项目中视觉序列名称的字符串表达式。
<i>Object</i>	含有序列 <i>Sequence</i> 中对象名称的字符串表达式。
<i>EdgeResultIndex</i>	表示 Edge 结果索引的整数表达式。
<i>Found</i>	将含有是否找到对象的 Boolean 变量。
<i>X</i>	将包含以毫米为单位的 x 坐标的实数变量。
<i>Y</i>	将包含以毫米为单位的 y 坐标的实数变量。
<i>U</i>	将包含以度为单位的角度的实数变量。

另见

VGetEdgeCameraXYU, VGetEdgeRobotXYU, VGetPixelXYU, VGetRobotXYU

VGetEdgePixelXYU 示例**VB 例:**

```
Dim found(10) As Boolean
Dim x(10) As Single, y(10) As Single, u(10) As Single
Dim seq As String, lineFinder As String

seq = "testSeq"
lineFinder = "LineFind01"
m_spel.VRun(seq)
' LineFinder 的 NumberOfEdges 为 10
For i = 1 To 10
    m_spel.VGetEdgePixelXYU(seq, lineFinder, i, found(i),
        x(i),
        y(i), u(i))
Next i
```

C# 例:

```
bool[] found = new bool[11];
float[] x = new float[11];
float[] y = new float[11];
float[] u = new float[11];
string seq, lineFinder;
seq = "testSeq";
lineFinder = "LineFind01";
m_spel.VRun(seq);
// LineFinder 的 NumberOfEdges 为 10
for(int i = 1; i <= 10; i++)
    m_spel.VGetEdgePixelXYU(seq, lineFinder, i, out found[i],
        out x[i], out y[i], out u[i]);
```

VGetEdgeRobotXYU 方法, Spel 类

描述

检索 Line Finder 和 Arc Finder 搜索中各边的机器人 X、Y 和 U 物理坐标。

语法

Sub **VGetEdgeRobotXYU** (*Sequence* As String, *Object* As String, *EdgeResultIndex* As Integer, *ByRef Found* As Boolean, *ByRef X* As Single, *ByRef Y* As Single, *ByRef U* As Single)

参数

<i>Sequence</i>	含有当前项目中视觉序列名称的字符串表达式。
<i>Object</i>	含有序列 <i>Sequence</i> 中对象名称的字符串表达式。
<i>EdgeResultIndex</i>	表示 Edge 结果索引的整数表达式。
<i>Found</i>	将含有是否找到对象的 Boolean 变量。
<i>X</i>	将包含以毫米为单位的 x 坐标的实数变量。
<i>Y</i>	将包含以毫米为单位的 y 坐标的实数变量。
<i>U</i>	将包含以度为单位的角度的实数变量。

另见

VGetEdgeCameraXYU, VGetEdgePixelXYU, VGetPixelXYU, VGetRobotXYU

VGetEdgeRobotXYU 示例**VB 例:**

```
Dim found(10) As Boolean
Dim x(10) As Single, y(10) As Single, u(10) As Single
Dim seq As String, lineFinder As String

seq = "testSeq"
lineFinder = "LineFind01"
m_spel.VRun(seq)
' LineFinder 的 NumberOfEdges 为 10
For i = 1 To 10
    m_spel.VGetEdgeRobotXYU(seq, lineFinder, i, found(i),
        x(i),
        y(i), u(i))
Next i
```

C# 例:

```
bool[] found = new bool[11];
float[] x = new float[11];
float[] y = new float[11];
float[] u = new float[11];
string seq, lineFinder;
seq = "testSeq";
lineFinder = "LineFind01";
m_spel.VRun(seq);
// LineFinder 的 NumberOfEdges 为 10
for(int i = 1; i <= 10; i++)
    m_spel.VGetEdgeRobotXYU(seq, lineFinder, i, out found[i],
        out x[i], out y[i], out u[i]);
```

VGetExtrema 方法, Spel 类

描述

检索 blob 对象的极坐标。

语法

Sub **VGetExtrema** (*Sequence* As String, *Object* As String, *Result* As Integer, ByRef *MinX* As Single, ByRef *MaxX* As Single, ByRef *MinY* As Single, ByRef *MaxY* As Single)

参数

<i>Sequence</i>	含有当前项目中视觉序列名称的字符串表达式。
<i>Object</i>	含有序列 <i>Sequence</i> 中对象名称的字符串表达式。
<i>Result</i>	表示结果编号的整数表达式。
<i>MinX</i>	将包含以像素为单位的最小 x 坐标的实数变量。
<i>MaxX</i>	将包含以像素为单位的最大 x 坐标的实数变量。
<i>MinY</i>	将包含以像素为单位的最小 y 坐标的实数变量。
<i>MaxY</i>	将包含以像素为单位的最大 y 坐标的实数变量。

另见

VGet

VGetExtrema 示例**VB 例:**

```
Dim xmin As Single, xmax As Single
Dim ymin As Single, ymax As Single
Dim seq As String, blob As String

seq = "testSeq"
blob = "blob01"
m_spel.VRun(seq)
m_spel.VGet(seq, blob, "found", found)
If found <> 0 Then
    m_spel.VGetExtrema(seq, blob, xmin, xmax, ymin, ymax)
End If
```

C# 例:

```
float xmin, xmax, ymin, ymax;
bool found;
string seq, blob;

seq = "testSeq";
blob = "blob01";
m_spel.VRun(seq);
m_spel.VGet(seq, blob, "found", found);

if(found == true)
    m_spel.VGetExtrema(seq, blob, out xmin, out xmax, out ymin,
        out ymax);
```

VGetModelWin 方法, Spel 类

描述

检索对象的模型窗口坐标。

语法

```
Sub VGetModelWin (Sequence As String, Object As String, ByRef Left As Integer,
                  ByRef Top As Integer, ByRef Width As Integer, ByRef Height As Integer)
```

参数

<i>Sequence</i>	含有当前项目中视觉序列名称的字符串表达式。
<i>Object</i>	含有序列 <i>Sequence</i> 中对象名称的字符串表达式。
<i>Left</i>	将包含以像素为单位的左坐标的整数变量。
<i>Top</i>	将包含以像素为单位的顶坐标的整数变量。
<i>Width</i>	将包含以像素为单位的宽度的整数变量。
<i>Height</i>	将包含以像素为单位的高度的整数变量。

另见

VSetModelWin, VGetSearchWin, VSetSearchWin

VGetModelWin 示例**VB 例:**

```
Dim left As Integer, top As Integer
Dim width As Integer, height As Integer

With m_spel
    .VGetModelWin("testSeq", "corr01", left, top, _
                  width, height)
    .VSetModelWin("testSeq", "corr01", left + 20, top, _
                  width, height)
    .VTeach("testSeq", "corr01")
End With
```

C# 例:

```
int left, top, width, height;

m_spel.VGetModelWin("testSeq", "corr01", out left, out top,
                    out width, out height);
m_spel.VSetModelWin("testSeq", "corr01", left + 20, top,
                    width, height);
m_spel.VTeach("testSeq", "corr01");
```

VGetPixelToCamera 方法, Spel 类

描述

获取指定像素坐标的相机坐标。

语法

Sub **VGetPixelToCamera** (Calibration As String, PixelX As Single, PixelY As Single, Angle As Single, ByRef CameraX As Single, ByRef CameraY As Single, ByRef CameraU As Single)

参数

<i>Calibration</i>	表示当前程序的校准名称的字符串表达式
<i>PixelX</i>	表示以像素单位显示的 x 坐标值的实数变量
<i>PixelY</i>	表示以像素单位显示的 y 坐标值的实数变量
<i>Angle</i>	表示以度为单位显示的角度值的实数变量
<i>CameraX</i>	表示以毫米为单位显示的 x 坐标的实数变量
<i>CameraY</i>	表示以毫米为单位显示的 y 坐标的实数变量
<i>CameraU</i>	表示以度为单位显示的角度值的实数变量

另见

VGetPixelXYU, VGetCameraXYU, VGetPixelToRobot

VGetPixelToCamera 示例**VB 例:**

```
Dim x As Single, y As Single, u As Single
Dim cal As String, seq As String

cal = "testCal"
seq = "testSeq"
m_spel.VRun(seq)
m_spel.VGetPixelToCamera(cal, 400, 300, 0, x, y, u)
```

C# 例:

```
float x, y, u
string cal, seq;

cal = "testCal";
seq = "testSeq";
m_spel.VRun(seq);
m_spel.VGetPixelToCamera(cal, 400, 300, 0, out x, out y, out u);
```


VGetPixelToRobot 方法, Spel 类

描述

获取指定像素坐标的机器人坐标。

语法

Sub **VGetPixelToRobot** (Calibration As String, PixelX As Single, PixelY As Single, Angle As Single, ByRef RobotX As Single, ByRef RobotY As Single, ByRef RobotU As Single)

参数

<i>Calibration</i>	表示当前程序的校准名称的字符串表达式
<i>PixelX</i>	表示以像素单位显示的 x 坐标值的实数变量
<i>PixelY</i>	表示以像素单位显示的 y 坐标值的实数变量
<i>Angle</i>	表示以度为单位显示的角度值的实数变量
<i>CameraX</i>	表示以毫米为单位显示的 x 坐标的实数变量
<i>CameraY</i>	表示以毫米为单位显示的 y 坐标的实数变量
<i>CameraU</i>	表示以度为单位显示的角度值的实数变量

另见

VGetPixelXYU, VGetRobotXYU, VGetPixelToCamera

VGetPixelToRobot 示例**VB 例:**

```
Dim x As Single, y As Single, u As Single
Dim cal As String, seq As String

cal = "testCal"
seq = "testSeq"
m_spel.VRun(seq)
m_spel.VGetPixelToRobot(cal, 400, 300, 0, x, y, u)
```

C# 例:

```
float x, y, u
string cal, seq;

cal = "testCal";
seq = "testSeq";
m_spel.VRun(seq);
m_spel.VGetPixelToRobot(cal, 400, 300, 0, out x, out y, out u);
```

VGetPixelXYU 方法, Spel 类

描述

检索任一对象的像素 X、Y 和 U 坐标。

语法

Sub **VGetPixelXYU** (*Sequence* As String, *Object* As String, *Result* As Integer, ByRef *Found* As Boolean, ByRef *X* As Single, ByRef *Y* As Single, ByRef *U* As Single)

参数

<i>Sequence</i>	含有当前项目中视觉序列名称的字符串表达式。
<i>Object</i>	含有序列 <i>Sequence</i> 中对象名称的字符串表达式。
<i>Result</i>	表示结果编号的整数表达式。
<i>Found</i>	将含有是否找到对象的 Boolean 变量。
<i>X</i>	将包含以像素为单位的 x 坐标的实数变量。
<i>Y</i>	将包含以像素为单位的 y 坐标的实数变量。
<i>U</i>	将包含以度为单位的角度的实数变量。

另见

VGetCameraXYU, VGetRobotXYU

VGetPixelXYU 示例**VB 例:**

```
Dim found As Integer
Dim x As Single, y As Single, u As Single
Dim seq As String, blob As String

seq = "testSeq"
blob = "blob01"
m_spel.VRun(seq)
m_spel.VGetPixelXYU(seq, blob, 1, found, x, y, u)
```

C# 例:

```
int found;
float x, y, u;
string seq, blob;

seq = "testSeq";
blob = "blob01";
m_spel.VRun(seq);
m_spel.VGetPixelXYU(seq, blob, 1, out found, out x, out y,
out u);
```

VGetRobotPlacePos 方法, Spel 类

描述

检索机器人放置位置。

语法

Sub **VGetRobotPlacePos** (*Sequence* As String, *Object* As String, *Result* As Integer, ByRef *Found* As Boolean, ByRef *PlacePoint* As SpelPoint)

参数

<i>Sequence</i>	含有当前项目中视觉序列名称的字符串表达式。
<i>Object</i>	含有序列 <i>Sequence</i> 中对象名称的字符串表达式。
<i>Result</i>	表示结果编号的整数表达式。
<i>Found</i>	将包含 boolean 查找状态的整数变量。如果 found 为 false , 则不会定义 <i>x</i> 、 <i>y</i> 和 <i>u</i> 。
<i>PlacePoint</i>	含有放置位置的 SpelPoint 变量

另见

VGetRobotPlaceTargetPos, VSetRobotPlaceTargetPos

VGetRobotPlacePos 示例**VB 例:**

```
Dim found As Integer
Dim x As Single, y As Single, u As Single
Dim seq As String, blob As String
Dim placePoint As SpelPoint

seq = "testSeq"
blob = "blob01"
' 将工件向上移动到相机上方
m_spel.Jump("camPos")
m_spel.VRun(seq)
m_spel.VGetRobotPlacePos(seq, blob, 1, found, placePoint)
' 使用 SCARA 时使用 +TLZ 结束
m_spel.Jump(placePoint, "+TLZ(10)")
m_spel.Go(placePoint)
```

C# 例:

```
bool found;
float x, y, u;
string seq, blob;
SpelPoint placePoint = new SpelPoint();
seq = "testSeq";
blob = "blob01";

// 将工件向上移动到相机上方
m_spel.Jump("camPos");
m_spel.VRun(seq);
m_spel.VGetRobotPlacePos(seq, blob, 1, out found, out
placePoint);
// 使用 SCARA 时使用 +TLZ 结束
m_spel.Jump(placePoint, "+TLZ(10)");
m_spel.Go(placePoint);
```

VGetRobotPlaceTargetPos 方法, Spel 类

描述

检索部件放置位置。

语法

```
Sub VGetRobotPlaceTargetPos (Sequence As String, Object As String, ByRef Point As SpelPoint)
```

参数

Sequence 含有当前项目中视觉序列名称的字符串表达式。

Object 含有序列 *Sequence* 中对象名称的字符串表达式。

Point 含有放置位置的 SpelPoint 变量。

另见

VGetRobotPlacePos, VSetRobotPlaceTargetPos

VGetRobotPlaceTargetPos 示例**VB 例:**

```
Dim seq As String, blob As String
Dim targetPoint As SpelPoint

seq = "testSeq"
blob = "blob01"
m_spel.VGetRobotPlaceTargetPos(seq, blob, targetPoint)

' 调整放置位置
targetPoint.X = targetPoint.X + 10
m_spel.VSetRobotPlaceTargetPos(seq, blob, targetPoint)
```

C# 例:

```
string seq, blob;
SpelPoint targetPoint = new SpelPoint();

seq = "testSeq";
blob = "blob01";
m_spel.VGetRobotPlaceTargetPos(seq, blob, out targetPoint);

// 调整放置位置
targetPoint.X = targetPoint.X + 10;
m_spel.VSetRobotPlaceTargetPos(seq, blob, targetPoint);
```

VGetRobotXYU 方法, Spel 类

描述

检索任一对象的机器人全局 X、Y 和 U 坐标。

语法

Sub **VGetRobotXYU** (*Sequence* As String, *Object* As String, *Result* As Integer, ByRef *Found* As Boolean, ByRef *X* As Single, ByRef *Y* As Single, ByRef *U* As Single)

参数

<i>Sequence</i>	含有当前项目中视觉序列名称的字符串表达式。
<i>Object</i>	含有序列 <i>Sequence</i> 中对象名称的字符串表达式。
<i>Result</i>	表示结果编号的整数表达式。
<i>Found</i>	将包含 boolean 查找状态的整数变量。如果 found 为 false , 则不会定义 <i>x</i> 、 <i>y</i> 和 <i>u</i> 。
<i>X</i>	将包含以毫米为单位的 <i>x</i> 坐标的实数变量。
<i>Y</i>	将包含以毫米为单位的 <i>y</i> 坐标的实数变量。
<i>U</i>	将包含以度为单位的角度的实数变量。

另见

VGetCameraXYU, VGetPixelXYU

VGetRobotXYU 示例**VB 例:**

```
Dim found As Integer
Dim x As Single, y As Single, u As Single
Dim seq As String, blob As String

seq = "testSeq"
blob = "blob01"
m_spel.VRun(seq)
m_spel.VGetRobotXYU(seq, blob, 1, found, x, y, u)
```

C# 例:

```
bool found;
float x, y, u;
string seq, blob;

seq = "testSeq";
blob = "blob01";
m_spel.VRun(seq);
m_spel.VGetRobotXYU(seq, blob, 1, out found, out x, out y,
out u);
```

VGetRobotToolXYU 方法, Spel 类

描述

检索工具定义中的机器人全局 X、Y 和 U 值。

语法

Sub **VGetRobotToolXYU** (*Sequence* As String, *Object* As String, *Result* As Integer, *ByRef Found* As Boolean, *ByRef X* As Single, *ByRef Y* As Single, *ByRef U* As Single)

参数

<i>Sequence</i>	含有当前项目中视觉序列名称的字符串表达式。
<i>Object</i>	含有序列 <i>Sequence</i> 中对象名称的字符串表达式。
<i>Result</i>	表示结果编号的整数表达式。
<i>Found</i>	将包含 boolean 查找状态的整数变量。如果 found 为 false , 则不会定义 <i>x</i> 、 <i>y</i> 和 <i>u</i> 。
<i>X</i>	将包含以毫米为单位的 <i>x</i> 坐标的实数变量。
<i>Y</i>	将包含以毫米为单位的 <i>y</i> 坐标的实数变量。
<i>U</i>	将包含以度为单位的角度的实数变量。

备注

使用 **VGetRobotToolXYU** 轻松定义向上相机所看到部件的工具。这可以拾取部件, 在向上相机 FOV 中对其进行搜索, 定义部件的工具, 然后放置部件。

另见

VGetCameraXYU, VGetPixelXYU, VGetRobotXYU

VGetRobotToolXYU 示例**VB 例:**

```
Dim found As Integer
Dim x As Single, y As Single, u As Single
Dim seq As String, blob As String

seq = "testSeq"
blob = "blob01"
' 将工件向上移动到相机上方
m_spel.Jump("camPos")
m_spel.VRun(seq)
m_spel.VGetRobotToolXYU(seq, blob, 1, found, x, y, u)
m_spel.TLSet(1, x, y, u)
```

C# 例:

```
bool fnd;
float x, y, u;
string seq, blob;

seq = "testSeq";
blob = "blob01";
// 将工件向上移动到相机上方
m_spel.Jump("camPos");
m_spel.VRun(seq);
m_spel.VGetRobotToolXYU(seq, blob, 1, out fnd, out x, out y,
out u);
m_spel.TLSet(1, x, y, u);
```


VGetSearchWin 方法, Spel 类

描述

检索搜索窗口坐标。

语法

Sub **VGetSearchWin** (*Sequence* As String, *Object* As String, ByRef *Left* As Integer, ByRef *Top* As Integer, ByRef *Width* As Integer, ByRef *Height* As Integer)

参数

<i>Sequence</i>	含有当前项目中视觉序列名称的字符串表达式。
<i>Object</i>	含有序列 <i>Sequence</i> 中对象名称的字符串表达式。
<i>Left</i>	将包含以像素为单位的左坐标的整数变量。
<i>Top</i>	将包含以像素为单位的顶坐标的整数变量。
<i>Width</i>	将包含以像素为单位的宽度的整数变量。
<i>Height</i>	将包含以像素为单位的高度的整数变量。

另见

VGetModelWin, VSetModelWin, VSetSearchWin

VGetSearchWin 示例**VB 例:**

```
Dim left As Integer, top As Integer
Dim width As Integer, height As Integer

With m_spel
    .VGetSearchWin("testSeq", "corr01", left, top, _
        width, height)
    .VSetSearchWin("testSeq", "corr01", newLeft, top, _
        width, height)
    .VRun("testSeq")
End With
```

C# 例:

```
int left, top, width, height;

m_spel.VGetSearchWin("testSeq", "corr01", out left, out top,
    out width, out height);
m_spel.VSetSearchWin("testSeq", "corr01", newLeft, top,
    width, height);
m_spel.VRun("testSeq");
```

VGoCenter 方法, Spel 类

描述

使用可通过视觉系统检测的特征点，将机器人移动至特征点位于相机图像中心的位置。

语法

Sub **VGoCenter**(Sequence As String, LocalNumber As Integer, TargetTolerance As Double)

Sub **VGoCenter**(Sequence As String, LocalNumber As Integer, TargetTolerance As Double, Parent As Form)

Sub **VGoCenter**(Sequence As String, LocalNumber As Integer, TargetTolerance As Double, RobotSpeed As Integer, RobotAccel As Integer)

Sub **VGoCenter**(Sequence As String, LocalNumber As Integer, TargetTolerance As Double, RobotSpeed As Integer, RobotAccel As Integer, Parent As Form)

参数

<i>Sequence</i>	表示当前项目中视觉序列名称的字符串表达式。
<i>LocalNumber</i>	表示要移动机器人的本地坐标编号的整数。 若指定 -1，机器人在工具旋转的 XY 平面移动。
<i>TargetTolerance</i>	表示将视觉检测结果视为与目标位置一致的像素距离的实数值。 值的范围：0 - 3 pixels
<i>Form</i>	窗口的 .NET 父窗体。(可选)
<i>RobotSpeed</i>	可选。将包含机器人速度(%)的整数变量。 值的范围：0 - 100 如果省略，则设为“5”。
<i>RobotAccel</i>	可选。将包含机器人加速度(%)的整数变量。 值的范围：0 - 99 如果省略，则设为“5”。

另见

VDefArm, VDefGetMotionRange, VDefLocal, VDefSetMotionRange, VDefTool

VGoCenter 示例**VB 例:**

```
m_spel.VGoCenter("myseq", 1, 1.0)
```

C# 例:

```
m_spel.VGoCenter("myseq", 1, 1.0);
```

VLoad 方法, Spel 类**描述**

加载当前项目的视觉属性。

语法

```
Sub VLoad ()
```

备注

如果想要在程序启动时将视觉属性设置、模型和字体恢复为初始设置，则可使用 VLoad 方法。

另见

VSave

VLoad 示例**VB 例:**

```
m_spel.VLoad()
```

C# 例:

```
m_spel.VLoad();
```

VLoadModel 方法, Spel 类

描述

加载磁盘文件的视觉模型。

语法

Sub **VLoadModel** (*Sequence As String, Object As String, Path As String*)

参数

<i>Sequence</i>	含有当前项目中序列名称的字符串。
<i>Object</i>	含有对象名称的字符串。对象必须为 Correlation 、 Geometric 或 Polar 。
<i>Path</i>	加载模型的文件的完整路径名称，不包括扩展名。

备注

如果文件中的模型数据为错误类型，则会发生错误。例如：如果尝试将 **polar** 模型载入 **correlation**，则会发生错误。

如果提供文件扩展名，则会将其忽略。共有两个与 **fileName** 相关的文件。

对于 **correlation** 和 **geometric** 模型，**ModelOrgX** 和 **ModelOrgY** 值会与模型窗口宽度和高度一起恢复。

对于 **polar** 模型，将恢复 **Radius**、**Thickness** 和 **AngleOffset**。

另见

VSaveModel

VLoadModel 示例**VB 例:**

```
m_spel.VLoadModel("seq01", "corr01", "d:\models\part1")
```

C# 例:

```
m_spel.VLoadModel("seq01", "corr01", @"d:\models\part1");
```

VRun 方法, Spel 类

描述

运行当前项目中的视觉序列。

语法

Sub **VRun** (*Sequence* As String)

参数

Sequence 含有当前项目中序列名称的字符串。

备注

VRun 会与采用任意类型相机校准或无校准的序列协同工作。

若要显示图形，需要使用 SPELVideo 控制并将 Spel 类实例的 SpelVideoControl 属性设为 SPELVideo 控制。

执行 VRun 后，使用 VGet 检索结果。

另见

VGet, VSet

VRun 示例**VB 例:**

```
Function FindPart(x As Single, y As Single, angle As Single)
As Boolean
    Dim found As Boolean
    Dim x, y, angle As Single
    With m_spel
        .VRun("seq01")
        .VGet("seq01", "corr01", "found", found)
        If found Then
            .VGet("seq01", "corr01", "cameraX", x)
            .VGet("seq01", "corr01", "cameraY", y)
            .VGet("seq01", "corr01", "angle", angle)
            FindPart = True
        End If
    End With
End Function
```

C# 例:

```
bool FindPart(float x, float y, float angle)
{
    bool found;
    m_spel.VRun("seq01");
    m_spel.VGet("seq01", "corr01", "found", found);
    if (found) {
        m_spel.VGet("seq01", "corr01", "cameraX", out x);
        m_spel.VGet("seq01", "corr01", "cameraY", out y);
        m_spel.VGet("seq01", "corr01", "angle", out angle);
    }
    return found;
}
```

VSave 方法, Spel 类

描述

保存当前项目中的所有视觉数据。

语法

```
Sub VSave ()
```

备注

使用 **VSave** 可永久保存视觉属性的所有更改。

另见

VSet

VSave 示例**VB 例:**

```
With m_spel
    .VSet("seq01", "blob01", "SearchWinLeft", 100)
    .VSet("seq01", "corr01", "Accept", userAccept)
    .VSave()
End With
```

C# 例:

```
m_spel.VSet("seq01", "blob01", "SearchWinLeft", 100);
m_spel.VSet("seq01", "corr01", "Accept", userAccept);
m_spel.VSave();
```

VSaveImage 方法, Spel 类

描述

将视觉视频窗口保存到 PC 磁盘文件。

语法

```
Sub VSaveImage (Sequence As String, Path As String)
Sub VSaveImage (Sequence As String, Path As String, WithGraphics As Boolean)
```

参数

Sequence 含有当前项目中序列名称的字符串。

Path 保存图像的文件的完整路径名称, 包括扩展名。

WithGraphics 设置是否将序列结果图形保存至图像文件的 Boolean 表达式。

备注

使用 VSaveImage 可将视频显示屏中的图像保存至磁盘。文件扩展名必须为 MIM(Vision Guide 的默认格式)、BMP、TIF 或 JPG。

另见

LoadImage(SPELVideo 控制)

VSaveImage 示例**VB 例:**

```
Dim found As Boolean
m_spel.VRun("Seq")
m_spel.VGet("Seq", SpelVisionProps.AllFound, found)
If Not found Then
    m_spel.VSaveImage("Seq", "d:\reject.mim")
End If
```

C# 例:

```
bool found;
m_spel.VRun("Seq");
m_spel.VGet("Seq", SpelVisionProps.AllFound, out found);

if (!found)
    m_spel.VSaveImage("Seq", @"d:\reject.mim");
```

VSaveModel 方法, Spel 类

描述

将视觉对象搜索模型保存到 PC 磁盘文件。

语法

Sub **VSaveModel** (*Sequence As String, Object As String, Path As String*)

