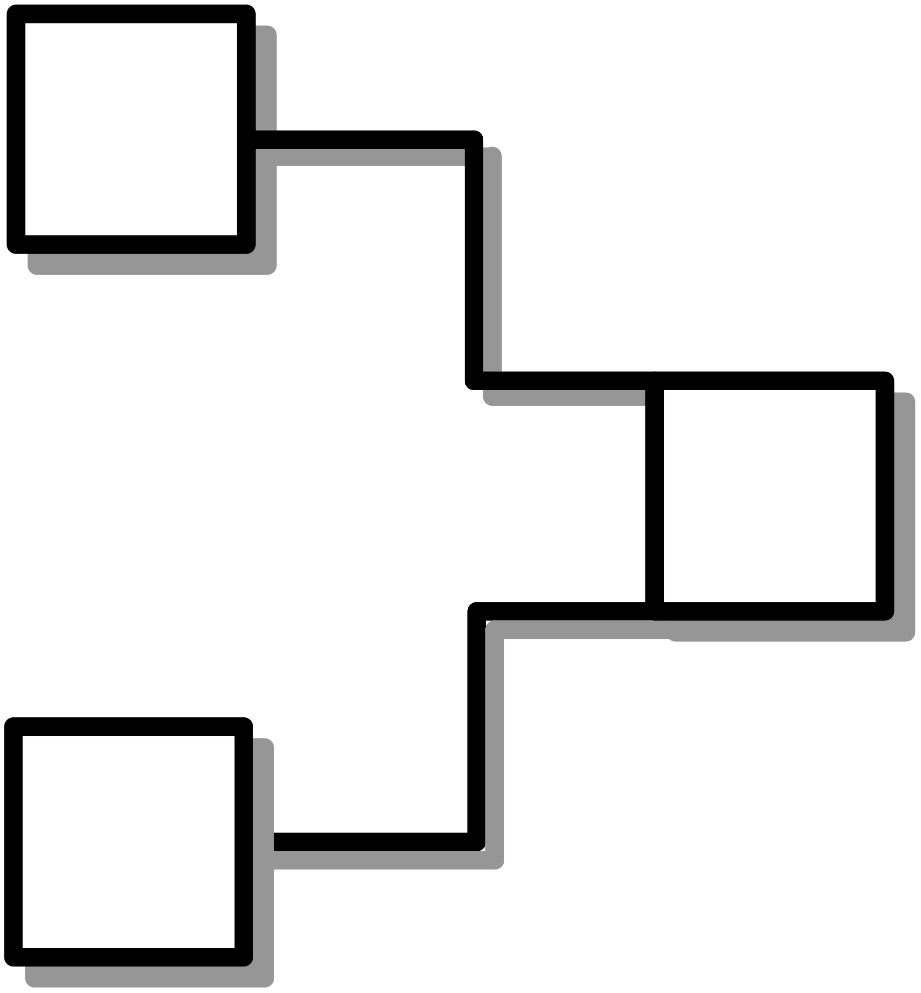
2104001–001 rev. AA

Totalflow®

EZ Block™ Builder Software

User's Guide



TOTALFLOW
MEASUREMENT & CONTROL SYSTEMS

ABB

Intellectual Property & Copyright Notice

EZ Block Builder
Software User Guide

Version 1.0

Document Version 1.0
June 2, 2009

Table of Contents

1 Introduction 5
2 Definitions and Acronyms 6
3 Main Form 8
3.1 Main Menu 10
3.1.1 File submenu 11
3.1.2 Edit submenu 12
3.1.3 Config submenu 13
3.1.4 Help submenu 14
3.2 Toolbar 15
3.3 Program elements 17
3.3.1 Array 18
3.3.2 Bit 20
3.3.3 Compare 21
3.3.4 Complex Block 23
3.3.5 Complex Block Inputs and Outputs 27
3.3.6 Connector 34
3.3.7 External Location 35
3.3.8 Lag 38
3.3.9 Lead 39
3.3.10 Limit 40
3.3.11 Math 41
3.3.12 Periodic 42
3.3.13 PID 46
3.3.14 Pulse 49
3.3.15 Queue 50
3.3.16 Ramp 51
3.3.17 Scale 53
3.3.18 Select 54
3.3.19 Text Box 55
4 Creating a Program 56
5 Register Assignment Strategies 58
6 Alias Lists 59
6.1 Alias List file structure 59
6.2 Loading Alias Lists 60
7 Profiles 61
7.1 Profile file structure 61
7.2 Loading Profiles 62
7.3 Editing Profiles 62
8 Templates 65
8.1 Creating templates 65
8.2 Using templates 66
9 Page Layout and Printing 67
10 Device Operations 68
10.1 Communications Configuration 68

