EPSON

EPSON RC+ 7.0 Option

RC+ API 7.0

Rev.11

EM186S3690F

EPSON RC+ 7.0 Option RC+ API 7.0 Rev.11

EPSON RC+ 7.0 Option

RC+ API 7.0

Rev.11

FOREWORD

Thank you for purchasing our robot products.

This manual contains the information necessary for the correct use of the Manipulator.

Please carefully read this manual and other related manuals before installing the robot system.

Keep this manual handy for easy access at all times.

WARRANTY

The robot and its optional parts are shipped to our customers only after being subjected to the strictest quality controls, tests, and inspections to certify its compliance with our high performance standards.

Product malfunctions resulting from normal handling or operation will be repaired free of charge during the normal warranty period. (Please ask your Regional Sales Office for warranty period information.)

However, customers will be charged for repairs in the following cases (even if they occur during the warranty period):

1. Damage or malfunction caused by improper use which is not described in the manual, or careless use.
2. Malfunctions caused by customers' unauthorized disassembly.
3. Damage due to improper adjustments or unauthorized repair attempts.
4. Damage caused by natural disasters such as earthquake, flood, etc.

Warnings, Cautions, Usage:

1. If the robot or associated equipment is used outside of the usage conditions and product specifications described in the manuals, this warranty is void.
2. If you do not follow the WARNINGS and CAUTIONS in this manual, we cannot be responsible for any malfunction or accident, even if the result is injury or death.
3. We cannot foresee all possible dangers and consequences. Therefore, this manual cannot warn the user of all possible hazards.

TRADEMARKS

Microsoft, Windows, and Windows logo are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other brand and product names are trademarks or registered trademarks of the respective holders.

TRADEMARK NOTATION IN THIS MANUAL

Microsoft® Windows® XP Operating system
Microsoft® Windows® Vista Operating system
Microsoft® Windows® 7 Operating system
Microsoft® Windows® 8 Operating system
Microsoft® Windows® 10 Operating system
Throughout this manual, Windows XP, Windows Vista, Windows 7, Windows 8, and Windows 10 refer to above respective operating systems. In some cases, Windows refers generically to Windows XP, Windows Vista, Windows 7, Windows 8, and Windows 10.

NOTICE

The contents of this manual are subject to change without notice.
Please notify us if you should find any errors in this manual or if you have any comments regarding its contents.

MANUFACTURER

SEIKO EPSON CORPORATION

1. Introduction

The EPSON RC+ 7.0 Option RC+ API enables you to use Microsoft Visual Basic or any other language that supports .NET technology to run your robotic applications. This gives you the power to create sophisticated user interfaces, use databases, and use third party products designed for use with .NET. A LabVIEW library is also included.

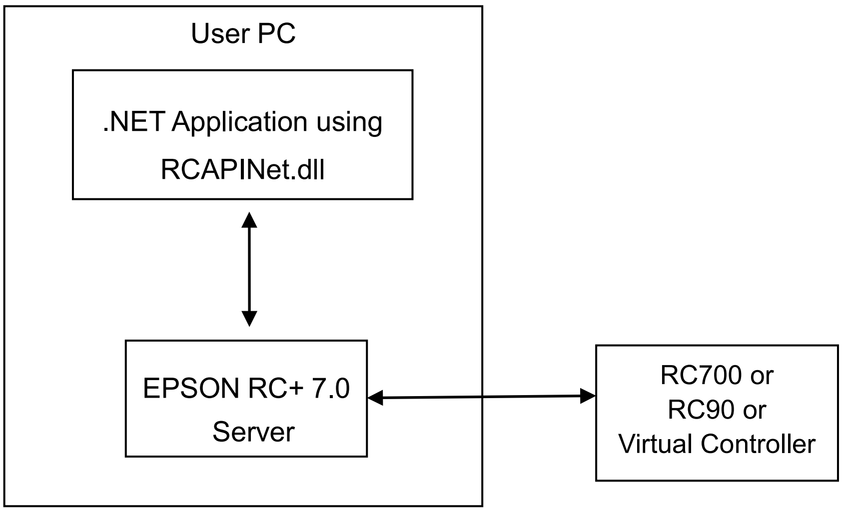
1.1 Features

The following features are supported in the RC+ API package:

- A .NET library and LabVIEW library.
- Supports 32 bit and 64 bit applications.
- Properties and methods for controlling multiple robots, I/O, and tasks from multiple controllers.
- Methods for executing vision and force sensing commands.
- Supports parallel execution of asynchronous commands by multi-threading.
- Several EPSON RC+ 7.0 windows and dialogs can be used by your .NET application, including:
  - Robot Manager
  - IO monitor
  - Task manager
  - Simulator
  - Controller Tools dialog

During development, EPSON RC+ 7.0 can be run along with Visual Basic. In production facilities, EPSON RC+ 7.0 can be run invisibly in the background.

The figure below shows the basic structure of a system using the RC+ API.



RC+ API Basic Structure for the .NET library

EPSON RC+ 7.0 is an out-of-process server for the RCAPINet library. Each instance of RCAPINet Spel class can start an instance of EPSON RC+ 7.0.

2. Installation

Please follow the instructions in this chapter to help ensure proper installation of the RC+ API software.

Before starting, ensure that all Windows applications have been closed.

2.1 Step by step instructions

1. Install one of the Visual Studio 2008, 2010, 2012, or 2013 Express versions, such as Visual Basic Express, Visual Studio 2015 Community, or install Visual Studio 2008, 2010, 2012, 2013, or 2015. Or install LabVIEW 2009 or greater.
2. Install EPSON RC+ 7.0.
3. If you are using LabVIEW, install the LabVIEW VI library.
4. Ensure that the software key has been enabled for RC+ API in the controllers you will be using. Refer to the EPSON RC+ 7.0 User's Guide for information on how to enable options in the controller.

This completes the RC+ API installation.

2.2 What's installed

The directories and files shown in the table below are installed on your PC during installation.

Directories and Files	Description
\EPSONRC70\API\VS20xx\VB\DEMOS	Visual Basic .NET demonstrations
\EPSONRC70\API\VS20xx\VCS\DEMOS	Visual C# .NET demonstrations
\EPSONRC70\API\VS20xx\VC\DEMOS	Visual C++ .NET demonstrations
\EPSONRC70\API\LabVIEW	LabVIEW VI Library installer
\EPSONRC70\PROJECTS\API_Demos	EPSON RC+ 7.0 projects for demos
\EPSONRC70\EXE\RCAPINet.dll	RCAPINet Class library (32 bit or 64 bit)
\EPSONRC70\EXE\SpelNetLib70.dll[1]	SpelNetLib70 Class library (32 bit)
\EPSONRC70\EXE\SpelNetLib70_x64.dll[1]	SpelNetLib70 Class library (64 bit)

[1]: These libraries are obsolete, and are provided for backwards compatibility. The RCAPINet library replaces these libraries and can be used with 32 bit or 64 bit applications.

3. Getting Started

This chapter contains information for getting started in the following development environments.

- Visual Basic .NET

- Visual C# .NET

- Visual C++ .NET

Demonstration programs are supplied with the RC+ API. It is recommended that you go through the demonstrations to get more familiar with the product.

LabVIEW users should now refer to chapter *16. Using the LabVIEW VI Library* for instructions on getting started and using the library.

3.1 Getting started using Visual Basic

To use RCAPINet in a Visual Basic .NET project, declare a Spel Class instance, as shown in the example below. *g_spel* can now be used in your project.

1. In Visual Studio .NET, select File | Project.
2. Create a Visual Basic project as Windows Forms Application.
3. From the Project menu, select Add Reference.
4. In the NET Components tab, browse to the \EpsonRC70\Exe directory and select the RCAPINet.dll file.
5. From the Project menu, create a new module and add the following code.

```
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\Demo1
\demo1.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
```

NOTE

When your application exits, you need to execute Dispose for each Spel class instance. This can be done in your main form's FormClosed event. If Dispose is not executed, the application will not shutdown properly.

```
g_spel.Dispose()
```

3.2 Getting started using Visual C#

1. In Visual Studio .NET, select File | Project.
2. Create a Visual C# project as Windows Forms Application.
3. From the Project menu, select Add Reference.
4. Select the Browse tab and browse to the \EpsonRC70\Exe directory and select the RCAPINet.dll file.
5. In the Form1 class, declare a Spel class variable as shown below.
```
private RCAPINet.Spel m_spel;
```
6. In the Form_Load event, add initialization code, as shown below.
```
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";

m_spel.EventReceived += new
   RCAPINet.Spel.EventReceivedEventHandler(m_spel_EventReceiv
ed);
```
7. Add the event handler, as shown below.
```
public void m_spel_EventReceived(object sender,
   RCAPINet.SpelEventArgs e)
{
}
```

NOTE

When your application exits, you need to execute Dispose for each Spel class instance. This can be done in your main form's FormClosed event. If Dispose is not executed, the application will not shutdown properly.

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

3.3 Getting started using Visual C++

1. In Visual Studio .NET, select File | Project.
2. Create a Visual C++ CLR Windows Forms Application project.
3. From the Project menu, select References
4. Click the Add New Reference button.
5. Select the Browse tab and browse to the \EpsonRC70\Exe directory and select the RCAPINet.dll file.
6. In the Form1 class, declare a Spel variable as shown below.
```
private RCAPINet::Spel^ m_spel;
```
7. In the Form_Load event, add initialization code, as shown below.
```
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";
   m_spel->EventReceived += gcnew
       RCAPINet::Spel::EventReceivedEventHandler(
       this, &Form1::m_spel_EventReceived);
}
```
8. Add the event handler, as shown below.
```
private System::Void m_spel_EventReceived(
    System::Object^ sender, RCAPINet::SpelEventArgs^ e)
{
   MessageBox::Show(e->Message);
}
```

NOTE

When your application exits, you need to delete each Spel class instance if it was allocated on the heap (using gcnew). This can be done in your main form's FormClosed event. If the Spel class instances are not deleted, then the application will not shutdown properly.

```
delete m_spel;
```

4. Environments

4.1 Development Environment

4.1.1 Development Startup

Typically, you would perform these steps to start development:

1. Declare a Spel class variable in a module in your .NET project.
2. Start EPSON RC+ 7.0.
3. Open the desired EPSON RC+ 7.0 project or create a new EPSON RC+ 7.0 project.
4. Build the EPSON RC+ 7.0 project.
5. Add initialization code for the SPEL class instance.
6. Run and debug the .NET project.

4.1.2 Spel Class Instance Initialization

After a new instance of the Spel class has been created, it needs to be initialized. When initialization occurs, the underlying EPSON RC+ 7.0 modules are loaded and initialized. Initialization is implicit with the first method call or property access. You can initialize the class by calling the Initialize method.

```
m_spel.Initialize()
```

4.1.3 Spel Class Instance Termination

When your application exits, you need to execute Dispose for each Spel class instance. This can be done in your main form's FormClosed event. If Dispose is not executed, the application will not shutdown properly.

For Visual Basic and Visual C#, use the Dispose method:

```
m_spel.Dispose()
```

For Visual C++, if your Spel class instance was created on the heap (with gcnew), then use delete:

```
delete m_spel;
```

4.1.4 Development Cycle

Follow these basic steps to edit and run your .NET code:

1. Stop the .NET project.
2. Edit the .NET project
3. Open EPSON RC+ 7.0.
4. Make changes in the EPSON RC+ 7.0 project.
5. Build the EPSON RC+ 7.0 project.
6. Close the RC+ 7.0.
7. Switch to Visual Studio.
8. Run the .NET project.

4.2 In Production Facilities

4.2.1 Opening EPSON RC+ 7.0 at Runtime

Decide if you want to allow the EPSON RC+ 7.0 environment to be opened from your application. This is especially useful for debugging. Set the **OperationMode** property to Program to put EPSON RC+ 7.0 in Program Mode and open the EPSON RC+ 7.0 GUI.

4.2.2 Using EPSON RC+ 7.0 Dialogs and Windows

At runtime, you can open and hide certain EPSON RC+ 7.0 windows from your .NET application. You can also run certain EPSON RC+ 7.0 dialogs. See the chapter *EPSON RC+ 7.0 Windows and Dialogs* for details.

4.2.3 Installation on Target System

You should make an installation program for your .NET project by using a Visual Studio setup project. Then follow these steps to setup a target system for your .NET application:

1. Install EPSON RC+ 7.0.
2. Install your EPSON RC+ 7.0 project.
3. Install your .NET application.

5. Executing Methods, Programs, Tasks

5.1 Executing Methods

There are several methods in the Spel class. For descriptions of available methods, see the section *14.3 Spel Class Methods*. When you execute a method, the associated internal functions are called in the EPSON RC+ server process, which in turn communicates with the controller to execute the associated function. There are two types of methods: immediate and asynchronous. For immediate methods, the internal function is executed in the controller and the reply is returned immediately. Immediate commands include all I/O commands. For asynchronous methods, the associated function is started in the controller, and then the Spel class instance waits for an event from the EPSON RC+ server process indicating that the function has completed. Asynchronous methods include all robot motion commands. While waiting for command completion, the Spel class instance dispatches Windows events, so that the user GUI is still responsive. For example, when the Go method is called, the robot is moving to a point, and the user may want to stop it by clicking a button. You can disable Windows event dispatching during asynchronous methods by setting DisableMsgDispatch to True. You can also wait for asynchronous methods to finish in your program by setting AsyncMode to True.

5.1.1 Using Multiple Threads

You can execute Spel methods in multiple threads in your application. The sections below describe the various scenarios.

One Spel class instance used in multiple threads

You can execute methods with the same Spel class instance in multiple threads, but only one asynchronous command at a time. If you attempt to execute an asynchronous command in one thread while another asynchronous command is already executing in another thread, you will get a "command in cycle" error. You can execute an immediate command in one thread while executing an asynchronous command in another thread.

Separate Spel class instance used in each thread

For each controller connection, you can have one or more Spel class instances. The first instance for each controller initializes an EPSON RC+ 7.0 server process and connects to the specified controller. To use one or more additional instances in other threads to communicate with the same controller, you must specify the ServerInstance property to be the same value. You call Initialize for the first instance before using additional Spel class instances.

```
' Initialize Spel class instance for thread 1
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)

' Initialize Spel class instance for thread 2
' This instance uses the same controller as m_spel_1
m_spel_2 = New Spel
m_spel_2.ServerInstance = 1
```

Thread 1

```
' Uses instance m_spel_1 for motion
m_spel_1.Robot = 1
Do
 m_spel_1.Go(1)
 m_spel_1.Go(2)
Loop Until m_stop
```

Thread 2

```
' Uses instance m_spel_2 for I/O
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
```

Using API threads in the controller

By default, only one API thread is supported in the controller. In this case, asynchronous methods are executed one at a time in the controller, even when controlling multiple robots. For most applications that use one robot, or execute robot motion using SPEL+ tasks, this is sufficient, but you can configure the system to use up to 10 API tasks in the controller to allow parallel processing for your .NET threads, such as when you are controlling more than one robot from the same controller.

There are two basic steps required to use more than one API task in the controller.

1. In the EPSON RC+ GUI, connect to the controller, then open [Setup]-[System Configuration]-[Controller]-[Preferences]. Set "Reserved tasks for API" to the desired number of API tasks. Note that the more tasks you reserve for the API, the fewer tasks will be available for your SPEL+ programs. For example, if you reserve 5 API tasks, then there will be 27 tasks (32 – 5) available for SPEL+.
2. In your application, set the CommandTask property to specify which API task you want to execute methods on.

In the simple example below, there is one thread for each robot in the same controller. The robot motion commands will execute in parallel, since a different CommandTask is used in each thread, and ServerInstance is set to 1 for both Spel instances.

```
' Initialize Spel class instance for thread 1
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)

' Initialize Spel class instance for thread 2
' This instance uses the same controller as m_spel_1
' And uses the second CommandTask in the controller.
m_spel_2 = New Spel
m_spel_2.ServerInstance = 1
m_spel_1.CommandTask = 2
```

Thread 1

```
' Uses instance m_spel_1 for Robot 1 motion
m_spel_1.Robot = 1
Do
 m_spel_1.Go(1)
 m_spel_1.Go(2)
Loop Until m_stop
```

Thread 2

```
' Uses instance m_spel_2 for Robot 2 motion
m_spel_2.Robot = 2
Do
 m_spel_2.Go(1)
 m_spel_2.Go(2)
Loop Until m_stop
```

5.2 Executing SPEL+ Programs

A SPEL+ program contains one or more functions, and the program is run by starting the main function of the program. You can run any of the 64 built-in main functions in the current controller project by using the *Start* method of the Spel class. The main function(s) that you start must be defined in your SPEL+ code. When you start a main function, all global variables and module variables are cleared to default values.

The table below shows the program numbers and their corresponding function names in the SPEL+ project.

Program Number	SPEL+ Function Name
0	main
1	main1
2	main2
3	main3
…	…
63	main63

Here is an example that starts function “main”:

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

    m_spel.Start(0)  ' Starts function main
    btnStart.Enabled = False
    btnStop.Enabled = True
End Sub
```

5.3 Executing SPEL+ Tasks

You can execute functions in your SPEL+ program as a normal task by using the Xqt method. When you execute a task, global variables are not cleared to default values, as they are when you use the Start method.

To suspend and resume a task, use the Halt and Resume methods.

To quit a task, use the Quit method.

You can also start controller background tasks using the StartBGTask method.

5.4 Aborting All Tasks

If you are running tasks and want to abort all tasks at once, you can use the *Stop* method of the Spel class. The Stop method has an optional parameter that allows you to additionally stop all background tasks.

For example:

```
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
```

6. Events

6.1 Overview

The Spel Class supports two types of events: system events and user events. System events are notifications of system status. User defined events are sent from any SPEL+ task to the .NET application.

6.2 System Events

There are several system events that are sent to the .NET application. Each system event indicates a change in status. There are events for Pause, Continue, Emergency Stop, etc. For complete details on all system events, see the description for *14.4 Spel Class Events - EventReceived*.

Use the Spel class EnableEvents method to control which system events are sent.

6.3 User Events from SPEL+

You can cause events to occur in your .NET application from your SPEL+ programs. For example, you can inform the .NET application about a continuous cycle loop. This is a better method to use than polling for variable values in the controller from .NET.

To fire an event to .NET from SPEL+, use the SPELCom_Event command in a SPEL+ program statement. For example:

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

The SPELCom_Event command is similar to a Print command. You can specify one or more pieces of data to be sent to the .NET application. See *13. SPELCom_Event* for details on SPELCom_Event.

Before you can receive events, you must declare your Spel class variable using the WithEvents clause.

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

Catch the event in the EventReceived routine for the Spel class instance. To edit this routine, in the module where the Spel class is declared select “m_spel” from the class name list and EventReceived from the procedure list.

Here is an example of code in the EventReceived routine that updates some labels when an event occurs.

```
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
```

7. Error Handling

7.1 Errors for Spel methods

When you execute a Spel class method, an exception is thrown if there are any errors.

When an error occurs, the Spel class instance throws it to the calling routine. You should use error handlers in your application to catch this error. In some cases, you will only want to display an error message. For example:

```
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
```

You can examine the error number associated with the exception by using the ErrorNumber property of SpelException.

```
  Try
    m_spel.Start(0)
  Catch ex As RCAPINet.SpelException
    MsgBox(ex.ErrorNumber)
  End Try
```

8. Handling Pause and Continue

8.1 Pause state

When a pause occurs, the controller and SPEL+ tasks are in the pause state.

The controller is in the pause state after one of the following occurs while tasks are running:

- The Spel class Pause method was executed
- A SPEL+ task executed Pause.
- The safeguard was opened.

8.2 Catching the Pause event

The Spel class will signal your .NET application that a pause has occurred.

You can catch the Pause event in the `EventReceived` event for the Spel class.

```
(Visual Basic)
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
```

8.3 Executing Pause

The following routine shows how to issue a PAUSE from Visual Basic using the *Pause* method.

```
(Visual Basic)
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
```

8.4 Continue after pause

To continue after a pause has occurred, use the *Continue* method.

```
(Visual Basic)
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
```

8.5 Abort after pause

You can also execute the *Stop* method if you don't want to continue after a pause.

```
(Visual Basic)
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
```

9. Handling Emergency Stop

When an Emergency stop occurs, you may want to perform some specific action in your program, such as displaying a dialog, or a message box.

The Spel class issues two standard events for emergency stop status: EStopOn and EStopOff.

9.1 Using system EStop events

You can catch the system EStop events in the EventReceived handler in your Visual Basic application.

```
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.EstopOn
            MsgBox "E-Stop detected"
            gEStop = True
            lblEStop.BackColor = Color.Red
            lblEStop.Text = "EStop ON"
        Case RCAPINet.EstopOn
            gEStop = False
            lblEStop.BackColor = Color.Green
            lblEStop.Text = "EStop OFF"
    End Select
End Sub
```

10. EPSON RC+ 7.0 Windows and Dialogs

You can open certain EPSON RC+ 7.0 windows and dialogs from your .NET application using the ShowWindow and RunDialog methods of the Spel class.

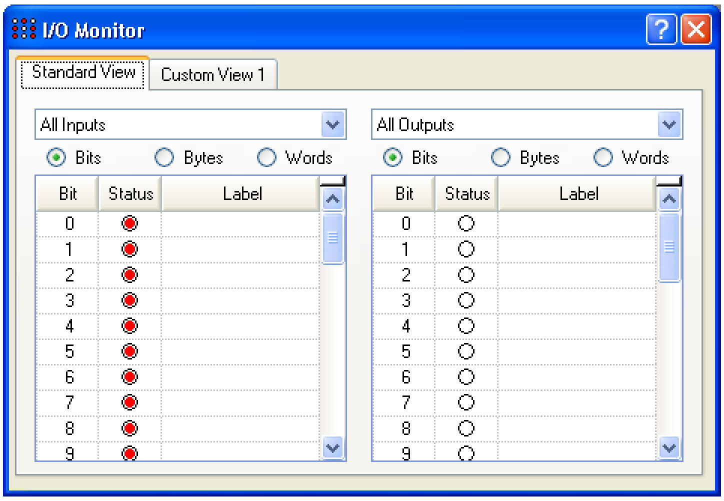
10.1 Windows

Windows are non-modal, meaning that they can remain open while other elements of your Visual Basic GUI can be used. You can show and hide EPSON RC+ 7.0 windows from your Visual Basic program.

For example, to open and close the I/O Monitor window:

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

WindowID	Window
RCAPINet.SpelWindows.IOMonitor	IO Monitor
RCAPINet.SpelWindows.TaskManager	Task Manager
RCAPINet.SpelWindows.ForceMonitor	Force Monitor
RCAPINet.SpelWindows.Simulator	Simulator



I/O Monitor Window

10.2 Dialogs

Dialogs are modal: when a dialog is opened, other elements of your .NET GUI cannot be used until the dialog is closed.

For example, to open the Robot Manager dialog:

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

Once a dialog has been opened, it must be closed by the operator. You cannot close a dialog from within your program. This is for safety reasons.

The following table shows the dialogs that can be opened.

DialogID	Dialog
RCAPINet.SpelDialogs.RobotManager	Robot Manager
RCAPINet.SpelDialogs.ControllerTools	Controller Tools
RCAPINet.SpelDialogs.VisionGuide	Vision Guide

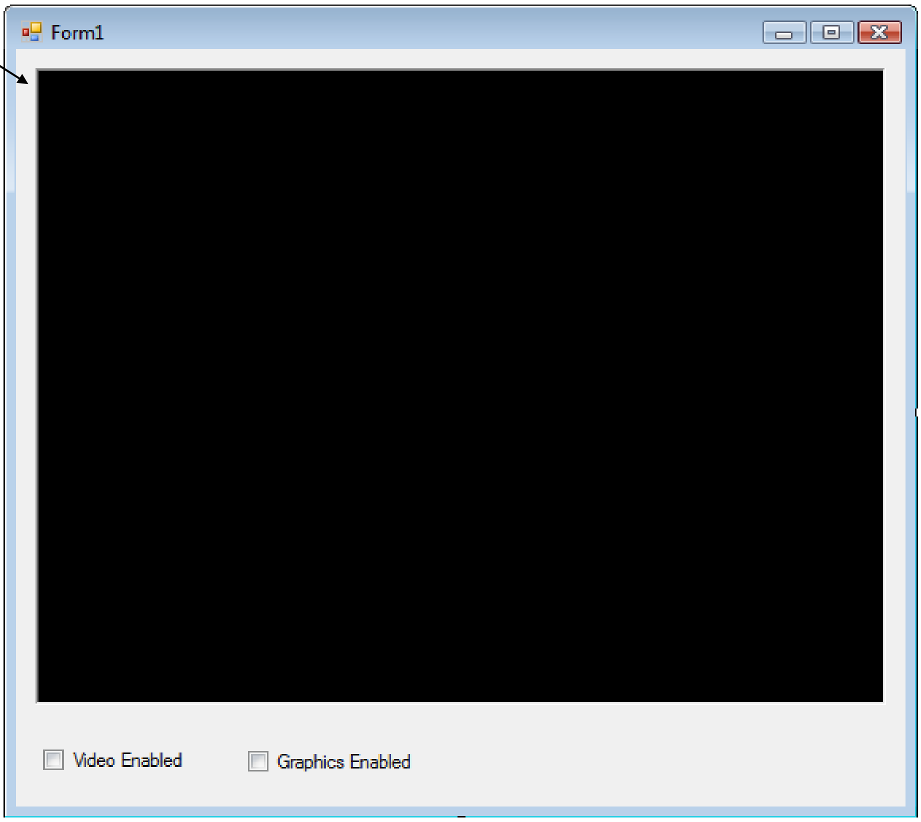
11. Displaying Video

You can display video on a form in your application by using the SPELVideo control. When you run a vision sequence, the graphics can also be displayed on the window.

Perform the following steps to create a video display:

1. Add the SPELVideo component to your project. To add the control to your Visual Studio .NET toolbox, right click on the toolbox and select Choose Items. Select the Browse tab and browse to the \EpsonRC70\Exe directory and select the RCAPINet.dll file. The SPELVideo control icon will be added to the toolbox.
2. Place a SPELVideo control on the form you want the video to be displayed. The control size can be changed up to the full size.
3. Set the VideoEnabled property to True.
4. Set the GraphicsEnabled property to True if you want to display vision graphics. You must also attach the SPELVideo control to a Spel class instance using the Spel class SpelVideoControl property.

SPELVideo control



SPELVideo control placed on a form

When the GraphicsEnabled property is True and the control is attached to a Spel class instance, then vision graphics will be displayed whenever the VRun method is executed on the controller connected to the Spel class instance.

Here is an example showing how to enable video and graphics on a Visual Basic form where a Spel class instance is used and a SPELVideo control have been placed:

```
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
```

Using multiple video displays

Starting with EPSON RC+ 7.0 version 7.3.0 or later, you can use multiple video displays in your application. For each display, you can select which camera video to display.

To use multiple displays, you must set the SpelVideoControl property for each display.

The example below shows initialization that includes two video displays.

```
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
```

12. Using AsyncMode

AsyncMode allows you to execute Spel methods while other methods are executing. Only the following Spel class methods are allowed to execute asynchronously:

Arc	Jump3
Arc3	Jump3CP
Curve	Mcal
CVMove	Move
ExecuteCommand	PTran
Go	Pulse
Home	TGo
JTran	TMove
Jump	

To execute a method asynchronously, set the AsyncMode property to True, then execute the method. When the AsyncMode property is true and you execute an asynchronous method, the method will be started and control will return immediately back to the .NET application for further processing.

If you execute another asynchronous method while a previous one is executing, SPEL will wait for the first method to complete, then start the next method and return back to .NET.

To wait for an asynchronous method to complete, you can use one of the following:

- Execute the WaitCommandComplete method.
- Set AsyncMode property to False.

If an asynchronous command cannot be started due to an error (e.g. point does not exist), then an exception will occur immediately. However, if an error occurs during a command running asynchronously, the error exception occurs on the next execution of an asynchronous command or execution of WaitCommandComplete, or AsyncMode is set to False. If the exception occurs on the next command, you do not know which statement caused the error (the previous statement or the current statement). If you need to check if an asynchronous command completed successfully before executing another command, then call WaitCommandComplete before the next command. If an error occurred during the previous asynchronous command, a SpelException exception will occur with the error number and message. See the example below.

```
Try
   m_spel.AsyncMode = True
   m_spel.Go(1)
   ' do other things here during motion
   ' When Go(2) executes, an exception occurs if Go(1) had
   ' an error during execution, so we don't know if error
   ' occurred for Go(1) or Go(2)
   m_spel.Go(2)

   m_spel.Go(3)
   ' do other things here during motion
   ' Check if Go(3) was successful
   m_spel.WaitCommandComplete()  ' Exception occurs if Go(3) had an error

   m_spel.Go(4)
Catch ex As SpelException
   ' Handle the error exception

End Try
```

13. SPELCom_Event

Generates a user event from a Spel class instance.

Syntax

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

Parameters

eventNumber	An integer expression whose value is from 1000 - 32767.
msgArg1, msgArg2, msgArg3...	Optional. Each message argument can be either a number, string literal, or a variable name.

Description

This instruction makes it easy to send real time information to an application from a Spel task running in the controller. For example, you can update parts count, lot number, etc. by sending an event.

SPELCom_Event Example

In this example, a SPEL+ task sends cycle data to an application using the RC+ API .

```
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
Fend
```

14. RCAPINet Reference

14.1 Spel Class

Description

This class allows you to execute commands and receive events from EPSON RC+ 7.0.

File Name

RCAPINet.dll (64-bit and 32-bit)

SpelNetLib70.dll (32-bit) (Obsolete)

SpelNetLib70_x64.dll (64-bit) (Obsolete)

14.2 Spel Class Properties

AsyncMode Property, Spel Class

Description

Sets / returns asynchronous execution mode.

Syntax

Property **AsyncMode** As Boolean

Default value

False

Return value

A Boolean value that is True if asynchronous mode is active, False if not.

See Also

Using AsyncMode, WaitCommandComplete

AsyncMode Example

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

AvoidSingularity Property, Spel Class

Description

Sets / returns singularity avoidance mode.

Syntax

Property **AvoidSingularity** As Boolean

Default value

False

Return value

A Boolean value that is True if singularity avoidance is active, False if not.

See Also

Go, Jump, Move

AvoidSingularity Example

```
m_spel.AvoidSingularity = True
```

CommandInCycle Property, Spel Class

Description

Returns whether a method is being executed.

Syntax

ReadOnly Property **CommandInCycle** As Boolean

Return value

A Boolean value that is True if a method is executing, False if not.

See Also

AsyncMode

CommandInCycle Example

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

CommandTask Property, Spel Class

Description

Specifies the reserved API task to use in the controller for executing robot commands.

Syntax

Property **CommandTask** As Integer

Default Value

The default value is 0 (do not use a reserved API task).

Remarks

Use CommandTask when you want to execute Spel robot commands on another thread in the controller. Normally, CommandTask is used on a multi-robot system. Before using CommandTask, you must first reserve tasks in the controller to be used by the API from EPSON RC+ menu -[Setup]-[System Configuration]-[Controller]-[Preferences]. You can reserve up to 16 API tasks in the controller.

See Also

ServerInstance

CommandTask Example

```
' In Robot1 thread
m_spel.CommandTask = 1
m_spel.Robot = 1

' In Robot2 thread
m_spel.CommandTask = 2
m_spel.Robot = 2
```

DisableMsgDispatch Property, Spel Class

Description

Sets / returns whether Windows messages should be processed during Spel method execution.

Syntax

DisableMsgDispatch

Type

Boolean

Default Value

False

Remarks

This property should normally not be used. It is intended for special applications that do not want keyboard or mouse processing while a Spel method is executing.

ErrorCode Property, Spel Class

Description

Returns the current controller error code.

Syntax

ReadOnly Property **ErrorCode** As Integer

Return Value

Integer value containing the error code.

See Also

ErrorOn

ErrorCode Example

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

ErrorOn Property, Spel Class

Description
Returns True if a critical error has occurred in the controller.

Syntax
ReadOnly Property **ErrorOn** As Boolean

Return Value
True if the controller is in the error state, False if not.

Remarks
When the controller is in the error state, the ErrorOn property returns True, and you can retrieve the error code by using the ErrorCode property.

See Also
ErrorCode

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

EStopOn Property, Spel Class

Description
Returns the status of the controller's emergency stop.

Syntax
ReadOnly Property **EStopOn** As Boolean

Return Value
True if the emergency stop is active, False if not.

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

Force_Sensor Property, Spel Class

Description
Sets and return the current force sensor number.

Syntax
Property **Force_Sensor** As Integer

Default value
1

Return value
An Integer value that is the current force sensor number

Remarks
Before using any force methods, you must set the current force sensor using this property.

See Also
Force_Calibrate, Force_GetForces, Force_SetTrigger

Force_Sensor Example
```
' Read the Z axis force for sensor 2
m_spel.Force_Sensor = 2
f = m_spel.Force_GetForce(3)
```

MotorsOn Property, Spel Class

Description
Sets and return the status of the motor power on or off for the current robot.

Syntax
Property **MotorsOn** As Boolean

Default value
False

Return value
A Boolean value that is True if motors are on, False if not.

See Also
PowerHigh, Reset, Robot

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

NoProjectSync Property, Spel Class

Description

Sets / returns whether the current project in the PC should be synchronized with the controller project.

Syntax

NoProjectSync

Type

Boolean

Default Value

False

Remarks

When NoProjectSync is set to False (default), then the Spel class ensures that the project on the PC is synchronized with the project on the controller.

When NoProjectSync is set to True, the Spel class does not check for any project on the PC and does not synchronize the PC project with the controller. This allows you to run programs in the controller without any project on the PC.

This property is not persistent. You must set it after creating a Spel class instance if you want to set it to True.

See Also

Start

NoProjectSync Examples

```
m_spel.Initialize
m_spel.NoProjectSync = True
```

OperationMode Property, Spel Class

Description

Reads or sets the EPSON RC+ 7.0 mode of operation.

Syntax

Property **OperationMode** As SpelOperationMode

Return value

SpelOperationMode value

Remarks

When **OperationMode** is set to Program, the EPSON RC+ 7.0 GUI for the current instance of the Spel class is opened and the controller operation mode is set to Program. If the user closes the GUI, **OperationMode** is set to Auto. If **OperationMode** is set to Auto from Visual Basic, the GUI also closes.

OperationMode Example

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

  Try
    m_spel.OperationMode = _
                  RCAPINet.SpelOperationMode.Program
    ' If you want to wait for the user to close the RC+ GUI,
    ' you can wait here for OperationMode to change to 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
```

ParentWindowHandle Property, Spel Class

Description

Sets / returns the handle for the parent window used for dialogs and windows.