参数

<i>Sequence</i>	含有当前项目中序列名称的字符串。
<i>Object</i>	含有对象名称的字符串。对象必须为 Correlation、Geometric 或 Polar。
<i>Path</i>	保存模型的文件的完整路径名称，不包括扩展名。

备注

执行 **VSaveModel** 时，EPSON RC+ 7.0 会创建两个文件(*Path* + 扩展名):

Path.VOB, *Path.MDL*

对于 correlation 和 geometric 模型，ModelOrgX 和 ModelOrgY 值会与模型窗口同时保存。

对于 Polar 模型，将保存 Radius、Thickness 和 AngleOffset 。

另见

VLoadModel

VSaveModel 示例**VB 例:**

```
m_spel.VSaveModel("seq01", "corr01", "d:\models\part1")
```

C# 例:

```
m_spel.VSaveModel("seq01", "corr01", @"d:\models\part1");
```


VSet 方法, Spel 类

描述

设置视觉序列或对象属性的值。

语法

```
Sub VSet ( Sequence As String, PropCode As SpelVisionProps, Value As Integer )
Sub VSet ( Sequence As String, PropCode As SpelVisionProps, Value As Boolean )
Sub VSet ( Sequence As String, PropCode As SpelVisionProps, Value As Double )
Sub VSet ( Sequence As String, PropCode As SpelVisionProps, Value As Single )
Sub VSet ( Sequence As String, PropCode As SpelVisionProps, Value As String )
Sub VSet ( Sequence As String, Object As String, PropCode As SpelVisionProps,
          Value As Integer )
Sub VSet ( Sequence As String, Object As String, PropCode As SpelVisionProps,
          Value As Boolean )
Sub VSet ( Sequence As String, Object As String, PropCode As SpelVisionProps, Value
          As Color )
Sub VSet ( Sequence As String, Object As String, PropCode As SpelVisionProps,
          Value As Double )
Sub VSet ( Sequence As String, Object As String, PropCode As SpelVisionProps, Value
          As Single )
Sub VSet ( Sequence As String, Object As String, PropCode As SpelVisionProps,
          Value As String )
```

参数

<i>Sequence</i>	含有当前项目中视觉序列名称的字符串表达式。
<i>Object</i>	含有序列 <i>Sequence</i> 中对象名称的字符串表达式。如果属性用于序列，则此字符串必须为空。
<i>PropCode</i>	指定属性代码的 SpelVisionProps 值。
<i>Value</i>	含有新值的表达式。表达式类型必须与属性类型匹配。

另见

VGet, VRun

VSet 示例**VB 例:**

```
m_spel.VSet("seq01", "corr01", SpelVisionProps.Accept, 250)
```

C# 例:

```
m_spel.VSet("seq01", "corr01", SpelVisionProps.Accept, 250);
```

VSetModelWin 方法, Spel 类

描述

设置模型窗口坐标。

语法

```
Sub VSetModelWin ( Sequence As String, Object As String, Left As Integer, Top As Integer,
                  Width As Integer, Height As Integer )
```

参数

<i>Sequence</i>	含有当前项目中视觉序列名称的字符串表达式。
<i>Object</i>	含有序列 <i>Sequence</i> 中对象名称的字符串表达式。
<i>Left</i>	以像素为单位表示左坐标的整数表达式。
<i>Top</i>	以像素为单位表示顶坐标的整数表达式。
<i>Width</i>	以像素为单位表示宽度的整数表达式。
<i>Height</i>	以像素为单位表示高度的整数表达式。

另见

VGetModelWin, VGetSearchWin, VSetSearchWin

VSetModelWin 示例**VB 例:**

```
Dim left As Integer, top As Integer
Dim width As Integer, height As Integer

With m_spel
    .VGetSearchWin("testSeq", "corr01", left, top, _
                  width, height)
    .VSetSearchWin("testSeq", "corr01", left + 50, _
                  top - 10, width, height)
    .VRun("testSeq")
End With
```

C# 例:

```
int left, top, width, height;

m_spel.VGetSearchWin("testSeq", "corr01", out left, out top,
                    out width, out height);
m_spel. .VSetSearchWin("testSeq", "corr01", left + 50,
                    top - 10, width, height);
m_spel.VRun("testSeq");
```

VSetRobotPlaceTargetPos 方法, Spel 类

描述

设置部件放置位置。

语法

```
Sub VSetRobotPlaceTargetPos (Sequence As String, Object As String, Point As SpelPoint)
```

参数

Sequence 含有当前项目中视觉序列名称的字符串表达式。

Object 含有序列 *Sequence* 中对象名称的字符串表达式。

Point 含有放置位置的 SpelPoint 变量。

另见

VGetRobotPlacePos, VGetRobotPlaceTargetPos

VSetRobotPlaceTargetPos 示例**VB 例:**

```
Dim seq As String, blob As String
Dim targetPoint As SpelPoint

seq = "testSeq"
blob = "blob01"
m_spel.VGetRobotPlaceTargetPos(seq, blob, targetPoint)

' 调整放置位置
targetPoint.X = targetPoint.X + 10
m_spel.VSetRobotPlaceTargetPos(seq, blob, targetPoint)
```

C# 例:

```
string seq, blob;
SpelPoint targetPoint = new SpelPoint();

seq = "testSeq";
blob = "blob01";
m_spel.VGetRobotPlaceTargetPos(seq, blob, out targetPoint);

// 调整放置位置
targetPoint.X = targetPoint.X + 10;
m_spel.VSetRobotPlaceTargetPos(seq, blob, targetPoint);
```

VSetSearchWin 方法, Spel 类

描述

设置搜索窗口坐标。

语法

```
Sub VSetSearchWin ( Sequence As String, Object As String, Left As Integer, Top As Integer,
                  Width As Integer, Height As Integer )
```

参数

<i>Sequence</i>	含有当前项目中视觉序列名称的字符串表达式。
<i>Object</i>	含有序列 <i>Sequence</i> 中对象名称的字符串表达式。
<i>Left</i>	以像素为单位表示左坐标的整数表达式。
<i>Top</i>	以像素为单位表示顶坐标的整数表达式。
<i>Width</i>	以像素为单位表示宽度的整数表达式。
<i>Height</i>	以像素为单位表示高度的整数表达式。

另见

VGetModelWin, VSetModel, VGetSearchWin

VSetSearchWin 示例**VB 例:**

```
Dim left As Integer, top As Integer
Dim width As Integer, height As Integer

With m_spel
    .VGetSearchWin("testSeq", "corr01", left, top, _
                  width, height)
    .VSetSearchWin("testSeq", "corr01", newLeft, top, _
                  width, height)
    .VRun("testSeq")
End With
```

C# 例:

```
int left, top, width, height;

m_spel.VGetSearchWin("testSeq", "corr01", out left, out top,
                    out width, out height);
m_spel.VSetSearchWin("testSeq", "corr01", left + 50,
                    top, width, height);
m_spel.VRun("testSeq");
```

VShowModel 方法, Spel 类

描述

显示对象模型。有关更多详细信息，请参阅 Vision Guide 属性参考中的 ShowModel 属性。

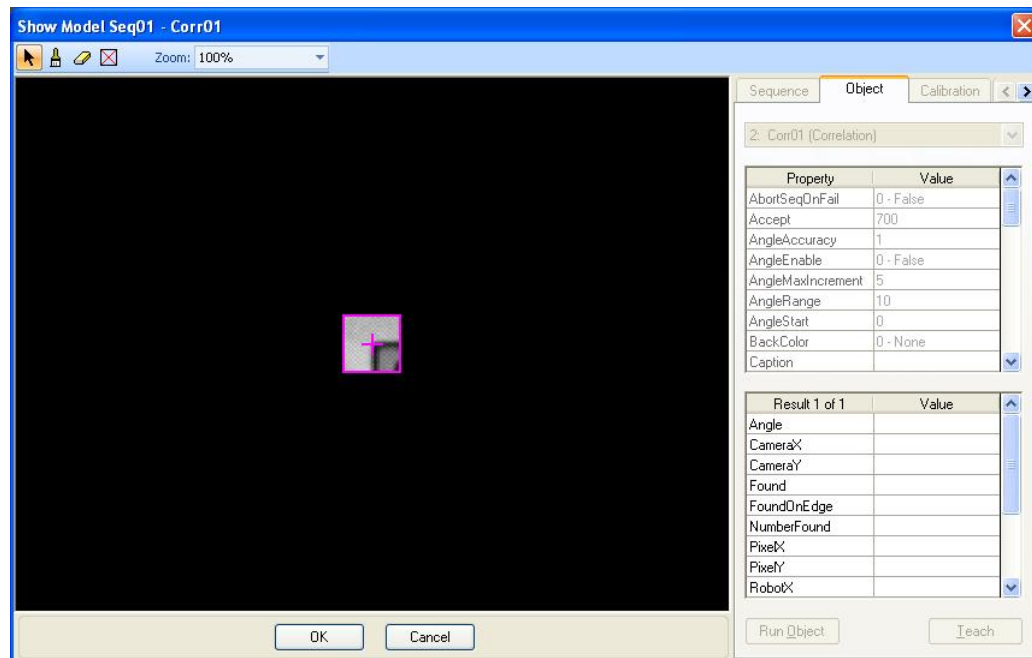
语法

Sub **VShowModel** (*Sequence As String*, *Object As String*)

参数

Sequence 含有当前项目中视觉序列名称的字符串表达式。

Object 含有当前项目中视觉对象名称的字符串表达式。

**另见**

VShowSequence, VTrain

VShowModel 示例**VB 例:**

```
m_spel.VShowModel("myseq", "myobj")
```

C# 例:

```
m_spel.VShowModel("myseq", "myobj");
```

VShowSequence 方法, Spel 类

描述

显示序列中的所有对象。

语法

Sub **VShowSequence** (*Sequence As String*)

参数

Sequence 含有待创建视觉序列名称的字符串表达式。

备注

使用 **VShowSequence** 可显示序列中的对象, 无需运行序列。有效对象颜色(洋红)将用于所有对象, 因此能够很容易看到这些对象。其适用于机器人相机移至使用多个序列扫描的工件特定部位上方的情况。机器人定位后, 可调用 **VShowSequence** 显示序列。

另见

VShowModel

VShowSequence 示例**VB 例:**

```
m_spel.VShowSequence("myseq")
```

C# 例:

```
m_spel.VShowSequence("myseq");
```

VStatsReset 方法, Spel 类

描述

重置当前项目中指定序列的视觉统计。

语法

Sub **VStatsReset** (*Sequence* As String)

参数

Sequence 含有当前项目中视觉序列名称的字符串表达式。

备注

VStatsReset 仅针对当前 EPSON RC+ 7.0 会话重置内存中指定序列的统计。如果想要永久保存更改, 应执行 **VStatsSave**。否则, 如果重启 EPSON RC+ 7.0, 则会恢复磁盘中的统计。

另见

VStatsResetAll, VStatsShow, VStatsSave

VStatsReset 示例**VB 例:**

```
Sub btnResetStats_Click()  
    m_spel.VStatsReset("seq01")  
End Sub
```

C# 例:

```
void btnResetStats_Click(object sender, EventArgs e)  
{  
    m_spel.VStatsReset("seq01");  
}
```

VStatsResetAll 方法, Spel 类

描述

重置所有序列的视觉统计。

语法

Sub **VStatsResetAll**

备注

VStatsResetAll 仅针对当前 EPSON RC+ 7.0 会话重置内存中的统计。如果想要永久保存更改, 应执行 **VStatsSave**。否则, 如果重启 EPSON RC+ 7.0, 则会恢复磁盘中的统计。

另见

VStatsReset, VStatsShow, VStatsSave

VStatsResetAll 示例**VB 例:**

```
Sub btnResetStats_Click()  
    m_spel.VStatsResetAll()  
End Sub
```

C# 例:

```
void btnResetStats_Click(object sender, EventArgs e)  
{  
    m_spel.VStatsResetAll();  
}
```


VStatsSave 方法, Spel 类**描述**

保存当前项目中所有序列的视觉统计。

语法

```
Sub VStatsSave ()
```

备注

如果您想要保存视觉统计的更改，则必须在关闭 EPSON RC+ 7.0 之前执行 **VStatsSave**。

另见

VStatsReset, VStatsResetAll, VStatsShow

VStatsSave 示例**VB 例:**

```
Sub btnResetStats_Click()  
    m_spel.VStatsSave()  
End Sub
```

C# 例:

```
void btnResetStats_Click(object sender, EventArgs e)  
{  
    m_spel.VStatsSave();  
}
```

VStatsShow 方法, Spel 类

描述

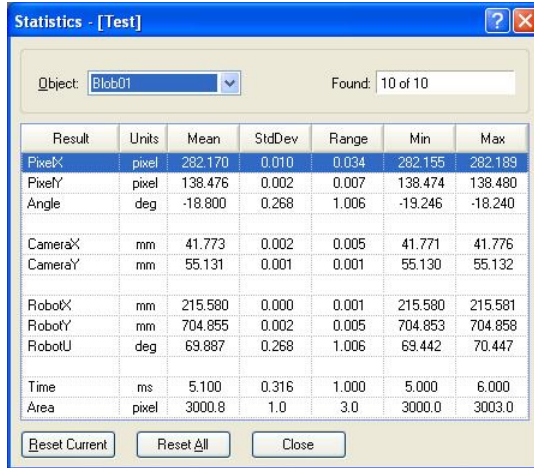
显示当前项目中指定序列的视觉统计对话框。

语法

Sub **VStatsShow** (*Sequence As String*)

参数

Sequence 含有当前项目中视觉序列名称的字符串表达式。



Result	Units	Mean	StdDev	Range	Min	Max
PixelX	pixel	282.170	0.010	0.034	282.155	282.189
PixelY	pixel	138.476	0.002	0.007	138.474	138.480
Angle	deg	-18.800	0.268	1.006	-19.246	-18.240
CameraX	mm	41.773	0.002	0.005	41.771	41.776
CameraY	mm	55.131	0.001	0.001	55.130	55.132
RobotX	mm	215.580	0.000	0.001	215.580	215.581
RobotY	mm	704.855	0.002	0.005	704.853	704.858
RobotU	deg	69.887	0.268	1.006	69.442	70.447
Time	ms	5.100	0.316	1.000	5.000	6.000
Area	pixel	3000.8	1.0	3.0	3000.0	3003.0

另见

VStatsReset, VStatsResetAll, VStatsSave

VStatsShow 示例**VB 例:**

```
Sub btnShowStats_Click()
    m_spel.VStatsShow("seq01")
End Sub
```

C# 例:

```
void btnShowStats_Click(object sender, EventArgs e)
{
    m_spel.VStatsShow("seq01");
}
```

VTeach 方法, Spel 类

描述

示教 correlation、geometric 或 polar 模型。

语法

Sub **VTeach** (*Sequence* As String, *Object* As String, *ByRef Status* as Integer)
 Sub **VTeach** (*Sequence* As String, *Object* As String, *AddSample* as Boolean,
KeepDontCares As Boolean, *ByRef Status* as Integer)

参数

Sequence 当前项目中视觉序列的名称。
Object *Sequence* 中的对象名称。可示教 Correlation、Geometric 或 Polar 对象。
AddSample 添加样品时为 True，添加新模型时则为 False。
KeepDontCares 使用旧检测掩码时为 True，废弃时则为 False。
Status 返回状态。如果成功，返回 1，否则返回 0。

备注

调用 **VTeach** 之前，必须确保模型窗口位于正确位置。
 对于 polar 对象，搜索窗口和 thickness 必须设置正确。使用 VSet 设置搜索窗口位置和 thickness。
 对于 correlation 和 geometric 对象，搜索窗口和模型窗口必须设置正确。使用 SearchWin 和 ModelWin 的 VSet 设置搜索和模型窗口位置。或者可使用 VTrain 命令，以使操作员能够交互更改窗口。
 示教模型后，可使用 VSaveModel 方法将其保存至 PC 磁盘文件内。

另见

VTrain, VSaveModel

VTeach 示例**VB 例:**

```
Dim status As Integer

' 变换窗口位置
m_spel.VTrain("seq01", "corr01", status)

' 示教模型
m_spel.VTeach("seq01", "corr01", status)
```

C# 例:

```
int status;

// 变换窗口位置
m_spel.VTrain("seq01", "corr01", status);

// 示教模型
m_spel.VTeach("seq01", "corr01", out status);
```

VTrain 方法, Spel 类

描述

该命令允许您培训整个序列中的对象或单个对象。

语法

Function VTrain (Sequence As String [, Object As String] [, Flags as Integer] [, Parent as Form]) As Boolean

参数

<i>Sequence</i>	当前项目中视觉序列的名称。
<i>Object</i>	<i>Sequence</i> 中的对象名称。您可培训任一类型的对象。如果 <i>Object</i> 为空字符串, 则可培训整个序列。
<i>Flags</i>	可选。配置 VTrain 对话框 1 - 显示示教按钮 2 - 不显示模型窗口。
<i>Parent</i>	可选。.NET 窗体将作为父窗口。

返回值

如果操作员点击 OK 按钮, 则 VTrain 返回 True, 否则返回 False。

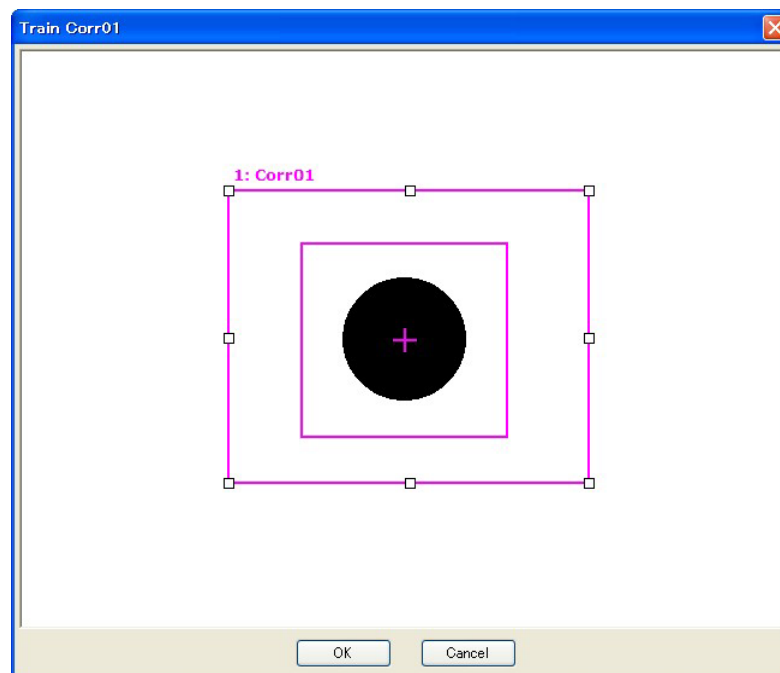
备注

执行 **VTrain** 时, 会打开一个对话框, 显示实时视频图像和显示的指定对象。操作员可缩放/移动搜索窗口, 并培训模型窗口(针对 **correlation** 和 **geometric** 对象)。当操作员完成时, 可点击 OK 保存更改, 或点击 Cancel 忽略更改。如果点击 OK, 则新信息会自动保存至当前项目。

如果设置了 *flags* 位 1, 则会显示示教按钮。对于 **Correlation**、**Geometric** 和 **Polar** 对象, 如果点击示教按钮, 则会示教模型。在运行 VTrain 检查是否培训模型之后, 可检索 ModelOK 属性。对于 **Blob** 对象, 按钮将打开直方图对话框, 且操作员可调节高低阈值, 然后查看更改效果。

如果设置了 *flags* 位 2, 则不会显示模型窗口。操作员仅可更改搜索窗口。

对于 correlation 和 geometric 对象，如果不需要显示示教按钮，则可在调用 **VTrain** 后调用 **VTech** 以示教模型。



另见

VTech, VSaveModel

VTrain 示例

VB 例:

```
Dim status As Integer

' 变换窗口位置
trainOK =m_spel.VTrain("seq01", "corr01")

' 示教模型
If trainOK Then
m_spel.VTech("seq01", "corr01", status)
EndIf
```

C# 例:

```
int status;
bool trainOK;

// 变换窗口位置
trainOK = m_spel.VTrain("seq01", "corr01");

// 示教模型
if (trainOK)
    m_spel.VTech("seq01", "corr01", out status);
```

WaitCommandComplete 方法, Spel 类

描述

该命令会等待以 AsyncMode = True 开始的命令完成。

语法

```
Sub WaitCommandComplete ()
```

另见

AsyncMode

WaitCommandComplete 示例**VB 例:**

```
With m_spel
    .AsyncMode = True
    .Jump("pick")
    .Delay(500)
    .On(1)
    .WaitCommandComplete()
End With
```

C# 例:

```
m_spel.AsyncMode = true;
m_spel.Jump("pick");
m_spel.Delay(500);
m_spel.On(1);
m_spel.WaitCommandComplete();
```

WaitMem 方法, Spel 类

描述

等待内存位状态发生改变。

语法

```
Sub WaitMem (BitNumber As Integer, Condition As Boolean, Timeout As Single)
Sub WaitMem (Label As String, Condition As Boolean, Timeout As Single)
```

参数

<i>BitNumber</i>	表示内存位编号的整数表达式。
<i>Label</i>	表示内存位标签的字符串。
<i>Condition</i>	表示内存位状态的 Boolean 表达式。
<i>Timeout</i>	表示最大等待时间(单位为秒)的单一表达式。

备注

应始终检查是否因使用 TW 方法而发生超时。请参见以下示例。

另见

WaitSw

WaitMem 示例**VB 例:**

```
' 等待内存位 1 变为 1(True)
' 最长时间为 5 秒
m_spel.WaitMem(1, True, 5)
' WaitMem 是否超时?
If m_spel.TW() Then
    MsgBox "memory bit time out occurred"
End If
```

C# 例:

```
// 等待内存位 1 变为 1(True)
// 最长时间为 5 秒
m_spel.WaitMem(1, True, 5);
// WaitMem 是否超时?
if (m_spel.TW())
    MessageBox.Show("memory bit time out occurred");
```

WaitSw 方法, Spel 类

描述

等待输入位状态发生改变。

语法

Sub **WaitSw** (*BitNumber* As Integer, *Condition* As Boolean, *Timeout* As Single)
Sub **WaitSw** (*Label* As String, *Condition* As Boolean, *Timeout* As Single)

参数

<i>BitNumber</i>	表示输入位编号的整数表达式。
<i>Label</i>	表示输入位标签的字符串。
<i>Condition</i>	表示输入位状态的 Boolean 表达式。
<i>Timeout</i>	表示最大等待时间(单位为秒)的单一表达式。

备注

应始终检查是否因使用 **TW** 方法而发生超时。请参见以下示例。

另见

WaitMem

WaitSw 示例**VB 例:**

```
Const PartPresent = 1
m_spel.WaitSw(PartPresent, True, 5)
If m_spel.TW() Then
    MsgBox "Part present time out occurred"
End If
```

C# 例:

```
const int PartPresent = 1;
m_spel.WaitSw(PartPresent, True, 5);
if (m_spel.TW())
    MessageBox.Show("Part Present time out occurred");
```


WaitTaskDone 方法, Spel 类

描述

等待任务完成并返回状态。

语法

Function **WaitTaskDone** (*TaskNumber* As Integer) As SpelTaskState
Function **WaitTaskDone** (*TaskName* As String) As SpelTaskState

参数

TaskNumber 待返回执行状态的任务编号。

TaskName 含有任务名称的字符串表达式。

返回值

SpelTaskState 值。

另见

SpelTaskState, TasksExecuting, TaskState, Xqt

WaitTaskDone 示例**VB 例:**

```
Dim taskState As SpelTaskState
m_spel.Xqt 2, "mytask"
'
' 其他处理
'
taskState = m_spel.WaitTaskDone(2)
```

C# 例:

```
SpelTaskState taskState;
m_spel.Xqt(2, "mytask");
//
// 其他处理
//
taskState = m_spel.WaitTaskDone(2);
```

Weight 方法, Spel 类

描述

指定当前机器人的 `weight` 参数。

语法

Sub **Weight** (*PayloadWeight* As Single, *ArmLength* As Single)

Sub **Weight** (*PayloadWeight* As Single, *Axis* As SpelAxis, [*Axis*])

参数

PayloadWeight 待承载的夹具末端重量，单位为 Kg。

ArmLength 第二机械臂的旋转中心至夹具末端的重心之间的距离，单位为 mm。

Axis 指定分配有效负载重量的附加轴(S 或 T)。

NOTE

请勿在 `PayLoadWeight` 和 `ArmLength` 参数中输入整数值。使用 `Single` 变量或直接输入 `Single` 类型值。

另见

`Inertia`, `JRange`, `Tool`

Weight 示例**VB 例:**

```
m_spel.Weight(2.0F, 2.5F)
```

C# 例:

```
m_spel.Weight(2.0F, 2.5F);
```

Xqt 方法, Spel 类

描述

启动一个 SPEL+ 任务。

语法

```
Sub Xqt (FuncName As String [, TaskType As SpelTaskType])
Sub Xqt (TaskNumber As Integer, FuncName As String [, TaskType As SpelTaskType])
```

参数

TaskNumber 待执行任务的任务号。任务号范围为 1 至 32。

FuncName 待执行函数的名称。还可为函数选择性提供参数。参数必须位于括号内，以逗号分隔。有关详细信息，请参阅 SPEL+ Xqt 语句。还可参阅示例。

TaskType 可选。指定任务类型为 Normal、NoPause 或 NoEmgAbort。

备注

执行 **Xqt** 时，控制将立即返回至调用程序。使用 Call 方法等待任务完成，或者可使用 EventReceived 和任务状态事件等待任务完成。

另见

Call, EnableEvent, EventReceived

Xqt 示例**VB 例:**

```
m_spel.Xqt(2, "conveyor")

' 为 RunPart 函数提供一个参数
m_spel.Xqt(3, "RunPart(3)")

Dim funcToExec As String
funcToExec = "RunPart(" & partNum & ")"
m_spel.Xqt(3, funcCall)
```

C# 例:

```
m_spel.Xqt(2, "conveyor");

// 为 RunPart 函数提供一个参数
m_spel.Xqt(3, "RunPart(3)");

string funcToExec;
funcToExec = string.Format("RunPart({0})", partNum);
m_spel.Xqt(3, funcToExec);
```

XYLim 方法, Spel 类

描述

设置机器人的允许动作范围限制。

语法

Sub **XYLim** (*XLowerLimit* As Single, *XUpperLimit* As Single, *YLowerLimit* As Single, *YUpperLimit* As Single [, *ZLowerLimit* As Single] [, *ZUpperLimit* As Single])