10.1.1 USB Connectivity 69
10.2 Loading a file from a device 71
10.3 Saving a file to a device 73
11 Appendix 75
11.1 Logical, bitwise, and mathematical operations 75
11.2 File structure 76
11.3 Keyboard and Mouse Interaction 77

1 Introduction

This document is intended to serve as a guide for users of the ABB EZ Block Builder application.

The EZ Block Builder, also referred to as EZBB, is a Microsoft Windows-based application that is used to visually design and program ABB X-Series devices.

The EZBB application provides a graphical user interface that allows the user to design programs for X-Series devices by dragging and dropping visual representations of device functions (called “EZ blocks”) onto a work area. The user may then use the application to create a device program by connecting various EZ block inputs and outputs and by configuring the property values of the EZ blocks. EZBB can then be used to transfer these programs to and from an X-Series device.

This document describes how to use the EZ Block Builder application to create these programs.

2 Definitions and Acronyms

CB – Complex Block.

CB Input –"CB Input" refers to a complex block input. These program elements are used to define input pins for complex blocks. CB Inputs are used to provide value data to child blocks contained inside the complex block. Refer to the CB Input section of this document for additional information.

CB Output –"CB Output" refers to a complex block output. These program elements are used to define output pins for complex blocks. CB Outputs are used to source value data to EZ blocks that reside outside the complex block. Refer to the CB Output section of this document for additional information.

CB Periodic Input – "CB Periodic Input" refers to a complex block periodic input. These program elements are used to define input pins for complex blocks that may be connected to periodic outputs external to the complex block. Unlike regular CB Inputs, CB Periodic Inputs are configured to push their values to a single register location inside the complex block, similar to a periodic block output. Refer to the CB Periodic Input section of this document for additional information.

CB Periodic Output – "CB Periodic Output" refers to a complex block periodic output. These program elements are used to define output pins for complex blocks that may be connected to periodic outputs internal to the complex block. CB Periodic Outputs are used to mirror the output pin of a periodic block to program elements external to the complex block. Refer to the CB Periodic Output section of this document for additional information.

Circular reference – A connection which establishes a logical loop is said to contain a circular reference. For example, if the output of EZ block A feeds the input of EZ block B and the output of EZ block B feeds the input of EZ block A, then a circular reference exists. Circular references are not permitted by the application unless one of the program elements in the circular reference is a Periodic block.

Complex block – A complex block (CB) is a user interface construct that encapsulates one or more program element(s). The complex block may be used when it is desired to represent large or common operation as a single block symbol. A user may create a complex block by dragging the toolbox item named "Complex" onto the work area and then double clicking the complex block to navigate inside the function. When templates are dragged onto the work area they appear as complex blocks.

Complex input – Complex inputs are program elements that may be used to define the input pins of a Complex block. Two program elements are used as complex inputs: CB Input and CB Periodic Input.

Complex output – Complex outputs are program elements that may be used to define the output pins of a Complex block. Two program elements are used as complex outputs: CB Output and CB Periodic Output.

Connector – A connector is a toolbox program element (displayed as a line) that represents a relationship between the EZ blocks shown in the work area. By placing a connector on an output pin of an EZ block then dragging the other end of the connector to the input pin of a different EZ block, the user defines the logical connection between to two EZ blocks. Refer to the Connector section of this document for additional information.

EZ block – EZ blocks are configurable program elements that represent device functions or locations. Complex blocks, defined elsewhere in this section, may represent a collection of device functions. External locations represent static register addresses within the device.

Main menu bar – Refer to the `Main Form` section for a complete description.