Syntax

Property **ParentWindowHandle** As Integer

Return Value

Integer value containing the window handle.

Remarks

Use ParentWindowHandle to specify the parent window from applications that do not have .NET forms, such as LabVIEW.

See Also

ShowWindow

ParentWindowHandle Example

```
m_spel.ParentWindowHandle = Me.Handle

m_spel.ShowWindow(RCAPINet.SpelWindows.IOMonitor)
```

PauseOn Property, Spel Class

Description
Returns status of the controller pause state.

Syntax
ReadOnly Property **PauseOn** As Boolean

Return Value
True if the controller is in the pause state, False if not.

See Also
Continue Pause

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

PowerHigh Property, Spel Class

Description
Sets and returns the power state for the current robot.

Syntax
Property **PowerHigh** As Boolean

Default Value
False

Return Value
True if the current robot power is high, False if not.

See Also
MotorsOn

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

Project Property, Spel Class

Description

Sets / returns the current project.

Syntax

Property **Project** As String

Default Value

Empty string.

Return Value

A string containing the project path and file.

Remarks

When setting the **Project**, you must supply the full path and name of the EPSON RC+ 7.0 project make file. The make file is the project name with a .SPRJ extension.

Project Example

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

ProjectBuildComplete Property, Spel Class

Description
Returns the status of the current project build.

Syntax
ReadOnly Property **ProjectBuildComplete** As Boolean

Return Value
True if the project build is complete, False if not.

See Also
BuildProject

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

ResetAbortEnabled Property, Spel Class

Description
Sets / returns whether ResetAbort method should be enabled or not.

Syntax
Property **ResetAbortEnabled** As Boolean

Default Value
True

Return Value
True if ResetAbort is enabled, False if not.

See Also
ResetAbort

ResetAbortEnabled Example
```
' Enable reset abort
m_spel.ResetAbortEnabled = True
```

Robot Property, Spel Class

Description
Sets / returns the current robot number.

Syntax
Property **Robot** As Integer

Default Value
If one or more robots exists, the default value for the first Spel instance is 1, otherwise it is 0. For all other Spel instances, the default value is 0.

Return Value
Integer value that contains the current robot number.

Remarks
On a system using multiple robots, use the **Robot** property to set the robot for subsequent robot related commands, such as motion commands.

See Also
RobotModel, RobotType

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

RobotModel Property, Spel Class

Description
Returns the model name for the current robot.

Syntax
ReadOnly Property **RobotModel** As String

Return Value
String that contains the current robot's model name.

See Also
Robot, RobotType

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

RobotType Property, Spel Class

Description
Returns the type of the current robot.

Syntax
ReadOnly Property **RobotType** As SpelRobotType

Return Value
SpelRobotType value

See Also
Robot, RobotModel

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

SafetyOn Property, Spel Class

Description
Returns status of the controller's safeguard input.

Syntax
ReadOnly Property **SafetyOn** As Boolean

Return Value
True if the safeguard is open, False if not.

Remarks
Use the SafetyOn property to obtain the safeguard status when your application starts, then use the SafeguardOpen and SafeguardClose events to update the status.

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

ServerInstance Property, Spel Class

Description

Specifies which instance of EPSON RC+ server to use.

Syntax

Property **ServerInstance** As Integer

Default Value

The default value is the next available server instance.

Remarks

Use ServerInstance when you want to use multiple instances of the Spel class for communication with the same controller.

By default, when you create a new Spel class instance, the ServerInstance is automatically set starting with 1. So each Spel class instance can control one robot controller.

Sometimes you may want multiple instances of the Spel class for the same controller. In that case, you can set the ServerInstance property.

See Also

CommandTask

ServerInstance Example

```
spel = New Spel
spel.ServerInstance = 1
```

SPELVideoControl Property, Spel Class

Description
Used to connect a SPELVideo control to the Spel class instance so that video and graphics can be displayed.

Syntax
Property **SpelVideoControl** As SpelVideo

See Also
Graphics Enabled, VideoEnabled, Camera

SpelVideoControl Example
```
m_spel.SpelVideoControl = SpelVideo1
```

Version Property, Spel Class

Description
Returns the current EPSON RC+ 7.0 software version.

Syntax
ReadOnly Property **Version** As String

Return Value
String that contains the current EPSON RC+ 7.0 software version.

Version Example
```
' Get version of software
curVer = m_spel.Version
```

WarningCode Property, Spel Class

Description
Returns controller warning code.

Syntax
ReadOnly Property **WarningCode** As Integer

Return Value
Integer value that contains the current controller warning code.

See Also
WarningOn

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

WarningOn Property, Spel Class

Description

Returns status of the controller warning state.

Syntax

ReadOnly Property **WarningOn** As Boolean

Return Value

True if the controller is in the warning state, False if not.

See Also

WarningCode

WarningOn Example

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

14.3 Spel Class Methods

Accel Method, Spel Class

Description

Sets acceleration and deceleration for point to point motion commands Go, Jump, and Pulse.

Syntax

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

Parameters

PointToPointAccel	Integer expression between 1-100 representing a percentage of maximum acceleration rate.
PointToPointDecel	Integer expression between 1-100 representing a percentage of maximum deceleration rate.
JumpDepartAccel	Integer expression between 1-100 representing a percentage of maximum acceleration rate for Jump command Z Axis upward motion.
JumpDepartDecel	Integer expression between 1-100 representing a percentage of maximum deceleration rate for Jump command Z Axis upward motion.
JumpApproAccel	Integer expression between 1-100 representing a percentage of maximum acceleration rate for Jump command Z Axis downward motion.
JumpApproDecel	Integer expression between 1-100 representing a percentage of maximum deceleration rate for Jump command Z Axis downward motion.

See Also

Accels, Speed

Accel Example

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

AccelR Method, Spel Class

Description
Sets acceleration and deceleration for tool rotation motion.

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

Parameters

Accel	Single expression from 0.1 to 5000 deg/sec^2 to define tool rotation acceleration when ROT is used in motion commands. If Decel is omitted, this value is used for both the Acceleration and Deceleration rates.
Decel	Optional. Single expression from 0.1 to 5000 deg/sec^2 to define tool rotation deceleration when ROT is used in motion commands.

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

AccelR Example

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

AccelS Method, Spel Class

Description
Sets acceleration and deceleration for linear interpolar (straight line) motion commands Jump3CP, Move, TMove.

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

Parameters

Accel	Single expression between 1-5000 represented in mm/sec^2 units to define acceleration and deceleration values for Straight Line and Continuous Path motion. If Decel is omitted, this value is used for both the Acceleration and Deceleration rates.
Decel	Single expression between 1-5000 represented in mm/sec^2 units to define deceleration values for Straight Line and Continuous Path motion. One parameter is used for representing both the Acceleration and Deceleration rates.
JumpDepartAccel	Single expression between 1-5000 representing a percentage of maximum acceleration rate for Jump3CP command Z Axis upward motion.
JumpDepartDecel	Single expression between 1-5000 representing a percentage of maximum deceleration rate for Jump3CP command Z Axis upward motion.
JumpApproAccel	Single expression between 1-5000 representing a percentage of maximum acceleration rate for Jump3CP command Z Axis downward motion.
JumpApproDecel	Single expression between 1-5000 representing a percentage of maximum deceleration rate for Jump3CP command Z Axis downward motion.

See Also
Accel, SpeedS, Jump3CP, Move, TMove

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

Agl Method, Spel Class

Description
Returns the joint angle for the selected rotational axis, or position for the selected linear axis.

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

Parameters
JointNumber Integer expression from 1-9 representing the joint number.

See Also
Pls, CX - CT

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

Arc Method, Spel Class

Description
Arc moves the arm to the specified point using circular interpolation in the XY plane.

Syntax
Sub **Arc** (*MidPoint* As Integer, *EndPoint* As Integer)
Sub **Arc** (*MidPoint* As SpelPoint, *EndPoint* As SpelPoint)
Sub **Arc** (*MidPoint* As String, *EndPoint* As String)

Parameters
Each syntax has two parameters that specify the mid point and end point of the arc.

MidPoint Specifies the mid point by using an integer, SpelPoint or string expression.

EndPoint Specifies the end point by using an integer, SpelPoint or string expression. When using a string expression, you can include ROT, CP, SYNC, a search expression for Till, and a parallel processing statement.

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

Arc Example
```
' Points specified using SpelPoint
Dim midPoint, endPoint As SpelPoint
midPoint = m_spel.GetPoint("P1")
endPoint = m_spel.GetPoint("P2")
m_spel.Arc(midPoint, endPoint)


' Points specified using expressions
m_spel.Arc("P1", "P2")
m_spel.Arc("P1", "P2 CP")

' Using parallel processing
m_spel.Arc("P1", "P2 !D50; On 1; D90; Off 1!")
```

Arc3 Method, Spel Class

Description

Arc3 moves the arm to the specified point using circular interpolation in 3 dimensions.

Syntax

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

Parameters

Each syntax has two parameters that specify the mid point and end point of the arc.

MidPoint	Specifies the mid point by using an integer, SpelPoint or string expression.
EndPoint	Specifies the end point by using an integer, SpelPoint or string expression. When using a string expression, you can include ROT, ECP, CP, SYNC, a search expression for Till, and a parallel processing statement.

See Also

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

Arc3 Example

```
' Points specified using SpelPoint
Dim midPoint, endPoint As SpelPoint
midPoint = m_spel.GetPoint("P1")
endPoint = m_spel.GetPoint("P2")
m_spel.Arc3(midPoint, endPoint)

' Points specified using expressions
m_spel.Arc3("P1", "P2")
m_spel.Arc3("P1", "P2 CP")

' Using parallel processing
m_spel.Arc3("P1", "P2 !D50; On 1; D90; Off 1!")
```

Arch Method, Spel Class

Description
Defines ARCH parameters (Z height to move before beginning horizontal motion) for use with the JUMP instructions.

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

Parameters

ArchNumber	The Arch number to define. Valid Arch numbers are (0-6) making a total of 7 entries into the Arch table.
DepartDist	The depart distance in millimeters moved at the beginning of the Jump instruction before starting horizontal motion.
ApproDist	The approach distance in millimeters above the target position of the Jump instruction.

See Also
Jump, Jump3, Jump3CP

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

Arm Method, Spel Class

Description
Selects the current robot arm.

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

Parameters

ArmNumber	Integer expression from 0-15. The user may select up to 16 different arms. Arm 0 is the standard (default) robot arm. Arm(s) 1-15 are auxiliary arms defined by the ArmSet instruction.

See Also
ArmSet, GetArm, Tool

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

ArmClr Method, Spel Class

Description
Clears (undefines) an arm for the current robot.

Syntax
Sub **ArmClr** (*ArmNumber* As Integer)

Parameters

ArmNumber	Integer expression from 1-15. Arm 0 is the standard (default) robot arm and cannot be cleared. Arm(s) 1-15 are auxiliary arms defined by the ArmSet instruction.

See Also
ArmSet, GetArm, Tool

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

ArmDef Method, Spel Class

Description
Returns whether a robot arm is defined or not.

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

Parameters

ArmNumber	Integer expression from 1-15. Arm 0 is the standard (default) robot arm and is always defined. Arm(s) 1-15 are auxiliary arms defined by using the ArmSet method.

Return Value
True if the specified arm is defined, False if not.

See Also
ArmSet, GetArm, Tool

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

ArmSet Method, Spel Class

Description
Defines an auxiliary robot arm.

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

Parameters

ArmNumber Integer number: Valid range from 1-15.

Param1 (For SCARA Robots) The horizontal distance from the center line of the elbow joint to the center line of the new orientation axis. (I.E. the position where the new auxiliary arm's orientation axis center line is located.)
(For Cartesian Robots) X axis direction position offset from the original X position specified in mm.

Param2 (For SCARA Robots) The offset (in degrees) between the line formed between the normal Elbow center line and the normal orientation Axis center line and the line formed between the new auxiliary arm elbow center line and the new orientation axis center line. (These 2 lines should intersect at the elbow center line and the angle formed is the *Param2*.)
(For Cartesian Robots) Y axis direction position offset from the original Y position specified in mm.

Param3 (For SCARA & Cartesian Robots) The Z height offset difference between the new orientation axis center and the old orientation axis center. (This is a distance.)

Param4 (For SCARA Robots) The distance from the shoulder center line to the elbow center line of the elbow orientation of the new auxiliary axis.
(For Cartesian Robots) This is a dummy parameter (Specify 0)

Param5 (For SCARA & Cartesian Robots) The angular offset (in degrees) for the new orientation axis vs. the old orientation axis.

See Also
Arm, Tool, TLSet

ArmSet Example
```
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
```

Atan Method, Spel Class

Description
Returns the arc tangent of a numeric expression.

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

Parameters
number Numeric expression representing the tangent of an angular value.

Return Value
Arc tangent of the specified value

See Also
Atan2

Atan Example
```
Dim angle As Double

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

Atan2 Method, Spel Class

Description
Returns the angle of the imaginary line connecting points (0,0) and (X, Y) in radians.

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

Parameters
Dx Numeric expression representing the X coordinate.

Dy Numeric expression representing the Y coordinate.

Return value
A double value containing the angle.

See Also
Atan

Atan2 Example
```
Dim angle As Double

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

AtHome Method, Spel Class

Description
Returns True if the current robot is at the home position.

Syntax
Function **AtHome** () As Boolean

Return Value
True if the current robot is at it's home position, False if not.

See Also
Home

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

AxisLocked Method, Spel Class

Description
Returns True if specified axis is under servo control.

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

Parameters
AxisNumber Numeric expression representing the axis number. The value can be from 1 – 9.

Return Value
True if the specified axis is under servo control.

See Also
SLock, SFree

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

AvoidSingularity Method, Spel Class

Description
Enables / disables the singularity avoidance feature for Move, Arc, and Arc3 motion methods.

Syntax
Sub **AvoidSingularity**(*Enable* As Boolean)

Parameters
Enable True enables singularity avoidance and False disables it.

See Also
SingularityAngle, SingularitySpeed

AvoidSingularity Example
```
With m_spel
    .AvoidSingularity(True)
    .Move(1)
End With
```

BoxClr Method, Spel Class

Description
Clears the definition of a box (approach check area).

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

Parameters
BoxNumber Integer expression representing the area number from 1 to 15.

See Also
Box, BoxDef

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

Base Method, Spel Class

Description
Defines the base coordinate system.

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

Parameters

OriginPoint	A SpelPoint representing the origin of the base coordinate system.
XAxisPoint	Optional. A SpelPoint located anywhere on the X axis of the base coordinate system.
YAxisPoint	Optional. A SpelPoint located anywhere on the Y axis of the base coordinate system.
Alignment	Optional. When supplying the *XAxisPoint* and *YAxisPoint* parameters, use the Alignment parameter to specify which axis to align the base with.

See Also
Local

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

BGo Method, Spel Class

Description
Executes Point to Point relative motion in the selected local coordinate system.

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

Parameters
Each syntax has one parameter that specifies the end point which the arm travels to during the BGo motion. This is the final position at the end of the point to point motion.

PointNumber	Specifies the end point by using the point number for a previously taught point in the controller's point memory for the current robot.
Point	Specifies the end point by using a SpelPoint data type.
AttribExpr	Specifies the end point attributes by using a string expression.
PointExpr	Specifies the end point by using a string expression.

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

BGo Example
```
' Using a point number
m_spel.Tool(1)
m_spel.BGo(100)

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

' Using an attribute expression
m_spel.BGo(pt, "ROT")

' Using a point expression
m_spel.BGo("P0 /L /2")
m_spel.BGo("P1 :Z(-20)")

' Using a parallel processing
m_spel.BGo("P1 !D50; On 1; D90; Off 1!")

' Using point label
m_spel.BGo("pick")
```

BMove Method, Spel Class

Description

Executes linear interpolated relative motion in the selected local coordinate system

Syntax

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

Parameters

Each syntax has one parameter that specifies the end point which the arm travels to during the BMove motion. This is the final position at the end of the linear interpolated motion.

PointNumber	Specifies the end point by using the point number for a previously taught point in the controller's point memory for the current robot.
Point	Specifies the end point by using a SpelPoint data type.
AttribExpr	Specifies the end point attributes by using a string expression.
PointExpr	Specifies the end point by using a string expression.

See Also

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

BMove Example

```
' Using a point number
m_spel.Tool(1)
m_spel.BMove(100)

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

' Using a point expression
m_spel.BMove("P0 /L /2")

' Using a parallel processing
m_spel.BMove("P1 !D50; On 1; D90; Off 1!")

' Using point label
m_spel.BMove("pick")
```

Box Method, Spel Class

Description
Specifies an approach check area defined within a box.

Syntax
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)

Parameters

AreaNumber	Integer number from 1-15 representing which of the 15 boxes to define.
MinX	The minimum X coordinate position of the approach check area.
MaxX	The maximum X coordinate position of the approach check area.
MinY	The minimum Y coordinate position of the approach check area.
MaxY	The maximum Y coordinate position of the approach check area.
MinZ	The minimum Z coordinate position of the approach check area.
MaxZ	The maximum Z coordinate position of the approach check area.
PolarityOn	Optional. Sets the remote output logic when the corresponding remote output is used. To set I/O output to on when the end effector is in the box area, use True. To set I/O output to off when the end effector is in the box area, use False.

See Also
BoxClr, BoxDef, Plane

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

BoxDef Method, Spel Class

Description
Returns whether Box has been defined or not.

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

Parameters
BoxNumber Integer expression representing the area number from 1 to 15.

Return Value
True if the specified box is defined, False if not.

See Also
Box, BoxClr

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

BTst Method, Spel Class

Description
Returns the status of 1 bit in a number.

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

Parameters

Number	Specifies the number for the bit test with an expression or numeric value.
BitNumber	Specifies the bit (integer from 0 to 31) to be tested.

Return Value
True if the specified bit is set, False if not.

See Also
On, Off

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

BuildProject Method, Spel Class

Description
Builds the EPSON RC+ 7.0 project specified by the Project property.

Syntax
Sub **BuildProject** ()

See Also
Project, ProjectBuildComplete

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

Call Method, Spel Class

Description
Calls (executes) a SPEL+ function which can optionally return a value.

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

Parameters
FuncName The name of a function which has already been defined in the current project.

Parameters Optional. A string expression containing the parameters for the call.

Return Value
The return value of the SPEL+ function. The data type matches the the data type of the function.

Remarks
Use the Call method to call a SPEL+ function and retrieve the return value. When assigning the result of Call to a variable, ensure that the correct data type is used, otherwise a type mismatch error will occur.

You can also call DLL functions declared in your SPEL+ code from your Visual Basic application.

See Also
Project, Xqt

Call Example
```
' Visual Basic Code
Dim errCode As Integer
errCode = m_spel.Call("GetPart")

' SPEL+ function
Function GetPart As Integer
    Long errNum
    OnErr GPErr
    errNum = 0
    Jump P1
    On vacuum
    Wait SW(vacOn) = 1, 2
    If TW(0) = 1 Then
        errNum = VAC_TIMEOUT
    EndIf
GPExit:
    GetPart = errNum
    Exit Function
GPErr:
    errNum = Err
    GoTo GPExit
Fend
```

ClearPoints Method, Spel Class

Description
Clears the points in memory for the current robot.

Syntax
Sub **ClearPoints** ()

See Also
LoadPoints, Robot, SavePoints, SetPoint

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

Connect Method, Spel Class

Description
Connects the Spel class instance with a controller.

Syntax
Sub **Connect** (*ConnectionNumber* As Integer)

Parameters

ConnectionNumber	Integer expression for the connection number. This currently must be set to 1.

Remarks
When a Spel class instance needs to communicate with the controller, it automatically connects. If you want to explicitly connect to the controller, use the Connect method.

See Also
Disconnect, Initialize

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

Continue Method, Spel Class

Description
Causes all tasks in the controller to resume if a pause has occurred.

Syntax
Sub **Continue** ()

Remarks
Use **Continue** to resume all tasks that have been paused by the Pause method or by safeguard open.

When the safeguard is open while tasks are running, the robot will decelerate to a stop and the robot motors will be turned off. After the safeguard has been closed, you can use **Continue** to resume the cycle.

See Also
Pause, Start, Stop

Continue Example
```
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
```

Ctr Method, Spel Class

Description
Returns the counter value of the specified input counter.

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

Parameters
BitNumber Number of the input bit set as a counter. Only 16 counters can be active at the same time.

Return Value
Returns the counter value.(integer from 0 to 65535)

See Also
CtReset

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

CtReset Method, Spel Class

Description
Resets the counter value of the specified input counter. Also defines the input as a counter Input.

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

Parameters

BitNumber	Number of the input bit set as a counter. Only 16 counters can be active at the same time.

See Also
Ctr

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

Curve Method, Spel Class

Description
Defines the data and points required to move the arm along a curved path. Many data points can be defined in the path to improve precision of the path.

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

Parameters
FileName A string expression for the path and name of the file in which the point data is stored. The specified *fileName* will have the extension CRV appended to the end so no extension is to be specified by the user. When the **Curve** instruction is executed, *fileName* will be created.

Closure A Boolean expression that specifies whether to connect the last point of the path to the first point.

Mode Specifies whether or not the arm is automatically interpolated in the tangential direction of the U-Axis.

Mode Setting	Tangental Correction
0	No
2	Yes

NumOfAxis Integer expression between 2 - 4 which specifies the number of Axes controlled during the curved motion as follows:

2 - Generate a curve in the XY plane with no Z Axis movement or U Axis rotation.

3 - Generate a curve in the XYZ plane with no U axis rotation. (Theta 1, Theta2, and Z)

4 - Generate a curve in the XYZ plane with U-Axis rotation. (Controls all 4 Axes)

PointList { point expression | P(*start***:***finish*) } [, *output command*] ...
This parameter is actually a series of Point Numbers and optional output statements either separated by commas or an ascended range of points separated by a colon. Normally the series of points are separated by commas as shown below:
`Curve MyFile, O, 0, 4, P1, P2, P3, P4`

Remarks
Use Curve to define a spline path to be executed with the CVMove method. See the SPEL+ command Curve for more details.

See Also
Curve (SPEL+ Statement), CVMove Method

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

CVMove Method, Spel Class

Description
Performs the continuous spline path motion defined by the **Curve** instruction.

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

Parameters
FileName String expression for the path and name of the file to use for the continuous path motion data. This file must be previously created by the Curve instruction.

OptionList Optional. String expression containing Till specification.

Remarks
Use CVMove to execute a path defined with the Curve method. See the SPEL+ command **CVMove** for more details.
If you need to execute CVMove with CP, it is recommended that you execute CVMove from a SPEL+ task rather than from your application. The reason for this is that for CP motion to perform properly, the system needs to know ahead of time where the next motion target is. Since the RC+ API commands are executed one at a time, the system does not know ahead of time where the next target is.

See Also
Curve, CVMove (SPEL+ Command)

CVMove Example

```
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 Methods, Spel Class

Description
Retrieves a coordinate value from a point

CV and CW are for the 6-axis robot
CS and CT are for the additional axis
CR is for the Joint 7-axis robot

Syntax
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

Parameters
PointExpr A string expression specifying the point from which to retrieve the specified coordinate. Any valid point expression can be used. P* can also be used to retrieve the coordinate from the current position.

Return Value
The specified coordinate value.
Return value of CX, CY, CZ : Real value (mm)
Return value of CU, CV, CW : Real value (deg)
Return value of CR, CS, CT : Real value

See Also
GetPoint, SetPoint

CX, CY, CZ, CU, CV, CW, CR, CS, CT Example
```
Dim x As Single, y As Single
x = m_spel.CX("P1")
y = m_spel.CY("P*")
```

Delay Method, Spel Class

Description
Delays for a specified number of milliseconds.

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

Parameters
Milliseconds Integer value containing the number of milliseconds to delay.

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

DegToRad Method, Spel Class

Description
Converts Degrees into Radians.

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

Parameters
degrees The number of degrees to convert into Radians.

Return value
A double value containing radians.

See Also
RadToDeg

DegToRad Example
```
Dim rad As Double

rad = m_spel.DegToRad(45)
```

Disconnect Method, Spel Class

Description
Disconnects the Spel class instance from the current connection.

Syntax
Sub **Disconnect** ()

Remarks
Use **Disconnect** to disconnect from the current controller connection.

See Also
Connect, Initialize

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

ECP Method, Spel Class

Description
Selects the current ECP definition.

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

Parameters

ECPNumber	Integer number from 0-15 representing which of 16 ECP definitions to use with the next motion instructions.

See Also
ECPSet

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

ECPClr Method, Spel Class

Description
Clears (undefines) an external control point for the current robot.

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

Parameters

ECPNumber	Integer expression representing which one of the 15 external control points to clear (undefine). (ECP 0 is the default and cannot be cleared.)

See Also
ECP, ECPDef

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

ECPDef Method, Spel Class

Description
Returns ECP definition status.

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

Parameters
ECPNumber Integer value representing which ECP to return status for.

Return Value
True if the specified ECP is defined, False if not.

See Also
ECP, ECPClr

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

ECPSet Method, Spel Class

Description
Defines an ECP (external control point).

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

Parameters

ECPNumber	Integer number from 1-15 representing which of 15 external control points to define.
XCoord	The external control point X coordinate.
YCoord	The external control point Y coordinate.
ZCoord	The external control point Z coordinate.
UCoord	The external control point U coordinate.
VCoord	Optional. The external control point V coordinate.
WCoord	Optional. The external control point W coordinate.

See Also
ArmSet, ECP, GetECP, TLSet

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

EnableEvent Method, Spel Class

Description
Enables certain system events for the EventReceived event.

Syntax
Sub **EnableEvent** (*Event* As SpelEvents, *Enabled* as Boolean)

Parameters

Event	The event to enable or disable.
Enabled	Set to True to enable the event and False to disable it.

See Also
EventReceived

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

ExecuteCommand Method, Spel Class

Description
Sends a command to EPSON RC+ 7.0 and waits for it to complete

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

Parameters

Command String containing SPEL+ command.

Reply Optional reply returned.

Remarks
Normally, **ExecuteCommand** is not required. Most operations can be performed by executing Spel methods. However, sometimes it is desirable to execute SPEL+ multi-statements. Multi-statements are one line commands that contain more than one statement separated by semi-colons. Use **ExecuteCommand** to execute multi-statements. For example:

```
m_spel.ExecuteCommand("JUMP pick; ON tipvac")
```

The maximum command line length is 200 characters.

See Also
Pause

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

Fine Method, Spel Class

Description
Specifies and displays the positioning accuracy for target points.

Syntax
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])

Parameters

J1MaxErr – J9MaxErr	Integer number ranging from (0-32767) which represents the allowable positioning error for the each joint. The values for joints 7, 8, and 9 are optional.

See Also
Weight

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

Force_Calibrate Method, Spel Class

Description

Sets zero offsets for all axes for the current force sensor.

Syntax

Sub **Force_Calibrate( )**

Remarks

You should call Force_Calibrate for each sensor when your application starts. This will account for the weight of the components mounted on the sensor.

See Also

Force_Sensor, Force_GetForces, Force_SetTrigger

Force_Calibrate Example

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

Force_ClearTrigger Method, Spel Class

Description

Clears all trigger conditions for the current force sensor.

Syntax

Sub **Force_ClearTrigger( )**

Remarks

Use Force_ClearTrigger to clear all conditions for the current force sensor's trigger.

See Also

Force_Sensor, Force_GetForces, Force_SetTrigger

Force_ClearTrigger Example

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

Force_GetForce Method, Spel Class

Description

Returns the force for a specified force sensor axis.

Syntax

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

Parameters

Axis The axis value to retrieve, as shown below:

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

Remarks

Use Force_GetForce to read the current force setting for one axis. The units are determined by the force sensor configuration.

See Also

Force_Sensor, Force_GetForces, Force_SetTrigger

Force_GetForce Example

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

Force_GetForces Method, Spel Class

Description

Returns the forces and torques for all force sensor axes in an array.

Syntax

Sub **Force_GetForces(** *Values()* As Single**)**

Parameters

Values	Single array that will be returned with 6 elements.

Remarks

Use Force_GetForces to read all force and torque values at once.

See Also

Force_Sensor, Force_GetForces, Force_SetTrigger

Force_GetForces Example

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

Force_SetTrigger Method, Spel Class

Description

Sets the force trigger for the Till command.

Syntax

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

Parameters

Axis The axis to use for the trigger, as shown below:

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

Threshold Single expression representing the threshold value.

CompareType LessOrEqual, or GreatorOrEqual.

Remarks

To stop motion with a force sensor, you must set the trigger for the sensor, then use Till Force in your motion statement.

You can set the trigger with multiple axes. Call Force_SetTrigger for each axis.

To clear all trigger conditions, use Force_ClearTrigger.

See Also

Force_ClearTrigger, Force_Sensor, Till

Force_SetTrigger Example

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

GetAccel Method, Spel Class

Description
Returns specified acceleration/deceleration value.

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

Parameters

ParamNumber Integer expression which can have the following values:
1: acceleration specification value
2: deceleration specification value
3: depart acceleration specification value for Jump
4: depart deceleration specification value for Jump
5: approach acceleration specification value for Jump
6: approach deceleration specification value for Jump

Return Value

Integer containing the specified acceleration/deceleration value.

See Also
Accel

GetAccel Example

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

GetArm Method, Spel Class

Description
Returns the current Arm number for the current robot.

Syntax
Function **GetArm** () As Integer

Return Value
Integer containing the current arm number.

See Also
Arm, ArmSet, Robot, Tool

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

GetConnectionInfo Method, Spel Class

Description
Returns information about the controller connections.

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

Return Value
An array of SpelConnectionInfo.

See Also
GetControllerInfo

Remarks
GetConnectionInfo returns an array of SpelConnectionInfo. The connection information is configured in EPSON RC+ from the [Setup]-[PC to Controller Communication] dialog.

GetConnectionInfo Example
```
Dim info() As SpelConnectionInfo

info = m_spel.GetConnectionInfo()
```

GetControllerInfo Method, Spel Class

Description
Returns information about the current controller.

Syntax
Function **GetControllerInfo**() As SpelControllerInfo

Return Value
A SpelControllerInfo instance.

See Also
GetErrorMessage

Remarks
GetControllerInfo returns a new instance of the SpelControllerInfo class, which contains controller information properties.

GetControllerInfo Example
```
Dim info As SpelControllerInfo
Dim msg As String

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

GetCurrentUser Method, Spel Class

Description
Returns the current EPSON RC+ 7.0 user.

Syntax
Function **GetCurrentUser** () As String

Return Value
String variable containing the current user.

See Also
Login

GetCurrentUser Example
```
Dim currentUser As String

currentUser = m_spel.GetCurrentUser()
```

GetECP Method, Spel Class

Description
Returns the current ECP number for the current robot.

Syntax
Function **GetECP** () As Integer

Return Value
Integer containing the current ECP number.

See Also
ECP, ECPSet

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

GetErrorMessage Method, Spel Class

Description
Returns the error message for the specified error or warning code.

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

Parameters
ErrorCode The error code for which to return the associated error message.

Return Value
String containing the error message.

