|  |
| --- |
| Click or drag to resize |
|  | I/O system domain |
|  | Overview |
|  | A robot system uses input and output signals to control processes. Signals can be of digital, analog, or group signal type. Such I/O signals are accessible using the SDK. |
|  |  |
|  | Signal changes in the robot system are often significant, and there are many scenarios where end-users of the system need notification of signal changes. |
|  |  |
|  | To speed up event notification from the controller, there is new functionality in PC SDK, which allows you to set up subscription priorities. This possibility applies to I/O signals and persistent RAPID data. This mechanism is further described in Implementing high priority event subscription. . |
|  |  |
|  | Accessing signals |
|  | Accessing signals is done through the Controller object and its property IOSystem , which represents the I/O signal space in the robot controller. |
|  |  |
|  | To access a signal you need the system name of the signal. The object that is returned from the IOSystem.GetSignal. method is of type Signal . |
|  |  |
|  | C# |
|  | Copy |
|  | Signal signal1 = aController.IOSystem.GetSignal("signal\_name"); |
|  | The returned Signal object has to be typecast to digital, analog or group signal. This example shows a how a signal of type DigitalSignal is created: |
|  |  |
|  | C# |
|  | Copy |
|  | DigitalSignal diSig = (DigitalSignal)signal1; |
|  | This example shows a how an AnalogSignal is created: |
|  |  |
|  | C# |
|  | Copy |
|  | AnalogSignal aiSig = (AnalogSignal)signal2 |
|  | This example shows a how a GroupSignal is created: |
|  |  |
|  | C# |
|  | Copy |
|  | GroupSignal giSig = (GroupSignal)signal3; |
|  | Note Note |
|  | Remember to call the Dispose method of the signal when it should no longer be used. |
|  |  |
|  | Getting signals using SignalFilter |
|  | Instead of just getting one signal at a time you can get a signal collection using a signal filter. Some of the SignalFilter flags are mutually exclusive, for example SignalFilter.Analog and SignalFilter.Digital. Others are inclusive, for example SignalFilter.Digital and SignalFilter.Input. You can combine the filter flags using the “|” character in C# and the or operator in VB: |
|  |  |
|  | C# |
|  | Copy |
|  | IOFilterTypes aSigFilter = IOFilterTypes.Digital | IOFilterTypes.Input; SignalCollection signals = |
|  | aController.IOSystem.GetSignals(aSigFilter); |
|  | The following code iterates the signal collection and adds all signals to a ListView control. The list has three columns displaying signal name, type, and value: |
|  |  |
|  | C# |
|  | Copy |
|  | foreach (Signal signal in signals) |
|  | { |
|  | item = new ListViewItem(signal.Name); |
|  | item.SubItems.Add(signal.Type.ToString()); |
|  | item.SubItems.Add(signal.Value.ToString()); |
|  | listView1.Items.Add(item); |
|  | } |
|  | If the signal objects are no longer needed they should be disposed of: |
|  |  |
|  | C# |
|  | Copy |
|  | foreach (Signal signal in signals) |
|  | { |
|  | signal.Dispose(); |
|  | } |
|  | Reading I/O signal values |
|  | The following examples show how to read a digital and an analog signal. |
|  |  |
|  | Digital signal |
|  | The following code reads the digital signal DO1 and selects a checkbox if the signal value is 1 (ON): |
|  |  |
|  | C# |
|  | Copy |
|  | Signal sig = aController.IOSystem.GetSignal("DO1"); |
|  | DigitalSignal |
|  | digitalSig = (DigitalSignal)sig;int val = digitalSig.Get(); |
|  | if (val == 1) |
|  | { |
|  | this.checkBox1.Checked = true; |
|  | } |
|  | Analog signal |
|  | The following of code reads the value of the analog signal AO1 and displays it in a textbox: |
|  |  |
|  | C# |
|  | Copy |
|  | Signal asig = aController.IOSystem.GetSignal("AO1"); |
|  | AnalogSignal analogSig = (AnalogSignal)asig; |
|  | float analogSigVal = analogSig.Value; |
|  | this.textBox1.Text = analogSigVal.ToString(); |
|  | Writing I/O signal values |
|  | The following section shows how the value of a digital or an analog I/O signal can be modified by a PC SDK application. |
|  |  |
|  | Note Note |
|  | In manual mode, a signal value can be modified only if the Access Level of the signal is ALL . If not, the controller has to be in auto mode. |
