EPSON

Epson RC+ 8.0 Option
GUI Builder 8.0

Original instructions

© Seiko Epson Corporation 2024-2026

Rev.6
ENM266S8659F

Table of Contents

1. FOREWORD 11

1.1 Introduction 12

1.2 Trademarks 12

1.3 TRADEMARK NOTATION IN THIS MANUAL 12

1.4 Terms of Use 12

1.5 Manufacturer 12

1.6 Contact Information 12

1.7 Before Use 13

1.7.1 The Installation Folder for Epson RC+ 8.0 13

2. Introduction 14

2.1 Introduction 15

2.2 Features 15

3. Installation 16

3.1 To install GUI Builder 17

4. Getting Started 18

4.1 Getting Started 19

4.2 GUI Builder Tutorial 19

5. The GUI Builder Environment 24

5.1 Overview 25

5.2 Basic Concepts Required to Understand GUI Builder 25

5.3 Open the GUI Builder Window 25

5.4 Parts of the GUI Builder Window 26

5.4.1 Design Area 26

5.4.2 Toolbar 26

5.4.3 Forms Explorer 27

5.4.4 Property Grid 28

5.4.5 Events Grids 28

5.4.6 Status Bar 28

5.5 How to work with forms and controls 28

5.5.1 Creating a Form 28

5.5.2 Deleting a Form 28
5.5.3 Opening and Closing Forms 29
5.5.4 Zooming In and Out a Form 29
5.5.5 Changing the Size of a Form 29
5.5.6 Editing Multiple Forms 29
5.5.7 Creating Controls 29
5.5.8 Deleting Controls 30
5.5.9 Sizing and Moving Controls 30
5.5.10 Copy, Cut, Paste for Controls 30
5.5.11 Editing Properties 30
5.5.12 Working with Event Handlers 31
5.5.13 Saving Your Work 31
5.6 Setting GUI Builder Preferences 32
5.7 Importing Forms From Other Projects 32

6. GUI Builder Components 34

6.1 Form 35
6.1.1 Description 35
6.1.2 Usage 35
6.1.3 Form Properties 36
6.1.4 Form Events 37
6.2 Button Control 37
6.2.1 Description 37
6.2.2 Usage 37
6.2.3 Button Control Properties 37
6.2.4 Button Control Events 39
6.3 Label Control 39
6.3.1 Description 39
6.3.2 Usage 39
6.3.3 Label Control Properties 40
6.3.4 Label Control Events 41
6.4 TextBox Control 41
6.4.1 Description 41
6.4.2 Usage 41
6.4.3 TextBox Control Properties 41
6.4.4 TextBox Control Events 43

6.5 RadioButton Control 43
6.5.1 Description 43
6.5.2 Usage 43
6.5.3 RadioButton Control Properties 44
6.5.4 RadioButton Control Events 45
6.6 CheckBox Control 45
6.6.1 Description 45
6.6.2 Usage 46
6.6.3 CheckBox Control Properties 46
6.6.4 CheckBox Control Events 47
6.7 ListBox Control 47
6.7.1 Description 47
6.7.2 Usage 47
6.7.3 ListBox Control Properties 48
6.7.4 ListBox Control Events 49
6.8 ComboBox Control 49
6.8.1 Description 49
6.8.2 Usage 49
6.8.3 ComboBox Control Properties 50
6.8.4 ComboBox Control Events 51
6.9 PictureBox Control 51
6.9.1 Description 51
6.9.2 Usage 51
6.9.3 PictureBox Control Properties 52
6.9.4 PictureBox Control Events 52
6.9.5 ImageList Dialog 53
6.10 GroupBox Control 53
6.10.1 Description 53
6.10.2 Usage 53
6.10.3 GroupBox Control Properties 53
6.10.4 GroupBox Control Events 54
6.11 Timer Control 54
6.11.1 Description 54
6.11.2 Usage 55
6.11.3 Timer Control Properties 55

6.11.4 Timer Control Events 55
6.12 VideoBox Control 55
6.12.1 Description 55
6.12.2 Usage 55
6.12.3 VideoBox Control Properties 56
6.12.4 VideoBox Control Events 57
6.13 LED Control 57
6.13.1 Description 57
6.13.2 Usage 57
6.13.3 LED Control Properties 57
6.13.4 LED Control Events 59
6.14 StatusBar Control 59
6.14.1 Description 59
6.14.2 Usage 59
6.14.3 StatusBar Control Properties 59
6.14.4 StatusBar Control Events 60
6.15 ProgressBar Control 60
6.15.1 Description 60
6.15.2 Usage 60
6.15.3 ProgressBar Control Properties 61
6.15.4 ProgressBar Control Events 62
6.16 TrackBar Control 62
6.16.1 Description 62
6.16.2 Usage 62
6.16.3 TrackBar Control Properties 62
6.16.4 TrackBar Control Events 63
6.17 Grid Control 63
6.17.1 Description 63
6.17.2 Usage 63
6.17.3 Grid Control Properties 64
6.17.4 Grid Control Events 65
6.17.5 GridEditor 65
6.18 Tab Control 65
6.18.1 Description 65
6.18.2 Usage 65

6.18.3 Tab Control Properties 66
6.18.4 Tab Control Events 67
6.19 Wizard Control 67
6.19.1 Description 67
6.19.2 Usage 67
6.19.3 Wizard Control Property 68
6.19.4 Wizard Control Events 69
6.20 Tree View Control 69
6.20.1 Description 69
6.20.2 Usage 69
6.20.3 Tree View Property 71
6.20.4 Tree View Event 72

7. Operation 73

7.1 Overview 74
7.2 GUI Development in Program Mode 74
7.2.1 Design the GUI 74
7.2.2 Debugging 74
7.3 Auto Mode 75
7.4 Handling Pause and Continue 75
7.5 Handling Emergency Stop 75
7.6 Using a Help File 76

8. GUI Builder Reference 77

8.1 Overview 78
8.2 GUI Builder Properties and Events Format Description 78
8.3 A 79
8.3.1 AcceptButton Property 79
8.3.2 AddItem Property 79
8.3.3 AddNode Property 80
8.3.4 AddRow Property 81
8.3.5 AppendText Property 81
8.3.6 AllowStateChange Property 82
8.4 B 83
8.4.1 BackColor Property 83
8.4.2 BackColorMode Property 83

8.4.3 BorderStyle Property 84
8.5 C 86
8.5.1 Camera Property 86
8.5.2 CancelButton Property 86
8.5.3 CellBackColor Property 87
8.5.4 CellChanged Event 88
8.5.5 CellForeColor Property 88
8.5.6 CellText Property 89
8.5.7 Checked Property 90
8.5.8 Click Event 90
8.5.9 Closed Event 91
8.5.10 Closing Event 91
8.5.11 ControlBox Property 93
8.5.12 Controls Property 93
8.5.13 Count Property 94
8.6 D 96
8.6.1 DblClick Event 96
8.6.2 DialogResult Property 96
8.6.3 DropDownStyle Property 97
8.7 E 99
8.7.1 Enabled Property 99
8.7.2 EventTaskType Property 99
8.8 F 101
8.8.1 Font Property 101
8.8.2 FontBold Property 101
8.8.3 FontItalic Property 102
8.8.4 FontName Property 102
8.8.5 FontSize Property 103
8.8.6 ForeColor Property 104
8.8.7 FormBorderStyle Property 104
8.8.8 FormTemplate Property 105
8.9 G 106
8.9.1 GClose Statement 106
8.9.2 GGet Statement 106
8.9.3 GraphicsEnabled Property 107

8.9.4 GridEditor Property 107
8.9.5 GSet Statement 108
8.9.6 GShow Statement 108
8.9.7 GShowDialog Function 109
8.9.8 GShowDialog Statement 110
8.10 H 111
8.10.1 Height Property 111
8.10.2 HelpButton Property 111
8.10.3 HelpID Property 112
8.11 I 113
8.11.1 Icon Property 113
8.11.2 Image Property 114
8.11.3 ImageAlign Property 114
8.11.4 ImageOff Property 115
8.11.5 ImageOn Property 116
8.11.6 ImageIndex Property 116
8.11.7 Interval Property 117
8.11.8 IOBit Property 117
8.11.9 IOType Property 118
8.12 K 120
8.12.1 KeyPress Event 120
8.13 L 121
8.13.1 LargeChange Property 121
8.13.2 Left Property 121
8.13.3 List Property 122
8.13.4 ListCount Property 123
8.13.5 Load Event 123
8.14 M 125
8.14.1 MaximizeBox Property 125
8.14.2 Maximum Property 125
8.14.3 MinimizeBox Property 126
8.14.4 Minimum Property 126
8.14.5 MultiLine Property 127
8.15 N 129
8.15.1 Name Property 129

8.16 O 130
8.16.1 Orientation Property 130
8.17 P 131
8.17.1 Parent Property 131
8.17.2 PasswordChar Property 132
8.17.3 PressDelay Property 132
8.17.4 PressSound Property 133
8.17.5 ProgressBarStyle Property 133
8.18 R 134
8.18.1 ReadOnly Property 134
8.18.2 RemoveNode Property 134
8.18.3 RemoveRow Property 136
8.18.4 Resize Event 136
8.18.5 RobotNumber Property 137
8.18.6 RowCount Property 137
8.19 S 139
8.19.1 Scroll Event 139
8.19.2 ScrollBars Property 139
8.19.3 SelectedNodeChanged Event 140
8.19.4 SelectedIndex Property 141
8.19.5 SelectionChanged Event 141
8.19.6 ShowDateTime Property 143
8.19.7 ShowEStop Property 143
8.19.8 ShowNavigators Property 144
8.19.9 ShowPrint Property 145
8.19.10 ShowRobot Property 145
8.19.11 ShowSafeguard Property 146
8.19.12 SizeMode Property 146
8.19.13 SmallChange Property 147
8.19.14 Sorted Property 148
8.19.15 StartPosition Property 148
8.20 T 150
8.20.1 TabAlignment Property 150
8.20.2 TabEditor Property 150
8.20.3 TabHeaderSize Property 152

8.20.4 TabHeaderTextAlign Property 154
8.20.5 TabHeaderAutoFit Property 155
8.20.6 TabHeaderHeight Property 156
8.20.7 TabHeaderWidth Property 157
8.20.8 TabIndex Property 158
8.20.9 TabPageBackColor Property 158
8.20.10 TabPageFont Property 159
8.20.11 TabPageFontBold Property 160
8.20.12 TabPageFontItalic Property 161
8.20.13 TabPageFontName Property 162
8.20.14 TabPageFontSize Property 163
8.20.15 TabPageIcon Property 164
8.20.16 TabPageText Property 165
8.20.17 Text Property 165
8.20.18 TextAlign Property 166
8.20.19 Tick Event 167
8.20.20 TickFrequency Property 167
8.20.21 TickStyle Property 168
8.20.22 ToolTipText Property 169
8.20.23 Top Property 169
8.20.24 TreeNodeEditor Property 170
8.20.25 Type Property 171
8.21 U 172
8.21.1 Update Property 172
8.22 V 173
8.22.1 Value Property 173
8.22.2 Variable Property 173
8.22.3 VideoEnabled Property 174
8.22.4 Visible Property 175
8.23 W 176
8.23.1 Width Property 176
8.23.2 WindowState Property 176
8.23.3 WizardEditor Property 177
8.23.4 WordWrap Property 178

1. FOREWORD

1.1 Introduction

Thank you for purchasing this Epson robot system. This manual provides the information necessary for correctly using the robot system.
Before using the system, please read this manual and related manuals to ensure correct use.
After reading this manual, store it in an easily accessible location for future reference.

Epson conducts rigorous testing and inspection to ensure that the performance of our robot systems meets our standards. Please note that if the Epson robot system is used outside the operating conditions described in the manual, the product will not perform up to its basic performance.

This manual describes potential hazards and problems that are foreseen. To use the Epson robot system safely and correctly, be sure to follow the safety information contained in this manual.

1.2 Trademarks

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

1.3 TRADEMARK NOTATION IN THIS MANUAL

MicrosoftR WindowsR 10 operating system
MicrosoftR WindowsR 11 operating system
Throughout this manual, Windows 10 , Windows 11 refer to above respective operating systems. In some cases, Windows refers generically to Windows 10 , Windows 11.

1.4 Terms of Use

No part of this instruction manual may be reproduced or reprinted in any form without express written permission.
The information in this document is subject to change without notice.
Please contact us if you find any errors in this document or if you have any questions about the information in this document.

1.5 Manufacturer

SEIKO EPSON CORPORATION

1.6 Contact Information

Contact information details are listed in the "Supplier" section in the following manual.
Note that the contact information may vary depending on your region.
"Safety Manual - Contact Information"

The Safety Manual is also available at the following site.
URL: https://download.epson.biz/robots/

1.7 Before Use

Before using this manual, be sure that you understand the following information.

1.7.1 The Installation Folder for Epson RC+ 8.0

You can change the path for the installation folder for Epson RC+ 8.0 anywhere. This manual assumes that Epson RC+ 8.0 is installed in C:\EpsonRC80.

2. Introduction

2.1 Introduction

The Epson RC+ GUI Builder 8.0 Option enables you create a GUI (Graphical User Interface) for your SPEL+ application. The design goal for GUI Builder was to create an easy to use integrated tool for creating a SPEL+ application GUI in the Epson RC+ development environment. It is ideal for users that need a simple GUI and do not want to use a third party product such as Visual Studio. Even users that never created a GUI before can easily make one with GUI Builder.

KEY POINTS

For advanced GUI applications, you may want to consider using the Epson RC+ 8.0 RC+ API option along with Visual Studio or another tool that can interface with the VB Guide .NET libraries.

2.2 Features

The following features are supported in the GUI Builder 8.0 package:

- Your GUI is integrated completely within the Epson RC+ environment for easy design, debugging, and display at runtime. No third party tools are required.
- You can create and debug GUI forms in your Epson RC+ project.
- Several standard controls are provided, including button, label, textbox, etc. In addition, controls are provided for displaying video, variable status, and I/O status.
- Form and control events are executed as SPEL+ tasks. You can specify whether these task run in Normal, NoPause, or NoEmgAbort modes.
- In Auto mode, Epson RC+ can automatically display your main form at startup, or you can show forms from your SPEL+ code.

3. Installation

3.1 To install GUI Builder

Please follow the instructions in this chapter to help ensure proper installation of the GUI Builder 8.0 software.
Before starting, ensure that all Windows applications have been closed.

1. Install Epson RC+ 8.0. The GUI Builder option is automatically installed.
2. Ensure that the software key has been enabled for GUI Builder 8.0 in the controller you will be using. Refer to the Epson RC+ 8.0 User’s Guide for information on how to enable options in the controller.

This completes the GUI Builder 8.0 installation.

4. Getting Started

4.1 Getting Started

This chapter contains information for getting started with GUI Builder 8.0.
Before continuing, ensure that the GUI Builder option is enabled. See the following chapter for details.
To install GUI Builder

If you have never used Epson RC+ before, you should read the Epson RC+ 8.0 User's Guide to get familiar with creating projects and programs.

The following section presents a tutorial illustrating some simple concepts.

4.2 GUI Builder Tutorial

In this section we will create a simple GUI application that runs a robot cycle. We will walk through the following tasks. For details of terms and descriptions of GUI Builder, refer to section 4.2 onward.

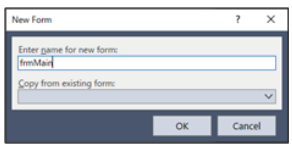
- Create a new Epson RC+ project with a function to run the robot cycle.
- Create a form with Start and Stop buttons to run the robot cycle. This shows how to create a form and add buttons with events to start and stop a SPEL+ task.
- Add Pause and Continue buttons to the form. This shows the use of the EventTaskType property.
- Add a setup form. This form will use the Label and TextBox controls to allow the user to change robot speeds.
- Add a button on the main form to display the setup form. This shows the use of the GShowDialog statement and the DialogResult property.

Follow these steps:

1. Create a new Epson RC+ 8.0 project called GUITest.
2. Using the Robot Manager, teach two robot points P0 and P1 in two different positions.
3. Add code to function main in Main.prg as shown below:

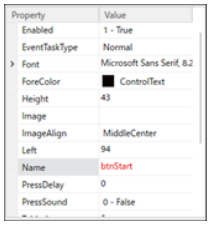
```
Function main
  Robot 1
  Motor On
  Do
    Go P0
    Wait 0.5

    Go P1
    Wait 0.5
  Loop
Fend
```

4. Select Tools | GUI Builder to open the GUI Builder window.
5. Click the New Form button on the GUI Builder window toolbar to create a form and name it frmMain. Click the OK button.

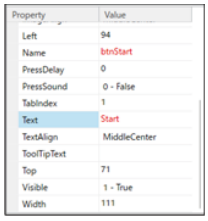


6. Click the New Button button on the GUI Builder window toolbar click the mouse on the form. A new button will be created.

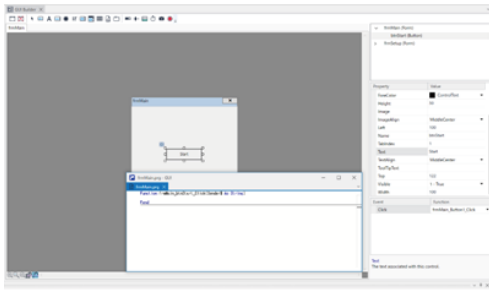
7. In the Property Grid, scroll down to the Name property,
   then change the name to btnStart and press ENTER.



8. In the Property Grid, scroll down to the Text property, then change the text from Button1 to Start and press ENTER.



9. Double click on the Start button on your form. A new program window named frmMain.prg will be opened with a new function for the button click event handler.

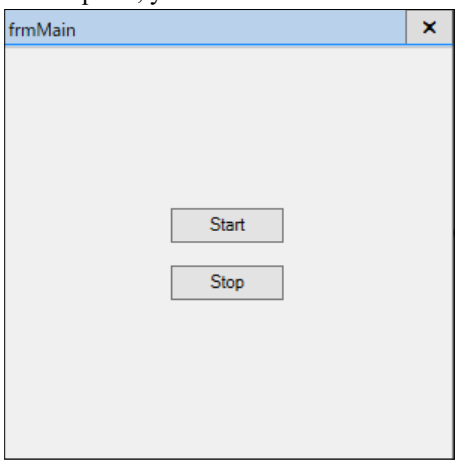


10. Change the frmMain_btnStart_Click function to start the main task as shown below.
    This will cause the main function to start when the user clicks the Start button.

```
Function frmMain_btnStart_Click(Sender$ As String)
  Xqt main
Fend
```

11. Click the New Button button on the GUI Builder window toolbar, then click on the form under the Start button to create another button.

12. For the new button, change the Name property to btnStop, and change the Text property to Stop.
    At this point, your form should look as shown below:



13. Double click the Stop button to create the button click event handler.
    Then change the code as shown below:

```
Function frmMain_btnStop_Click(Sender$ As String)
  Quit main
Fend
```

14. Press F5 to build the project and display the Run Window.
    If any build errors occur, correct your code and press F5 again.

15. Select the Form radio button on the Run Window.

16. Click the Start button on the Run Window.

17. frmMain will be displayed. Click the Start button on frmMain.
    The robot should now be moving between P0 and P1.

18. Click the Stop button on your form.
    The robot task will stop.

19. Now click the X button in the upper right of your form.
    The form will close.
    We will now add Pause and Continue buttons to our GUI.

20. Click the New Button button on the GUI Builder window toolbar and click the form to the right of the Start button to create a new button.

21. For the new button, change the Name property to btnPause, and change the Text property to Pause.

22. Change the EventTaskType for the Pause button to 1 – NoPause.
    This allows the button click event handler to execute the Pause statement without pausing the task itself.

23. Double click the Pause button to create an event handler function.
    Then change the code by adding the Pause statement as shown below:

```
Function frmMain_btnPause_Click(Sender$ As String)
  Pause
Fend
```

24. Click the New Button button on the GUI Builder window toolbar and click the form to the right of the Stop button to create a new button.

25. For the new button, change the Name property to "btnCont", and change the Text property to "Continue".

26. Change the EventTaskType for the Continue button to 1 – NoPause.
    This allows the button click event handler to execute the Cont statement when normal tasks are paused.

27. Double click the Continue button to create an event handler function.
    Then change the code by adding the Cont statement as shown below:

```
Function frmMain_btnCont_Click(Sender$ As String)
  Cont
Fend
```

28. Press F5 to build the project and display the Run Window.
    If any build errors occur, correct your code and press F5 again.

29. Click the Start button on the Run Window. Your form will be displayed.

30. Click the Start button on your form.
    The robot cycle will execute.

31. Click the Pause button on your form.
    The robot cycle will pause.

32. Click the Continue button on your form.
    The robot cycle will continue.

33. Click the Stop button on your form, then close the form by clicking the X button in the upper right corner.

We will now add a setup form to our GUI.

34. Click the New Form button the GUI Builder window toolbar and name the new form frmSetup, then click OK. A new tab page will be opened showing the new form.

35. Click the New Label button and click on frmSetup to create a label.

36. In the Property Grid, change the name of the new label to "lblSpeed".

37. Change the Text property to "Speed:".

38. Click the New TextBox button and click on frmSetup to the right of the Speed label.

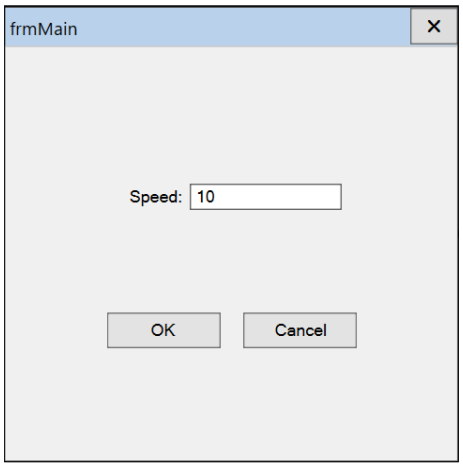
39. Change the Name property to "txtSpeed" and change the Text property to "10".

40. Double click on frmSetup to create the Load event handler function for the form and change the code as shown below:

```
Function frmSetup_Load(Sender$ As String)
  GSet frmSetup.txtSpeed.Text, Str$(Speed(1))
Fend
```

41. Click the New Button button on the GUI Builder window toolbar and click the form to add the button.
    Name this button "btnOK" and set the Text property to "OK".

42. Click the New Button button on the GUI Builder window toolbar and click the form to add the button.
    Name this button "btnCancel" and set the Text property to "Cancel". At this point, you setup form should look similar to the one shown below:



43. Double click the OK button to create an event handler function. Then change the code to set the form's DialogResult property and call GClose as shown below:

```
Function frmSetup_btnOK_Click(Sender$ As String)
  GSet frmSetup.DialogResult, DIALOGRESULT_OK
```

```
  GClose frmSetup
Fend
```

44. Double click the Cancel button to create an event handler function. Then change the code to set the form's DialogResult property and call GClose as shown below:

```
Function frmSetup_btnCancel_Click(Sender$ As String)
  GSet frmSetup.DialogResult, DIALOGRESULT_CANCEL
  GClose frmSetup
Fend
```

45. Click the frmMain tab on the GUI Builder window to work with frmMain again.

46. Click the New Button button on the GUI Builder window toolbar and click the form to add the button.
    Name this button "btnSetup" and set the Text property to "Setup".

47. Double click the Setup button to create an event handler function. Then change the code to show the setup dialog and set the new robot speed as shown below:

```
Function frmMain_btnSetup_Click(Sender$ As String)
  Integer result
  String value$

  result = GShowDialog(frmSetup)
  If result = DIALOGRESULT_OK Then
    GGet frmSetup.txtSpeed.Text, value$
    Speed Val(value$)
  EndIf
Fend
```

48. Press F5 to build the project and open the Run Window.

49. Click the Start button on the Run Window. The main form will be displayed.

50. Click the Start button on the main form.
    The robot cycle will execute.

51. Click the Setup button on the main form. The setup dialog will be displayed with the current robot speed in the textbox.

52. Enter a new speed and click OK.

53. The robot cycle is run at the new speed.
    If the Stop button is clicked and the robot stops, the motion speed will be reset to the default.
    This completes the tutorial.

5. The GUI Builder Environment

5.1 Overview

In this chapter we will focus on some concepts and definitions so you can gain a complete understanding of GUI Builder and its components. We will cover the following topics:

- Basic concepts which you should understand to use GUI Builder
- How to open the GUI Builder window
- Parts of the GUI Builder Window
- How to work with forms and controls
- Setting GUI Builder Preferences

5.2 Basic Concepts Required to Understand GUI Builder

A quick explanation of some of the basic concepts will help you understand this chapter much better. Please review the concepts described below before proceeding through the rest of this chapter.

- **What is a GUI?**
  GUI stands for Graphical User Interface. A GUI allows your operators to easily interact with your SPEL+ application to run cycles or perform setup functions. The basic building block for your GUI is the Form.

- **What is a Form?**
  A form is a window or dialog box that contains controls. It is the basic unit of your GUI application. When a form is displayed at runtime, the controls on the form are active and ready to receive keyboard and mouse events from the user. Your GUI project can have just one form or several forms.

- **What is a Control**? Controls are objects that are contained within a form, such as buttons, checkboxes, textboxes, etc. Each type of control has its own set of properties and events.

- **What is an Event?**
  An event is a SPEL+ function created by you that is called by the GUI when a form or control event occurs. For example, when the user clicks a button control, the button click event can call the SPEL+ function that you designated to run when the click occurs.