See Also
ErrorCode

GetErrorMessage Example
```
Dim msg As String

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

GetIODef Method, Spel Class

Description

Gets the definition information for an input, output, or memory I/O bit, byte, or word.

Syntax

Sub **GetIODef**(*Type* As SpelIOLabelTypes, *Index* As Integer, ByRef *Label* as String, ByRef *Description* As String)

Parameters

Type	Specifies the I/O type as shown below: InputBit = 1, InputByte = 2, InputWord = 3 OutputBit = 4, OutputByte = 5, OutputWord = 6, MemoryBit = 7, MemoryByte = 8, MemoryWord = 9
Index	Specifies the bit or port number.
Label	Returns the label.
Description	Returns the description.

Return Value

The values are returned in the Label and Description parameters.

Remarks

Use GetIODef to get the labels and descriptions used for all I/O in the current project.

See Also

SetIODef

GetIODef Example

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

GetLimitTorque Method, Spel Class

Description
Returns the limit torque for the specified joint for the current robot.

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

Parameters

JointNumber Integer expression for the desired joint.

Return Value
Integer value between 1 and 9 which represents the limit torque setting for the specified joint.

See Also
GetRealTorque, GetRobotPos, LimitTorque

GetLimitTorque Example

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

GetLimZ Method, Spel Class

Description
Returns the current LimZ setting.

Syntax
Function **GetLimZ** () As Single

Return Value

Real value containing the LimZ value.

See Also
LimZ, Jump

GetLimZ Example

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

GetPoint Method, Spel Class

Description
Retrieves coordinate data for a robot point.

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

Parameters

PointNumber	Integer expression for a point in the controller's point memory for the current robot.
PointName	String expression. This can be a point label, "Pxxx", "P*" or "*".

See Also
SetPoint

GetPoint Example

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

GetRealTorque Method, Spel Class

Description
Returns the torque for the specified joint for the current robot.

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

Parameters
JointNumber Integer expression for the desired joint.

Return Value
Double value between 0 and 1 which represents the portion of maximum torque for the current power mode and for the specified joint.

See Also
GetLimitTorque, GetRobotPos

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

GetRobotPos Method, Spel Class

Description

Returns the current robot position.

Syntax

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

Parameters

PosType	Specifies the type of position data to return.
Arm	Integer expression that specifies the robot arm.
Tool	Integer expression that specifies the robot tool.
Local	Integer expression that specifies the robot local.

Returns

Single data type array containing 9 elements. The data returned depends on the specified *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

See Also

GetPoint

GetRobotPos Example

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

GetSpeed Method, Spel Class

Description
Returns one of the three speed settings for the current robot.

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

Parameters

ParamNumber	Integer expression which evaluates to one of the values shown below. 1: PTP motion speed 2: Jump depart speed 3: Jump approach speed

See Also
Speed

GetSpeed Example

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

GetTool Method, Spel Class

Description
Returns the current Tool number for the current robot.

Syntax
Function **GetTool** () As Integer

Return Value
Integer containing the current tool number.

See Also
Arm, TLSet, Tool

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

GetVar Method, Spel Class

Description
Returns the value of a SPEL+ global preserve variable in the controller.

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

Parameters
VarName The name of the SPEL+ global preserve variable. For an array, the entire array can be returned or just one element.

Return Value
Returns the value whose data type is determined by the type of the SPEL+ variable.

Remarks
You can use GetVar to retrieve values of any global preserve variables in the controller's current project. Before you can retrieve values, the project must be successfully built.

If you want to retrieve an entire array, then supply the array name in *VarName*. To retrieve one element of an array, supply the subscript in *VarName*.

See Also
SetVar

GetVar Example
In the SPEL+ project, the variable is declared:

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

In the Visual Basic project:

Since g_myIntVar is declared as in integer, the Visual Basic variable used to retrieve the value of g_myInVar must be declared as an Integer. For g_myRealArray, the Visual Basic variable must be declared as a Single array.

```
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$")
```

Go Method, Spel Class

Description
Moves the arm in a Point to Point fashion from the current position to the specified point or XY position. The **GO** instruction can move any combination of the robot axes at the same time.

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

Parameters
Each syntax has one parameter that specifies the end point which the arm travels to during the Go motion. This is the final position at the end of the point to point motion.

PointNumber	Specifies the end point by using the point number for a previously taught point in the controller's point memory for the current robot.
Point	Specifies the end point by using a SpelPoint data type.
AttribExpr	Specifies the end point attributes by using a string expression.
PointExpr	Specifies the end point by using a string expression.

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

Go Example
```
' Point specified using point number
m_spel.Tool(1)
m_spel.Go(100)

' Point specified using SpelPoint
Dim pt As SpelPoint
pt = m_spel.GetPoint("P*")
pt.X = 125.5
m_spel.Go(pt)

' Point specified using expression
m_spel.Go("P0 /L /2")
m_spel.Go("P1 :Z(-20)")

' Using parallel processing
m_spel.Go("P1 !D50; On 1; D90; Off 1!")

' Point specified using label
m_spel.Go("pick")
```

Halt Method, Spel Class

Description
Suspends execution of the specified task.

Syntax
Sub **Halt** (*TaskNumber* As Integer)
Sub **Halt** (*TaskName* As String)

Parameters
TaskNumber	The task number of the task to be suspended. The range of the task number is 1 to 32.
TaskName	A string expression containing the name of the task to be suspended.

See Also
Resume, Xqt

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

Here Method, Spel Class

Description
Teaches a point at the current position.

Syntax
Sub **Here** (*PointNumber* As Integer)
Sub **Here** (*PointName* As String)

Parameters

PointNumber	Integer expression for a point in the point memory for the current robot. Any valid point number can be used starting with 0.
PointName	A string expression for a point label.

See Also
SetPoint

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

HideWindow Method, Spel Class

Description

Hides an EPSON RC+ 7.0 window that was previously displayed with ShowWindow.

Syntax

Sub **HideWindow** (*WindowID* As SpelWindows, *Parent* As Form)

Parameters

WindowID The ID of the EPSON RC+ 7.0 window to hide.

See Also

RunDialog, ShowWindow

HideWindow Example

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

    m_spel.HideWindow(RCAPINet.SpelWindows.IOMonitor)
End Sub
```

Home Method, Spel Class

Description
Moves the robot arm to the user defined home position that is set with the HomeSet method.

Syntax
Sub **Home** ()

See Also
HomeSet, MCal

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

HomeSet Method, Spel Class

Description
Specifies the position used by the Home method.

Syntax
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])

Parameters

J1Pulses – J9Pulses	The Home position encoder pulse value for each joint. Joints 7, 8, and 9 are optional.

See Also
Home, MCal

HomeSet Example
```
' Set the home position at the current position
With m_spel
    .HomeSet(.Pls(1), .Pls(2), .Pls(3), .Pls(4), 0, 0)
End With
```

Hordr Method, Spel Class

Description
Specifies the order of the axes returning to their HOME positions.

Syntax
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])

Parameters

Step 1 - 9	Bit pattern that tells which axes should home during each step of the Home process. Any number of axes between 0 to all axes may home during the 1st step. Steps 7 – 9 are optional and are used with robots that have more than 6 axes.

See Also
Home, HomeSet, Mcordr

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

Hour Method, Spel Class

Description
Returns the accumulated system operating time in hours.

Syntax
Function **Hour** () As Single

Return Value
Integer expression representing time.

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

ImportPoints Method, Spel Class

Description
Imports a point file into the current project for the current robot.

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

Parameters

SourcePath	String expression containing the specific path and file to import into the current project. The extension must be .PTS.
ProjectFileName	String expression containing the specific file to be imported to in the current project for the current robot or specified robot if *RobotNumber* is supplied. The extension must be .PTS.
RobotNumber	Optional. Integer expression for the robot that the point file will be used for. Specify 0 to make it a common point file.

See Also
SavePoints

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

In Method, Spel Class

Description
Returns the status of the specified input port. Each port contains 8 input bits (one byte).

Syntax
Function **In** (*PortNumber* As Integer) As Integer
Function **In** (*Label* As String) As Integer

Parameters

PortNumber	Integer expression representing one of the input ports. Each port contains 8 input bits (one byte).
Label	String expression containing an input byte label.

Return Value
Integer from 0 to 255 representing the status of the input port.

See Also
InBCD, Out, OpBCD, Sw

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

InBCD Method, Spel Class

Description
Returns the input status of 8 inputs using BCD format. (Binary Coded Decimal)

Syntax
Function **InBCD** (*PortNumber* As Integer) As Integer
Function **InBCD** (*Label* As String) As Integer

Parameters

PortNumber	Integer expression representing one of the input ports.
Label	String expression containing an input byte label.

Return Value
Integer from 0 to 9 representing the status of the input port.

See Also
In, Out, OpBCD, Sw

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

Inertia Method, Spel Class

Description
Specifies the load inertia and eccentricity for the current robot.

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

Parameters

LoadInertia	Real expression that specifies total moment of inertia in kgm^2 around the center of the end effector joint, including end effector and part.
Eccentricity	Real expression that specifies eccentricity in mm around the center of the end effector joint, including end effector and part.

See Also
Weight

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

Initialize Method, Spel Class

Description
Initializes the Spel class instance.

Syntax
Sub **Initialize** ()

Remarks
Normally, the Spel class instance is automatically initialized when the first method has been executed. Initialization can take several seconds as EPSON RC+ 7.0 loads into memory. So in some cases, you may want to call initialize first in your application during startup.

See Also
Connect, Disconnect

Initialize Example
```
m_spel.Initiialize()
```

InsideBox Method, Spel Class

Description
Returns the check status of the approach check area.

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

Parameters

BoxNumber	Integer expression from 1 to 15 representing which approach check area to return status for.

Return Value
True if the robot end effector is inside the specified box, False if not.

See Also
Box, InsidePlane

InsideBox Example
```
x = m_spel.InsideBox(1)
```

InsidePlane Method, Spel Class

Description

Returns the check status of the approach check plane.

Syntax

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

Parameters

PlaneNumber	Integer expression from 1 to 15 representing which approach check plane to return status for.

Return Value

True if the robot end effector is inside the specified box, False if not.

See Also

InsideBox, Plane

InsidePlane Example

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

InW Method, Spel Class

Description
Returns the status of the specified input word port. Each word port contains 16 input bits.

Syntax
Function **InW** (*PortNumber* As Integer) As Integer
Function **InW** (*Label* As String) As Integer

Parameters
PortNumber	Integer number representing an input port.
Label	String expression containing an input word label.

Return Value
Integer value from 0 to 65535 representing the input port

See Also
In, InBCD, Out, OpBCD, Sw

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

JRange Method, Spel Class

Description
Defines the permissible working range of the specified robot joint in pulses.

Syntax
Sub **JRange** (*JointNumber* As Integer, *LowerLimitPulses* As Integer, *UpperLimitPulses* As Integer)

Parameters

JointNumber	Integer number between 1 - 9 representing the joint for which JRange will be specified.
LowerLimitPulses	Integer number representing the encoder pulse count position for the lower limit range of the specified joint.
UpperLimitPulses	Integer number representing the encoder pulse count position for the upper limit range of the specified joint

See Also
XYLim

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

JS Method, Spel Class

Description
Jump Sense detects whether the arm stopped prior to completing a JUMP instruction (which used a SENSE input) or if the arm completed the JUMP move.

Syntax
Function **JS** () As Boolean

Return Value
True if the SENSE input was detected during motion, False if not.

See Also
JT, Jump, Jump3, Jump3CP, Sense, Till

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

JT Method, Spel Class

Description
Returns the status of the most recent Jump, Jump3, or Jump3CP instruction for the current robot.

Syntax
Function **JT** () As Integer

Return Value
JT returns an integer with the following bits set or cleared:

Bit 0 Set to 1 when rising motion has started or rising distance is 0.

Bit 1 Set to 1 when horizontal motion has started or horizontal distance is 0.

Bit 2 Set to 1 when descent motion has started or descent distance is 0.

Bit 16 Set to 1 when rising motion has completed or rising distance is 0.

Bit 17 Set to 1 when horizontal motion has completed or horizontal distance is 0.

Bit 18 Set to 1 when descent motion has completed or descent distance is 0.

See Also
JS, Jump, Jump3, Jump3CP, Sense, Till

JT Example
```
Dim status As Integer
With m_spel
    .Till("Sw(1) = On")
    .Jump("P1 Till")
    If .JT() And 4 = 4 Then
        MessageBox.Show("Motion stopped during decent")
    EndIf
End With
```

JTran Method, Spel Class

Description
Executes a relative joint move.

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

Parameters

JointNumber	The specific joint to move.
Distance	The distance to move. Units are in degrees for rotary joints and millimeters for linear joints.

See Also
PTran, Pulse

JTran Example
```
' Move joint 1 45 degrees in the plus direction.
m_spel.JTran(1, 45.0)
```

Jump Method, Spel Class

Description
Moves the arm from the current position to the specified point using point to point motion while first moving in a vertical direction up, then horizontally and then finally vertically downward to arrive on the final destination point.

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

Parameters
Each syntax has one parameter that specifies the end point which the arm travels to during the Jump motion. This is the final position at the end of the point to point motion.

PointNumber	Specifies the end point by using the point number for a previously taught point in the controller's point memory for the current robot.
Point	Specifies the end point by using a SpelPoint data type.
AttribExpr	Specifies the end point attributes by using a string expression.
PointExpr	Specifies the end point by using a string expression.

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

Jump Example
```
' Point specified using point number
m_spel.Tool(1)
m_spel.Jump(100)

' Point specified using SpelPoint
Dim pt As SpelPoint
pt = m_spel.GetPoint("P*")
pt.X = 125.5
m_spel.Jump(pt)

' Point specified using expression
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")

' Using parallel processing
m_spel.Jump("P1 !D50; On 1; D90; Off 1!")

' Point specified using label
m_spel.Jump("pick")
```

Jump3 Method, Spel Class

Description
Motion with 3D gate using a combination of two CP motions and one PTP motion. The robot moves to the depart point, then the approach point, and finally the destination point.

Syntax
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)

Parameters

DepartPoint	The departure point above the current position using a point number or string point expression.
ApproPoint	The approach point above the destination position using a point number or string point expression.
DestPoint	The target destination of the motion using a point number or string point expression.

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

Jump3 Example

```
' Points specified using point numbers
m_spel.Tool(1)
m_spel.Jump3(1, 2, 3)

' Points specified using SpelPoint
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)

' Points specified using expressions
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")

' Using parallel processing
m_spel.Jump3("P1", "P2", "P3 !D50; On 1; D90; Off 1!")

' Points specified using labels
m_spel.Jump3("depart", "approach", "place")
```

Jump3CP Method, Spel Class

Description
Motion with 3D gate using a combination of three CP motions.

Syntax
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)

Parameters

DepartPoint	The departure point above the current position using a point number or string point expression.
ApproPoint	The approach point above the destination position using a point number or string point expression.
DestPoint	The target destination of the motion using a point number or string point expression.

See Also
AccelR, AccelS, SpeedR, SpeedS
Arc, Arc3, CVMove, Go, Jump, Jump3, Move
BGo, BMove, TGo, TMove
Arch, CP, Sense, Till

Jump3CP Example

```
' Points specified using point numbers
m_spel.Tool(1)
m_spel.Jump3CP(1, 2, 3)

' Points specified using SpelPoint
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)

' Points specified using expressions
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")

' Using parallel processing
m_spel.Jump3CP("P1", "P2", "P3 !D50; On 1; D90; Off 1!")

' Points specified using labels
m_spel.Jump3CP("depart", "approch", "place")
```

LimitTorque Method, Spel Class

Description
Sets the upper limit torque in high power mode for the current robot.

Syntax
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)

Parameters

AllJointsMax	Integer expression for the desired upper limit of torque for all joints in high power mode.
J1Max – J6Max	Integer expression for the desired upper limit of torque for each joint in high power mode.

Return Value
Integer value between 1 and 9 which represents the limit torque setting for the specified joint.

See Also
GetRealTorque, GetRobotPos, LimitTorque

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

LimZ Method, Spel Class

Description
Sets the default value of the Z axis height for JUMP commands.

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

Parameters

ZLimit	A coordinate value within the movable range of the Z axis.

See Also
Jump

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

LoadPoints Method, Spel Class

Description
Loads a SPEL+ point file into the controller's point memory for the current robot.

Syntax
Sub **LoadPoints** (*FileName* As String)

Parameters

FileName A valid point file in the current project.

See Also
ImportPoints, SavePoints

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

Local Method, Spel Class

Description
Defines local coordinate systems.

Syntax
Sub **Local** (*LocalNumber* As Integer, *OriginPoint* As SpelPoint, [*XAxisPoint* As SpelPoint], [*YAxisPoint* As SpelPoint])

Parameters

LocalNumber	The local coordinate system number. A total of 15 local coordinate systems (of the integer value from 1 to 15) may be defined.
OriginPoint	SpelPoint variable for the origin of the local coordinate system.
XAxisPoint	Optional. SpelPoint variable for a point along the X axis of the local coordinate system.
YAxisPoint	Optional. SpelPoint variable for a point along the Y axis of the local coordinate system.

See Also
Base

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

LocalClr Method, Spel Class

Description
Clears a Local defined for the current robot.

Syntax
Sub **LocalClr** (*LocalNumber* As Integer)

Parameters

LocalNumber	Integer expression representing which of 15 locals (integer from 1 to 15) to clear (undefine).

See Also
Local, LocalDef

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

LocalDef Method, Spel Class

Description
Returns local definition status.

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

Parameters
LocalNumber Integer expression representing which local coordinate to return status for.

Return Value
True if the specified local is defined, False if not.

See Also
Local, LocalClr

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

Login Method, Spel Class

Description
Log into EPSON RC+ 7.0 as another user.

Syntax
Sub **Login** (*LoginID* As String, *Password* As String)

Parameters

LoginID	String expression containing user login ID.
Password	String expression containing user password.

Remarks
You can utilize EPSON RC+ 7.0 security in your application. For example, you can display a menu that allows different users to log into the system. Each type of user can have its own security rights. For more details on security, see the EPSON RC+ 7.0 User's Guide.

If security is enabled and you do not execute LogIn, then your Visual Basic application will be logged in as the guest user. Or if Auto LogIn is enabled in EPSON RC+ 7.0, your application will automatically be logged in as the current Windows user if such a user has been configured in EPSON RC+ 7.0.

See Also
GetCurrentUser

Login Example
```
With m_spel
    .Project = "c:\EpsonRC70\projects\myproject\myproject.sprj"
    .LogIn("operator", "oprpass")
End With
```

MCal Method, Spel Class

Description
Executes machine calibration for robots with incremental encoders.

Syntax
Sub **MCal** ()

See Also
MCalComplete, MotorsOn

MCal Example
```
If Not m_spel.MCalComplete() Then
    m_spel.MCal()
End If
```

MCalComplete Method, Spel Class

Description

Returns True if MCal has been completed successfully.

Syntax

Function **MCalComplete** () As Boolean

Return Value

True if the MCal has completed, False if not.

See Also

MCal

MCalComplete Example

```
If m_spel.MCalComplete() Then
    lblStatus.Text = "MCal Complete"
Else
    lblStatus.Text = "MCal Not Complete"
End If
```

Mcordr Method, Spel Class

Description
Specifies the moving axis order for machine calibration MCal.

Syntax
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])

Parameters

Step 1 - 9	Bit pattern that tells which axes should home during each step of the MCal process. Any number of axes between 0 to all axes may home during the 1st step. Steps 7 – 9 are optional and are used with robots that have more than 6 axes.

See Also
Home, HomeSet, Hordr, MCal

Mcordr Example
```
m_spel.Mcordr(2, 13, 0, 0, 0, 0)
```

MemIn Method, Spel Class

Description
Returns the status of the specified memory I/O byte port. Each port contains 8 memory I/O bits.

Syntax
Function **MemIn** (*PortNumber* As Integer) As Integer
Function **MemIn** (*Label* As String) As Integer

Parameters

PortNumber	Integer expression representing one of the memory I/O ports.
Label	String expression containing a memory I/O byte label.

Return Value
Integer containing the port value.

See Also
In, InBCD, MemOut, MemSw, Sw, Off, On, Oport

MemIn Example
```
data = m_spel.MemIn(1)
```

MemInW Method, Spel Class

Description
Returns the status of the specified memory I/O word port. Each word port contains 16 memory I/O bits.

Syntax
Function **MemInW** (*PortNumber* As Integer) As Integer
Function **MemInW** (*Label* As String) As Integer

Parameters

PortNumber	Integer expression representing the memory I/O word.
Label	String expression containing a memory I/O word label.

Return Value
Integer containing the port value.

See Also
In, InBCD, MemIn, MemSw, Sw, Off, On, Oport

MemInW Example
```
data = m_spel.MemInW(1)
```

MemOff Method, Spel Class

Description
Turns off the specified bit of the S/W memory I/O.

Syntax
Sub **MemOff** (*BitNumber* As Integer)
Sub **MemOff** (*Label* As String)

Parameters

BitNumber	Integer expression representing one of the memory I/O bits.
Label	String expression containing a memory I/O bit label.

See Also
In, InBCD, MemOut, MemSw, Sw, Off, On, Oport

MemOff Example
```
m_spel.MemOff(500)
```

MemOn Method, Spel Class

Description
Turns on the specified bit of memory I/O.

Syntax
Sub **MemOn** (*BitNumber* As Integer)
Sub **MemOn** (*Label* As String)

Parameters

BitNumber	Integer expression representing one of the memory I/O bits.
Label	String expression containing a memory I/O bit label.

See Also
In, InBCD, MemOut, MemSw, Sw, Off, On, Oport

MemOn Example
```
m_spel.MemOn(500)
```

MemOut Method, Spel Class

Description
Simultaneously sets 8 memory I/O bits based on the 8 bit value specified by the user.

Syntax
Sub **MemOut** (*PortNumber* As Integer, *Value* As Integer)
Sub **MemOut** (*Label* As String, *Value* As Integer)

Parameters

PortNumber	Integer expression representing one of the memory I/O bytes.
Label	String expression containing a memory I/O byte label.
Value	Integer expression containing the output pattern for the specified byte. Valid values are from 0 - 255.

See Also
In, InBCD, MemIn, MemSw, Sw, Off, On, Oport

MemOut Example
```
m_spel.MemOut(2, 25)
```

MemOutW Method, Spel Class

Description
Simultaneously sets 16 memory I/O bits based on the 16 bit value specified by the user..

Syntax
Sub **MemOutW** (*PortNumber* As Integer, *Value* As Integer)
Sub **MemOutW** (*Label* As String, *Value* As Integer)

Parameters

PortNumber	Integer expression representing one of the memory I/O words.
Label	String expression containing a memory I/O word label.
Value	Specifies output data (integer from 0 to 65535) using an expression or numeric value.

See Also
In, InBCD, MemIn, MemSw, Sw, Off, On, Oport

MemOutW Example
```
m_spel.MemOutW(2, 25)
```

MemSw Method, Spel Class

Description
Returns the specified memory I/O bit status.

Syntax
Function **MemSw** (*BitNumber* As Integer) As Boolean
Function **MemSw** (*Label* As String) As Boolean

Parameters

BitNumber	Integer expression representing one of the memory I/O bits.
Label	String expression containing a memory I/O bit label.

Return Value
True if the specified memory I/O bit is on, False if not.

See Also
In, InBCD, MemIn, Sw, Off, On, Oport

MemSw Example
```
If m_spel.MemSw(10) Then
    m_spel.On(2)
End If
```

Move Method, Spel Class

Description
Moves the arm from the current position to the specified point using linear interpolation (i.e. moving in a straight line).

Syntax
Sub **Move** (*PointNumber* As Integer)
Sub **Move** (*Point* As SpelPoint)
Sub **Move** (*Point* As SpelPoint, *AttribExpr* As String)
Sub **Move** (*PointExpr* As String)

Parameters
Each syntax has one parameter that specifies the end point which the arm travels to during the Move motion. This is the final position at the end of the linear interpolated motion.

PointNumber	Specifies the end point by using the point number for a previously taught point in the controller's point memory for the current robot.
Point	Specifies the end point by using a SpelPoint data type.
AttribExpr	Specifies the end point attributes by using a string expression.
PointExpr	Specifies the end point by using a string expression.

See Also
AccelR, AccelS, SpeedR, SpeedS
Arc, Arc3, CVMove, Go, Jump, Jump3, Jump3CP
BGo, BMove, TGo, TMove
Arch, CP, Till

Move Example
```
' Point specified using point number
m_spel.Tool(1)
m_spel.Move(100)

' Point specified using SpelPoint
Dim pt As SpelPoint
pt = m_spel.GetPoint("P*")
pt.X = 125.5
m_spel.Move(pt)

' Point specified using expression
m_spel.Move("P0 /L /2 ROT")
m_spel.Move("P1 :Z(-20)")

' Using parallel processing
m_spel.Move("P1 !D50; On 1; D90; Off 1!")

' Point specified using label
m_spel.Move("pick")
```

Off Method, Spel Class

Description
Turns off the specified output.

Syntax
Sub **Off** (*BitNumber* As Integer)
Sub **Off** (*Label* As String)

Parameters

BitNumber	Integer expression representing one of the standard or expansion outputs. This tells the **Off** instruction which output to turn off.
Label	String expression containing an output bit label.

See Also
On, Oport, Out, OutW

Off Example
```
m_spel.Off(1)
```

On Method, Spel Class

Description
Turns on the specified output.

Syntax
Sub **On** (*BitNumber* As Integer)
Sub **On** (*Label* As String)

Parameters

BitNumber	Integer expression representing one of the standard or expansion outputs. This tells the **On** instruction which output to turn on
Label	String expression containing an output bit label.

See Also
Off, Oport, Out, OutW

On Example
```
m_spel.On(1)
```

OpBCD Method, Spel Class

Description
Simultaneously sets 8 output bits using BCD (Binary Coded Decimal) format.

Syntax
OpBCD (*PortNumber* As Integer, *Value* As Integer)
OpBCD (*Label* As String, *Value* As Integer)

Parameters

PortNumber	Integer number representing one of the ports. Each port contains 8 output bits (one byte).
Value	Integer number between 0-99 representing the output pattern for the specified port. The 2nd digit (called the 1's digit) represents the lower 4 outputs in the port and the 1st digit (called the 10's digit) represents the upper 4 outputs in the port.

See Also
Off, Out, Sw

OpBCD Example
```
m_spel.OpBCD(1, 25)
```

Oport Method, Spel Class

Description
Returns the state of the specified output bit.

Syntax
Function **Oport** (*BitNumber* As Integer) As Boolean
Function **Oport** (*Label* As String) As Boolean

Parameters

BitNumber	Integer expression representing one of the standard and expansion discrete outputs.
Label	String expression containing an output byte label.

Return Value
True if the specified output bit is on, False if not.

See Also
Off, On, OpBCD, Out, Sw

Oport Example
```
If m_spel.Oport(1) Then
    m_spel.On(2)
End If
```

Out Method, Spel Class

Description
Simultaneously reads or sets 8 output bits (one byte).

Syntax
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

Parameters

PortNumber	Integer number representing one of the output ports.
Label	String expression containing an output byte label.
Value	Integer number between 0-255 representing the output pattern for the output port. If represented in hexadecimal form the range is from &H0 to &HFF.

Return Value

Integer number between 0-255 containing the port value.

See Also
InBCD, OpBCD, Oport, OutW, Sw

Out Example
```
m_spel.Out(1, 240)
```

OutW Method, Spel Class

Description
Simultaneously reads or sets 16 output bits (one word).

Syntax
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

Parameters

PortNumber	Integer number representing one of the output ports.
Label	String expression containing an output word label.
Value	Integer number between 0-65535 representing the output pattern for the output port. If represented in hexadecimal form the range is from &H0 to &HFFFF.

Return Value
Integer number between 0-65535 containing the port value.

See Also
InBCD, OpBCD, Oport, Out, Sw

OutW Example
```
m_spel.OutW(1, 240)
```

PAgl Method, Spel Class

Description
Returns the joint angle for the selected rotational axis, or position for the selected linear axis, of the specified point.

Syntax
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

Parameters

PointNumber	Integer expression representing the point number of a point in the current robot's point memory.
Point	A previously initialized SpelPoint.
Label	A string expression containing a point label of a point in the current robot's point memory.
JointNumber	Integer expression representing the desired joint number. The value can be from 1 ~ 9.

Return Value
Single containing the angle for the specified joint in degrees or millimeters.

See Also
Agl, Pls, CX – CT

PAgl Example
```
Dim t1Angle As Single
t1Angle = m_spel.PAgl(1, 1)
```

Pallet Method, Spel Class

Description
Defines pallets.

Syntax
Sub **Pallet** (*PalletNumber* As Integer, *Point1* As String, *Point2* As String, *Point3* As String
[, *Point4* As String] , *rows* As Integer, *columns* As Integer)

Parameters

PalletNumber	Pallet number represented by an integer number from 0 to 15.
Point1	Point variable which defines first pallet position.
Point2	Point variable which defines second pallet position.
Point3	Point variable which defines third pallet position.
Point4	Optional. Point variable which defines fourth pallet position.
Rows	Numbers of points on lateral side of the pallet. Each number is an integer from 1 to 32767.
Columns	Numbers of points on longitudinal side of the pallet. Each number is an integer from 1 to 32767.

See Also
Jump, Go, SetPoint

Pallet Example
```
m_spel.Pallet(1, 1, 2, 3, 4, 3, 4)
```

Pass Method, Spel Class

Description
Specifies the PTP motion to pass a neighborhood of a specified point without stopping motion.

Syntax
Sub **Pass**(PointNumber As Integer)
Sub **Pass**(PassExpr As String)

Parameters

PointNumber Specifies a point using the taught point from point memory of the current robot saved in the Controller.

PassExpr Specifies a point using string expression.

Point specification [, {On | Off | MemOn | MemOff} bit number [,point specification ...]] [LJM [Orientation flag]]

Point specification	Specifies a point number, P(expression), or point label. If the point data is complete and listed in ascending or descending order, two point numbers can be combined using a colon and specified like P(1:5).
Bit number	Specifies I/O output bit or memory I/O bit to turn on/off using an integer or output label.
LJM	Optional. Converts the departure coordinates, approach coordinates, and target coordinates using LJM function.
Orientation flag	Optional. Specifies an orientation flag parameter for the LJM function.

See Also
Accel, Go, Jump, Speed

Pass Example
```
m_spel.Jump(1)
m_spel.Pass(2)  'Move the Arm #2 closer to P2 and execute the following command
                 before reaching P2
m_spel.On(2)
m_spel.Pass(3)
m_spel.Pass(4)
m_spel.Off(0)
m_spel.Pass(5)
```

Pause Method, Spel Class

Description
Causes all normal SPEL+ tasks in the controller to pause. If the robot is moving, it will immediately decelerate to a stop.

Syntax
Sub **Pause** ()

See Also
Continue, EventReceived, Stop

Pause Example
```
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
```

PDef Method, Spel Class

Description
Returns the definition status of a specified point.

Syntax
Function **PDef** (*PointNumber* As Integer) As Boolean

Parameters

PointNumber	Integer expression for the point number of a point in the current robot's point memory.

Return Value
True if the specified point is defined, False if not.

See Also
PDel

PDef Example
```
x = m_spel.PDef(1)
```

PDel Method, Spel Class

Description
Deletes specified position data.

Syntax
Sub **PDel** (*FirstPointNumber* As Integer, [*LastPointNumber* As Integer])

Parameters

FirstPointNumber	Integer expression that specifies the first point in the range to delete.
LastPointNumber	Optional. Integer expression the specifies the last point in range to delete. If omitted, only the point specified in *FirstPointNumber* is deleted.

See Also
PDef, LoadPoints, Clear, SavePoints

PDel Example

```
m_spel.PDel(1, 10)
m_spel.SavePoints("model1.pts")
```

Plane Method, Spel Class

Description
Defines a Plane.

Syntax
Sub **Plane** (*PlaneNumber* As Integer, *Point* As SpelPoint)

Parameters

PlaneNumber	Integer number from 1-15 representing which of the 15 Planes to define.
Point	Point data representing the coordinate data of the approach check plane.

See Also
PlaneClr, PlaneDef

Plane Example
```
m_spel.Plane(1, -5, 5, -10, 10, -20, 20)
```

PlaneClr Method, Spel Class

Description
Clears (undefines) a Plane.

Syntax
Sub **PlaneClr** (*PlaneNumber* As Integer)

Parameters

PlaneNumber	Integer number from 1-15 representing which of the 15 Planes to clear.

See Also
Plane, PlaneDef

PlaneClr Example
```
m_spel.PlaneClr(1)
```

PlaneDef Method, Spel Class

Description
Returns whether a plane is defined.

Syntax
Function **PlaneDef** (*PlaneNumber* As Integer) As Boolean

Parameters
PlaneNumber Integer expression representing the plane number from 1 to 15.

Return Value
True if the specified plane is defined, False if not.

See Also
Plane, PlaneClr

PlaneDef Example
```
x = m_spel.PlaneDef(1)
```

Pls Method, Spel Class

Description
Returns the current encoder pulse count for each axis at the current position.

Syntax
Function **Pls** (*JointNumber* As Integer) As Integer

Parameters
JointNumber The specific axis for which to get the current encoder pulse count. (1 to 9)

Return Value
Integer containing the current pulse count for the specified joint.

See Also
Agl, Pulse

Pls Example
```
j1Pulses = m_spel.Pls(1)
```

PTPBoost Method, Spel Class

Description
Sets the boost parameters for short distance PTP (point to point) motion.

Syntax
Sub **PTPBoost** (*BoostValue* As Integer [, *DepartBoost* As Integer] [, *ApproBoost* As Integer])

Parameters

BoostValue	Integer expression from 0 - 100.
DepartBoost	Optional. Jump depart boost value. Integer expression from 0 - 100.
ApproBoost	Optional. Jump approach boost value. Integer expression from 0 - 100.

See Also
PTPBoostOK

PTPBoost Example
```
m_spel.PTPBoost(50)
m_spel.PTPBoost(50, 30, 30)
```

PTPBoostOK Method, Spel Class

Description
Returns whether or not the PTP (Point to Point) motion from a current position to a target position is a small travel distance.

Syntax
Function **PTPBoostOK** (*PointNumber* As Integer) As Boolean
Function **PTPBoostOK** (*Point* As SpelPoint) As Boolean
Function **PTPBoostOK** (*PointExpr* As String) As Boolean

Parameters
Each syntax has one parameter that specifies the target point to check.

PointNumber	Specifies the target point by using the point number for a previously taught point in the controller's point memory for the current robot.
Point	Specifies the target point by using a SpelPoint data type.
PointExpr	Specifies the target point by using a string expression.

Return Value
True if PTPBoost will be used, False if not.

See Also
PTPBoost

PTPBoostOK Example
```
If m_spel.PTPBoostOK(1) Then
    m_spel.Go(1)
End If
```

PTran Method, Spel Class

Description
Executes a relative joint move in pulses.

Syntax
Sub **PTran** (*JointNumber* As Integer, *Pulses* As Integer)

Parameters

JointNumber	The specific joint to move.
Pulses	The number of pulses to move.

See Also
JTran, Pulse

PTran Example
```
' Move joint 1 5000 pulses in the plus direction.
m_spel.PTran(1, 5000)
```

Pulse Method, Spel Class

Description
Moves the robot arm by Point to Point control to the point specified by the pulse values for all robot joints.