|  |  |
|  | Digital signal |
|  | The following code changes the value of a digital signal in the controller when you select/unselect a checkbox: |
|  |  |
|  | C# |
|  | Copy |
|  | private void checkBox1\_Click(object sender, EventArgs e) |
|  | { |
|  | if (this.checkBox1.Checked) |
|  | { |
|  | digitalSig.Set(); |
|  | } |
|  | else |
|  | { |
|  | digitalSig.Reset(); |
|  | } |
|  | } |
|  | Note Note |
|  | You can also set the value using the Value property. |
|  | Analog signal |
|  | The following code writes the value entered in a text box to the analog signal AO1. The value is converted from string to a float before it is written to the controller: |
|  |  |
|  | C# |
|  | Copy |
|  | float analogSigVal = Convert.ToSingle(this.textBox1.Text); |
|  | analogSig.Value = analogSigVal; |
|  | Listening to signal changes |
|  | Once a Signal object is available it is possible to add a subscription to its Changed event, which is triggered at a signal change such as changed value, changed simulated status or changed signal quality. |
|  |  |
|  | C# |
|  | Copy |
|  | sig.Changed += new |
|  | EventHandler<SignalChangedEventArgs>(sig\_Changed); |
|  | ... |
|  | private void sig\_Changed(object sender, SignalChangedEventArgs e) |
|  | {..... } |
|  | Start and stop subscriptions |
|  | It is recommended that you activate and deactivate subscriptions to the Changed event if these are not necessary throughout the lifetime of the application: |
|  |  |
|  | C# |
|  | Copy |
|  | sig.Changed += new EventHandler<SignalChangedEventArgs>(sig\_Changed); |
|  | sig.Changed -= new EventHandler<SignalChangedEventArgs>(sig\_Changed); |
|  | Implementing high priority event subscription |
|  | To speed up event notification from the controller, it is possible to set up subscription priorities for I/O signals. To do this, you can use the Subscribe method and the enumeration EventPriority as argument. The example shows an ordinary signal subscription and a subscription with high priority: |
|  |  |
|  | C# |
|  | Copy |
|  | signal.Changed += new EventHandler<SignalChangedEventArgs>(sig\_Changed); |
|  | signal.Subscribe(sig\_Changed, EventPriority.High); |
|  | To deactivate subscriptions with high priority you call the Unsubscribe method like this: |
|  |  |
|  | C# |
|  | Copy |
|  | signal.Unsubscribe(sig\_Changed); |
|  | Limitations for high priority events |
|  | High priority subscriptions can be used for I/O signals and RAPID data declared PERS. The controller can handle 64 high priority subscriptions. |
|  |  |
|  | Avoiding threading conflicts |
|  | The controller events use their own threads, which are different from the application GUI thread. This can cause problems if you want to display signal changes in the application GUI. For more information, see Controller events and threads. |
|  |  |
|  | If an update of the user interface is not necessary, you do not need to take any special action, but can execute the event handler on the event thread. If, however, you need to show to the user that the signal has changed you should use the Invoke method. It forces execution to the window control thread and thus provides a solution to potential threading conflicts. |
|  |  |
|  | C# |
|  | Copy |
|  | this.Invoke(new EventHandler<SignalChangedEventArgs>(UpdateUI), new Object[] { sender, e }); |
|  | Reading the new value |
|  | The SignalChangedEventArgs object has a NewSignalState property, which has information about signal value, signal quality and whether the signal is simulated or not: |
|  |  |
|  | C# |
|  | Copy |
|  | private void UpdateUI(object sender, SignalChangeEventArgs e) |
|  | { |
|  | SignalState state = e.NewSignalState; |
|  | .... |
|  | float val = state.Value |
|  | this.textBox1.Text = val.ToString() |
|  | } |
|  | Note Note |
|  | While setting up a subscription, you may or may not receive an initial event. Use Value property to read the initial information about the value of the signal. |
|  |  |
|  | Note Note |
|  | You must ensure to remove subscription before disposing the signal. For more information, see Accessing the controller. |