Program element – A program element is any graphical object that may be dragged from the toolbox to the work area of the main form.

Program tab – Refer to the `Main Form` section for a complete description.

Properties pane – Refer to the `Main Form` section for a complete description.

Program treeview – Refer to the `Main Form` section for a complete description.

Status bar – Refer to the `Main Form` section for a complete description.

Template – A template is an interconnected set of program elements which are 'pre-wired' internally to a specific number of inputs and outputs. When a template is dragged onto the work area from the toolbox, a complex block is instantiated. Templates may be created by the user via the **`File->Create Template…`** operation.

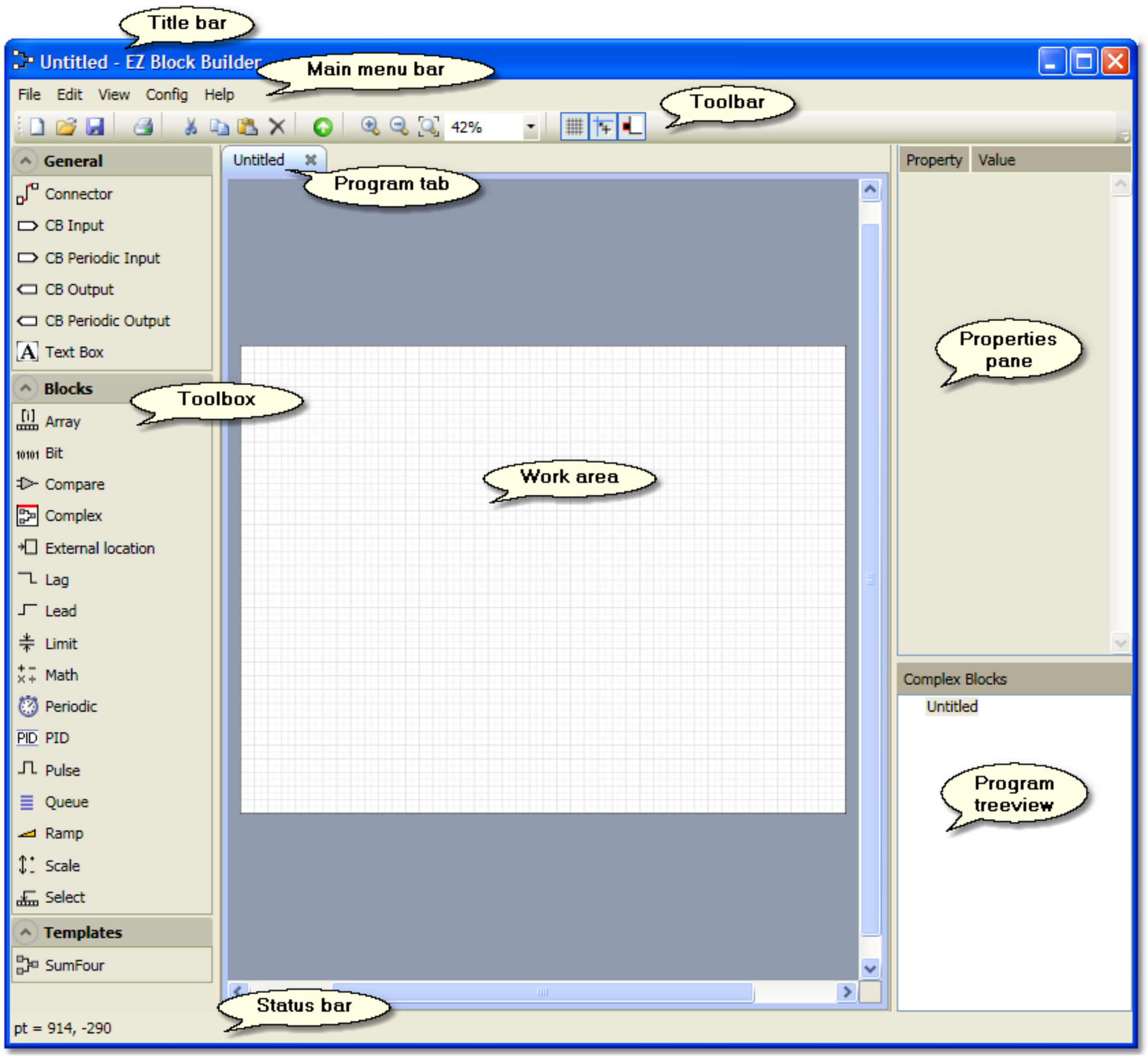
Title bar – Refer to the `Main Form` section for a complete description.