参数

<i>XLowerLimit</i>	机器人可行进的最小 X 坐标位置。(机器人不得移至 X 坐标小于 minX 的位置。)
<i>XUpperLimit</i>	机器人可行进的最大 X 坐标位置。(机器人不得移至 X 坐标大于 maxX 的位置。)
<i>YLowerLimit</i>	机器人可行进的最小 Y 坐标位置。(机器人不得移至 Y 坐标小于 minY 的位置。)
<i>YUpperLimit</i>	机器人可行进的最大 Y 坐标位置。(机器人不得移至 Y 坐标大于 maxY 的位置。)
<i>ZLowerLimit</i>	可选。机器人可行进的最小 Z 坐标位置。(机器人不得移至 Z 坐标小于 minZ 的位置。)
<i>ZUpperLimit</i>	可选。机器人可行进的最大 Z 坐标位置。(机器人不得移至 Z 坐标大于 maxZ 的位置。)

备注

XYLim 用于定义动作范围限制。很多机器人系统允许用户定义关节限制，而 SPEL+ 语言允许定义关节限制和动作范围限制。实际上，这便允许用户创建其应用的工作范围。(请牢记，也可通过 SPEL 定义关节范围限制。)

通过 XYLim 值确定的动作范围仅适用于动作命令目标位置，但不适用于从起始位置到目标位置的动作路径。因此，机械臂可能会在动作期间移至 XYLim 范围以外。(即，XYLim 范围不影响脉冲。)

若要关闭动作范围限制，在范围限制参数指定 0。

另见

JRange

XYLim 示例**VB 例:**

```
m_spel.XYLim(0, 0, 0, 0)
```

C# 例:

```
m_spel.XYLim(0, 0, 0, 0);
```

XYLimClr 方法, Spel 类**描述**

清除(取消定义)XYLim 定义。

语法

Sub **XYLimClr** ()

另见

XYLim, XYLimDef

XYLimClr 示例**VB 例:**

```
m_spel.XLLimClr()
```

C# 例:

```
m_spel.XYLimClr();
```

XYLimDef 方法, Spel 类

描述

返回是否已定义 XYLim。

语法

Function **XYLimDef** () As Boolean

返回值

如果 XYLim 已定义, 则返回 True, 否则返回 False。

另见

XYLim, XYLimClr

XYLimDef 示例

VB 例:

```
Dim xyLimDefined As Boolean  
xyLimDefined = m_spel.XYLimDef()
```

C# 例:

```
bool xyLimDefined;  
xyLimDefined = m_spel.XYLimDef();
```

14.4 Spel 类事件

EventReceived 事件, Spel 类

描述

当 EPSON RC+ 7.0 发送系统事件或当 SPEL+ 中运行的程序使用 SPELCom_Event 语句发送事件时发生。

语法

EventReceived (ByVal sender As Object, ByVal e As RCAPINet.SpelEventArgs)

参数

e.Event 表示特定用户定义事件的编号。

e.Message 含有事件消息的字符串。

备注

EPSON RC+ 7.0 会发出多个系统事件。如下表所述。

系统事件

一些事件默认为禁用。若要使用这些事件，应先通过 `EnableEvent` 方法启用它们。

事件编号	事件消息	常量	描述
1	"PAUSE"	SpelEvents.Pause	任务暂停时发生。默认为启用。
2	"SAFE GUARD OPEN"	SpelEvents.SafeGuardOpen	安全防护打开时发生。默认为启用。
3	"SAFE GUARD CLOSE"	SpelEvents.SafeGuardClose	安全防护关闭时发生。默认为启用。
4	Project build status text	SpelEvents.ProjectBuildStatus	每条构建状态消息会在 <code>BuildProject</code> 方法期间发送。根据需要添加 CRLF。这些消息与 EPSON RC+ 7.0 GUI 中 Project Build Status 窗口显示的消息相同。此事件必须使用 <code>EnableEvent</code> 方法启用。默认为禁用。
5	"Error xxx!: mmm in task at line yyy"	SpelEvents.Error	因未处理的错误或产生系统错误而中止任务时发生。默认为启用。
6	Text from print statement	SpelEvents.Print	从 SPEL+ 任务中执行 Print 语句时发生。默认为禁用。
7	"ESTOP ON"	SpelEvents.EStopOn	紧急停止条件改为 ON 时发生。默认为启用。
8	"ESTOP OFF"	SpelEvents.EStopOff	紧急停止条件改为 OFF 时发生。默认为启用。
9	"CONTINUE"	SpelEvents.Continue	执行完 <code>Cont</code> 后发生。默认为启用。
10	<Robot #>,"MOTOR ON"	SpelEvents.MotorOn	所示机器人的电机处于 ON 时发生。默认为禁用。
11	<Robot #>,"MOTOR OFF"	SpelEvents.MotorOff	所示机器人的电机处于 OFF 时发生。默认为禁用。

14. RCAPINet 参考

事件编号	事件消息	常量	描述
12	<Robot #>,"POWER HIGH"	SpelEvents.PowerHigh	所示机器人的电量处于 HIGH 时发生。 默认为禁用。
13	<Robot #>,"POWER LOW"	SpelEvents.PowerLow	所示机器人的电量处于 LOW 时发生。 默认为禁用。
14	"TEACH MODE"	SpelEvents.TeachMode	示教模式激活时发生。 默认为启用。
15	"AUTO MODE"	SpelEvents.AutoMode	自动模式激活时发生。 默认为启用。
16	"<TaskID>,<Status>,<FuncName>" Status: "RUN", "HALT", "PAUSE", "FINISHED", "ABORTED"	SpelEvents.TaskState	任务状态变更时发生。 默认为无效。
17	"SHUTDOWN"	SpelEvents.Shutdown	RC+ 正在关闭时发生。 默认为无效。
18	"ALL TASKS STOPPED"	SpelEvents.AllTasksStopped	所有任务已停止时发生。 默认为无效用。
19	"DISCONNECTED"	SpelEvents.Disconnected	已从 PC 断开控制器通信时发生。 启用时, RC+ 不显示表示连接断开的消息框。 默认为无效。
20	"MOTION STARTED"	SpelEvents.MotionStarted	控制命令已开始 默认为无效。
21	"MOTION COMPLETE"	SpelEvents.MotionComplete	控制命令已结束 默认为无效。

用户事件

您可以使用 **SPELCom_Event** 命令将事件从 SPEL+ 程序发送至 Visual Basic 应用。

```
Spelcom_Event 3000, cycNum, lotNum, cycTime
```

该语句执行时，将通过事件编号和消息调用 EventReceived 例程。有关 SPELCom_Event 的详细信息，请参阅 *EPSON RC+ 7.0 在线帮助* 或 *13. SPELCom_Event*。

Use 示例

VB 例:

```
Sub m_spel_EventReceived ( _
    ByVal sender As Object, _
    ByVal e As RCAPINet.SpelEventArgs) _
    Handles m_spel.EventReceived
    Redim tokens(0) As String
    Select Case e.Event
        Case 3000
            tokens = e.Message.Split(New [Char]() {" "c}, _

System.StringSplitOptions.RemoveEmptyEntries)
            lblCycCount.Text = tokens(0)
            lblLotNumber.Text = tokens(1)
            lblCycTime.Text = tokens(2)
    End Select
End Sub
```

C# 例:

```
public void m_spel_EventReceived(object sender,
SpeleEventArgs e)
{
    string[] tokens = new string[3];
    switch ((int) e.Event){
        case 3000:
            tokens = e.Message.Split(` `);
            lblCycCount.Text = tokens(0);
            lblLotNumber.Text = tokens(1);
            lblCycTime.Text = tokens(2);
            break;
        default:
            break;
    }
}
```

处理事件

从 Spel 类实例中调用 **EventReceived** 时，EPSON RC+ 7.0 进程服务器需要等待事件处理例程完成。因此，不要尝试从 **EventReceived** 例程中执行任何 RC+ API 命令或进行长时间运行处理。如果您想根据发生的事件执行命令，则在 **EventReceived** 中设置标记并通过事件处理函数以外的应用中的主循环处理此标记。

例如：在您的 Visual Basic 主窗体 Load 程序中，可创建接收来自 SPEL+ 事件的事件循环。在 spel_EventReceived 例程中，设置全局标记指明已接收的事件。然后，可通过 Load 程序中创建的事件循环执行实际的事件处理。

显示事件消息

在窗体中添加 TextBox 控制。

每次收到事件时，可在 TextBox 控制的 Text 属性中显示事件消息。

VB 例:

```
Private Sub m_spel_EventReceived(ByVal sender As Object, _
    ByVal e As SpelEventArgs) Handles
    m_spel.EventReceived
    txtEvents.AppendText(e.Event & ": " & e.Message &
    vbCrLf)
End Sub
```

C# 例:

```
private void m_spel_EventReceived(object sender,
    SpelEventArgs e)
{
    txtEvents.AppendText(e.Event + ": " + e.Message);
}
```

另见

EnableEvent(Spel 类)

14.5 SPELVideo 控制

描述

此控制允许显示视觉系统的视频。有关此控制使用方法的详细信息，请参阅第 11 章 *显示视频*。

文件名

RCAPINet.dll

14.6 SPELVideo 控制属性

此控制支持除 .NET 标准组件属性以外的下列属性，如 Left、Top、Width 和 Height。有关标准属性的文件，请参阅 Visual Basic 在线帮助。

- Camera
- GraphicsEnabled
- VideoEnabled

Camera 属性, SPELVideo 控制

描述

设置/获取显示视频的相机编号。这对于想在步进操作、实时视频监控等过程中显示视频是非常有用的。如果正在使用此控制显示视觉序列的图形，那么在序列运行时，将使用序列的相机编号代替此属性值。

语法

Property **Camera** As Integer

默认值

0 - 显示的任一相机

返回值

含有当前相机编号的整数值

另见

VideoEnabled, GraphicsEnabled

示例

VB 例:

```
SpelVideo1.Camera = 1
```

C# 例:

```
SpelVideo1.Camera = 1;
```

GraphicsEnabled 属性, SPELVideo 控制

描述

设置/返回是否在运行序列后显示视觉图形。为查看图形，必须使用 SPELVideo 控制属性将控制连接至 Spel 类实例。此属性可“联机”设置，这样便可在序列运行时打开/关闭图形。

语法

Property **GraphicsEnabled** As Boolean

默认值

False

返回值

如果显示视觉图形，则返回 True，否则返回 False。

另见

Camera, VideoEnabled

示例**VB 例:**

```
SpelVideo1.GraphicsEnabled = True
```

C# 例:

```
SpelVideo1.GraphicsEnabled = true;
```

VideoEnabled 属性, SPELVideo 控制

描述

确定是否显示视频。

语法

Property **VideoEnabled** As Boolean

默认值

False

返回值

如果显示视频, 则返回 True, 否则返回 False。

另见

Camera, GraphicsEnabled

示例

VB 例:

```
SpelVideo1.VideoEnabled = True
```

C# 例:

```
SpelVideo1.VideoEnabled = true;
```

14.7 SPELVideo 控制方法

LoadImage 方法, SPELVideo 控制

描述

加载文件中的图像以便显示。

语法

Sub **LoadImage** (*Path* As String)

参数

Path 加载图像的文件的完整路径名称, 包括扩展名。

备注

使用 LoadImage 加载之前保存的图像以便显示。文件扩展名必须为 BMP、TIF 或 JPG。

另见

VSaveImage(Spel 类)

LoadImage 示例

VB 例:

```
m_spelVideo.LoadImage ("c:\RejectImages\reject001.bmp")
```

C# 例:

```
m_spelVideo.LoadImage (@"c:\RejectImages\reject001.bmp");
```

14.8 SPELVideo 控制事件

此控制的所有事件均为标准 .NET 事件。有关详细信息，请参阅 Visual Basic 在线帮助。

14.9 SpelConnectionInfo 类

成员名称	类型	描述
ConnectionIPAddress	String	EPSON RC+ 中配置的地址。 * 当通过 USB 或虚拟控制器连接时，ConnectionIPAddress 显示空白。
ConnectionNumber	Integer	EPSON RC+ 中配置的连接编号。
ConnectionName	String	EPSON RC+ 中配置的连接名称。
ConnectionType	SpelConnectionType	EPSON RC+ 中配置的连接类型。

以下为示例。

VB 例:

```
Dim connectionInfo() As RCAPINet.SpelConnectionInfo
connectionInfo = m_spel.GetConnectionInfo()
```

C# 例:

```
SpelConnectionInfo[] info = m_spel.GetConnectionInfo();
```

14.10 SpelControllerInfo 类

成员名称	类型	描述
ProjectName	String	控制器中的项目名称。
ProjectID	String	控制器中项目的唯一项目 ID。
Options	List<SpelOptionInfo>	控制器中的选项清单

以下为示例。

VB 例:

```
Dim info As RCAPINet.SpelControllerInfo
info = m_spel.GetControllerInfo()
Label1.Text = info.ProjectID + " " + info.ProjectName
```

C# 例:

```
SpelControllerInfo info;
info = m_spel.GetControllerInfo();
Label1.text = info.ProjectID + " " + info.ProjectName;
```

14.11 SpelException 类

SpelException 类由 ApplicationException 类派生而来。它增加了 ErrorNumber 属性和一些构造函数。

下例所示即如何检索错误编号和错误消息。

VB 例:

```
Try
    m_spel.Go(1)
Catch (ex As RCAPINet.SpelException)
    MsgBox(ex.ErrorNumber & " " & ex.Message)
End Try
```

C# 例:

```
try{
    m_spel.Go(1);
}
catch(RCAPINet.SpelException ex){
    MessageBox.Show(string.Format("{0}: {1}", ex.ErrorNumber,
ex.Message));
}
```

SpelException 属性

ErrorNumber As Integer

SpelException 方法

Sub New ()

默认构造函数。

Sub New (Message As String)

指定错误消息的可选构造函数。

Sub New (ErrorNumber As Integer, Message As String)

指定错误编号和相关消息的可选构造函数。

Sub New (Message As String, Inner As Exception)

指定错误消息和内部异常的可选构造函数。

Sub New (ErrorNumber As Integer, Message As String, Inner As Exception)

指定错误编号、错误消息和内部异常的可选构造函数。

14.12 SpelOptionInfo 类

成员名称	类型	描述
Name	String	机器人名称
Status	SpelOptionStatus	机器人编号

以下为示例。

VB 例:

```
Dim info As SpelControllerInfo
info = m_spel.GetControllerInfo()
Label1.Text = info.Options(1).Name + " " +
info.Options(1).Status
```

C# 例:

```
SpelControllerInfo info;
info = m_spel.GetControllerInfo();
Label1.Text = info.Options[1].Name + " " +
info.Options[1].Status;
```

14.13 SpelPoint 类

SpelPoint 类可用于多个动作方法以及 Spel 类的 GetPoint 和 SetPoint 方法。

下面是一些 Visual Basic 的示例:

1:

```
Dim pt As New RCAPINet.SpelPoint(25.5, 100.3, -21, 0)
m_spel.Go(pt)
```

2:

```
Dim pt As New RCAPINet.SpelPoint
pt.X = 25.5
pt.Y = 100.3
pt.Z = -21
m_spel.Go(pt)
```

3:

```
Dim pt As New RCAPINet.SpelPoint
pt = m_spel.GetPoint("P*")
pt.Y = 222
m_spel.Go(pt)
```

下面是一些 Visual C# 的示例：

```
1:
SpelPoint pt = new SpelPoint(25.5, 100.3, -21, 0);
m_spel.Go(pt);
2:
SpelPoint pt = new SpelPoint();
pt.X = 25.5;
pt.Y = 100.3;
pt.Z = -21;
m_spel.Go(pt);
3:
SpelPoint pt = new SpelPoint();
pt = m_spel.GetPoint("P0");
pt.Y = 222;
m_spel.Go(pt);
```

14.13.1 SpelPoint 属性

VB 例:

X As Single
 Y As Single
 Z As Single
 U As Single
 V As Single
 W As Single
 R As Single
 S As Single
 T As Single
 Hand As SpelHand
 Elbow As SpelElbow
 Wrist As SpelWrist
 Local As Integer
 J1Flag As Integer
 J2Flag As Integer
 J4Flag As Integer
 J6Flag As Integer
 J1Angle As Single
 J4Angle As Single

C# 例:

float x
 float y
 float z
 float u
 float v
 float w
 float r
 float s
 float t
 SpelHand Hand
 SpelElbow Elbow
 SpelWrist Wrist
 int Local
 int J1Flag
 int J2Flag
 int J4Flag
 int J6Flag
 float J1Angle
 float J4Angle

14.13.2 SpelPoint 方法

Sub Clear ()

清除所有点数据。

Sub New ()

默认构造函数。创建空点(清除所有数据)。

Sub New (X As Single, Y As Single, Z As Single, U As Single [, V As Single] [, W As Single])

指定坐标的新点的可选构造函数。

Function ToString ([Format As String]) As String

允许指定 Format 的 ToString 覆盖。它可以返回 SPEL+ 中定义的点。

格式可为:

Empty 返回具有所有坐标和属性的整个点。

"XY" 返回 "XY(...)"

"XYST" 返回 "XY(...):ST(...)"

14.14 SpelRobotInfo 类

成员名称	类型	描述
RobotModel	String	机器人型号
RobotName	String	机器人名称
RobotNumber	Integer	机器人编号
RobotSerial	String	机器人序列号
RobotType	SpelRobotType	机器人类型

以下为示例。

VB 例:

```
Dim info As SpelRobotInfo
info = m_spel.GetRobotInfo()
Label1.Text = info.RobotNumber + " " + info.RobotModel
```

C# 例:

```
SpelRobotInfo info;
info = m_spel.GetRobotInfo();
Label1.Text = info.RobotNumber + " " + info.RobotModel;
```

14.15 SpelTaskInfo 类

成员名称	类型	描述
CPU	Integer	SPEL 任务的 CPU 负载率
ErrorCode	Integer	错误代码
StartTime	DateTime	任务开始日期
TaskName	String	任务名称
TaskNumber	Integer	任务编号
TaskState	SpelTaskState	任务状态

以下为示例。

VB 例:

```
Dim info As SpelTaskInfo
info.TaskState = m_spel.TaskState(1)
Label1.Text = info.TaskNumber + " " + info.TaskState
```

C# 例:

```
SpelTaskInfo info;
info.TaskState = m_spel.TaskState(1);
Label1.Text = info.TaskNumber + " " + info.TaskState;
```

14.16 枚举

14.16.1 SpelArmDefMode 枚举

成员名称	值	描述
Rough	1	使用一个姿势定义机械臂。
Fine	1	使用两个姿势定义机械臂。

14.16.2 SpelArmDefType 枚举

成员名称	值	描述
J2Camera	1	定义安装 J2 相机的机械臂。

14.16.3 SpelAxis 枚举

成员名称	值	描述
X	1	X 轴
Y	2	Y 轴
Z	3	Z 轴
U	4	U 轴
V	5	V 轴
W	6	W 轴
R	7	R 轴
S	8	S 轴
T	9	T 轴

14.16.4 SpelBaseAlignment 枚举

成员名称	值	描述
XAxis	0	与 X 轴对齐。
YAxis	1	与 Y 轴对齐。

14.16.5 SpelCalPlateType 枚举

成员名称	值	描述
None	0	无校准板。
Large	1	大校准板。
Medium	2	中校准板。
Small	3	小校准板。
XSmall	4	极小校准板。

14.16.6 SpelConnectionType 枚举

成员名称	值	描述
USB	1	USB 连接。
Ethernet	2	Ethernet 连接。
Virtual	3	连接虚拟控制器。

14.16.7 SpelDialogs 枚举

成员名称	值	描述
RobotManager	1	工具 机器人管理器对话框 ID
ControllerTools	2	工具 控制器对话框 ID
VisionGuide	3	工具 Vision Guide 对话框 ID
ForceGuide	4	Force Guide 对话框 ID
PartFeeding	5	Part Feeding 对话框 ID

14.16.8 SpelElbow 枚举

成员名称	值	描述
Above	1	肘的方向为上方。
Below	2	肘的方向为下方。

14.16.9 SpelEvents 枚举

成员名称	值	描述
Pause	1	Pause 事件 ID。
SafeguardOpen	2	安全防护打开事件 ID。
SafeguardClose	3	安全防护关闭事件 ID。
ProjectBuildStatus	4	项目构建状态事件 ID。
Error	5	错误事件 ID。
Print	6	打印事件 ID。
EstopOn	7	紧急停止 on 事件 ID。
EstopOff	8	紧急停止 off 事件 ID。
Continue	9	Continue 事件 ID。
MotorOn	10	电机 on 事件 ID。
MotorOff	11	电机 off 事件 ID。
PowerHigh	12	电量 high 事件 ID。
PowerLow	13	电量 low 事件 ID。
TeachMode	14	示教模式事件 ID。
AutoMode	15	自动模式事件 ID。
TaskState	16	任务状态事件 ID。
Shutdown	17	Shutdown 事件 ID。
AllTasksStopped	18	所有任务停止事件 ID。
Disconnected	19	断开事件 ID
MotionStarted	20	控制命令开始事件 ID
MotionComplete	21	控制命令结束事件 ID

14.16.10 SpelForceAxis 枚举

成员名称	值	描述
XForce	1	指定 X 力轴。
YForce	2	指定 Y 力轴。
ZForce	3	指定 Z 力轴。
XTorque	4	指定 X 扭矩轴。
YTorque	5	指定 Y 扭矩轴。
ZTorque	6	指定 Z 扭矩轴。

14.16.11 SpelForceCompareType 枚举

成员名称	值	描述
LessOrEqual	0	力小于或等于指定阈值时触发 Till。
GreaterOrEqual	1	力大于或等于指定阈值时触发 Till。

14.16.12 SpelForceProps枚举

成员名称	值	描述
EndStatus	28340	力觉引导序列或力觉引导对象的结束状态
ForceCondOK	28440	力觉引导对象的力的结束条件的到达状态
IOCondOK	28590	力觉引导对象的 I/O 结束条件的到达状态
PosCondOK	28860	力觉引导对象的位置结束条件的到达状态
Time	29070	力觉引导序列或力觉引导对象的执行时间
TimeOut	29080	力觉引导对象的超时的到达状态
LastExecObject	30100	最后执行的力觉引导对象的名称
MeasuredHeight	30500	高度检测序列测量出的高度
FailedStatus	30510	力觉引导序列的失败原因
AvgForcesFx	100211	力觉引导对象执行中, Fx 力的平均值
AvgForcesFy	100212	力觉引导对象执行中, Fy 力的平均值
AvgForcesFz	100213	力觉引导对象执行中, Fz 力的平均值
AvgForcesTx	100214	力觉引导对象执行中, Tx 扭矩的平均值
AvgForcesTy	100215	力觉引导对象执行中, Ty 扭矩的平均值
AvgForcesTz	100216	力觉引导对象执行中, Tz 扭矩的平均值
EndForcesFx	102411	力觉引导序列或力觉引导对象结束时 Fx 的力
EndForcesFy	102412	力觉引导序列或力觉引导对象结束时 Fy 的力
EndForcesFz	102413	力觉引导序列或力觉引导对象结束时 Fz 的力
EndForcesTx	102414	力觉引导序列或力觉引导对象结束时 Tx 的扭矩
EndForcesTy	102415	力觉引导序列或力觉引导对象结束时 Ty 的扭矩
EndForcesTz	102416	力觉引导序列或力觉引导对象结束时 Tz 的扭矩
EndPosX	102421	力觉引导对象结束时的位置 (X 坐标)
EndPosY	102422	力觉引导对象结束时的位置 (Y 坐标)
EndPosZ	102423	力觉引导对象结束时的位置 (Z 坐标)
EndPosU	102424	力觉引导对象结束时的位置 (U 坐标)
EndPosV	102425	力觉引导对象结束时的位置 (V 坐标)
EndPosW	102426	力觉引导对象结束时的位置 (W 坐标)
PeakForcesFx	105711	力觉引导序列或力觉引导对象执行中, Fx 力的峰值
PeakForcesFy	105712	力觉引导序列或力觉引导对象执行中, Fy 力的峰值
PeakForcesFz	105713	力觉引导序列或力觉引导对象执行中, Fz 力的峰值
PeakForcesTx	105714	力觉引导序列或力觉引导对象执行中, Tx 扭矩的峰值

PeakForcesTy	105715	力觉引导序列或力觉引导对象执行中, Ty 扭矩的峰值
PeakForcesTz	105716	力觉引导序列或力觉引导对象执行中, Tz 扭矩的峰值
TriggeredForcesFx	109411	力觉引导对象的力达成结束条件时 Fx 的力
TriggeredForcesFy	109412	力觉引导对象的力达成结束条件时 Fy 的力
TriggeredForcesFz	109413	力觉引导对象的力达成结束条件时 Fz 的力
TriggeredForcesTx	109414	力觉引导对象的力达成结束条件时 Tx 的扭矩
TriggeredForcesTy	109415	力觉引导对象的力达成结束条件时 Ty 的扭矩
TriggeredForcesTz	109416	力觉引导对象的力达成结束条件时 Tz 的扭矩
TriggeredPosX	109421	力觉引导对象的力达成结束条件时的位置 (X 坐标)
TriggeredPosY	109422	力觉引导对象的力达成结束条件时的位置 (Y 坐标)
TriggeredPosZ	109423	力觉引导对象的力达成结束条件时的位置 (Z 坐标)
TriggeredPosU	109424	力觉引导对象的力达成结束条件时的位置 (U 坐标)
TriggeredPosV	109425	力觉引导对象的力达成结束条件时的位置 (V 坐标)
TriggeredPosW	109426	力觉引导对象的力达成结束条件时的位置 (Z 坐标)

14.16.13 SpellHand 枚举

成员名称	值	描述
Righty	1	右手姿态。
Lefty	2	左手姿态。

14.16.14 SpellOLabelTypes 枚举

成员名称	值	描述
InputBit	1	指定输入位。
InputByte	2	指定输入字节。
InputWord	3	指定输入字。
OutputBit	4	指定输出位。
OutputByte	5	指定输出字节。
OutputWord	6	指定输出字。
MemoryBit	7	指定内存位。
MemoryByte	8	指定内存字节。
MemoryWord	9	指定内存字。
InputReal	10	指定实数输入。
OutputReal	11	指定实数输出。

14.16.15 SpellLocalDefType 枚举

成员名称	值	描述
J5Camera	1	定义 J5 安装相机的本地
J6Camera	2	定义 J6 安装相机的本地
FixedUpwardCamera	3	使用向上固定相机, 定义本地
FixedDownwardCamera	4	使用向下固定相机, 定义本地