Syntax
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])

Parameters
J1Pulses – J9Pulses	Integer expression containing the pulse value for joints 1 – 9. Joints 5 – 9 are optional.

Note: The pulse values must be within the range specified each joint.

See Also
Go, Move, Jump

Pulse Example
```
m_spel.Pulse(5000, 1000, 0, 0)
```

Quit Method, Spel Class

Description
Terminates execution of the specified task.

Syntax
Sub **Quit** (*TaskNumber* As Integer)
Sub **Quit** (*TaskName* As String)

Parameters

TaskNumber	The task number of the task to be interrupted. The range of the task number is 1 to 32.
TaskName	A string expression containing the name of the task.

See Also
Halt, Resume, Xqt

Quit Example
```
m_spel.Quit(3)
```

RadToDeg Method, Spel Class

Description
Converts Radians into Degrees.

Syntax
Function **RadToDeg** (*Radians* As Double) As Double

Parameters
Radians　Double expression containing the radians to convert into degrees.

Return Value
Double containing the converted value in degrees.

See Also
DegToRad

RadToDeg Example
```
Dim deg As Double

deg = m_spel.RadToDeg(1)
```

RebuildProject Method, Spel Class

Description
Completely rebuilds the current project specified in the Project property.

Syntax
Sub **RebuildProject** ()

See Also
BuildProject, EnableEvent, EventReceived, Project, ProjectBuildComplete

RebuildProject Example
```
With m_spel
    .Project = "c:\EpsonRC70\projects\myproject\myproject.sprj"
    .RebuildProject()
End With
```

Recover Method, Spel Class

Description

Recover moves the robot back to the position is was in when the safeguard was open.

Syntax

Function **Recover** () As Boolean

Remarks

The Recover method can be used after the safeguard is closed to turn on the robot motors and slowly move the robot back to the position it was in when the safeguard was open. After Recover has completed successfully, you can execute the Cont method to continue the cycle.
If Recover was completed successfully, it will return True. Recover will return False if a pause, abort, or safeguard open occurred during recover motion.

Return Value

True if the recover motion was completed, False if not.

See Also

Continue, Pause

Recover Example

This example executes a recover, then 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 With
```

This example shows how a button can be used to execute recover as long as the button is down. If the button is released during recover motion, a pause is issued and recover is aborted. If the button is held down until recover has completed, then a message is displayed.

```
    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
```

Reset Method, Spel Class

Description
Resets the controller to the initialized state.

Syntax
Sub **Reset** ()

See Also
ResetAbort

Reset Example
```
m_spel.Reset()
```

ResetAbort Method, Spel Class

Description
Resets the abort flag that is set with the Stop method.

Syntax
Sub **ResetAbort** ()

Remarks
When the Stop method is executed and no other Spel method is in cycle, then the next Spel method will generate a user abort error. This is done so that no matter when the Stop is issued, the routine that is executing Spel methods will receive the error. Use **ResetAbort** to clear this condition.

Note: The ResetAbortEnabled property must be set to True for the ResetAbort feature to work.

See Also
Abort, Reset, ResetAbortEnabled

ResetAbort Example
```
Sub btnMcal_Click() Handles btnMcal.Click
  m_spel.ResetAbort()
  m_spel.MCal()
End Sub
```

Resume Method, Spel Class

Description
Resumes a task which was suspended by the Halt method.

Syntax
Sub **Resume** (*TaskNumber* As Integer)
Sub **Resume** (*TaskName* As String)

Parameters

TaskNumber	The task number of the task that was interrupted. The range of the task number is 1 to 32.
TaskName	A string expression containing the name of the task.

See Also
Quit, Xqt

Resume Example
```
m_spel.Resume(2)
```

RunDialog Method, Spel Class

Description
Runs an EPSON RC+ 7.0 dialog.

Syntax
Sub **RunDialog** (*DialogID* As SpelDialogs)

Parameters

DialogID	The ID of the EPSON RC+ 7.0 dialog to run.

See Also
ShowWindow

RunDialog Example
```
Sub btnRobotManager_Click( _
       ByVal sender As System.Object, _
       ByVal e As System.EventArgs) _
       Handles btnRobotManager.Click

    m_spel.RunDialog(SpelDialogs.RobotManager)
End Sub
```

SavePoints Method, Spel Class

Description
Save points for the current robot into a file.

Syntax
Sub **SavePoints** (*FileName* As String)

Parameters

FileName	The file name to save the points in the current project.

See Also
LoadPoints

SavePoints Example
```
With m_spel
    .SavePoints("part1.pts")
End With
```

Sense Method, Spel Class

Description
Specifies input condition that, if satisfied, completes the Jump in progress by stopping the robot above the target position.

Syntax
Sub **Sense** (*Condition* As String) As Boolean

Parameters
Condition	Specifies the I/O condition. For details see the Sense Statement in the SPEL+ Language Reference manual.

See Also
Jump, JS

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

SetIODef Method, Spel Class

Description
Sets the I/O label and description for an input, output, or memory I/O bit, byte, or word.

Syntax
Sub **SetIODef** (*Type* As SpelLabelTypes, *Index* As Integer, *Label* As String, *Description* As String)

Parameters
Type Specifies the I/O type as shown below:

InputBit = 1, InputByte = 2, InputWord = 3

OutputBit = 4, OutputByte = 5, OutputWord = 6

MemoryBit = 7, MemoryByte = 8, MemoryWord = 9

Index Specifies the bit or port number.

Label Specifies the new label.

Description Specifies the new description.

Remarks
Use SetIODef to define the label and description for any I/O point.

See Also
GetIODef

SetIODef Example
```
Dim label, desc As String
label = "StartCycle"
desc = "Starts the robot cycle"
m_spel.SetIODef(SpelLabelTypes.InputBit, 0, label, desc)
```

SetPoint Method, Spel Class

Description

Sets the coordinate data for a point for the current robot.

Syntax

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)

Parameters

PointNumber	Integer expression that specifies the point number for a point in the current robot's point memory.
X	The X coordinate for the specified point.
Y	The Y coordinate for the specified point.
Z	The Z coordinate for the specified point.
U	The U coordinate for the specified point.
V	The V coordinate for the specified point.
W	The W coordinate for the specified point.
S	The S coordinate for the specified point.
T	The T coordinate for the specified point.
Local	The Local Number for the specified point. Use 0 when there is no local.
Hand	The hand orientation of the specified point.
Elbow	The elbow orientation of the specified point.
Wrist	The wrist orientation of the specified point.

Note

Do not enter integer values to X, Y, Z, U, V, W, S, and T parameters. Use Single variables or directly enter Single type values.

See Also

GetPoint, LoadPoints, SavePoints

SetPoint Example

```
Dim pt As SpelPoint
' Get coordinates of P1
pt = m_spel.GetPoint(1)
' Set it with changes
pt.U = pt.U - 10.5
m_spel.SetPoint(1, pt)
```

SetVar Method, Spel Class

Description
Sets the value of a SPEL+ global preserve variable in the controller.

Syntax
Sub **SetVar** (*VarName* As String, *Value* As Object)

Parameters
VarName	The name of the SPEL+ global preserve variable.
Value	The new value.

Remarks
You can use SetVar to set the values for single variables and array variables. See the examples below.

See Also
GetVar

SetVar Example
```
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))
```

SFree Method, Spel Class

Description
Frees the specified robot axes from servo control.

Syntax
Sub **SFree** ()
Sub **SFree** (ParamArray *Axes*() As Integer)

Parameters
Axes An integer parameter array containing one element for each robot axis to free. You can specify axis numbers from 1 – 9.

See Also
SLock

SFree Example
```
' Free Axes 1 & 2
m_spel.SFree(1, 2)
```

ShowWindow Method, Spel Class

Description
Shows an EPSON RC+ 7.0 window.

Syntax
Sub **ShowWindow** (*WindowID* As SpelWindows, [*Parent* As Form])

Parameters

WindowID	The ID of the EPSON RC+ 7.0 window to show.
Parent	Optional. A .NET form that will be the parent of the window.

Remarks
You can use the Parent parameter to specify the .NET parent form for the window. If you cannot use a .NET parent form, you must omit the Parent parameter and use the ParentWindowHandle property to set the handle of the parent.

See Also
HideWindow, ParentWindowHandle, RunDialog

ShowWindow Example
```
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
```

Shutdown Method, Spel Class

Description
Shutdown or restart Windows.

Syntax
Sub **Shutdown** (*Mode* As SpelShutdownMode)

Parameters

Mode	0 = Shutdown Windows. 1 = Restart Windows.

See Also
Reset

Shutdown Example
```
' Restart Windows
m_spel.Shutdown(1)
```

SLock Method, Spel Class

Description
Returns specified axes to servo control.

Syntax
Sub **SLock** ()
Sub **SLock** (ParamArray *Axes*() As Integer)

Parameters
Axes An integer parameter array containing one element for each robot axis to lock. You can specify axis numbers from 1 – 9.

See Also
SFree

SLock Example
```
' Return Axes 1 and 2 to servo control
m_spel.SLock(1, 2)
```

Speed Method, Spel Class

Description
Specifies the arm speed for use with the point to point instructions Go, Jump and Pulse.

Syntax
Sub **Speed** (*PointToPointSpeed* As Integer [, *JumpDepartSpeed* As Integer]
[, *JumpApproSpeed* As Integer])

Parameters

PointToPointSpeed	Specifies the arm speed for use with the point to point instructions Go, Jump and Pulse.
JumpDepartSpeed	Integer number between 1-100 representing the Z axis upward motion speed for the Jump instruction.
JumpApproSpeed	Integer number between 1-100 representing the Z axis downward motion speed for the Jump instruction.

See Also
Accel, Jump, Go

Speed Example
```
m_spel.Speed(50)
```

SpeedR Method, Spel Class

Description
Specifies the tool rotation speed when ROT is used.

Syntax
Sub **SpeedR** (*RotationSpeed* As Single)

Parameters
RotationSpeed Specifies the tool rotation speed in degrees / second.

See Also
Arc, Arc3, BMove, Jump3CP, Power, TMove

SpeedR Example
```
m_spel.SpeedR(100)
```

SpeedS Method, Spel Class

Description

Specifies the arm speed for use with the Continuous Path instructions Jump3CP, Move, Arc, and CVMove.

Syntax

Sub **SpeedS** (*LinearSpeed* As Single [, *JumpDepartSpeed* As Single] [, *JumpApproSpeed* As Single])

Parameters

LinearSpeed	Specifies the arm speed for use with the Continuous Path instructions Jump3CP, Move, Arc, and CVMove.
JumpDepartSpeed	Single expression between 1-5000 representing the Z axis upward motion speed for the Jump3CP instruction.
JumpApproSpeed	Single expression between 1-5000 representing the Z axis downward motion speed for the Jump3CP instruction.

See Also

AccelS, Jump3CP, Move, TMove

SpeedS Example

```
m_spel.SpeedS(500)
```

Start Method, Spel Class

Description
Start one SPEL+ program.

Syntax
Sub **Start** (*ProgramNumber* As Integer)

Parameters
ProgramNumber The program number to start, corresponding to the 64 main functions in SPEL+ as shown in the table below. The range is 0 to 63.

Program Number	SPEL+ Function Name
0	main
1	main1
2	main2
3	main3
4	main4
5	main5
…	…
63	main63

Remarks
When **Start** is executed, control will return immediately to the calling program. You cannot start a program that is already running. Note that Start causes global variables in the controller to be cleared and default robot points to be loaded.

See Also
Continue, Pause, Stop, Xqt

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

    m_spel.Start(0)
End Sub
```

StartBGTask Method, Spel Class

Description
Start one SPEL+ task as a background task.

Syntax
Sub **StartBGTask** (*FuncName* As String)

Parameters
FuncName The name of the function to be executed.

Remarks
Use StartBGTask to start a Spel+ background task in the controller. Background tasks must be enabled in the controller.

Note that BGMain automatically starts when the controller switches to auto mode, so normally StartBGTask is not required. StartBGTask is provided in case you need to stop all tasks, then start background tasks again.

See Also
Call, Start, Stop, Xqt

StartBGTask Example
```
' Stop all tasks, including background tasks
m_spel.Stop(SpelStopType.StopAllTasks)
...
m_spel.RebuildProject()

' Start the main background task
m_spel.StartBGTask("BGMain")
```

Stop Method, Spel Class

Description

Stops all normal SPEL+ tasks running in the controller and optionally stop all background tasks.

Syntax

Sub **Stop** ()
Sub **Stop** (SpelStopType *StopType*)

Parameters

StopType	Optional. Specifies whether to stop only normal tasks (StopNormalTasks) or all tasks (StopAllTasks). If omitted, StopNormalTasks is specified.

Note: If the Stop method is executed when ResetAbortEnabled is True, the error 10101 occurs when executing Start or Reset methods.
To release the error, execute ResetAbort method after executing Stop method.

See Also

Continue, Pause, ResetAbort, ResetAbortEnabled, Start, SpelStopType

Stop Example

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

    m_spel.Stop()
End Sub
```

Sw Method, Spel Class

Description
Returns the selected input bit status.

Syntax
Function **Sw** (*BitNumber* As Integer) As Boolean
Function **Sw** (*Label* As String) As Boolean

Parameters
BitNumber Integer expression representing one of the standard or expansion inputs.

Label String expression containing an input bit label.

Return Value
True if the specified input bit is on, False if not.

See Also
In, InBCD, MemSw, Off, On, Oport

Sw Example
```
If m_spel.Sw(1) Then
    m_spel.On(2)
End If
```

TargetOK Method, Spel Class

Description
Returns a status indicating whether or not the PTP (Point to Point) motion from the current position to a target position is possible.

Syntax
Function **TargetOK** (*PointNumber* As Integer) As Boolean
Function **TargetOK** (*Point* As SpelPoint) As Boolean
Function **TargetOK** (*PointExpr* As String) As Boolean

Parameters
Each syntax has one parameter that specifies the target point to check.

PointNumber	Specifies the target point by using the point number for a previously taught point in the controller's point memory for the current robot.
Point	Specifies the target point by using a SpelPoint data type.
PointExpr	Specifies the target point by using a string expression.

Return Value
True if the target can be moved to from the current position, False if not.

See Also
Go, Jump, Move, TGo, TMove

TargetOK Example
```
If m_spel.TargetOK("P1 /F") Then
    m_spel.Go("P1 /F")
End If
```

TasksExecuting Method, Spel Class

Description
Returns True if any SPEL+ tasks are executing.

Syntax
Function **TasksExecuting** () As Boolean

Return Value
True if any SPEL+ tasks are executing, False if not.

See Also
TaskState, Xqt

TasksExecuting Example
```
tasksRunning = m_spel.TasksExecuting()
```

TaskState Method, Spel Class

Description
Returns the status of a task.

Syntax
Function **TaskState** (*TaskNumber* As Integer) As SpelTaskState
Function **TaskState** (*TaskName* As String) As SpelTaskState

Parameters

TaskNumber	Task Number to return the execution status of.
TaskName	String expression containing the name of the task.

Return Value
A SpelTaskState value.

See Also
TasksExecuting, Xqt

TaskState Example
```
Dim taskState As SpelTaskState
taskState = m_spel.TaskState(2)
```

TeachPoint Method, Spel Class

Description
Runs a dialog that allows an operator to jog and teach one point.

Syntax
Function **TeachPoint** (*PointFile* As String, *PointNumber* As Integer, *Prompt* As String) As Boolean

Parameters

PointFile	A string containing the name of the point file.
PointNumber	The point number to teach.
Prompt	A string containing the instructional text that is displayed on the bottom of the teach dialog.

Return Value
Returns True if the operator clicked the Teach button, False if the operator clicked Cancel.

Remarks
Use TeachPoints to allow an operator to teach one robot point in the controller. When TeachPoints is executed, the point file is loaded from the controller. When the Teach button is clicked, the point is taught in the controller and the point file is saved on the controller.

TeachPoint Example
```
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
```

Till Method, Spel Class

Description
Specifies event condition that, if satisfied, completes the motion command (Jump, Go, Move, etc.) in progress by decelerating and stopping the robot at an intermediate position.

Syntax
Sub **Till** (*Condition* As String) As Boolean

Parameters
Condition Specifies the I/O condition. For details see the Till Statement in the SPEL+ Language Reference manual.

See Also
Go, Jump, JS, Sense, TillOn

Till Example
```
With m_spel
    .Till("Sw(1) = On")
    .Go("P1 TILL")
End With
```

TillOn Method, Spel Class

Description
Returns True if a stop has occurred from a till condition during the last Go/Jump/Move statement.

Syntax
Function **TillOn** () As Boolean

Return Value
True if the robot stopped due to a Till condition, False if not.

Remarks
Use **TillOn** to check if the Till condition turned on during the last motion command using Till.

TillOn is equivalent to `((Stat(1) And 2) <> 0)`

See Also
Jump, Till

TillOn Example
```
If m_spel.TillOn() Then
    m_spel.Jump(2)
End If
```

TGo Method, Spel Class

Description
Executes Point to Point relative motion, in the current tool coordinate system.

Syntax
Sub **TGo** (*PointNumber* As Integer)
Sub **TGo** (*Point* As SpelPoint)
Sub **TGo** (*Point* As SpelPoint, *AttribExpr* As String)
Sub **TGo** (*PointExpr* As String)

Parameters
Each syntax has one parameter that specifies the end point which the arm travels to during the TGo motion. This is the final position at the end of the point to point motion.

PointNumber	Specifies the end point by using the point number for a previously taught point in the controller's point memory for the current robot.
Point	Specifies the end point by using a SpelPoint data type.
AttribExpr	Specifies the end point attributes by using a string expression.
PointExpr	Specifies the end point by using a string expression.

See Also
Accel, Speed
Arc, Arc3, CVMove, Go, Jump, Jump3, Jump3CP, Move
BGo, BMove, TMove
CP, Till

TGo Example
```
' Point specified using point number
m_spel.Tool(1)
m_spel.TGo(100)

' Point specified using SpelPoint
Dim pt As SpelPoint
pt = m_spel.GetPoint("P*")
pt.X = 125.5
m_spel.TGo(pt)

' Point specified using expression
m_spel.TGo("P0 /L /2")
m_spel.TGo("P1 :Z(-20)")

' Using parallel processing
m_spel.TGo("P1 !D50; On 1; D90; Off 1!")

' Point specified using label
m_spel.TGo("pick")
```

TLClr Method, Spel Class

Description
Clears (undefines) a tool coordinate system.

Syntax
Sub **TLClr** (*ToolNumber* As Integer)

Parameters

ToolNumber	Integer expression representing which of the tools to clear (undefine). (Tool 0 is the default tool and cannot be cleared.)

See Also
Tool, ToolDef

ToolClr Example
```
m_spel.ToolClr(1)
```

TLDef Method, Spel Class

Description
Returns tool definition status.

Syntax
Function **TLDef** (*ToolNumber* As Integer) As Boolean

Parameters
ToolNumber Integer expression representing which tool to return status for.

Return Value
True if the specified tool is defined, False if not.

See Also
Tool, ToolClr

ToolDef Example
```
m_spel.ToolDef(1)
```

TLSet Method, Spel Class

Description
Defines a tool coordinate system.

Syntax
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)

Parameters

ToolNumber	Integer expression from 1-15 representing which of 15 tools to define. (Tool 0 is the default tool and cannot be modified.)
Point	A SpelPoint containing the point data.
XCoord	The tool coordinate system origin X coordinate.
YCoord	The tool coordinate system origin Y coordinate.
ZCoord	The tool coordinate system origin Z coordinate.
UCoord	The tool coordinate system rotation about the Z axis.
VCoord	The tool coordinate system rotation about the Y axis.
WCoord	The tool coordinate system rotation about the X axis.

See Also
Arm, Armset, GetTool, Tool

TLSet Example
```
m_spel.TLSet(1, .5, 4.3, 0, 0, 0, 0)
```

TMove Method, Spel Class

Description
Executes linear interpolation relative motion, in the current tool coordinate system

Syntax
Sub **TMove** (*PointNumber* As Integer)
Sub **TMove** (*Point* As SpelPoint)
Sub **TMove** (*Point* As SpelPoint, *AttribExpr* As String)
Sub **TMove** (*PointExpr* As String)

Parameters
Each syntax has one parameter that specifies the end point which the arm travels to during the TMove motion. This is the final position at the end of the linear interpolated motion.

PointNumber	Specifies the end point by using the point number for a previously taught point in the controller's point memory for the current robot.
Point	Specifies the end point by using a SpelPoint data type.
AttribExpr	Specifies the end point attributes by using a string expression.
PointExpr	Specifies the end point by using a string expression.

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

TMove Example
```
' Point specified using point number
m_spel.Tool(1)
m_spel.TMove(100)

' Point specified using SpelPoint
Dim pt As SpelPoint
pt = m_spel.GetPoint("P*")
pt.X = 125.5
m_spel.TMove(pt)

' Point specified using expression
m_spel.TGo("P0")
m_spel.TGo("XY(0, 0, -20, 0)")

' Using parallel processing
m_spel.TMove("P1 !D50; On 1; D90; Off 1!")

' Point specified using label
m_spel.TMove("pick")
```

Tool Method, Spel Class

Description
Selects the current robot tool.

Syntax
Sub **Tool** (*ToolNumber* As Integer)

Parameters

ToolNumber	Integer number from 0-15 representing which of 16 tool definitions to use with the subsequent motion instructions.

See Also
TLSet, Arm, TGo, TMove

Tool Example
```
m_spel.Tool(1)
m_spel.TGo(100)
```

TrapStop Method, Spel Class

Description
Returns True if the current robot was stopped by a trap during the previous motion command.

Syntax
Function **TrapStop** () As Boolean

Return Value
True if the robot was stopped by a trap, False if not.

See Also
EStopOn, ErrorOn

TrapStop Example
```
If m_spel.TrapStop() Then
    MsgBox "Robot stopped by Trap"
End If
```

TW Method, Spel Class

Description
Returns the status of the WAIT condition and WAIT timer interval.

Syntax
Function **TW** () As Boolean

Return Value
True if a timeout occurred, False if not.

See Also
WaitMem, WaitSw

TW Example
```
Const PartPresent = 1
m_spel.WaitSw(PartPresent, True, 5)
If m_spel.TW() Then
    MsgBox "Part present time out occurred"
End If
```

UserHasRight Method, Spel Class

Description
Returns whether the currently logged in user has the specified right.

Syntax
Function **UserHasRight** (SpelUserRights *Right*) As Boolean

Parameters
Right The right you want to check for the current logged in user.

Return Value
True if the user has the specified right, False if not.

See Also
Login, GetCurrentUser

UserHasRight Example
```
Dim hasRight As Boolean
hasRight = m_spel.UserHasRight(SpelUserRights.EditPoints)
```

VCal Method, Spel Class

Description
This command allows you to execute a vision calibration cycle.

Syntax
Sub **VCal** (*CalibName* As String)

Parameters

CalibName	A string expression that evaluates to the name of a calibration scheme in the current project.

Remarks
When you execute the **VCal** method, the robot will move. You should verify that the operator is ready before executing **VCal**.

VCal only executes the calibration cycle. It does not allow you to teach points. Use VCalPoints to teach points. Also, you must first set up a calibration in EPSON RC+ 7.0. See your Vision Guide manuals for details.

See Also
VCalPoints

VCal Example
```
m_spel.VCal("CAMCAL1")
```

VCalPoints Method, Spel Class

Description
This command enables you to teach vision calibration points.

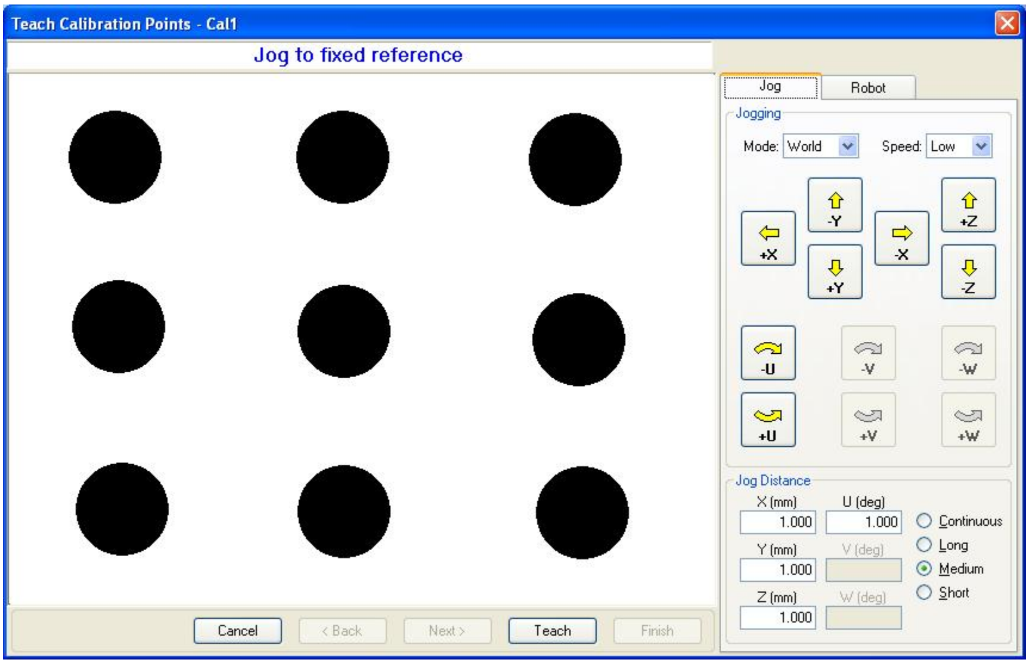
Syntax
Sub **VCalPoints** (*CalibName* As String)

Parameters

CalibName	A string expression that evaluates to the name of a calibration scheme in the current project.

Remarks
When you execute the **VCalPoints** command, the Teach Calibration Points dialog is opened. When OK is clicked, the calibration data is automatically saved.

You must have already created the calibration scheme from EPSON RC+ 7.0.



See Also
VCal

VCalPoints Example
```
m_spel.VCalPoints("CAMCAL1")
```

VCls Method, Spel Class

Description
Clears vision graphics.

Syntax
Sub **VCls** ()

Remarks
Use the VCls method to clear the vision screen.

See Also
VRun

VCls Example
```
m_spel.VCls()
```

VCreateCalibration Method, Spel Class

Description
Creates a new vision calibration in the current project.

Syntax
Sub **VCreateCalibration** (*CameraNumber* As Integer, *CalibName* As String)
Sub **VCreateCalibration** (*CameraNumber* As Integer, *CalibName* As String,
CopyCalibName As String)

Parameters

CameraNumber	Integer expression containing the number of the camera to be calibrated.
CalibName	String expression containing the name of a vision calibration to create.
CopyCalibName	Optional. String expression containing the name of a vision calibration to copy.

See Also
VCreateObject, VCreateSequence, VDeleteCalibration

VCreateCalibration Example
```
m_spel.VCreateCalibration(1, "mycal")
```

VCreateObject Method, Spel Class

Description
Creates a vision object in the current project.

Syntax
Sub **VCreateObject** (*Sequence* As String, *ObjectName* As String, *ObjectType* As SpelVisionObjectTypes)

Parameters

Sequence	String expression containing the name of a vision sequence in the current project.
ObjectName	String expression containing the name of an object to create in sequence *Sequence*.
ObjectType	A SpelVisionObjectTypes that specifies the vision object type.

Object Type	SpelVisionObjectTypes	Value
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

See Also
VCreateSequence, VDeleteObject, VDeleteSequence

VCreateObject Example
```
m_spel.VCreateObject("myseq", "myblob",
SpelVisionObjectTypes.Blob)
```

VCreateSequence Method, Spel Class

Description
Creates a new vision sequence in the current project.

Syntax
Sub **VCreateSequence** (*CameraNumber* As Integer, *SequenceName* As String)
Sub **VCreateSequence** (*CameraNumber* As Integer, *SequenceName* As String,
CopySequenceName As String)

Parameters

CameraNumber	Integer expression containing the number of the camera to be used.
SequenceName	String expression containing the name of a vision sequence to create.
CopySequenceName	Optional. String expression containing the name of a vision sequence to copy.

See Also
VCreateObject, VDeleteObject, VDeleteSequence

VCreateSequence Example
```
m_spel.VCreateSequence(1, "myseq")
```

VDefArm Method, Spel Class

Description

Calculates an arm set value of a mobile camera using a feature point detectable by the vision system.

Syntax

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)

Parameters

ArmNumber Integer expression that contains the arm number to perform arm set (1 to 15).

ArmDefType Integer expression that contains the arm type.

J2Camera: Calculates a center of mobile J2 camera image.

ArmDefMode Integer expression that contains the arm set mode.

Rough: A mode to run a rough arm set.
Robot will move with setting accuracy of 1 mm as a target.
Robot motion will be small.

Fine: A mode to run a fine arm set.
Robot will move largely with arm orientation change and provide arm set with more high accuracy.

Sequence String expression containing a vision sequence name of current project.

Rotation Real expression that contains rotation angle (degrees) for a rough arm set.

Value range: 0 to 45

TargetTolerance Real expression containing a pixel distance to consider that the vision detection result matches the target position.

Value range: 0 to 3 pixels

Parent Optional. Parent .NET form of a window.

RobotSpeed Optional. Integer variable that will contain the robot speed (%).
Value range: 0 to 100
If omitted, set to “5”.

RobotAccel Optional. Integer variable that will contain the robot acceleration (%).
Value range: 0 to 99
If omitted, set to “5”.

ShowWarning Optional. Integer variable that determines whether to display warning when *ArmSetMode* is Fine.

Always	: Always display
DependsOnSpeed	: Display when either *RobotSpeed* or *RobotAccel* is larger than 5.
None	: Do not display

If omitted, set to “DependsOnSpeed”.

See Also
VDefGetMotionRange, VDefLocal, VDefSetMotionRange, VDefTool, VGoCenter

VDefArm Example
```
m_spel.VDefArm(1, SpelArmDefType.J2Camera, SpelArmDefMode.Rough,
"myseq", 5, 1)
```

VDefGetMotionRange Method, Spel Class

Description
Acquires values of the motion range limited by VDefTool, VDefArm, VDefLocal, and VGoCenter.

Syntax
Sub **VDefGetMotionRange**(ByRef MaxMoveDist As Double, ByRef MaxPoseDiffAngle As Double, ByRef LjmMode As Integer)

Parameters

MaxMoveDist	Real variable representing the maximum distance of move. If 0 is specified, the range is not limited. (0 to 500. Default: 200) VDeopfTool, VDefArm, VDefLocal, and VGoCenter are used to limit the range.
MaxPoseDiffAngle	Real variable representing the maximum displacement angle (degrees) of tool orientation (UVW). If 0 is specified, the angle is not limited. It only affects VDefLocal. (0 to 180. Default: 45 degrees)
LjmMode	Integer variable representing the LJM mode.

See Also
VDefTool, VDefArm, VDefLocal, VGoCenter, VDefSetMotionRange

VDefGetMotionRange Example

```
Dim maxMoveDist As Double
Dim maxPoseDiffAngle As Double
Dim ljmMode As Integer
m_spel.VDefGetMotionRange(maxMoveDist, maxPoseDiffAngle,
ljmMode)
```

VDefLocal Method, Spel Class

Description

Detects a calibration plate placed on a work plane by a mobile camera, and defines local coordinates parallel to the work plane.
It also detects user's workpiece at the tool end by a fixed camera and defines a local plane which is parallel to a fixed camera sensor.

Syntax

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)

Parameters

LocalNumber Integer representing a tool number to set local coordinates. (1-15)

LocalDefType Integer representing a local type.

J5Camera: Specifies local coordinates parallel to a calibration plate by using the mobile J5 camera.

J6Camera: Specifies local coordinates parallel to a calibration plate by using the mobile J6 camera.

FixedUpwardCamera: Specifies local coordinates parallel to an image sensor by using the upward fixed camera.

FixedDownwardCamera: Specifies local coordinates parallel to an image sensor by using the downward fixed camera.

CalPlateType Integer representing a type of calibration plate.

Large : Large calibration plate

Medium : Medium calibration plate

Small : Small calibration plate

XSmall : Extra small calibration plate

Sequence String expression representing a vision sequence name of current project.

When using the mobile camera, this is a vision sequence to take a picture of the calibration plate.
When using the fixed camera, this is a vision sequence to detect a feature point at tool end, such as user's workpiece.

TargetTolerance Real value representing a threshold value to judge scale coincidence.

CameraTool Fixed camera: Specifies a tool number that holds a tool offset of the detection target. To perform auto calibration, specify -1.

Mobile J6 camera: If auto calibration has been executed, specify a tool number of mobile camera.
To perform auto calibration, specify -1.

Mobile J5 camera: Setting of this option is ignored.

RefPoint Point number which a local plane parallel to a work plane passes.

This point is used to specify local plane height.

Parent	Optional. Parent .NET form of a window.
RobotSpeed	Optional. Integer variable that will contain the robot speed (%). Value range: 0 to 100 If omitted, set to “5”.
RobotAccel	Optional. Integer variable that will contain the robot acceleration (%). Value range: 0 to 99 If omitted, set to “5”.

See Also

VDefArm, VDefGetMotionRange,VDefSetMotionRange, VDefTool, VGoCenter

VDefLocal Example