Toolbar – Refer to the `Main Form` section for a complete description.

Work area – Refer to the `Main Form` section for a complete description.

3 Main Form

After the application is launched, the application's main form will appear. The main form serves as the workspace for the tasks that are performed within the EZ Block Builder application.



- The ***title bar*** contains the file name of the currently selected program along with buttons that allow the user to minimize, maximize or close the application.

- The ***main menu bar*** contains a set of text buttons that describe the various actions that may be taken within the program.

- The ***toolbar*** contains a set of graphical icons representing functions that are frequently or commonly used in the application.

- The ***program tab*** displays the name of the program associated with the tab. Multiple programs may be opened within the application and each program is assigned a tab.

- The ***toolbox*** contains the set of program elements that may be dragged onto the work area.

- The ***work area*** is the part of user interface where program elements such as EZ blocks are dropped and arranged. A user arranges relationships between program elements by drawing lines between these elements. The work area consists of a white "page" and a grey outer area. This white page represents a sheet of paper and is the printable area of the program. The size of the work area page may be adjusted by changing the drawing settings as described in the section titled Page Layout and Printing.

- The ***properties pane*** is populated with the properties of the work area's currently selected program element. Once a program element has been selected, a user may configure the properties the individual program element using the properties pane.

- The ***program treeview*** displays the currently selected program's EZ block hierarchy and allows the user to navigate between any complex blocks defined by the program.

- The ***status bar*** displays various context-sensitive notices relating to the current state of the program.

3.1 Main Menu

The main menu bar contains the following four text buttons:

- **File** – Selecting this menu option displays a submenu of options that provide for opening, saving, printing and closing program files as well as saving template files, viewing a list of the most recently opened programs and exiting the application.

- **Edit** – Selecting this menu option displays a submenu of options that define various actions that may be performed in the application's work area such as cut, copy, paste and delete operations.

- **Config** – Selecting this menu option displays a submenu of options that allow the user to configure device communications options, and other user interface and program options. A validate program option available from this submenu allows the user to verify that the currently selected program has connections defined for its required inputs.

- **Help** – Selecting this menu option displays a submenu of options that provide access to an application help file and information about the application's version number.

3.1.1 File submenu

After clicking on the word `File` in the main menu, the file submenu is displayed. The following options are available from the file submenu:

- **New** – This option creates a new, blank EZ Block Builder program named "Untitled" and creates a tab for the new program in the work area.
- **Open…** – This option displays a dialog that allows a user to load a program file from disk.
- **Open from Device…** – This option displays a dialog that allows a user to connect to an X-Series device, select an operations application for upload from the device and load that operations application's program data into the EZ Block Builder.
- **Close** – Close the currently selected program.
- **Close All** – Close all currently open programs.
- **Save** – This option displays a dialog that allows a user to save the currently selected program to disk.
- **Save to Device…** – This option displays a dialog that allows a user to connect to an X-Series device, select a valid location on the device, and download the current program to the selected location on the device.
- **Create Template…** – This option allows the user to save the current complex block hierarchy as a template. The template is then placed in the application's Template directory and is available for use from the "Templates" section of the toolbox the next time the program is started. Refer to the section titled Templates for additional information.
- **Page Setup…** – Allows the user to define the page size used by the application's work area and the size and orientation of the page used by the printer.
- **Print…** – Print the current program using the settings defined by the **`Page Setup`** option.
- **Exit** – When this option is selected, the application exits. It is functionally equivalent to selecting the ☒ button from the title bar.

3.1.2 Edit submenu

After clicking on the word `Edit` in the main menu, the edit submenu is displayed. The following options are available from the edit submenu:

- **Cut** – Remove any currently selected program element(s) from the work area while copying them to the clipboard.
- **Copy** – Copy any currently selected program element(s) to the clipboard.
- **Paste** – Insert any program element(s) present in the clipboard into the program shown in the work area.
- **Delete** – Remove the currently selected program element(s).