14.16.16 SpelOperationMode 枚举

成员名称	值	描述
Auto	1	EPSON RC+ 7.0 处于 auto 模式。
Program	2	EPSON RC+ 7.0 处于 program 模式。

14.16.17 SpelOptions 枚举

成员名称	值	描述
ECP	1	ECP 选件
API	2	RC+ API 选件
PCVision	3	PC 视觉选件
ConveyorTracking	5	传送带跟踪选件
GUIBuilder	6	GUI Builder 选件
OCR	7	OCR 选件
FieldbusMaster	8	现场总线主站选件
LegacyForceSensing	9	Force Sensing 选件
PartFeeding	11	Part Feeding 选件
ThirdPartyForceSensors	13	第三方力觉传感器选件

14.16.18 SpelOptionStatus 枚举

成员名称	值	描述
Inactive	0	无效
Active	1	有效

14.16.19 SpelRobotPosType 枚举

成员名称	值	描述
World	0	指定全局坐标。
Joint	1	指定关节坐标。
Pulse	2	指定脉冲。

14.16.20 SpelRobotType 枚举

成员名称	值	描述
Joint	1	机器人类型为 joint。
Cartesian	2	机器人类型为 Cartesian。
Scara	3	机器人类型为 SCARA。
Cylindrical	4	机器人类型为 Cylindrical。
SixAxis	5	机器人类型为 6 轴式。
RS	6	机器人类型为 SCARA RS 系列。
N	7	机器人类型为 N 系列

14.16.21 SpelShutdownMode 枚举

成员名称	值	描述
ShutdownWindows	0	将关闭 Windows。
RebootWindows	1	将重启 Windows。

14.16.22 SpelSimObjectType 枚举

成员名称	值	描述
Unknown	-1	未设置对象类型
Layout	0	布局对象
Part	1	零件对象
MountedDevice	3	手臂安装设备

14.16.23 SpelSimProps 枚举

成员名称	值	描述
PositionX	100	X 坐标位置
PositionY	200	Y 坐标位置
PositionZ	300	Z 坐标位置
RotationX	400	X 轴旋转角度
RotationY	500	Y 轴旋转角度
RotationZ	600	Z 轴旋转角度
CollisionCheck	700	开启或关闭碰撞检测
CollisionCheckSelf	800	开启或关闭机器人自身碰撞检测
Visible	900	显示或隐藏状态
Type	1000	对象的类型
HalfSizeX	1500	Box 对象 X 方向的长度
HalfSizeY	1600	Box 对象 Y 方向的长度
HalfSizeZ	1700	Box 对象 Z 方向的长度
HalfSizeHeight	1800	Plane 对象的高度
HalfSizeWidth	1900	Plane 对象的宽度
PlaneType	2000	Plane 对象的类型
Radius	2100	Sphere 或 Cylinder 对象的半径
Height	2200	Cylinder 对象的高度
Name	2300	对象名称
Color	2400	对象的显示颜色

14.16.24 SpelStopType 枚举

成员名称	值	描述
StopNormalTasks	0	仅停止正常任务(非后台任务)。
StopAllTasks	1	停止所有任务，包括后台任务。

14.16.25 SpelTaskState 枚举

成员名称	值	描述
Quit	0	任务处于 quit 状态。
Run	1	任务处于 run 状态。
Aborted	2	任务被中止。
Finished	3	任务已完成。
Breakpoint	4	任务位于断点处。
Halt	5	任务处于 halt 状态。
Pause	6	任务处于 pause 状态。
Step	7	任务正在执行 step。
Walk	8	任务正在执行 walk。
Error	9	任务处于错误状态。
Waiting	10	任务处于 wait 状态。

14.16.26 SpelTaskType 枚举

成员名称	值	描述
Normal	0	任务为正常任务。
NoPause	1	任务不受 pause 影响。
NoEmgAbort	2	任务不受紧急停止影响。

14.16.27 SpelToolDefType 枚举

成员名称	值	描述
J4Camera	1	定义 J4 安装相机的工具。
J6Camera	2	定义 J6 安装相机的工具。
FixedCamera	3	通过使用未校准的固定相机定义工具。
FixedCameraWithCal	4	通过使用已校准的向上相机定义工具。

14.16.28 SpelToolDefType3D 枚举

成员名称	值	描述
Bar	1	为条形类型定义 3D 工具。
Plane	2	为平面类型定义 3D 工具。

14.16.29 SpelUserRights 枚举

成员名称	值	描述
All	-1	用户具有所有权限。
None	0	用户没有权限。
EditSecurity	1	用户可以配置安全性。
SysConfig	2	用户可以更改系统配置。
EditPrograms	4	用户可以编辑程序。
EditPoints	8	用户可以编辑点。
EditVision	16	用户可以更改视觉属性。
JogAndTeach	32	用户可以步进和示教。
CommandWindow	64	用户可以使用命令窗口。
EditRobotParameters	128	用户可以编辑机器人参数。
ConfigureOptions	256	用户可以配置选项。
ViewAudit	512	用户可以查看安全审计日志。
EditProject	1024	用户可以编辑项目配置。
DeleteAudit	2048	用户可以删除安全审计日志条目。
TeachPoints	4096	用户可以示教点。
ChangeOutputs	8192	用户可以更改输出状态。
ChangeMemIO	16384	用户可以更改内存 I/O 状态。
EditGUIBuilder	32768	用户可以在 GUI Builder 中进行更改。
EditForce	65536	用户可以在 Force Guide 和 Force Control 中进行更改。
EditPartFeeding	131072	用户可以在 Part Feeding 中进行更改。

14.16.30 SpelVDefShowWarning 枚举

成员名称	值	描述
None	-1	不显示警告。
Always	0	始终显示警告。
DependsOnSpeed	1	RobotSpeed 或 RobotAccel 大于 5 时显示。

14.16.31 SpelVisionImageSize 枚举

成员名称	值	描述
Size320x240	1	320 x 240 图像尺寸
Size640x480	2	640 x 480 图像尺寸
Size800x600	3	800 x 600 图像尺寸
Size1024x768	4	1024 x 768 图像尺寸
Size1280x1024	5	1280 x 1024 图像尺寸
Size1600x1200	6	1600 x 1200 图像尺寸
Size2048x1536	7	2048 x 1536 图像尺寸
Size2560x1920	8	2560 x 1920 图像尺寸
Size3664x2748	9	3664 x 2748 图像尺寸
Size5472x3648	10	5472 x 3648 图像尺寸
Size4024x3036	11	4024 x 3036 图像尺寸

14.16.32 SpelVisionObjectTypes 枚举

成员名称	值	描述
Correlation	1	Correlation 对象
Blob	2	Blob 对象
Edge	3	Edge 对象
Polar	4	Polar 对象
Line	5	Line 对象
Point	6	Point 对象
Frame	7	Frame 对象
ImageOp	8	ImageOp 对象
OCR	9	OCR 对象
CodeReader	10	CodeReader 对象
Geometric	11	Geometric 对象
ColorMatch	14	Color Match 对象
LineFinder	15	Line Finder 对象
ArcFinder	16	Arc Finder 对象
DefectFinder	17	Defect Finder 对象
LineInspector	18	Line Inspector 对象
ArcInspector	19	Arc Inspector 对象
BoxFinder	20	Box Finder 对象
CornerFinder	21	Corner Finder 对象
Contour	22	Contour 对象
Text	23	Text 对象
Decision	26	Decision 对象
Coordinates	27	Coordinates 对象

14.16.33 SpelVisionProps 枚举

此枚举适用于所有视觉属性和结果。有关详细信息，请参阅 Vision Guide Reference 手册。

14.16.34 SpelWrist 枚举

成员名称	值	描述
NoFlip	1	手腕方向为无翻转。
Flip	2	手腕方向为翻转。

14.16.35 SpelWindows 枚举

成员名称	值	描述
IOMonitor	1	I/O 监视器窗口 ID
TaskManager	2	任务管理器窗口 ID
ForceMonitor	3	力监视器窗口 ID
Simulator	4	仿真器窗口 ID

14.17 Spel 错误代码和消息

有关错误代码和错误消息，请参阅《状态和错误代码》手册。

15. 32 位和 64 位应用

EPSON RC+ 7.0 版本 7.1.0 和更高版本提供的 RCAPINet 库自动支持 32 位和 64 位应用。

在早于 7.1.0 的 EPSON RC+ 7.0 版本中，分别提供支持 32 位和 64 位的库。这些旧式库(SpelNetLib70.dll 和 SpelNetLib70_x64.dll) 因兼容目的在版本 7.5.0 仍提供。有关使用旧式库的详细信息，请参阅您所使用之前版本的 EPSON RC+ 7.0 的 RC+ API 手册。

16. 使用 LabVIEW VI Library

16.1 概述

在早于 v7.1.0 的 EPSON RC+ 7.0 版本中，API .NET 库可直接在 LabVIEW 中使用。在 EPSON RC+ 7.0 v7.1.0 中引进了新 LabVIEW VI 库。新的库具有以下特点：

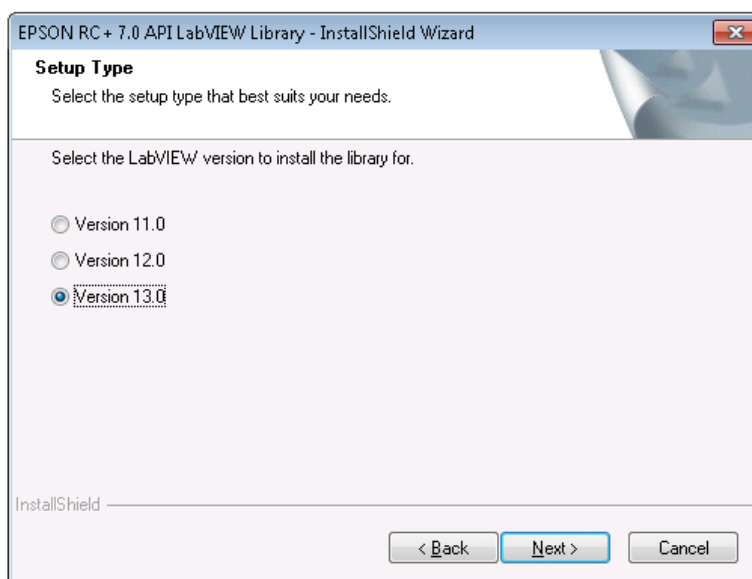
- 使用 VI(虚拟仪器)的 EPSON RC+ 7.0 高级界面。
- 用户无需操作 EPSON RC+ 7.0 的 .NET 界面 - 将自动处理。
- 每个 Spel 命令被包装在单个 VI 内。
- VI 被组合在几个工具面板中。
- 支持 32 位和 64 位 LabVIEW 应用。
- 支持 LabVIEW 版本 2009 和更高版本。

若要使用 LabVIEW VI 库，必须为连接的每个控制器购买 EPSON RC+ 7.0 API 软件许可。

16.2 安装

若要使用 EPSON RC+ 7.0 LabVIEW VI 库，必须使用您 PC 的 \EpsonRC70\API\LabVIEW 文件夹中提供的安装程序进行安装。

1. 安装 LabVIEW 版本 2009 或更高版本。
2. 导航至您 PC 上的 \EpsonRC70\API\LabVIEW 文件夹，并运行 EpsonRC70_vxxx_LabVIEW.exe 安装程序，xxx 是 EPSON RC+ 7.0 的版本。例如，EpsonRC70_v710_LabVIEW.exe。
3. 安装程序启动时，将显示检测到的您 PC 上安装的 LabVIEW 版本。默认选择最新版本。选择要使用 EPSON RC+ 7.0 LabVIEW VI 库的版本。



4. 点击 Next，然后点击 Install。将安装所选 LabVIEW 版本的 VI、控制和面板。

16.3 工具和控制面板

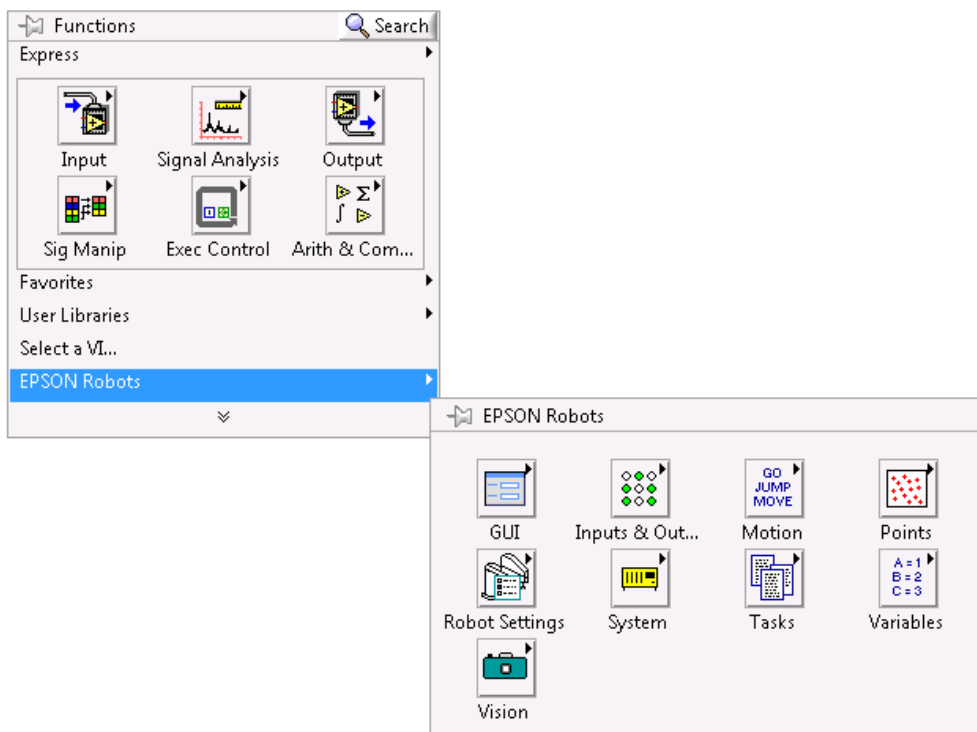
安装 EPSON RC+ 7.0 LabVIEW VI 库后，可以从[EPSON Robots]工具面板和 [EPSON Robots]控制面板访问库中可用的 VI 和控制。

工具面板

从方框图访问工具面板。Epson Robots 工具面板有如下所示的几个子面板：

面板	描述
System	用于初始化和关闭 API。
Robot Settings	更改机器人参数。
Points	加载、保存和更改机器人点。
Motion	执行机器人动作。
Inputs & Outputs	控制和监视控制器输入和输出。
Tasks	管理机器人控制器中的任务。
Variables	在控制器中读取和写入变量。
Vision	执行视觉命令。
GUI	显示 GUI 功能。

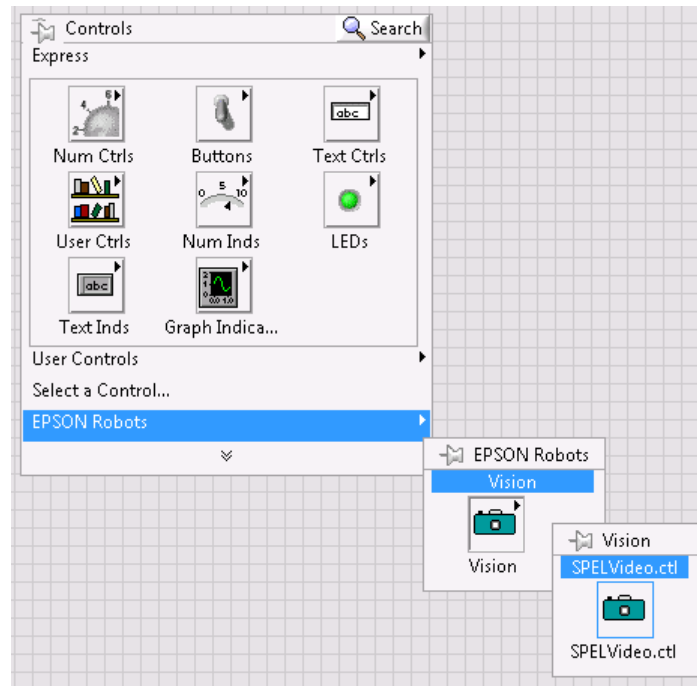
若要访问[Epson Robots]工具面板，打开 VI 方框图，然后右键单击空白区域并选择 [Epson Robots]，即可看见上面介绍的子面板。



控制面板

从前面板访问控制面板。

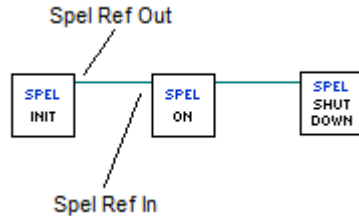
面板	描述
Vision	含有用于显示视频的 SPELVideo 控制。



16.4 操作入门

若要使用 LabVIEW VI 库，应用必须首先为要使用的每个控制器调用 Spel Initialize VI。Initialize VI 启动将连接至机器人控制器并处理后续 Spel command VI 的 EPSON RC+ 7.0 服务器进程。Initialize VI 有 Spel Ref Out 输出。这必须连接至下一个 Spel VI Spel Ref In 输入。然后对于每个后续 VI，来自上一个 Spel VI 的 Spel Ref Out 输出必须连接至下一个 Spel VI 的 Spel Ref In 输入。

例如，以下流程图中所示为每个 Spel 节点之间的 Spel Ref Out 和 Spel Ref In 连接。



应用关闭时，必须调用 Spel Shutdown VI。这将从机器人控制器断开连接，并关闭相关的 EPSON RC+ 7.0 服务器进程。

按照以下步骤启动。首先从 EPSON RC+ 7.0 GUI 在 LabVIEW 默认 Spel+ 项目创建两个安全机器人点。然后可以在 LabVIEW 创建在两点之间移动机器人的小应用。

1. 确保已在您 PC 上安装 EPSON RC+ 7.0 和 EPSON RC+ 7.0 LabVIEW VI 库。有关安装 LabVIEW VI 库的详细信息，请参阅章节 16.2。
2. 启动 EPSON RC+ 7.0。
3. 从 Project 菜单选择 Open，然后导航至 LabVIEW 文件夹，并选择 LabVIEW_Default 项目。点击 Open。
4. 从 Tools 菜单选择 Robot Manager。点击 Motor On 按钮。
5. 在 Robot Manager 选择 Jog & Teach 页面。步进机器人至安全位置。
6. 点击 Teach 示教点 0。
7. 步进机器人至另一个安全位置。
8. 从点列表中选择 P1，然后点击 Teach 示教点 1。
9. 点击主工具栏上的 Save 按钮保存点。
10. 关闭 EPSONRC+ 7.0。
11. 启动 LabVIEW，并创建新 VI。
12. 打开新 VI 的方框图。
13. 从 Epson RC+ API | System 工具面板将 Init VI 拖至方框图中。您接口的每个控制器需要 Initialize VI。
14. 从 Epson RC+ API | Robot Settings 工具面板将 MotorOn VI 拖至方框图。将来自 Initialize VI 的 Spel Ref Out 输出连接至 MotorOn VI 的 Spel Ref In 输入。
15. 从 Epson RC+ API | Motion 工具面板将 Go VI 拖至方框图中。将来自 MotorOn VI 的 Spel Ref Out 输出连接至 Go VI 的 Spel Ref In 输入。在 Point Number 输入中添加常量，并将值设为 0。
16. 从 Epson RC+ API | Motion 工具面板将另一个 Go VI 拖至方框图中。将上一个 Go VI 的 Spel Ref Out 输出连接至第二个 Go VI 的 Spel Ref In 输入。在 Point Number 输入中添加常量，并将值设为 1。

17. 从 Epson RC+ API | Robot Settings 工具面板将 MotorOff VI 拖至方框图。将来自 Go VI 的 Spel Ref Out 输出连接至 MotorOff VI 的 Spel Ref In 输入。
18. 从 Epson RC+ API | System 工具面板将 Shutdown VI 拖至方框图中。Shutdown VI 必须用于每个 Init VI。
方框图应类似于此：



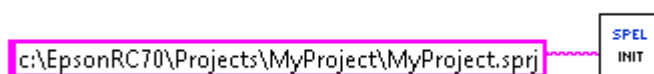
19. 运行应用。机器人电机打开，然后机器人移动至点 0，接着移动至点 1 后，机器人电机关闭。

16.5 在Spel+ 项目中作业

在 LabVIEW 应用中使用 Spel+ 项目为可选。但若将保存点数据，或想要使用点标签和/或 I/O 标签、任务或者视觉序列，则需要使用 Spel+ 项目。

默认项目为 LabVIEW_Default，位于 \EpsonRC70\Projects\LabVIEW 文件夹中。

可以根据需要使用 EPSON RC+ 7.0 创建自己的项目，然后通过 Initialize VI *Project* 输入参数指定要使用的项目，如下所示：



若要使用 EPSON RC+ 7.0 项目作业，启动 EPSON RC+ 7.0 应用。使用 Project 菜单创建、打开和编辑项目。有关更多详细信息，请参阅 EPSON RC+ 7.0 用户指南。

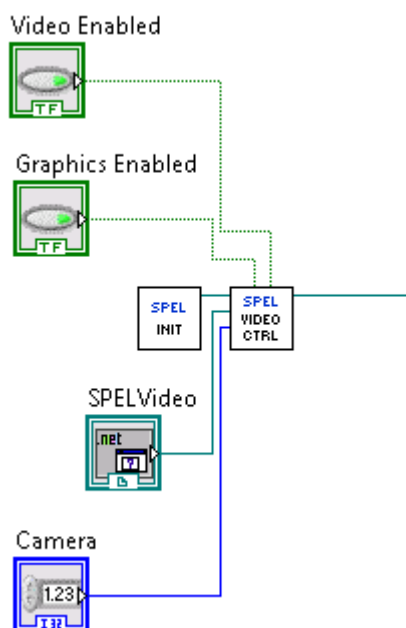
16.6 播放视频

可以使用 SPELVideo 控制和 VideoControl VI 在 Vision Guide 序列显示视频。

播放视频：

1. 在前面板中添加 SPELVideo 控制。
2. 在相应的方框图中添加 VideoControl.vi。
3. 将来自 SPELVideo 控制的输出连接至 VideoControl VI 的 SPELVideo Ref In 输入。
4. 将 VideoControl VI 的 Spel Ref In 和 Spel Ref Out 参数相连。
5. 添加 VideoControl VI 的 *Camera*、*Graphics Enabled* 和 *Video Enabled* 参数的常量或控制。为显示视频，*Video Enabled* 必须设为 true。

以下流程图所示为 SPELVideo 控制和 SPEL VideoControl VI 的连接。



Video Enabled 为 true 时，从 VRun VI 或在控制器任务中执行 VRun，您可以看到根据 Camera 设置产生的视频。

默认 *Camera* 输入参数为零，允许显示来自任何相机的视频。若在 *Camera* 设置非零的数字，则将显示使用指定相机的序列视频。

Graphics Enabled 为 true 时若执行 VRun，将在视频图像上显示序列结果图形。在应用中一次仅可以使用一个 SPEL Video 控制。

16.7 VI 参考

本章节包含在 EPSON RC+ 7.0 LabVIEW VI 库使用的所有 VI 的信息。

每个 VI 提供以下信息：

工具面板	含有 VI 的工具面板。
描述	函数的简要描述。
输入	输入参数
输出	输出参数
备注	附加详细信息。

Accel VI

工具面板

Epson Robots | Robot Settings

描述

设置当前机器人 PTP 的加速度和减速度。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Accel</i>	表示 PTP 加速度的整数值。
<i>Decel</i>	表示 PTP 减速度的整数值。
<i>Depart Accel</i>	可选。表示 Jump 起始加速度的整数值。
<i>Depart Decel</i>	可选。表示 Jump 起始减速度的整数值。
<i>Appro Accel</i>	可选。表示 Jump 结束加速度的整数值。
<i>Appro Decel</i>	可选。表示 Jump 结束减速度的整数值。

输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

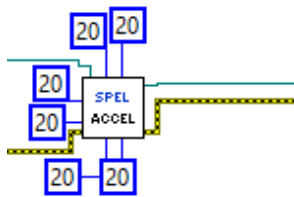
备注

使用 Accel 设置当前机器人 PTP 的加速度和减速度值。所有数值可为 1 至 100%。
若指定 Depart Accel，则也需指定余下的输入。

另见

AccelS, Speed, SpeedS

Accel 示例



AccelS VI

工具面板

Epson Robots | Robot Settings

描述

设置当前机器人的线性加速度和减速度。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Accel</i>	表示线性加速度的 Double 值。
<i>Decel</i>	表示线性减速度的 Double 值。
<i>Depart Accel</i>	可选。表示 Jump 起始加速度的 Double 值。
<i>Depart Decel</i>	可选。表示 Jump 起始减速度的 Double 值。
<i>Appro Accel</i>	可选。表示 Jump 结束加速度的 Double 值。
<i>Appro Decel</i>	可选。表示 Jump 结束减速度的 Double 值。

输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

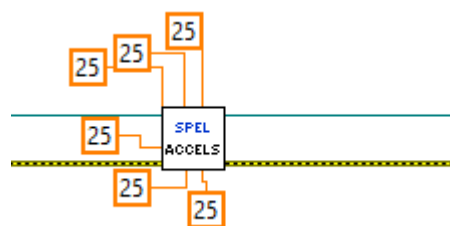
备注

使用 AccelS 设置当前机器人线性加速度和减速度值。所有值均以 mm/sec^2 为单位。若指定 Depart Accel，则也需指定余下的输入。

另见

Accel, Speed, SpeedS

AccelS 示例



Arc VI

工具面板

Epson Robots | Motion

描述

Arc 可利用 XY 平面内的圆弧插补将机械臂移至指定点处。

输入

- Spel Ref In* 来自之前 Spel Ref Out 的 Spel 参考。
- Error In* 来自之前 Spel 节点的错误状态。
- Mid Point Number* 通过使用整数指定中间点。
- Mid Point Expr* 通过使用字符串表达式指定中间点。若使用该输入，则必须也使用字符串表达式指定端点。
- End Point Number* 通过使用整数指定端点。
- End Point Expr* 通过使用字符串表达式指定端点。可包括 ROT、CP、SYNC、Till 搜索表达式以及并行处理语句。