```
Dim p2 = m_spel.GetPoint("P2")
m_spel.VDefLocal(1, SpelLocalDefType.J6Camera,
SpelCalPlateType.Large, "myseq", 1.0, 1, p2)
```

VDefSetMotionRange Method, Spel Class

Description
Limits a motion range by VDefTool, VDefArm, VDefLocal, and VGoCenter.

Syntax
Sub **VDefSetMotionRange**(MaxMoveDist As Double, MaxPoseDiffAngle As Double, LjmMode As Integer)

Parameters

MaxMoveDist	Real value representing the maximum distance of move. If 0 is specified, the range is not limited. (0 to 500. Default: 200) VDefTool, VDefArm, VDefLocal, and VGoCenter are used to limit the range.
MaxPoseDiffAngle	Real value representing the maximum displacement angle (degrees) of tool orientation (UVW). If 0 is specified, the angle is not limited. It only affects VDefLocal. (0 to 180. Default: 45 degrees)
LjmMode	Integer representing the LJM mode.

See Also
VDefTool, VDefArm, VDefLocal, VGoCenter, VDefGetMotionRange

VDefSetMotionRange Example
```
m_spel.VDefSetMotionRange(100, 30, 1)
```

VDefTool Method, Spel Class

Description
Using vision detection, calculates a tool offset value for TPC and mobile camera position.

Syntax
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)

Parameters

ToolNumber	Integer representing a tool number to perform tool set (1-15)
ToolDefType	Integer representing a tool type. FixedCamera: Tool set by using the fixed camera which is not calibrated. J4Camera: Calculates image center of the mobile J4 camera. J6Camera: Calculates image center of the mobile J6 camera. FixedCameraWithCal: Tool set by using the fixed camera which is calibrated.
Sequence	String expression representing the name of a vision sequence in the current project.
Object	String expression representing a vision object in the specified sequence. This parameter is required when *ToolDefType* is FixedCameraWithCal. When *ToolDefType* is not FixedCameraWithCal, Object should be an empty string.
FinalAngle	Real value representing an angle (degrees) to rotate the tool or camera tool. Value range: 0, 5 to 180, −5 to −180 If omitted, set to “90”.
InitAngle	Real value representing an angle (degrees) to rotate the tool or camera tool in provisional tool setting. This value must be smaller than *FinalAngle.* Value range: −10 to 10 If omitted, set to “5”.
TargetTolerance	Real value representing a pixel distance to consider that the vision detection result matches the target position. Value range: 0 to 3 pixels If omitted, set to “1”.
Parent	Optional. Parent .NET form of a window.
RobotSpeed	Optional. Integer variable that will contain the robot speed (%). Value range: 0 to 100 If omitted, set to “5”.
RobotAccel	Optional. Integer variable that will contain the robot acceleration (%). Value range: 0 to 99 If omitted, set to “5”.

See Also
VDefArm, VDefGetMotionRange, VDefLocal, VDefSetMotionRange, VGoCenter

VDefTool Example

```
m_spel.VDefTool(1, SpelToolDefType.J6Camera, "myseq", 45, 5,
3.0)
m_spel.VDefTool(1, SpelToolDefType.FixedCameraWithCal, "myseq",
"myobj")
```

VDeleteCalibration Method, Spel Class

Description
Deletes a vision calibration in the current project.

Syntax
Sub **VDeleteCalibration** (*CalibName* As String)

Parameters

CalibName	String expression containing the name of a vision calibration in the current project.

See Also
VCreateCalibration, VDeleteObject, VDeleteSequence

VDeleteCalibration Example

```
m_spel.VDeleteCalibration("mycal")
```

VDeleteObject Method, Spel Class

Description
Deletes a vision object in the current project.

Syntax
Sub **VDeleteObject** (*Sequence* As String, *ObjectName* As String)

Parameters
Sequence String expression containing the name of a vision sequence in the current project.

ObjectName String expression containing the name of a vision object in the current project.

See Also
VCreateObject, VCreateSequence, VDeleteSequence

VDeleteObject Example
```
m_spel.VDeleteObject("myseq", "myobj")
```

VDeleteSequence Method, Spel Class

Description
Deletes a vision sequence in the current project.

Syntax
Sub **VDeleteSequence** (*Sequence* As String)

Parameters

Sequence	String expression containing the name of a vision sequence in the current project.

See Also
VCreateObject, VCreateSequence, VDeleteObject

VDeleteSequence Example
```
m_spel.VDeleteSequence("myseq")
```

VGet Method, Spel Class

Description
Gets the value of a vision sequence or object property or result.

Syntax
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 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 Double)

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 String)

Parameters

Sequence	String expression containing the name of a vision sequence in the current project.
Object	String expression containing the name of an object in sequence *Sequence*. If the property is for a sequence, then this string must be empty.
PropCode	A SpelVisionProps value that specifies the property code.
Value	Variable containing property or result value. The type of the variable must match the property or result type.

See Also
VSet, VRun

VGet Example
```
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
```

VGetCameraXYU Method, Spel Class

Description
Retrieves camera X, Y, and U physical coordinates for any object.

Syntax
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)

Parameters

Sequence	String expression containing the name of a vision sequence in the current project.
Object	String expression containing the name of an object in sequence *Sequence*.
Result	Integer expression representing the result number.
Found	Boolean variable that will contain whether or not the object was found.
X	Real variable that will contain x coordinate in millimeters.
Y	Real variable that will contain y coordinate in millimeters.
U	Real variable that will contain angle in degrees.

See Also
VGetPixelXYU, VGetRobotXYU

VGetCameraXYU Example
```
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)
```

VGetEdgeCameraXYU Method, Spel Class

Description
Retrieves camera X, Y, and U physical coordinates for each edge of a Line Finder, Arc Finder search.

Syntax
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)

Parameters

Sequence	String expression containing the name of a vision sequence in the current project.
Object	String expression containing the name of an object in sequence *Sequence*.
EdgeResultIndex	Integer expression representing the edge result index.
Found	Boolean variable that will contain whether or not the object was found.
X	Real variable that will contain x coordinate in millimeters.
Y	Real variable that will contain y coordinate in millimeters.
U	Real variable that will contain angle in degrees.

See Also
VGetEdgePixelXYU, VGetEdgeRobotXYU, VGetPixelXYU, VGetRobotXYU

VGetEdgeCameraXYU Example
```
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)
' The NumberOfEdges for the LineFinder is 10
For i = 1 To 10
  m_spel.VGetEdgeCameraXYU(seq, lineFinder, i, found(i), x(i),
     y(i), u(i))
Next i
```

VGetEdgePixelXYU Method, Spel Class

Description
Retrieves X, Y, and U pixel coordinates for each edge of a Line Finder, Arc Finder search.

Syntax
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)

Parameters

Sequence	String expression containing the name of a vision sequence in the current project.
Object	String expression containing the name of an object in sequence *Sequence*.
EdgeResultIndex	Integer expression representing the edge result index.
Found	Boolean variable that will contain whether or not the object was found.
X	Real variable that will contain x coordinate in millimeters.
Y	Real variable that will contain y coordinate in millimeters.
U	Real variable that will contain angle in degrees.

See Also
VGetEdgeCameraXYU, VGetEdgeRobotXYU, VGetPixelXYU, VGetRobotXYU

VGetEdgePixelXYU Example
```
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)
' The NumberOfEdges for the LineFinder is 10
For i = 1 To 10
  m_spel.VGetEdgePixelXYU(seq, lineFinder, i, found(i), x(i),
     y(i), u(i))
Next i
```

VGetEdgeRobotXYU Method, Spel Class

Description
Retrieves robot X, Y, and U physical coordinates for each edge of a Line Finder, Arc Finder search.

Syntax
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)

Parameters

Sequence	String expression containing the name of a vision sequence in the current project.
Object	String expression containing the name of an object in sequence *Sequence*.
EdgeResultIndex	Integer expression representing the edge result index.
Found	Boolean variable that will contain whether or not the object was found.
X	Real variable that will contain x coordinate in millimeters.
Y	Real variable that will contain y coordinate in millimeters.
U	Real variable that will contain angle in degrees.

See Also
VGetEdgeCameraXYU, VGetEdgePixelXYU, VGetPixelXYU, VGetRobotXYU

VGetEdgeRobotXYU Example
```
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)
' The NumberOfEdges for the LineFinder is 10
For i = 1 To 10
  m_spel.VGetEdgeRobotXYU(seq, lineFinder, i, found(i), x(i),
     y(i), u(i))
Next i
```

VGetExtrema Method, Spel Class

Description
Retrieves extrema coordinates of a blob object.

Syntax
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)

Parameters

Sequence	String expression containing the name of a vision sequence in the current project.
Object	String expression containing the name of an object in sequence *Sequence*.
Result	Integer expression representing the result number.
MinX	Real variable that will contain minimum x coordinate in pixels.
MaxX	Real variable that will contain maximum x coordinate in pixels.
MinY	Real variable that will contain minimum y coordinate in pixels.
MaxY	Real variable that will contain maximum y coordinate in pixels.

See Also
VGet

VGetExtrema Example
```
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
```

VGetModelWin Method, Spel Class

Description
Retrieves model window coordinates for objects.

Syntax
Sub **VGetModelWin** (*Sequence* As String, *Object* As String, ByRef *Left* As Integer,
ByRef *Top* As Integer, ByRef *Width* As Integer, ByRef *Height* As Integer)

Parameters

Sequence	String expression containing the name of a vision sequence in the current project.
Object	String expression containing the name of an object in sequence *Sequence*.
Left	Integer variable that will contain left coordinate in pixels.
Top	Integer variable that will contain top coordinate in pixels.
Width	Integer variable that will contain width in pixels.
Height	Integer variable that will contain height in pixels.

See Also
VSetModelWin, VGetSearchWin, VSetSearchWin

VGetModelWin Example

```
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
```

VGetPixelXYU Method, Spel Class

Description
Retrieves pixel X, Y, and U coordinates for any object.

Syntax
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)

Parameters

Sequence	String expression containing the name of a vision sequence in the current project.
Object	String expression containing the name of an object in sequence *Sequence*.
Result	Integer expression representing the result number.
Found	Boolean variable that will contain whether or not the object was found.
X	Real variable that will contain x coordinate in pixels.
Y	Real variable that will contain y coordinate in pixels.
U	Real variable that will contain the angle in degrees.

See Also
VGetCameraXYU, VGetRobotXYU

VGetPixelXYU Example
```
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)
```

VGetRobotPlacePos Method, Spel Class

Description
Retrieves robot place position.

Syntax
Sub **VGetRobotPlacePos** (*Sequence* As String, *Object* As String, *Result* As Integer, ByRef *Found* As Boolean, ByRef *PlacePoint*As SpelPoint)

Parameters

Sequence	String expression containing the name of a vision sequence in the current project.
Object	String expression containing the name of an object in sequence *Sequence*.
Result	Integer expression representing the result number.
Found	Integer variable that will contain boolean found status. If found is false, then *x, y,* and *u* are undefined.
PlacePoint	SpelPoint variable that will contain the place position

See Also
VGetRobotToolXYU

VGetRobotPlacePos Example
```
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"
' Move part above upward camera
m_spel.Jump("camPos")
m_spel.VRun(seq)
m_spel.VGetRobotPlacePos(seq, blob, 1, found, placePoint)
' Using a SCARA, to use +TLZ for approach
m_spel.Jump(placePoint, "+TLZ(10)")
m_spel.Go(placePoint)
```

VGetRobotXYU Method, Spel Class

Description
Retrieves robot world X, Y, and U coordinates for any object.

Syntax
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)

Parameters

Sequence	String expression containing the name of a vision sequence in the current project.
Object	String expression containing the name of an object in sequence *Sequence*.
Result	Integer expression representing the result number.
Found	Integer variable that will contain boolean found status. If found is false, then *x, y,* and *u* are undefined.
X	Real variable that will contain x coordinate in millimeters.
Y	Real variable that will contain y coordinate in millimeters.
U	Real variable that will contain the angle in degrees.

See Also
VGetCameraXYU, VGetPixelXYU

VGetRobotXYU Example
```
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)
```

VGetRobotToolXYU Method, Spel Class

Description
Retrieves robot world X, Y, and U values for tool definition.

Syntax
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)

Parameters

Sequence	String expression containing the name of a vision sequence in the current project.
Object	String expression containing the name of an object in sequence *Sequence*.
Result	Integer expression representing the result number.
Found	Integer variable that will contain boolean found status. If found is false, then *x, y,* and *u* are undefined.
X	Real variable that will contain x coordinate in millimeters.
Y	Real variable that will contain y coordinate in millimeters.
U	Real variable that will contain the angle in degrees.

Remarks
Use VGetRobotToolXYU to easily define a tool for a part viewed by an upward camera. This allows you to pick up a part, search for it in the upward camera FOV, define a tool for the part, then place the part.

See Also
VGetCameraXYU, VGetPixelXYU, VGetRobotPlacePos, VGetRobotXYU

VGetRobotToolXYU Example
```
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"
' Move part above upward camera
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)
```

VGetSearchWIn Method, Spel Class

Description
Retrieves search window coordinates.

Syntax
Sub **VGetSearchWin** (*Sequence* As String, *Object* As String, ByRef *Left* As Integer,
ByRef *Top* As Integer, ByRef *Width* As Integer, ByRef *Height* As Integer)

Parameters

Sequence	String expression containing the name of a vision sequence in the current project.
Object	String expression containing the name of an object in sequence *Sequence*.
Left	Integer variable that will contain left coordinate in pixels.
Top	Integer variable that will contain top coordinate in pixels.
Width	Integer variable that will contain width in pixels.
Height	Integer variable that will contain height in pixels.

See Also
VGetModelWin, VSetModelWin, VSetSearchWin

VGetSearchWin Example

```
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
```

VGoCenter Method, Spel Class

Description
Using a feature point that can be detected by the vision system, moves the robot to a position where the feature point is on the center of the camera image.

Syntax
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)

Parameters

Sequence	String expression representing a vision sequence name of current project.
LocalNumber	Integer representing the local coordinate number where the robot is moved. If -1 is specified, the robot moves in the XY plane of the tool rotation
TargetTolerance	Real value representing a pixel distance to consider that the vision detection result matches the target position. Value range: 0 to 3 pixels
Form	Parent .NET form of a window (optional)
RobotSpeed	Optional. Integer variable that will contain the robot speed (%). Value range: 0 to 100 If omitted, set to “5”.
RobotAccel	Optional. Integer variable that will contain the robot acceleration (%). Value range: 0 to 99 If omitted, set to “5”.

See Also
VDefArm, VDefGetMotionRange, VDefLocal, VDefSetMotionRange, VDefTool

VGoCenter Example
```
m_spel.VGoCenter("myseq", 1, 1.0)
```

VLoad Method, Spel Class

Description
Loads vision properties from the current project.

Syntax
Sub **VLoad** ()

Remarks
Use the VLoad method when you want to return the vision property settings, models, and fonts back to their original settings when the program was started.

See Also
VSave

VLoad Example
```
m_spel.VLoad()
```

VLoadModel Method, Spel Class

Description
Load a vision model from a disk file.

Syntax
Sub **VLoadModel** (*Sequence* As String, *Object* As String, *Path* As String)

Parameters

Sequence	String containing the name of a sequence in the current project.
Object	String containing the name of an object. The object must be a Correlation, Geometric, or Polar.
Path	Full path name of the file to load the model from, excluding extension.

Remarks
An error will occur if the model data in the file is the wrong type. For example, if you try to load a polar model into a correlation, an error will occur.
If you supply a file extension, it is ignored. There are two files associated with fileName.

For correlation and geometric models, the ModelOrgX and ModelOrgY values are restored along with the model window width and height.
For polar models, the Radius, Thickness, and AngleOffset are restored.

See Also
VSaveModel

VLoadModel Example
```
m_spel.VLoadModel("seq01", "corr01", "d:\models\part1")
```

VRun Method, Spel Class

Description
Run a vision sequence in the current project.

Syntax
Sub **VRun** (*Sequence* As String)

Parameters
Sequence String containing the name of a sequence in the current project.

Remarks
VRun works with sequences using any type of camera calibration or no calibration.

To display graphics, you need to use a SPELVideo control and set the SpelVideoControl property of the Spel class instance to the SPELVideo control.

After you execute VRun, use VGet to retrieve results.

See Also
VGet, VSet

VRun Example
```
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", SpelVisionProps.CameraX, x)
      .VGet("seq01", "corr01", SpelVisionProps.CameraY, y)
      .VGet("seq01", "corr01", SpelVisionProps.Angle, angle)
      FindPart = True
    End If
  End With
End Function
```

VSave Method, Spel Class

Description
Saves all vision data in the current project.

Syntax
Sub **VSave** ()

Remarks
Use **VSave** to make any changes to vision properties permanent.

See Also
VSet

VSave Example
```
With m_spel
 .VSet("seq01", "blob01", SpelVisionProps.SearchWinLeft, 100)
 .VSet("seq01", "corr01", SpelVisionProps.Accept, userAccept)
 .VSave()
End With
```

VSaveImage Method, Spel Class

Description
Save a vision video window to a PC disk file.

Syntax
Sub **VSaveImage** (*Sequence* As String, *Path* As String)
Sub **VSaveImage** (*Sequence* As String, *Path* As String, *WithGraphics* As Boolean)

Parameters

Sequence String containing the name of a sequence in the current project.

Path Full path name of the file to save the image to, including the extension.

WithGraphics Boolean expression that sets whether to save the sequence result graphics in the image file.

Remarks
Use VSaveImage to save an image on the Video display to disk. The file extension must be MIM (default format for Vision Guide), BMP, TIF, or JPG.

See Also
LoadImage (SPELVideo Control)

VSaveImage Example
```
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
```

VSaveModel Method, Spel Class

Description
Save a vision object search model to a PC disk file.

Syntax
Sub **VSaveModel** (*Sequence* As String, *Object* As String, *Path* As String)

Parameters

Sequence	String containing the name of a sequence in the current project.
Object	String containing the name of an object. The object must be a Correlation, Geometric, or Polar.
Path	Full path name of the file to save the model to, excluding the extension.

Remarks
When **VSaveModel** is executed, EPSON RC+ 7.0 creates two files (*Path* + extensions):
*Path***.VOB**, *Path***.MDL**
For correlation and geometric models, the ModelOrgX and ModelOrgY values are saved along with the model window.
For Polar models, the Radius, Thickness, and AngleOffset are saved.

See Also
VLoadModel

VSaveModel Example
```
m_spel.VSaveModel("seq01", "corr01", "d:\models\part1")
```

VSet Method, Spel Class

Description
Sets the value of a vision sequence or object property.

Syntax
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 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 Double)
Sub **VSet** (*Sequence* As String, *Object* As String, *PropCode* As SpelVisionProps, *Value* As String)

Parameters

Sequence	String expression containing the name of a vision sequence in the current project.
Object	String expression containing the name of an object in sequence *Sequence*. If the property is for a sequence, then this string must be empty.
PropCode	A SpelVisionProps value that specifies the property code.
Value	Expression containing the new value. The expression type must match the property type.

See Also
VGet, VRun

VSet Example
```
m_spel.VSet("seq01", "corr01", SpelVisionProps.Accept, 250)
```

VSetModelWin Method, Spel Class

Description
Sets model window coordinates.

Syntax
Sub **VSetModelWin** (*Sequence* As String, *Object* As String, *Left* As Integer, *Top* As Integer, *Width* As Integer, *Height* As Integer)

Parameters

Sequence	String expression containing the name of a vision sequence in the current project.
Object	String expression containing the name of an object in sequence *Sequence*.
Left	Integer expression representing left coordinate in pixels.
Top	Integer expression representing top coordinate in pixels.
Width	Integer expression representing width in pixels.
Height	Integer expression representing height in pixels.

See Also
VGetModelWin, VGetSearchWin, VSetSearchWin

VSetModelWin Example
```
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
```

VSetSearchWin Method, Spel Class

Description
Sets search window coordinates.

Syntax
Sub **VSetSearchWin** (*Sequence* As String, *Object* As String, *Left* As Integer, *Top* As Integer, *Width* As Integer, *Height* As Integer)

Parameters

Sequence	String expression containing the name of a vision sequence in the current project.
Object	String expression containing the name of an object in sequence *Sequence*.
Left	Integer expression representing left coordinate in pixels.
Top	Integer expression representing top coordinate in pixels.
Width	Integer expression representing width in pixels.
Height	Integer expression representing height in pixels.

See Also
VGetModelWin, VSetModel, VGetSearchWin

VSetSearchWin Example
```
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
```

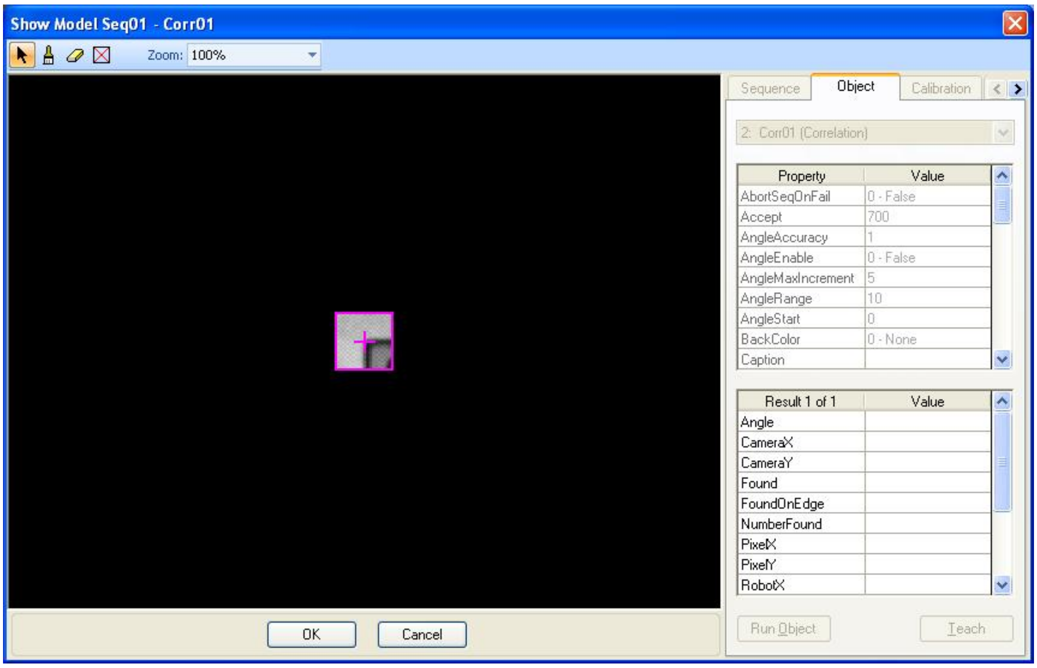
VShowModel Method, Spel Class

Description
Display the object model. For more details, see the ShowModel Property in the Vision Guide Properties reference.

Syntax
Sub **VShowModel** (*Sequence* As String, *Object* As String)

Parameters

Sequence	String expression containing the name of a vision sequence in the current project.
Object	String expression containing the name of a vision object in the current project.



See Also
VShowSequence, VTrain

VShowModel Example
```
m_spel.VShowModel("myseq", "myobj")
```

VShowSequence Method, Spel Class

Description
Displays all objects in a sequence.

Syntax
Sub **VShowSequence** (*Sequence* As String)

Parameters

Sequence	String expression containing the name of a vision sequence to create.

Remarks
Use VShowSequence to display the objects in a sequence without running the sequence. The active object color (magenta) is used for all objects so that they can be seen easily. One use is for when a robot camera is moved over a particular portion of a part being scanned with several sequences. After the robot is positioned, VShowSequence can be called to display the sequence.

See Also
VShowModel

VShowSequence Example
```
m_spel.VShowSequence("myseq")
```

VStatsReset Method, Spel Class

Description
Resets vision statistics for a specified sequence in the current project.

Syntax
Sub **VStatsReset** (*Sequence* As String)

Parameters
Sequence String expression containing the name of a vision sequence in the current project.

Remarks
VStatsReset resets the statistics for the specified sequence in memory only for the current EPSON RC+ 7.0 session. You should execute VStatsSave if you want changes to be permanent. Otherwise, if you restart EPSON RC+ 7.0, the statistics are restored from disk.

See Also
VStatsResetAll, VStatsShow, VStatsSave

VStatsReset Example
```
Sub btnResetStats_Click()
    m_spel.VStatsReset("seq01")
End Sub
```

VStatsResetAll Method, Spel Class

Description
Resets vision statistics for all sequences.

Syntax
Sub **VStatsResetAll**

Remarks
VStatsResetAll resets the statistics in memory only for the current EPSON RC+ 7.0 session. You should execute VStatsSave if you want changes to be permanent. Otherwise, if you restart EPSON RC+ 7.0, the statistics are restored from disk.

See Also
VStatsReset, VStatsShow, VStatsSave

VStatsResetAll Example
```
Sub btnResetStats_Click()
    m_spel.VStatsResetAll()
End Sub
```

VStatsSave Method, Spel Class

Description
Saves vision statistics for all sequences in the current project.

Syntax
Sub **VStatsSave ()**

Remarks
VStatsSave must be executed before EPSON RC+ 7.0 is shut down if you want to preserve changes made to vision statistics.

See Also
VStatsReset, VStatsResetAll, VStatsShow

VStatsSave Example
```
Sub btnResetStats_Click()
    m_spel.VStatsSave()
End Sub
```

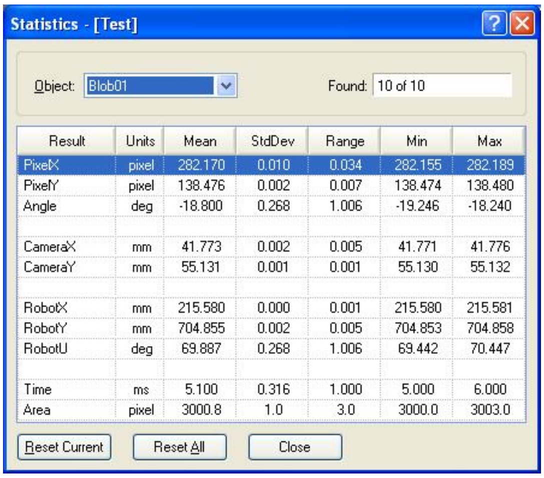
VStatsShow Method, Spel Class

Description
Displays the vision statistics dialog for a specified sequence in the current project.

Syntax
Sub **VStatsShow** (*Sequence* As String)

Parameters

Sequence	String expression containing the name of a vision sequence in the current project.



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

See Also
VStatsReset, VStatsResetAll, VStatsSave

VStatsShow Example
```
Sub btnShowStats_Click()
    m_spel.VStatsShow("seq01")
End Sub
```

VTeach Method, Spel Class

Description
Teach a correlation, geometric, or polar model.

Syntax
Sub **VTeach** (*Sequence* As String, *Object* As String, ByRef *Status* as Integer)

Parameters

Sequence The name of a vision sequence in the current project.

Object The name of an object in *Sequence*. You can teach Correlation, Geometric, or Polar objects.

Status Return status. 1 if successful, 0 if not.

Remarks
Before you call **VTeach**, you must ensure that the model window is in the correct position.
For polar objects, the search window and thickness must be set properly. Set the search window location and thickness using VSet.
For correlation and geometric objects, the search window and the model window must be set properly. Set the search and model window locations using VSet for SearchWin and ModelWin. Or you can use the VTrain command so the operator can interactively change the windows.
After teaching the models, you can save them to a PC disk file using the VSaveModel method.

See Also
VTrain, VSaveModel

VTeach Example
```
Dim status As Integer

' First let the operator change the window position
m_spel.VTrain("seq01", "corr01", status)

' Now teach the model
m_spel.VTeach("seq01", "corr01", status)
```

VTrain Method, Spel Class

Description
This command allows you to train objects in an entire sequence or individual objects.

Syntax
Sub **VTrain** (*Sequence* As String [, *Object* As String] [, *Flags* as Integer])

Parameters

Sequence	The name of a vision sequence in the current project.
Object	The name of an object in *Sequence*. You can train any type of object. If *Object* is an empty string, then the entire sequence can be trained.
Flags	Optional. Configures VTrain dialog 1 - Show Teach button 2 - Don't show Model windows.

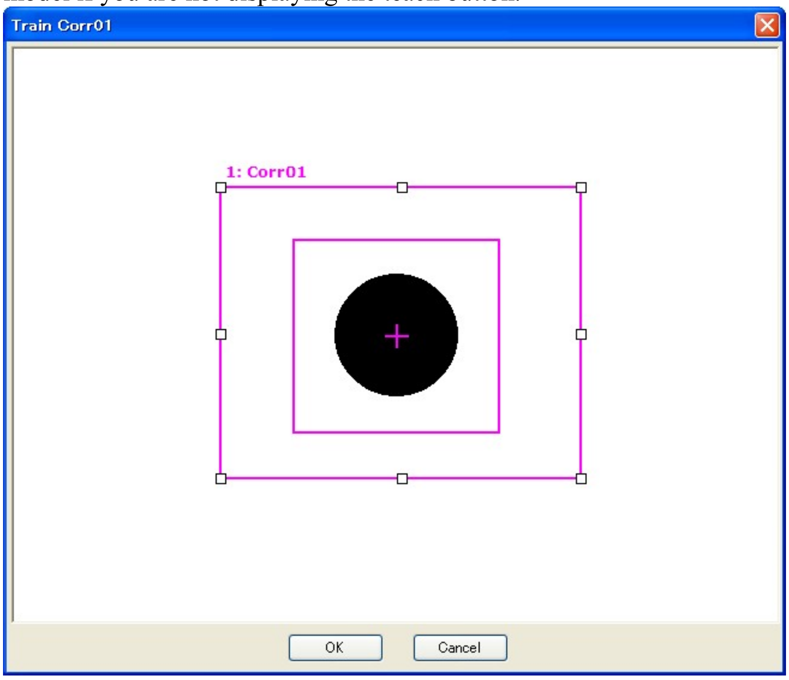
Return Values
If the operator clicks the OK button, VTrain returns True, otherwise it returns False.

Remarks
When **VTrain** is executed, a dialog is opened showing a live video image with the specified object displayed. The operator can size/move the search window, and train the model window (for correlation and geometric objects). When the operator is finished, he can click on OK to save the changes, or Cancel to ignore the changes. If OK is clicked, then the new information is automatically saved in the current project.

If *flags* bit 1 is set, a teach button will be displayed. For Correlation, Geometric, and Polar objects, the model will be taught if the teach button is clicked. You can retrieve the ModelOK property after running VTrain to check if a model was trained. For Blob objects, the button will open the Histogram dialog and the operator can adjust both high and low thresholds and then view the effects of changes.

If *flags* bit 2 is set, model windows will not be displayed. The operator can only change search windows.

For correlation and geometric objects, you can call VTeach after calling **VTrain** to teach the model if you are not displaying the teach button.



See Also
VTeach, VSaveModel

VTrain Example

```
Dim status As Integer

' First let the operator change the window position
m_spel.VTrain("seq01", "corr01")

' Now teach the model
m_spel.VTeach("seq01", "corr01", status)
```

WaitCommandComplete Method, Spel Class

Description
This command waits for a command started with AsyncMode = True to complete.

Syntax
Sub **WaitCommandComplete** ()

See Also
AsyncMode

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

WaitMem Method, Spel Class

Description
Waits for a memory bit status to change.

Syntax
Sub **WaitMem** (*BitNumber* As Integer, *Condition* As Boolean, *Timeout* As Single)

Parameters

BitNumber	Integer expression representing the memory bit number.
Condition	Boolean expression representing the memory bit status.
Timeout	Single expression representing the maximum time to wait in seconds.

Remarks
You should always check if a time out occurred by using the TW method. See the example below.

See Also
WaitSw

WaitMem Example
```
' Wait for memory bit 1 to be 1 (True)
' Max time is 5 seconds
m_spel.WaitMem(1, True, 5)
' Did WaitMem time out?
If m_spel.TW() Then
    MsgBox "memory bit time out occurred"
End If
```

WaitSw Method, Spel Class

Description
Waits for input bit status to change.

Syntax
Sub **WaitSw** (*BitNumber* As Integer, *Condition* As Boolean, *Timeout* As Single)

Parameters

BitNumber	Integer expression representing the input bit number.
Condition	Boolean expression representing the input bit status.
Timeout	Single expression representing the maximum time to wait in seconds.

Remarks
You should always check if a time out occurred by using the TW method. See the example below.

See Also
WaitMem

WaitSw Example
```
Const PartPresent = 1
m_spel.WaitSw(PartPresent, True, 5)
If m_spel.TW() Then
    MsgBox "Part present time out occurred"
End If
```

WaitTaskDone Method, Spel Class

Description
Waits for a task to finish and returns the status.

Syntax
Function **WaitTaskDone** (*TaskNumber* As Integer) As SpelTaskState
Function **WaitTaskDone** (*TaskName* As String) As SpelTaskState

Parameters

TaskNumber	Task Number to return the execution status of.
TaskName	String expression containing the name of the task.

Return Value
A SpelTaskState value.

See Also
SpelTaskState, TasksExecuting, TaskState, Xqt

WaitTaskDone Example
```
Dim taskState As SpelTaskState
m_spel.Xqt 2, "mytask"
…
taskState = m_spel.WaitTaskDone(2)
```

Weight Method, Spel Class

Description
Specifies the weight parameters for the current robot.

Syntax
Sub **Weight** (*PayloadWeight* As Single, *ArmLength* As Single)
Sub **Weight** (*PayloadWeight* As Single, *Axis* As SpelAxis, [Axis])

Parameters

PayloadWeight	The weight of the end effector to be carried in Kg units.
ArmLength	The distance from the rotational center of the second arm to the center of the gravity of the end effector in mm units.
Axis	Specifies which additional axis (S or T) is assign the payload weight.

Note
Do not enter integer values to PayLoadWeight and ArmLength parameters. Use Single variables or directly enter Single type values.

See Also
Inertia, JRange, Tool

Weight Example
```
m_spel.Weight(2.0F, 2.5F)
```

Xqt Method, Spel Class

Description
Start one SPEL+ task.

Syntax
Sub **Xqt** (*FuncName* As String [, *TaskType* As SpelTaskType])
Sub **Xqt** (*TaskNumber* As Integer, *FuncName* As String [, *TaskType* As SpelTaskType])

Parameters

TaskNumber	The task number for the task to be executed. The range of the task number is 1 to 32.
FuncName	The name of the function to be executed. You can also optionally supply arguments to the function. Arguments must be in parenthesis, separated by commas. For details, see the SPEL+ Xqt Statement. Also, see the example.
TaskType	Optional. Specifies the task type as Normal, NoPause, or NoEmgAbort.

Remarks
When **Xqt** is executed, control will return immediately to the calling program. Use the Call method to wait for a task to complete, or you can use EventReceived with the task state event to wait for a task to finish.

See Also
Call, EnableEvent, EventReceived

Xqt Example
```
m_spel.Xqt(2, "conveyor")