5.3 Open the GUI Builder Window

The GUI Builder window is opened from within the Epson RC+ development environment.
After Epson RC+ has been started, the GUI Builder window can be opened in 2 different ways:

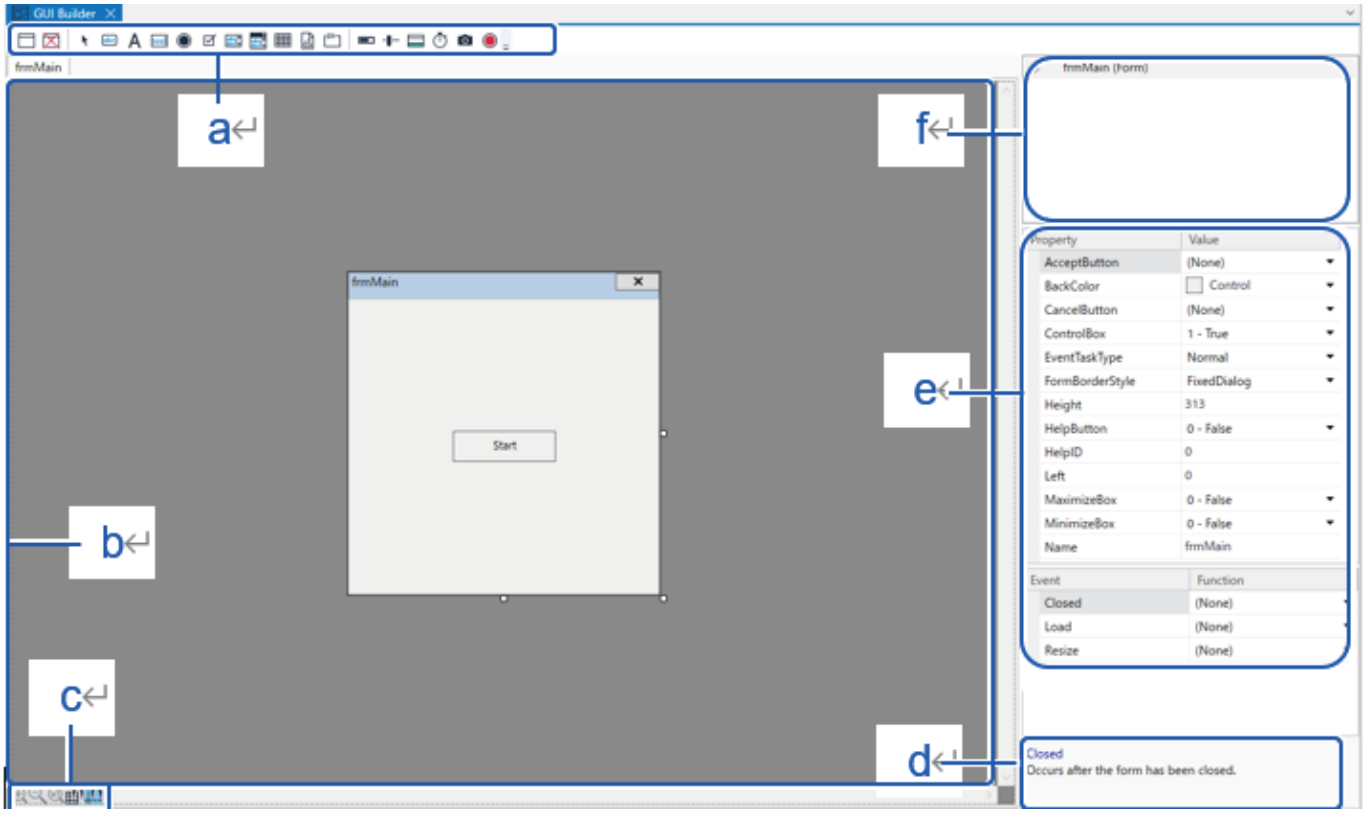
- **From the main Toolbar: From the main toolbar in Epson RC+ you should see the GUI Builder icon .**
  Clicking on the GUI Builder icon will open the GUI Builder window.

- **From the Tools menu: Selecting GUI Builder from the Tools menu will open the GUI Builder window.**

Once the GUI Builder window is open you can now begin using GUI Builder to design your GUI application.
The next few pages describe the basic parts of the GUI Builder window.

5.4 Parts of the GUI Builder Window

The GUI Builder window is shown below. Each of the indicated parts are described in the following sections.



Symbol	Item
a	Toolbar
b	Design Area
c	Status Bar
d	Events Grids
e	Property Grid
f	Forms Explorer

5.4.1 Design Area

This is where your forms are displayed at design time.
Each opened form is displayed on its own tab. You can easily switch between forms by clicking on the associated tab or by double clicking the form in the Forms Explorer.
If a form is larger than the design area, then scroll bars will be displayed and you can scroll to access all areas of the form.

5.4.2 Toolbar

The toolbar contains buttons for creating forms and controls.
The GUI Builder toolbar is located at the top of the GUI Builder window just below the title bar and appears as follows:

Shown below are general descriptions for each of the GUI Builder toolbar buttons.

Icon	Name	Description
	New form	Creates a new form. A dialog pops up and the user is asked to enter the name for the new form.
	Delete Form	Deletes a form in the current project. This button is dimmed if there are no forms for the current project.
	Pointer	Click this button to abort the addition of a new control.
	New Button	Creates a new button control.
A	New Label	Creates a new label control used to display static text.
	New TextBox	Creates a new textbox control that allows operators to input text.
	New RadioButton	Creates a new radio button control.
	New CheckBox	Creates a new checkbox control
	New ListBox	Creates a new listbox control.
	New ComboBox	Creates a new combobox control.
	New Grid	Creates a new grid control.
	New PictureBox	Creates a new picturebox.
	New GroupBox	Creates a new groupbox control.
	New ProgressBar	Creates a new progress bar control.
	New TrackBar	Creates a new track bar control.
	New StatusBar	Creates a new status bar control.
	New Timer	Creates a new timer control.
	New VideoBox	Creates a new videobox control. This control allows you to display video for Vision Guide operations.
	New LED	Creates a new LED control. This control allows you to display I/O status. You can optionally allow the operator to double click on the control to change output status. The control can use built-in pictures with various colors, or you can use your own images for off and on status.

5.4.3 Forms Explorer

The Forms Explorer is a tree that contains each form for the current project and its associated controls. When a new form or control is created, it is added to the tree.

Click on a form opens the form in its own tab in the design area. The properties and events for the form are displayed. Right click on a form to open a menu to set the start form, delete the form, or close it.

Clicking on a control in the tree activates its associated form and sets the current design focus on the control. The properties and events for the control are then displayed

5.4.4 Property Grid

The Property Grid is used to display and edit form and control properties.
When you select a form or control, the associated properties are displayed in the grid. There are two columns: Property and Value.

Property: The name of the property.
Value: The current value that you can edit.

5.4.5 Events Grids

The Events Grid is used to display and change the events for the associated form or control. Each event has a user function that is called when the event occurs.

5.4.6 Status Bar

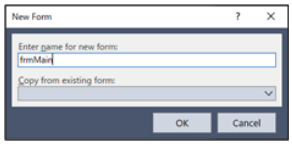
In the status bar, you can zoom in and out a form, and select a layout mode.

5.5 How to work with forms and controls

5.5.1 Creating a Form

To create a form:

1. Open the GUI Builder window.
2. Click on the New Form button on the toolbar.



3. Type a name for the form. You cannot name as follows:

- A number cannot be used at the beginning of the form name.
- You cannot use the same form name as the project name you are editing.

From the [New Form] dialog, you can optionally copy another form by selecting it from the [Copy from existing form] list.

5.5.2 Deleting a Form

To delete a form, right click on the form in the Forms Explorer, then select Delete. A confirmation message will be displayed. Click Yes to delete the form.
Also, you can use the following methods.

- Click on the [Delete Form] button on the tool bar
- Right click on the form tab, then select Delete.
- Right-click the form in the project explorer and select [Delete].

5.5.3 Opening and Closing Forms

Two ways to change the window size of a form:

- Double-click a form in the project explorer.
- Click a form in the form explorer.

To close a form, right click on the form in Forms Explorer and select Close, or right click on the form tab and select Close.

5.5.4 Zooming In and Out a Form

Two ways to adjust the display size of the form:

- After clicking anywhere in the GUI Builder window design area, move a scroll wheel of a mouse while holding down the [Ctrl] key.
- Click the following buttons on the status bar of the GUI Builder window.

Button	Description
+	Zooming in the form.
-	Zooming out the form.
×1	Makes the window the default size.

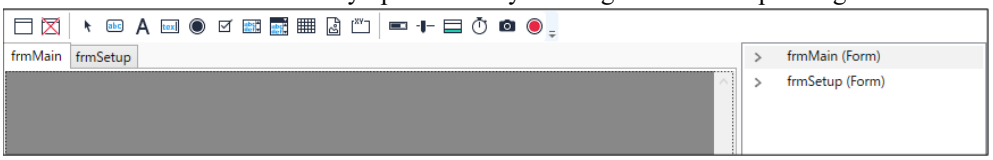
5.5.5 Changing the Size of a Form

You can change the size of a form using two methods:

- Use the mouse to drag one of the form size handles in the GUI Builder window design area.
- Change the Height and Width properties in the Property Grid at design time or using GSet at runtime.

5.5.6 Editing Multiple Forms

The GUI Builder window allows you to work with multiple forms using a tab interface.
You can switch between currently open forms by clicking on the corresponding tab.



You can copy controls from one form and paste them in another form.

5.5.7 Creating Controls

To create a control

1. Open the form to which you want to add a control.
2. Click on the control button on the GUI Builder window toolbar.
   A cross cursor will be displayed.
3. You can single click on the form near the location where you want the control and the control will be created with the default size.
   Or, you can use the mouse to draw an outline of the control size on the form and release the mouse to create the control using the size of the outline.
4. Sets each property in the property grid. The first character of a Name property cannot be a numeric character.

5.5.8 Deleting Controls

To delete a control

1. Open the form from which you want to delete one or more controls.
2. Click on the control you want to delete.
   To select more controls, hold down the Ctrl or Shift key and click on each additional control.
3. Press the Del key on the keyboard. The controls will be deleted.

5.5.9 Sizing and Moving Controls

To change the size of a control

- Use the mouse to drag one of the control's size handles in the GUI Builder window design area.
  Note
  This Operation cannot be performed when multiple controls are selected.
- Change the Height and Width properties in the Property Grid at design time or using GSet at runtime.
- While holding down the Ctrl key, click the arrow [↑/↓/←/→] in the direction you want to resize. If you want to increase the amount of resizing, hold down the Shift key in addition to the Ctrl key and click the arrows [↑/↓/←/→].

To move a control

- Click the control to activate it, then release the mouse. A move cursor is now displayed when the mouse is over the control. Click and drag the control to its new position.
- Change the Left and Top properties in the Property Grid at design time, or use GSet at runtime to change the Left and Top properties.

5.5.10 Copy, Cut, Paste for Controls

Selecting controls for copy or cut
First, click on one control. To select more controls, hold down the Ctrl or Shift key and click on each additional control. The selected controls are indicated by white (first control) and black size handles.
Copy selected controls

- To copy the selected controls, type Ctrl+C,
- or click the Copy button on the main toolbar,
- or select Copy from the Edit menu.

Cut selected controls

- To cut the selected controls, type Ctrl+X,
- or click the Cut button on the main toolbar,
- or select Cut from the Edit menu.

Pasting controls

- To paste controls that have been copied or cut, type Ctrl+V,
- or click the Paste button on the main toolbar,
- or select Paste from the Edit menu.

5.5.11 Editing Properties

To edit properties, first click on the form or control to display the associated properties in the Property Grid.

- For properties that require text input
  Click on the property to change in the Property Grid. Then type in the new value and press Enter or select another property row to apply the change.

- For properties that provide a dropdown list of values
  Click on the property to change in the Property Grid. Click the down arrow button on the right side of the value. Select the new value from the dropdown list. The change is applied after you make the selection.

- For properties that provide a button for selecting values

  Click on the property to change in the Property Grid. Click browse button on the right side of the value. Set the new value(s) from the dialog.

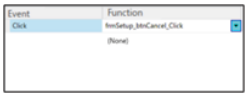
5.5.12 Working with Event Handlers

Event handlers are SPEL+ functions with the correct parameters for the specified event.
Event handler functions can be in any SPEL+ program file in the current project.
By default, event handlers are created in a program file that is created for a form. For example, when you create an event handler for a Button control Click event, a form program file is created first if it does not already exist, then the function is added to the file.

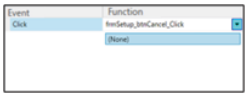
To create an event handler Three ways to create an event handler:

- Double click on the form or control to create the default event handler. For example, if you double click on a Button control, the Click event handler function is created.
- In the Events grid, double click on the event name.
- In the Events grid, select a function that already exists in the project in the value dropdown list for the event. Only functions with the correct parameters are shown in the list.

To change an event handler
You can change the event handler function by selecting another function in the event value dropdown list.



To disable an event handler
Sometimes after creating an event handler, you may decide that it is no longer needed.
To disable the event handler, select (None) in the event value dropdown list. The associated function is not deleted. But it will not be called when the event occurs.



Using EventTaskType
When an event occurs, the event handler function is started as a SPEL+ task.
The EventTaskType property lets you specify which type of task will be executed. This is important for events that need to execute in a pause condition or emergency stop condition.
For example, if a button click event handler will execute Pause or Cont, the EventTaskType must be set to 1 – NoPause. Or if an event handler must execute Reset during an emergency stop condition, then the EventTaskType must be set to 2 – NoEmgAbort.

5.5.13 Saving Your Work

After making changes in the GUI Builder window, you can save your work using three methods:

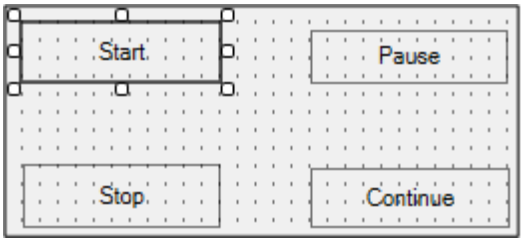
- Press Ctrl+S.

- From the File menu, select Save.
- Click the Project Save toolbar button on the Epson RC+ main toolbar.

5.6 Setting GUI Builder Preferences

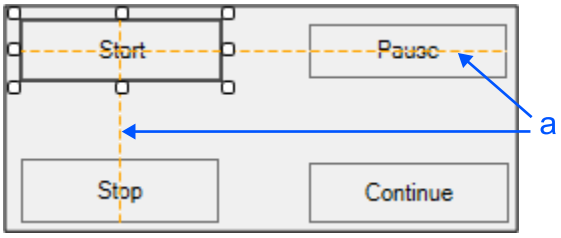
Use [Snap To Grid] and [Snap Lines] in the status bar of the GUI Builder window. You can change how controls are managed on the forms in the GUI Builder window design area by setting the GUI Builder preferences.
Click the [Snap To Grid] in the status bar.

When the [Snap To Grid] is selected, GUI Builder shows grid and places the controls on the grid.



To change the size of the grid (distance between dots), right-click the [Snap To Grid] and change the value of [Grid Width].

When the [Snap Lines] is selected, GUI Builder shows line (a) and place controls along the line.

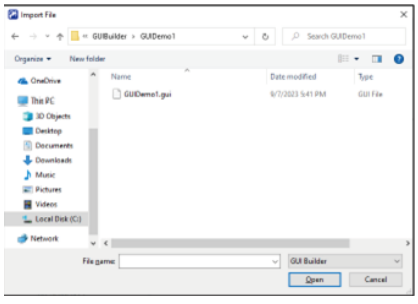


5.7 Importing Forms From Other Projects

Following shows the procedure of importing a GUI form from another project.

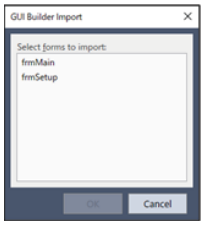
1. Select the [File] menu - [Import], or right-click the [Form] in the project explorer and select the [Add Form] - [Existing Form].

2. Navigate to the desired project folder to import from and select the GUI Builder file type.



3. Select the GUI file name and click Open.

4. A dialog will appear that has a list of forms in the project you are importing from.



5. Select one or more forms you want to import, then click OK.

6. The selected forms will be added to the current project. If a form already existed, you will be prompted whether to overwrite it or not.

KEY POINTS

When using an event handler that has been defined in another project, you must import the form's program [form name.prg].

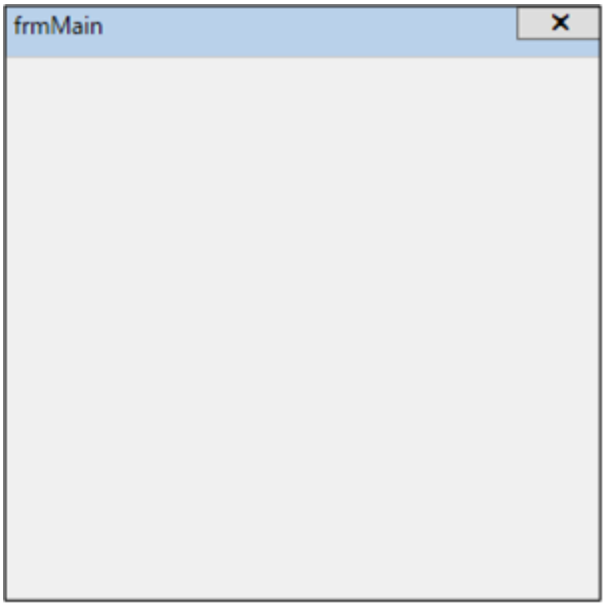
6. GUI Builder Components

This chapter describes the Form and all of the controls that can be used on a Form. Each component section has information on usage and all properties and events associated with the component.
For more details on properties, events, and statements, refer to the GUI Builder Reference chapter.

6.1 Form

6.1.1 Description

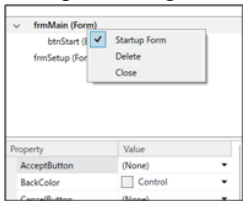
The Form is the basic building block of your GUI application. It allows you to display a window or dialog box with controls for user interaction. Your GUI application can have one form or many forms.



6.1.2 Usage

Displaying a form A Form can be displayed in four ways:

1. If the Form is set as the startup form, then in auto mode when the system starts, it is displayed automatically. To set the startup form, right click on the form in the Forms Explorer, then select Startup Form, as shown below:

   A form is shown in bold to indicate it is the startup form. You can also set the startup form for your project from Project | Properties | GUI Builder.

2. Using the GShowDialog function. This displays the form as a dialog box and returns the DialogResult value.

   ```
   result = GShowDialog(frmSetSpeed)
   ```

   Typically, the DialogResult value is set by buttons on the form, such as an OK button and a Cancel button.

3. Using the GShow statement. This displays the form as a window.

   ```
   GShow frmIODiags
   ```

4. Select the form you want to execute from the drop down list in the Run window, and click [Start] button then the selected form will be displayed.
   Setting Form Appearance
   Set the FormBorderStyle property. For forms used as dialogs, you should use 3 – FixedDialog.

Set ControlBox, MaximizeBox, MinimizeBox to configure the form's title bar.
Set the WindowState to show the form as Normal size, Maximized, or Minimized when it is displayed.

Using Help
You can display help topics from your own help file by setting the HelpButton property to True and the HelpID property to a topic ID in the help file.
For the details, refer to the following.
Using a Help File

6.1.3 Form Properties

Property	Description
AcceptButton	Sets the button control whose click event executes when the user types Enter. Default: None
BackColor	Sets the background color for the form. Default: Control
CancelButton	Sets the button control shoes click event executes when the user types Esc. After executed, the form closes. Default: None
ControlBox	Sets whether the title bar control box items are displayed. Default: True
Controls	Array of controls on the form.
Count	Gets the number of controls in the Controls array.
Dialog Result	Sets the return value when the form is closed (runtime only).
EventTaskType	Sets the type of task used for events. Default: Normal
FormBorderStyle	Sets the style of the border used for the form. Default: FixedDialog
FormTemplate	Selects and sets the size of a form from the template.
Height	Sets the height of the control in pixels.
HelpButton	Sets whether the title bar help button is displayed. Default: False
HelpID	Sets the ID for the help topic that is displayed when the user clicks the title bar help button Default: 0
Icon	Gets or sets an image to be displayed in the title bar of the form.
Left	Sets the left coordinate of the form in pixels.
MaximizeBox	Sets whether the title bar maximize button is displayed. Default: False
MinimizeBox	Sets whether the title bar minimize button is displayed. Default: False
Name	Sets the name of the control.

Property	Description
StartPosition	Sets the start position of the form. Default: CenterScreen
Text	Used to set the text of the control. Default: Name of form
Top	Gets or sets the top coordinate of the form in pixels.
Type	Gets the type name of the control.
Width	Sets the width of the form in pixels.
WindowState	Sets the default window state for the form. Default: Normal

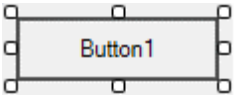
6.1.4 Form Events

Event	Description
Closed	Executes after the form is closed.
Closing	Executes before the form is closed.
Load	Executes when the form loads.
Resize	Executes when the form is resized.

6.2 Button Control

6.2.1 Description

The Button control allows the user to click it to perform an action. The Button control can display both text and images. When the button is clicked, it looks as if it is being pushed in and released.



6.2.2 Usage

Button controls allow the operator to initiate some action by clicking with the mouse.
You can change the look of a button using several properties, such as ForeColor (used for the button text), BackColor, Font, TextAlign, Image, and ImageAlign. You must provide a click event function to determine whether the operator clicked the button, and then take some action in the event handler.

6.2.3 Button Control Properties

Property	Description
BackColor	The background color for the control. Default: Control
BackColorMode	The background color mode for the control. Default: Visual Style

Property	Description
Enabled	Sets whether the control is enabled at runtime. Default: True
EventTaskType	The type of task used for events. Default: Normal
Font	The font for the control text. Default: Microsoft Sans Serif 8.25 pt
FontBold	The bold attribute for the current font Default: False
FontItalic	The italic attribute for the current font Default: False
FontName	The name for the current font . Default: Microsoft Sans Serif
FontSize	The size for the current font in points . Default: 8.25
ForeColor	The foreground color for the control text. Default: ControlText
Height	The height of the control in pixels.
Image	The picture for the control. To delete, press the [Delete] key. Default: Empty
ImageAlign	The picture alignment for the control. Default: MiddleCenter
Left	The left coordinate of the control in pixels.
Name	The name of the control. Default: ButtonXX
PressDelay	Sets the time in msec until the event is executed after the control is clicked. If you release the click button before the set time elapses, the event will not execute. Default: Immediately Execute
PressSound	Sets whether a beep sounds when the control is clicked.
TabIndex	The tab index of the control.
Text	The text of the control. Default: Name of control
TextAlign	Selects the text alignment. Default: MiddleCenter
ToolTipText	The text used in the control’s tooltip. Default: Empty
Top	The top coordinate of the control in pixels.

Property	Description
Visible	Whether the control is visible or not at runtime. Default: True
Width	The width of the control in pixels.

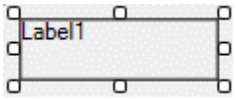
6.2.4 Button Control Events

Event	Description
Click	Executes when the user clicks the control with the mouse.

6.3 Label Control

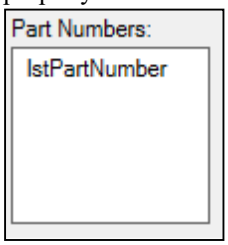
6.3.1 Description

Label controls are used to display text or images that cannot be edited by the user. They are used to identify objects on a form to provide a description of what a certain control will do if clicked, for example, or to display information in response to a run-time event or process in your application. Because the Label control cannot receive focus, it can also be used to create access keys for other controls.



6.3.2 Usage

Use the Label control to display static text on your form.
Typically, label controls are used to provide a text label for another control. For example, if you have a ListBox control on a form, you can put a Label control above the listbox with the name of the list. In the example below, a label with the Text property set to "Part Numbers:" is used to label a list box.



Setting Label Appearance

Use the BorderStyle, ForeColor, BackColor, Font, Image, ImageAlign, and TextAlign properties to change the appearance of the label.

Using hotkeys (mnemonics)

You can designate a character in the label's Text property to be a hotkey.
In the Text property of the label, use the ampersand character (&) before the hotkey.
For example, in the example shown above, the Text property for the label is "Part &Numbers:".
At runtime, when the user types Alt+N, the focus will go to the listbox.
When using hotkeys, be sure that the tab order is set so that the label's TabIndex is one before the control's TabIndex that is labeled. See the section Changing Tab Order.

6.3.3 Label Control Properties

Property	Description
BackColor	The background color for the control. Default: Control
BorderStyle	The border used for the control. Default: None
Enabled	Sets whether the control is enabled at runtime. Default: True
EventTaskType	The type of task used for events. Default: Normal
Font	The font for the control text. Default: Microsoft Sans Serif 8.25 pt
FontBold	The bold attribute for the current font Default: False
FontItalic	The italic attribute for the current font Default: False
FontName	The name for the current font Default: Microsoft Sans Serif
FontSize	The size for the current font in points Default: 8.25
ForeColor	The foreground color for the control text. Default: ControlText
Height	The height of the control in pixels.
Image	The picture for the control. To delete, press the [Delete] key. Default: Empty
ImageAlign	The picture alignment for the control. Default: MiddleCenter
Left	The left coordinate of the control in pixels.
Name	The name of the control. Default: LabelXX
TabIndex	The tab index of the control.
Text	The text of the control. Default: Name of control
TextAlign	Selects the text alignment. Default: TopLeft
ToolTipText	The text used in the control’s tooltip. Default: Empty
Top	The top coordinate of the control in pixels.

