

WonderWare InControl and SeaIO ActiveX

The purpose of this White Paper is to document how to configure WonderWare InControl to use Sealevel Systems SeaIO ActiveX control.

Sealevel Systems SeaIO ActiveX control is implemented as an ActiveX DLL. This means the SeaIO ActiveX control will be loaded into the process space of the ActiveX container application. Since the SeaIO ActiveX control will reside in the process space of the container application, there is no need for Marshaling the data between the container application and the SeaIO ActiveX object. This reduces overhead and increases performance over an ActiveX EXE implementation. Another advantage is the ability to be used by any OLE automation client, such as Microsoft Office applications.

This paper assumes that WonderWare InControl is already installed on your system. It also assumes that SeaIO is already installed on your system and that all installed SeaIO cards are functioning properly. Begin by starting a new project. The main screen should appear as follows:



The first step is to add the Sealevel ActiveIO Control to your new project. Right click "Programs" on the left side of the main screen and select "New Program ..." from the pop-up dialog. The following dialog should now appear. Highlight "Factory Object" and click "OK".

Programs	
Factory Object H RLL Program S SFC Program S Structured Text	
Program (POU) type	
• Program	C Function Block
Eunction	(Macro

The dialog below now appears. Select the "Install Control" tool button (Blue) and the next dialog appears.



stall Control	?
Control:	ОК
ActionBvr Class	▲ <u> </u>
Activelo_Device Control	Cancel
Activelo PortDisplay Control	
ActiveMovie Control Object	
ActiveXPlugin Object	
ActorBvr Class	
adbanner Class	
Application Data Control	
CodeMax Edit Control	
ColorBvr Class	
COMNSView Class	
Cr Behavior Factory	
CTreeView Control	
DBGrid Control	
Desaware SpuWorks Subclassing OLE Control	

Since Sealevels' SeaIO software is already installed, highlight "ActiveIo_Device Control" and click "OK". You are now returned to the earlier dialog which should now appear as follows:

Select Factory Object		×
Category:	Factory Object:	ОК
FactoryObjects	Wonderware Analog Alarm Control Wonderware InControl Project Information	Cancel
	Wonderware InControl Serial Port Wonderware PID Control	Help

With the ActiveIo Control highlighted, click "OK" and the main project screen should now appears as follows:



Click "Save" and the main project screen will be updated to show the added ActiveX control.

JnControl Development Environment - InControl_ActiveX	Z
File Edit Mew Insert Runtime Tools Window Help	
D 😂 🖬 🕼 📇 X Po 🛍 🛛 🖬 🖉 🛄 🗸 🖓	
Project Wew J. Execution View	
insity	

Right click "Programs" again and select "New Program ..." to bring up the following dialog. Highlight "Structured Text" and click "OK".

New			×
🗋 Programs			l.
Factory Object Factory Object H RLL Program SFC Program Si Structured Text			
Program (POU) type —			
Program		C Function B	lock
C Function		C Macro	
	OK	Cancel	Help

When the next dialog appears, name your structured text language program and click "Save". The main screen should now look something like this:

ontrol Development Environment - InControl_ActiveX	<u>=1</u> 8
át Vew Incert Runtime Tools Window Holp ☞ 🖬 🕼 🍘 🏭 🕺 Ros 🛤 🔽 🌉 📖 🗸 ↓	
Teject View S. Execution View InCost J. Actively Sectors Program Progra	

Next click the "Insert" menu and choose "Symbol". The following dialog should appear. After selecting "ActiveIo_Device1" the items on the right should appear. Select "deviceNumber" and click "OK" to continue.

🚭 Global	Name	Туре	Address	*
- 🗇 Activelo_Device1	32 BackColor	DWORD	N	
🚽 🗇 RTEngine	123 cardld	DINT		
🔄 🗗 STL1	123 deviceHandle	DINT		
📲 User Type Definitions	deviceNumber	INT		
(X) Functions		DWORD		
Tunction Blocks	f(x) GetDirection	Method (INT)		
Macros	123 InputPortCount	DINT		
	123 ioPort	DINT		
	123 irq	INT		
	123 Mode	INT		
	f(x) NotifyInputChange	Method		•
	1		►	
	-]	ОК	

This will insert the ActiveIO "deviceNumber" command into your STL program. To complete this instruction you must assign your SeaIO device number to this command. After doing so the main screen should now look like this:

🕼 InControl Development Environment - InControl_Active8	<u></u> 2
File Edit View Insert Runtime Tools Window Help	
日 📽 🖬 🥔 📾 👗 🕸 応 🔟 🔳 🖉 💷 🗸 💱	
Priesc Were + Execution Vere Syntod Sy	
	<u> </u>

This command is used to open the card for access. In this example our SeaIO device number is 5. Your device number can be determined by using Device Manager to look at the installed SeaIO Devices. This will be shown in parenthesis as a card number (Card X). The next step is to create a Global symbol that can be used to return the status of one input bit. Start by clicking the "Tools" menu and selecting "Symbol Manager". The following dialog should now be active:

Scope: Global 🔄 🧾	Name	ම ් ව ව [Type	Address
Activelo_Device1 Activelo_Device1 BTEngine STL1 User Type Definitions Functions Function Blocks Macros	 Activelo_Device1 	Program Program Program	
Filter: ANY			Close

Make sure the "Global" category is selected and click the "New" symbol button. The following dialog will now appear. Enter the information as it is shown.