' Supply an argument to the RunPart function
m_spel.Xqt(3, "RunPart(3)")

Dim funcCall As String
funcCall = "RunPart(" & partNum & ")"
m_spel.Xqt(3, funcCall)
```

XYLim Method, Spel Class

Description
Sets the permissible motion range limits for the manipulator.

Syntax
Sub **XYLim** (*XLowerLimit* As Single, *XUpperLimit* As Single, *YLowerLimit* As Single, *YUpperLimit* As Single [, *ZLowerLimit* As Single] [, *ZUpperLimit* As Single])

Parameters
XLowerLimit	The minimum X coordinate position to which the manipulator may travel. (The manipulator may not move to a position with the X Coordinate less than minX.)
XUpperLimit	The maximum X coordinate position to which the manipulator may travel. (The manipulator may not move to a position with the X Coordinate greater than maxX.)
YLowerLimit	The minimum Y coordinate position to which the manipulator may travel. (The manipulator may not move to a position with the Y Coordinate less than minY.)
YUpperLimit	The maximum Y coordinate position to which the manipulator may travel. (The manipulator may not move to a position with the Y Coordinate greater than maxY.)
ZLowerLimit	Optional. The minimum Z coordinate position to which the manipulator may travel. (The manipulator may not move to a position with the Z Coordinate less than minZ.)
ZUpperLimit	Optional. The maximum Z coordinate position to which the manipulator may travel. (The manipulator may not move to a position with the Z Coordinate greater than maxZ.)

Remarks
XYLim is used to define motion range limits. Many robot systems allow users to define joint limits but the SPEL+ language allows both joint limits and motion range limits to be defined. In effect this allows users to create a work envelope for their application. (Keep in mind that joint range limits are also definable with SPEL.)

The motion range established with XYLim values applies to motion command target positions only, and not to motion paths from starting position to target position. Therefore, the arm may move outside the XYLim range during motion. (i.e. The XYLim range does not affect Pulse.)

To turn off motion range limits, specify 0 for the range limit parameters.

See Also
JRange

XYLim Example
```
m_spel.XYLim(0, 0, 0, 0)
```

XYLimClr Method, Spel Class

Description
Clears (undefines) the XYLim definition.

Syntax
Sub **XYLimClr** ()

See Also
XYLim, XYLimDef

XYLimClr Example
```
m_spel.XLLimClr()
```

XYLimDef Method, Spel Class

Description
Returns whether XYLim has been defined or not.

Syntax
Function **XYLimDef** () As Boolean

Return Value
True if XYLim is defined, False if not.

See Also
XYLim, XYLimClr

XYLimDef Example
```
m_spel.XLLimDef()
```

14.4 Spel Class Events

EventReceived Event, Spel Class

Description

Occurs when EPSON RC+ 7.0 sends a system event or when a program running in SPEL$^+$ sends an event using a SPELCom_Event statement.

Syntax

EventReceived (ByVal *sender* As Object, ByVal *e* As RCAPINet.SpelEventArgs)

Parameters

e.Event	Number representing a specific user-defined event.
e.Message	String containing event message.

Remarks

There are several system events that EPSON RC+ 7.0 issues. The following table describes them.

System Events

Some events are disabled by default. To use these events you must first enable them using the EnableEvent Method.

Event Number	Event Message	Constant	Description
1	"PAUSE"	SpelEvents.Pause	Occurs when tasks are paused. Enabled by default.
2	"SAFE GUARD OPEN"	SpelEvents.SafeGuardOpen	Occurs when safe guard is open. Enabled by default.
3	"SAFE GUARD CLOSE"	SpelEvents.SafeGuardClose	Occurs when safe guard is closed. Enabled by default.
4	Project build status text	SpelEvents.ProjectBuildStatus	Each build status message is sent during the BuildProject method. CRLFs are added as needed. These messages are the same ones displayed on the Project Build Status window in EPSON RC+ 7.0 GUI. This event must be enabled with the EnableEvent method. Disabled by default.
5	"Error xxx!: mmm in task at line yyy"	SpelEvents.Error	Occurs when a task is aborted due to an unhandled error or a system error is generated. Enabled by default.
6	Text from print statement	SpelEvents.Print	Occurs when a Print statement executes from a SPEL$^+$ task. Disabled by default.
7	"ESTOP ON"	SpelEvents.EStopOn	Occurs when emergency stop condition changes to ON. Enabled by default.
8	"ESTOP OFF"	SpelEvents.EStopOff	Occurs when emergency stop condition changes to OFF. Enabled by default.

Event Number	Event Message	Constant	Description
9	"CONTINUE"	SpelEvents.Continue	Occurs after a Cont has been executed. Enabled by default.
10	<Robot #>,"MOTOR ON"	SpelEvents.MotorOn	Occurs when motors go ON for the robot indicated. Disabled by default.
11	<Robot #>,"MOTOR OFF"	SpelEvents.MotorOff	Occurs when motors go OFF for the robot indicated. Disabled by default.
12	<Robot #>,"POWER HIGH"	SpelEvents.PowerHigh	Occurs when power goes HIGH for the robot indicated. Disabled by default.
13	<Robot #>,"POWER LOW"	SpelEvents.PowerLow	Occurs when power goes LOW for the robot indicated. Disabled by default.
14	"TEACH MODE"	SpelEvents.TeachMode	Occurs when teach mode is activated. Enabled by default.
15	"AUTO MODE"	SpelEvents.AutoMode	Occurs when auto mode is activated. Enabled by default.
16	"<TaskID>,<Status>, <FuncName>" Status: "RUN", "HALT", "PAUSE", “FINISHED”, “ABORTED”	SpelEvents.TaskState	Occurs when task state changes. Disabled by default.
17	"SHUTDOWN"	SpelEvents.Shutdown	Occurs when RC+ is shutting down. Disabled by default.
18	"ALL TASKS STOPPED"	SpelEvents.AllTasksStopped	Occurs when all tasks have been stopped. Disabled by default.
19	"DISCONNECTED"	SpelEvents.Disconnected	Occurs when controller communication has been disconnected from the PC. When enabled, RC+ does not display a message box indicating disconnection. Disabled by default.

User Events

You can send events from your SPEL+ program to your Visual Basic application using the **SPELCom_Event** command.

```
Spelcom_Event 3000, cycNum, lotNum, cycTime
```

When this statement executes, the EventReceived routine will be called with the event number and message. See *EPSON RC+ 7.0 Online Help* or *13. SPELCom_Event* for details on SPELCom_Event.

Use Example

```
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
```

Handling Events

When **EventReceived** is called from the Spel class instance, the EPSON RC+ 7.0 process server will wait for the event handling routine to finish. Therefore, you should never try to execute any RC+ API commands or perform long running processing from within the **EventReceived** routine. If you want to execute commands based on an event that occurred, set a flag in **EventReceived** and handle the flag from the main loop of your application, outside of the event handling function.

For example, in your Visual Basic main form Load procedure, you can create an event loop that receives events from SPEL+. In the spel_EventReceived routine, set global flags to indicate which events were received. Then, you can execute an actual event handling from the event loop created in Load procedure.

To display event message

Add a TextBox control to a form.

Each time the event is received, you can display the event message in the Text property of the TextBox control.

```
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
```

See Also

EnableEvent (Spel Class)

14.5 SPELVideo Control

Description

This control allows you to display video from Vision system. For details on how to use this control, see chapter 11, *Displaying Video*.

File Name

RCAPINet.dll

14.6 SPELVideo Control Properties

This control supports the properties listed below in addition to standard .NET component properties, such as Left, Top, Width, and Height. See the Visual Basic on-line Help for documentation on the standard properties.

- Camera
- GraphicsEnabled
- VideoEnabled

Camera Property, SPELVideo Control

Description

Sets/gets the camera number to display video from. This is useful when you want to display video during jogging operations, live video monitoring, etc. If you are using the control to display graphics for vision sequences, then when the sequence is run, the camera number for the sequence will be used instead of this property value.

Syntax

Property **Camera** As Integer

Default Value

0 – any camera is displayed

Return value

Integer value containing the current camera number

See Also

VideoEnabled, GraphicsEnabled

Examples

```
SpelVideo1.Camera = 1
```

GraphicsEnabled Property, SPELVideo Control

Description

Sets / returns whether vision graphics are displayed after a sequence is run. In order to see graphics, you must attach the control to a Spel class instance using the SPELVideo Control property. This property can be set "on the fly" so that graphics can be turned on/off while sequences are being run.

Syntax

Property **GraphicsEnabled** As Boolean

Default Value

False

Return value

True if vision graphics are displayed, False if not.

See Also

Camera, VideoEnabled

Examples

```
SpelVideo1.GraphicsEnabled = True
```

VideoEnabled Property, SPELVideo Control

Description

Determines whether video is displayed.

Syntax

Property **VideoEnabled** As Boolean

Default Value

False

Return value

True if video is displayed, False if not.

See Also

Camera, GraphicsEnabled

Examples

```
SpelVideo1.VideoEnabled = True
```

14.7 SPELVideo Control Methods

LoadImage Method, SPELVideo Control

Description
Loads an image from a file for display.

Syntax
Sub **LoadImage** (*Path* As String)

Parameters

Path	Full path name of the file to load the image from, including the extension.

Remarks
Use LoadImage to load a previously saved image for display. The file extension must be BMP, TIF, or JPG.

See Also
VSaveImage (Spel class)

LoadImage Example
```
m_spelVideo.LoadImage("c:\RejectImages\reject001.bmp")
```

14.8 SPELVideo Control Events

All of the events for this control are standard .NET events. See the Visual Basic on-line Help for details.

14.9 SpelConnectionInfo Class

Member name	Type	Description
ConnectionNumber	Integer	The number of the connection as configured in EPSON RC+.
ConnectionName	String	The name of the connection as configured in EPSON RC+.
ConnectionType	SpelConnectionType	The type of the connection as configured in EPSON RC+.

Here is an example.

```
Dim connectionInfo() As RCAPINet.SpelConnectionInfo
connectionInfo = m_spel.GetConnectionInfo()
```

14.10 SpelControllerInfo Class

Member name	Type	Description
ProjectName	String	The name of the project in the controller.
ProjectID	String	The unique project ID of the project in the controller.

Here is an example.

```
Dim info As RCAPINet.SpelControllerInfo
info = m_spel.GetControllerInfo()
Label1.Text = info.ProjectID + " " + info.ProjectName
```

14.11 SpelException Class

The SpelException class is derived from the ApplicationException class. It adds an ErrorNumber property and some constructors.

Here is an example, showing how to retrieve the error number and the error message.

```
Try
  m_spel.Go(1)
Catch (ex As RCAPINet.SpelException)
  MsgBox(ex.ErrorNumber & " " & ex.Message)
End Try
```

SpelException Properties

ErrorNumber As Integer

SpelException Methods

Sub New ()

The default constructor.

Sub New (Message As String)

The optional constructor that specifies an error message.

Sub New (ErrorNumber As Integer, Message As String)

The optional constructor that specifies the error number and associated message.

Sub New (Message As String, Inner As Exception)

The optional constructor that specifies the error message and inner exception.

Sub New (ErrorNumber As Integer, Message As String, Inner As Exception)

The optional constructor that specifies the error number, error message, and inner exception.

14.12 SpelPoint Class

The SpelPoint class can be used in several motion methods and also in the GetPoint and SetPoint methods of Spel class.

Here are some examples:

```
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)
```

14.12.1 SpelPoint Properties

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

14.12.2 SpelPoint Methods

Sub Clear ()

Clears all point data.

Sub New ()

The default constructor. Creates an empty point (all data is cleared).

Sub New (X As Single, Y As Single, Z As Single, U As Single [, V As Single] [, W As Single])

The optional constructor for a new point that specifies coordinates.

Function ToString ([Format As String]) As String

Override for ToString that allows a Format to be specified. This returns the point as defined in SPEL+.

Format can be:

Empty	Returns the entire point with all coordinates and attributes.
"XY"	Returns "XY(…)"
"XYST"	Returns "XY(…) :ST(…)"

14.13 Enumerations

14.13.1 SpelArmDefMode Enumeration

Member name	Value	Description
Rough	1	Define the arm using one posture.
Fine	1	Define the arm using two postures.

14.13.2 SpelArmDefType Enumeration

Member name	Value	Description
J2Camera	1	Define the arm for a J2 mounted camera.

14.13.3 SpelAxis Enumeration

Member name	Value	Description
X	1	X axis.
Y	2	Y axis.
Z	3	Z axis.
U	4	U axis.
V	5	V axis.
W	6	W axis.
R	7	R axis.
S	8	S axis.
T	9	T axis.

14.13.4 SpelBaseAlignment Enumeration

Member name	Value	Description
XAxis	0	Align with X axis.
YAxis	1	Align with Y axis.

14.13.5 SpelCalPlateType Enumeration

Member name	Value	Description
None	0	No calibration plate.
Large	1	Large calibration plate.
Medium	2	Medium calibration plate.
Small	3	Small calibration plate.
XSmall	4	Extra small calibration plate.

14.13.6 SpelConnectionType Enumeration

Member name	Value	Description
USB	1	USB connection.
Ethernet	2	Ethernet connection.
Virtual	3	Connection to virtual controller.

14.13.7 SpelDialogs Enumeration

Member name	Value	Description
RobotManager	1	ID for Tools | Robot Manager dialog
ControllerTools	2	ID for Tools | Controller dialog
VisionGuide	3	ID for Tools | Vision Guide dialog

14.13.8 SpelElbow Enumeration

Member name	Value	Description
Above	1	Elbow orientation is above.
Below	2	Elbow orientation is below.

14.13.9 SpelEvents Enumeration

Member name	Value	Description
Pause	1	ID for pause event.
SafeguardOpen	2	ID for safeguard open event.
SafeguardClose	3	ID for safeguard close event.
ProjectBuildStatus	4	ID for project build status event.
Error	5	ID for error event.
Print	6	ID for print event.
EstopOn	7	ID for emergency stop on event.
EstopOff	8	ID for emergency stop off event.
Continue	9	ID for continue event.
MotorOn	10	ID for motor on event.
MotorOff	11	ID for motor off event.
PowerHigh	12	ID for power high event.
PowerLow	13	ID for power low event.
TeachMode	14	ID for teach mode event.
AutoMode	15	ID for auto mode event.
TaskState	16	ID for task state event.
Shutdown	17	ID for shutdown event.
AllTasksStopped	18	ID for all tasks stopped event.

14.13.10 SpelForceAxis Enumeration

Member name	Value	Description
XForce	1	Specifies the X force axis.
YForce	2	Specifies the Y force axis.
ZForce	3	Specifies the Z force axis.
XTorque	4	Specifies the X torque axis.
YTorque	5	Specifies the Y torque axis.
ZTorque	6	Specifies the Z torque axis.

14.13.11 SpelForceCompareType Enumeration

Member name	Value	Description
LessOrEqual	0	Till is triggered when the force is less than or equal to the specified threshold.
GreaterOrEqual	1	Till is triggered when the force is greater than or equal to the specified threshold.

14.13.12 SpelHand Enumeration

Member name	Value	Description
Righty	1	Hand orientation is righty.
Lefty	2	Hand orientation is lefty.

14.13.13 SpelIOLabelTypes Enumeration

Member name	Value	Description
InputBit	1	Specifies input bit.
InputByte	2	Specifies input byte.
InputWord	3	Specifies input word.
OutputBit	4	Specifies output bit.
OutputByte	5	Specifies output byte.
OutputWord	6	Specifies output word.
MemoryBit	7	Specifies memory bit.
MemoryByte	8	Specifies memory byte.
MemoryWord	9	Specifies memory word.
InputReal	10	Specifies real number input.
OutputReal	11	Specifies real number output.

14.13.14 SpelOperationMode Enumeration

Member name	Value	Description
Auto	1	EPSON RC+ 7.0 is in auto mode.
Program	2	EPSON RC+ 7.0 is in program mode.

14.13.15 SpelRobotPosType Enumeration

Member name	Value	Description
World	0	Specifies world coordinates.
Joint	1	Specifies joint coordinates.
Pulse	2	Specifies pulses.

14.13.16 SpelRobotType Enumeration

Member name	Value	Description
Joint	1	Robot type is joint.
Cartesian	2	Robot type is Cartesian.
Scara	3	Robot type is SCARA.
Cylindrical	4	Robot type is Cylindrical.
SixAxis	5	Robot type is 6-axis.
RS	6	Robot type is SCARA RS series.

14.13.17 SpelShutdownMode Enumeration

Member name	Value	Description
ShutdownWindows	0	Windows will be shutdown.
RebootWindows	1	Windows will be rebooted.

14.13.18 SpelStopType Enumeration

Member name	Value	Description
StopNormalTasks	0	Stop only normal tasks (not background tasks).
StopAllTasks	1	Stop all tasks, including background tasks.

14.13.19 SpelTaskState Enumeration

Member name	Value	Description
Quit	0	Task is in the quit state.
Run	1	Task is in the run state.
Aborted	2	Task was aborted.
Finished	3	Task was finished.
Breakpoint	4	Task is at a breakpoint.
Halt	5	Task is in the halt state.
Pause	6	Task is in the pause state.
Step	7	Task is being stepped.
Walk	8	Task is being walked.
Error	9	Task is in the error state.
Waiting	10	Task is in the wait state.

14.13.20 SpelTaskType Enumeration

Member name	Value	Description
Normal	0	Task is a normal task.
NoPause	1	Task is not affected by pause.
NoEmgAbort	2	Task is not affected by emergency stop.

14.13.21 SpelToolDefType Enumeration

Member name	Value	Description
J4Camera	1	Define the tool for a J4 mounted camera.
J6Camera	2	Define the tool for a J6 mounted camera.
FixedCamera	3	Define the tool by using the fixed camera which is not calibrated.
FixedCameraWithCal	4	Define the tool by using the upward camera which is calibrated.

14.13.22 SpelUserRights Enumeration

Member name	Value	Description
All	-1	User has all rights.
None	0	User has no rights.
EditSecurity	1	User can configure security.
SysConfig	2	User can change system configuration.
EditPrograms	4	User can edit programs.
EditPoints	8	User can edit points.
EditVision	16	User can change vision properties.
JogAndTeach	32	User can Jog & Teach.
CommandWindow	64	User can use the command window.
EditRobotParameters	128	User can edit robot parameters.
ConfigureOptions	256	User can configure options.
ViewAudit	512	User can view the security audit log.
EditProject	1024	User can edit the project configuration.
DeleteAudit	2048	User can delete security audit log entries.
TeachPoints	4096	User can teach points.
ChangeOutputs	8192	User can change output status.
ChangeMemIO	16384	User can change memory I/O status.
EditGUIBuilder	32768	User can make changes in GUI Builder.
EditForce	65536	User can make changes in Force Guide and Force Control.
EditPartFeeding	131072	User can make changes in Part Feeding.

14.13.23 SpelVDefShowWarning Enumeration

Member name	Value	Description
None	-1	Do not display a warning.
Always	0	Always display a warning.
DependsOnSpeed	1	Display when either RobotSpeed or RobotAccel is larger than 5.

14.13.24 SpelVisionImageSize Enumeration

Member name	Value	Description
Size320x240	1	320 x 240 image size.
Size640x480	2	640 x 480 image size.

Size800x600	3	800 x 600 image size.
Size1024x768	4	1024 x 768 image size.
Size1280x1024	5	1280 x 1024 image size.
Size1600x1200	6	1600 x 1200 image size.
Size2048x1536	7	2048 x 1536 image size.
Size2560x1920	8	2560 x 1920 image size.
Size3664x2748	9	3664 x 2748 image size.

14.13.25 SpelVisionObjectTypes Enumeration

Member name	Value	Description
Correlation	1	Correlation object.
Blob	2	Blob object.
Edge	3	Edge object.
Polar	4	Polar object.
Line	5	Line object.
Point	6	Point object.
Frame	7	Frame object.
ImageOp	8	ImageOp object.
OCR	9	OCR object.
CodeReader	10	CodeReader object.
Geometric	11	Geometric object.
ColorMatch	14	Color Match object.
LineFinder	15	Line Finder object.
ArcFinder	16	Arc Finder object.
DefectFinder	17	Defect Finder object.
LineInspector	18	Line Inspector object.
ArcInspector	19	Arc Inspector object.
BoxFinder	20	Box Finder object
CornerFinder	21	Corner Finder object
Contour	22	Contour object
Text	23	Text object

14.13.26 SpelVisionProps Enumeration

This enumeration is for all vision properties and results. Refer to the Vision Guide Reference manual for details.

14.13.27 SpelWrist Enumeration

Member name	Value	Description
NoFlip	1	Wrist orientation is no flip.
Flip	2	Wrist orientation is flip.

14.13.28 SpelWindows Enumeration

Member name	**Value**	**Description**
IOMonitor	1	ID for the I/O Monitor window.
TaskManager	2	ID for the Task Manager window.
ForceMonitor	3	ID for the Force Monitor window.
Simulator	4	ID for the Simulator window.

14.14 Spel Error Numbers and Messages

For error numbers and error messages, see the *SPEL+ Language Reference*.

15. 32 Bit and 64 Bit Applications

The RCAPINet library provided in EPSON RC+ 7.0 version 7.1.0 and greater automatically supports 32 bit and 64 bit applications.

In versions of EPSON RC+ 7.0 prior to 7.1.0, separate libraries were supplied for 32 bit and 64 bit support. These obsolete libraries (SpelNetLib70.dll and SpelNetLib70_x64.dll) are still provided in version 7.1.0 for compatibility. For details on using the obsolete libraries, see the RC+ API manual for the previous version of EPSON RC+ 7.0 that you were using.

16. Using the LabVIEW VI Library

16.1 Overview

In versions of EPSON RC+ 7.0 prior to v7.1.0, the API .NET library could be used directly in LabVIEW. In EPSON RC+ 7.0 v7.1.0, a new LabVIEW VI library was introduced. The new library has the following features:

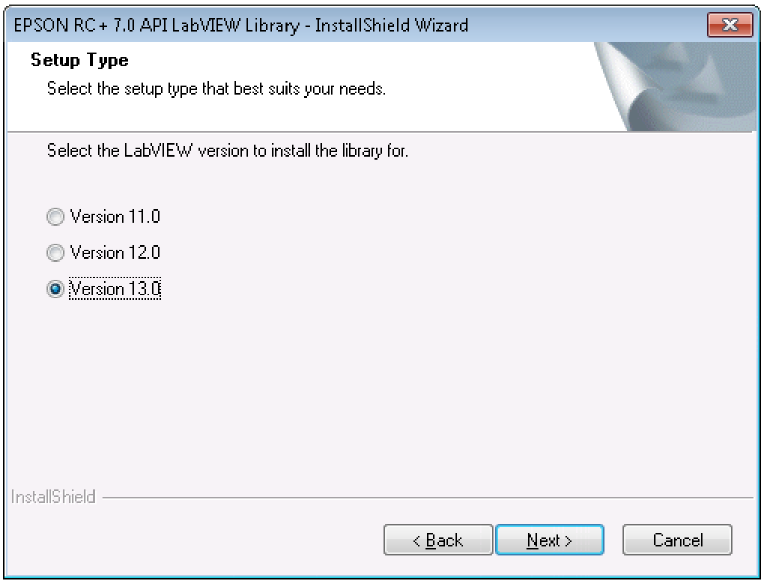
- High level interface to EPSON RC+ 7.0 using VIs (Virtual Instruments).
- The user no longer needs to deal with the .NET interface to EPSON RC+ 7.0 – it is handles automatically.
- Each Spel command is wrapped in an individual VI.
- The VIs are organized in several Tool Palettes.
- Supports both 32 bit and 64 bit LabVIEW applications.
- Supports LabVIEW versions 2009 and greater.

To use the LabVIEW VI library, you must purchase an EPSON RC+ 7.0 API software license for each controller that you connect with.

16.2 Installation

To use the EPSON RC+ 7.0 LabVIEW VI Library, you must install it using the installer provided in the \EpsonRC70\API\LabVIEW folder on your PC.

1. Install LabVIEW version 2009 or greater.
2. Navigate to the \EpsonRC70\API\LabVIEW folder on your PC and run the EpsonRC70_vxxx_LabVIEW.exe installer, where xxx is the version number for EPSON RC+ 7.0. For example, EpsonRC70_v710_LabVIEW.exe.
3. When the installer starts, it will display the detected versions of LabVIEW installed on your PC. The latest version is selected by default. Select the version that will use the EPSON RC+ 7.0 LabVIEW VI Library.



4. Click Next, then click Install. The VIs, Controls, and palettes will be installed for the selected version of LabVIEW.

16.3 Tool and Control Palettes

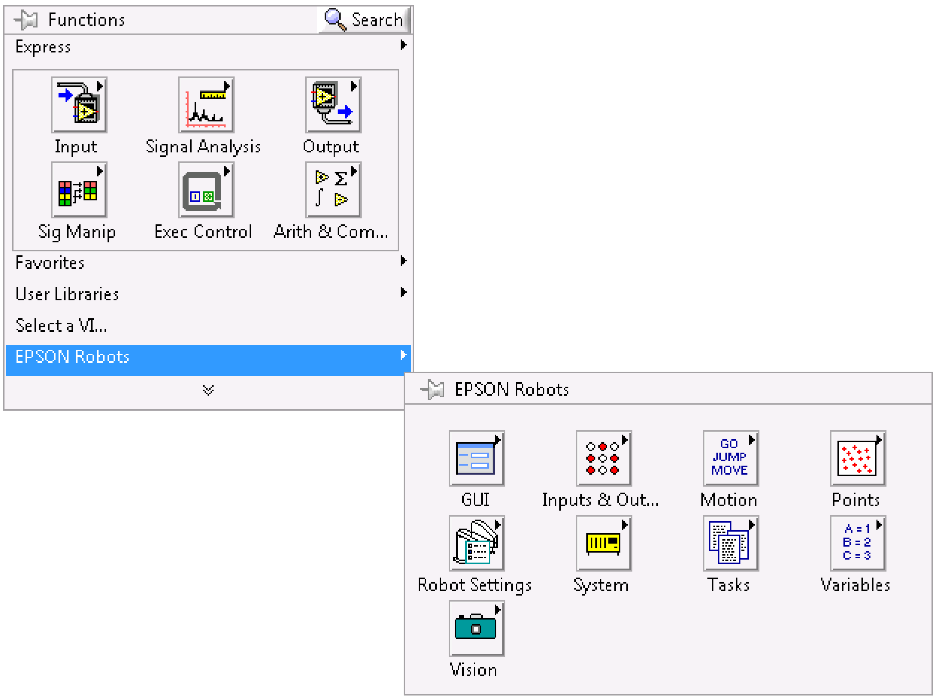
After the EPSON RC+ 7.0 LabVIEW VI Library is installed, you can access the VIs and controls available in the library from the [EPSON Robots] tool palette and [EPSON Robots] control palette.

Tool Palette

The tool palette is accessed from the block diagram. Inside the Epson Robots tool palette are several sub-palettes, described in the following table:

Palette	Description
System	Used to initialize and shutdown the API.
Robot Settings	Change robot parameters.
Points	Load, save, change robot points.
Motion	Execute robot motion.
Inputs & Outputs	Control and monitor controller inputs and outputs.
Tasks	Manage tasks in the robot controller.
Variables	Read and write variables in the controller.
Vision	Execute vision commands.
GUI	Display GUI functions.

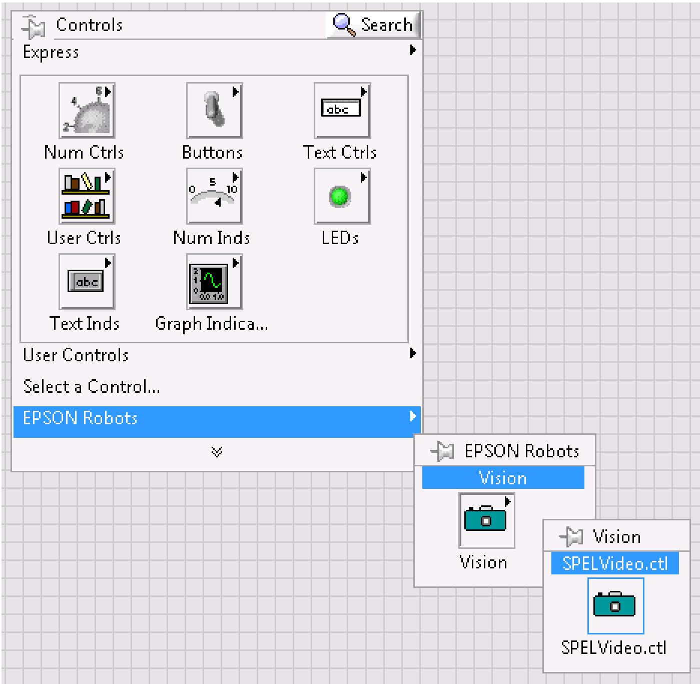
To access the [Epson Robots] tool palette, open the block diagram for your VI, then right click on an empty area and select [Epson Robots] to see the sub-palettes described above.



Control Palette

The control palette is accessed from the front panel.

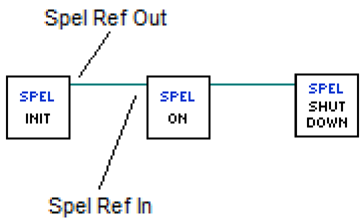
Palette	Description
Vision	Contains the SPELVideo control used to display video.



16.4 Getting started

To use the LabVIEW VI library, your application must first call the Spel Initialize VI for each controller you want to use. The Initialize VI starts an EPSON RC+ 7.0 server process that will connect to the robot controller and process the subsequent Spel command VIs. The Initialize VI has a Spel Ref Out output. This must be wired to the next Spel VI Spel Ref In input. Then for each subsequent VI, the Spel Ref Out output from a previous Spel VI must be wired to the Spel Ref In input of the next Spel VI.

For example, the flow diagram below shows the wires for Spel Ref Out and Spel Ref In between each Spel node.



When the application is shutting down, you must call the Spel Shutdown VI. This will disconnect from the robot controller and shutdown the associated EPSON RC+ 7.0 server process.

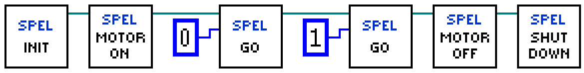
Follow these steps to get started. First, you will create two safe robot points from within the EPSON RC+ 7.0 GUI for the LabVIEW default Spel+ project. Then you will build a small application in LabVIEW to move the robot between the two points.

1. Ensure that the EPSON RC+ 7.0 and the EPSON RC+ 7.0 LabVIEW VI Library are installed on your PC. See section 16.2 for details for installing the LabVIEW VI Library.
2. Start EPSON RC+ 7.0.
3. From the Project menu, select Open, then navigate to the LabVIEW folder and select the LabVIEW_Default project. Click Open.
4. From the Tools menu, select Robot Manager. Click the Motor On button.
5. Select the Jog & Teach page on the Robot Manager. Jog the robot to some safe position.
6. Click Teach to teach point 0.
7. Jog the robot to another safe position.
8. Select P1 from the Point list, then click Teach to teach point 1.
9. Click the Save button on the main toolbar to save the points.
10. Close EPSONRC+ 7.0.
11. Start LabVIEW and create a new VI.
12. Open the block diagram for the new VI.
13. From the Epson RC+ API | System tool palette, drag the Init VI onto the block diagram. The Initialize VI is required for each controller that you interface with.
14. From the Epson RC+ API | Robot Settings tool palette, drag the MotorOn VI onto the block diagram. Wire the Spel Ref Out output from the Inititalize VI to the Spel Ref In input on the MotorOn VI.
15. From the Epson RC+ API | Motion tool palette, drag the Go VI onto the block diagram. Wire the Spel Ref Out output from the MotorOn VI to the Spel Ref In input on the Go VI. Add a constant for the Point Number input and set the value to 0.
16. From the Epson RC+ API | Motion tool palette, drag another Go VI onto the block diagram. Wire the Spel Ref Out output from the previous Go VI to the Spel Ref In

input on the second Go VI. Add a constant for the Point Number input and set the value to 1.

17. From the Epson RC+ API | Robot Settings tool palette, drag the MotorOff VI onto the block diagram. Wire the Spel Ref Out output from the Go VI to the Spel Ref In input on the MotorOff VI.

18. From the Epson RC+ API | System tool palette, drag the Shutdown VI onto the block diagram. The Shutdown VI must be used for each Init VI.
The block diagram should look similar to this:



19. Run the application. The robot motors should turn on, then the robot should move to point 0, then move to point 1, and then the robot motors will turn off.

16.5 Working with Spel+ projects

Using a Spel+ project with your LabVIEW application is optional. However, if you will be saving point data, or you want to use point labels and / or I/O labels, tasks, or vision sequences, then you will need to use a Spel+ project.

By default, the project is LabVIEW_Default, located in the \EpsonRC70\Projects\LabVIEW folder.

If desired, you can create your own projects using EPSON RC+ 7.0, and then specify which project you want to use with the Initialize VI *Project* input parameter, as shown below:

To work with EPSON RC+ 7.0 projects, start the EPSON RC+ 7.0 application. Use the Project menu to create, open, and edit projects. For more information, see the EPSON RC+ 7.0 User's Guide.

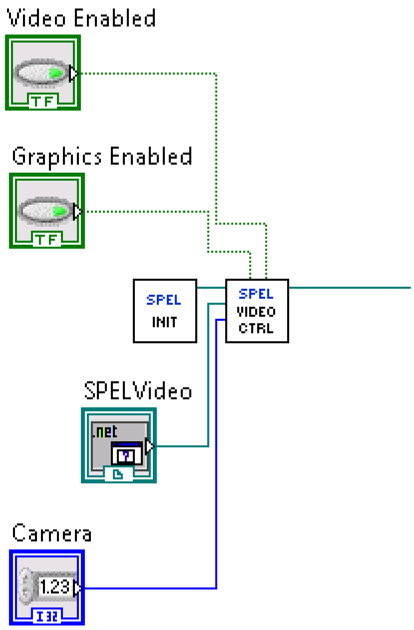
16.6 Displaying Video

You can display video for your Vision Guide sequences using the SPELVideo control and the VideoControl VI.

To display video:

1. Add a SPELVideo control to a front panel.
2. Add a VideoControl.vi to the corresponding block diagram.
3. Wire the output from the SPELVideo control to the SPELVideo Ref In input on the VideoControl VI.
4. Wire the Spel Ref In and Spel Ref Out parameters for the VideoControl VI.
5. Add constants or controls for the *Camera*, *Graphics Enabled*, and *Video Enabled* parameters on the VideoControl VI. Video Enabled must be set to true in order to display video.

The flow diagram below shows the connections for the SPELVideo control and the SPEL VideoControl VI.



When *Video Enabled* is true, and VRun executes from the VRun VI or in a controller task, you will see the resulting video, depending on the Camera setting.

By default, the *Camera* input parameter is zero, which allows video from any camera to be displayed. If you set *Camera* to a number other than zero, then video will be displayed for sequences using the specified camera.

When *Graphics Enabled* is true, and VRun executes, then the sequence result graphics are displayed over the video image.

You can only use one SPEL Video control at a time in your application.

16.7 VI Reference

This section contains information for all VIs used in the EPSON RC+ 7.0 LabVIEW VI Library.

The following information is provided for each VI:

Tool Palette	The tool palette where the VI is contained.
Description	Brief description of the function.
Inputs	Input parameters
Outputs	Output parameters
Remarks	Additional details.

Accel VI

Tool Palette

Epson Robots | Robot Settings

Description

Sets the point to point acceleration and deceleration for the current robot.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Accel	Integer value for point to point acceleration.
Decel	Integer value for point to point acceleration.
Depart Accel	Optional. Integer value for Jump depart acceleration.
Depart Decel	Optional. Integer value for Jump depart deceleration.
Appro Accel	Optional. Integer value for Jump approach acceleration.
Appro Decel	Optional. Integer value for Jump approach deceleration.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

Remarks

Use Accel to set the point to point acceleration and deceleration values for the current robot. All values can be from 1 to 100%. If Depart Accel is specified, then the remaining inputs must also be specified.

See Also

AccelS, Speed, SpeedS

AccelS VI

Tool Palette

Epson Robots | Robot Settings

Description

Sets the linear acceleration and deceleration for the current robot.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Accel	Double value for linear acceleration.
Decel	Double value for linear acceleration.
Depart Accel	Optional. Double value for Jump depart acceleration.
Depart Decel	Optional. Double value for Jump depart deceleration.
Appro Accel	Optional. Double value for Jump approach acceleration.
Appro Decel	Optional. Double value for Jump approach deceleration.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

Remarks

Use AccelS to set the linear acceleration and deceleration values for the current robot. All values are in millimeters / sec^2. If Depart Accel is specified, then the remaining inputs must also be specified.

See Also

Accel, Speed, SpeedS

Arc VI

Tool Palette

Epson Robots | Motion

Description

Arc moves the arm to the specified point using circular interpolation in the XY plane.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Mid Point Number	Specifies the mid point by using an integer.
Mid Point Expr	Specifies the mid point by using a string expression. If this input is used, then you must also specify the end point with a string expression.
End Point Number	Specifies the end point by using an integer.
End Point Expr	Specifies the end point by using a string expression. You can include ROT, CP, SYNC, a search expression for Till, and a parallel processing statement.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Accel, Arc3, BMove, Go, Jump, Jump3, Move, Speed, TGo, TMove

Arc3 VI

Tool Palette

Epson Robots | Motion

Description

Arc3 moves the arm to the specified point using circular interpolation in 3 dimensions.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Mid Point Number	Specifies the mid point by using an integer.
Mid Point Expr	Specifies the mid point by using a string expression. If this input is used, then you must also specify the end point with a string expression.
End Point Number	Specifies the end point by using an integer.
End Point Expr	Specifies the end point by using a string expression. You can include CP, SYNC, a search expression for Till, and a parallel processing statement.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

AccelS, Arc, BMove, Go, Jump, Jump3, Move, SpeedS, TGo, TMove

Arch VI

Tool Palette

Epson Robots | Robot Settings

Description

Defines ARCH parameters (Z height to move before beginning horizontal motion) for use with the JUMP instructions.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Arch Number	The depart distance in millimeters moved at the beginning of the Jump instruction before starting horizontal motion.
Depart Dist	The depart distance in millimeters moved at the beginning of the Jump instruction before starting horizontal motion.
Appro Dist	The approach distance in millimeters above the target position of the Jump instruction.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Jump, Jump3

Arm VI

Tool Palette

Epson Robots | Robot Settings

Description

Selects the current robot arm.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Arm Number	Integer from 0-15. The user may select up to 16 different arms. Arm 0 is the standard (default) robot arm. Arm(s) 1-15 are auxiliary arms defined by the ArmSet instruction.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Armset, GetArm, Tool

Armset VI

Tool Palette

Epson Robots | Robot Settings

Description

Defines an auxiliary robot arm.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
ArmNumber	Integer number: Valid range from 1-15.
Param1	(For SCARA Robots) The horizontal distance from the center line of the elbow joint to the center line of the new orientation axis. (I.E. the position where the new auxiliary arm's orientation axis center line is located.) (For Cartesian Robots) X axis direction position offset from the original X position specified in mm.
Param2	(For SCARA Robots) The offset (in degrees) between the line formed between the normal Elbow center line and the normal orientation Axis center line and the line formed between the new auxiliary arm elbow center line and the new orientation axis center line. (These 2 lines should intersect at the elbow center line and the angle formed is the *Param2*.) (For Cartesian Robots) Y axis direction position offset from the original Y position specified in mm.
Param3	(For SCARA & Cartesian Robots) The Z height offset difference between the new orientation axis center and the old orientation axis center. (This is a distance.)
Param4	(For SCARA Robots) The distance from the shoulder center line to the elbow center line of the elbow orientation of the new auxiliary axis. (For Cartesian Robots) This is a dummy parameter (Specify 0)
Param5	(For SCARA & Cartesian Robots) The angular offset (in degrees) for the new orientation axis vs. the old orientation axis.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

AtHome VI

Tool Palette

Epson Robots | Motion

Description

Returns True if the current robot is at the home position.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
At Home	Boolean indicating if the current robot is at the home position.

AvoidSing VI

Tool Palette

Epson Robots | Motion

Description

Enables / disables the singularity avoidance feature for Move, Arc, and Arc3 motion methods.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Enable	True enables singularity avoidance and False disables it.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

BGo VI

Tool Palette

Epson Robots | Motion

Description

Executes Point to Point relative motion in the selected local coordinate system.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Point Number	Optional. Specifies the target end point by using the point number for a previously taught point in the controller's point memory for the current robot. If *Point Expression* is specified, then *Point Number* is ignored.
Point Expression	Optional. Specifies the target end point by using a string expression. If *Point Expression* is not specified, then the *Point Number* input will be used.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Accel, Arc, Arc3, BMove, Go, Jump, Jump3, Move, Speed, TGo, TMove

BMove VI

Tool Palette

Epson Robots | Motion

Description

Executes linear interpolated relative motion in the selected local coordinate system

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Point Number	Optional. Specifies the target end point by using the point number for a previously taught point in the controller's point memory for the current robot. If *Point Expression* is specified, then *Point Number* is ignored.
Point Expression	Optional. Specifies the target end point by using a string expression. If *Point Expression* is not specified, then the *Point Number* input will be used.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

AccelS, Arc, Arc3, BGo, Go, Jump, Jump3, Move, SpeedS, TGo, TMove

Box VI

Tool Palette

Epson Robots | Robot Settings

Description

Specifies an approach check area defined within a box.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
AreaNumber	Integer number from 1-15 representing which of the 15 boxes to define.
Min X	The minimum X coordinate position of the approach check area.
Max X	The maximum X coordinate position of the approach check area.
Min Y	The minimum Y coordinate position of the approach check area.
Max Y	The maximum Y coordinate position of the approach check area.
Min Z	The minimum Z coordinate position of the approach check area.
Max Z	The maximum Z coordinate position of the approach check area.
Polarity On	Set the remote output logic when the corresponding remote output is used. To set I/O output to On when the end effector is in the box area, use True. To set I/O output to Off when the end effector is in the box area, use False.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

XYLim

Continue VI

Tool Palette

Epson Robots | Tasks

Description

Causes all tasks in the controller that were paused to resume.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

Delay VI

Tool Palette

Epson Robots | System

Description

Delays processing for the specified number of milliseconds.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Milliseconds	The number of milliseconds to delay.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

ECP VI

Tool Palette

Epson Robots | Robot Settings

Description

Selects the current ECP definition.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
ECPNumber	Integer number from 0-15 representing which of 16 ECP definitions to use with the next motion instructions.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

ECPSet, GetECP

ECPset VI

Tool Palette

Epson Robots | Robot Settings

Description

Defines an ECP (external control point).

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
ECPNumber	Integer number from 1-15 representing which of 15 external control points to define.
X	The external control point X coordinate.
Y	The external control point Y coordinate.
Z	The external control point Z coordinate.
U	The external control point U coordinate.
V	The external control point V coordinate.
W	The external control point W coordinate.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

ECP, GetECP

Find VI

Tool Palette

Epson Robots | Motion

Description

Specifies a condition for storing coordinates during motion.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Condition	String expression that contains functions and operators .

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

Remarks

Use Find to specify when a position should be stored during motion. When the condition is satisfied, the current position is stored in FindPos.

See Also

FindPos

FindPos VI

Tool Palette

Epson Robots | Motion

Description

Sets a point with the FindPos coordinates.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Point Number	Optional. Specifies the target end point by using the point number for a previously taught point in the controller's point memory for the current robot. If *Point Expression* is specified, then *Point Number* is ignored.
Point Name	Optional. Specifies the name of the point. If *Point Name* is not specified, then the *Point Number* input will be used.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Find

Fine VI

Tool Palette

Epson Robots | Robot Settings

Description

Specifies and displays the positioning accuracy for target points.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
J1MaxErr – J9MaxErr	Integer number ranging from (0-32767) which represents the allowable positioning error for the each joint. The values for joints 7, 8, and 9 are optional.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

GetArm VI

Tool Palette

Epson Robots | Robot Settings

Description

Returns the current Arm number for the current robot.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Arm Number	The current arm number.

GetECP VI

Tool Palette

Epson Robots | Robot Settings

Description

Returns the current ECP number for the current robot.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
ECP Number	The current ECP number.

GetMotor VI

Tool Palette

Epson Robots | Robot Settings

Description

Returns the motor on status for the current robot.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Motors On	True if motors are on and false if not.

See Also

GetPower, MotorOn, MotorOff

GetOprMode VI

Tool Palette

Epson Robots | System

Description

Reads the EPSON RC+ 7.0 mode of operation.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Operation Mode	The mode of operation for the associated EPSON RC+ 7.0 server process.

Mode	ID	Description
Auto	1	EPSON RC+ 7.0 is in auto mode.
Program	2	EPSON RC+ 7.0 is in program mode..

Remarks

When *Operation Mode* is set to Program, the EPSON RC+ 7.0 GUI for the associated server process is opened and the controller operation mode is set to Program. If the user closes the RC+ GUI, *Operation Mode* is set to Auto. If *Operation Mode* is set to Auto, the RC+ GUI also closes.

See Also

OprMode

GetPoint VI

Tool Palette

Epson Robots | Points

Description

Retrieves coordinate data for a robot point.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Point Number	Optional. Specifies the target end point by using the point number for a previously taught point in the controller's point memory for the current robot. If *Point Expression* is specified, then *Point Number* is ignored.
Point Expression	Optional. Specifies the target end point by using a string expression. If *Point Expression* is not specified, then the *Point Number* input will be used.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
X – W	X, Y, Z, U, V, W coordinates of the specified point.

See Also

LoadPoints, Robot, SavePoints, SetPoint

GetPower VI

Tool Palette

Epson Robots | Robot Settings

Description

Returns the power high status for the current robot.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Power High	True if power is high and false if not.

See Also

PowerHigh, PowerLow

GetRobot VI

Tool Palette

Epson Robots | Robot Settings

Description

Returns the current robot number.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Robot Number	The current robot number.

See Also

Robot

GetTool VI

Tool Palette

Epson Robots | Robot Settings

Description

Returns the current Tool number for the current robot.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Tool Number	The current tool number.

GetVar VI

Tool Palette

Epson Robots | Variables

Description

Returns the value of a SPEL+ global preserve variable in the controller.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Var Name	The name of the SPEL+ global preserve variable. For an array, the entire array can be returned or just one element.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Value	A variant containing the value.

See Also

SetVar

Go VI

Tool Palette

Epson Robots | Motion

Description

Moves the arm in a Point to Point fashion from the current position to the specified point or XY position. The **GO** instruction can move any combination of the robot axes at the same time.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Point Number	Optional. Specifies the target end point by using the point number for a previously taught point in the controller's point memory for the current robot. If *Point Expression* is specified, then *Point Number* is ignored.
Point Expression	Optional. Specifies the target end point by using a string expression. If *Point Expression* is not specified, then the *Point Number* input will be used.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Accel, Arc, Arc3, BGo, BMove, Jump, Jump3, Move, Speed, TGo, TMove

Halt VI

Tool Palette

Epson Robots | Tasks

Description

Suspends execution of the specified task.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Task Number	Optional. The task number of the task to be suspended. The range of the task number is 1 to 32. If *Task Name* is specified, then *Task Number* is ignored.
Task Name	Optional. Specifies the name of the task to be suspended. If *Task Name* is not specified, then the *Task Number* input will be used.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Quit, Resume

Here VI

Tool Palette

Epson Robots | Points

Description

Teaches a point at the current position.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Point Number	Optional. Specifies the target end point by using the point number for a previously taught point in the controller's point memory for the current robot. If *Point Expression* is specified, then *Point Number* is ignored.
Point Name	Optional. Specifies the name of the point. If *Point Name* is not specified, then the *Point Number* input will be used.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

HideWindow VI

Tool Palette

Epson Robots | GUI

Description

Hides an EPSON RC+ 7.0 window that was previously displayed with ShowWindow.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Window ID	The ID of the EPSON RC+ 7.0 window to show.

Window name	ID	Description
IOMonitor	1	ID for the I/O Monitor window.
TaskManager	2	ID for the Task Manager window.
ForceMonitor	3	ID for the Force Monitor window.
Simulator	4	ID for the Simulator window.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

RunDialog, ShowWindow

In VI

Tool Palette

Epson Robots | Inputs & Outputs

Description

Returns the status of the specified input port. Each port contains 8 input bits (one byte).

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Port Number	Optional. Integer expressing one of the input ports. Each port contains 8 input bits (one byte). If *Label* is not specified, then *Port Number* is used.
Label	Optional. String containing an input byte label. If *Label* is specified, then *Port Number* is ignored.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Value	Integer from 0 to 255 representing the status of the input port.

See Also

InBCD, InW, Sw

InBCD VI

Tool Palette

Epson Robots | Inputs & Outputs

Description

Returns the input status of 8 inputs using BCD format. (Binary Coded Decimal)

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Port Number	Optional. Integer expressing one of the input ports. Each port contains 8 input bits (one byte). If *Label* is not specified, then *Port Number* is used.
Label	Optional. String containing an input byte label. If *Label* is specified, then *Port Number* is ignored.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Value	Integer from 0 to 9 representing the status of the input port.

See Also

In, InW, Sw

InsideBox VI

Tool Palette

Epson Robots | Motion

Description

Returns whether the current robot end effector is inside a specified box area.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Area Number	Integer number from 1-15 representing which of the 15 boxes to check.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Status	Boolean that is True if the robot end effector is inside the box.

See Also

Box, InsidePlane, Plane

InsidePlane VI

Tool Palette

Epson Robots | Motion

Description

Returns whether the current robot end effector is inside a specified plane.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
AreaNumber	Integer number from 1-15 representing which of the 15 boxes to check.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Status	Boolean that is True if the robot end effector is inside the plane.

See Also

Box, InsideBox, Plane

InW VI

Tool Palette

Epson Robots | Inputs & Outputs

Description

Returns the status of the specified input word port. Each word port contains 16 input bits.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Port Number	Optional. Integer expressing one of the input ports. Each port contains 8 input bits (one byte). If *Label* is not specified, then *Port Number* is used.
Label	Optional. String containing an input byte label. If *Label* is specified, then *Port Number* is ignored.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Value	Integer value from 0 to 65535 representing the input port.

See Also

In, InBCD, Sw

Inertia VI

Tool Palette

Epson Robots | Robot Settings

Description

Specifies the load inertia and eccentricity for the current robot.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
LoadInertia	Double value that specifies total moment of inertia in kgm2 around the center of the end effector joint, including end effector and part.
Eccentricity	Double value that specifies eccentricity in mm around the center of the end effector joint, including end effector and part.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Weight

Initialize VI

Tool Palette

Epson Robots | System

Description

Initializes the instance of Spel used by the LabVIEW VI library.

Inputs

Server Product Type	Optional. Specifies which EPSON RC+ product to interface with.
Connection Number	Optional. Specifies which controller connection to use.
Project	Optional. Specifies the EPSON RC+ project to be used.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

Remarks

The Initialize VI must be called for each instance of the library that will be used.

Server Product Type is used to specify which EPSON RC+ product to use. The default is RC70 (EPSON RC+ 7.0), which interfaces with RC700 and RC90 robot controllers.

When *Connection Number* is not specified, then the connection last used in the EPSON RC+ 7.0 will be used.

When *Project* is not specified, the default LabVIEW EPSON RC+ 7.0 project will be used. The project must be used in the EPSON RC+ product specified with *Server Product Type*.

See Also

Shutdown

JRange VI

Tool Palette

Epson Robots | Robot Settings

Description

Defines the permissible working range of the specified robot joint in pulses.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
JointNumber	Integer number between 1 - 9 representing the joint for which JRange will be specified.
LowerLimitPulses	Integer number representing the encoder pulse count position for the lower limit range of the specified joint.
UpperLimitPulses	Integer number representing the encoder pulse count position for the upper limit range of the specified joint

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

JS VI

Tool Palette

Epson Robots | Motion

Description

Jump Sense detects whether the arm stopped prior to completing a JUMP instruction (which used a SENSE input) or if the arm completed the JUMP move. JS returns the Jump Sense status.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
JS	True if the SENSE input was detected during motion. False if not.

See Also

Jump, Sense

JTran VI

Tool Palette

Epson Robots | Motion

Description

Executes a relative joint move.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
JointNumber	The specific joint to move.
Distance	The distance to move. Units are in degrees for rotary joints and millimeters for linear joints.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

Jump VI

Tool Palette

Epson Robots | Motion

Description

Moves the arm from the current position to the specified point using point to point motion while first moving in a vertical direction up, then horizontally and then finally vertically downward to arrive on the final destination point.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Point Number	Optional. Specifies the target end point by using the point number for a previously taught point in the controller's point memory for the current robot. If *Point Expression* is specified, then *Point Number* is ignored.
Point Expression	Optional. Specifies the target end point by using a string expression. If *Point Expression* is not specified, then the *Point Number* input will be used.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Accel, Arc, Arc3, BGo, BMove, Go, Jump3, Move, Speed, TGo, TMove

Jump3 VI

Tool Palette

Epson Robots | Motion

Description

Motion with 3D gate using a combination of two CP motions and one PTP motion. The robot moves to the depart point, then the approach point, and finally the destination point.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Depart Point Number	Specifies the depart point by using an integer.
Depart Point Expr	Specifies the depart point by using a string expression. If this input is used, then you must also specify the approach and destination points with string expressions.
Appro Point Number	Specifies the approach point by using an integer.
Appro Point Expr	Specifies the approach point by using a string expression. If this input is used, then you must also specify the depart and destination points with string expressions.
Dest Point Number	Specifies the destination point by using an integer.
Dest Point Expr	Specifies the destination point by using a string expression. If this input is used, then you must also specify the depart and approach points with string expressions.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Accel, Arc, Arc3, BGo, BMove, Go, Jump, Move, Speed, TGo, TMove

Jump3CP VI

Tool Palette

Epson Robots | Motion

Description

Motion with 3D gate using a combination of three CP motions. The robot moves to the depart point, then the approach point, and finally the destination point.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Depart Point Number	Specifies the depart point by using an integer.
Depart Point Expr	Specifies the depart point by using a string expression. If this input is used, then you must also specify the approach and destination points with string expressions.
Appro Point Number	Specifies the approach point by using an integer.
Appro Point Expr	Specifies the approach point by using a string expression. If this input is used, then you must also specify the depart and destination points with string expressions.
Dest Point Number	Specifies the destination point by using an integer.
Dest Point Expr	Specifies the destination point by using a string expression. If this input is used, then you must also specify the depart and approach points with string expressions.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Accel, Arc, Arc3, BGo, BMove, Go, Jump, Jump3, Move, Speed, TGo, TMove

LimZ VI

Tool Palette

Epson Robots | Motion

Description

Sets the default value of the Z axis height for JUMP commands.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Z Limit	A coordinate value within the movable range of the Z axis.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Jump, Jump3

LoadPoints VI

Tool Palette

Epson Robots | Points

Description

Loads a SPEL+ point file into the controller's point memory for the current robot.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
File Name	A valid point file that is in the current Spel project or was previously saved with the SavePoints VI.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

GetPoint, Robot, SavePoints, SetPoint

MemIn VI

Tool Palette

Epson Robots | Inputs & Outputs

Description

Returns the status of the specified memory I/O byte port. Each port contains 8 memory I/O bits.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Port Number	Optional. Integer expressing one of the input ports. Each port contains 8 input bits (one byte). If *Label* is not specified, then *Port Number* is used.
Label	Optional. String containing an input byte label. If *Label* is specified, then *Port Number* is ignored.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Value	Integer from 0 to 255 representing the status of the port.

See Also

MemInW, MemOut, MemOutW

MemInW VI

Tool Palette

Epson Robots | Inputs & Outputs

Description

Returns the status of the specified memory I/O word port. Each word port contains 16 memory I/O bits.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Port Number	Optional. Integer expressing one of the input ports. Each word port contains 16 input bits. If *Label* is not specified, then *Port Number* is used.
Label	Optional. String containing an input byte label. If *Label* is specified, then *Port Number* is ignored.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Value	Integer from 0 to 255 representing the status of the port.

See Also

MemIn, MemOut, MemOutW

MemOut VI

Tool Palette

Epson Robots | Inputs & Outputs

Description

Simultaneously sets 8 memory I/O bits based on the 8 bit value specified by the user.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Port Number	Optional. Integer expressing one of the input ports. Each port contains 8 input bits (one byte). If *Label* is not specified, then *Port Number* is used.
Label	Optional. String containing an input byte label. If *Label* is specified, then *Port Number* is ignored.
Value	Integer containing the output pattern for the specified byte. Valid values are from 0 - 255.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

MemIn, MemInW, MemOutW

MemOff VI

Tool Palette

Epson Robots | Inputs & Outputs

Description

Turns off the specified bit of memory I/O.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Bit Number	Optional. Integer representing one of the memory I/O bits. If *Label* is not specified, then *Bit Number* is used.
Label	Optional. String containing an input bit label. If *Label* is specified, then *Bit Number* is ignored.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

MemOn, MemOut, MemOutW

MemOn VI

Tool Palette

Epson Robots | Inputs & Outputs

Description

Turns on the specified bit of memory I/O.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Bit Number	Optional. Integer representing one of the memory I/O bits. If *Label* is not specified, then *Bit Number* is used.
Label	Optional. String containing an input bit label. If *Label* is specified, then *Bit Number* is ignored.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

MemOff, MemOut, MemOutW

MemOut VI

Tool Palette

Epson Robots | Inputs & Outputs

Description

Simultaneously sets 8 memory I/O bits based on the 8 bit value specified by the user.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Port Number	Optional. Integer expressing one of the memory I/O ports. Each port contains 8 memory I/O bits (one byte). If *Label* is not specified, then *Port Number* is used.
Label	Optional. String containing a memory I/O byte label. If *Label* is specified, then *Port Number* is ignored.
Value	Integer containing the output pattern for the specified byte. Valid values are from 0 - 255.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

MemOn, MemOff, MemOutW

MemOutW VI

Tool Palette

Epson Robots | Inputs & Outputs

Description

Simultaneously sets 16 memory I/O bits based on the 16 bit value specified by the user.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Port Number	Optional. Integer representing one of the memory I/O ports. Each word port contains 16 input bits. If *Label* is not specified, then *Port Number* is used.
Label	Optional. String containing an memory I/O byte label. If *Label* is specified, then *Port Number* is ignored.
Value	Integer containing the output pattern for the specified word. Valid values are from 0 - 65535.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

MemOn, MemOff, MemOut

MemSw VI

Tool Palette

Epson Robots | Inputs & Outputs

Description

Returns the status of the specified memory I/O bit.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Bit Number	Optional. Integer representing one of the memory I/O bits. If *Label* is not specified, then *Bit Number* is used.
Label	Optional. String containing a memory I/O bit label. If *Label* is specified, then *Bit Number* is ignored.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Value	Boolean that is True with the memory I/O bit is on.

See Also

MemIn, MemInW

MotorOff VI

Tool Palette

Epson Robots | Robot Settings

Description

Turns motors off for the current robot.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

MotorOn, PowerHigh, PowerLow, Robot

MotorOn VI

Tool Palette

Epson Robots | Robot Settings

Description

Turns motors on for the current robot.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

MotorOff, PowerHigh, PowerLow, Robot

Move VI

Tool Palette

Epson Robots | Motion

Description

Moves the arm from the current position to the specified point using linear interpolation (i.e. moving in a straight line).

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Point Number	Optional. Specifies the target end point by using the point number for a previously taught point in the controller's point memory for the current robot. If *Point Expression* is specified, then *Point Number* is ignored.
Point Expression	Optional. Specifies the target end point by using a string expression. If *Point Expression* is not specified, then the *Point Number* input will be used.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

AccelS, Arc, Arc3, BGo, BMove, Go, Jump, Jump3, SpeedS, TGo, TMove

Off VI

Tool Palette

Epson Robots | Inputs & Outputs

Description

Turns off the specified output bit.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Bit Number	Optional. Integer representing one of the output bits. If *Label* is not specified, then *Bit Number* is used.
Label	Optional. String containing an input bit label. If *Label* is specified, then *Bit Number* is ignored.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

On VI

Tool Palette

Epson Robots | Inputs & Outputs

Description

Turns on the specified output bit.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Bit Number	Optional. Integer representing one of the output bits. If *Label* is not specified, then *Bit Number* is used.
Label	Optional. String containing an input bit label. If *Label* is specified, then *Bit Number* is ignored.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

OPort VI

Tool Palette

Epson Robots | Inputs & Outputs

Description

Returns the status of the specified output bit.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Bit Number	Optional. Integer representing one of the output bits. If *Label* is not specified, then *Bit Number* is used.
Label	Optional. String containing an output bit label. If *Label* is specified, then *Bit Number* is ignored.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Value	Boolean that is True with the output bit is on.

See Also

In, InW, On, Off, Out, Sw

OprMode VI

Tool Palette

Epson Robots | System

Description

Sets the EPSON RC+ 7.0 mode of operation..

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Operation Mode	The mode of operation for the associated EPSON RC+ 7.0 server process.

Mode	**ID**	**Description**
Auto	1	EPSON RC+ 7.0 is in auto mode.
Program	2	EPSON RC+ 7.0 is in program mode..

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

Remarks

When *Operation Mode* is set to Program, the EPSON RC+ 7.0 GUI for the associated server process is opened and the controller operation mode is set to Program. If the user closes the RC+ GUI, *Operation Mode* is set to Auto. If *Operation Mode* is set to Auto, the RC+ GUI also closes.

See Also

GetOprMode

Pause VI

Tool Palette

Epson Robots | Tasks

Description

Causes all normal SPEL+ tasks in the controller to pause. If the robot is moving, it will immediately decelerate to a stop.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Continue, Stop

Plane VI

Tool Palette

Epson Robots | Robot Settings

Description

Defines a plane.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Plane Number	Integer expression from 1-15 representing which of 15 planes to define.
X	The plane coordinate system origin X coordinate.
Y	The plane coordinate system origin Y coordinate.
Z	The plane coordinate system origin Z coordinate.
U	The plane coordinate system rotation about the Z axis.
V	The plane coordinate system rotation about the Y axis.
W	The plane coordinate system rotation about the X axis.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Box, InsideBox, InsidePlane

PowerHigh VI

Tool Palette

Epson Robots | Robot Settings

Description

Sets motor power to high for the current robot.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

MotorOff, MotorOn, PowerLow, Robot

PowerLow VI

Tool Palette

Epson Robots | Robot Settings

Description

Sets motor power to low for the current robot.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

MotorOff, MotorOn, PowerHigh, Robot

Quit VI

Tool Palette

Epson Robots | Tasks

Description

Terminates execution of the specified task.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Task Number	Optional. The task number of the task to be terminated. The range of the task number is 1 to 32. If *Task Name* is specified, then *Task Number* is ignored.
Task Name	Optional. Specifies the name of the task to be terminated. If *Task Name* is not specified, then the *Task Number* input will be used.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Halt, Resume

Reset VI

Tool Palette

Epson Robots | System

Description

Resets the controller to the initialized state.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

Resume VI

Tool Palette

Epson Robots | Tasks

Description

Resumes a task which was suspended by the Halt VI.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Task Number	Optional. The task number of the task to be resumed. The range of the task number is 1 to 32. If *Task Name* is specified, then *Task Number* is ignored.
Task Name	Optional. Specifies the name of the task to be resumed. If *Task Name* is not specified, then the *Task Number* input will be used.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Halt, Quit

Robot VI

Tool Palette

Epson Robots | Robot Settings

Description

Selects the current robot.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Robot Number	Integer from 1-16.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

GetRobot, MotorOff, MotorOn, PowerHigh, PowerLow

RunDialog VI

Tool Palette

Epson Robots | GUI

Description

Runs an EPSON RC+ 7.0 dialog.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Dialog ID	The ID of the EPSON RC+ 7.0 dialog to run.

Dialog name	ID	Description
RobotManager	1	ID for Tools | Robot Manager dialog
ControllerTools	2	ID for Tools | Controller dialog
VisionGuide	3	ID for Tools | Vision Guide dialog

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

SavePoints VI

Tool Palette

Epson Robots | Points

Description

Save points for the current robot into a file.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
File Name	The name of a point file that is in the current Spel project or a new file that will be stored in the controller.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

GetPoint, LoadPoints, Robot, SetPoint

Sense VI

Tool Palette

Epson Robots | Motion

Description

Specifies input condition that, if satisfied, completes the Jump in progress by stopping the robot above the target position.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Condition	Specifies the I/O condition using a string expression. For details see the Sense Statement in the SPEL+ Language Reference manual.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

JS, Jump

SetPoint VI

Tool Palette

Epson Robots | Points

Description

Sets the coordinate data for a point for the current robot.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Point Number	Optional. Specifies the target end point by using the point number for a previously taught point in the controller's point memory for the current robot. If *Point Name* is specified, then *Point Number* is ignored.
Point Name	Optional. Specifies the point by using a string expression for the point name. If *Point Name* is not specified, then the *Point Number* input will be used.
X – W	X, Y, Z, U, V, W coordinates of the specified point.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

GetPoint, LoadPoints, Robot, SavePoints

SetVar VI

Tool Palette

Epson Robots | Variables

Description

Sets the value of a SPEL+ global preserve variable in the controller.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Var Name	The name of the SPEL+ global preserve variable.
Value	A variant containing the value.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

GetVar

SFree VI

Tool Palette

Epson Robots | Robot Settings

Description

Frees the specified robot axes from servo control.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Axes	Optional. Integer array specifying which axes to free. If omitted, all axes are freed.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

Remarks

If Axes is omitted, then all axes are freed.

See Also

MotorOff, MotorOn, SLock

ShowWindow VI

Tool Palette

Epson Robots | GUI

Description

Displays an EPSON RC+ 7.0 window.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Window ID	The ID of the EPSON RC+ 7.0 window to show.

Window name	ID	Description
IOMonitor	1	ID for the I/O Monitor window.
TaskManager	2	ID for the Task Manager window.
ForceMonitor	3	ID for the Force Monitor window.
Simulator	4	ID for the Simulator window.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

HideWindow, RunDialog

Shutdown VI

Tool Palette

Epson Robots | System

Description

Shuts down the EPSON RC+ 7.0 server process that was started when the Initialize VI was called.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

Outputs

Error Out	Error condition output for subsequent Spel nodes.

Remarks

The Shutdown VI must be called for each instance of the library. This will shutdown the associated EPSON RC+ 7.0 server process.

See Also

Initialize

SLock VI

Tool Palette

Epson Robots | Robot Settings

Description

Returns specified the specified robot axes to servo control.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Axes	Optional. Integer array specifying which axes to lock. If omitted, all axes are locked.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

Remarks

If Axes is omitted, then all axes are locked.

See Also

MotorOff, MotorOn, SFree

Speed VI

Tool Palette

Epson Robots | Robot Settings

Description

Specifies the arm speed for use with the point to point instructions Go, Jump and Pulse.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
PointToPoint Speed	Specifies the arm speed for use with the point to point instructions Go, Jump and Pulse.
Depart Speed	Integer number between 1-100 representing the Z axis upward motion speed for the Jump instruction.
Appro Speed	Integer number between 1-100 representing the Z axis downward motion speed for the Jump instruction.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

Remarks

Use Speed to set the point to point speed for the current robot. All values can be from 1 to 100%. If *Depart Speed* is specified, then *Appro Speed* must also be specified.

See Also

Accel, AccelS, SpeedS

SpeedS VI

Tool Palette

Epson Robots | Robot Settings

Description

Specifies the arm speed for use with the Continuous Path instructions Jump3CP, Move, Arc, and CVMove.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Linear Speed	Specifies the arm speed for use with the Continuous Path instructions Jump3CP, Move, Arc, and CVMove.
Depart Speed	Double value between 1-5000 representing the Z axis upward motion speed for the Jump3CP instruction.
Appro Speed	Double value between 1-5000 representing the Z axis downward motion speed for the Jump3CP instruction.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

Remarks

Use Speed to set the linear speed for the current robot in millimeters / sec. If *Depart Speed* is specified, then *Appro Speed* must also be specified.

See Also

Accel, AccelS, Speed

Start VI

Tool Palette

Epson Robots | Tasks

Description

Starts a program that will run in the controller.

Inputs

Spel Ref In Spel reference from a previous Spel Ref Out.

Error In Error condition from a previous Spel node.

ProgramNumber The program number to start, corresponding to the 8 main functions in SPEL+ as shown in the table below. The range is 0 to 7.

Program Number	SPEL+ Function Name
0	main
1	main1
2	main2
3	main3
4	main4
5	main5
6	main6
7	main7

Outputs

Spel Ref Out Spel reference output for next VI to use.

Error Out Error condition output for subsequent Spel nodes.

Remarks

When **Start** is executed, control will return immediately to the calling VI. You cannot start a program that is already running. Note that Start causes global variables in the controller to be cleared and default robot points to be loaded.

See Also

Continue, Pause, Stop, Xqt

Stop VI

Tool Palette

Epson Robots | Tasks

Description

Stops all normal SPEL+ tasks running in the controller and optionally stops all background tasks.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Stop Type	Optional. Specify StopNormalTasks (default) or StopAllTasks (also stop background tasks).

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Continue, Pause, Start, Xqt

Sw VI

Tool Palette

Epson Robots | Inputs & Outputs

Description

Returns the status of the specified input bit.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Bit Number	Optional. Integer representing one of the input bits. If *Label* is not specified, then *Bit Number* is used.
Label	Optional. String containing an input bit label. If *Label* is specified, then *Bit Number* is ignored.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Value	Boolean that is True with the output bit is on.

See Also

In, InW, On, Off, OPort, Out

TargetOK VI

Tool Palette

Epson Robots | Motion

Description

Returns a status indicating whether or not the PTP (Point to Point) motion from the current position to a target position is possible.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Point Number	Optional. Specifies the target end point by using the point number for a previously taught point in the controller's point memory for the current robot. If *Point Expression* is specified, then *Point Number* is ignored.
Point Expression	Optional. Specifies the target end point by using a string expression. If *Point Expression* is not specified, then the *Point Number* input will be used.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Target OK	The robot can move to the target position.
At Home	Boolean indicating if the current robot is at the home position.

See Also

BGo, Go, Jump, TGo

TGo VI

Tool Palette

Epson Robots | Motion

Description

Executes Point to Point relative motion, in the current tool coordinate system.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Point Number	Optional. Specifies the target end point by using the point number for a previously taught point in the controller's point memory for the current robot. If *Point Expression* is specified, then *Point Number* is ignored.
Point Expression	Optional. Specifies the target end point by using a string expression. If *Point Expression* is not specified, then the *Point Number* input will be used.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Accel, Arc, Arc3, BGo, BMove, Go, Jump, Jump3, Move, Speed, TMove

Till VI

Tool Palette

Epson Robots | Motion

Description

Specifies event condition that, if satisfied, completes the motion command (Jump, Go, Move, etc.) in progress by decelerating and stopping the robot at an intermediate position.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Condition	Specifies the I/O condition using a string expression. For details see the Sense Statement in the SPEL+ Language Reference manual.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Accel, Arc, Arc3, BGo, BMove, Jump, Jump3, Move, Speed, TGo, TillOn, TMove

TillOn VI

Tool Palette

Epson Robots | Motion

Description

Returns True if a stop has occurred from a till condition during the last Go/Jump/Move statement.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Till On	True if the till condition was was detected during motion. False if not.

See Also

Accel, Arc, Arc3, BGo, BMove, Jump, Jump3, Move, Speed, TGo, Till, TMove

TLSet VI

Tool Palette

Epson Robots | Robot Settings

Description

Defines an ECP (external control point).

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
ToolNumber	Integer expression from 1-15 representing which of 15 tools to define. (Tool 0 is the default tool and cannot be modified.)
X	The tool coordinate system origin X coordinate.
Y	The tool coordinate system origin Y coordinate.
Z	The tool coordinate system origin Z coordinate.
U	The tool coordinate system rotation about the Z axis.
V	The tool coordinate system rotation about the Y axis.
W	The tool coordinate system rotation about the X axis.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

TMove VI

Tool Palette

Epson Robots | Motion

Description

Executes linear interpolation relative motion, in the current tool coordinate system.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Point Number	Optional. Specifies the target end point by using the point number for a previously taught point in the controller's point memory for the current robot. If *Point Expression* is specified, then *Point Number* is ignored.
Point Expression	Optional. Specifies the target end point by using a string expression. If *Point Expression* is not specified, then the *Point Number* input will be used.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

AccelS, Arc, Arc3, BGo, BMove, Go, Jump, Jump3, Move, SpeedS, TGo

Tool VI

Tool Palette

Epson Robots | Robot Settings

Description

Selects the current robot tool.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Tool Number	Integer number from 0-15 representing which of 16 tool definitions to use with subsequent motion instructions.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Arm, Armset, GetTool, TLSet

VGetBool VI

Tool Palette

Epson Robots | Vision

Description

Retrieves a vision property or result that returns a Boolean value.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Sequence	The name of a vision sequence in the current project.
Object	Optional. The name of a vision object in the specified sequence. Omit when retrieving a property or result for a sequence.
Property Code	The property or result code.
Result Index	Optional. The index of the result.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Value	The Boolean value.

Remarks

See the Vision Guide 7.0 Properties and Results Reference manual for details on Vision Guide properties and results.

See Also

VRun, VGetDbl, VGetInt, VGetStr, VSetBool, VSetDbl, VSetInt, VSetStr

VGetDbl VI

Tool Palette

Epson Robots | Vision

Description

Retrieves a vision property or result that returns a double value.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Sequence	The name of a vision sequence in the current project.
Object	Optional. The name of a vision object in the specified sequence. Omit when retrieving a property or result for a sequence.
Property Code	The property or result code.
Result Index	Optional. The index of the result.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Value	The double value.

Remarks

See the Vision Guide 7.0 Properties and Results Reference manual for details on Vision Guide properties and results.

See Also

VRun, VGetBool, VGetInt, VGetStr, VSetBool, VSetDbl, VSetInt, VSetStr

VGetInt VI

Tool Palette

Epson Robots | Vision

Description

Retrieves a vision property or result that returns an integer value.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Sequence	The name of a vision sequence in the current project.
Object	Optional. The name of a vision object in the specified sequence. Omit when retrieving a property or result for a sequence.
Property Code	The property or result code.
Result Index	Optional. The index of the result.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Value	The integer value.

Remarks

See the Vision Guide 7.0 Properties and Results Reference manual for details on Vision Guide properties and results.

See Also

VRun, VGetBool, VGetDbl, VGetStr, VSetBool, VSetDbl, VSetInt, VSetStr

VGetStr VI

Tool Palette

Epson Robots | Vision

Description

Retrieves a vision property or result that returns a string value.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Sequence	The name of a vision sequence in the current project.
Object	Optional. The name of a vision object in the specified sequence. Omit when retrieving a property or result for a sequence.
Property Code	The property or result code.
Result Index	Optional. The index of the result.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Value	The string value.

Remarks

See the Vision Guide 7.0 Properties and Results Reference manual for details on Vision Guide properties and results.

See Also

VRun, VGetBool, VGetDbl, VGetInt, VSetBool, VSetDbl, VSetInt, VSetStr

VideoControl VI

Tool Palette

Epson Robots | Vision

Description

Controls a SPEL Video control.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Sequence	The name of a vision sequence in the current project.
VideoRef In	Reference from a SPEL Video control.
Camera	Sets which camera video to display. Default is 0, which displays any camera.
Graphics Enabled	Sets whether graphics should be displayed.
Video Enabled	Sets whether video should be displayed.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Displaying Video, VGet, VRun

VRun VI

Tool Palette

Epson Robots | Vision

Description

Run a vision sequence in the current project.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Sequence	The name of a vision sequence in the current project.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

Remarks

Refer to the Vision Guide 7.0 software manual for information on running vision sequences.

See Also

VGetBool, VGetDbl, VGetInt, VGetStr, VSetBool, VSetDbl, VSetInt, VSetStr

VSetBool VI

Tool Palette

Epson Robots | Vision

Description

Sets the value of a vision property whose data type is Boolean.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Sequence	The name of a vision sequence in the current project.
Object	Optional. The name of a vision object in the specified sequence. Omit when setting a property for a sequence.
Property Code	The property code.
Value	The new Boolean value of the property.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

Remarks

See the Vision Guide 7.0 Properties and Results Reference manual for details on Vision Guide properties and results.

See Also

VGetBool, VGetDbl, VGetInt, VGetStr, VRun, VSetDbl, VSetInt, VSetStr

VSetDbl VI

Tool Palette

Epson Robots | Vision

Description

Sets the value of a vision property whose data type is real or double.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Sequence	The name of a vision sequence in the current project.
Object	Optional. The name of a vision object in the specified sequence. Omit when setting a property for a sequence.
Property Code	The property code.
Value	The new double value of the property.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

Remarks

See the Vision Guide 7.0 Properties and Results Reference manual for details on Vision Guide properties and results.

See Also

VGetBool, VGetDbl, VGetInt, VGetStr, VRun, VSetBool, VSetInt, VSetStr

VSetInt VI

Tool Palette

Epson Robots | Vision

Description

Sets the value of a vision property whose data type is integer.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Sequence	The name of a vision sequence in the current project.
Object	Optional. The name of a vision object in the specified sequence. Omit when setting a property for a sequence.
Property Code	The property code.
Value	The new integer value of the property.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

Remarks

See the Vision Guide 7.0 Properties and Results Reference manual for details on Vision Guide properties and results.

See Also

VGetBool, VGetDbl, VGetInt, VGetStr, VRun, VSetBool, VSetDbl, VSetStr

VSetStr VI

Tool Palette

Epson Robots | Vision

Description

Sets the value of a vision property whose data type is string.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Sequence	The name of a vision sequence in the current project.
Object	Optional. The name of a vision object in the specified sequence. Omit when setting a property for a sequence.
Property Code	The property code.
Value	The new string value of the property.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

Remarks

See the Vision Guide 7.0 Properties and Results Reference manual for details on Vision Guide properties and results.

See Also

VGetBool, VGetDbl, VGetInt, VGetStr, VRun, VSetBool, VSetDbl, VSetInt

WaitTaskDone VI

Tool Palette

Epson Robots | Tasks

Description

Waits for a task to finish and returns the status.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Task Number	Optional. The task number of the task to be suspended. The range of the task number is 1 to 32. If *Task Name* is specified, then *Task Number* is ignored.
Task Name	Optional. Specifies the name of the task to be suspended. If *Task Name* is not specified, then the *Task Number* input will be used.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Task State	Indicates the final status of the task (Quit, Aborted, Finished).

See Also

Xqt

Weight VI

Tool Palette

Epson Robots | Robot Settings

Description

Specifies the weight parameters for the current robot.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Payload Weight	The weight of the end effector to be carried in Kg units.
Arm Length	The distance from the rotational center of the second arm to the center of the gravity of the end effector in mm units.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Inertia

Xqt VI

Tool Palette

Epson Robots | Tasks

Description

Start one SPEL+ task.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Task Number	Optional. The task number of the task to execute. The range of the task number is 1 to 32. If *Task Number* is omitted, then a task number will automatically be assigned.
Func Name	Specifies the name of the function to be executed.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Halt, Quit, Resume, WaitForTaskDone

XYLim VI

Tool Palette

Epson Robots | Robot Settings

Description

Sets the permissible motion range limits for the manipulator.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Min X	The minimum X coordinate position to which the manipulator may travel. (The manipulator may not move to a position with the X Coordinate less than min X.)
Max X	The maximum X coordinate position to which the manipulator may travel. (The manipulator may not move to a position with the X Coordinate greater than max X.)
Min Y	The minimum Y coordinate position to which the manipulator may travel. (The manipulator may not move to a position with the Y Coordinate less than min Y.)
Max Y	The maximum Y coordinate position to which the manipulator may travel. (The manipulator may not move to a position with the Y Coordinate greater than max Y.)
Min Z	The minimum Z coordinate position to which the manipulator may travel. (The manipulator may not move to a position with the Z Coordinate less than min Z.)
Max Z	The maximum Z coordinate position to which the manipulator may travel. (The manipulator may not move to a position with the Z Coordinate greater than max Z.)

Remarks

XYLim is used to define motion range limits. Many robot systems allow users to define joint limits but the SPEL+ language allows both joint limits and motion range limits to be defined. In effect this allows users to create a work envelope for their application. (Keep in mind that joint range limits are also definable with SPEL.)

The motion range established with XYLim values applies to motion command target positions only, and not to motion paths from starting position to target position. Therefore, the arm may move outside the XYLim range during motion. (i.e. The XYLim range does not affect Pulse.)

To turn off motion range limits, specify 0 for the range limit parameters.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Box

17. Using LabVIEW with RCNetLib

17.1 Overview

The LabVIEW VI library described in the chapter Using the LabVIEW VI Library is a high level interface that uses the RCAPINet.dll. Some users may want to interface with RCAPINet.dll directly instead of using the high level library. This chapter contains information for using LabVIEW with RCAPINet.dll. The following topics are described.

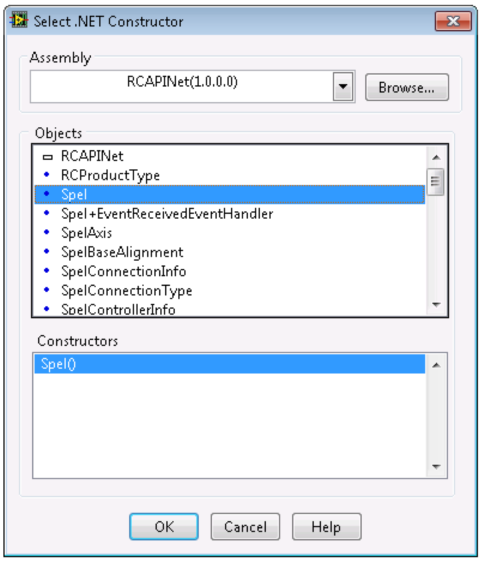
- Initialization
- Use Spel properties and methods in your application
- Shutdown
- Using dialogs and windows

17.2 Initialization

17.2.1 Add a constructor node for the Spel class

Before you can call methods or use properties from the Spel class, you must create an instance of the Spel class using a Constructor Node. You should use one Spel class instance in your application.

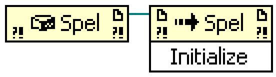
In the Block Diagram view of the VI that will contain the Spel class instance, add a Constructor Node from the [RC+ API] – [.NET palette]. The [Select .NET Constructor] dialog will appear. Select "RCAPINet" in the [Assembly] list and select "Spel" in the [Objects] list, as shown below.



Click <OK> to create a constructor node for Spel in the block diagram.

17.2.2 Initialize the Spel class instance

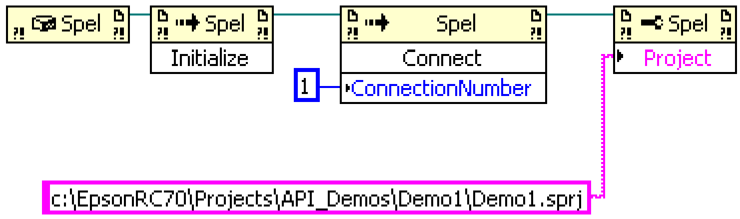
Add an Invoke node for the Spel class Initialize method. When Initialize executes, it will configure and start RC+ as a server in the background.



17.2.3 Connect to controller and set project

Add an Invoke node for the Spel class Connect method. Set the ConnectionNumber parameter for the controller connection you want to use. To view the connection numbers, start EPSON RC+ 7.0, then select [Setup]-[PC to Controller Communications].

Add a Property node for the Spel class Project property. Set the Project parameter to the desired project file.



17.3 Use Spel properties and methods

Add more nodes to use the Spel properties and methods for your application. You must wire the reference output from the previous node to the reference input of the current node. This allows each property or method to use the Spel class instance you created and initialized in the steps above. Refer to the RCAPINet Reference chapter for information on the properties and methods that can be used.

17.4 Shutdown

When you are finished using the Spel class instance, you need to invoke the Dispose method. This will shutdown the EPSON RC+ 7.0 server that is associated with the Spel class instance. Normally, you should call Dispose at the end of your application.

If your application is aborted without calling Dispose, then the RC+ process continues to run, because LabVIEW (the client process) continues to run. If you start your application again, the RC+ process is restarted if it was running. But if you try to run the RC+ GUI, it will ask if you want to run another instance of RC+. In this case, you can terminate the RC+ process (erc70.exe) from the Windows Task Manager first, then run the EPSON RC+ 7.0 GUI.

17.5 Using Dialogs and Windows

When used with .NET applications, a .NET parent form is normally used as the parent for dialogs and windows that are displayed from the Spel class instance. But LabVIEW does not use .NET forms, so to display windows and dialogs from LabVIEW, use the ParentWindowHandle property. Set it to the window handle of your VI. You can call the Windows API FindWindow method to get the window handle.

When using ParentWindowHandle, you must call Spel.ShowWindow without the Parent parameter.

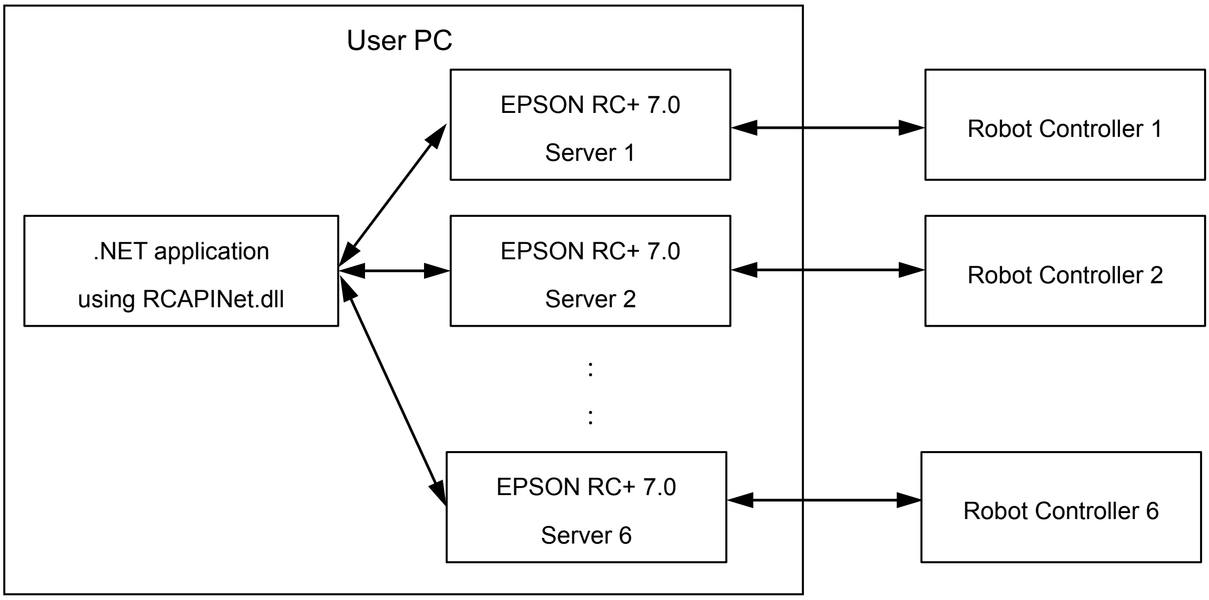
18. How to Control Multiple Controllers from One PC

18.1 Overview

Using the RC+ API, you can control up to six robot controllers from one PC.

To control multiple controllers, an RCAPINet Spel class needs to be instantiated for each controller.

The figure below shows the basic system configuration diagram for controlling multiple controllers using the RC+ API.



The application controls the multiple controllers via the servers (RCAPINet Spel class) prepared for each controller.

18.1.1 System Condition

We recommend a PC that satisfies the following requirements.

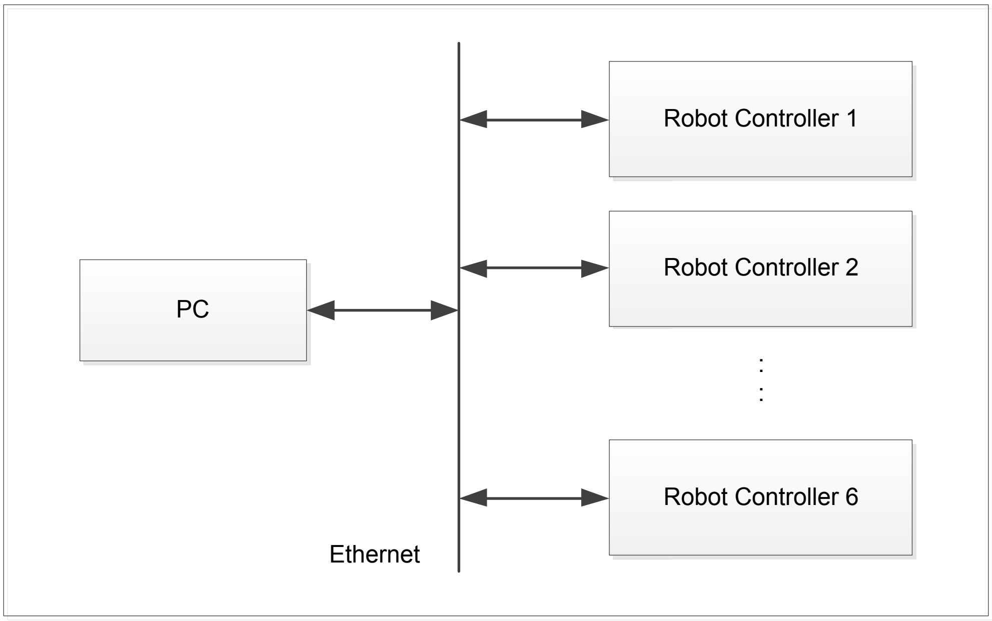
OS	Windows 7 Professional 32 bit or 64 bit version Windows 8.1 Windows 10
CPU	CPU with a capacity of Core i5 or later
Memory	4 GB or larger

CAUTION	■ If a low-performance PC is used, the controllers may not be controlled reliably. When using a PC that does not satisfy the above requirements, make sure you check operation adequately before using the system in production.

18.1.2 Connection of PC and controllers

The connection type for the first controller can be USB or Ethernet. The connection type for the remaining controllers must be Ethernet.

The figure below shows the basic configuration diagram of the PC and the multiple controllers.



The RC+ API supports the RC700 controller, the RC90 controller, and a virtual controller.

NOTE

RC700 controllers, RC90 controllers, and one virtual controller can be connected at the same time.

NOTE

One virtual controller can be selected.

NOTE

One controller can be connected by the USB communication.
When using the USB communication, connect only one controller by the USB communication and connect other controllers by the Ethernet.

CAUTION	■ If the anti-virus software is installed on the PC, communication with the controllers may be disconnected abnormally when running a virus scan. To run a virus scan, disconnected the communication with the controllers beforehand.

18.2 Restrictions on controlling multiple controllers

Controlling multiple controllers has restrictions as described in the following sections.

18.2.1 Restrictions on controller options

The following controller options controlled by each controller have restrictions.

- PC vision
- Fieldbus master
- Force sensing

When one of the above three options is already connected to the active controller, these options cannot be used for other (the second or later) controllers.

18.2.2 Restrictions on simulator

Simulator window display

EPSON RC+ 7.0 simulator window can be used from the .NET application.
For details, refer to *10.1 Windows* in this manual.

If the simulator window is opened for each controller when multiple controllers are connected, the cycle time may increase by 100 to 200 msec compared to not displaying the simulator windows.

Also, if the program is executed with the simulator windows open, the CPU utilization increases near 100 % and a huge load may be put on the PC.

It is recommended to use the system with the simulator windows closed, except when debugging the program.

Collision detection

To avoid collision with peripherals using the simulator, set 15 mm or more margins around the simulator object to avoid collision detection.

Collision detection in the simulator does not guarantee the accuracy. When applying to the actual system, make sure to set the margins and check operation adequately.

For details on each restriction, refer to *8.4 Simulator Specifications and Restrictions* in *EPSON RC+ User's Guide*.

18.3 Sample Program for connecting multiple controllers

The following sections describe sample programs to connect the PC with Controller 1 and Controller 2 using a .NET application.

NOTE

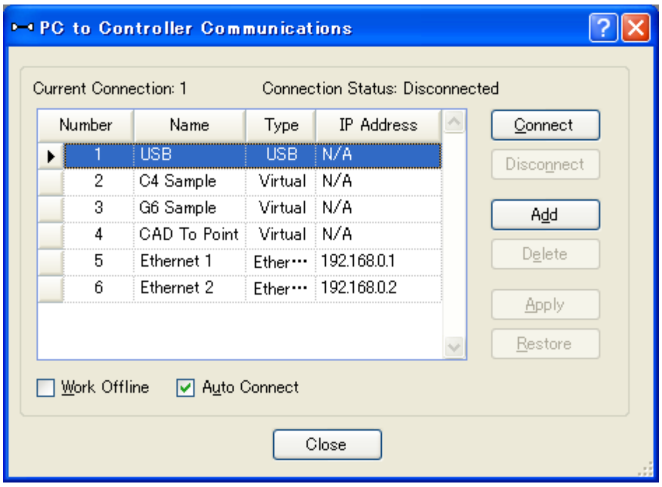
For details on available properties and methods, refer to *14. RCAPINet Reference* in this manual.

18.3.1 Controller connection setting

When connecting the multiple controllers at the same time, specify the connection using the Connect method of Spel class.