Property	Description
Visible	Whether the control is visible or not at runtime. Default: True
Width	The width of the control in pixels.

6.3.4 Label Control Events

Event	Description
Click	Executes when the user clicks the control with the mouse.
DblClick	Executes when the user double clicks the control with the mouse.

6.4 TextBox Control

6.4.1 Description

Text boxes are used to get input from the operator or to display text. The TextBox control is generally used for editable text, although it can also be made read-only.
Text boxes can display multiple lines, and wrap text to the size of the control.
The TextBox control allows a single format for text displayed or entered in the control.

6.4.2 Usage

A TextBox can be used in single line mode or multiline mode.

Single line textbox

By default, when a TextBox is created, it is in single line mode. You can only change the width of the control. The user can only type in one line of characters.

Multiline textbox

To use multiline mode, set the Multiline property to True. In this mode, you can change the height and width of the control. You can display scrollbars by setting the Scrollbars property.

Setting TextBox Appearance

Use the BorderStyle, ForeColor, BackColor, Font, and TextAlign properties to change the appearance of the textbox.

Displaying SPEL+ Global Variable (Except Arrays) Status

You can display the value of a SPEL+ global variable automatically by setting the Variable property.

NOTE
If the SPEL+ global variable is not displayed by the setting of Variable property, rebuild the project.

6.4.3 TextBox Control Properties

Property	Description
AppendText	Append text at runtime. (Only available from SPEL+ programs)
BackColor	The background color for the control. Default: Window

Property	Description
BorderStyle	The border used for the control. Default: Fixed3D
Enabled	Sets whether the control is enabled at runtime. Default: True
EventTaskType	The type of task used for events. Default: Normal
Font	The font for the control text. Default: Microsoft Sans Serif 8.25 pt
FontBold	The bold attribute for the current font Default: False
FontItalic	The italic attribute for the current font Default: False
FontName	The name for the current font Default: Microsoft Sans Serif
FontSize	The size for the current font in points Default: 8.25
ForeColor	The foreground color for the control text. Default: WindowText
Height	The height of the control in pixels.
Left	The left coordinate of the control in pixels.
Multiline	Sets whether to display one line or multiple lines. Default: False
Name	The name of the control. Default: TextBoxXX
PasswordChar	Sets the character used to hide each character that is entered. Default: Empty
ReadOnly	Sets whether the user can edit text or not. Default: False
ScrollBars	Sets how to display the scrollbars for the control. Default: None
ShowPrint	Sets whether to display output from Print statements or not. Default False
TabIndex	The tab index of the control.
Text	The text of the control. Default: Empty
TextAlign	Selects the text alignment. Default: Left

Property	Description
ToolTipText	The text used in the control's tooltip. Default: Empty
Top	The top coordinate of the control in pixels.
Update	Updates the display of textbox control (Only available from SPEL+ programs).
Variable	This is an optional SPEL+ global variable (except arrays). Default: None
Visible	Whether the control is visible or not at runtime. Default: True
Width	The width of the control in pixels.
WordWrap	Sets whether to wrap words or not. Default: True

6.4.4 TextBox Control Events

Event	Description
KeyPress	This event occurs when the control has focus and a key is pressed.
Click	Executes when the user clicks the control with the mouse.

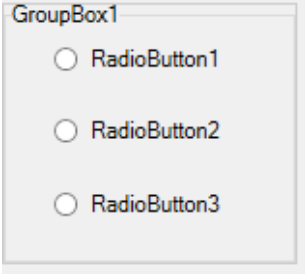
6.5 RadioButton Control

6.5.1 Description

RadioButton controls present a set of two or more mutually exclusive choices to the operator.
While radio buttons and check boxes may appear to function similarly, there is an important difference: when a user selects a radio button, the other radio buttons in the same group cannot be selected as well.

6.5.2 Usage

RadioButton controls should be grouped together in a GroupBox control.
Each time the operator clicks on one of the radiobuttons in a group, the other buttons are deselected.



Use the Click event to determine if the operator clicked the radiobutton. Use the Checked property to determine if the user checked the box.

In some cases, it may be convenient to use one click event handler for all radiobuttons in a group.
You can use the Sender$ parameter to determine which radiobutton was clicked. Sender$ is the name of the control that sent the event.

```
Function frmSetup_OptionsClick(Sender$ As String)
  Boolean checked
  GGet frmSetup.Sender$.Checked, checked
  If checked Then
    Select Sender$
      Case "RadioButton1":
        g_Option1 = True
      Case "RadioButton2":
        g_Option2 = True
    Send
  EndIf
Fend
```

Setting RadioButton Appearance
Use the BorderStyle, ForeColor, BackColor, Font, Image, ImageAlign, and TextAlign properties to change the appearance of the radiobutton.

6.5.3 RadioButton Control Properties

Property	Description
BackColor	The background color for the control. Default: Control
Checked	Sets whether the control is selected at runtime. Default: False
Enabled	Sets whether the control is enabled at runtime. Default: True
EventTaskType	The type of task used for events. Default: Normal
Font	The font for the control text. Default: Microsoft Sans Serif 8.25 pt
FontBold	The bold attribute for the current font Default: False
FontItalic	The italic attribute for the current font Default: False
FontName	The name for the current font Default: Microsoft Sans Serif
FontSize	The size for the current font in points Default: 8.25
ForeColor	The foreground color for the control text. Default: ControlText
Height	The height of the control in pixels.
Image	The picture for the control. To delete, press the [Delete] key. Default: Empty

Property	Description
ImageAlign	The picture alignment for the control. Default: MiddleCenter
Left	The left coordinate of the control in pixels.
Name	The name of the control. Default: RadioButtonXX
TabIndex	The tab index of the control.
Text	The text of the control. Default: Name of control
TextAlign	Selects the text alignment. Default: Middle Left
ToolTipText	The text used in the control's tooltip. Default: Empty
Top	The top coordinate of the control in pixels.
Visible	Whether the control is visible or not at runtime. Default: True
Width	The width of the control in pixels.

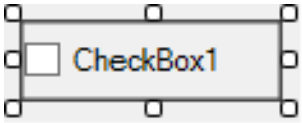
6.5.4 RadioButton Control Events

Event	Description
Click	Executes when the user clicks the control with the mouse.

6.6 CheckBox Control

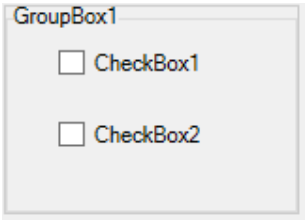
6.6.1 Description

The CheckBox control indicates whether a particular condition is on or off. It is commonly used to present a Yes/No or True/False selection to the user.
You can use check box controls in groups to display multiple choices from which the user can select one or more.
It is similar to the RadioButton control, but any number of grouped CheckBox controls may be selected.



6.6.2 Usage

Use checkboxes to allow the operator to turn application preferences on or off.
Typically, you would add the checkboxes to a GroupBox control.
Use the Checked property to determine if the user checked the box.



Setting CheckBox Appearance

Use the BorderStyle, ForeColor, BackColor, Font, Image, ImageAlign, and TextAlign properties to change the appearance of the checkbox.

6.6.3 CheckBox Control Properties

Property	Description
BackColor	The background color for the control. Default: Control
Checked	Sets whether the control is checked at runtime. Default False
Enabled	Sets whether the control is enabled at runtime. Default: True
EventTaskType	The type of task used for events. Default: Normal
Font	The font for the control text. Default: Microsoft Sans Serif 8.25 pt
FontBold	The bold attribute for the current font Default False
FontItalic	The italic attribute for the current font Default False
FontName	The name for the current font Default: Microsoft Sans Serif
FontSize	The size for the current font in points Default: 8.25
ForeColor	The foreground color for the control text. Default: ControlText
Height	The height of the control in pixels.
Image	The picture for the control. To delete, press the [Delete] key. Default: Empty
ImageAlign	The picture alignment for the control. Default: MiddleCenter

Property	Description
Left	The left coordinate of the control in pixels.
Name	The name of the control. Default: CheckBoxXX
TabIndex	The tab index of the control.
Text	The text of the control. Default: Name of control
TextAlign	Selects the text alignment. Default: MiddleLeft
ToolTipText	The text used in the control’s tooltip. Default: Empty
Top	The top coordinate of the control in pixels.
Visible	Whether the control is visible or not at runtime. Default: True
Width	The width of the control in pixels.

6.6.4 CheckBox Control Events

Event	Description
Click	Executes when the user clicks the control with the mouse.

6.7 ListBox Control

6.7.1 Description

A ListBox control displays a list of items from which the operator can select one or more.

6.7.2 Usage

Use the ListBox control to display a static list of items from which the operator can choose.

To add items to a list box

Use the AddItem property at runtime to add items to a listbox. This is normally done in the form load event.

```
GSet frmSetup.lstModels.AddItem, "Model1"
GSet frmSetup.lstModels.AddItem, "Model2"
GSet frmSetup.lstModels.AddItem, "Model3"
```

Optionally, you can have the list sorted by set the Sorted property to True.

To determine which item was selected

Use the SelectedIndex property to determine the user selection. If no item has been selected, then SelectedIndex is -1.

```
  Integer index
  GGet frmSetup.lstModels.SelectedIndex, index
```

The List array

You can access all of the items in a list using the List array property.

```
  Integer i, count
  String item$
  GGet frmSetup.lstModels.ListCount, count
  For i = 0 To count- 1
    GGet frmSetup.lstModels.List(i), item$
  Next i
```

You can delete an item by setting the List property to an empty string.

```
  GSet frmSetup.lstModels.List(0), ""
  Setting ListBox Appearance
  Use the BorderStyle, ForeColor, BackColor, Font,
  properties to change the appearance of the listbox.
```

Setting ListBox Appearance

Use the BorderStyle, ForeColor, BackColor, Font, properties to change the appearance of the listbox.

6.7.3 ListBox Control Properties

Property	Description
AddItem	Adds an item to the list at runtime. (Only available from SPEL+ programs)
BackColor	The background color for the control. Default: Window
BorderStyle	The border used for the control. Default: Fixed3D
Enabled	Sets whether the control is enabled at runtime. Default: True
EventTaskType	The type of task used for events. Default: Normal
Font	The font for the control text. Default: Microsoft Sans Serif 8.25 pt
FontBold	The bold attribute for the current font Default: False
FontItalic	The italic attribute for the current font Default: False
FontName	The name for the current font Default: Microsoft Sans Serif
FontSize	The size for the current font in points Default: 8.25
ForeColor	The foreground color for the control text. Default: WindowText

Property	Description
Height	The height of the control in pixels.
Left	The left coordinate of the control in pixels.
List	Use to access the items in the list at runtime. (Only available from SPEL+ programs)
ListCount	Use to get the number of items in the list at runtime.
Name	The name of the control. Default: ListBoxXX
SelectedIndex	Use to determine which item has been selected by the operator at runtime. (Only available from SPEL+ programs)
Sorted	Sets whether to sort the items in the list or not. Default: False
TabIndex	The tab index of the control.
ToolTipText	The text used in the control's tooltip. Default: Empty
Top	The top coordinate of the control in pixels.
Visible	Whether the control is visible or not at runtime. Default: True
Width	The width of the control in pixels.

6.7.4 ListBox Control Events

Event	Description
Click	Executes when the user clicks the control with the mouse.
DblClick	Executes when the user double clicks the control with the mouse.

6.8 ComboBox Control

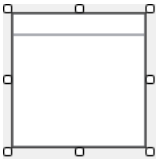
6.8.1 Description

The ComboBox control is used to display data in a dropdown combo box.
By default, the ComboBox control appears in two parts: the top part is a text box that allows the user to type a list item. The second part is a list box that displays a list of items from which the user can select one.

6.8.2 Usage

The ComboBox control can behave in three different ways, depending on the value of the DropDownStyle property.

DropDownStyle = Simple



In simple mode, there is a textbox above a list box. The operator can edit or add selections by typing in the textbox area.

DropDownStyle = DropDown

In this mode, the listbox portion is not displayed until the operator clicks the down arrow button located on the right side of the text area. The operator can edit the selected text in the textbox area.

DropDownStyle = DropDownList

When DropDownList is specified, the operator cannot type text in the textbox area. Only items in the list can be selected.
For information on how to add items to the listbox portion of the ComboBox control, see the Usage information for the ListBox control for information on AddItem, List, ListCount, and SelectedIndex.

Setting ComboBox Appearance
Use the ForeColor, BackColor, and Font properties to change the appearance of the combobox.

6.8.3 ComboBox Control Properties

Property	Description
AddItem	Adds an item to the list at runtime. (Only available from SPEL+ programs)
BackColor	The background color for the control. Default: Window
DropDownStyle	Specifies the style of the combo box. Default: DropDown
Enabled	Sets whether the control is enabled at runtime. Default: True
EventTaskType	The type of task used for events. Default: Normal
Font	The font for the control text. Default: Microsoft Sans Serif 8.25 pt
FontBold	The bold attribute for the current font Default: False
FontItalic	The italic attribute for the current font Default: False
FontName	The name for the current font Default: Microsoft Sans Serif
FontSize	The size for the current font in points Default: 8.25
ForeColor	The foreground color for the control text. Default: WindowText
Height	The height of the control in pixels.
Left	The left coordinate of the control in pixels.

Property	Description
List	Use to access the items in the list at runtime. (Only available from SPEL+ programs)
ListCount	Use to get the number of items in the list at runtime. (Only available from SPEL+ programs)
Name	The name of the control. Default: ComboBoxXX
SelectedIndex	Use to determine which item has been selected by the operator at runtime. (Only available from SPEL+ programs)
Sorted	Sets whether to sort the items in the list or not. Default: False
TabIndex	The tab index of the control.
Text	The text of the control. Default: Empty
ToolTipText	The text used in the control's tooltip. Default: Empty
Top	The top coordinate of the control in pixels.
Visible	Whether the control is visible or not at runtime. Default: True
Width	The width of the control in pixels.

6.8.4 ComboBox Control Events

Event	Description
Click	Occurs when a control is clicked.

6.9 PictureBox Control

6.9.1 Description

The PictureBox control is used to display graphics in bitmap, GIF, JPEG, PNG, and icon format.

6.9.2 Usage

Use a PictureBox control to display an image from a file.
You can set the image for a picturebox control at design time or runtime.

Setting the Image

When designing, use the Image property to add an image file on the ImageList dialog. Use the ImageIndex property to select the number of the image file to display from the added image files. To set the image file when executing, specify the full path of the image file. When connecting to RC800 Controller, change the value of the ImageIndex property to switch the images to be displayed.

Changing image size

You can specify how the image is sized by using the SizeMode property.

Setting PictureBox Appearance

Use the BackColor, BorderStyle, and SizeMode properties to change the appearance of the picturebox.

6.9.3 PictureBox Control Properties

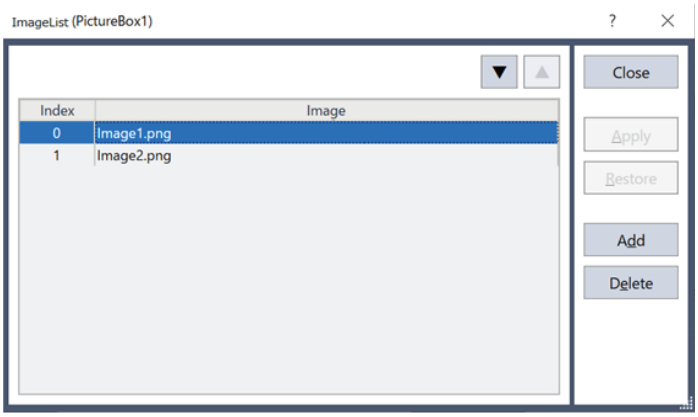
Property	Description
BackColor	The background color for the control. Default: Control
BorderStyle	The border used for the control. Default: None
Enabled	Sets whether the control is enabled at runtime. Default: True
EventTaskType	The type of task used for events. Default: Normal
Height	The height of the control in pixels.
Image	Specifies the image to be displayed. To delete, press the [Delete] key.Displays the ImageList dialog when designing. You can add, delete, and rearrange image files.
ImageIndex	Select an image file to display on the control from the image file you added in the ImageList dialog.
Left	The left coordinate of the control in pixels.
Name	The name of the control. Default: PictureBoxXX
SizeMode	Specifies how the image is sized. Default: Normal
TabIndex	The tab index of the control.
ToolTipText	The text used in the control’s tooltip. Default: Empty
Top	The top coordinate of the control in pixels.
Visible	Whether the control is visible or not at runtime. Default: True
Width	The width of the control in pixels.

6.9.4 PictureBox Control Events

Event	Description
Click	Executes when the user clicks the control with the mouse.
DblClick	Executes when the user double clicks the control with the mouse.

6.9.5 ImageList Dialog

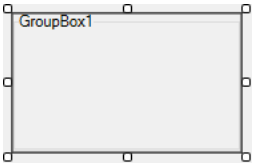
In the ImageList dialog, you can add, delete, and rearrange image files to be displayed in the picture box. You can call the ImageList dialog from the Image property.



6.10 GroupBox Control

6.10.1 Description

GroupBox controls are used to provide an identifiable grouping for other controls. Typically, you use group boxes to subdivide a form by function. For example, you may have a setup form that specifies robot speed options.. Grouping all options in a group box gives the user a logical visual cue.



6.10.2 Usage

Use the GroupBox control to group together controls for selecting options (using RadioButtons) or setting preferences (using CheckBoxes). You can also group together other controls as well.
First, add a GroupBox control to your form.
Next, either create new controls on the groupbox, or drag existing controls to the groupbox.
Set the Text property of the groupbox to the name of the group.

Setting GroupBox Appearance

Use the BackColor, ForeColor, and Font properties to change the appearance of the picturebox.

6.10.3 GroupBox Control Properties

Property	Description
BackColor	The background color for the control. Default: Control
Enabled	Sets whether the control is enabled at runtime. Default: True
EventTaskType	The type of task used for events. Default: Normal

Property	Description
Font	The font for the control text. Default: Microsoft Sans Serif 8.25 pt
FontBold	The bold attribute for the current font Default: False
FontItalic	The italic attribute for the current font Default: False
FontName	The name for the current font
FontSize	The size for the current font in points Default: 8.25
ForeColor	The foreground color for the control text. Default: ControlText
Height	The height of the control in pixels.
Left	The left coordinate of the control in pixels.
Name	The name of the control. Default: GroupBoxXX
TabIndex	The tab index of the control.
Text	The text of the control. Default: Name of control
ToolTipText	The text used in the control's tooltip. Default: Empty
Top	The top coordinate of the control in pixels.
Visible	Whether the control is visible or not at runtime. Default: True
Width	The width of the control in pixels.

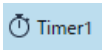
6.10.4 GroupBox Control Events

Event	Description
Click	Executes when the user clicks the control with the mouse.

6.11 Timer Control

6.11.1 Description

The Timer is a control that raises an event at regular intervals. It is executable even when the emergency stop occurs or other normal tasks are paused.



6.11.2 Usage

Use the Timer control to periodically execute code.
For example, a Timer control could be used to update a status label every two seconds.

To use a Timer

1. Add a Timer control to a form. The control is displayed below the form in the design area, since this control is invisible at runtime.
2. Set the Interval property to the desired time period in milliseconds.
3. Set the Enabled property to True at design time if the timer should always run. Otherwise, set the Enabled property to True at runtime in your code.
4. Add an event handler for the Tick event to execute your code.

KEY POINTS

When a timer is executing the Tick event handler, other tick events from the same timer are ignored.

6.11.3 Timer Control Properties

Property	Description
Enabled	Sets whether the control is enabled. Default: False
Interval	Sets the time interval in milliseconds Default: 100
Name	The name of the control. Default: TimerXX

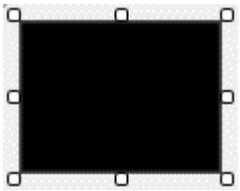
6.11.4 Timer Control Events

Event	Description
Tick	Occurs when the time is enabled and the interval has been reached.

6.12 VideoBox Control

6.12.1 Description

The VideoBox control allows you to display video for the Vision Guide option in your forms.



6.12.2 Usage

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

Perform the following steps to create a vision window:

1. Place a VideoBox control on the form you want the video to be displayed.
   The control size can be changed up to the full size.
2. Set the VideoEnabled property to True.
3. Set the GraphicsEnabled property to True if you want to display vision graphics.
4. By default, the Camera property value is 0.
   This allows the videobox to display video from any camera when a sequence is run. Set the Camera property to any camera number in the project to show the video and sequence graphics for that camera.

The video is automatically scaled to fit the size of the VideoBox. When you change the height or width of the videobox, the aspect ratio is maintained.

Setting VideoBox Appearance

Use the BorderStyle property to change the appearance of the videobox.

6.12.3 VideoBox Control Properties

Property	Description
BorderStyle	The border used for the control. Default: None
Camera	Selects which camera to display video for. Default:
Enabled	Sets whether the control is enabled at runtime. Default: True
EventTaskType	The type of task used for events. Default: Normal
GraphicsEnabled	Sets whether vision graphics are displayed or not. Default False
Height	The height of the control in pixels.
Left	The left coordinate of the control in pixels.
Name	The name of the control. Default: VideoBoxXX
TabIndex	The tab index of the control.
ToolTipText	The text used in the control’s tooltip.
Top	The top coordinate of the control in pixels.
VideoEnabled	Sets whether video is displayed on not. Default False
Visible	Whether the control is visible or not at runtime. Default: True
Width	The width of the control in pixels.

6.12.4 VideoBox Control Events

Event	Description
Click	Executes when the user clicks the control with the mouse.
DblClick	Executes when the user double clicks the control with the mouse.

6.13 LED Control

6.13.1 Description

The LED control is used to display I/O status.

6.13.2 Usage

To use the LED control

1. Set the Text property to the name of the status.
2. Set the IOType property.
   You can choose from input, output, or memory I/O.
3. Set the IOBit property. This is the bit you want to display the status for.

For outputs, you can optionally allow the operator to double click on the LED control to toggle the output status. To do this, set the AllowStateChange property to True.

Setting LED Appearance

Use the BackColor, BorderStyle, ForeColor, Font, ImageAlign, and TextAlign properties to change the appearance of the LED.

You can also change the ImageOn and ImageOff properties to use the built-in color images, or you can provide your own images.

6.13.3 LED Control Properties

Property	Description
AllowStateChange	Allows the operator to toggle output status Default: False
BackColor	The background color for the control. Default: Control
BorderStyle	The border used for the control. Default: None
Enabled	Sets whether the control is enabled at runtime. Default: True
EventTaskType	The type of task used for events. Default: Normal
Font	The font for the control text. Default: Microsoft Sans Serif 8.25 pt

Property	Description
FontBold	The bold attribute for the current font Default: False
FontItalic	The italic attribute for the current font Default: False
FontName	The name for the current font Default: Microsoft Sans Serif
FontSize	The size for the current font in points Default: 8.25
ForeColor	The foreground color for the control text. Default: ControlText
Height	The height of the control in pixels.
ImageAlign	The picture alignment for the control. Default: MiddleLeft
ImageOff	This sets the image displayed when the I/O status is off. Default: LedOff.ico
ImageOn	This sets the image displayed when the I/O status is on. Default: LedRed.ico
IOBit	Range: 0 to 9999 Default: 0
IOType	This sets the type of I/O to monitor. Default: Input
Left	The left coordinate of the control in pixels.
Name	The name of the control. Default: LedXX
TabIndex	The tab index of the control.
Text	The text of the control. Default: Name of control
TextAlign	Selects the text alignment. Default: MiddleRight
ToolTipText	The text used in the control’s tooltip. Default: Empty
Top	The top coordinate of the control in pixels.
Visible	Whether the control is visible or not at runtime. Default: True
Width	The width of the control in pixels.

6.13.4 LED Control Events

Event	Description
DblClick	Executes when the user double clicks the control with the mouse.

6.14 StatusBar Control

6.14.1 Description

The StatusBar control displays status text and optional panels for date/time, Estop status, Safeguard status, and robot information.

6.14.2 Usage

Use a StatusBar control to display status text and other built-in status. The text panel is always visible.
Use the Text property to set the text in the text panel. There are other status panels that by default are not visible: DateTime, Estop, Robot, Safeguard.
Use ShowDateTime, ShowEStop, ShowRobot, and ShowSafeguard properties to make these panels visible or not. These panels are automatically updated by the system. You can use one status bar per one form. If it is already used in a form, the [New StatusBar] is grayed out and it cannot to be clicked.

6.14.3 StatusBar Control Properties

Property	Description
BackColor	The background color for the control. Default: Control
BorderStyle	The border used for the control. Default: None
Enabled	Sets whether the control is enabled at runtime. Default: True
EventTaskType	Sets the type of task used for events. Default: Normal
Font	The font for the control text. Default: Microsoft Sans Serif 8.25 pt
ForeColor	The foreground color for the control text. Default: ControlText
Height	The height of the control in pixels.
Left	The left coordinate of the control in pixels.
Name	The name of the control. Default: StatusBarXX
RobotNumber	Specified the robot number used for ShowRobot.