ymbol Prope	erties			?
Name:	USB_Input1			
Туре:	BOOL	▼ boole	an value	
Description:				4
				-
Initial value:	False			
	E Retentive v	alue	Constant value	
	🗖 Array	Lower bound:	0	
		Upper bound:	0	
	Indexed bit	Source:		-
		Bit number:	0	
	[]		1	
	Add Global	Cancel	Help	

			Address
Activelo_Device1	Activelo_Device1	Program	1
🚽 🗇 RTEngine	1 RTEngine	Program	
🗖 STL1	₫ STL1	Program	
🚽 🚹 User Type Definitions	SB_Input1	BOOL	
			<u> </u>
	•		

Click the "Add Global" button and you are returned to the previous dialog which now looks like this:

Click the "Close" button to complete adding this symbol. We will now use this symbol in conjunction with our ActiveIO "ReadBit" command to read the status of one input. The symbol and command must be inserted into the STL program. Click the "Insert" menu and select "Symbol" and a familiar dialog now appears:

Scope: Global	<u> </u>	• 6 3 8 1	×?
Global Activelo_Device1 RTEngine STL1 User Type Definitions Functions Function Blocks Macros	Name Pactivelo_Device1 RTEngine STL1 USB_Input1	Type Program Program Program BOOL	Address
Dot Field: <none> Filter: ANY</none>	•	[OK Cancel
Global - 4 Member(s)			

Select "Global" and "USB_Input1" then click "OK" to continue. This process must be repeated to insert the "ReadBit" command. Once again select the "Insert" menu and choose "Symbol". Be sure to select "ActiveIo_Device1" and "ReadBit". The dialog should appear as below:

🚭 Global	Name	Туре	Address 4
Activelo_Device1	f(x) GetDirection	Method (INT)	
🚽 🗇 RTEngine	123 InputPortCount	DINT	
STL1	123 ioPort	DINT	
📲 User Type Definitions	123 irq	INT	
f(x) Functions	123 Mode	INT	
→ Hunction Blocks Macros	f(x) NotifyInputChange	Method	
	123 OutputPortCount	DINT	
	<mark>f(∞)</mark> ReadBit	Method (B	
	f(x) ReadByte	Method (INT)	_
	123 sampleInterval	DINT	
	f(x) SetDirection	Method	
			•
Field: Versex	-	Γ	ΩΚ

The main screen should now appear as below. Please note the parameters associated with the ReadBit command have already been entered. The first parameter specifies the bit to be read and the second parameter specifies the addressing mode, absolute or relative. In this example we are reading the first bit, which is zero based, and the addressing mode is relative. For more information consult SeaIO Help which was installed during the setup of SeaIO.



Now we will add to the project a means to verify that the input is changing state. Right click "Programs" and select "New Program ...".

		2
D Programs		
Factory Object Factory Object FALL Program SFC Program S Structured Text		
Program (POU) typ	C Function Black	

Select "RLL Program" and click "OK". When the pop-up dialog appears, name your RLL program and click "Save". The main screen should now look something like this:



Now we will add a Contact to this rung by using the "Contact Tool" button. Click the "Contact Tool" button and drop the contact on the rung. The following dialog now pops up:

USB_Input1	C Add Local Add Global
Activelo_Device1.ReadBit RTEngine.DivideZero RTEngine.Error RTEngine.FirstScan RTEngine.FirstScanOnAutoSta RTEngine.PowerFail RTEngine.ScanOverRun USB_Input1	Contact type Contact type - / - O Open - / - O Closed - P - O Positive Transition - N - O Negative Transition

Select "USB_Input1" and be sure the contact type is "Closed". In our example we are using a closed contact because the example is done with one of our USB TTL digital I/O cards. With no input connected, the pull-up resistor on the input biases the input to a logical 1. When the input is connected, it is actually switched to ground. The closed contact is used as a means of inverting the input. Click "OK" to continue. The main screen should now appear as below:

🐼 InControl Development Environment	- InControl_ActiveX			
File Edit View Insert Runtime Tools	Window Help			
0 🖻 🖬 💋 🖨 🔠 🔺 🗞 🖻	2 2 3 4 7 7	· · · • • • • • • • • •	24 Ca Ca Car	
Project Wew S+ Execution View Project Wew S+ Execution View Project View Project View Projec	Image: Second	RULIAI (Run)	deviceNamber := \$; ivsIo_Devical ReadBit(0,	
Runtime Engine : Loading project complete, s Setting Runtime Engine to Plum Mode Run Complete Runtime Engine : Dianget to Flum Inode. Runtime Engine : warming Scan exerum her	rating execution of 1/10 divers.		Watch Type	
Ready				Append Normally Closed Contact

You are now ready to run your project. To change the state of the input, we are using one of our Test Adapters, Part No. TA01, which connects to the digital I/O card thru an industry standard 50 pin ribbon cable. This test adapter has dip switches to simulate inputs and LED's to simulate outputs. Though this is a very basic example, it provides the necessary concepts to use our ActiveX control for more complex control applications. To run your project, select the "Runtime" menu and choose "Run Project". The following dialog should appear:

I he H modif config	luntime Engine is up-to-date with all the latest cations. Select Continue to run this guration.
Rel	oad Options
•	Full Restart Reload project configuration and all program files. All currently executing programs will be stopped.
0	Smart Start Reload modified programs only. All modified programs will be stopped, downloaded, and restarted.
0	Continue Resume execution of current Runtime Engine configuration

Be sure "Full Restart" is selected and click "OK". After the project loads, change the state of the input and watch the contact in the RLL program. When the input is true the contact should be green.