```
m_spel.Connect(1)
```

The parameter in the Connect method indicates the connection number. This number is same as the one shown in "Number" in the dialog box below (EPSON RC+ 7.0 menu-[Setup]-[PC to Controller Communications]). If a value of -1 is used, it means to use the most recent connection.



18.3.2 Project setting

To connect the multiple controllers, specify the project using the Project property of the Spel class. Each controller must use a separate project.

m_spel.Project = "c:\EpsonRC70\projects\Demo1\Demo1.sprj"

18.3.3 Sample program using Visual Basic

(1) Select menu-[File]-[New]-[Project] in Visual Studio .NET.

(2) Create a Visual Basic project.

(3) Select [Project]-[Add Reference].

(4) Select the [Browse] tab, reference "\EpsonRC70\Exe" directory, and then select the "RCAPINet.dll" file.

(5) Add two buttons (btnController1, btnController2) to the Form1 class.

(6) Add quick events of each button and create the thread to control each robot controller.

```
Private trd1 As System.Threading.Thread     ' for robot controller 1
Private trd2 As System.Threading.Thread     ' for robot controller 2

Private Sub btnController1_Click(ByVal sender As System.Object, _
             ByVal e As System.EventArgs) Handles
             btnController1.Click
        ' Start thread for robot controller 1
        trd1 = New System.Threading.Thread( _
            New System.Threading.ThreadStart(AddressOf
            StartController1))
        trd1.Start()
End Sub
Private Sub StartController1()
     ' Control robot controller 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
        ' Start thread for robot controller 2
        trd2 = New System.Threading.Thread( _
         New System.Threading.ThreadStart(AddressOf StartController2))
        trd2.Start()
End Sub
Private Sub StartController2()
    ' Control robot controller 2
    Try
       Dim frm2 As New frmDemo2
       frm2.ShowDialog()
       frm2.Dispose()
    Catch ex As Exception
       MsgBox(ex.Message)
    End Try
End Sub
```

(7) Add a form (frmDemo1) for Controller 1.

```
Private WithEvents m_spel1 As New Spel
Private Sub frmDemo1_Load(ByVal sender As System.Object, _
                   ByVal e As System.EventArgs) Handles MyBase.Load

     Try
        m_spel1.Initialize()
        m_spel1.ServerInstance=1
        m_spel1.Connect(5)
        m_spel1.Project = "
c:\\EpsonRC70\\Projects\\Demo1\\Demo1.sprj "
     Catch ex As SpelException
        MsgBox(ex.Message)
     End Try
End Sub
Private Sub m_spel1_EventReceived(ByVal sender As Object, ByVal e
As
                      SpelEventArgs) _Handles m_spel1.EventReceived
    ' for robot controller 1
End Sub
Private Sub frmDemo1_FormClosed(ByVal sender As System.Object, _
             ByVal e As System.Windows.Forms.FormClosedEventArgs) _
             Handles MyBase.FormClosed
    m_spel1.Dispose()
End Sub
```

(8) Add a form (frmDemo2) for Controller 2.

```
Private WithEvents m_spel2 As New Spel
Private Sub frmDemo2_Load(ByVal sender As System.Object, _
                   ByVal e As System.EventArgs) Handles MyBase.Load

    Try
       m_spel2.Initialize()
       m_spel2.ServerInstance=2
       m_spel2.Connect(6)
       m_spel2.Project = "
c:\\EpsonRC70\\Projects\\Demo2\\Demo2.sprj "
    Catch ex As SpelException
       MsgBox(ex.Message)
    End Try
End Sub

Private Sub m_spel2_EventReceived(ByVal sender As Object, ByVal e As
                      SpelEventArgs) _Handles m_spel2.EventReceived
    ' for robot controller 2
End Sub
Private Sub frmDemo2_FormClosed(ByVal sender As System.Object, _
                ByVal e As
System.Windows.Forms.FormClosedEventArgs) _
                Handles MyBase.FormClosed
    m_spel2.Dispose()
End Sub
```

18.3.4 Sample program using Visual C#

(1) Select menu-[File]-[New]-[Project] in Visual Studio .NET.

(2) Create a Visual C# project.

(3) Select menu-[Project]-[Add Reference].

(4) Select the [Browse] tab, reference "\EpsonRC70\Exe" directory, and then select the "RCAPINet.dll" file.

(5) Add two buttons (btnController1, btnController2) to the Form1 class.

(6) Add quick events of each button and create the thread to control each robot controller.

```
private System.Threading.Thread trd1;  // for robot controller1
private System.Threading.Thread trd2;  // for robot controller2


private void btnController1_Click(object sender, EventArgs e)
{
    // Start thread for robot controller 1
    trd1 = new System.Threading.Thread(new _
              System.Threading.ThreadStart(StartController1));
    trd1.Start();
}
private void StartController1()
{
    // Control robot controller 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)
{
    // Start thread for robot controller 2
    trd2 = new System.Threading.Thread(new _
                    System.Threading.ThreadStart(StartController2));
    trd2.Start();
}
private void StartController2()
{
    // Control robot controller 2
    try
    {
       frmDemo2 frm2 = new frmDemo2();
       frm2.ShowDialog();
       frm2.Dispose();
    }
    catch (System.Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

(7) Add a form (frmDemo1) for Controller 1.

```
private Spel m_spel1;
private void frmDemo1_Load(object sender, EventArgs e)
{
    m_spel1 = new Spel();
    try
    {
       m_spel1.Initialize();
       m_spel1.ServerInstance = 1;
```

```
            m_spel1.Connect(5);
            m_spel1.Project = "c:\\EpsonRC70\\Projects\\Demo1\\Demo1.sprj";
            m_spel1.EventReceived += new _
                  Spel.EventReceivedEventHandler(m_spel1_EventReceived);
        }
        catch (SpelException ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
    public void m_spel1_EventReceived(object sender, SpelEventArgs e)
    {
        // for robot controller 1
    }
    private void frmDemo1_FormClosed(object sender, FormClosedEventArgs e)
    {
        m_spel1.Dispose();
    }
```

(8) Add a form (frmDemo2) for Controller 2.

```
    private Spel m_spel2;
    private void frmDemo2_Load(object sender, EventArgs e)
    {
        m_spel2 = new Spel();
        try
        {
            m_spel2.Initialize();
            m_spel2.ServerInstance = 2;
            m_spel2.Connect(6);
            m_spel2.Project = "c:\\EpsonRC70\\Projects\\Demo2\\Demo2.sprj";
            m_spel2.EventReceived += new _
                  Spel.EventReceivedEventHandler(m_spel2_EventReceived);
        }
        catch (SpelException ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
    public void m_spel2_EventReceived(object sender, SpelEventArgs e)
    {
        // for robot controller 2
    }
    private void frmDemo2_FormClosed(object sender, FormClosedEventArgs e)
    {
        m_spel2.Dispose();
    }
```