Property	Description
ShowDateTime	Sets whether to display the current date and time. Default: False
ShowEStop	Sets whether to display Estop status. Default: False
ShowRobot	Sets whether to display the robot that is selected by the RobotNumber property. Default: False
ShowSafeguard	Sets whether to display the Safeguard status. Default: False
TabIndex	The tab index of the control.
Text	The text displayed in the StatusBar text panel. Default: Name of control
ToolTipText	The text used in the control's tooltip. Default: Empty
Top	The top coordinate of the control in pixels.
Visible	Whether the control is visible or not at runtime. Default: True
Width	The width of the control in pixels.

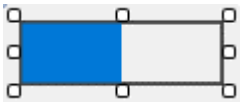
6.14.4 StatusBar Control Events

Event	Description
Click	Executes when the user clicks the control with the mouse.

6.15 ProgressBar Control

6.15.1 Description

A ProgressBar control is used to graphically display the status of processing or a value.



6.15.2 Usage

Use the ProgressBar control to show the status of a long running operation or the value of a variable.

Displaying SPEL+ Global Variable (Except Arrays) Status

You can display the value of a SPEL+ global variable automatically by setting the Variable property.

NOTE

If the SPEL+ global variable is not displayed by the setting of Variable property, rebuild the project.

6.15.3 ProgressBar Control Properties

Property	Description
BackColor	The background color for the control. Default: Control
BorderStyle	The border used for the control. Default: None
Enabled	Sets whether the control is enabled at runtime. Default: True
EventTaskType	Sets the type of task used for events. Default: Normal
ForeColor	The foreground color for the control text. Default: ControlText
Height	The height of the control in pixels.
Left	The left coordinate of the control in pixels.
Maximum	Sets the maximum of the range that the control is processing. Default: 100
Minimum	Sets the minimum of the range that the control is processing. Default: 0
Name	The name of the control. Default: ProgressBarXX
Orientation	Sets the orientation of the control. Default: Horizontal
ProgressBarStyle	Sets the style of the ProgressBar control. Default: Continuous
TabIndex	The tab index of the control.
ToolTipText	The text used in the control’s tooltip. Default: Empty
Top	The top coordinate of the control in pixels.
Value	Sets the value that specifies the progress bar position. Default: 50
Variable	This is an optional SPEL+ global variable (except arrays). When specified, the Value property is automatically set to the variable value at runtime. Default: None
Visible	Sets whether the control is visible or not at runtime. Default: True
Width	The width of the control in pixels.

6.15.4 ProgressBar Control Events

Event	Description
Click	Executes when the user clicks the control with the mouse.
DblClick	Executes when the user double clicks the control with the mouse.

6.16 TrackBar Control

6.16.1 Description

A TrackBar control is used to allow the operator to set a value by dragging a slider.

6.16.2 Usage

Use the TrackBar control when you want to support changing a value graphically.

Displaying SPEL+ Global Variable (Except Arrays) Status

You can set the value of a SPEL+ global variable automatically according to the trackbar value by using the Variable property.

NOTE

If the SPEL+ global variable is not updated by the setting of Variable property, rebuild the project.

6.16.3 TrackBar Control Properties

Property	Description
BackColor	The background color for the control. Default: Control
Enabled	Sets whether the control is enabled at runtime. Default: True
EventTaskType	Sets the type of task used for events. Default: Normal
Height	The height of the control in pixels.
LargeChange	Sets the number of positions that the slider moves when clicking the mouse on either side of the bar or by pressing Page Up and Page Down keys. Default: 5
Left	The left coordinate of the control in pixels.
Maximum	Sets the maximum of the range that the control is processing. Default: 10
Minimum	Sets the minimum of the range that the control is processing. Default: 0
Name	The name of the control. Default: TrackBarXX

Property	Description
Orientation	Sets the orientation of the control. Default: Horizontal
SmallChange	Sets the number of positions that the slider moves for keyboard inputs (arrow orientation keys). Default: 1
TabIndex	The tab index of the control.
TickFrequency	Sets the interval of the tick marks. Default: 1
TickStyle	Sets where to display the tick marks on the track bar. Default: 2- BottomRight
ToolTipText	The text used in the control's tooltip. Default: Empty
Top	The top coordinate of the control in pixels.
Value	Sets the value that represents the control's current position. Default: 50
Variable	Sets the SPEL variable whose value will be set at runtime when the component is scrolled. Default: None
Visible	Sets whether the control is visible or not at runtime. Default: True
Width	The width of the control in pixels.

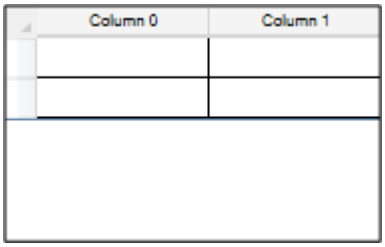
6.16.4 TrackBar Control Events

Event	Description
Scroll	Executes after the slider on the track bar has been moved.

6.17 Grid Control

6.17.1 Description

The Grid control is used to display and allow editing of data in a spreadsheet format.



6.17.2 Usage

The Grid control has cells that contain data in rows and columns. The operator can select rows and cells. The operator can optionally edit the cells in the specified columns.

At design time you can configure the Grid control:

1. Click the GridEditor property to open the designer of the grid control.
2. Design the grid by setting properties to the desired values.
3. Close the designer.

At runtime:

- You can read and write cell text using the CellText property.
- You can add or remove rows using AddRow and RemoveRow properties.
- You can change a cell forecolor and backcolor using the CellForeColor and CellBackColor properties.

6.17.3 Grid Control Properties

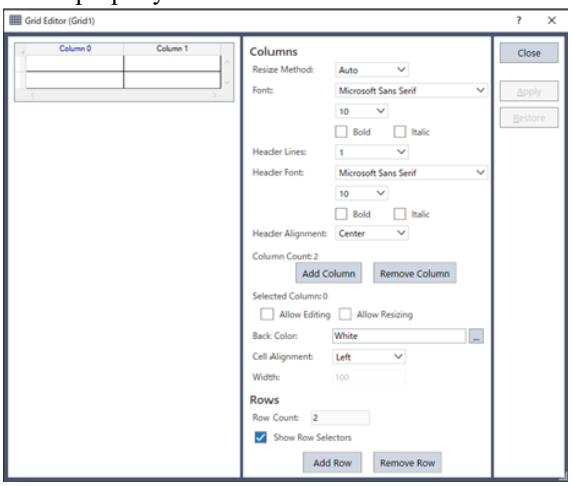
Property	Description
AddRow	Adds a row at runtime.
BorderStyle	The background color for the control. Default: Control
CellBackColor	Sets or gets the back color of a cell at runtime.
CellForeColor	Sets or gets the fore color of a cell at runtime.
CellText	Sets or gets the text in the cell at runtime.
Enabled	Sets whether the control is enabled at runtime. Default:True
EventTaskType	Sets the type of task used for events. Default: Normal
GridEditor	Opens the designer to setup the Grid control.
Height	The height of the control in pixels.
Left	The left coordinate of the control in pixels.
Name	The name of the control. Default: GridXX
RemoveRow	Removes a row at runtime.
RowCount	Sets or gets the number of rows at runtime.
ScrollBars	Sets how to display the scrollbars for the control. Default: None
SelectedIndex	Sets or gets the selected row at runtime.(Only available from SPEL+ programs)
Top	The top coordinate of the control in pixels.
Visible	Sets whether the control is visible or not at runtime. Default:True
Width	The width of the control in pixels.

6.17.4 Grid Control Events

Event	Description
CellChanged	Executes when the user leaves a cell that was changed.
Click	Executes when the user clicks the control with the mouse.
DblClick	Executes when the user double clicks the control with the mouse.

6.17.5 GridEditor

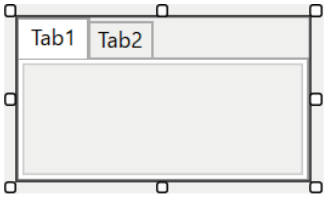
The grid editor is a dialog where the user can modify and set up the grid control. It is only available at design time.
It has many properties to set in order to best suit the user's needs. To get more information on the Grid Editor, refer to the Grid Editor property.



6.18 Tab Control

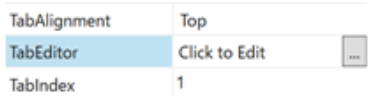
6.18.1 Description

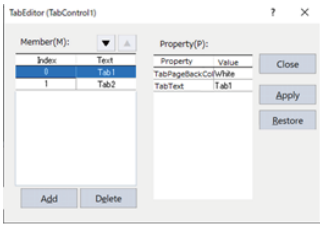
The Tab control is used to create a page that can be switched its tabs. It is possible to create a screen that allows you to switch pages to be displayed. You can create pages for different purposes, such as checking settings or status.



6.18.2 Usage

Use the Tab control to create custom pages. After placing the Tab control, click [TabEditor] to display TabEditor.





Add or delete tab pages on TabEditor. After customizing background color for each tab text and font, tab page, close TabEditor. After changing the size and the position of the Tab control, select the tab that corresponds to the page to add. The tab page switches, so place the controls to display on the tab page.

6.18.3 Tab Control Properties

Property	Description
Enabled	Sets whether the control is enabled. Default: True
EventTaskType	Sets the type of task used for events. Default: Normal
Height	Sets the height of the control in pixels.
Left	Sets the left coordinate of the control in pixels.
Name	Sets the name of the control. Default: PictureBoxXX
SelectedIndex	Use to determine which item has been selected by the operator at runtime. (Only available from SPEL+ programs)
TabAlignment	Sets the alignment of the tab on the control. Default: Top
TabEditor	Displays the TabEditor dialog for adding, removing, and rearranging tabs.
TabHeaderSize	Sets the size of tab header. (Whether to automatically adjust the width or height, manually adjust the width and height)
TabHeaderTextAlign	Sets the alignment of the tab header text. Default:Center
TabHeaderAutoFit	Sets whether to automatically adjust the tab header size to the text size. Default:True
TabHeaderHeight	Sets the height of the tab header in pixels. The value will be reflected when the TabHeaderAutoFit is "False".
TabHeaderWidth	Sets the width of the tab header in pixels. The value will be reflected when the TabHeaderAutoFit is "False".
TabIndex	Sets the tab index of the control.
ToolTipText	Sets the text used in the control's tool tip. Default: Empty
Top	Sets the top coordinate of the control in pixels.
Visible	Sets whether the control is visible or not. Default: True

Property	Description
Width	Sets the width of the control in pixels.

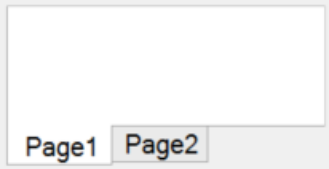
6.18.4 Tab Control Events

Event	Description
Click	Executes when the user clicks the control with the mouse.
SelectionChanged	Executes when the selected tab page is changed.

6.19 Wizard Control

6.19.1 Description

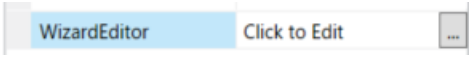
The wizard control is used to create screens in wizard format.
The wizard control can be used in the Standard and Premium editions.



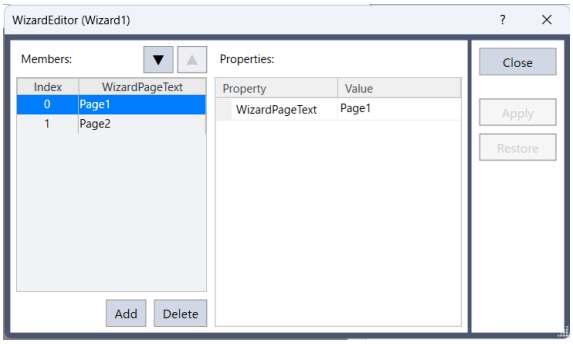
6.19.2 Usage

Multiple pages can be added on to the wizard control.
During Design:

1. Click the WizardEditor property to open the [WizardEditor] dialog.

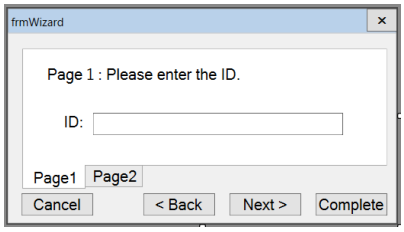


2. Add the desired page and set properties to the desired value.
3. Close the dialog.
4. Layout an optional control for each page and set properties to the desired value.



During runtime:

- Use the SelectedIndex property to switch the displayed pages.
  Example: layout the “back” and “next” button control and set the SelectedIndex property with the Click event.




```
'The Click event function for the back button
Function frmWizard_btnBack_Click(Sender$ As String)
        'Display previous page
        Integer index
        GGet frmWizard.Wizard1.SelectedIndex, index
        index = index - 1
        GSet frmWizard.Wizard1.SelectedIndex, index
Fend

'Click event function for the next button
Function frmWizard_btnNext_Click(Sender$ As String)
        'Display next page
        Integer index
        GGet frmWizard.Wizard1.SelectedIndex, index
        index = index + 1
        GSet frmWizard.Wizard1.SelectedIndex, index
Fend
```

6.19.3 Wizard Control Property

Property	Description
BackColor	Sets the background color for the control. Default: Window
Enabled	Sets whether the control is enabled. Default: True
EventTaskType	Sets the type of task used for events. Default: Normal
Height	Sets the height of the control in pixels.
Left	Sets the left coordinate of the control in pixels.
Name	Sets the name of the control. Default: WizardXX
SelectedIndex	Get and set the number of the selected item. (Only available from SPEL+ programs)
ShowNavigators	Set whether to display the header that can switch pages. (Reflected only for displaying in the design area)
TabIndex	Sets the tab index of the control.
ToolTipText	Sets the text used in the control's tool tip. Default: Empty
Top	Sets the top coordinate of the control in pixels.
Visible	Sets whether the control is visible or not. Default: True

Property	Description
Width	Sets the width of the control in pixels.
WizardEditor	Displays the wizard editor screen for adding, removing, and rearranging tabs.

6.19.4 Wizard Control Events

Event	Description
Click	Executes when the user clicks the control with the mouse.
SelectionChanged	Executes when the selected tab page is changed.

6.20 Tree View Control

6.20.1 Description

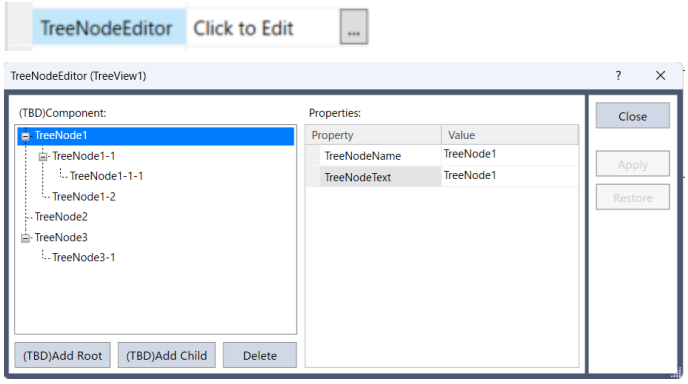
The tree view control uses the node that can be folded to display information in hierarchal order.
The tree view control can be used in the Standard and Premium editions.

6.20.2 Usage

Use the tree view control and display information in hierarchal order.

During Design:

1. Click the TreeNodeEditor property button and open the [TreeNodeEditor] dialog.



2. Add, delete, and rearrange nodes in the [TreeNodeEditor] dialog.

3. After setting the name and text for each node, close the [TreeNodeEditor] dialog.

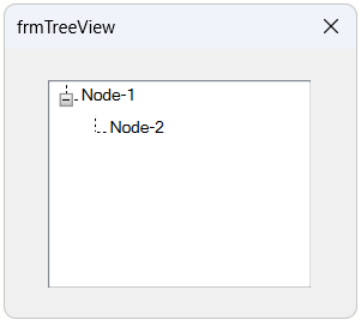
For the details of [TreeNodeEditor] dialog, refer to the following.
TreeNodeEditor Property

During runtime:
You can add, delete, rearrange, and change the text of the node by using the following properties.

- AddNode property
- RemoveNode property
- Parent property
- Text property

```
Function AddNode
    String parentNodeName$, childNodeName$
    parentNodeName$ = "Node1"
    childNodeName$ = "Node2"

    'Add node.
    GSet frmMain.TreeView1.AddNode, parentNodeName$
    GSet frmMain.TreeView1.AddNode, childNodeName$

    'Set the strings to be displayed in the node.
    GSet frmMain.TreeView1.Text(parentNodeName$), "Node-1"
    GSet frmMain.TreeView1.Text(childNodeName$), "Node-2"

    'Set the node's parent-child relationship. (Set the parent node of
childNodeName$ as parentNodeName$.)
    GSet frmMain.TreeView1.Parent(childNodeName$), parentNodeName$
Fend

Function RemoveNode(nodeName$ As String)
    'Delete node. (When the parent node is deleted, the child node will also be
deleted.)
    GSet frmMain.TreeView1.RemoveNode, nodeName$
Fend
```

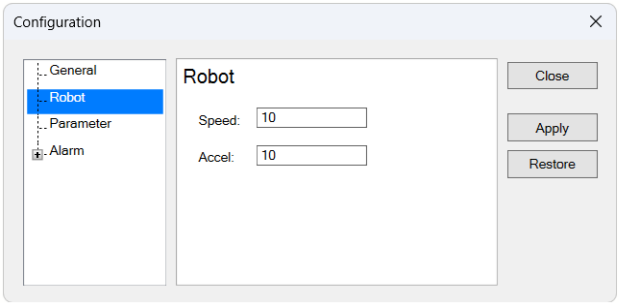
When you run the AddNode function, the following screen will be displayed.

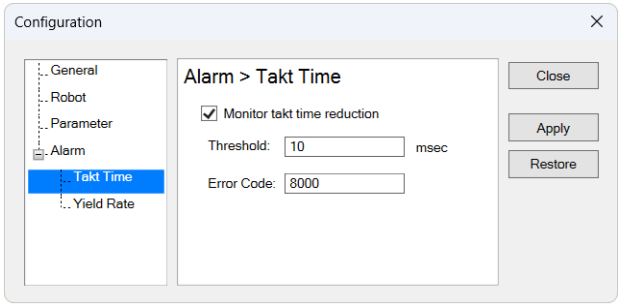


TreeView application (Setting screen for multiple categories):

By combining it with other controls, it can be applied to a screen used to set items separated into multiple categories.

1. Placing the control as follows.
   - Left: Tree View control
   - Center: Wizard control
   - Right: Button control





2. When the node selected in the tree view control is changed using the SelectedNodeChange event, the wizard control page will switch.

```
Function frmMain_TreeView1_SelectedNodeChanged(Sender$ As String, NodeName$ As
String)
    Integer wizardIndex
```

```
    ' Switch the page displayed in the wizard control based on the selected node
    Select NodeName$
        Case "TreeNode1"; wizardIndex = 0
        Case "TreeNode2"; wizardIndex = 1
        Case "TreeNode3"; wizardIndex = 2
        Case "TreeNode4"; wizardIndex = 3
        Case "TreeNode5"; wizardIndex = 4
        Case "TreeNode6"; wizardIndex = 5
    Send

    GSet frmMain.Wizard1.SelectedIndex, wizardIndex
Fend
```

6.20.3 Tree View Property

Property	Description
---	---
AddNode	Adds the specified node. (Only available from SPEL+ programs)
BackColor	Sets the background color for the control. Default: Window
BorderStyle	Sets the border used for the control. Default: Fixed3D
Enabled	Sets whether the control is enabled. Default: True
EventTaskType	Sets the type of task used for events. Default: Normal
Font	Sets the font for the control text. () Default: Microsoft Sans Serif 8.25 pt
FontBold	The bold attribute for the current font. Default: False
FontItalic	The italic attribute for the current font. Default: False
FontName	Sets the font name of the control. Default: Microsoft Sans Serif
FontSize	The size for the current font in points. Default: 8.25
ForeColor	Sets the foreground color for the control text. Default: WindowText
Height	Sets the height of the control in pixels.
Left	Sets the left coordinate of the control in pixels.
Name	Sets the name of the control. Default: TreeViewXX
Parent	Gets and sets the specified node. (Only available from SPEL+ programs)
RemoveNode	Deletes the specified node. (Only available from SPEL+ programs)
TabIndex	Sets the tab index of the control.
Text	Sets the displayed text of the control. Default: Blank

Property	Description
ToolTipText	Sets the text used in the control's tool tip. Default: Blank
Top	Sets the top coordinate of the control in pixels.
TreeNodeEditor	Displays the tree node editor screen for adding, removing, and rearranging nodes.
Visible	Sets whether the control is visible or not. Default: True
Width	Sets the width of the control in pixels.

6.20.4 Tree View Event

Event	Description
Click	Executes when the user clicks the control with the mouse.
SelectedNodeChanged	Executes when the selected node is changed.

7. Operation

7.1 Overview

This chapter contains the following topics:

- Developing your GUI in Program Mode
- Configuring your GUI startup in Auto Mode
- Handling pause and continue
- Handling emergency stop
- Using a help file

7.2 GUI Development in Program Mode

You design and debug your GUI application in program mode.
To begin, start Epson RC+ 8.0 in program mode.

7.2.1 Design the GUI

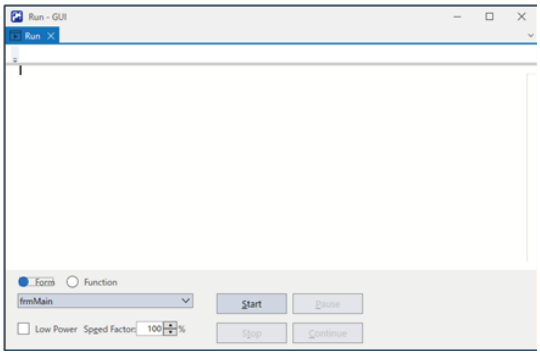
To design the GUI for your application, follow these steps as a simple guideline.

1. Create one or more forms for the application. Typically you will designate one of the forms to be your main form.
2. Decide on how you want to start the GUI. You can set one of the forms as your startup form.
   This form will automatically be displayed when RC+ starts in auto mode.
   If you do not want to display a form at startup, then the Operator Window will be displayed and you can display any form from SPEL+ code using GShow or GShowDialog.
3. Add controls to the forms.
4. Add desired events to the controls.
5. For details on working with forms and controls, see the following chapter.
   How to work with forms and controls

7.2.2 Debugging

You can run any of the forms in the project from the Run Window.

1. Open the Run Window



2. Select the form you want to run from the dropdown list.
3. Click Start. The selected form will be displayed and you can use the controls on the form.
4. To stop, close all forms, or click the Stop button on the Run Window.

GUI event handlers run as SPEL+ tasks, so you can set a breakpoint in any event handler, step through code, and view variable values.

7.3 Auto Mode

After you have developed your application, you will need to setup Epson RC+ to run in Auto mode. At startup time, you can choose to show one of your GUI forms, or you can show the Operator Window and then display one of your forms from your program.

To set Epson RC+ to start in Auto mode

1. Open Setup | System Configuration.
2. In the Start Mode page, check Auto and click Apply, then click Close.

To configure a form to be displayed at startup

1. Open the GUI Builder window.
2. In the Forms Explorer, right click on the form you want displayed at startup and select Startup Form.
3. Click the Project Save button on the main toolbar.

7.4 Handling Pause and Continue

You may want to use the pause and continue feature of the controller in your GUI.
Typically, you will have a dialog with a pause button and a continue button. For these buttons to work properly, you need to set the EventTaskType property for each button to 1 – NoPause.
This is because when the button click event executes, the event task must ignore the pause state of the controller.

```
Function frmMain_btnPause_Click(Sender$ As String)
    Pause
Fend
Function frmMain_btnCont_Click(Sender$ As String)
    Cont
Fend
```

7.5 Handling Emergency Stop

After an emergency stop, you must reset the emergency stop condition before you can run the robot again.
To do this, provide a reset button with the EventTaskType set to 2 – NoEmgAbort.

```
Function frmMain_btnReset_Click(Sender$ As String)
    Reset
    If EstopOn Then
        MsgBox "EStop could not be reset"
    EndIf
Fend
```

⚠ CAUTION

Errors that occur during program execution cannot be cleared by Reset. To do so, execute Reset Error with a different function.

```
Function frmMain_btnResetError_Click(Sender$ As String)
    Reset Error
    MsgBox "Reset Error Excuted"
Fend
```

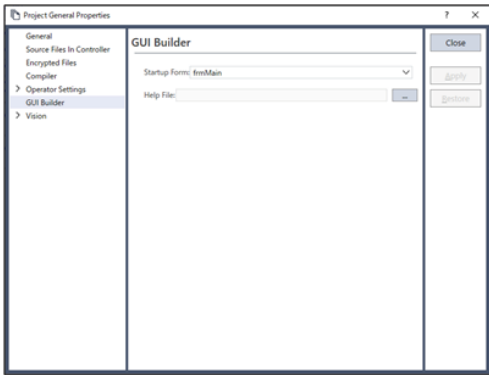
To execute Reset Error from a program, go to the Epson RC+ [Setup]-[System Configuration]-[Controller]-[Preferences], check [Enable advanced task commands] checkbox, and set the EventTaskType property to 2-NoEmgAbort.

7.6 Using a Help File

GUI Builder supports a simple help system by allowing you to specify a help file for the project from which you can show a topic when a form help button is clicked by the operator.
Help files must be in the Microsoft HTML help format.