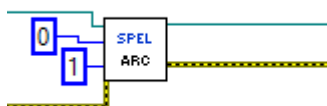
输出

- Spel Ref Out* 对使用的下一个 VI 的 Spel 参考输出。
- Error Out* 向后续 Spel 节点的错误条件输出。

另见

Accel, Arc3, BMove, Go, Jump, Jump3, Move, Speed, TGo, TMove

Arc 示例



Arc3 VI

工具面板

Epson Robots | Motion

描述

Arc3 可利用 3 维空间的圆弧插补将机械臂移至指定点。

输入

- Spel Ref In* 来自之前 Spel Ref Out 的 Spel 参考。
- Error In* 来自之前 Spel 节点的错误状态。
- Mid Point Number* 通过使用整数指定中间点。
- Mid Point Expr* 通过使用字符串表达式指定中间点。若使用该输入，则必须也使用字符串表达式指定端点。
- End Point Number* 通过使用整数指定端点。
- End Point Expr* 通过使用字符串表达式指定端点。可包括 CP、SYNC、Till 搜索表达式以及并行处理语句。

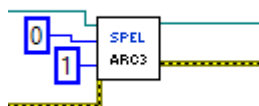
输出

- Spel Ref Out* 对使用的下一个 VI 的 Spel 参考输出。
- Error Out* 向后续 Spel 节点的错误条件输出。

另见

AccelS, Arc, BMove, Go, Jump, Jump3, Move, SpeedS, TGo, TMove

Arc3 示例



Arch VI

工具面板

Epson Robots | Robot Settings

描述

定义与 JUMP 指令一同使用的 ARCH 参数(开始水平动作前需移动的 Z 高度)。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Arch Number</i>	启动水平动作之前, Jump 指令开始时移动的起始距离, 单位为毫米。
<i>Depart Dist</i>	启动水平动作之前, Jump 指令开始时移动的起始距离, 单位为毫米。
<i>Appro Dist</i>	Jump 指令目标位置上方的结束距离, 单位为毫米。

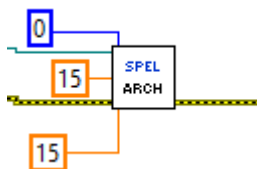
输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

另见

Jump, Jump3

Arch 示例



Arm VI

工具面板

Epson Robots | Robot Settings

描述

选择当前的机器人机械臂。

输入

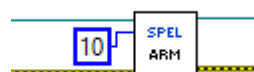
<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Arm Number</i>	整数范围为 0-15。用户最多可选择 16 个不同的机械臂。机械臂 0 是标准(默认)机器人机械臂。机械臂 1-15 是 ArmSet 指令定义的辅助机械臂。

输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

另见

Armset, GetArm, Tool

Arm 示例

Armset VI

工具面板

Epson Robots | Robot Settings

描述

定义辅助机器人机械臂。

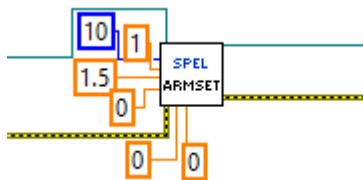
输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>ArmNumber</i>	整数值：1-15 的有效范围。
<i>Param1</i>	(对于 SCARA 机器人)肘关节中心线至新定向轴中心线的水平距离。(即新辅助机械臂的定向轴中心线所在的位置。) (对于 Cartesian 机器人)X 轴方向位置相对于 X 轴原点位置的偏移量，单位为 mm。
<i>Param2</i>	(对于 SCARA 机器人)正常肘中心线与正常定向轴中心线之间的成线以及新辅助机械臂肘中心线和新定向轴中心线之间的成线之间的偏移量(度)。(这 2 条线应在肘中心线处相交，所成角度即 <i>Param2</i> 。) (对于 Cartesian 机器人)Y 轴方向位置相对于 Y 轴原点位置的偏移量，单位为 mm。
<i>Param3</i>	(对于 SCARA 和 Cartesian 机器人)新定向轴中心与旧定向轴中心之间的 Z 高度偏移量。(此为距离量。)
<i>Param4</i>	(对于 SCARA 机器人)新辅助轴肩中心线至肘中心线的距离。 (对于 Cartesian 机器人)这是一个虚拟参数(指定为 0)。
<i>Param5</i>	(对于 SCARA 和 Cartesian 机器人)新定向轴与旧定向轴之间的角度偏移量(度)。

输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

ArmSet 示例



AtHome VI

工具面板

Epson Robots | Motion

描述

如果当前机器人位于起始点位置，则返回 True。

输入

Spel Ref In 来自之前 Spel Ref Out 的 Spel 参考。

Error In 来自之前 Spel 节点的错误状态。

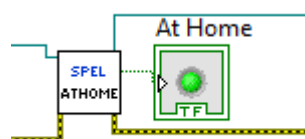
输出

Spel Ref Out 对使用的下一个 VI 的 Spel 参考输出。

Error Out 向后续 Spel 节点的错误条件输出。

At Home Boolean 指示当前机器人是否位于起始点位置。

AtHome 示例



AvoidSing VI

工具面板

Epson Robots | Motion

描述

在 Move、Arc 和 Arc3 动作方法中启用/禁用奇点回避功能。

输入

Spel Ref In 来自之前 Spel Ref Out 的 Spel 参考。

Error In 来自之前 Spel 节点的错误状态。

Enable True 启用奇点回避，False 禁用。

输出

Spel Ref Out 对使用的下一个 VI 的 Spel 参考输出。

Error Out 向后续 Spel 节点的错误条件输出。

AvoidSing 示例



BGo VI

工具面板

Epson Robots | Motion

描述

在选定的本地坐标系中执行 PTP 的相对运动。

输入

- Spel Ref In* 来自之前 Spel Ref Out 的 Spel 参考。
- Error In* 来自之前 Spel 节点的错误状态。
- Point Number* 可选。通过使用当前机器人控制器点内存中之前示教点的点编号来指定目标端点。若指定 *Point Expression*，则将 *Point Number* 忽略。
- Point Expression* 可选。通过使用字符串表达式来指定目标端点。若未指定 *Point Expression*，则使用 *Point Number* 输入。

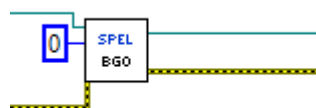
输出

- Spel Ref Out* 对使用的下一个 VI 的 Spel 参考输出。
- Error Out* 向后续 Spel 节点的错误条件输出。

另见

Accel, Arc, Arc3, BMove, Go, Jump, Jump3, Move, Speed, TGo, TMove

BGo 示例



BMove VI

工具面板

Epson Robots | Motion

描述

在选定的本地坐标系中执行线性内插相对运动。

输入

- Spel Ref In* 来自之前 Spel Ref Out 的 Spel 参考。
- Error In* 来自之前 Spel 节点的错误状态。
- Point Number* 可选。通过使用当前机器人控制器点内存中之前示教点的点编号来指定目标端点。若指定 *Point Expression*，则将 *Point Number* 忽略。
- Point Expression* 可选。通过使用字符串表达式来指定目标端点。若未指定 *Point Expression*，则使用 *Point Number* 输入。

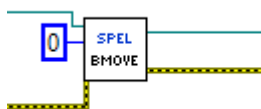
输出

- Spel Ref Out* 对使用的下一个 VI 的 Spel 参考输出。
- Error Out* 向后续 Spel 节点的错误条件输出。

另见

AccelS, Arc, Arc3, BGo, Go, Jump, Jump3, Move, SpeedS, TGo, TMove

BMove 示例



Box VI

工具面板

Epson Robots | Robot Settings

描述

指定 box 内定义的结束检查区域。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>AreaNumber</i>	1-15 之间表示 15 个待定义 box 的整数。
<i>Min X</i>	结束检查区域的最小 X 坐标位置。
<i>Max X</i>	结束检查区域的最大 X 坐标位置。
<i>Min Y</i>	结束检查区域的最小 Y 坐标位置。
<i>Max Y</i>	结束检查区域的最大 Y 坐标位置。
<i>Min Z</i>	结束检查区域的最小 Z 坐标位置。
<i>Max Z</i>	结束检查区域的最大 Z 坐标位置。
<i>Polarity On</i>	使用对应远程输出时设置远程输出逻辑。若要设置为夹具末端在 box 区域内时 I/O 输出打开，使用 True。若要设置为夹具末端在 box 区域内时 I/O 输出关闭，使用 False。

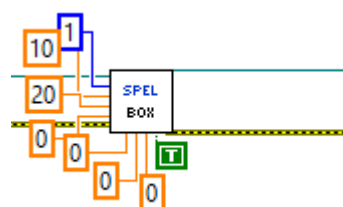
输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

另见

XYLim

Box 示例



Continue VI

工具面板

Epson Robots | Tasks

描述

使控制器中所有暂停的任务恢复。

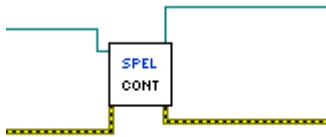
输入

Spel Ref In 来自之前 Spel Ref Out 的 Spel 参考。*Error In* 来自之前 Spel 节点的错误状态。

输出

Spel Ref Out 对使用的下一个 VI 的 Spel 参考输出。*Error Out* 向后续 Spel 节点的错误条件输出。

Continue 示例



Delay VI

工具面板

Epson Robots | System

描述

处理延迟指定的毫秒数。

输入

Spel Ref In 来自之前 Spel Ref Out 的 Spel 参考。*Error In* 来自之前 Spel 节点的错误状态。*Milliseconds* 待延迟毫秒数。

输出

Spel Ref Out 对使用的下一个 VI 的 Spel 参考输出。*Error Out* 向后续 Spel 节点的错误条件输出。

Delay 示例



ECP VI

工具面板

Epson Robots | Robot Settings

描述

选择当前的 ECP 定义。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>ECPNumber</i>	0-15 之间表示 16 个 ECP 定义中将与下一动作指令一同使用的定义的整数。

输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

另见

ECPSet, GetECP

ECP 示例



ECPset VI

工具面板

Epson Robots | Robot Settings

描述

定义 ECP(外部控制点)。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>ECPNumber</i>	1-15 之间表示 15 个外部控制点中待定义点的整数值。
<i>X</i>	外部控制点 X 坐标。
<i>Y</i>	外部控制点 Y 坐标。
<i>Z</i>	外部控制点 Z 坐标。
<i>U</i>	外部控制点 U 坐标。
<i>V</i>	外部控制点 V 坐标。
<i>W</i>	外部控制点 W 坐标。

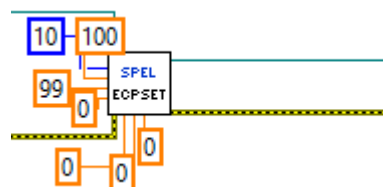
输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

另见

ECP, GetECP

ECPSet 示例



EStopOn VI

工具面板

Epson Robots | System

描述

返回紧急停止状态。

输入

Spel Ref In 来自上一个 Spel Ref Out 的 Spel 参考

Error In 来自上一个 Spel 节点的错误状态

输出

Spel Ref Out 对使用的下一个 VI 的 Spel 参考输出

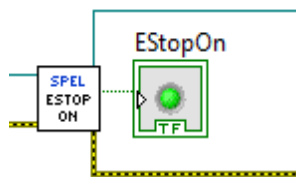
Error Out 向后续 Spel 节点的错误条件输出

EStopOn 紧急停止状态时为 True，否则为 False

另见

SafetyOn

EStopOn 示例



Find VI

工具面板

Epson Robots | Motion

描述

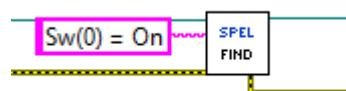
指定动作期间保存坐标的条件。

输入*Spel Ref In* 来自之前 Spel Ref Out 的 Spel 参考。*Error In* 来自之前 Spel 节点的错误状态。*Condition* 含有函数和运算符的字符串表达式。**输出***Spel Ref Out* 对使用的下一个 VI 的 Spel 参考输出。*Error Out* 向后续 Spel 节点的错误条件输出。**备注**

使用 Find 指定在动作期间何时应存储位置。满足条件时，当前位置存储在 FindPos 中。

另见

FindPos

Find 示例

Fine VI

工具面板

Epson Robots | Robot Settings

描述

指定并显示目标点的定位精度。

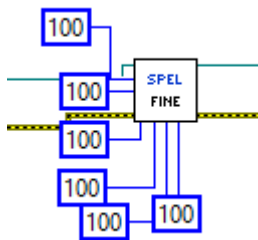
输入

Spel Ref In 来自之前 Spel Ref Out 的 Spel 参考。*Error In* 来自之前 Spel 节点的错误状态。*J1MaxErr* - *J9MaxErr* 0-32767 之间表示各关节允许定位误差的整数值。关节 7、8 和 9 的此数值是可选的。

输出

Spel Ref Out 对使用的下一个 VI 的 Spel 参考输出。*Error Out* 向后续 Spel 节点的错误条件输出。

Fine 示例



GetArm VI

工具面板

Epson Robots | Robot Settings

描述

返回当前机器人的当前机械臂编号。

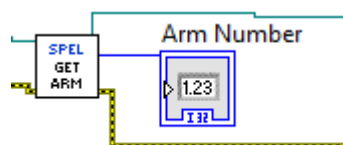
输入

Spel Ref In 来自之前 Spel Ref Out 的 Spel 参考。*Error In* 来自之前 Spel 节点的错误状态。

输出

Spel Ref Out 对使用的下一个 VI 的 Spel 参考输出。*Error Out* 向后续 Spel 节点的错误条件输出。*Arm Number* 当前机械臂编号。

GetArm 示例



GetAvoidSing VI

工具面板

Epson Robots | Motion

描述

AvoidSingularity 返回返回值

输入

Spel Ref In 来自上一个 Spel Ref Out 的 Spel 参考

Error In 来自上一个 Spel 节点的错误状态

输出

Spel Ref Out 对使用的下一个 VI 的 Spel 参考输出

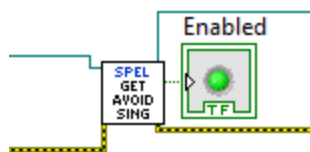
Error Out 对后续 Spel 节点的错误状态输出

EStopOn 紧急停止状态时为 True，否则为 False

另见

AvoidSing

GetAvoidSing 示例



GetECP VI

工具面板

Epson Robots | Robot Settings

描述

返回当前机器人的当前 ECP 编号。

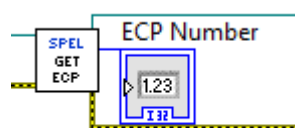
输入

Spel Ref In 来自之前 Spel Ref Out 的 Spel 参考。*Error In* 来自之前 Spel 节点的错误状态。

输出

Spel Ref Out 对使用的下一个 VI 的 Spel 参考输出。*Error Out* 向后续 Spel 节点的错误条件输出。*ECP Number* 当前 ECP 编号。

GetECP 示例



GetLimZ VI

工具面板

Epson Robots | Motion

描述

返回当前的 LimZ 设置

输入

Spel Ref In 来自上一个 Spel Ref Out 的 Spel 参考

Error In 来自上一个 Spel 节点的错误状态

输出

Spel Ref Out 对使用的下一个 VI 的 Spel 参考输出

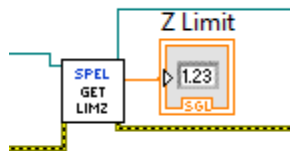
Error Out 向后续 Spel 节点的错误状态输出

Z Limit 当前的 LimZ 设置

另见

LimZ

GetLimZ 示例



GetMotor VI

工具面板

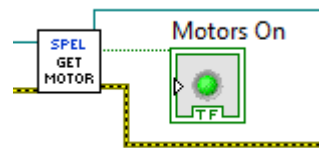
Epson Robots | Robot Settings

描述

返回当前机器人的电机开启状态。

输入*Spel Ref In* 来自之前 Spel Ref Out 的 Spel 参考。*Error In* 来自之前 Spel 节点的错误状态。**输出***Spel Ref Out* 对使用的下一个 VI 的 Spel 参考输出。*Error Out* 向后续 Spel 节点的错误条件输出。*Motors On* 电机打开为 True，关闭为 false。**另见**

GetPower, MotorOn, MotorOff

GetMotor 示例

GetOprMode VI

工具面板

Epson Robots | System

描述

读取 EPSON RC+ 7.0 的操作模式。

输入

Spel Ref In 来自之前 Spel Ref Out 的 Spel 参考。

Error In 来自之前 Spel 节点的错误状态。

输出

Spel Ref Out 对使用的下一个 VI 的 Spel 参考输出。

Error Out 向后续 Spel 节点的错误条件输出。

Operation Mode 相关 EPSON RC+ 7.0 服务器进程的操作模式。

模式

ID 描述

Auto	1	EPSON RC+ 7.0 处于 auto 模式。
Program	2	EPSON RC+ 7.0 处于 program 模式。

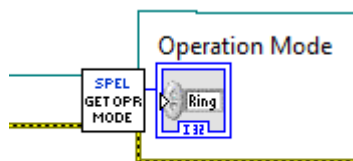
备注

当 *Operation Mode* 设至 Program 时，相关服务器进程的 EPSON RC+ 7.0 GUI 会打开且控制器操作模式会设至 Program。如果用户关闭 RC+ GUI，*Operation Mode* 会设至 Auto。若 *Operation Mode* 设为 Auto，RC+ GUI 也关闭。

另见

OprMode

GetOprMode 示例



GetPoint VI

工具面板

Epson Robots | Points

描述

检索机器人点的坐标数据。

输入

- Spel Ref In* 来自之前 Spel Ref Out 的 Spel 参考。
- Error In* 来自之前 Spel 节点的错误状态。
- Point Number* 可选。通过使用当前机器人控制器点内存中之前示教点的点编号来指定目标端点。若指定 *Point Expression*，则将 *Point Number* 忽略。
- Point Expression* 可选。通过使用字符串表达式来指定目标端点。若未指定 *Point Expression*，则使用 *Point Number* 输入。

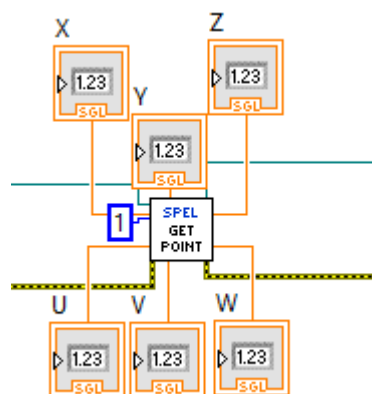
输出

- Spel Ref Out* 对使用的下一个 VI 的 Spel 参考输出。
- Error Out* 向后续 Spel 节点的错误条件输出。
- X - W* 指定点的 X、Y、Z、U、V、W 坐标。

另见

LoadPoints, Robot, SavePoints, SetPoint

GetPoint 示例



GetPower VI

工具面板

Epson Robots | Robot Settings

描述

返回当前机器人的电量 high 状态。

输入

Spel Ref In 来自之前 Spel Ref Out 的 Spel 参考。

Error In 来自之前 Spel 节点的错误状态。

输出

Spel Ref Out 对使用的下一个 VI 的 Spel 参考输出。

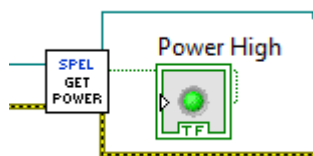
Error Out 向后续 Spel 节点的错误条件输出。

Power High 高功率为 True，不是为 false。

另见

PowerHigh, PowerLow

GetPower 示例



GetRobot VI

工具面板

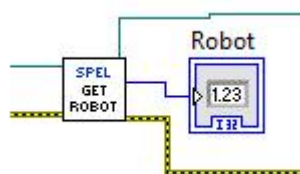
Epson Robots | Robot Settings

描述

返回当前机器人的编号。

输入*Spel Ref In* 来自之前 Spel Ref Out 的 Spel 参考。*Error In* 来自之前 Spel 节点的错误状态。**输出***Spel Ref Out* 对使用的下一个 VI 的 Spel 参考输出。*Error Out* 向后续 Spel 节点的错误条件输出。*Robot Number* 当前机器人编号。**另见**

Robot

GetRobot 示例

GetTool VI

工具面板

Epson Robots | Robot Settings

描述

返回当前机器人的当前工具编号。

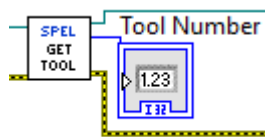
输入

Spel Ref In 来自之前 Spel Ref Out 的 Spel 参考。*Error In* 来自之前 Spel 节点的错误状态。

输出

Spel Ref Out 对使用的下一个 VI 的 Spel 参考输出。*Error Out* 向后续 Spel 节点的错误条件输出。*Tool Number* 当前工具编号。

GetTool 示例



GetVar VI

工具面板

Epson Robots | Variables

描述

返回控制器中 SPEL+ 全局保留变量的值。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Var Name</i>	SPEL+ 全局保留变量的名称。对于数组，可返回整个数组或仅返回一个元素。

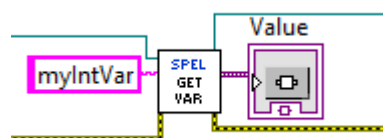
输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。
<i>Value</i>	含有值的变量。

另见

SetVar

GetVar 示例



工具面板

Epson Robots | Motion

描述

以 PTP 的形式将机械臂从当前位置移至指定点或 XY 位置。**GO** 指令可同时移动任意组合的机器人轴。

输入

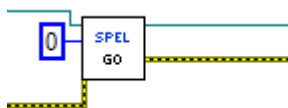
- Spel Ref In* 来自之前 Spel Ref Out 的 Spel 参考。
- Error In* 来自之前 Spel 节点的错误状态。
- Point Number* 可选。通过使用当前机器人控制器点内存中之前示教点的点编号来指定目标端点。若指定 *Point Expression*，则将 *Point Number* 忽略。
- Point Expression* 可选。通过使用字符串表达式来指定目标端点。若未指定 *Point Expression*，则使用 *Point Number* 输入。

输出

- Spel Ref Out* 对使用的下一个 VI 的 Spel 参考输出。
- Error Out* 向后续 Spel 节点的错误条件输出。

另见

Accel, Arc, Arc3, BGo, BMove, Jump, Jump3, Move, Speed, TGo, TMove

Go 示例

Halt VI

工具面板

Epson Robots | Tasks

描述

暂停指定任务的执行。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Task Number</i>	可选。待暂停任务的任务号。任务号范围为 1 至 32。若指定 <i>Task Name</i> ，则将 <i>Task Number</i> 忽略。
<i>Task Name</i>	可选。指定待暂停任务的名称。若未指定 <i>Task Name</i> ，则使用 <i>Task Number</i> 输入。

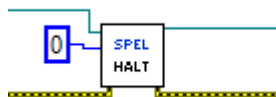
输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

另见

Quit, Resume

Halt 示例



Here VI

工具面板

Epson Robots | Points

描述

示教当前位置的点。

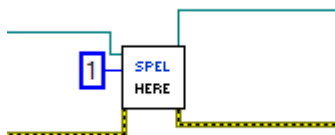
输入

- Spel Ref In* 来自之前 Spel Ref Out 的 Spel 参考。
- Error In* 来自之前 Spel 节点的错误状态。
- Point Number* 可选。通过使用当前机器人控制器点内存中之前示教点的点编号来指定目标端点。若指定 *Point Expression*，则将 *Point Number* 忽略。
- Point Name* 可选。指定点的名称。若未指定 *Point Name*，则使用 *Point Number* 输入。

输出

- Spel Ref Out* 对使用的下一个 VI 的 Spel 参考输出。
- Error Out* 向后续 Spel 节点的错误条件输出。

Here 示例



HideWindow VI

工具面板

Epson Robots | GUI

描述

隐藏之前用 ShowWindow 显示的 EPSON RC+ 7.0 窗口。

输入

Spel Ref In 来自之前 Spel Ref Out 的 Spel 参考。

Error In 来自之前 Spel 节点的错误状态。

Window ID 待显示 EPSON RC+ 7.0 窗口的 ID。

窗口名称	ID	描述
IOMonitor	1	I/O 监视器窗口 ID
TaskManager	2	任务管理器窗口 ID
ForceMonitor	3	力监视器窗口 ID
Simulator	4	仿真器窗口 ID

输出

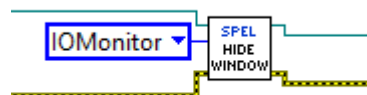
Spel Ref Out 对使用的下一个 VI 的 Spel 参考输出。

Error Out 向后续 Spel 节点的错误条件输出。

另见

RunDialog, ShowWindow

HideWindow 示例



工具面板

Epson Robots | Inputs & Outputs

描述

返回指定输入端口的状态。每个端口含有 8 个输入位(一个字节)。

输入

- Spel Ref In* 来自之前 Spel Ref Out 的 Spel 参考。
- Error In* 来自之前 Spel 节点的错误状态。
- Port Number* 可选。表示其中一个输入端口的整数。每个端口含有 8 个输入位(一个字节)。若未指定 *Label*，则使用 *Port Number*。
- Label* 可选。含有输入字节标签的字符串。若指定 *Label*，则将 *Port Number* 忽略。

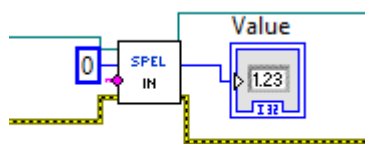
输出

- Spel Ref Out* 对使用的下一个 VI 的 Spel 参考输出。
- Error Out* 向后续 Spel 节点的错误条件输出。
- Value* 0 至 255 之间表示输入端口状态的整数。

另见

InBCD, InW, Sw

In 示例



InBCD VI

工具面板

Epson Robots | Inputs & Outputs

描述

使用 BCD 格式返回 8 个输入的输入状态。(二进制编码的十进制)

输入

- Spel Ref In* 来自之前 Spel Ref Out 的 Spel 参考。
- Error In* 来自之前 Spel 节点的错误状态。
- Port Number* 可选。表示其中一个输入端口的整数。每个端口含有 8 个输入位 (一个字节)。若未指定 *Label*, 则使用 *Port Number*。
- Label* 可选。含有输入字节标签的字符串。若指定 *Label*, 则将 *Port Number* 忽略。

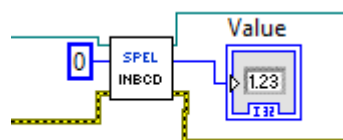
输出

- Spel Ref Out* 对使用的下一个 VI 的 Spel 参考输出。
- Error Out* 向后续 Spel 节点的错误条件输出。
- Value* 0 至 9 之间表示输入端口状态的整数。