3.1.3 Config submenu

After clicking on the word `Config` in the main menu, the configuration submenu is displayed. The following options are available from the configuration submenu:

- **Communications Setup…** – Displays a dialog that allows the user to configure device communication settings such as connection type, com port or IP address, workstation name, etc. Refer to the Communications Configuration section of this document for additional information.

- **Options…** – Displays a dialog that allows the user to configure various global options such as register assignment strategies and profile and alias list preferences.

- **Validate Program** – This submenu option allows the user to verify that the currently selected program has connections defined for its required inputs. Should a problem with the program be detected, a report is generated listing the EZ blocks that need attention.

3.1.4 Help submenu

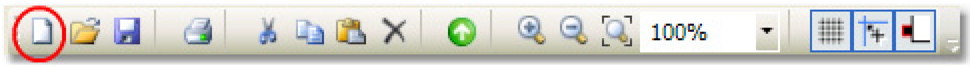
After clicking on the word `Help` in the main menu, the help submenu is displayed. The following options are available from the help submenu:

- **User Guide** – Launches this user guide in a help file viewer.

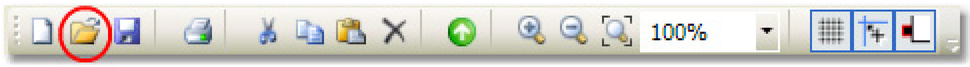
- **About EZ Block Builder** – Displays the application's copyright notice, version number and other build information.

3.2 Toolbar

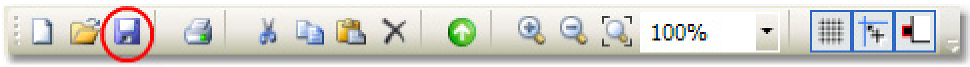
The toolbar appears directly under the main menu and provides one-click access to some commonly used application features. Each feature available on the toolbar is described below:



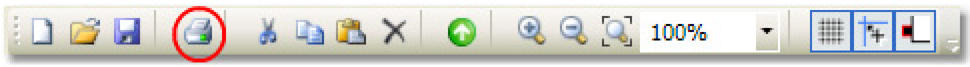
New – This option creates a new, blank EZ Block Builder program named "Untitled" and creates a tab for the new program in the work area.



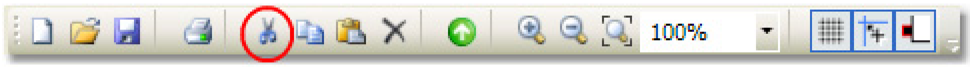
Open – This option displays a dialog that allows a user to load a program file from disk.



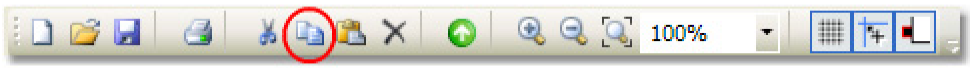
Save – This option displays a dialog that allows a user to save the currently selected program to disk.



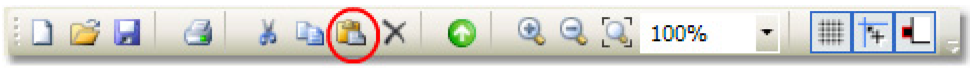
Print – Print the current program using the settings defined by the **`Page Setup`** option.



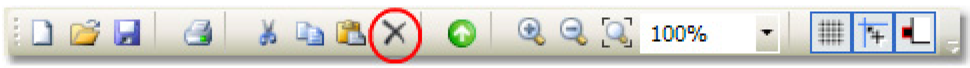
Cut – Remove any currently selected program element(s) from the work area while copying them to the clipboard.



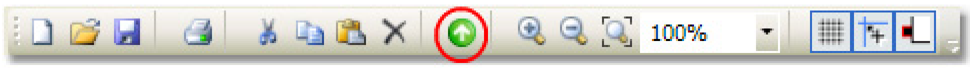
Copy – Copy any currently selected program element(s) to the clipboard.