To use a help file:

1. Open Project | Properties | GUI Builder.
2. Browse to the help file and click open to set the help file.



3. For each form that will show help, set the HelpButton property to True, and set the HelpID property for the topic you want to display.

8. GUI Builder Reference

8.1 Overview

This chapter explains all GUI Builder form and control properties and events, and all GUI Builder SPEL+ commands.
For more information on how to use GUI Builder, refer to the GUI Builder manual.

8.2 GUI Builder Properties and Events Format Description

All GUI Builder properties and events are listed in the pages that follow.
An explanation of the headings for the property and result reference pages is given below:

Item	Description
Applies To	If the property or event is used with GUI Builder objects, then this section simply lists the objects for which this property applies. (Ex. Button, Label, Checkbox...) If the property or event is used with forms then the word Forms will appear in this section.
Description	A simple description is given for each property or event. This section is normally very short for simplicity.
Usage	The Usage Section describes how to access the property or event from the SPEL+ Language.
Values	Describes the range of acceptable values which the property can be set to or which the result will return. A default value is also shown for those properties that have a default.
Remarks	Explains more details than the Description Section. This section is normally used to describe any caveats or special information that may apply to the specific property or event. (It is highly recommended to read the Remarks Section for each property prior to its usage.)
See Also	Gives a list of related properties, results, objects and other topics that may prove useful to review.
Runtime only	This is displayed under the property or result name when it applies. Runtime only properties and results cannot be accessed from the GUI Builder. They can only be accessed from the SPEL+ language.

8.3 A

8.3.1 AcceptButton Property

Applies To
Form

Description
Gets or sets the button on the form that is clicked when the user presses the ENTER key.

Usage

```
GGet Form.AcceptButton, var
GSet  Form.AcceptButton, value
```

Form
: Name of a form or string variable containing a form name.

var
: String variable that will contain the value of the property.

value
: String expression for the new value of the property.

Values
The name of any button on the form.
Default: None

Remarks
This property allows you to designate a default action to occur when the user presses the ENTER key in your application. The button assigned to this property must be a Button that is on the current form or located within a container on the current form. You can use this property to allow the user to quickly navigate a simple form by allowing them to simply press the ENTER key when they are finished instead of manually clicking the accept button with their mouse.
The accept button might not be activated if the currently selected control on the form intercepts the ENTER key and processes it. For example, a multiline text box control allows the ENTER key to be pressed when it is selected to insert a new line character in the control.

See Also
Form, **Button Control**, **CancelButton Property**

Example

```
GSet frmMain.AcceptButton, "btnOK"
```

8.3.2 AddItem Property

Applies To
ListBox, ComboBox

Description
Used to add an item to a ListBox or ComboBox control.

Usage

```
GSet Form.Control.AddItem, value
```

Form
Name of a form or string variable containing a form name.
Control
Name of a control or string variable containing a control name. The control must exist in the specified form.
value
String expression for the new value of the property.

Values
A string value to be added to the list.

See Also
ComboBox Control, **ListBox Control**, **List Property**, **ListCount Property**, **Sorted Property**

Example

```
GSet frmMain.lstModels.AddItem, "Model1"
```

8.3.3 AddNode Property

Applies To
TreeView

Description
Add an item (node) to the tree view. Added to the first level. Continue to set the Parent property to hang it as a child of other items.

Usage

```
GSet Form.Control.AddNode, value
```

Form
Name of a form or string variable containing a form name.
Control
Name of a control or string variable containing a control name. The control must exist in the specified form.
value
String expression for the new value of the property

Values
A string that indicates the node to be added to the tree view This is different from the name displayed.

See Also
Tree View Control, **Parent Property**, **RemoveNode Property**, **Text Property**

Example

```
GSet frmMain.lstModels.AddNode, "Model1"
GSet frmMain.lstModels.AddNode, "Model1-1"
GSet frmMain.lstModels.Parent("Model1-1"), "Model1"
```

8.3.4 AddRow Property

Applies To
Grid

Description
Used to add a row to a Grid control during runtime.

Usage

```
GSet Form.Control.AddRow, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name. The control must exist in the specified form.

value
: Integer representing the index of the newly added row. Specifying “-1” causes the new row to be appended after last row. Otherwise, the new row is inserted at the specified row index.

See Also
Grid Control, **RemoveRow Property**, **RowCount Property**

Example

```
GSet frmMain.Grid01.AddRow, 0
```

8.3.5 AppendText Property

Applies To
TextBox

Description
Appends text to a TextBox control.

Usage

```
GSet Form.Control.AppendText, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name. The control must exist in the specified form.

value
: String expression for the new value of the property.

Values
A string that will be added to the end of the text in the control.

See Also
TextBox Control

Example

```
GSet frmMain.txtStatus.AppendText, "Cycle Complete"
```

8.3.6 AllowStateChange Property

Applies To

LED

Description

Gets or sets the inversion actuation of LED control when double-clicked.

Usage

```
GGet Form.Control.AllowStateChange, var
GSet  Form.Control.AllowStateChange, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name. The control must exist in the specified form.

var
: Boolean variable that will contain the value of the property.

value
: Boolean expression for the new value of the property.

Values

False
True

Default: False

Remarks

To invert output, set IOType property in 1 - Output or 2 - Memory.

See Also

LED Control, **IOType Property**

Example

```
GSet frmMain.Led1.AllowStateChange, True
```

8.4 B

8.4.1 BackColor Property

Applies To

Form, Button, Label, TextBox, RadioButton, CheckBox, ListBox, ComboBox, PictureBox, GroupBox, LED, ProgressBar, TrackBar, StatusBar, Wizard, TreeView

Description

Gets or sets the background color of a form or control.

Usage

```
GGet Form.Control.BackColor, var
GSet  Form.Control.BackColor, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: String variable that will contain the value of the property.

value
: String expression for the new value of the property.

Values

The name of the color to be used for the background color of the form or control.
Default:

- Control (Form, Button, Label, RadioButton, CheckBox, PictureBox, GroupBox, LED)
- Window (TextBox, ListBox, ComboBox)

See Also

BackColorMode Property, **Form**, **Button Control**, **Label Control**, **TextBox Control**, **RadioButton Control**, **CheckBox Control**, **ListBox Control**, **ComboBox Control**, **PictureBox Control**, **GroupBox Control**, **LED Control**, **ProgressBar Control**, **TrackBar Control**, **StatusBar Control**, **Wizard Control**, **Tree View Control**

Example

```
GSet frmMain.lblStatus.BackColor, "Red"
```

8.4.2 BackColorMode Property

Applies To

Button

Description

Gets or sets the BackColorMode used for a button control.

Usage

```
GGet Form.Control.BackColorMode, var
GSet  Form.Control.BackColorMode, value
```

Form

Name of a form or string variable containing a form name.

Control
Name of a control or string variable containing a control name.

var
Integer variable that will contain the value of the property.

value
Integer expression for the new value of the property.

Values

0 - VisualStyle. Uses the system specified BackColor.

1 - User. Uses the specified BackColor.

Default: Visual Style

Remarks

When BackColorMode is 0, the background color for the control is set by Windows, depending on the current theme.

When BackColorMode is 1, the background color is set by the BackColor property value.

See Also

BackColor Property, **Button Control**

Example

```
GSet frmMain.btnOK.BackColorMode, BACKCOLORMODE_USER
```

8.4.3 BorderStyle Property

Applies To

Label, TextBox, ListBox, PictureBox, VideoBox, LED, ProgressBar, StatusBar, Grid, TreeView

Description

Gets or sets the control's border style.

Usage

```
GGet Form.Control.BorderStyle, var
GSet  Form.Control.BorderStyle, value
```

Form
Name of a form or string variable containing a form name.

Control
Name of a control or string variable containing a control name.

var
Integer variable that will contain the value of the property.

value
Integer expression for the new value of the property.

Values

0 - None

1 - FixedSingle

2 - Fixed3D

Default:

- None (Label, PictureBox, VideoBox, LED)
- Fixed3D (TextBox, ListBox)

Remarks

You can use this property to add a border to the control. For example, this property can be used to differentiate a Label that labels another control from a Label that displays the status of a process in an application.

See Also

Label Control, **TextBox Control**, **ListBox Control**, **PictureBox Control**, **VideoBox Control**, **LED Control**, **ProgressBar Control**, **StatusBar Control**, **Tree View Control**

Example

```
GSet frmMain.lblStatus.BorderStyle, BORDERSTYLE_NONE
```

8.5 C

8.5.1 Camera Property

Applies To
VideoBox

Description
Gets or sets the camera number for the VideoBox control.

Usage

```
GGet Form.Control.Camera, var
GSet  Form.Control.Camera, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: Integer variable that will contain the value of the property.

value
: Integer expression for the new value of the property.

Values
The camera number for the video display. If the value is 0, then the video from any camera is displayed when VRun executes. To display the video, set the VideoEnabled property to “True”.

See Also
VideoBox Control, **VideoEnabled Property**

Example

```
GSet frmMain.VideoBox1.Camera, 1
```

8.5.2 CancelButton Property

Applies To
Form

Description
Gets or sets the button on the form that is clicked when the user presses the ESC key.

Usage

```
GGet Form.CancelButton, var
GSet  Form.CancelButton, value
```

Form
: Name of a form or string variable containing a form name.

var
: String variable that will contain the value of the property.

value
: String expression for the new value of the property.

Values
The name of any button on the form.
Default: None

Remarks
The cancel button for a form is the button control that is clicked whenever the user presses the ESC key. The button assigned to this property must be a button that is on the current form or located within a container on the current form.
This property allows you to designate a default action to occur when the user presses the ESC key in your application. You can use this property to allow the user to quickly navigate a simple form by allowing them to simply press the ESC key to close a window without committing changes instead of manually clicking the cancel button with their mouse.
CancelButton may not work if another control on the form intercepts the ESC key. For example, if you have a ComboBox open on your form, ESC will close the ComboBox instead of closing the Form.
The Button control assigned to CancelButton must be visible on the form, or else pressing the ESC key will have no effect.

See Also
Form, **Button Control**, **AcceptButton Property**

Example
```
GSet frmMain.CancelButton, "btnCancel"
```

8.5.3 CellBackColor Property

Applies To
Grid

Description
Gets or sets the background color for a Grid cell at runtime.

Usage
```
GGet Form.Control.CellBackColor (row, column), var
GSet  Form.Control.CellBackColor (row, column), value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

row
: Integer expression for the cell row. The index of the first row is “0”.

column
: Integer expression for the cell column. The index of the first column is “0”.

var
: String variable that will contain the value of the property.

value
: String expression for the new value of the property.

Values
A string containing the name of the background color for the specified grid cell.

See Also
Grid Control, **CellForeColor Property**, **CellText Property**

Example

```
GSet frmMain.gridMyData.CellBackColor(1, 1), "Red"
```

8.5.4 CellChanged Event

Applies To

Grid

Description

Occurs when you leave a cell that has been changed.

Usage

```
Form_Control_CellChanged
(sender$ As String, CellText$ As String, RowNumber As Integer, ColumnNumber As
Integer)
```

Sender$
: Name of a control that sent the event.

CellText$
: Text of the changed cell.

RowNumber
: Row number of the changed cell.

ColumnNumber
: Column number of the changed cell.

Remarks

The CellChanged event is used to respond to when the user leaves a cell that has been modified. You can use the Sender$ parameter to determine which control sent the event, the CellText$ parameter to see what the value the cell now has, and the RowNumber and ColumnNumber parameters to determine which cell triggered the event. This is useful when you want to see when specific cells are changed or whenever any cell is changed.

See Also

Grid Control

Example

```
Function frmMain_Grid1_CellChanged(Sender$ As String, CellText$ As
String, RowNumber As Integer, ColumnNumber As Integer)

Xqt main
Fend
```

8.5.5 CellForeColor Property

Applies To

Grid

Description

Gets or sets the foreground color for a Grid cell at runtime.

Usage

```
GGet Form.Control.CellForeColor (row, column), var
GSet  Form.Control.CellForeColor (row, column), value
```

Form

Name of a form or string variable containing a form name.

Control
Name of a control or string variable containing a control name.

row
Integer expression for the cell row. The index of the first row is 0.

column
Integer expression for the cell column. The index of the first column is 0.

var
String variable that will contain the value of the property.

value
String expression for the new value of the property.

Values

A string containing the name of the foreground color for the specified grid cell.

See Also

Grid Control, **CellBackColor Property**, **CellText Property**

Example

```
GSet frmMain.gridMyData.CellForeColor(1, 1), "Red"
```

8.5.6 CellText Property

Applies To

Grid

Description

Gets or sets the text in a Grid cell at runtime.

Usage

```
GGet Form.Control.CellText (row, column), var
GSet  Form.Control.CellText (row, column), value
```

Form
Name of a form or string variable containing a form name.

Control
Name of a control or string variable containing a control name.

row
Integer expression for the cell row. The index of the first row is 0.

column
Integer expression for the cell column. The index of the first column is 0.

var
String variable that will contain the value of the property.

value
String expression for the new value of the property.

Values

A string containing the text of the specified grid cell.

See Also

Grid Control, **CellForeColor Property**, **CellBackColor Property**

Example

```
GSet frmMain.gridMyData.CellText(1, 1), "Value1"
```

8.5.7 Checked Property

Applies To

RadioButton, Checkbox

Description

Gets or sets a value indicating whether the CheckBox or RadioButton control is checked.

Usage

```
GGet Form.Control.Checked, var
GSet  Form.Control.Checked, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: Boolean variable that will contain the value of the property.

value
: Boolean expression for the new value of the property.

Values

False
True
Default: False

Remarks

Use this property to determine whether the CheckBox or RadioButton control is checked. This property can also be used to programmatically set the state of the CheckBox or RadioButton control.

See Also

RadioButton Control, **CheckBox Control**

Example

```
GSet frmMain.chkHiPower.Checked, False
```

8.5.8 Click Event

Applies To

Button, Label, RadioButton, Checkbox, ListBox, ComboBox, PictureBox, GroupBox, VideoBox, TextBox, ProgressBar, StatusBar, Grid, TabControl, Wizard, TreeView

Description

Occurs when a control is clicked.

Usage

```
Form_Control_Click (Sender$ As String)
```

Sender$
: Name of a control that sent the event.

Remarks

The Click event is used to respond to a click on a control by the user. You can use the Sender$ parameter to determine which control sent the event. This is useful when you want to use one function to handle more than one control, such as for radio buttons and check boxes.

See Also

Button Control, **Label Control**, **RadioButton Control**, **CheckBox Control**, **ListBox Control**, **ComboBox Control**, **PictureBox Control**, **GroupBox Control**, **VideoBox Control**, **ProgressBar Control**, **StatusBar Control**, **Wizard Control**, **Tree View Control**

Example

```
Function frmMain_btnStart_Click(Sender$ As String)  Xqt mainFend
```

8.5.9 Closed Event

Applies To

Form

Description

Occurs when a form has been closed.

Usage

```
Form_Closed (Sender$As String)
```

Sender$
Name of the form that sent the event.

Remarks

You can use this event to perform tasks after a form has been closed.

See Also

Form, **Load Event**, **Resize Event**

Example

```
Function frmMain_Closed(Sender$ As String)
  Print "frmMain was closed"
Fend
```

8.5.10 Closing Event

Applies To

Form

Description

This is an event that is run before the form is closed.

Usage

```
Form_Closing(Sender$ As String, ByRef Cancel As Boolean)
```

Sender$
Sets the name of the form that sent the event.

Cancel
Sets whether to cancel closing the form.

Sender$
Sets the name of the form that sent the event.

Remarks

You can use this event to perform tasks before a form has been closed.
You can also cancel closing the form by setting Cancel as True.

See Also

Form

Example

```
Global Boolean g_CanClose

Function Form_Closing(Sender$ As String, ByRef Cancel As Boolean)
        If g_CanClose = False Then
                'Cancel closing the form
                Cancel = True
        EndIf
Fend
```

8.5.11 ControlBox Property

Applies To
Form

Description
Gets or sets a value indicating whether a control box is displayed in the caption bar of the form.

Usage

```
GGet Form.ControlBox, var
GSet  Form.ControlBox, value
```

Form
: Name of a form or string variable containing a form name.

var
: Boolean variable that will contain the value of the property.

value
: Boolean expression for the new value of the property.

Values
False
True
Default: True

Remarks
If the ControlBox property is set to true, the control box is displayed in the upper-left corner of the form's title bar. The control box can be used to close the form.

See Also
Form, **MaximizeBox Property**, **MinimizeBox Property**

Example

```
GSet frmMain.ControlBox, False
```

8.5.12 Controls Property

Applies To
Form

Description
Array of controls in the form. Used to access control properties with an index.

Usage

```
GGet Form.Controls(Index As Integer).Property, var
GSet  Form. Controls(Index As Integer).Property, value
```

Form
: Name of a form.

Index
: Integer expression containing the index of a control on the form.

Property
: Name of the control property to access.

var
Variable that will contain the value of the property. The data type depends on which property is specified.
value
Expression for the new value of the property. The data type depends on which property is specified.

Remarks
Use the Controls property to access controls on the form using an index. This allows you to iterate through the control collection and get or set common properties.

See Also
Count Property, **Type Property**

Example

```
Integer i, count
String type$

GGet frmMain.Controls.Count, count
For i = 1 To count
        GGet frmMain.Controls(i).Type, type$
        If type$ = "Button" Then
                GSet frmMain.Controls(i).Enabled, False
        EndIf
Next i
```

8.5.13 Count Property

Applies To
Form.Controls

Description
Gets the number of controls on a form.

Usage

```
GGet Form.Controls.Count, var
```

Form
Name of a form.
var
Variable that will contain the value of the property.

Values
Returns the number of controls on a form.

Remarks
Use the Count property to determine the number of controls on a form. Then you can iterate through all the controls using the Controls property.

See Also
Controls Property, **Type Property**

Example

```
Integer count

GSet frmMain.Controls.Count, count
For i = 1 To count
        GGet frmMain.Controls(i).Type, type$
```

```
        If type$ = "Button" Then
                GSet frmMain.Controls(i).Enabled, False
        EndIf
Next i
```

8.6 D

8.6.1 DblClick Event

Applies To

Label, ListBox, PictureBox, VideoBox, LED, ProgressBar, Grid

Description

Occurs when a control is double clicked.

Usage

```
Form_Control_DblClick (Sender$ As String)
```

Sender$
: Name of a control that sent the event.

Remarks

The DblClick event is used to respond to a double click on a control by the user. You can use the Sender$ parameter to determine which control sent the event.

See Also

Label Control, **ListBox Control**, **PictureBox Control**, **VideoBox Control**, **LED Control**, **ProgressBar Control**

Example

```
Function frmMain_lstModels_DblClick(Sender$ As String)  Integer index
  GGet frmMain.lstModels.SelectedIndex, index
  GGet frmMain.lstModels.List(index), g_CurrModel$
Fend
```

8.6.2 DialogResult Property

Applies To

Form

Description

Gets or sets the dialog result for a form.

Usage

```
GGet Form.DialogResult, var
GSet  Form.DialogResult, value
```

Form
: Name of a form or string variable containing a form name.

var
: Integer variable that will contain the value of the property.

value
: Integer expression for the new value of the property.

Values

0 - None
1 - OK

2 - Cancel
Default: Cancel

Remarks
DialogResult is used to indicated if the user accepted a form or canceled it.
Typically, a dialog has an OK button and a Cancel button. When the user clicks the OK button, the DialogResult property is set to 1 - OK in the button click event. If the Cancel button is clicked, the DialogResult property is set to 2 - Cancel.

See Also
Form

Example

```
Function frmSetSpeed_btnOK_Click(Sender$ As String)
  GSet frmSetSpeed.DialogResult, DIALOGRESULT_OK
  GClose frmSetSpeed
Fend
Function frmMain_btnSetSpeed_Click(Sender$ As String)
  Integer result
  String speed$
  result = GShowDialog(frmSetSpeed)
  If result = DIALOGRESULT_OK Then
     GGet frmSetSpeed.txtSpeed.Text, speed$
    g_RobotSpeed = Val(speed$)
Fend
```

8.6.3 DropDownStyle Property

Applies To
ComboBox

Description
Gets or sets a value specifying the style of the combo box.

Usage

```
GGet Form.Control.DropDownStyle, var
GSet  Form.Control.DropDownStyle, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: Integer variable that will contain the value of the property.

value
: Integer expression for the new value of the property.

Values
0 - Simple
1 - DropDown
2 - DropDownList
Default: DropDown

Remarks
The DropDownStyle property controls the interface that is presented to the user.

- You can enter a value that provides a simple drop-down list box, where the list always displays,

- a drop-down list box, where the text portion is not editable and you must select an arrow to view the drop-down,
- or the default drop-down list box, where the text portion is editable and the user must press the arrow key to view the list.

To always display a list that the user cannot edit, use a ListBox control.

See Also

ComboBox Control

Example

```
GSet frmMain.cmbPartNames.DropDownStyle, DROPDOWNSTYLE_SIMPLE
```

8.7 E

8.7.1 Enabled Property

Applies To

Button, Label, TextBox, RadioButton, CheckBox, ListBox, ComboBox, PictureBox, GroupBox, Timer, VideoBox, LED, ProgressBar, TrackBar, StatusBar, Grid , TabControl, Wizard, TreeView

Description

Gets or sets whether a control is enabled or not at runtime.

Usage

```
GGet Form.Control.Enabled, var
GSet  Form.Control.Enabled, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: Boolean variable that will contain the value of the property.

value
: Boolean expression for the new value of the property.

Values

False
True
Default: True (for others than Timer) False (for Timer)

Remarks

In certain situations, you need to prevent the operator from clicking a control, such as a button. For example, when SPEL+ tasks are not running, you can dim a Pause button by setting Enabled to False.

See Also

Button Control, **Label Control**, **TextBox Control**, **RadioButton Control**, **CheckBox Control**, **ListBox Control**, **ComboBox Control**, **PictureBox Control**, **GroupBox Control**, **Timer Control**, **VideoBox Control**, **LED Control**, **ProgressBar Control**, **TrackBar Control**, **StatusBar Control**, **Tab Control**, **Wizard Control**, **Tree View Control**

Example

```
GSet frmMain.btnPause.Enabled, False
```

8.7.2 EventTaskType Property

Applies To

Form, Button, Label, TextBox, RadioButton, CheckBox, ListBox, ComboBox, PictureBox, GroupBox, VideoBox, LED, ProgressBar, TrackBar, StatusBar, Grid , TabControl, Wizard, TreeView

Description

Gets the task type of the function that starts with the event.

Usage

```
GGet Form.[Control].EventTaskType, var
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: Integer variable that will contain the value of the property.

Values

0 - Normal
1 - NoPause
2 - NoEmgAbort
Default: Normal

Remarks

When this property is set to 1 - NoPause, the event handler function is able to be run when other tasks are paused. When this property is set to 2 - NoEmgAbort, the event handler function is able to be run in the emergency stop state.

See Also

Form, **Button Control**, **Label Control**, **TextBox Control**, **RadioButton Control**, **CheckBox Control**, **ListBox Control**, **ComboBox Control**, **PictureBox Control**, **GroupBox Control**, **VideoBox Control**, **LED Control**, **ProgressBar Control**, **TrackBar Control**, **StatusBar Control**, **Tab Control**, **Wizard Control**, **Tree View Control**

Example

```
Integer Ttype
GGet frmMain.btnPause.EventTaskType, Ttype
```

8.8 F

8.8.1 Font Property

From GUI Only

Applies To

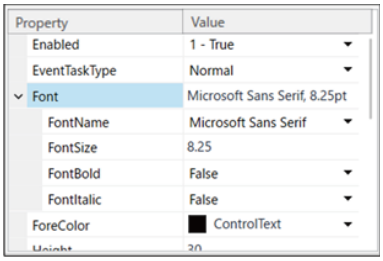
Button, Label, TextBox, RadioButton, CheckBox, ListBox, ComboBox, GroupBox, LED, StatusBar, TreeView

Description

The Font property allows you to change font parameters for a control at design time. (Font name, style, size)

Usage

The font of the display can be changed by setting the Font property value shown in the property grid of GUI Builder.



After the settings have been changed, and the project saved, the settings will be saved.

See Also

Button Control, **Label Control**, **TextBox Control**, **RadioButton Control**, **CheckBox Control**, **ListBox Control**, **ComboBox Control**, **GroupBox Control**, **LED Control**, **StatusBar Control**, **Tree View Control**

8.8.2 FontBold Property

Applies To

Button, Label, TextBox, RadioButton, CheckBox, ListBox, ComboBox, GroupBox, LED, StatusBar, TreeView

Description

Gets or sets if the font for the text used on a control is bold at runtime.

Usage

```
GGet Form.Control.FontBold, var
GSet  Form.Control.FontBold, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: Boolean variable that will contain the value of the property.

value
: Boolean expression for the new value of the property.

Values

False

True

Default: False

See Also

Button Control, **Label Control**, **TextBox Control**, **RadioButton Control**, **CheckBox Control**, **ListBox Control**, **ComboBox Control**, **GroupBox Control**, **LED Control**, **StatusBar Control**, **Tree View Control**, **FontName Property**, **FontItalic Property**, **FontSize Property**

Example

```
GSet frmMain.btnOK.FontBold, True
```

8.8.3 FontItalic Property

Applies To

Button, Label, TextBox, RadioButton, CheckBox, ListBox, ComboBox, GroupBox, LED, StatusBar, TreeView

Description

Gets or sets if the font for a control is italic at runtime.

Usage

```
GGet Form.Control.FontItalic, var
GSet  Form.Control.FontItalic, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: Boolean variable that will contain the value of the property.

value
: Boolean expression for the new value of the property.