另见

In, InW, Sw

InBCD 示例



InsideBox VI

工具面板

Epson Robots | Motion

描述

返回当前机器人夹具末端是否位于指定 box 区域内。

输入

Spel Ref In 来自之前 Spel Ref Out 的 Spel 参考。

Error In 来自之前 Spel 节点的错误状态。

Area Number 1-15 之间表示 15 个待检查 box 的整数编号。

输出

Spel Ref Out 对使用的下一个 VI 的 Spel 参考输出。

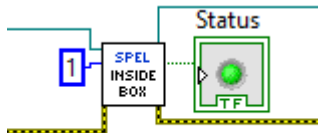
Error Out 向后续 Spel 节点的错误条件输出。

Status 机器人夹具末端位于 box 内时为 True 的 Boolean。

另见

Box, InsidePlane, Plane

InsideBox 示例



InsidePlane VI

工具面板

Epson Robots | Motion

描述

返回当前机器人夹具末端是否位于指定平面内。

输入

- Spel Ref In* 来自之前 Spel Ref Out 的 Spel 参考。
- Error In* 来自之前 Spel 节点的错误状态。
- AreaNumber* 1-15 之间表示 15 个待检查 box 的整数编号。

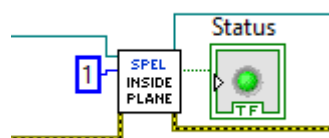
输出

- Spel Ref Out* 对使用的下一个 VI 的 Spel 参考输出。
- Error Out* 向后续 Spel 节点的错误条件输出。
- Status* 机器人夹具末端位于平面内时为 True 的 Boolean。

另见

Box, InsideBox, Plane

InsidePlane 示例



InW VI

工具面板

Epson Robots | Inputs & Outputs

描述

返回指定输入字端口的状态。每个字端口含有 16 个输入位。

输入

- Spel Ref In* 来自之前 Spel Ref Out 的 Spel 参考。
- Error In* 来自之前 Spel 节点的错误状态。
- Port Number* 可选。表示其中一个输入端口的整数。每个端口含有 8 个输入位 (一个字节)。若未指定 *Label*，则使用 *Port Number*。
- Label* 可选。含有输入字节标签的字符串。若指定 *Label*，则将 *Port Number* 忽略。

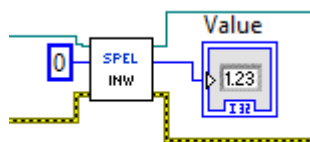
输出

- Spel Ref Out* 对使用的下一个 VI 的 Spel 参考输出。
- Error Out* 向后续 Spel 节点的错误条件输出。
- Value* 0 至 65535 之间表示输入端口的整数值。

另见

In, InBCD, Sw

InW 示例



Inertia VI

工具面板

Epson Robots | Robot Settings

描述

指定当前机器人的负载惯性和偏心距。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>LoadInertia</i>	指定包含夹具末端和部件的夹具末端关节中心附近的总装载惯性的 Double 值，单位为 kgm ² 。
<i>Eccentricity</i>	指定包含夹具末端和部件的夹具末端关节中心附近的偏心距的 Double 值，单位为 mm。

输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

另见

Weight

Inertia 示例



Initialize VI

工具面板

Epson Robots | System

描述

初始化通过 LabVIEW VI 库使用的 Spel 实例。

输入

Server Product Type 可选。指定接口的 EPSON RC+ 产品。*Connection Number* 可选。指定要使用的控制器连接。*Project* 可选。指定要使用的 EPSON RC+ 项目。*ServerInstance* 在指定要使用 EPSON RC+服务器中的实例。(可省略)*ConnectionPassword* 表示连接密码的字符串
若控制器中有连接控制器的密码，并且该密码并未在 EPSON RC+中[设置] - [电脑与控制器连接]对话框中设置，则需要设置控制器连接密码。(可省略)

输出

Spel Ref Out 对使用的下一个 VI 的 Spel 参考输出。*Error Out* 向后续 Spel 节点的错误条件输出。

备注

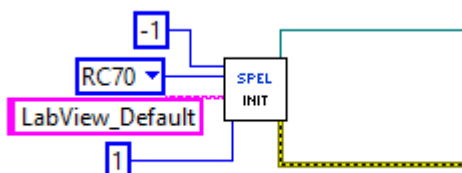
Initialize VI 必须在要使用库的每个实例调用。

Server Product Type 用于指定要使用的 EPSON RC+ 产品。默认为 RC70 (EPSON RC+ 7.0)。未指定 *Connection Number* 时，将使用 EPSON RC+ 7.0 上次使用的连接。未指定 *Project* 时，将使用默认 LabVIEW EPSON RC+ 7.0 项目。该项目必须在通过 *Server Product Type* 指定的 EPSON RC+ 产品中使用。

另见

Shutdown

Initialize 示例



JRange VI

工具面板

Epson Robots | Robot Settings

描述

以脉冲形式定义指定机器人关节的允许工作范围。

输入

Spel Ref In 来自之前 Spel Ref Out 的 Spel 参考。

Error In 来自之前 Spel 节点的错误状态。

JointNumber 1-9 之间表示将指定的 JRange 关节的整数值。

LowerLimitPulses 表示指定关节下限范围的编码器脉冲计数位置的整数值。

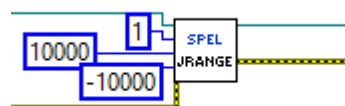
UpperLimitPulses 表示指定关节上限范围的编码器脉冲计数位置的整数值。

输出

Spel Ref Out 对使用的下一个 VI 的 Spel 参考输出。

Error Out 向后续 Spel 节点的错误条件输出。

JRange 示例



JS VI

工具面板

Epson Robots | Motion

描述

Jump Sense 会检测机械臂在完成 JUMP 指令(使用 SENSE 输入)之前是否已停止或机械臂是否已完成 JUMP 移动。JS 返回 Jump Sense 状态。

输入

Spel Ref In 来自之前 Spel Ref Out 的 Spel 参考。

Error In 来自之前 Spel 节点的错误状态。

输出

Spel Ref Out 对使用的下一个 VI 的 Spel 参考输出。

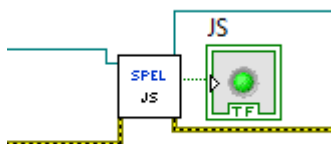
Error Out 向后续 Spel 节点的错误条件输出。

JS 如果在动作期间检测到 SENSE 输入, 则返回 True。否则, 返回 False。

另见

Jump, Sense

JS 示例



JTran VI

工具面板

Epson Robots | Motion

描述

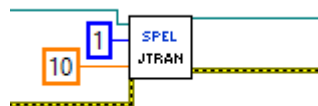
执行相对关节移动。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>JointNumber</i>	待移动的特定关节。
<i>Distance</i>	待移动的距离。对于旋转关节，单位为度；对于线性关节，单位为毫米。

输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

JTran 示例

Jump VI

工具面板

Epson Robots | Motion

描述

使用 PTP 动作将机械臂从当前位置移至指定点，首先垂直上移，然后水平移动，最后再垂直下移到达最终目标点。

输入

- Spel Ref In* 来自之前 Spel Ref Out 的 Spel 参考。
- Error In* 来自之前 Spel 节点的错误状态。
- Point Number* 可选。通过使用当前机器人控制器点内存中之前示教点的点编号来指定目标端点。若指定 *Point Expression*，则将 *Point Number* 忽略。
- Point Expression* 可选。通过使用字符串表达式来指定目标端点。若未指定 *Point Expression*，则使用 *Point Number* 输入。

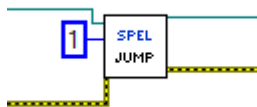
输出

- Spel Ref Out* 对使用的下一个 VI 的 Spel 参考输出。
- Error Out* 向后续 Spel 节点的错误条件输出。

另见

Accel, Arc, Arc3, BGo, BMove, Go, Jump3, Move, Speed, TGo, TMove

Jump 示例



Jump3 VI

工具面板

Epson Robots | Motion

描述

使用两个 CP 动作和一个 PTP 动作的结合通过 3 维闸极动作。机器人移至起始点，然后移至结束点，最后移至目标点。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Depart Point Number</i>	通过使用整数指定起始点。
<i>Depart Point Expr</i>	通过使用字符串表达式指定起始点。若使用该输入，则必须也使用字符串表达式指定结束和目标点。
<i>Appro Point Number</i>	通过使用整数指定结束点。
<i>Appro Point Expr</i>	通过使用字符串表达式指定结束点。若使用该输入，则必须也使用字符串表达式指定起始和目标点。
<i>Dest Point Number</i>	通过使用整数指定目标点。
<i>Dest Point Expr</i>	通过使用字符串表达式指定目标点。若使用该输入，则必须也使用字符串表达式指定起始和结束点。

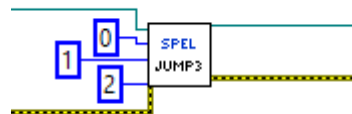
输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

另见

Accel, Arc, Arc3, BGo, BMove, Go, Jump, Move, Speed, TGo, TMove

Jump3 示例



Jump3CP VI

工具面板

Epson Robots | Motion

描述

使用三个 CP 动作的组合通过 3 维闸极动作。机器人移至起始点，然后移至结束点，最后移至目标点。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Depart Point Number</i>	通过使用整数指定起始点。
<i>Depart Point Expr</i>	通过使用字符串表达式指定起始点。若使用该输入，则必须也使用字符串表达式指定结束和目标点。
<i>Appro Point Number</i>	通过使用整数指定结束点。
<i>Appro Point Expr</i>	通过使用字符串表达式指定结束点。若使用该输入，则必须也使用字符串表达式指定起始和目标点。
<i>Dest Point Number</i>	通过使用整数指定目标点。
<i>Dest Point Expr</i>	通过使用字符串表达式指定目标点。若使用该输入，则必须也使用字符串表达式指定起始和结束点。

输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

另见

Accel, Arc, Arc3, BGo, BMove, Go, Jump, Jump3, Move, Speed, TGo, TMove

Jump3CP 示例



LimZ VI

工具面板

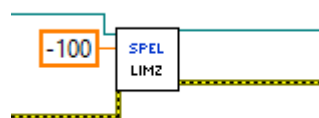
Epson Robots | Motion

描述

设置 JUMP 命令的 Z 轴高度默认值。

输入*Spel Ref In* 来自之前 Spel Ref Out 的 Spel 参考。*Error In* 来自之前 Spel 节点的错误状态。*Z Limit* Z 轴可移动范围内的坐标值。**输出***Spel Ref Out* 对使用的下一个 VI 的 Spel 参考输出。*Error Out* 向后续 Spel 节点的错误条件输出。**另见**

Jump, Jump3

LimZ 示例

LoadPoints VI

工具面板

Epson Robots | Points

描述

将 SPEL+ 点文件载入当前机器人的控制器点内存。

输入

- Spel Ref In* 来自之前 Spel Ref Out 的 Spel 参考。
- Error In* 来自之前 Spel 节点的错误状态。
- File Name* 当前 Spel 项目中或之前通过 SavePoints VI 保存的有效点文件。

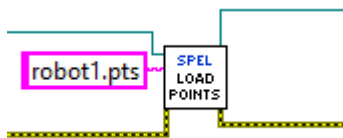
输出

- Spel Ref Out* 对使用的下一个 VI 的 Spel 参考输出。
- Error Out* 向后续 Spel 节点的错误条件输出。

另见

GetPoint, Robot, SavePoints, SetPoint

LoadPoints 示例



MemIn VI

工具面板

Epson Robots | Inputs & Outputs

描述

返回指定内存 I/O 字节端口的状态。每个端口含有 8 个内存 I/O 位。

输入

- Spel Ref In* 来自之前 Spel Ref Out 的 Spel 参考。
- Error In* 来自之前 Spel 节点的错误状态。
- Port Number* 可选。表示其中一个输入端口的整数。每个端口含有 8 个输入位 (一个字节)。若未指定 *Label*，则使用 *Port Number*。
- Label* 可选。含有输入字节标签的字符串。若指定 *Label*，则将 *Port Number* 忽略。

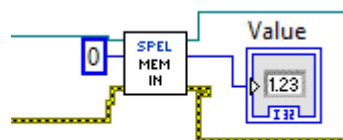
输出

- Spel Ref Out* 对使用的下一个 VI 的 Spel 参考输出。
- Error Out* 向后续 Spel 节点的错误条件输出。
- Value* 0 至 255 之间表示端口状态的整数。

另见

MemInW, MemOut, MemOutW

MemIn 示例



MemInW VI

工具面板

Epson Robots | Inputs & Outputs

描述

返回指定内存 I/O 字端口的状态。每个字端口含有 16 个内存 I/O 位。

输入

- Spel Ref In* 来自之前 Spel Ref Out 的 Spel 参考。
- Error In* 来自之前 Spel 节点的错误状态。
- Port Number* 可选。表示其中一个输入端口的整数。每个字端口含有 16 个输入位。若未指定 *Label*，则使用 *Port Number*。
- Label* 可选。含有输入字节标签的字符串。若指定 *Label*，则将 *Port Number* 忽略。

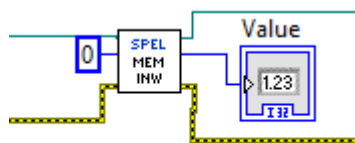
输出

- Spel Ref Out* 对使用的下一个 VI 的 Spel 参考输出。
- Error Out* 向后续 Spel 节点的错误条件输出。
- Value* 0 至 255 之间表示端口状态的整数。

另见

MemIn, MemOut, MemOutW

MemInW 示例



MemOut VI

工具面板

Epson Robots | Inputs & Outputs

描述

根据用户指定的 8 位值同时设置 8 个内存 I/O 位。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Port Number</i>	可选。表示其中一个输入端口的整数。每个端口含有 8 个输入位 (一个字节)。若未指定 <i>Label</i> , 则使用 <i>Port Number</i> 。
<i>Label</i>	可选。含有输入字节标签的字符串。若指定 <i>Label</i> , 则将 <i>Port Number</i> 忽略。
<i>Value</i>	含有指定字节输出模式的整数。有效值范围为 0-255。

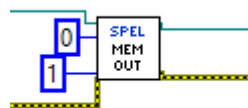
输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

另见

MemIn, MemInW, MemOutW

MemOut 示例



MemOff VI

工具面板

Epson Robots | Inputs & Outputs

描述

关闭内存 I/O 的指定位。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Bit Number</i>	可选。表示其中一个内存 I/O 位的整数。若未指定 <i>Label</i> ，则使用 <i>Bit Number</i> 。
<i>Label</i>	可选。含有输入位标签的字符串。若指定 <i>Label</i> ，则将 <i>Bit Number</i> 忽略。

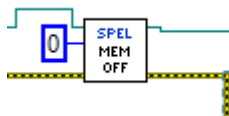
输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

另见

MemOn, MemOut, MemOutW

MemOff 示例



MemOn VI

工具面板

Epson Robots | Inputs & Outputs

描述

打开内存 I/O 的指定位。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Bit Number</i>	可选。表示其中一个内存 I/O 位的整数。若未指定 <i>Label</i> ，则使用 <i>Bit Number</i> 。
<i>Label</i>	可选。含有输入位标签的字符串。若指定 <i>Label</i> ，则将 <i>Bit Number</i> 忽略。

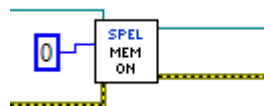
输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

另见

MemOff, MemOut, MemOutW

MemOn 示例



MemOut VI

工具面板

Epson Robots | Inputs & Outputs

描述

根据用户指定的 8 位值同时设置 8 个内存 I/O 位。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Port Number</i>	可选。其中一个内存 I/O 端口的整数表达式。每个端口含有 8 个内存 I/O 位(一个字节)。若未指定 <i>Label</i> ，则使用 <i>Port Number</i> 。
<i>Label</i>	可选。含有内存 I/O 字节标签的字符串。若指定 <i>Label</i> ，则将 <i>Port Number</i> 忽略。
<i>Value</i>	含有指定字节输出模式的整数。有效值范围为 0-255。

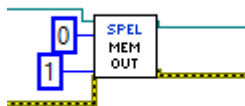
输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

另见

MemOn, MemOff, MemOutW

MemOut 示例



MemOutW VI

工具面板

Epson Robots | Inputs & Outputs

描述

根据用户指定的 16 位值同时设置 16 个内存 I/O 位。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Port Number</i>	可选。表示其中一个内存 I/O 端口的整数。每个字端口含有 16 个输入位。若未指定 <i>Label</i> ，则使用 <i>Port Number</i> 。
<i>Label</i>	可选。含有内存 I/O 字节标签的字符串。若指定 <i>Label</i> ，则将 <i>Port Number</i> 忽略。
<i>Value</i>	含有指定字输出模式的整数。有效值范围为 0-65535。

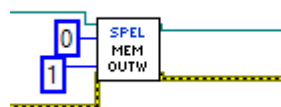
输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

另见

MemOn, MemOff, MemOut

MemOutW 示例



MemSw VI

工具面板

Epson Robots | Inputs & Outputs

描述

返回指定内存 I/O 位的状态。

输入

- Spel Ref In* 来自之前 Spel Ref Out 的 Spel 参考。
- Error In* 来自之前 Spel 节点的错误状态。
- Bit Number* 可选。表示其中一个内存 I/O 位的整数。若未指定 *Label*，则使用 *Bit Number*。
- Label* 可选。含有内存 I/O 位标签的字符串。若指定 *Label*，则将 *Bit Number* 忽略。

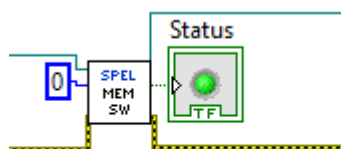
输出

- Spel Ref Out* 对使用的下一个 VI 的 Spel 参考输出。
- Error Out* 向后续 Spel 节点的错误条件输出。
- Value* 内存 I/O 位打开时为 True 的 Boolean。

另见

MemIn, MemInW

MemSW 示例



MotorOff VI

工具面板

Epson Robots | Robot Settings

描述

关闭当前机器人的电机。

输入*Spel Ref In* 来自之前 Spel Ref Out 的 Spel 参考。*Error In* 来自之前 Spel 节点的错误状态。**输出***Spel Ref Out* 对使用的下一个 VI 的 Spel 参考输出。*Error Out* 向后续 Spel 节点的错误条件输出。**另见**

MotorOn, PowerHigh, PowerLow, Robot

MotorOff 示例

MotorOn VI

工具面板

Epson Robots | Robot Settings

描述

打开当前机器人的电机。

输入

Spel Ref In 来自之前 Spel Ref Out 的 Spel 参考。

Error In 来自之前 Spel 节点的错误状态。

输出

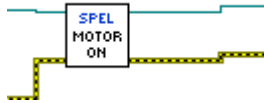
Spel Ref Out 对使用的下一个 VI 的 Spel 参考输出。

Error Out 向后续 Spel 节点的错误条件输出。

另见

MotorOff, PowerHigh, PowerLow, Robot

MotorOn 示例



Move VI

工具面板

Epson Robots | Motion

描述

使用线性内插(即直线移动)将机械臂从当前位置移至指定点。

输入

- Spel Ref In* 来自之前 Spel Ref Out 的 Spel 参考。
- Error In* 来自之前 Spel 节点的错误状态。
- Point Number* 可选。通过使用当前机器人控制器点内存中之前示教点的点编号来指定目标端点。若指定 *Point Expression*, 则将 *Point Number* 忽略。
- Point Expression* 可选。通过使用字符串表达式来指定目标端点。若未指定 *Point Expression*, 则使用 *Point Number* 输入。

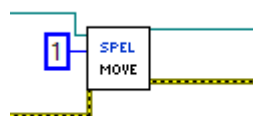
输出

- Spel Ref Out* 对使用的下一个 VI 的 Spel 参考输出。
- Error Out* 向后续 Spel 节点的错误条件输出。

另见

AccelS, Arc, Arc3, BGo, BMove, Go, Jump, Jump3, SpeedS, TGo, TMove

Move 示例



工具面板

Epson Robots | Inputs & Outputs

描述

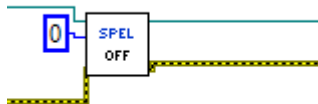
关闭指定输出位。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Bit Number</i>	可选。表示其中一个输出位的整数。若未指定 <i>Label</i> ，则使用 <i>Bit Number</i> 。
<i>Label</i>	可选。含有输入位标签的字符串。若指定 <i>Label</i> ，则将 <i>Bit Number</i> 忽略。

输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

Off 示例

工具面板

Epson Robots | Inputs & Outputs

描述

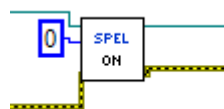
打开指定输出位。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Bit Number</i>	可选。表示其中一个输出位的整数。若未指定 <i>Label</i> ，则使用 <i>Bit Number</i> 。
<i>Label</i>	可选。含有输入位标签的字符串。若指定 <i>Label</i> ，则将 <i>Bit Number</i> 忽略。

输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

On 示例

OPort VI

工具面板

Epson Robots | Inputs & Outputs

描述

返回指定输出位的状态。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Bit Number</i>	可选。表示其中一个输出位的整数。若未指定 <i>Label</i> ，则使用 <i>Bit Number</i> 。
<i>Label</i>	可选。含有输出位标签的字符串。若指定 <i>Label</i> ，则将 <i>Bit Number</i> 忽略。

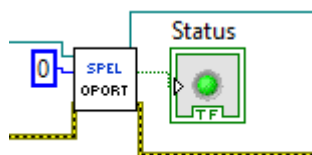
输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。
<i>Value</i>	输出位打开时为 True 的 Boolean。

另见

In, InW, On, Off, Out, Sw

Oport 示例



OprMode VI

工具面板

Epson Robots | System

描述

设置 EPSON RC+ 7.0 的操作模式。

输入

Spel Ref In 来自之前 Spel Ref Out 的 Spel 参考。*Error In* 来自之前 Spel 节点的错误状态。*Operation Mode* 相关 EPSON RC+ 7.0 服务器进程的操作模式。

模式	ID	描述
Auto	1	EPSON RC+ 7.0 处于 auto 模式。
Program	2	EPSON RC+ 7.0 处于 program 模式。

输出

Spel Ref Out 对使用的下一个 VI 的 Spel 参考输出。*Error Out* 向后续 Spel 节点的错误条件输出。

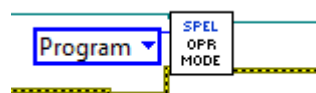
备注

当 *Operation Mode* 设至 Program 时，相关服务器进程的 EPSON RC+ 7.0 GUI 会打开且控制器操作模式会设至 Program。如果用户关闭 RC+ GUI，*Operation Mode* 会设至 Auto。若 *Operation Mode* 设为 Auto，RC+ GUI 也关闭。

另见

GetOprMode

OprMode 示例



Pause VI

工具面板

Epson Robots | Tasks

描述

暂停控制器中所有正常的 SPEL+ 任务。如果机器人正在移动，则其会自动减速直至停止。

输入

Spel Ref In 来自之前 Spel Ref Out 的 Spel 参考。

Error In 来自之前 Spel 节点的错误状态。

输出

Spel Ref Out 对使用的下一个 VI 的 Spel 参考输出。

Error Out 向后续 Spel 节点的错误条件输出。

另见

Continue, Stop

Pause 示例



Plane VI

工具面板

Epson Robots | Robot Settings

描述

定义平面。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Plane Number</i>	1-15 之间表示 15 个待定义平面的整数表达式。
<i>X</i>	平面坐标系原点 X 坐标。
<i>Y</i>	平面坐标系原点 Y 坐标。
<i>Z</i>	平面坐标系原点 Z 坐标。
<i>U</i>	平面坐标系绕着 Z 轴旋转。
<i>V</i>	平面坐标系绕着 Y 轴旋转。
<i>W</i>	平面坐标系绕着 X 轴旋转。

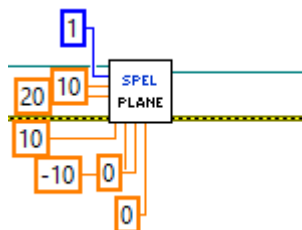
输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

另见

Box, InsideBox, InsidePlane

Plane 示例



PowerHigh VI

工具面板

Epson Robots | Robot Settings

描述

设置当前机器人的电机电量为 high。

输入

Spel Ref In 来自之前 Spel Ref Out 的 Spel 参考。

Error In 来自之前 Spel 节点的错误状态。

输出

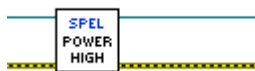
Spel Ref Out 对使用的下一个 VI 的 Spel 参考输出。

Error Out 向后续 Spel 节点的错误条件输出。

另见

MotorOff, MotorOn, PowerLow, Robot

PowerHigh 示例



PowerLow VI

工具面板

Epson Robots | Robot Settings

描述

设置当前机器人的电机电量为 low。

输入

Spel Ref In 来自之前 Spel Ref Out 的 Spel 参考。*Error In* 来自之前 Spel 节点的错误状态。

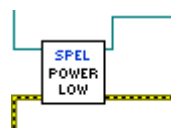
输出

Spel Ref Out 对使用的下一个 VI 的 Spel 参考输出。*Error Out* 向后续 Spel 节点的错误条件输出。

另见

MotorOff, MotorOn, PowerHigh, Robot

PowerLow 示例



Quit VI

工具面板

Epson Robots | Tasks

描述

终止指定任务的执行。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Task Number</i>	可选。待终止任务的任务号。任务号范围为 1 至 32。若指定 Task Name, 则将 Task Number 忽略。
<i>Task Name</i>	可选。指定待终止任务的名称。若未指定 Task Name, 则使用 Task Number 输入。

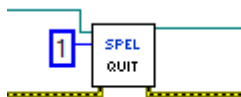
输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

另见

Halt, Resume

Quit 示例



Reset VI

工具面板

Epson Robots | System

描述

将控制器设为初始化状态。

输入

Spel Ref In 来自之前 Spel Ref Out 的 Spel 参考。*Error In* 来自之前 Spel 节点的错误状态。

输出

Spel Ref Out 对使用的下一个 VI 的 Spel 参考输出。*Error Out* 向后续 Spel 节点的错误条件输出。

Reset 示例



Resume VI

工具面板

Epson Robots | Tasks

描述

恢复被 Halt VI 暂停的任务。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Task Number</i>	可选。待恢复任务的任务号。任务号范围为 1 至 32。若指定 Task Name, 则将 Task Number 忽略。
<i>Task Name</i>	可选。指定待恢复任务的名称。若未指定 Task Name, 则使用 Task Number 输入。

输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

另见

Halt, Quit

Resume 示例



Robot VI

工具面板

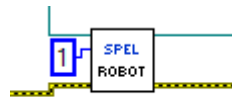
Epson Robots | Robot Settings

描述

选择当前的机器人。

输入*Spel Ref In* 来自之前 Spel Ref Out 的 Spel 参考。*Error In* 来自之前 Spel 节点的错误状态。*Robot Number* 整数范围为 1-16。**输出***Spel Ref Out* 对使用的下一个 VI 的 Spel 参考输出。*Error Out* 向后续 Spel 节点的错误条件输出。**另见**