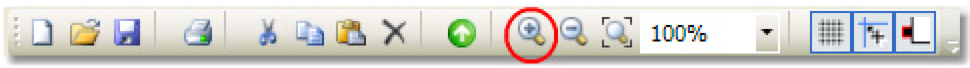


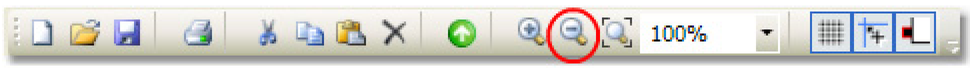
Paste – Insert any program element(s) present in the clipboard into the program shown in the work area.

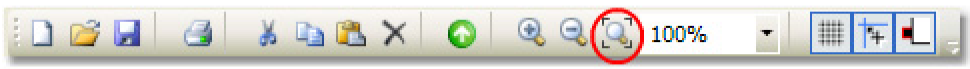


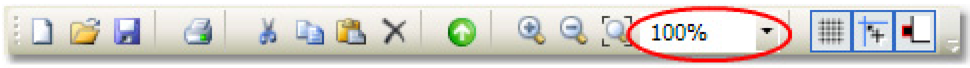
Delete – Remove any currently selected program element(s).

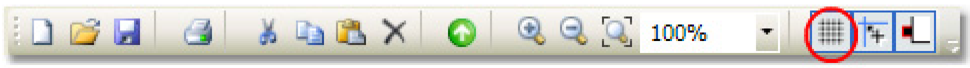

Up – Navigate up through the program treeview. This button is used to quickly navigate to the parent of the currently displayed complex block.

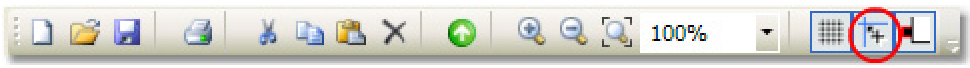

Zoom in – Perform a stepwise increase of the work area zoom level.

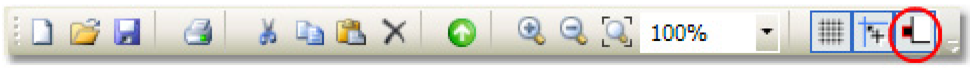

Zoom out – Perform a stepwise decrease of the work area zoom level.


Zoom page – Change the zoom level to a value such that the entire page fits in the work area.


Zoom level – This control can be used to either select a zoom level from the drop down list or to manually enter a desired zoom level.

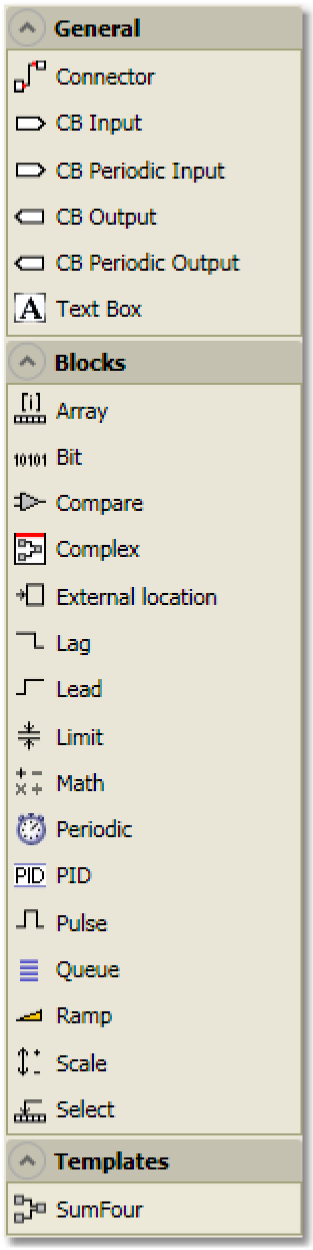

Toggle Grid – Enable/Disable grid lines in the work area.


Toggle Snap – Enable/Disable the work area’s snap-to-grid feature.


Toggle Pins – Enable/Disable the display of pins on EZ blocks.

3.3 *Program elements*

Program elements are the graphical objects that may be placed in the work area and arranged and configured to form a program. This section describes each of the available program elements and provides details about their use.