Values

False
True
Default: False

See Also

Button Control, **Label Control**, **TextBox Control**, **RadioButton Control**, **CheckBox Control**, **ListBox Control**, **ComboBox Control**, **GroupBox Control**, **LED Control**, **StatusBar Control**, **Tree View Control**, **FontName Property**, **FontBold Property**, **FontSize Property**

Example

```
GSet frmMain.btnOK.FontItalic, True
```

8.8.4 FontName Property

Applies To

Button, Label, TextBox, RadioButton, CheckBox, ListBox, ComboBox, GroupBox, LED, StatusBar, TreeView

Description

Gets or sets the font name of the text used on a control.

Usage

```
GGet Form.Control.FontName, var
GSet  Form.Control.FontName, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: String variable that will contain the value of the property.

value
: String expression for the new value of the property.

Values

A string containing the name of the font.
Default: Microsoft Sans Serif

See Also

Button Control, **Label Control**, **TextBox Control**, **RadioButton Control**, **CheckBox Control**, **ListBox Control**, **ComboBox Control**, **GroupBox Control**, **LED Control**, **StatusBar Control**, **Tree View Control**, **FontSize Property**, **FontItalic Property**, **FontBold Property**

Example

```
GSet frmMain.txtStatus.FontName, "Courier New"
```

8.8.5 FontSize Property

Applies To

Button, Label, TextBox, RadioButton, CheckBox, ListBox, ComboBox, GroupBox, LED, StatusBar, TreeView

Description

Gets or sets the font size of the text used on a control.

Usage

```
GGet Form.Control.FontSize, var
GSet  Form.Control.FontSize, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: Real variable that will contain the value of the property.

value
: Real expression for the new value of the property.

Values

The value of the font size in points.
Default: 8.25

See Also

Button Control, **Label Control**, **TextBox Control**, **RadioButton Control**, **CheckBox Control**, **ListBox Control**, **ComboBox Control**, **GroupBox Control**, **LED Control**, **StatusBar Control**, **Tree View Control**, **FontName Property**, **FontItalic Property**, **FontBold Property**

Example

```
GSet frmMain.btnOK.FontSize, 16
```

8.8.6 ForeColor Property

Applies To

Button, Label, TextBox, RadioButton, CheckBox, ListBox, ComboBox, GroupBox, LED, ProgressBar, StatusBar, TreeView

Description

Gets or sets the foreground color of a control.

Usage

```
GGet Form.Control.ForeColor, var
GSet  Form.Control.ForeColor, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: String variable that will contain the value of the property.

value
: String expression for the new value of the property.

Values

A string that represents the foreground color of a control.
Default: Control Text

See Also

Button Control, **Label Control**, **TextBox Control**, **RadioButton Control**, **CheckBox Control**, **ListBox Control**, **ComboBox Control**, **GroupBox Control**, **LED Control**, **ProgressBar Control**, **StatusBar Control**, **Tree View Control**

Example

```
GSet frmMain.btnOK.ForeColor, "Blue"
```

8.8.7 FormBorderStyle Property

Applies To

Form

Description

Gets or sets the border style of the form.

Usage

```
GGet Form.FormBorderStyle, var
GSet  Form.FormBorderStyle, value
```

Form
: Name of a form or string variable containing a form name.

var
: Integer variable that will contain the value of the property.

value
: Integer expression for the new value of the property.

Values
0 - None
1 - Fixed Single
2 - Fixed3D
3 - FixedDialog
4 - Sizable
Default: FixedDialog

See Also
Form

Example

```
GSet frmMain.FormBorderStyle, FORMBORDERSYTLE_NONE

GClose
```

8.8.8 FormTemplate Property

Applies To
Form

Description
Selects and sets the size of a form from the template.
Use this when you want to resize the form size for the display size of Teach Pendant TP4.

Usage
This property does not support settings from the SPEL+ program.
Set and specify the value in the property grid of the target form.

Values
0 - Standard
1 - TP4Vertical
2 - TP4Horizontal
Default: Standard

Remarks
If you set the value to 1 or 2, the following properties are fixed.

- FormBorderStyle
- Height
- Left
- Width
- Top
- StartPosition If you want to set them manually, set the value to 0.

See Also
Form, **FormBorderStyle Property**, **Height Property**, **Left Property**, **Width Property**

8.9 G

8.9.1 GClose Statement

Applies To
Form

Description
Close the form.

Usage

```
GClose  Form
```

Form
: Name of a form or string variable containing a form name. You can also supply a system form ID to close one of the following system windows:

WIN_IOMON
: Closes the I/O Monitor window.

WIN_TASKMGR
: Closes the Task Manager window.

WIN_FORCEMON
: Closes the Force Monitor window.

WIN_SIMULATOR
: Closes the Simulator window.

Remarks
Do not use GClose for a form which is not displayed.

See Also
GShow Statement, **GShowDialog Function**

Example

```
GClose frmSetup

GClose WIN_TASKMGR
```

8.9.2 GGet Statement

Description
GGet is used to get the values of properties for forms and controls in the current project.

Usage

```
GGet  Form .Property, var
GGet  Form .Control.Property, var
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name. The control must exist in the specified form.

Property
: Name of the property to set or return the value of.

var
: Variable that will contain the value of the property.

Remarks

GGet is used to retrieve the value of a property at runtime. If you enter strings over 255 characters into the property, the 1st to 255th characters are stored in the variable var.

See Also

GSet Statement

8.9.3 GraphicsEnabled Property

Applies To

VideoBox

Description

Gets or sets whether the VideoBox displays vision graphics or not.

Usage

```
GGet Form.Control.GraphicsEnabled, var
GSet  Form.Control.GraphicsEnabled, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: Boolean variable that will contain the value of the property.

value
: Boolean expression for the new value of the property.

Values

False

True

Default: False

See Also

VideoBox Control, **VideoEnabled Property**

Example

```
GSet frmMain.VideoBox1.GraphicsEnabled, True
```

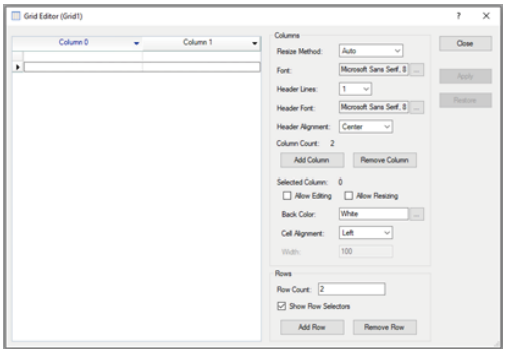
8.9.4 GridEditor Property

Applies To

Grid

Description

Opens a designer to set up the Grid control.



Usage

The GridEditor property can only be used at design time. Allows the user to set up the grid in order to meet their needs.

Sets/displays the number of rows in the grid. To manually change this value, type in the desired rows and click button.

Remarks

Use the designer to configure the grid and how it will first appear at runtime. You can add text to cells at designtime and/or add data at runtime.

Columns can be moved by dragging and dropping them to desired locations. They can also be swapped with other columns by using the drop down arrow in the column header.

See Also

Grid Control

8.9.5 GSet Statement

Description

GSet is used to set the values of properties in GUI Builder.

Usage

```
GSet  Form .Property, value
GSet  Form .Control.Property, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name. The control must exist in the specified form.

Property
: Name of the property to set the value of.

value
: The expression that holds the data to be set.

Remarks

GSet is used to set the value of a form or control property at runtime.

See Also

GGet Statement

8.9.6 GShow Statement

Applies To

Form

Description

GShow displays a form as a non-modal window.

Usage

```
GShow  Form
```

Form
: Name of a form or string variable containing a form name. You can also supply a system form ID to show one of the following system windows:

WIN_IOMON
: Opens the I/O Monitor window.

WIN_TASKMGR
: Opens the Task Manager window.

WIN_FORCEMON
: Opens the Force Monitor window.

WIN_SIMULATOR
: Opens the Simulator window.

See Also

GClose Statement, **GShowDialog Function**

Example

```
GShow frmIODiags

GShow WIN_TASKMGR
```

8.9.7 GShowDialog Function

Applies To

Form

Description

GShowDialog displays a form as a modal dialog box and returns the DialogResult value.

Usage

```
GShowDialog(Form)
```

Form
: Name of a form or string variable containing a form name.

Returns

The value of the form's DialogResult property is returned.

See Also

GShow Statement, **GShowDialog Statement**

Example

```
result = GShowDialog(frmSetup)
If result = DIALOGRESULT_OK Then
  Call SaveSettings
EndIf
```

8.9.8 GShowDialog Statement

Applies To

Form

Description

GShowDialog displays a form as a modal dialog box without returning the DialogResult value.

Usage

```
GShowDialog Form
```

Form
: Name of a form or string variable containing a form name.

See Also

GShow Statement, **GShowDialog Function**

Example

```
GShowDialog frmInfoDisplay
```

8.10 H

8.10.1 Height Property

Applies To

Form, Button, Label, TextBox, RadioButton, CheckBox, ListBox, ComboBox, PictureBox, GroupBox, VideoBox, LED, ProgressBar, TrackBar, StatusBar, Grid , TabControl, Wizard, TreeView

Description

Get or sets the height of a form or control in pixels.

Usage

```
GGet Form.[Control].Height, var
GSet  Form.[Control].Height, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: Integer variable that will contain the value of the property.

value
: Integer expression for the new value of the property.

Values

An integer representing the height in pixels.

See Also

Form, **Button Control**, **Label Control**, **TextBox Control**, **RadioButton Control**, **CheckBox Control**, **ListBox Control**, **ComboBox Control**, **PictureBox Control**, **GroupBox Control**, **VideoBox Control**, **LED Control**, **ProgressBar Control**, **TrackBar Control**, **StatusBar Control**, **Tab Control**, **Wizard Control**, **Tree View Control**, **Left Property**, **Top Property**, **Width Property**

Example

```
GSet frmMain.btnOK.Height, 48
```

8.10.2 HelpButton Property

Applies To

Form

Description

Gets or sets a value indicating whether a Help button should be displayed in the caption box of the form.

Usage

```
GGet Form.HelpButton, var
GSet  Form.HelpButton, value
```

Form
: Name of a form or string variable containing a form name.

var
: Boolean variable that will contain the value of the property.

value
Boolean expression for the new value of the property.

Values
False
True
Default: False

Remarks
When this property is set to True, a small button with a question mark appears in the caption bar to the left of the Close button (when MaximizeBox, MinimizeBox properties are set to “False”). You can use this button to display help for your application.

See Also
Form, **HelpID Property**

Example

```
GSet frmMain.HelpButton, True
```

8.10.3 HelpID Property

Applies To
Form

Description
Returns or sets an Integer containing the HelpID for a topic in a Help file.

Usage

```
GGet Form.HelpID, var
GSet  Form.HelpID, value
```

Form
Name of a form or string variable containing a form name.
var
Integer variable that will contain the value of the property.
value
Integer expression for the new value of the property.

Values
An integer value that is a topic ID in the help file. (0 ~ 999999)
Default:

Remarks
The HelpID property is used to display context-sensitive Help for an application. The help file is specified in the EPSON RC+ Project | Properties | GUI Builder page.
When HelpID is not 0 and the user types F1 or clicks the form help button (if the HelpButton property is True), then the help topic is displayed.

See Also
Form, **HelpButton Property**

Example

```
GSet frmMain.HelpID, 50
```

8.11 I

8.11.1 Icon Property

Applies To

Form

Description

Gets or sets the image displayed in the icon of the form’s title bar.

Usage

```
GGet Form.Icon, var
GSet  Form.Icon, value
```

Form
: Specifies form name or string variable containing a form name.

var
: String variable that will contain the value of the property.

value
: Specifies string expression for the new value of the property.

Values

A string containing the full path to an image file.
Default: Empty

See Also

Form

Example

```
GSet frmMain.Icon, "C:\EpsonRC80\GUI\Icons\LedGreen.ico"
```

8.11.2 Image Property

Applies To

Button, Label, RadioButton, CheckBox, PictureBox

Description

Gets or sets the image that is displayed on a control.

Usage

```
GGet Form.Control.Image, var
GSet  Form.Control.Image, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: String variable that will contain the value of the property.

value
: String expression for the new value of the property.

Values

A string containing the full path to an image file.
Default: Blank

See Also

Button Control, **Label Control**, **RadioButton Control**, **CheckBox Control**, **PictureBox Control**, **ImageAlign Property**, **ImageIndex Property**

Example

```
GSet frmMain.btnTools.Image, "c:\Images\Tools.bmp"
```

8.11.3 ImageAlign Property

Applies To

Button, Label, RadioButton, CheckBox, LED

Description

Gets or sets the image alignment for the image that is displayed on a control.

Usage

```
GGet Form.Control.ImageAlign, var
GSet  Form.Control.ImageAlign, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: Integer variable that will contain the value of the property.

value
: Integer expression for the new value of the property.

Values

1 - TopLeft
2 - TopCenter
3 - TopRight
4 - MiddleLeft
5 - MiddleCenter
6 - MiddleRight
7 - BottomLeft
8 - BottomCenter
9 - BottomRight

Default:

- 5 - MiddleCenter (Button, Label, RadioButton, CheckBox)
- 4 - MiddleLeft (LED)

See Also

Button Control, **Label Control**, **RadioButton Control**, **CheckBox Control**, **LED Control**, **Image Property**, **TextAlign Property**

Example

```
GSet frmMain.btnTools.ImageAlign, IMAGEALIGN_MIDDLECENTER
```

8.11.4 ImageOff Property

Applies To

LED

Description

Gets or sets the image that is displayed when LED control is off.

Usage

```
GGet Form.Control.ImageOff, var
GSet  Form.Control.ImageOff, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: String variable that will contain the value of the property.

value
: String expression for the new value of the property.

Values

A string containing the full path to an image file. The file extensions should be bmp, ico, jpeg, gif, and png.
Default: LedOff.ico

See Also

LED Control, **ImageOn Property**

Example

```
GSet frmMain.Led1.ImageOff, "c:\EpsonRC80\GUI\Icons\LedOff.ico"
```

8.11.5 ImageOn Property

Applies To

LED

Description

Gets or sets the image that is displayed when LED control is on.

Usage

```
GGet Form.Control.ImageOn, var
GSet  Form.Control.ImageOn, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: String variable that will contain the value of the property.

value
: String expression for the new value of the property.

Values

A string containing the full path to an image file. The file extensions should be bmp, ico, jpeg, gif, and png.
Default: LedRed.ico

See Also

LED Control, **ImageOff Property**

Example

```
GSet frmMain.Led1.ImageOn, "c:\EpsonRC80\GUI\Icons\LedRed.ico"
```

8.11.6 ImageIndex Property

Applies To

PictureBox

Description

Sets and gets an index of the image to be displayed on a control.

Usage

```
GGet Form.Control.ImageIndex, var
GSet Form.Control.ImageIndex, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: Integer variable that will contain the value of the property.

value
: Integer expression for the new value of the property

This property can be used with GGet and GSet only when RC800 Controller is connected.

Values
The index of the image file that is on the control. Default: -1

See Also
PictureBox Control, **Image Property**

Example

```
'  Gets the index of the image file that is currently set.
GGet frmMain.pictureBox1.ImageIndex, index
'  Sets the image file index to 2.
GSet frmMain.pictureBox1.ImageIndex, 2
```

8.11.7 Interval Property

Applies To
Timer

Description
Gets or sets the time, in milliseconds, between timer ticks.

Usage

```
GGet Form.Control.Interval, var
GSet  Form.Control.Interval, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: Integer variable that will contain the value of the property.

value
: Integer expression for the new value of the property.

Values
The number of milliseconds between each timer tick. Enter integer 1 to 9999.
Default: 100

Remarks
To get the number of seconds in the interval, divide this number by 1,000.

See Also
Timer Control, **Enabled Property**

Example

```
GSet frmMain.tmrMain.Interval, 500
```

8.11.8 IOBit Property

Applies To
LED

Description

Gets or sets the number of the I/Obit that indicates it to the LED Control.

Usage

```
GGet Form.Control.IOBit, var
GSet  Form.Control.IOBit, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: Integer variable that will contain the value of the property.

value
: Integer expression for the new value of the property.

Values

The integer value from 0 to 9999
Default:

See Also

LED Control, **IOType Property**

Example

```
GSet frmMain.Led1.IOBit, 10
```

8.11.9 IOType Property

Applies To

LED

Description

Gets or sets the type of the I/O that indicates it to the LED Control.

Usage

```
GGet Form.Control.IOType, var
GSet  Form.Control.IOType, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: Integer variable that will contain the value of the property.

value
: Integer expression for the new value of the property.

Values

0 - Input
1 - Output
2 - Memory
Default: Input

See Also

LED Control, **IOBit Property**

Example

```
GSet frmMain.Led1.IOType, IOTYPE_OUTPUT
```

8.12 K

8.12.1 KeyPress Event

Applies To

TextBox

Description

Occurs when a key is pressed.

Usage

```
Form_Control_KeyPress (Sender$ As String, ByRef Key$ As String)
```

Sender$
: Name of a control that sent the event.

Key$
: The key that was pressed by the operator.

Remarks

The Keypress event can be used to detect the keys that are input by the operator.

This event can be used as a filter. You can change the key that was pressed by setting Key$ to another character. You can cancel a keypress by setting Key$ to an empty string.

See Also

TextBox Control

Example

```
Function frmMain_txtSpeed_KeyPress(Sender$ As String, ByRef Key$ As String)
  ' Allow only digits to be entered
  If Instr("0123456789" Key$) < 0 Then
    Key$ = ""
  EndIf
Fend
```

8.13 L

8.13.1 LargeChange Property

Applies To
TrackBar

Description
Gets or sets the number of positions that the slider moves when clicking the mouse or pressing Page Up and Page Down keys.

Usage

```
GGet Form.Control.LargeChange, var
GSet  Form.Control.LargeChange, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: Integer variable that will contain the value of the property.

value
: Integer expression for the new value of the property.

Values
An integer representing the number of positions that the slider moves.

See Also
TrackBar Control, **SmallChange Property**

Example

```
GSet frmMain.TrackBar1.LargeChange, 10
```

8.13.2 Left Property

Applies To
Form, Button, Label, TextBox, RadioButton, CheckBox, ListBox, ComboBox, PictureBox, GroupBox, VideoBox, LED, ProgressBar, TrackBar, StatusBar, Grid , TabControl, Wizard, TreeView

Description
Get or sets the left coordinate of a form or control in pixels.

Usage

```
GGet Form.[Control].Left, var
GSet  Form.[Control].Left, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: Integer variable that will contain the value of the property.

value

Integer expression for the new value of the property.

Values
An integer representing the left coordinate in pixels.

See Also
Form, **Button Control**, **Label Control**, **TextBox Control**, **RadioButton Control**, **CheckBox Control**, **ListBox Control**, **ComboBox Control**, **PictureBox Control**, **GroupBox Control**, **VideoBox Control**, **LED Control**, **ProgressBar Control**, **TrackBar Control**, **StatusBar Control**, **Tab Control**, **Wizard Control**, **Tree View Control**, **Top Property**, **Height Property**, **Width Property**

Example
```
GSet frmMain.btnOK.Left, 200
```

8.13.3 List Property

Applies To
ListBox, ComboBox

Description
Get or sets the string value of the specified element in the control item list.

Usage
```
GGet Form.Control.List(index), var
GSet Form.Control.List(index), value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name. The control must exist in the specified form.

index
: Integer expression for the list array index staring with 0.

var
: string variable that will contain the value of the property.

value
: String expression for the new value of the property.

Values
String containing the specified list item.

Remarks
After using AddItem to add items to the ListBox or ComboBox list, you can use the List property to get or edit the value of any of the items.
Setting a blank string, you can delete an item.

See Also
ComboBox Control, **ListBox Control**, **ListCount Property**, **AddItem Property**

Example
```
String part$GGet frmMain.lstPartNames.List(0), part$
```

8.13.4 ListCount Property

Applies To
ListBox, ComboBox

Description
Gets or sets the count of the number of items in a Listbox or Combobox.

Usage

```
GGet Form.Control.ListCount, var
GSet  Form.Control.ListCount, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name. The control must exist in the specified form.

var
: Integer variable that will contain the value of the property.

value
: Integer expression for the new value of the property.

Values
The number of items in the list.

Remarks
To clear the items in a ListBox or ComboBox, you can set ListCount to 0. If ListCount is set to a value that is higher than the number of items in the list, then empty text is added for the extra items.

See Also
ComboBox Control, **ListBox Control**, **AddItem Property**, **List Property**

Example

```
Integer count
' Retrieve the number of items in a list
GGet frmMain.lstPartNames.ListCount, count

' Clear the items in a list
GSet frmMain.lstPartName.ListCount, 0
```

8.13.5 Load Event

Applies To
Form

Description
Occurs before a form is displayed for the first time.

Usage

```
Form_Load (Sender$ As String)
```

Sender$
: Name of the form that sent the event.

Remarks
You can use this event to perform tasks before a form is shown to the user.

See Also

Form, **Resize Event**, **Closed Event**

Example

```
Function frmMain_Load(Sender$ As String)
  GSet frmMain.txtSpeed.Text, Str$(g_RobotSpeed)
Fend
```

8.14 M

8.14.1 MaximizeBox Property

Applies To
Form

Description
Gets or sets a value indicating whether the maximize button is displayed in the title bar of the form.

Usage

```
GGet Form.MaximizeBox, var
GSet  Form.MaximizeBox, value
```

Form
: Name of a form or string variable containing a form name.

var
: Boolean variable that will contain the value of the property.

value
: Boolean expression for the new value of the property.

Values
False
True
Default: False

Remarks
A maximize button enables users to maximize a form to full screen.

See Also
Form, **MinimizeBox Property**

Example

```
GSet frmMain.MaximizeBox, True
```

8.14.2 Maximum Property

Applies To
ProgressBar, TrackBar

Description
Get or sets the maximum value of the processing range of the control.

Usage

```
GGet Form.Control.Maximum, var
GSet  Form.Control.Maximum, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name. The control must exist in the specified form.

var

Integer variable that will contain the value of the property.

value

Integer expression for the new value of the property.

Values

An integer representing the maximum value of the processing range.

See Also

ProgressBar Control, **TrackBar Control**, **Minimum Property**, **Value Property**

Example

```
GSet frmMain.TrackBar1.Maximum, 60
```

8.14.3 MinimizeBox Property

Applies To

Form

Description

Gets or sets a value indicating whether the minimize button is displayed in the title bar of the form.

Usage

```
GGet Form.MinimizeBox, var
GSet  Form.MinimizeBox, value
```

Form

Name of a form or string variable containing a form name.

var

Boolean variable that will contain the value of the property.

value

Boolean expression for the new value of the property.

Values

False

True

Default: False

Remarks

A minimize button enables users to minimize a form to an icon.

See Also

Form, **MaximizeBox Property**

Example

```
GSet frmMain.MinimizeBox, True
```

8.14.4 Minimum Property

Applies To

ProgressBar, TrackBar

Description
Get or sets the minimum value of the processing range of the control.

Usage
```
GGet Form.Control.Minimum, var
GSet  Form.Control.Minimum, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name. The control must exist in the specified form.

var
: Integer variable that will contain the value of the property.

value
: Integer expression for the new value of the property.

Values
An integer representing the minimum value of the processing range.

See Also
ProgressBar Control, **TrackBar Control**, **Maximum Property**, **Value Property**

Example
```
GSet frmMain.TrackBar1.Minimum, 10
```

8.14.5 MultiLine Property

Applies To
TextBox

Description
Gets or sets a value indicating whether this is a multiline TextBox control.

Usage
```
GGet Form.Control.MultiLine, var
GSet  Form.Control.MultiLine, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name. The control must exist in the specified form.

var
: Boolean variable that will contain the value of the property.

value
: Boolean expression for the new value of the property.

Values
False

True

Default: False

Remarks
If you want to display multiple lines in a TextBox control, you must set MultiLine to True.

See Also

TextBox Control, **ScrollBars Property**

Example

```
GSet frmMain.txtStatus.MultiLine, True
```

8.15 N

8.15.1 Name Property

Applies To

Form, Button, Label, TextBox, RadioButton, CheckBox, ListBox, ComboBox, PictureBox, GroupBox, Timer, VideoBox, LED, ProgressBar, TrackBar, StatusBar, Grid , TabControl, Wizard, TreeView

Description

Gets the name of a form or control at runtime. Used to set the Name of a form or control at design time.

Usage

```
GGet Form.[Control].Name, var
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: String variable that will contain the value of the property.

Values

A string containing the name of the form or control. You cannot name as follows:

- A number cannot be used at the beginning of the form name or control name.
- You cannot use the same form name as the project name you are editing.

See Also

Form, **Button Control**, **Label Control**, **TextBox Control**, **RadioButton Control**, **CheckBox Control**, **ListBox Control**, **ComboBox Control**, **PictureBox Control**, **GroupBox Control**, **Timer Control**, **VideoBox Control**, **LED Control**, **ProgressBar Control**, **TrackBar Control**, **StatusBar Control**, **Tab Control**, **Wizard Control**, **Tree View Control**

Example

```
String name$
GGet frmMain.btnOK.Name, name$
```

8.16 O

8.16.1 Orientation Property

Applies To

ProgressBar, TrackBar

Description

Get or sets the orientation of the control.

Usage

```
GGet Form.Control.Orientation, var
GSet  Form.Control.Orientation, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: Integer variable that will contain the value of the property.

value
: Integer expression for the new value of the property.