GetRobot, MotorOff, MotorOn, PowerHigh, PowerLow

Robot 示例

RunDialog VI

工具面板

Epson Robots | GUI

描述

运行 EPSON RC+ 7.0 对话框。

输入

Spel Ref In 来自之前 Spel Ref Out 的 Spel 参考。

Error In 来自之前 Spel 节点的错误状态。

Dialog ID 待运行 EPSON RC+ 7.0 对话框的 ID。

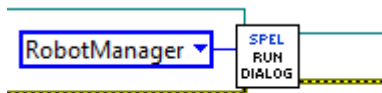
对话框名称	ID	描述
RobotManager	1	工具 机器人管理器对话框 ID
ControllerTools	2	工具 控制器对话框 ID
VisionGuide	3	工具 Vision Guide 对话框 ID

输出

Spel Ref Out 对使用的下一个 VI 的 Spel 参考输出。

Error Out 向后续 Spel 节点的错误条件输出。

RunDialog 示例



SafetyOn VI

工具面板

Epson Robots | System

描述

返回安全门打开的状态。

输入*Spel Ref In* 来自上一个 Spel Ref Out 的 Spel 参考*Error In* 来自上一个 Spel 节点的错误状态**输出***Spel Ref Out* 对使用的下一个 VI 的 Spel 参考输出*Error Out* 向后续 Spel 节点的错误条件输出*SafetyOn* 安全门打开时为 True，否则为 False**另见**

EStopOn

SafetyOn 示例

SavePoints VI

工具面板

Epson Robots | Points

描述

将当前机器人的点保存到文件中。

输入

- Spel Ref In* 来自之前 Spel Ref Out 的 Spel 参考。
- Error In* 来自之前 Spel 节点的错误状态。
- File Name* 当前 Spel 项目中的点文件或要保存在控制器中的新文件名称。

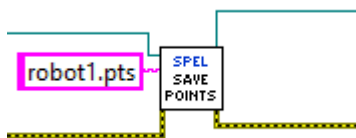
输出

- Spel Ref Out* 对使用的下一个 VI 的 Spel 参考输出。
- Error Out* 向后续 Spel 节点的错误条件输出。

另见

GetPoint, LoadPoints, Robot, SetPoint

SavePoints 示例



Sense VI

工具面板

Epson Robots | Motion

描述

指定输入条件，若此条件满足，则会在目标位置上方停止机器人，进而完成正在进行的 Jump。

输入

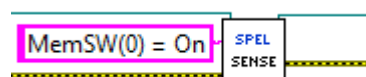
<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Condition</i>	通过使用字符串表达式指定 I/O 条件。有关详细信息，请参阅 SPEL+ 语言参考手册中的 Sense 语句。

输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

另见

JS, Jump

Sense 示例

SetPoint VI

工具面板

Epson Robots | Points

描述

设置当前机器人点的坐标数据。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Point Number</i>	可选。通过使用当前机器人控制器点内存中之前示教点的点编号来指定目标端点。若指定 <i>Point Name</i> ，则将 <i>Point Number</i> 忽略。
<i>Point Name</i>	可选。通过使用点文件名称的字符串表达式指定点。若未指定 <i>Point Name</i> ，则使用 <i>Point Number</i> 输入。
<i>X - W</i>	指定点的 X、Y、Z、U、V、W 坐标。

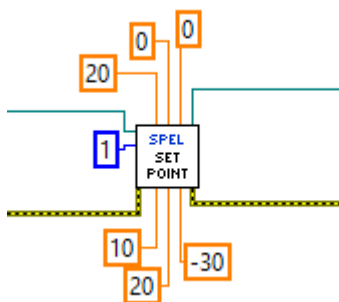
输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

另见

GetPoint, LoadPoints, Robot, SavePoints

SetPoint 示例



SetVar VI

工具面板

Epson Robots | Variables

描述

设置控制器中 SPEL+ 全局保留变量的值。

输入

Spel Ref In 来自之前 Spel Ref Out 的 Spel 参考。

Error In 来自之前 Spel 节点的错误状态。

Var Name SPEL+ 全局保留变量的名称。

Value 含有值的变量。

输出

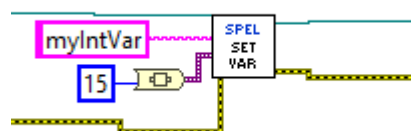
Spel Ref Out 对使用的下一个 VI 的 Spel 参考输出。

Error Out 向后续 Spel 节点的错误条件输出。

另见

GetVar

SetVar 示例



SFree VI

工具面板

Epson Robots | Robot Settings

描述

释放伺服控制的指定机器人轴。

输入

- Spel Ref In* 来自之前 Spel Ref Out 的 Spel 参考。
- Error In* 来自之前 Spel 节点的错误状态。
- Axes* 可选。指定要释放轴的整数数组。如果忽略，则会释放所有轴。

输出

- Spel Ref Out* 对使用的下一个 VI 的 Spel 参考输出。
- Error Out* 向后续 Spel 节点的错误条件输出。

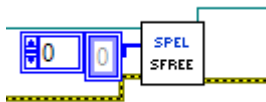
备注

如果忽略 Axes，则会释放所有轴。

另见

MotorOff, MotorOn, SLock

SFree 示例



ShowWindow VI

工具面板

Epson Robots | GUI

描述

显示 EPSON RC+ 7.0 窗口。

输入

Spel Ref In 来自之前 Spel Ref Out 的 Spel 参考。*Error In* 来自之前 Spel 节点的错误状态。*Window ID* 待显示 EPSON RC+ 7.0 窗口的 ID。

窗口名称	ID	描述
IOMonitor	1	I/O 监视器窗口 ID
TaskManager	2	任务管理器窗口 ID
ForceMonitor	3	力监视器窗口 ID
Simulator	4	仿真器窗口 ID

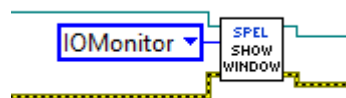
输出

Spel Ref Out 对使用的下一个 VI 的 Spel 参考输出。*Error Out* 向后续 Spel 节点的错误条件输出。

另见

HideWindow, RunDialog

ShowWindow 示例



Shutdown VI

工具面板

Epson Robots | System

描述

关闭调用 Initialize VI 时启动的 EPSON RC+ 7.0 服务器进程。

输入

Spel Ref In 来自之前 Spel Ref Out 的 Spel 参考。

Error In 来自之前 Spel 节点的错误状态。

输出

Error Out 向后续 Spel 节点的错误条件输出。

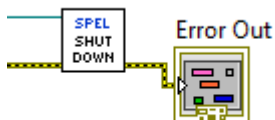
备注

Shutdown VI 必须在库的每个实例调用。这将关闭相关 EPSON RC+ 7.0 服务器进程。

另见

Initialize

Shutdown 示例



SLock VI

工具面板

Epson Robots | Robot Settings

描述

将指定机器人轴返回指定伺服控制。

输入

- Spel Ref In* 来自之前 Spel Ref Out 的 Spel 参考。
- Error In* 来自之前 Spel 节点的错误状态。
- Axes* 可选。指定要锁定轴的整数数组。如果忽略，则会锁定所有轴。

输出

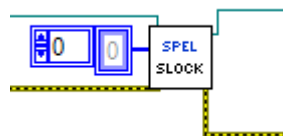
- Spel Ref Out* 对使用的下一个 VI 的 Spel 参考输出。
- Error Out* 向后续 Spel 节点的错误条件输出。

备注

如果忽略 Axes，则会锁定所有轴。

另见

MotorOff, MotorOn, SFree

SLock 示例

Speed VI

工具面板

Epson Robots | Robot Settings

描述

指定用于 PTP 指令 Go、Jump 和 Pulse 的机械臂速度。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>PointToPoint Speed</i>	指定用于 PTP 指令 Go、Jump 和 Pulse 的机械臂速度。
<i>Depart Speed</i>	1-100 之间表示 Jump 指令 Z 轴向上动作速度的整数。
<i>Appro Speed</i>	1-100 之间表示 Jump 指令 Z 轴向下动作速度的整数。

输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

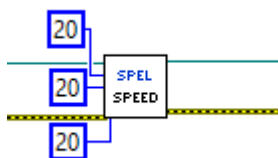
备注

使用 Speed 设置当前机器人的 PTP 速度。所有数值可为 1 至 100%。若指定 *Depart Speed*，则也需指定 *Appro Speed*。

另见

Accel, AccelS, SpeedS

Speed 示例



SpeedS VI

工具面板

Epson Robots | Robot Settings

描述

指定用于连续路径指令 Jump3CP、Move、Arc 和 CVMove 的机械臂速度。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Linear Speed</i>	指定用于连续路径指令 Jump3CP、Move、Arc 和 CVMove 的机械臂速度。
<i>Depart Speed</i>	1-5000 之间表示 Jump3CP 指令 Z 轴向上动作速度的双值。
<i>Appro Speed</i>	1-5000 之间表示 Jump3CP 指令 Z 轴向下动作速度的双值。

输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

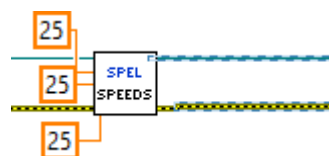
备注

使用 Speed 设置当前机器人的线性速度，单位为 mm/sec。若指定 *Depart Speed*，则也需指定 *Appro Speed*。

另见

Accel, AccelS, Speed

SpeedS 示例



Start VI

工具面板

Epson Robots | Tasks

描述

开始要在控制器中运行的程序。

输入

Spel Ref In 来自之前 Spel Ref Out 的 Spel 参考。*Error In* 来自之前 Spel 节点的错误状态。*ProgramNumber* 启动程序编号，对应于下表所示 SPEL+ 中的 8 个 main 函数。范围为 0-7。

程序编号	SPEL+ 函数名称
0	main
1	main1
2	main2
3	main3
4	main4
5	main5
6	main6
7	main7

输出

Spel Ref Out 对使用的下一个 VI 的 Spel 参考输出。*Error Out* 向后续 Spel 节点的错误条件输出。

备注

执行 **Start** 时，控制将立即返回至调用 VI。但不能启动正在运行的程序。请注意，Start 会导致控制器中的全局变量被清除并加载默认的机器人点。

另见

Continue, Pause, Stop, Xqt

Start 示例



Stop VI

工具面板

Epson Robots | Tasks

描述

停止控制器中正在运行的所有正常 SPEL+ 任务，以及可选停止所有后台任务。

输入

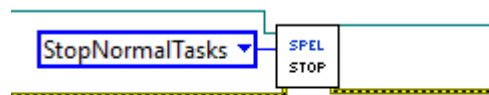
<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Stop Type</i>	可选。指定 StopNormalTasks(默认值)或 StopAllTasks(也停止后台任务)。

输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

另见

Continue, Pause, Start, Xqt

Stop 示例

工具面板

Epson Robots | Inputs & Outputs

描述

返回指定输入位的状态。

输入

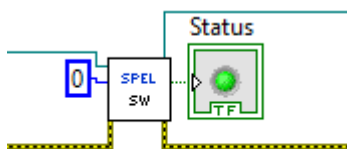
<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Bit Number</i>	可选。表示其中一个输入位的整数。若未指定 <i>Label</i> ，则使用 <i>Bit Number</i> 。
<i>Label</i>	可选。含有输入位标签的字符串。若指定 <i>Label</i> ，则将 <i>Bit Number</i> 忽略。

输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。
<i>Value</i>	输出位打开时为 True 的 Boolean。

另见

In, InW, On, Off, OPort, Out

SW 示例

TargetOK VI

工具面板

Epson Robots | Motion

描述

返回表示是否能够通过 PTP(点到点)动作从当前位置移至目标位置的状态。

输入

- Spel Ref In* 来自之前 Spel Ref Out 的 Spel 参考。
- Error In* 来自之前 Spel 节点的错误状态。
- Point Number* 可选。通过使用当前机器人控制器点内存中之前示教点的点编号来指定目标端点。若指定 *Point Expression*, 则将 *Point Number* 忽略。
- Point Expression* 可选。通过使用字符串表达式来指定目标端点。若未指定 *Point Expression*, 则使用 *Point Number* 输入。

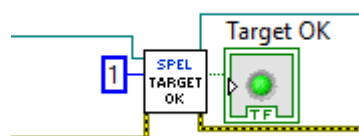
输出

- Spel Ref Out* 对使用的下一个 VI 的 Spel 参考输出。
- Target OK* 机器人可以移至目标位置。
- At Home* Boolean 指示当前机器人是否位于起始点位置。

另见

BGo, Go, Jump, TGo

TargetOK 示例



TGo VI

工具面板

Epson Robots | Motion

描述

在当前工具坐标系中执行 PTP 的相对运动。

输入

- Spel Ref In* 来自之前 Spel Ref Out 的 Spel 参考。
- Error In* 来自之前 Spel 节点的错误状态。
- Point Number* 可选。通过使用当前机器人控制器点内存中之前示教点的点编号来指定目标端点。若指定 *Point Expression*，则将 *Point Number* 忽略。
- Point Expression* 可选。通过使用字符串表达式来指定目标端点。若未指定 *Point Expression*，则使用 *Point Number* 输入。

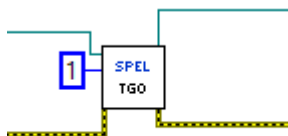
输出

- Spel Ref Out* 对使用的下一个 VI 的 Spel 参考输出。
- Error Out* 向后续 Spel 节点的错误条件输出。

另见

Accel, Arc, Arc3, BGo, BMove, Go, Jump, Jump3, Move, Speed, TMove

TGo 示例



Till VI

工具面板

Epson Robots | Motion

描述

指定事件条件，如果此条件满足，则会在中间位置减速并停止机器人，以完成正在进行的动作命令(Jump、Go、Move 等)。

输入

Spel Ref In 来自之前 Spel Ref Out 的 Spel 参考。

Error In 来自之前 Spel 节点的错误状态。

Condition 通过使用字符串表达式指定 I/O 条件。有关详细信息，请参阅 SPEL+ 语言参考手册中的 Sense 语句。

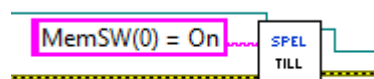
输出

Spel Ref Out 对使用的下一个 VI 的 Spel 参考输出。

Error Out 向后续 Spel 节点的错误条件输出。

另见

Accel, Arc, Arc3, BGo, BMove, Jump, Jump3, Move, Speed, TGo, TillOn, TMove

Till 示例

TillOn VI

工具面板

Epson Robots | Motion

描述

如果在最后的 Go/Jump/Move 语句期间因 till 条件而发生停止，则返回 True。

输入

Spel Ref In 来自之前 Spel Ref Out 的 Spel 参考。

Error In 来自之前 Spel 节点的错误状态。

输出

Spel Ref Out 对使用的下一个 VI 的 Spel 参考输出。

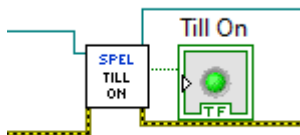
Error Out 向后续 Spel 节点的错误条件输出。

Till On 如果在动作期间检测到 Till 条件，则返回 True。否则，返回 False。

另见

Accel, Arc, Arc3, BGo, BMove, Jump, Jump3, Move, Speed, TGo, Till, TMove

TillOn 示例



TLSet VI

工具面板

Epson Robots | Robot Settings

描述

定义 ECP(外部控制点)。

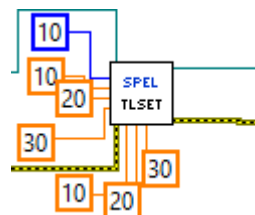
输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>ToolNumber</i>	1-15 之间表示 15 个待定义工具的整数表达式。 (Tool 0 为默认工具, 无法更改。)
<i>X</i>	工具坐标系原点 X 坐标。
<i>Y</i>	工具坐标系原点 Y 坐标。
<i>Z</i>	工具坐标系原点 Z 坐标。
<i>U</i>	工具坐标系绕着 Z 轴旋转。
<i>V</i>	工具坐标系绕着 Y 轴旋转。
<i>W</i>	工具坐标系绕着 X 轴旋转。

输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

TLSet 示例



TMove VI

工具面板

Epson Robots | Motion

描述

在当前工具坐标系中执行线性内插相对运动。

输入

- Spel Ref In* 来自之前 Spel Ref Out 的 Spel 参考。
- Error In* 来自之前 Spel 节点的错误状态。
- Point Number* 可选。通过使用当前机器人控制器点内存中之前示教点的点编号来指定目标端点。若指定 *Point Expression*，则将 *Point Number* 忽略。
- Point Expression* 可选。通过使用字符串表达式来指定目标端点。若未指定 *Point Expression*，则使用 *Point Number* 输入。

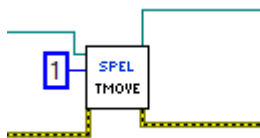
输出

- Spel Ref Out* 对使用的下一个 VI 的 Spel 参考输出。
- Error Out* 向后续 Spel 节点的错误条件输出。

另见

AccelS, Arc, Arc3, BGo, BMove, Go, Jump, Jump3, Move, SpeedS, TGo

TMove 示例



Tool VI

工具面板

Epson Robots | Robot Settings

描述

选择当前机器人的工具。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Tool Number</i>	0-15 之间表示 16 个工具定义中将与随后的动作指令一同使用的定义的整数。

输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

另见

Arm, Armset, GetTool, TLSet

Tool 示例

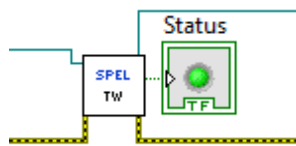
TW VI

工具面板

Epson Robots | Inputs & Outputs

描述

返回 Wait 命令的条件是否成立。

输入*Spel Ref In* 来自之前 Spel Ref Out 的 Spel 参考*Error In* 来自之前 Spel 节点的错误状态**输出***Spel Ref Out* 对使用的下一个 VI 的 Spel 参考输出*Error Out* 向后续 Spel 节点的错误条件输出*Status* 返回 Wait 命令的条件是否成立**TW 示例**

VGetBool VI

工具面板

Epson Robots | Vision

描述

检索返回 Boolean 值的视觉属性或结果。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Sequence</i>	当前项目中视觉序列的名称。
<i>Object</i>	可选。指定序列中的视觉对象名称。检索序列的属性或结果时忽略。
<i>Property Code</i>	属性或结果代码。
<i>Result Index</i>	可选。结果的索引。

输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。
<i>Value</i>	Boolean 值。

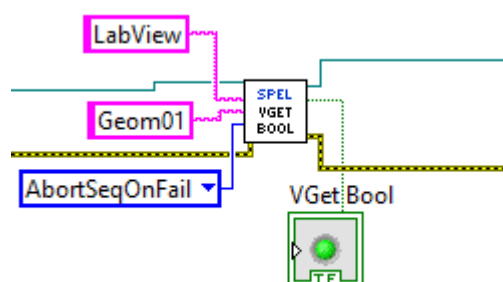
备注

有关 Vision Guide 属性和结果的详细信息，请参阅 Vision Guide 7.0 属性和结果参考手册。

另见

VRun, VGetDbl, VGetInt, VGetStr, VSetBool, VSetDbl, VSetInt, VSetStr

VGetBool 示例



VGetDbI VI

工具面板

Epson Robots | Vision

描述

检索返回 double 值的视觉属性或结果。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Sequence</i>	当前项目中视觉序列的名称。
<i>Object</i>	可选。指定序列中的视觉对象名称。检索序列的属性或结果时忽略。
<i>Property Code</i>	属性或结果代码。
<i>Result Index</i>	可选。结果的索引。

输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。
<i>Value</i>	Double 值。

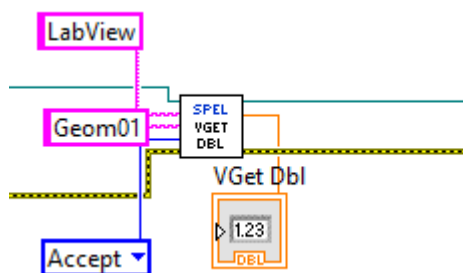
备注

有关 Vision Guide 属性和结果的详细信息，请参阅 Vision Guide 7.0 属性和结果参考手册。

另见

VRun, VGetBool, VGetInt, VGetStr, VSetBool, VSetDbI, VSetInt, VSetStr

VGetDbI 示例



VGetInt VI

工具面板

Epson Robots | Vision

描述

检索返回整数值的视觉属性或结果。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Sequence</i>	当前项目中视觉序列的名称。
<i>Object</i>	可选。指定序列中的视觉对象名称。检索序列的属性或结果时忽略。
<i>Property Code</i>	属性或结果代码。
<i>Result Index</i>	可选。结果的索引。

输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。
<i>Value</i>	整数值。

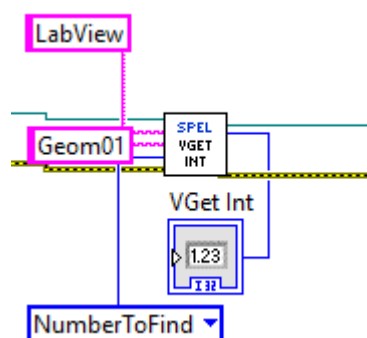
备注

有关 Vision Guide 属性和结果的详细信息，请参阅 Vision Guide 7.0 属性和结果参考手册。

另见

VRun, VGetBool, VGetDbl, VGetStr, VSetBool, VSetDbl, VSetInt, VSetStr

VGetInt 示例



VGetStr VI

工具面板

Epson Robots | Vision

描述

检索返回字符串值的视觉属性或结果。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Sequence</i>	当前项目中视觉序列的名称。
<i>Object</i>	可选。指定序列中的视觉对象名称。检索序列的属性或结果时忽略。
<i>Property Code</i>	属性或结果代码。
<i>Result Index</i>	可选。结果的索引。

输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。
<i>Value</i>	字符串值。

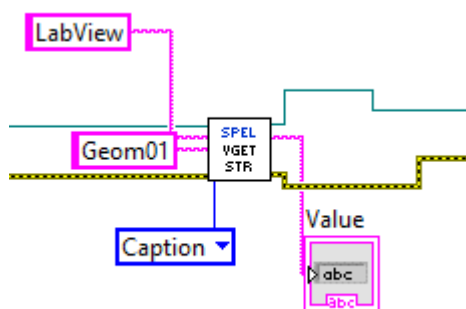
备注

有关 Vision Guide 属性和结果的详细信息，请参阅 Vision Guide 7.0 属性和结果参考手册。

另见

VRUN, VGETBOOL, VGETDBL, VGETINT, VSETBOOL, VSETDBL, VSETINT, VSETSTR

VGetStr 示例



VideoControl VI

工具面板

Epson Robots | Vision

描述

配置 SPEL Video 控件的设置，以显示视觉系统图像。

输入

- Spel Ref In* 来自之前 Spel Ref Out 的 Spel 参考。
- Error In* 来自之前 Spel 节点的错误状态。
- Sequence* 当前项目中视觉序列的名称。
- VideoRef In* 来自 SPEL Video 控制的参考。
- Camera* 设置要显示哪台相机。默认为 0，任何相机都显示。
- Graphics Enabled* 设置是否显示图形。
- Video Enabled* 设置是否显示视频。

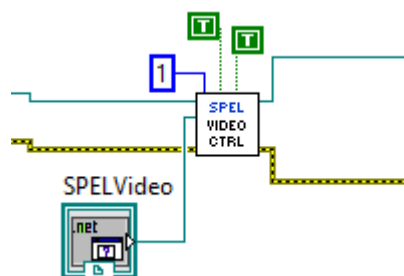
输出

- Spel Ref Out* 对使用的下一个 VI 的 Spel 参考输出。
- Error Out* 向后续 Spel 节点的错误条件输出。

另见

Displaying Video, VGet, VRun

VideoControl 示例



VRun VI

工具面板

Epson Robots | Vision

描述

运行当前项目中的视觉序列。

输入*Spel Ref In* 来自之前 Spel Ref Out 的 Spel 参考。*Error In* 来自之前 Spel 节点的错误状态。*Sequence* 当前项目中视觉序列的名称。**输出***Spel Ref Out* 对使用的下一个 VI 的 Spel 参考输出。*Error Out* 向后续 Spel 节点的错误条件输出。**备注**

有关运行视频序列的信息，请参阅 Vision Guide 7.0 软件手册。

另见

VGetBool, VGetDbl, VGetInt, VGetStr, VSetBool, VSetDbl, VSetInt, VSetStr

VRun 示例

VSetBool VI

工具面板

Epson Robots | Vision

描述

设置数据类型为 Boolean 的视觉属性值。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Sequence</i>	当前项目中视觉序列的名称。
<i>Object</i>	可选。指定序列中的视觉对象名称。设置序列的属性时忽略。
<i>Property Code</i>	属性代码。
<i>Value</i>	属性的新 Boolean 值。

输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

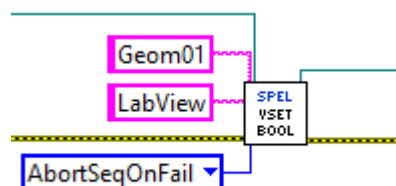
备注

有关 Vision Guide 属性和结果的详细信息，请参阅 Vision Guide 7.0 属性和结果参考手册。

另见

VGetBool, VGetDbl, VGetInt, VGetStr, VRun, VSetDbl, VSetInt, VSetStr

VSetBool 示例



VSetDbI VI

工具面板

Epson Robots | Vision

描述

设置数据类型为 `real` 或 `double` 的视觉属性值。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Sequence</i>	当前项目中视觉序列的名称。
<i>Object</i>	可选。指定序列中的视觉对象名称。设置序列的属性时忽略。
<i>Property Code</i>	属性代码。
<i>Value</i>	属性的新 <code>double</code> 值。

输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

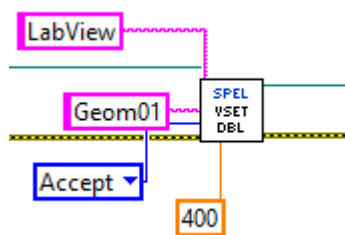
备注

有关 Vision Guide 属性和结果的详细信息，请参阅 Vision Guide 7.0 属性和结果参考手册。

另见

VGetBool, VGetDbI, VGetInt, VGetStr, VRun, VSetBool, VSetInt, VSetStr

VSetDbI 示例



VSetInt VI

工具面板

Epson Robots | Vision

描述

设置数据类型为整数的视觉属性值。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Sequence</i>	当前项目中视觉序列的名称。
<i>Object</i>	可选。指定序列中的视觉对象名称。设置序列的属性时忽略。
<i>Property Code</i>	属性代码。
<i>Value</i>	属性的新整数值。