The application's toolbox contains all of the program elements that a user can drag onto a work area to help define a program. To drag a program element from the toolbox to the work area, left-click on a toolbox item and position the mouse on the work area at the location where you wish to place the element. Release the left mouse button to drop the element on the work area.

The toolbox contains three different collections of program elements titled `General`, `Blocks`, and `Templates`.

The `General` collection contains the Connector element, four special elements used to define the input and output pins of Complex blocks, and a Text Box element.

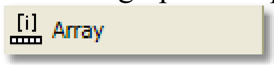
There are 16 blocks provided in the toolbox's `Blocks` collection. The `Complex block` is a special kind of program element that is used to host other EZ blocks in a program. The `External location` block is another special type of EZ Block that is used to represent a discrete register address in the device. These external locations may point to any register location in the device, even if the location is external to the current operations app. The remaining 14 EZ blocks correspond to actual functions available in X-Series devices.

The `Templates` collection contains any EZ Block Builder templates that have defined and saved to the application's Template directory. These template files are read once when the EZ Block Builder application begins but are available for use during design-time just like any other program element. Dragging and dropping a template from the toolbox to the work area results in a 'pre-wired' Complex block being created and placed on the work area.

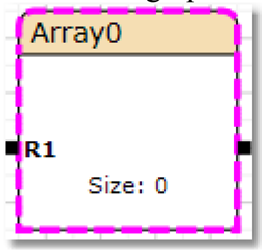
3.3.1 Array

The Operation property of this EZ block defines the calculation to be performed on a user-defined array. The result of this calculation is the output value for this block.

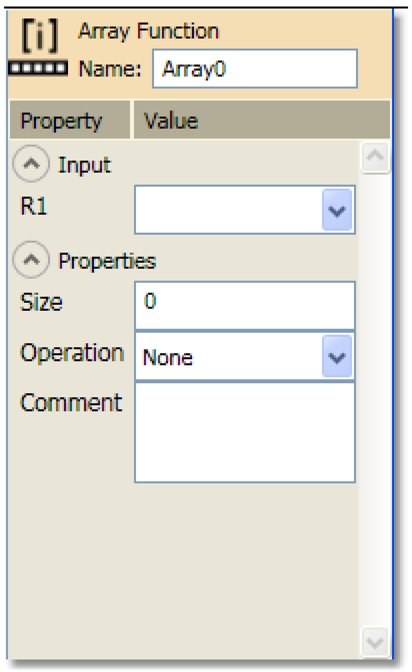
Toolbox graphical representation:



Work area graphical representation:



Properties pane graphical representation:



Name: Array0

Property	Value
R1	
Size	0
Operation	None
Comment	

R1 – The register value of the input assigned to *R1* becomes the starting location of the array.

The input value for this block may be set using multiple methods. From the property pane, the input value may be set by choosing the output of an EZ block from among those listed in the drop-down box. From the work area, the input value may be set by connecting the output of a source block to the input pin.

Size – The number of consecutive registers to be operated on by the selected array operation.

Operation – Specifies the operation to be performed on the Array specified by the register value of the *R1* input and the array size as specified in the *Size* property.

Available operations are:

- ***None*** – Select for no operation.
- ***Sum*** – Provides the *Sum* of the values in the array specified by the register value of the *R1* input and the array size as specified in the *Size* property.
- ***Mean*** – Provides the *Mean Average* of the values in the array specified by the register value of the *R1* input and the array size as specified in the *Size* property.

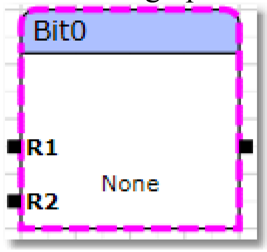
- ***Sqrt Mean*** – Provides the *Square Root Mean Average* of the values in the array specified by the register value of the *R1* input and the array size as specified in the *Size* property.
- ***Min*** – Provides the *Minimum* value in the array specified by the register value of the *R1* input and the array size as specified in the *Size* property.
- ***Max*** – Provides the *Maximum* value in the array specified by the register value of the *R1* input and the array size as specified in the *Size* property.