Values

0 - Horizontal

1 - Vertical

Default: Horizontal

See Also

ProgressBar Control, **TrackBar Control**

Example

```
GSet frmMain.ProgressBar1.Orientation, ORIENT_VERTICAL
```

8.17 P

8.17.1 Parent Property

Applies To

TreeView

Description

Gets and sets the parents node for the items (node) that has been added to the tree view.

Usage

```
GGet Form.Control.Parent(name), var
GSet Form.Control.Parent(name), value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name. The control must exist in the specified form.

name
: Specifies the string that indicates the child node in which the parent node will be set.

var
: String variable that will contain the value of the property.

value
: String expression for the new value of the property

Values

Specify the string that indicates the parent node.

See Also

Tree View Control, **AddNode Property**, **RemoveNode Property**, **Text Property**

Example

```
GSet frmMain.lstModels.AddNode, "Model1"
GSet frmMain.lstModels.AddNode, "Model1-1"
GSet frmMain.lstModels.Parent("Model1-1"), "Model1"
```

8.17.2 PasswordChar Property

Applies To
TextBox

Description
Gets or sets the character used to mask characters of a password in a single-line TextBox control.

Usage

```
GGet Form.Control.PasswordChar, var
GSet  Form.Control.PasswordChar, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name. The control must exist in the specified form.

var
: String variable that will contain the value of the property.

value
: String expression for the new value of the property.

Values
A string containing the character used for the mask.

Remarks
Use PasswordChar to hide the characters of a password. When the user types a character, the PasswordChar is displayed. Note that the Multiline property for the TextBox control must be set to False.

See Also
TextBox Control

Example

```
GSet frmMain.txtPassword.PasswordChar, "*"
```

8.17.3 PressDelay Property

Applies To
Button

Description
Sets the time until the event is executed after the button control is clicked.

Usage
This property does not support settings from the SPEL+ program.
Specify and set the value in the property grid of the target button control.

Values
Specify the time from clicking to starting event in msec (millisecond). Enter integer 0 to 9999.
Default: Emmediately Execute

Remarks
If you release the click button before the set time elapses, the event will not execute.

See Also
Button Control

8.17.4 PressSound Property

Applies To
Button

Description
Sets whether a beep sounds when the control is clicked.

Usage
This property does not support settings from the SPEL+ program.
Set from the property grid of the target button control.

Values
False
True
Default: False

See Also
Button Control

8.17.5 ProgressBarStyle Property

Applies To
ProgressBar

Description
Get or sets the style of the ProgressBar.

Usage

```
GGet Form.Control.ProgressBarStyle, var
GSet  Form.Control.ProgressBarStyle, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: Integer variable that will contain the value of the property.

value
: Integer expression for the new value of the property.

Values
1 - Continuous
2 - Marquee
Default: Continuous

See Also
ProgressBar Control

Example

```
GSet frmMain.ProgressBar1.ProgressBarStyle, PROGRESSBAR_STYLE_MARQUEE
```

8.18 R

8.18.1 ReadOnly Property

Applies To
TextBox

Description
Gets or sets which the textbox control is ReadOnly or not.

Usage

```
GGet Form.Control.ReadOnly, var
GSet  Form.Control.ReadOnly, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name. The control must exist in the specified form.

var
: Boolean variable that will contain the value of the property.

value
: Boolean expression for the new value of the property.

Values
False

True

Default: False

Remarks
When this property is set to True, Operator is not able to edit textbox.

See Also
TextBox Control

Example

```
GSet frmMain.txtSpeed.ReadOnly, True
```

8.18.2 RemoveNode Property

Applies To
TreeView

Description
Deletes an item from an item (node) in the Tree View. The child node of the specified item will also be deleted.

Usage

```
GSet Form.Control.RemoveNode, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name. The control must exist in the specified form.

value
Specifies the string that indicates the item (node) to be deleted.

Values

Specifies the string that indicates the item (node) to delete.

See Also

Tree View Control, **AddNode Property** ,**Parent Property**, **Text Property**

Example

```
GSet frmMain.lstModels.RemoveNode, "Model1"
```

8.18.3 RemoveRow Property

Applies To
Grid

Description
Used to remove a row from a Grid control during runtime.

Usage

```
GSet Form.Control.RemoveRow, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name. The control must exist in the specified form.

value
: Integer representing the index of the row being removed.

Values
An integer value containing the index of the removed row.

See Also
AddRow Property, **Grid Control**, **RowCount Property**

Example

```
GSet frmMain.Grid01.RemoveRow, 0
```

8.18.4 Resize Event

Applies To
Form

Description
Occurs when the form is resized.

Usage

```
Form_Resize (Sender$ As String)
```

Sender$
: Name of a form that sent the event.

See Also
Form, **Load Event**, **Closed Event**

Example

```
Function frmMain_Resize(Sender$ As String)
  Integer width
  GGet frmMain.Width, width
  GSet frmMain.btnOK.Left, width / 2 - 100
```

```
  GSet frmMain.btnCancel.Left, width / 2 + 10
Fend
```

8.18.5 RobotNumber Property

Applies To

StatusBar

Description

Get or sets the robot number to use when ShowRobot is True.

Usage

```
GGet Form.Control.RobotNumber, var
GSet  Form.Control.RobotNumber, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name. The control must exist in the specified form.

var
: Integer variable that will contain the value of the property.

value
: Integer expression for the new value of the property.

Values

An integer that specifies the robot number used to display the robot information.

Default: 1

See Also

StatusBar Control, **ShowRobot Property**

Example

```
GSet frmMain.StatusBar1.RobotNumber, 2
```

8.18.6 RowCount Property

Applies To

Grid

Description

Used to set or get the number of rows in a Grid control.

If set to a value greater than the current number of lines, a new line will be added to the end.

If set to a value less than the current number of lines, a new line will be removed from the end.

Usage

```
GGet Form.Control.RowCount, var
GSet Form.Control.RowCount, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name. The control must exist in the specified form.

var

Integer variable that will contain the number of rows in the control.

value

Integer representing the total number of rows in the control.

Values

Integer value specifying the number of rows in the Grid.

See Also

AddRow Property, **Grid Control**, **RemoveRow Property**, **RowCount Property**

Example

```
Integer count
GSet frmMain.Grid01.RowCount, 50
GGet frmMain.Grid01.RowCount, count
```

8.19 S

8.19.1 Scroll Event

Applies To
TrackBar

Description
Occurs when the track bar slider has been moved.

Usage

```
Form_Control_Scroll (Sender$ As String)
```

Sender$
Name of a control that sent the event.

See Also
TrackBar Control

Example

```
Function frmMain_TrackBar1_Scroll(Sender$ As String, Value As Integer)
    Print "The trackbar value after scroll is", Value
Fend
```

8.19.2 ScrollBars Property

Applies To
TextBox, Grid

Description
Gets or sets which scroll bars should appear in a multiline TextBox control.

Usage

```
GGet Form.Control.ScrollBars, var
GSet  Form.Control.ScrollBars, value
```

Form
Name of a form or string variable containing a form name.
Control
Name of a control or string variable containing a control name. The control must exist in the specified form.
var
Integer variable that will contain the value of the property.
value
Integer expression for the new value of the property.

Values
0 - None
1 - Horizontal
2 - Vertical
3 - Both
Default: None

Remarks

To use ScrollBars, the Multiline property for the TextBox control must be set to True.
To display the horizontal scroll bar, set WordWrap property to False.

See Also

TextBox Control, **MultiLine Property**, **WordWrap Property**

Example

```
GSet frmMain.txtStatus.Scrollbars, SCROLLBARS_VERT
```

8.19.3 SelectedNodeChanged Event

Applies To

TreeView

Description

Occurs when the selected node has been changed.

Usage

```
Form_Control_SelectedNodeChanged(Sender$ As String, NodeName$ As String)
```

Sender$
: Name of a control that sent the event.

NodeName$
: Name of the selected node.

See Also

Tree View Control

Example

```
Function frmMain_TreeView1_SelectedNodeChanged(Sender$ As String, NodeName$ As
String)
    Print NodeName$, " was selected."

    'Switch the page displayed in the wizard control based on the selected node
    Integer wizardIndex
    Select NodeName$
        Case "TreeNode1"; wizardIndex = 0
        Case "TreeNode2"; wizardIndex = 1
        Case "TreeNode3"; wizardIndex = 2
    Send

    GSet frmMain.Wizard1.SelectedIndex, wizardIndex
Fend
```

8.19.4 SelectedIndex Property

Applies To

ListBox, ComboBox, Grid, TabControl, Wizard

Description

Gets or sets the SelectedIndex property returns an index of the currently selected item

Usage

```
GGet Form.Control.SelectedIndex, var
GSet  Form.Control.SelectedIndex, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name. The control must exist in the specified form.

var
: Integer variable that will contain the value of the property.

value
: Integer expression for the new value of the property.

Values

Returns the index of the selected item, starting with 0. -1 is returned if no items are selected.

Remarks

Use SelectedIndex to determine which item the user has selected in a ListBox or ComboBox. Typically, you would do this in the Click event of the control.

See Also

AddItem Property, **ComboBox Control**, **List Property**, **ListBox Control**, **Grid Control**, **Wizard Control**

Example

```
Integer index
GGet frmMain.lstParts.SelectedIndex, index
```

8.19.5 SelectionChanged Event

Applies To

TabControl, Wizard

Description

Occurs when the selected page is changed.

Usage

```
Form_Control_SelectionChanged(Sender$ As String, SelectedIndex As Integer,
PreviousIndex As Integer)
```

Sender$
: Name of a control that sent the event.

SelectedIndex
: The index number of the selected page.

PreviousIndex
: The index number of the page that was selected before.

Values

Returns the number of the selected item with an integer, starting with 0. -1 is returned if no items are selected.

Remarks

Use SelectedIndex property to determine which item the user has selected in a ListBox or ComboBox. Typically, you would do this in the Click event of the control.

See Also

Tab Control, **Wizard Control**

Example

```
Function frmMain_TabControl1_SelectionChanged(Sender$ As String, SelectedIndex As
Integer, PreviousIndex As Integer)
        Print "The TabControl selection was changed to ", Str$(SelectedIndex), "
from ", Str$(PreviousIndex)
Fend
```

8.19.6 ShowDateTime Property

Applies To

StatusBar

Description

Gets or sets whether to display the current date and time on a StatusBar control.

Usage

```
GGet Form.Control.ShowDateTime, var
GSet  Form.Control.ShowDateTime, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name. The control must exist in the specified form.

var
: Boolean variable that will contain the value of the property.

value
: Boolean expression for the new value of the property.

Values

False

True

Default: False

See Also

StatusBar Control, **ShowEStop Property**, **ShowRobot Property**, **ShowSafeguard Property**

Example

```
GSet frmMain.StatusBar1.ShowDateTime, True
```

8.19.7 ShowEStop Property

Applies To

StatusBar

Description

Gets or sets whether to display EStop status on a StatusBar control.

Usage

```
GGet Form.Control.ShowEStop, var
GSet  Form.Control.ShowEStop, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name. The control must exist in the specified form.

var
: Boolean variable that will contain the value of the property.

value
: Boolean expression for the new value of the property.

Values
False
True
Default: False

See Also
StatusBar Control, ShowDateTime Property, ShowRobot Property, ShowSafeguard Property

Example

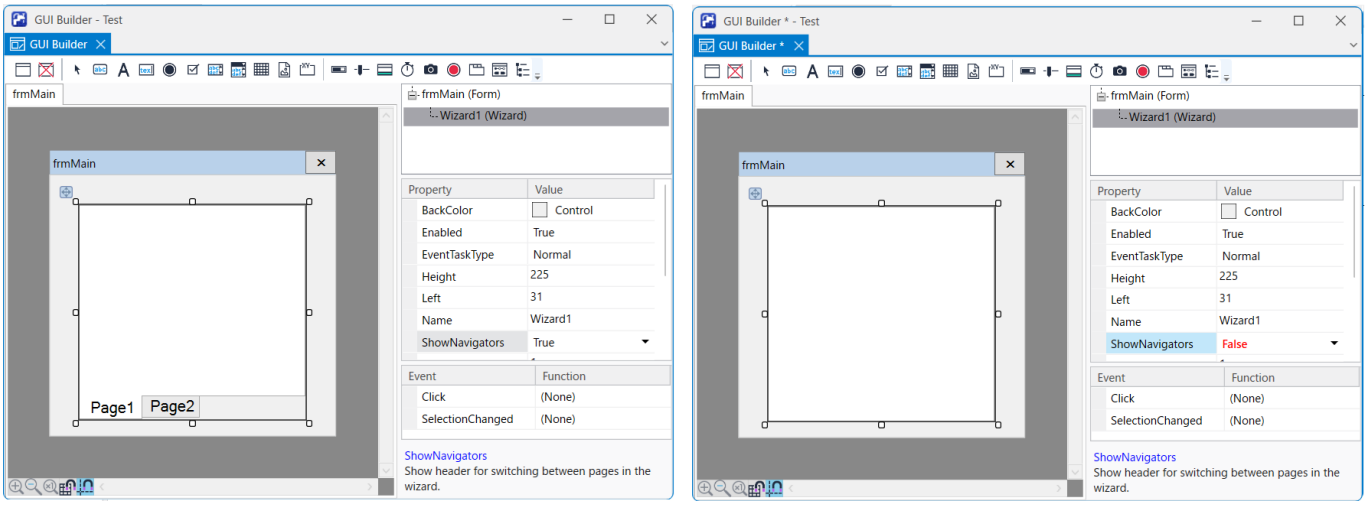
```
GSet frmMain.StatusBar1.ShowEStop, True
```

8.19.8 ShowNavigators Property

Applies To
Wizard

Description
Sets whether to display or hide the header which allows you to switch pages of the wizard.

Usage
If the ShowNavigators property is set to True, the header will be displayed. If it is set to False, the header will be hidden.



Remarks This property is only reflected when displayed in the design area.
During runtime, the header will not be displayed regardless of the property's setting.

Values
False
True
Default: True

See Also
Wizard Control

8.19.9 ShowPrint Property

Applies To
TextBox

Description
Gets or sets whether to display output from Print statements or not.

Usage

```
GGet Form.Control.ShowPrint, var
GSet  Form.Control.ShowPrint, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name. The control must exist in the specified form.

var
: Boolean variable that will contain the value of the property.

value
: Boolean expression for the new value of the property.

Values
False
True
Default: False

Remarks
When ShowPrint is True, then all output from Print statements executed in Spel tasks will be displayed in the text box. If multiple text boxes have ShowPrint set to True, then all of the text boxes will show the print output.

See Also
TextBox Control, **ScrollBars Property**

Example

```
GSet frmMain.txtStatus.ShowPrint, True
```

8.19.10 ShowRobot Property

Applies To
StatusBar

Description
Get or sets whether to display the robot name and model that is specified by the RobotNumber property on a StatusBar control.

Usage

```
GGet Form.Control.ShowRobot, var
GSet  Form.Control.ShowRobot, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name. The control must exist in the specified form.

var
Boolean variable that will contain the value of the property.
value
Boolean expression for the new value of the property.

Values
False
True
Default: False

See Also
StatusBar Control, **RobotNumber Property**, **ShowDateTime Property**, **ShowEStop Property**, **ShowSafeguard Property**

Example

```
GSet frmMain.StatusBar1.ShowRobot, True
```

8.19.11 ShowSafeguard Property

Applies To
StatusBar

Description
Get or sets whether to display the Safeguard status on a StatusBar control.

Usage

```
GGet Form.Control.ShowSafeguard, var
GSet  Form.Control.ShowSafeguard, value
```

Form
Name of a form or string variable containing a form name.
Control
Name of a control or string variable containing a control name. The control must exist in the specified form.
var
Boolean variable that will contain the value of the property.
value
Boolean expression for the new value of the property.

Values
False
True
Default: False

See Also
StatusBar Control, **ShowDateTime Property**, **ShowEStop Property**, **ShowRobot Property**

Example

```
GSet frmMain.StatusBar1.ShowSafeguard, True
```

8.19.12 SizeMode Property

Applies To
PictureBox

Description
Indicates how the image is displayed.

Usage
```
GGet Form.Control.SizeMode, var
GSet  Form.Control.SizeMode, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name. The control must exist in the specified form.

var
: Integer variable that will contain the value of the property.

value
: Integer expression for the new value of the property.

Values
0 - Normal
1 - StretchImage
2 - AutoSize
3 - CenterImage
4 - Zoom
Default: Normal

Remarks
By default, in Normal mode, the image is placed in the upper left corner of the PictureBox, and any part of the image too big for the PictureBox is clipped.
Using the StretchImage value causes the image to stretch to fit the PictureBox.
Using the AutoSize value causes the control to resize to always fit the image.
Using the CenterImage value causes the image to be centered in the PictureBox.
Using the Zoom value causes the image to stretch to fit the PictureBox keeping the same aspect ratio.

See Also
PictureBox Control, **Image Property**, **ImageAlign Property**

Example
```
GSet frmMain.picLogo.SizeMode, SIZEMODE_AUTOSIZE
```

8.19.13 SmallChange Property

Applies To
TrackBar

Description
Gets or sets the number of positions that the slider moves for keyboard inputs (arrow orientation keys).

Usage
```
GGet Form.Control.SmallChange, var
GSet  Form.Control.SmallChange, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: Integer variable that will contain the value of the property.

value
: Integer expression for the new value of the property.

Values

An integer representing the number of positions that the slider moves.

See Also

TrackBar Control, **LargeChange Property**

Example

```
GSet frmMain.TrackBar1.SmallChange, 5
```

8.19.14 Sorted Property

Applies To

ComboBox, ListBox

Description

Gets or sets a value indicating whether the items in the ComboBox or ListBox are sorted alphabetically.
Sorting order: Numeric character (regardless of whether one-byte or two byte), alphabetical character, Katakana, Hiragana, Kanji character

Usage

```
GGet Form.Control.Sorted, var
GSet  Form.Control.Sorted, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name. The control must exist in the specified form.

var
: Boolean variable that will contain the value of the property.

value
: Boolean expression for the new value of the property.

Values

False
True
Default: False

See Also

ComboBox Control, **ListBox Control**

Example

```
GSet frmMain.lstParts.Sorted, True
```

8.19.15 StartPosition Property

Applies To

Form

Description

Gets or sets the starting position of the form at run time.

Usage

```
GGet Form.StartPosition, var
GSet  Form.StartPosition, value
```

Form
: Name of a form or string variable containing a form name.

var
: Integer variable that will contain the value of the property.

value
: Integer expression for the new value of the property.

Values

0 - Manual

1 - CenterScreen

2 - CenterParent

Default: CenterScreen

See Also

Form, **WindowState Property**

Example

```
GSet frmMain.StartPosition, STARTPOSITION_CENTERSCREEN
```

8.20 T

8.20.1 TabAlignment Property

Applies To
TabControl

Description
Sets or gets the tab alignment that is displayed on a control.

Usage

```
GGet Form.Control.TabAlignment, var
GSet Form.Control.TabAlignment, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: Integer variable that will contain the value of the property.

value
: Integer expression for the new value of the property

This property can be used with GGet and GSet only when RC800 Controller is connected.

Values
0 – Left
1 – Top
2 – Right
3 – Bottom
Default: TOP

See Also
Tab Control

Example

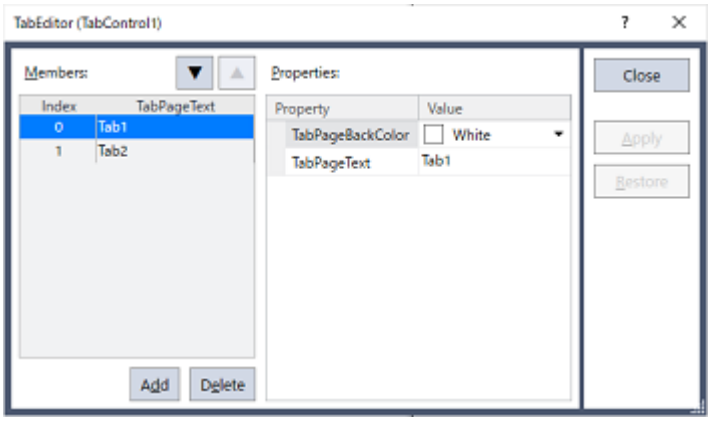
```
' Gets the tab alignment that is currently set.
GGet frmMain.tabControl1.TabAlignment, alignment
' Sets the tab alignment to 0 (Left).
GSet frmMain.tabControl1.TabAlignment, 0
```

8.20.2 TabEditor Property

Applies To
TabControl

Description

Opens TabEditor to set up the Tab control.



Usage

You can use the TabEditor property only when designing. You can add or delete tab pages, set the background color of the tab pages, and set the tab text.

Setting	Description
Add Column	Adds a column at the end of the grid. Minimum: 1 Maximum: 25
Add Row	Adds a row at the bottom of the grid. Minimum: 0 Maximum: 10000
Allow Editing	Sets whether the user can edit cells in the selected column at runtime.
Allow Resizing	Sets whether the user can resize the selected column at runtime.
Back Color	Sets the back color of the selected column.
Cell Alignment	Sets the alignment of the cell text. Left: Aligns the cell text to the left of the column Center: Aligns the cell text in the center of the column Right: Aligns the cell text to the right of the column
Font	Sets the font for the column headers.
Header Alignmnent	Sets the alignment of the column headers. Left: Aligns the column name to the left of the column Center: Aligns the column name in the center of the column Right: Aligns the column name to the right of the column
Header Lines	Sets the amount of lines available in the column headers. Minimum: 1 Maximum: 3
Remove Column	Removes the selected column.
Remove Row	Removes the selected row.

Setting	Description
Resize Method	Decides how to resize the columns in the grid whenever a column is added, removed, or resized. Auto: Automatically resizes columns to fit the grid control. Free: Each column has their own independent width. Synchronized: Makes all of the columns the same size when a column is resized.
Row Count	
Show Row Selectors	Sets whether to show the row selectors that are located on the left side of the grid.
Width	Sets/displays the width of the selected column. To manually change this value, Resize Method must not be Auto.

Setting	Description
Add	Adds a new tab page after the last page.
Delete	Deletes the selected tab page.
TabPageBackColor	Sets the background color for the selected tab page.
TabPageText	Sets the tab text for the selected tab page.

Values
0 – Left
1 – Top
2 – Right
3 – Bottom
Default: TOP

See Also
Tab Control

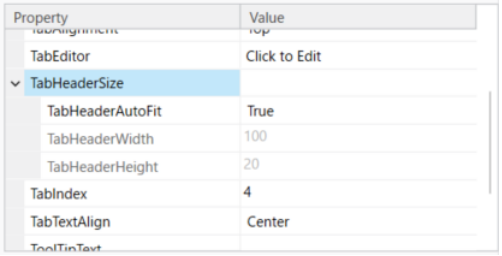
8.20.3 TabHeaderSize Property

From GUI Only

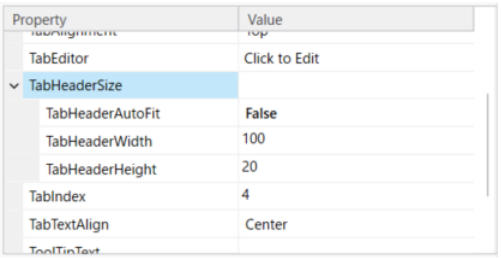
Applies To
TabControl

Description
Sets the size of tab header. (Whether to automatically adjust the width or height, manually adjust the width and height)

Usage
By changing the value of the TabHeaderSize property, which is displayed in the GUI Builder's property grid, you can change size of the tab header.



After the settings have been changed, and the project saved, the settings will be saved. If the TabHeaderAutoFit property is set to False, the TabheaderWidth and TabHeaderHeight property can be changed.



See Also

Tab Control

8.20.4 TabHeaderTextAlign Property

Applies To

TabControl

Description

Gets or sets the alignment of the tab header text.

Usage

```
GGet Form.Control.TabHeaderTextAlign, var
GSet Form.Control.TabHeaderTextAlign, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: String variable that will contain the value of the property.

value
: String expression for the new value of the property

Values

1 – Left

2 – Center

3 – Right

Default:Center

See Also

Tab Control

Example

```
'Gets the alignment of the tab header text that is currently set.
GGet frmMain.tabControl1.TabAlignment, alignment
'Sets the alignment of the tab to 1(Left).
GSet frmMain.tabControl1.TabAlignment, 1
```

8.20.5 TabHeaderAutoFit Property

Applies To

TabControl

Description

Gets or sets whether to automatically adjust the tab header size to the text size.

Usage

```
GGet Form.Control.TabHeaderAutoFit, var
GSet Form.Control.TabHeaderAutoFit, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: String variable that will contain the value of the property.

value
: String expression for the new value of the property

Values

0 – False

1 – True

Default:0 – False

See Also

Tab Control

Example

```
'Gets the adjustment mode of the tab header that is currently set.
GGet frmMain.tabControl1.TabHeaderAutoFit, autoFit
'Sets the tab header's adjustment mode to False (Manual adjustment).
GSet frmMain.tabControl1.TabHeaderAutoFit, False
```

8.20.6 TabHeaderHeight Property

Applies To

TabControl

Description

Gets or sets the height of tab header in pixels.

Usage

```
GGet Form.Control.TabHeaderHeight, var
GSet Form.Control.TabHeaderHeight, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: String variable that will contain the value of the property.

value
: String expression for the new value of the property

Values

An integer representing the height in pixels.