输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

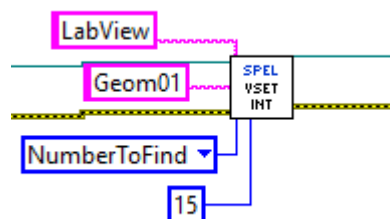
备注

有关 Vision Guide 属性和结果的详细信息，请参阅 Vision Guide 7.0 属性和结果参考手册。

另见

VGetBool, VGetDbl, VGetInt, VGetStr, VRun, VSetBool, VSetDbl, VSetStr

VSetInt 示例



VSetStr VI

工具面板

Epson Robots | Vision

描述

设置数据类型为字符串的视觉属性值。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Sequence</i>	当前项目中视觉序列的名称。
<i>Object</i>	可选。指定序列中的视觉对象名称。设置序列的属性时忽略。
<i>Property Code</i>	属性代码。
<i>Value</i>	属性的新字符串值。

输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

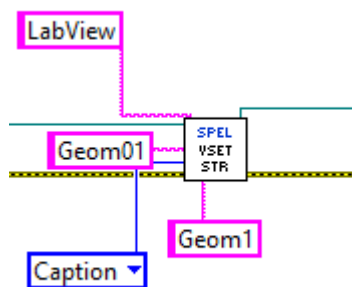
备注

有关 Vision Guide 属性和结果的详细信息，请参阅 Vision Guide 7.0 属性和结果参考手册。

另见

VGetBool, VGetDbl, VGetInt, VGetStr, VRun, VSetBool, VSetDbl, VSetInt

VSetStr 示例



WaitTaskDone VI

工具面板

Epson Robots | Tasks

描述

等待任务完成并返回状态。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Task Number</i>	可选。待暂停任务的任务号。任务号范围为 1 至 32。若指定 Task Name, 则将 Task Number 忽略。
<i>Task Name</i>	可选。指定待暂停任务的名称。若未指定 <i>Task Name</i> , 则使用 <i>Task Number</i> 输入。

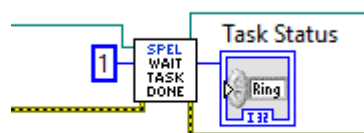
输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。
<i>Task State</i>	表示任务的最终状态(Quit、Aborted、Finished)。

另见

Xqt

WaitTaskDone 示例



Weight VI

工具面板

Epson Robots | Robot Settings

描述

指定当前机器人的 weight 参数。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Payload Weight</i>	待承载的夹具末端重量，单位为 Kg。
<i>Arm Length</i>	第二机械臂的旋转中心至夹具末端的重心之间的距离，单位为 mm。

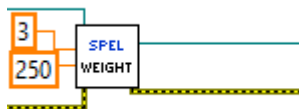
输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

另见

Inertia

Weight 示例



Xqt VI

工具面板

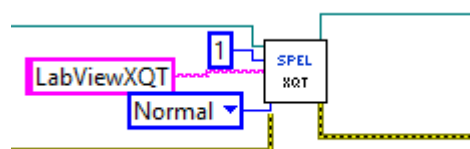
Epson Robots | Tasks

描述

启动一个 SPEL+ 任务。

输入*Spel Ref In* 来自之前 Spel Ref Out 的 Spel 参考。*Error In* 来自之前 Spel 节点的错误状态。*Task Number* 可选。待执行任务的任务号。任务号范围为 1 至 32。如果忽略 *Task Number*，则任务编号会自动配。*Func Name* 指定待执行函数的名称。**输出***Spel Ref Out* 对使用的下一个 VI 的 Spel 参考输出。*Error Out* 向后续 Spel 节点的错误条件输出。**另见**

Halt, Quit, Resume, WaitForTaskDone

XQT 示例

XYLim VI

工具面板

Epson Robots | Robot Settings

描述

设置机器人的允许动作范围限制。

输入

<i>Spel Ref In</i>	来自之前 Spel Ref Out 的 Spel 参考。
<i>Error In</i>	来自之前 Spel 节点的错误状态。
<i>Min X</i>	机器人可行进的最小 X 坐标位置。 (机器人不得移至 X 坐标小于 min X 的位置。)
<i>Max X</i>	机器人可行进的最大 X 坐标位置。 (机器人不得移至 X 坐标大于 max X 的位置。)
<i>Min Y</i>	机器人可行进的最小 Y 坐标位置。 (机器人不得移至 Y 坐标小于 min Y 的位置。)
<i>Max Y</i>	机器人可行进的最大 Y 坐标位置。 (机器人不得移至 Y 坐标大于 max Y 的位置。)
<i>Min Z</i>	机器人可行进的最小 Z 坐标位置。 (机器人不得移至 Z 坐标小于 min Z 的位置。)
<i>Max Z</i>	机器人可行进的最大 Z 坐标位置。 (机器人不得移至 Z 坐标大于 max Z 的位置。)

备注

XYLim 用于定义动作范围限制。很多机器人系统允许用户定义关节限制，而 SPEL+ 语言允许定义关节限制和动作范围限制。实际上，这便允许用户创建其应用的工作范围。(请牢记，也可通过 SPEL 定义关节范围限制。)

通过 XYLim 值确定的动作范围仅适用于动作命令目标位置，但不适用于从起始位置到目标位置的动作路径。因此，机械臂可能会在动作期间移至 XYLim 范围以外。(即，XYLim 范围不影响脉冲。)

若要关闭动作范围限制，在范围限制参数指定 0。

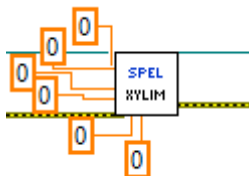
输出

<i>Spel Ref Out</i>	对使用的下一个 VI 的 Spel 参考输出。
<i>Error Out</i>	向后续 Spel 节点的错误条件输出。

另见

Box

XYLim 示例



17. 在 RCNetLib 使用 LabVIEW

17.1 概述

使用 LabVIEW VI 库章节中介绍的 LabVIEW VI 库是使用 RCAPINet.dll 的高级界面。部分用户可能想要直接与 RCAPINet.dll 接口，而不使用高级库。本章包含搭配 RCAPINet.dll 使用 LabVIEW 的信息。将介绍以下主题。

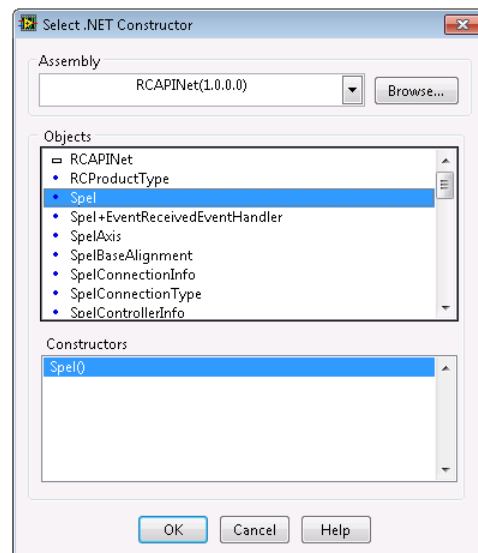
- 初始化
- 在应用中使用 Spel 属性和方法
- 关闭
- 使用对话框和窗口

17.2 初始化

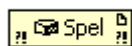
17.2.1 添加 Spel 类的构造函数节点

在您可以从 Spel 类调用方法或使用属性之前，必须使用构造函数节点创建 Spel 类实例。您应在应用中使用一个 Spel 类实例。

在将包含 Spel 类实例的 VI 方框图视图中，从 [RC+ API] - [.NET palette] 添加构造函数节点。将出现 [Select .NET Constructor] 对话框。在 [Assembly] 列表中选择“RCAPINet”并在 [Objects] 列表中选择“Spel”，如下所示。

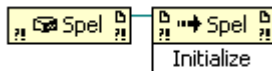


点击<OK> 在方框图中创建 Spel 构造函数节点。



17.2.2 初始化 Spel 类实例

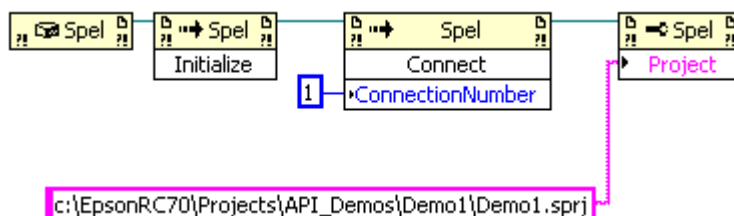
添加用于 Spel 类 Initialize 方法的 Invoke 节点。Initialize 执行时，将在后台作为服务器配置和启动 RC+。



17.2.3 连接至控制器和设置项目

添加用于 Spel 类 Connect 方法的 Invoke 节点。设置用于需使用的控制器连接的 ConnectionNumber 参数。若要查看连接编号，首先启动 EPSON RC+ 7.0，然后选择 [设置]-[PC 至控制器通信]。

添加用于 Spel 类 Project 属性的 Property 节点。将 Project 参数设为所需的项目文件。



17.3 使用 Spel 属性和方法

添加更多节点，以将 Spel 属性和方法用于您的应用。必须将前一节点的参考输出连接至当前节点的参考输入。这样便允许每个属性或方法使用您在上述步骤中创建和初始化的 Spel 类实例。有关可使用的属性和方法信息，请参阅 RCAPINet 参考章节。

17.4 关闭

使用 Spel 类实例完成时，需要调用 Dispose 方法。这样将关闭与 Spel 类实例相关的 EPSON RC+ 7.0 服务器。通常情况下，应在应用结束时调用 Dispose。

如果您的应用在未调用 Dispose 情况下中止，由于 LabVIEW(客户端过程)会继续运行，所以 RC+ 过程也将继续运行。当您再次启动应用时，如果 RC+ 过程正在运行，则过程会重新开始。但是，如果您试图运行 RC+ GUI，则其会询问您是否想要运行 RC+ 的另一实例。在这种情况下，可首先从 Windows 任务管理器终止 RC+ 过程 (erc70.exe)，然后再运行 EPSON RC+ 7.0 GUI。

17.5 使用对话框和窗口

与 .NET 应用共用时，通常会将 .NET 父窗体用作通过 Spel 类实例显示的父对话框和窗口。但是 LabVIEW 不会使用 .NET 窗体，因此，要通过 LabVIEW 显示窗口和对话框，需使用 ParentWindowHandle 属性。将其设至 VI 的窗口句柄。然后，您便可以调用 Windows API FindWindow 方法获取窗口句柄。

使用 ParentWindowHandle 时，必须调用没有父参数的 Spel.ShowWindow。

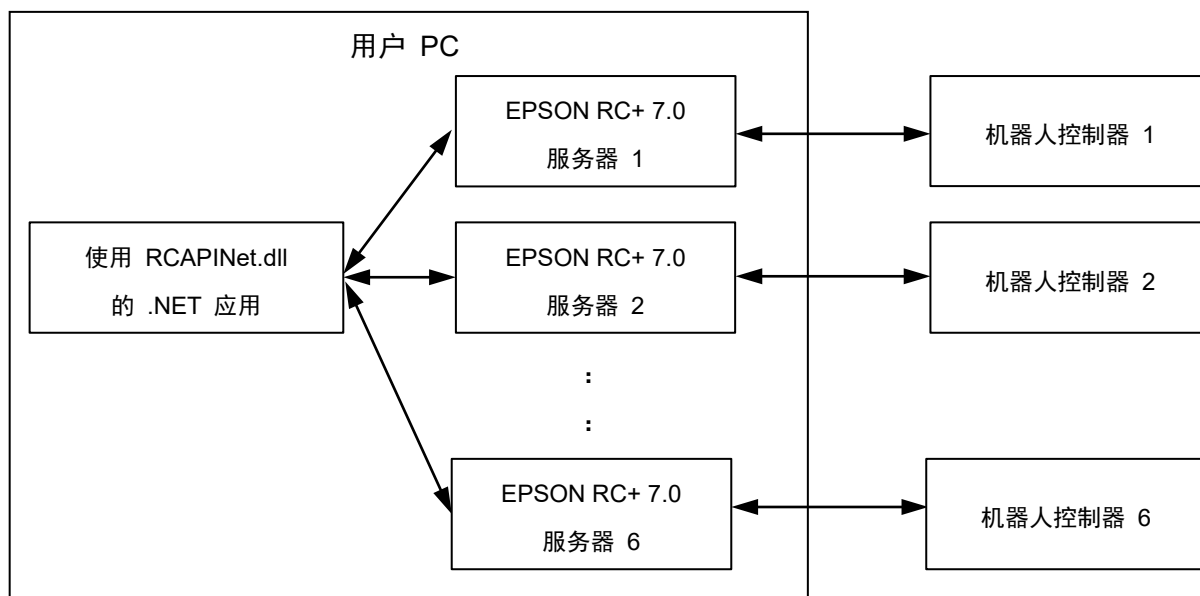
18. 如何从一台 PC 控制多个控制器

18.1 概述

使用 RC+ API 可以从一台 PC 控制最多六个机器人控制器。

若要控制多个控制器，每个控制器都需要进行 RCAPINet Spel 类实例化。

下图所示为使用 RC+ API 控制多个控制器的基本系统配置图。



应用通过为每个控制器准备的服务器(RCAPINet Spel 类)控制多个控制器。

18.1.1 系统要求

建议满足以下要求的 PC。

操作系统	Windows 8.1 Pro 64 位版本 Windows 10 Pro 64 位版本 Windows 11 Pro 64 位版本
CPU	Core i5 或更高版本
内存	2 GB 或更大

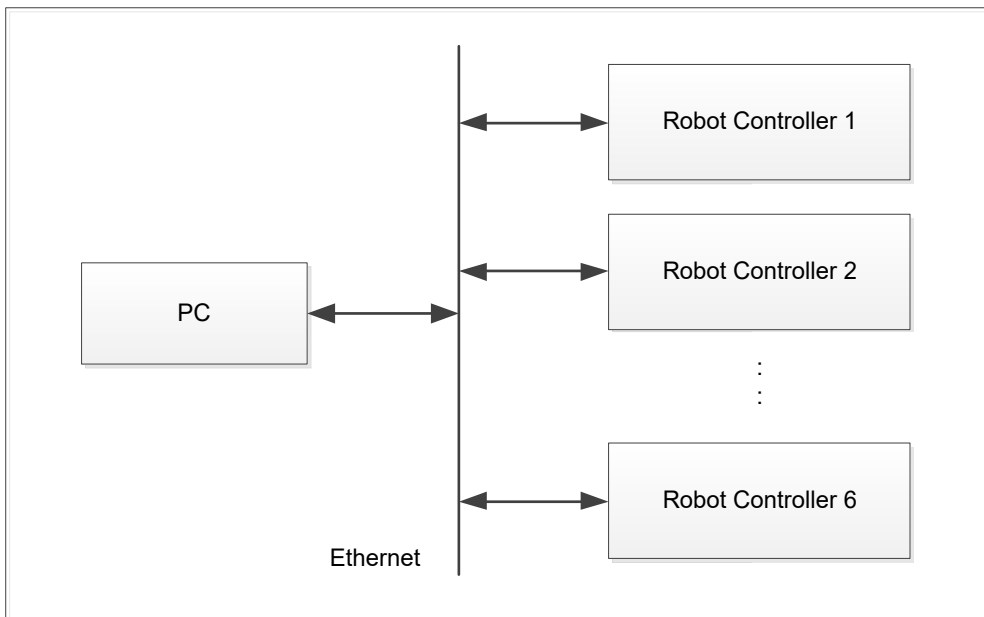


- 如果您使用的电脑低于推荐性能，请在使用前确认是否可以正常操作。使用不满足上述要求的电脑，可能无法稳定的控制。

18.1.2 PC 和控制器连接

第一个控制器的连接类型可以为 USB 或以太网。其余控制器的连接类型必须为以太网。

下图所示为 PC 和多个控制器的基本配置图。



机器人控制器支持连接至 EPSON RC+ 7.0 的控制器。



连接至 EPSON RC+ 7.0 的控制器可以同时连接。



可以选择一个虚拟控制器。



可以通过 USB 通信连接一个控制器。
使用 USB 通信时，仅通过 USB 通信连接一个控制器，并通过以太网连接其他控制器。



注意

- 若 PC 上安装了防病毒软件，运行病毒扫描时，与控制器的通信可能会异常断开。若要运行病毒扫描，请先断开与控制器的通信。

18.2 控制多个控制器的限制事项

控制多个控制器有以下章节中介绍的限制事项。

18.2.1 控制器选件的限制事项

每个控制器控制的以下控制器选件有限制条件。

- PC 视觉
 - 现场总线主站
 - 力觉
- (力觉与力传感器不同。)

上述三项选项中的一项已连接至激活的控制器时，这些选件无法用于其他(第二个或之后)控制器。

18.2.2 仿真器限制

仿真器窗口显示

可以从 .NET 应用使用 EPSON RC+ 7.0 仿真器窗口。
有关详细信息请参阅本手册中的 *10.1 窗口*。

若连接多个控制器时打开每个控制器的仿真器窗口，周期时间可能比不显示仿真器窗口时增加 100 至 200 毫秒。

此外，若在打开仿真器窗口状态下执行程序，CPU 使用率可能会增加至接近 100 %，PC 上可能被施加大负荷。

除调试程序时之外，建议在仿真器窗口关闭状态下使用系统。

碰撞检测

若使用仿真器避免与外围设备的碰撞，应在进行避免碰撞检测的仿真器对象周围设置 15 mm 或更大的边距。

仿真器中的碰撞检测不保证精度。应用到实际系统时，务必设置边距并充分确认操作。

有关各限制事项的详细信息，请参阅 *EPSON RC+ 用户指南* 中的 *8.4 仿真器规格和限制事项*。

18.3 多个控制器连接的样本程序

以下章节介绍使用 .NET 应用将 PC 与控制器 1 和控制器 2 连接的样本程序。



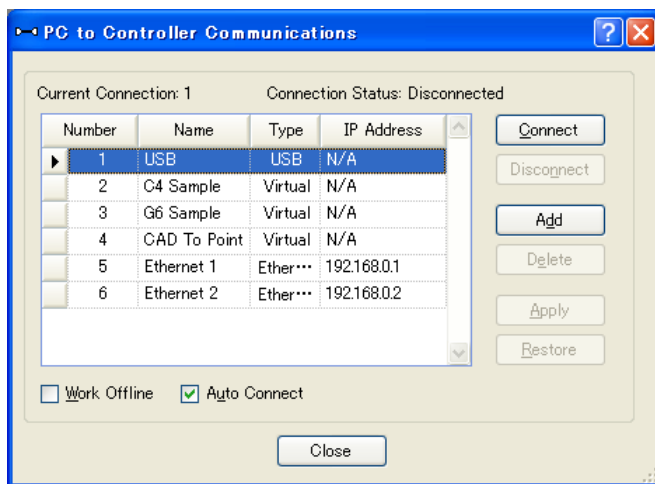
有关可用属性和方法的详细信息，请参阅本手册中的 14. *RCAPINet* 参考。

18.3.1 控制器连接设置

同时连接多个控制器时，使用 Spel 类的 Connect 方法指定连接。

```
m_spel.Connect(1)
```

Connect 方法中的参数表示连接编号。此编号与以下对话框(EPSON RC+ 7.0 菜单-[Setup]-[PC to Controller Communications])的“Number”中显示的相同。若使用 -1 值，这表示使用最近一次的连接。



从 RC+7.5.0 版本开始，可以使用连接名称(如上图所示)连接到 Connect 方法。

18.3.2 项目设置

若要连接多个控制器，使用 Spel 类的 Project 属性指定项目。每个控制器必须使用独立的项目。

```
m_spel.Project = "c:\EpsonRC70\projects\API_Demos\Demo1\Demo1.sprj"
```

18.3.3 使用 Visual Basic 的样本程序

- (1) 在 Visual Studio .NET 中选择菜单-[File]-[New]-[Project]。
- (2) 创建 Visual Basic 项目。
- (3) 选择[Project]-[Add Reference]。
- (4) 选择[Browse]选项卡，参照“\EpsonRC70\Exe”目录并选择“RCAPINet.dll”文件。
- (5) 添加两个按钮(btnController1、btnController2)至 Form1 类。
- (6) 添加每个按钮的快捷事件，并创建控制每个机器人控制器的线程。

```

Private trd1 As System.Threading.Thread      ' 机器人控制器 1
Private trd2 As System.Threading.Thread      ' 机器人控制器 2

Private Sub btnController1_Click(ByVal sender As
System.Object, _
    ByVal e As System.EventArgs) Handles
    btnController1.Click
    ' 开始机器人控制器 1 的线程
    trd1 = New System.Threading.Thread( _
        New System.Threading.ThreadStart(AddressOf
        StartController1))
    trd1.Start()
End Sub
Private Sub StartController1()
    ' 控制机器人控制器 1
    Try
        Dim frm1 As New frmDemo1
        frm1.ShowDialog()
        frm1.Dispose()
    Catch ex As Exception
        MsgBox(ex.Message)
    End Try
End Sub
Private Sub btnController2_Click(ByVal sender As
System.Object, _
    ByVal e As System.EventArgs) Handles btnController2.Click
    ' 开始机器人控制器 2 的线程
    trd2 = New System.Threading.Thread( _
        New System.Threading.ThreadStart(AddressOf
        StartController2))
    trd2.Start()
End Sub
Private Sub StartController2()
    ' 控制机器人控制器 2
    Try
        Dim frm2 As New frmDemo2
        frm2.ShowDialog()
        frm2.Dispose()
    Catch ex As Exception
        MsgBox(ex.Message)
    End Try
End Sub

```

- (7) 添加控制器 1 的窗体(frmDemo1)。

```

Private WithEvents m_spell As New Spel
Private Sub frmDemo1_Load(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles MyBase.Load

    Try
        m_spell.ServerInstance=1
        m_spell.Initialize()
        m_spell.Connect(5)
        m_spell.Project = "
c:\\EpsonRC70\\Projects\\Demo1\\Demo1.sprj "
    Catch ex As SpelException
        MsgBox(ex.Message)
    End Try
End Sub
Private Sub m_spell_EventReceived(ByVal sender As Object,
    ByVal e As SpelEventArgs) _Handles m_spell.EventReceived
    ' 机器人控制器 1
End Sub
Private Sub frmDemo1_FormClosed(ByVal sender As
System.Object, _
    ByVal e As
System.Windows.Forms.FormClosedEventArgs) _
    Handles MyBase.FormClosed
    m_spell.Dispose()
End Sub

```

- (8) 添加控制器 2 的窗体(frmDemo2)。

```

Private WithEvents m_spell2 As New Spel
Private Sub frmDemo2_Load(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles MyBase.Load

    Try
        m_spell2.ServerInstance=2
        m_spell2.Initialize()
        m_spell2.Connect(6)
        m_spell2.Project = "
c:\\EpsonRC70\\Projects\\Demo2\\Demo2.sprj "
    Catch ex As SpelException
        MsgBox(ex.Message)
    End Try
End Sub

Private Sub m_spell2_EventReceived(ByVal sender As Object,
    ByVal e As SpelEventArgs) _Handles m_spell2.EventReceived
    ' 机器人控制器 2
End Sub
Private Sub frmDemo2_FormClosed(ByVal sender As
System.Object, _
    ByVal e As
System.Windows.Forms.FormClosedEventArgs) _
    Handles MyBase.FormClosed
    m_spell2.Dispose()
End Sub

```

18.3.4 使用 Visual C# 的样本程序

- (1) 在 Visual Studio .NET 中选择菜单-[File]-[New]-[Project]。
- (2) 创建 Visual C# 项目。
- (3) 选择菜单-[Project]-[Add Reference]。
- (4) 选择 [Browse] 选项卡，参照 “\EpsonRC70\Exe” 目录并选择 “RCAPINet.dll” 文件。
- (5) 添加两个按钮(btnController1、btnController2)至 Form1 类。
- (6) 添加每个按钮的快捷事件，并创建控制每个机器人控制器的线程。

```
private System.Threading.Thread trd1; // 机器人控制器 1
private System.Threading.Thread trd2; // 机器人控制器 2

private void btnController1_Click(object sender,
EventArgs e)
{
    // 开始机器人控制器 1 的线程
    trd1 = new System.Threading.Thread(new _
System.Threading.ThreadStart(StartController1));
    trd1.Start();
}
private void StartController1()
{
    // 控制机器人控制器 1
    try
    {
        frmDemo1 frm1 = new frmDemo1();
        frm1.ShowDialog();
        frm1.Dispose();
    }
    catch (System.Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
private void btnController2_Click(object sender,
EventArgs e)
{
    // 开始机器人控制器 2 的线程
    trd2 = new System.Threading.Thread(new _
System.Threading.ThreadStart(StartController2));
    trd2.Start();
}
private void StartController2()
{
    // 控制机器人控制器 2
    try
    {
        frmDemo2 frm2 = new frmDemo2();
        frm2.ShowDialog();
        frm2.Dispose();
    }
    catch (System.Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

- (7) 添加控制器 1 的窗体(frmDemo1)。

```

private Spel m_spell1;
private void frmDemo1_Load(object sender, EventArgs e)
{
    m_spell1 = new Spel();
    try
    {
        m_spell1.ServerInstance = 1;
        m_spell1.Initialize();
        m_spell1.Connect(5);
        m_spell1.Project =
"c:\\EpsonRC70\\Projects\\Demo1\\Demo1.sprj";
        m_spell1.EventReceived += new _
Spel.EventReceivedEventHandler(m_spell1_EventReceive
d);
    }
    catch (SpelException ex)
    {
        MessageBox.Show(ex.Message);
    }
}
public void m_spell1_EventReceived(object sender,
SpelEventArgs e)
{
    // 机器人控制器 1
}
private void frmDemo1_FormClosed(object sender,
FormClosedEventArgs e)
{
    m_spell1.Dispose();
}

```

- (8) 添加控制器 2 的窗体(frmDemo2)。

```

private Spel m_spell2;
private void frmDemo2_Load(object sender, EventArgs e)
{
    m_spell2 = new Spel();
    try
    {
        m_spell2.ServerInstance = 2;
        m_spell2.Initialize();
        m_spell2.Connect(6);
        m_spell2.Project =
"c:\\EpsonRC70\\Projects\\Demo2\\Demo2.sprj";
        m_spell2.EventReceived += new _
Spel.EventReceivedEventHandler(m_spell2_EventReceived);
    }
    catch (SpelException ex)
    {
        MessageBox.Show(ex.Message);
    }
}
public void m_spell2_EventReceived(object sender,
SpelEventArgs e)
{
    // 机器人控制器 2
}
private void frmDemo2_FormClosed(object sender,
FormClosedEventArgs e)
{
    m_spell2.Dispose();
}

```