See Also

Tab Control

Example

```
'Gets the height of the tab header that is currently set.
GGet frmMain.tabControl1.TabHeaderHeight, height
'Sets the tab header's height. (To reflect the set value, TabHeaderAutoFit must be
set to False.)
GSet frmMain.tabControl1.TabHeaderAutoFit, False
GSet frmMain.tabControl1.TabHeaderHeight, 48
```

8.20.7 TabHeaderWidth Property

Applies To

TabControl

Description

Gets or sets the width of Tab Header in pixels.

Usage

```
GGet Form.Control.TabHeaderWidth, var
GSet Form.Control.TabHeaderWidth, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: String variable that will contain the value of the property.

value
: String expression for the new value of the property

Values

An integer representing the width in pixels.

See Also

Tab Control

Example

```
'Gets the width of the tab header that is currently set
GGet frmMain.tabControl1.TabHeaderWidth, width
'Sets the tab header width. (To reflect the set value, the TavHeaderAutoFit must be
set to False.)
GSet frmMain.tabControl1.TabHeaderAutoFit, False
GSet frmMain.tabControl1.TabHeaderWidth, 120
```

8.20.8 TabIndex Property

Applies To

Button, Label, TextBox, RadioButton, CheckBox, ListBox, ComboBox, PictureBox, GroupBox, VideoBox, LED, ProgressBar, TrackBar, StatusBar, Grid,TabControl, Wizard, TreeView

Description

Gets or sets the tab order of the control within its container.

Usage

```
GGet Form.Control.TabIndex, var
GSet  Form.Control.TabIndex, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: Integer variable that will contain the value of the property.

value
: Integer expression for the new value of the property.

Values

The index of the tab order of the control.

See Also

Button Control, **Label Control**, **TextBox Control**, **RadioButton Control**, **CheckBox Control**, **ListBox Control**, **ComboBox Control**, **PictureBox Control**, **GroupBox Control**, **VideoBox Control**, **LED Control**, **ProgressBar Control**, **TrackBar Control**, **StatusBar Control**, **Wizard Control**, **Tree View Control**

Example

```
GSet frmMain.txtStatus.TabIndex, 3
```

8.20.9 TabPageBackColor Property

Applies To

TabControl

Description

Sets the background color for the tab page that is displayed on the Tab control.

Usage

```
GGet Form.Control.TabPageBackColor(index), var
GSet Form.Control.TabPageBackColor(index), value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

Index
: An index of the Tab control.

var
: String variable that will contain the value of the property.

value
String expression for the new value of the property

This property can be used with GGet and GSet only when RC800 Controller is connected.

Values
The name for the color to assign to the tab page background color.
Default: Control

See Also
Tab Control, **TabEditor Property**

Example

```
' Gets the tab page background color for index 2 that is currently set.
GGet frmMain.tabControl1.TabPageBackColor(2), backColor
' Sets the tab page background color for index 0 to "Red."
GSet frmMain.tabControl1.TabPageBackColor(0), "Red"
```

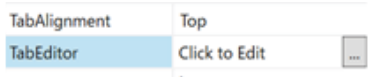
8.20.10 TabPageFont Property

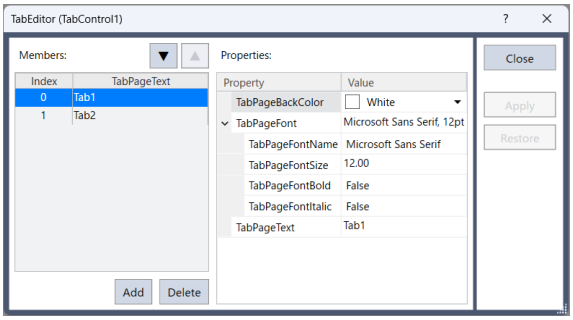
From GUI Only

Applies To
TabControl

Description
The TabPageFont property allows you to change the font of the text in the TabControl's tab page during design time. (Font name, style, size)

Usage
Displays the tab editor by pressing down the TabEditor button displayed in the GUI Builder's property grid. The font displayed can be changed by changing the TabPageFont property value with the tab editor.





After the settings have been changed, and the project saved, the settings will be saved.

See Also
Tab Control

8.20.11 TabPageFontBold Property

Applies To

TabControl

Description

The TabPageFontBold property allows you to get and set the font of the text in the TabControl tab page to "standard" or "bold" during design time.

Usage

```
GGet Form.Control.TabPageFontBold(index), var
GSet Form.Control.TabPageFontBold(index), value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

Index
: An index of the Tab control.

var
: Boolean variable that will contain the value of the property.

value
: Boolean expression for the new value of the property.

This property can be used with GGet and GSet only when RC800 Controller is connected.

Values

0 - False

1 - True

Default: 0 - False

See Also

Tab Control, **TabPageFontName Property**, **TabPageFontItalic Property**, **TabPageFontSize Property**

Example

```
'Get the font style (standard, bold) for the tab page text of index number 2
GGet frmMain.tabControl1.TabPageFontBold(2), fontBold
'Set the font style for the tab page text of index number 0 to bold.
GSet frmMain.tabControl1.TabPageFontBold(0), True
```

8.20.12 TabPageFontItalic Property

Applies To

TabControl

Description

The TabPageFontItalic property sets and gets the font of the text in the TabControl tab page to "standard" or "italic" during design time.

Usage

```
GGet Form.Control.TabPageFontItalic(index), var
GSet Form.Control.TabPageFontItalic(index), value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

Index
: An index of the Tab control.

var
: Boolean variable that will contain the value of the property.

value
: Boolean expression for the new value of the property.

Values

0 - False 1 - True

Default: 0 - False

See Also

Tab Control, **TabPageFontName Property**, **TabPageFontBold Property**, **TabPageFontSize Property**

Example

```
'Get the font style (standard, italic) for the tab page text of index number 2 that
is currently set.
GGet frmMain.tabControl1.TabPageFontItalic(2), fontItalic
'Set the font style for the tab page text of index number 0 as italic.
GSet frmMain.tabControl1.TabPageFontItalic(0), True
```

8.20.13 TabPageFontName Property

Applies To

TabControl

Description

The TabPageFontName property allows you to get and set the font name of the text in the TabControl's tab page.

Usage

```
GGet Form.Control.TabPageFontName(index), var
GSet Form.Control.TabPageFontName(index), value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

Index
: An index of the Tab control.

var
: String variable that will contain the value of the property.

value
: String expression for the new value of the property.

Values

A string containing the name of the font.
Default: Microsoft Sans Serif

See Also

Tab Control, **TabPageFontSize Property**, **TabPageFontItalic Property**, **TabPageFontBold Property**

Example

```
'Get the font name for the tab page text of index number 2 that is currently set
GGet frmMain.tabControl1.TabPageFontName(2), fontName
'Set the font name for the tab page text of index number 0 as "Courier New".
GSet frmMain.tabControl1.TabPageFontName(0), "Courier New"
```

8.20.14 TabPageFontSize Property

Applies To

TabControl

Description

The TabPageFontSize property allows you to get and set the font size of the text in the TabControl's tab page.

Usage

```
GGet Form.Control.TabPageFontSize(index), var
GSet Form.Control.TabPageFontSize(index), value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

Index
: An index of the Tab control.

var
: Real variable that will contain the value of the property.

value
: Real expression for the new value of the property.

This property can be used with GGet and GSet only when RC800 Controller is connected.

Values

The value of the font size in points.

Default: 12

See Also

Tab Control, **TabPageFontName Property**, **TabPageFontItalic Property**, **TabPageFontBold Property**

Example

```
'Get the font size for the tab page text of index number 2 that is currently set
GGet frmMain.tabControl1.TabPageFontSize(2), fontSize
'Set the font size for the tab page text of index number 0 to 20.
GSet frmMain.tabControl1.TabPageFontSize(0), 20
```

8.20.15 TabPageIcon Property

Applies To

TabControl

Description

Sets or gets the icon image of the header for the tab page that is displayed on the Tab control.

Usage

```
GGet Form.Control.TabPageIcon(index), var
GSet Form.Control.TabPageIcon(index), value
```

Form
: Usage

Control
: Name of a control or string variable containing a control name.

Index
: An index of the Tab control.

var
: String variable that will contain the value of the property.

value
: String expression for the new value of the property.

Values

A string containing the full path to an image file.

Default: Empty

See Also

Tab Control, **TabEditor Property**

Example

```
' Gets the icon for the header of the tab page for index 2 that is currently set.
GGet frmMain.tabControl1.TabPageIcon(2), icon$
' Sets the tab page background color for index 0 to "LedGreen.ico."
GSet frmMain.tabControl1.TabPageIcon(0), ”C:\EpsonRC80\GUI\Icons\LedGreen.ico”
```

8.20.16 TabPageText Property

Applies To

TabControl

Description

Sets the tab text that is displayed on the Tab control.

Usage

```
GGet Form.Control.TabPageText(index), var
GSet Form.Control.TabPageText(index), value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

Index
: An index of the Tab control.

var
: String variable that will contain the value of the property.

value
: String expression for the new value of the property

This property can be used with GGet and GSet only when RC800 Controller is connected.

Values

The text to assign to a tab.

Default: TabXX

See Also

Tab Control, **TabEditor Property**

Example

```
' Gets the tab text for index 2 that is currently set.
GGet frmMain.tabControl1.TabPageText(2), backColor
' Sets the tab text for index 0 to "Monitor."
GSet frmMain.tabControl1.TabPageText(0), "Monitor"
```

8.20.17 Text Property

Applies To

Form, Button, Label, TextBox, RadioButton, CheckBox, ComboBox, GroupBox, LED, StatusBar,TreeView

Description

Gets or sets the text associated with the control.

Usage

```
GGet Form.[Control].Text, var
GGet Form.[Control].Text(name), var
GSet Form.[Control].Text, value
GSet Form.[Control].Text(name), var
```

Form

Name of a form or string variable containing a form name.

Control
Name of a control or string variable containing a control name.

name
Specify the string that indicates the item (node) of the tree view which sets the text.

var
String variable that will contain the value of the property

value
String expression for the new value of the property.

Values

A string containing the text of the form or control.

See Also

Form, **Button Control**, **Label Control**, **TextBox Control**, **RadioButton Control**, **CheckBox Control**, **ComboBox Control**, **GroupBox Control**, **LED Control**, **StatusBar Control**, **TextAlign Property**, **Tree View Control**

Example

```
GSet frmMain.lblName.Text, "Name: "
```

8.20.18 TextAlign Property

Applies To

Button, Label, TextBox, RadioButton, CheckBox, LED

Description

Gets or sets the alignment of text in the control.

Usage

```
GGet Form.Control.TextAlign, var
GSet  Form.Control.TextAlign, value
```

Form
Name of a form or string variable containing a form name.

Control
Name of a control or string variable containing a control name.

var
Integer variable that will contain the value of the property.

value
Integer expression for the new value of the property.

Values

- For Button, Label, RadioButton, CheckBox controls
  1 - TopLeft
  2 - TopCenter
  3 - TopRight
  4 - MiddleLeft
  5 - MiddleCenter
  6 - MiddleRight
  7 - BottomLeft
  8 - BottomCenter
  9 - BottomRight
  Default: 1 - TopLeft (Label 1)
  4 - MiddleLeft (RadioButton, CheckBox)

5 - MiddleCenter (Button)
6 - MiddleRight (LED)

- For TextBox
  1 - Left
  2 - Center
  3 - Right
  Default: Left

See Also

Button Control, **Label Control**, **TextBox Control**, **RadioButton Control**, **CheckBox Control**, **LED Control**, **Text Property**

Example

```
GSet frmMain.lblName.TextAlign, TEXTALIGN_LEFT
```

8.20.19 Tick Event

Applies To

Timer

Description

Occurs when a Timer control reaches its Interval value.

Usage

```
Form_Control_Tick (Sender$ As String)
```

Sender$
Name of a control that sent the event.

See Also

Timer Control, **Interval Property**, **Enabled Property**

Example

```
Function frmMain_Timer1_Tick(Sender$ As String)
  GSet frmMain.lblDateTime.Text, Date$ + " " + Time$
Fend
```

8.20.20 TickFrequency Property

Applies To

TrackBar

Description

Gets or sets the interval of the tick marks on a TrackBar control.

Usage

```
GGet Form.Control.TickFrequency, var
GSet  Form.Control.TickFrequency, value
```

Form

Name of a form or string variable containing a form name.

Control
Name of a control or string variable containing a control name.

var
Integer variable that will contain the value of the property.

value
Integer expression for the new value of the property.

Values
An integer representing the interval of the tick marks.

See Also
TrackBar Control

Example

```
GSet frmMain.TrackBar1.TickFrequency, 5
```

8.20.21 TickStyle Property

Applies To
TrackBar

Description
Gets or sets where to display the tick marks on a TrackBar control.

Usage

```
GGet Form.Control.TickStyle, var
GSet  Form.Control.TickStyle, value
```

Form
Name of a form or string variable containing a form name.

Control
Name of a control or string variable containing a control name.

var
Integer variable that will contain the value of the property.

value
Integer expression for the new value of the property.

Values
0 - None
1 - TopLeft
2 - BottomRight
3 - Both
Default: BottomRight

See Also
TrackBar Control

Example

```
GSet frmMain.TrackBar1.TickStyle, TICKSTYLE_BOTH
```

8.20.22 ToolTipText Property

Applies To

Button, Label, TextBox, RadioButton, CheckBox, ListBox, ComboBox, PictureBox, GroupBox, VideoBox, LED, ProgressBar, TrackBar, StatusBar, Grid , TabControl, Wizard, TreeView

Description

Used to show a small rectangular pop-up window that displays a brief description of a control's purpose when the user rests the mouse pointer over the control.

Usage

```
GGet Form.Control.ToolTipText, var
GSet  Form.Control.ToolTipText, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: String variable that will contain the value of the property.

value
: String expression for the new value of the property.

Values

A string containing the tool tip text of the control.
Default: Empty string

See Also

Button Control, **Label Control**, **TextBox Control**, **RadioButton Control**, **CheckBox Control**, **ListBox Control**, **ComboBox Control**, **PictureBox Control**, **GroupBox Control**, **VideoBox Control**, **LED Control**, **ProgressBar Control**, **TrackBar Control**, **StatusBar Control**, **Tab Control**, **Wizard Control**, **Tree View Control**

Example

```
GSet frmMain.btnStart.ToolTipText, "Click Here to Start"
```

8.20.23 Top Property

Applies To

Form, Button, Label, TextBox, RadioButton, CheckBox, ListBox, ComboBox, PictureBox, GroupBox, VideoBox, LED, ProgressBar, TrackBar, StatusBar, Grid , TabControl, Wizard, TreeView

Description

Get or sets the top coordinate of a form or control in pixels.

Usage

```
GGet Form.[Control].Top, var
GSet  Form.[Control].Top, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: Integer variable that will contain the value of the property.

value
Integer expression for the new value of the property.

Values
An integer representing the top coordinate in pixels.

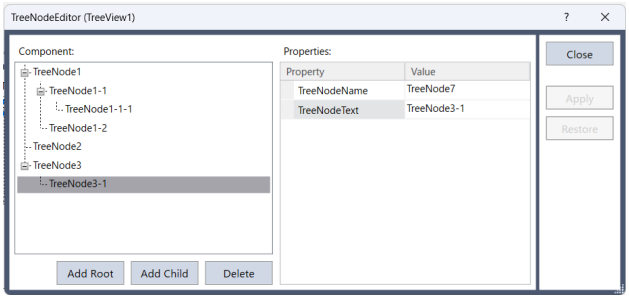
See Also
Form, **Button Control**, **Label Control**, **TextBox Control**, **RadioButton Control**, **CheckBox Control**, **ListBox Control**, **ComboBox Control**, **PictureBox Control**, **GroupBox Control**, **VideoBox Control**, **LED Control**, **ProgressBar Control**, **TrackBar Control**, **StatusBar Control**, **Tab Control**, **Wizard Control**, **Tree View Control**, **Left Property**, **Height Property**, **Width Property**

Example
```
GSet frmMain.txtStatus.Top, 200
```

8.20.24 TreeNodeEditor Property

Applies To
TreeView

Description
Opens the TreeNodeEditor to setup the tree view control.



Usage
You can use the TreeNodeEditor property only when designing. You can add or delete nodes, set the node name and the displayed text.

Setting	Description
Add to root	Add a new node to the top of the level.
Add child element	Add a new child node below the currently selected node.
Delete	Delete both the currently selected node and its child node.
TreeNodeName	Sets the name of the selected node.
TreeNodeText	Sets the displayed text for the selected node.

Selecting, dragig, and ropping the node allows you to rearrange the node

Values
An integer representing the top coordinate in pixels.

See Also
Tree View Control

8.20.25 Type Property

Applies To

All Controls

Description

Gets the type name of a control.

Usage

```
GGet Form.Control.Type, var
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: String variable that will contain the value of the property.

Values

A string containing the name of the type of the control.

See Also

Controls Property

Example

```
GGet frmMain.Controls(index).Type, typeName$
```

8.21 U

8.21.1 Update Property

Applies To
TextBox

Description
Gets or sets if the value of textbox is updated automatically or not when it is set to display the global variable.

Usage

```
GGet Form.Control.Update, var
GSet  Form.Control.Update, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: Boolean variable that will contain the value of the property.

value
: Boolean expression for the new value of the property.

Values
False

True

Default: True

Remarks
To show a SPEL+ global variable (except arrays) on a textbox, set the global variable name in the Variable property. If the Update property is set to False, the global variable value is not updated.

When you set the TextBox Variable property to a global variable, by default RC+ automatically updates the TextBox Text value with the value of the variable at runtime. The Update property default value is True. If you want to allow a user to change the value of the same variable, you must set Update to False. This stops the auto update so the user can type in a new value.

When Update is set from False to True, RC+ sets the value of the variable to the new value entered by the user in the TextBox.

One method is to add a CheckBox above the TextBox called “Change Value” or some similar name. In the ChechkBox Click event, use GGet to get the checked state of the CheckBox and then use GSset to set the TextBox Update property. When the box is checked, set Update to False.

See Also
TextBox Control, **Variable Property**

Example

```
GSet frmMain.txtStatus.Update, False
```

8.22 V

8.22.1 Value Property

Applies To
ProgressBar, TrackBar

Description
Gets or sets the value that represents the control's current position.

Usage

```
GGet Form.Control.Value, var
GSet  Form.Control.Value, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name. The control must exist in the specified form.

var
: Integer variable that will contain the value of the property.

value
: Integer expression for the new value of the property.

Values
An integer representing the current value.

See Also
ProgressBar Control, **TrackBar Control**, **Maximum Property**, **Minimum Property**, **Variable Property**

Example

```
GSet form1.ProgressBar1.Value, 75
```

8.22.2 Variable Property

Applies To
TextBox, ProgressBar, TrackBar

Description
Gets or sets the name of a SPEL+ global variable (except arrays) used for the textbox display.

Usage

```
GGet Form.Control.Variable, var
GSet  Form.Control.Variable, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name. The control must exist in the specified form.

var
: String variable that will contain the value of the property.

value
: String expression for the new value of the property.

Values
The name of a SPEL+ global variable (except arrays) in the current project.
Default: None

Remarks
When the Variable property is set to a SPEL+ global variable (except arrays), then the Text value is automatically updated at runtime to show the value of the variable.
When you set the TextBox Variable property to a global variable, by default RC+ automatically updates the TextBox Text value with the value of the variable at runtime. The Update property default value is True. If you want to allow a user to change the value of the same variable, you must set Update to False. This stops the auto update so the user can type in a new value.
When Update is set from False to True, RC+ sets the value of the variable to the new value entered by the user in the TextBox. One method is to add a CheckBox above the TextBox called “Change Value” or some similar name. In the ChechkBox Click event, use GGet to get the checked state of the CheckBox and then use GSset to set the TextBox Update property. When the box is checked, set Update to False.
When you set the ProgressBar Variable property to a global variable, the progress bar will display the value of the variable.
When you set the TrackBar Variable property to a global variable, the variable value is set by the track bar.

See Also
TextBox Control, **ProgressBar Control**, **TrackBar Control**, **Value Property**

Example
```
GSet frmMain.txtStatus.Variable, "g_Status$"
```

8.22.3 VideoEnabled Property

Applies To
VideoBox

Description
Gets or sets whether the VideoBox displays video or not.

Usage
```
GGet Form.Control.VideoEnabled, var
GSet  Form.Control.VideoEnabled, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: Boolean variable that will contain the value of the property.

value
: Boolean expression for the new value of the property.

Values
False
True
Default: False

See Also
VideoBox Control, **GraphicsEnabled Property**

Example

```
GSet frmMain.VideoBox1.VideoEnabled, True
```

8.22.4 Visible Property

Applies To

Button, Label, TextBox, RadioButton, CheckBox, ListBox, ComboBox, PictureBox, GroupBox, VideoBox, LED, ProgressBar, StatusBar, TrackBar, Grid , TabControl, Wizard, TreeView

Description

Gets or sets a value indicating whether the control is displayed at runtime.

Usage

```
GGet Form.Control.Visible, var
GSet  Form.Control.Visible, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: Boolean variable that will contain the value of the property.

value
: Boolean expression for the new value of the property.

Values

True Display the control

False Don't display the control

Default: True

Remarks

If GGet is executed without loading a form by GShow or GShowDialog, the return value will be False.

See Also

Button Control, **Label Control**, **TextBox Control**, **RadioButton Control**, **CheckBox Control**, **ListBox Control**, **ComboBox Control**, **PictureBox Control**, **GroupBox Control**, **VideoBox Control**, **LED Control**, **ProgressBar Control**, **StatusBar Control**, **TrackBar Control**, **Enabled Property**, **Tab Control**, **Wizard Control**, **Tree View Control**

Example

```
GSet frmMain.txtStatus.Visible, True
```

8.23 W

8.23.1 Width Property

Applies To

Form, Button, Label, TextBox, RadioButton, CheckBox, ListBox, ComboBox, PictureBox, GroupBox, VideoBox, LED, ProgressBar, TrackBar, StatusBar, Grid , TabControl, Wizard, TreeView

Description

Get or sets the width of a form or control in pixels.

Usage

```
GGet Form.[Control].Width, var
GSet  Form.[Control].Width, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name.

var
: Integer variable that will contain the value of the property.

value
: Integer expression for the new value of the property.

Values

An integer representing the width in pixels.

See Also

Form, **Button Control**, **Label Control**, **TextBox Control**, **RadioButton Control**, **CheckBox Control**, **ListBox Control**, **ComboBox Control**, **PictureBox Control**, **GroupBox Control**, **VideoBox Control**, **LED Control**, **ProgressBar Control**, **TrackBar Control**, **StatusBar Control**, **Tab Control**, **Wizard Control**, **Tree View Control**, **Left Property**, **Top Property**, **Height Property**

Example

```
GSet frmMain.txtStatus.Width, 300
```

8.23.2 WindowState Property

Applies To

Form

Description

Gets or sets the form's window state.

Usage

```
GGet Form.WindowState, var
GSet  Form.WindowState, value
```

Form
: Name of a form or string variable containing a form name.

var
: Integer variable that will contain the value of the property.

value
Integer expression for the new value of the property.

Values
0 - Normal
1 - Minimized
2 - Maximized
Default: Normal

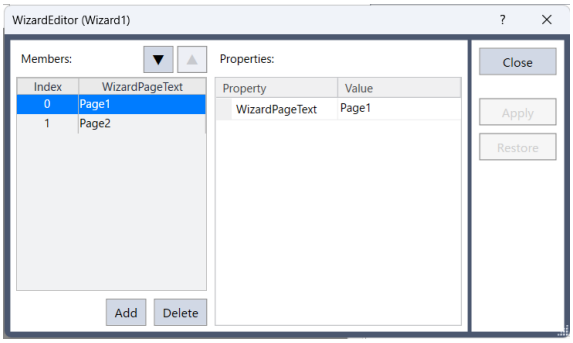
See Also
Form

Example
```
GSet frmMain.WindowState, WINDOWSTATE_MAXIMIZED
```

8.23.3 WizardEditor Property

Applies To
Wizard

Description
Opens WizardEditor to set up the Wizard control.



Usage
You can use the WizardEditor property only when designing. You can add or delete pages, and set the characters displayed on the page.

Setting	Description
Add	Adds a new page after the last page.
Delete	Deletes the selected page.
WizardPageText	Sets the text of the tab for the selected page.

Values
True : The multiline text box control wraps words
False : The text box control automatically scrolls horizontally when the user types past the right edge of the control.
Default: True

See Also
Wizard Control

8.23.4 WordWrap Property

Applies To

TextBox

Description

Indicates whether a multiline text box control automatically wraps words to the beginning of the next line when necessary.

Usage

```
GGet Form.Control.WordWrap, var
GSet  Form.Control.WordWrap, value
```

Form
: Name of a form or string variable containing a form name.

Control
: Name of a control or string variable containing a control name. The control must exist in the specified form.

var
: Boolean variable that will contain the value of the property.

value
: Boolean expression for the new value of the property.

Values

True : The multiline text box control wraps words

False : The text box control automatically scrolls horizontally when the user types past the right edge of the control.

Default: True

See Also

TextBox Control, **MultiLine Property**, **ScrollBars Property**

Example

```
GSet frmMain.txtStatus.WordWrap, True
```