3.3.2 Bit

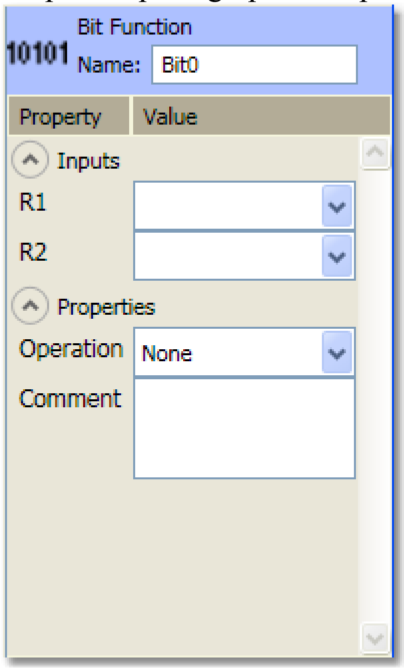
The output of this EZ Block is the value resulting from the computation defined by the Operation property.

Toolbox graphical representation:

Work area graphical representation:



Properties pane graphical representation:



Name: Bit0

R1:

R2:

Operation: None

R1, R2 – The input values for this block may be set using multiple methods. From the property pane, an input value may be set by choosing the output of an EZ block from among those listed in the drop-down box. From the work area, an input value may be set by connecting the output of a source block to an input pin.

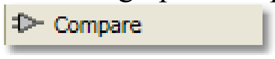
Operation – Specified the operation to be performed on the values provided to inputs *R1* and *R2*.
Available operations are:

- ***R1 & R2*** – Perform a bitwise AND operation on the values provided to inputs *R1* and *R2*.
- ***R1* | *R2*** – Perform a bitwise OR operation on the values provided to inputs *R1* and *R2*.
- ***R1* ^| *R2*** – Perform a bitwise exclusive-OR (XOR) operation on the values provided to inputs *R1* and *R2*.
- ***R1* << *R2*** – The value provided to input *R1* is shifted left the number of spaces specified by the value provided to input *R2*.
- ***R1* >> *R2*** – The value provided to input *R1* is shifted right the number of spaces specified by the value provided to input *R2*.
- **~R1** – Performs a bitwise complement on the value provided to input *R1* (reverses the state of each bit).
- ***! R1*** – Performs the Not function on the value provided to input *R1*. Converts zero to one and non-zero to a zero.

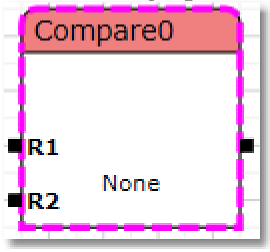
3.3.3 Compare

This operation outputs a value of 1 (true) or 0 (false) based on the comparison operation selected.

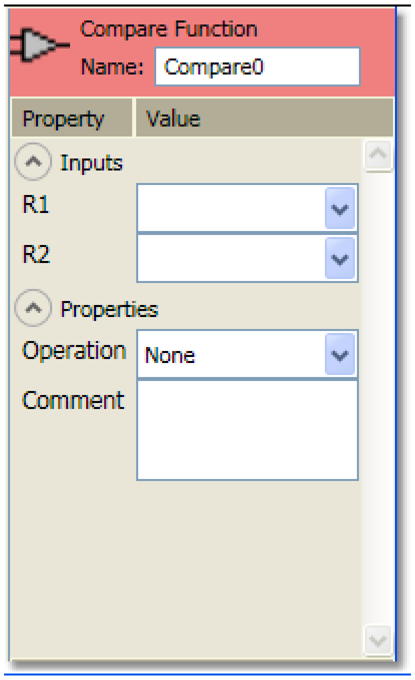
Toolbox graphical representation:



Work area graphical representation:



Properties pane graphical representation:


Compare Function
Name: Compare0

Property	Value
Inputs	
R1	
R2	
Properties	
Operation	None
Comment	

R1, R2 – The input values for this block may be set using multiple methods. From the property pane, an input value may be set by choosing the output of an EZ block from among those listed in the drop-down box. From the work area, an input value may be set by connecting the output of a source block to an input pin.

Operation – Specifies the operation to be performed on the values provided to inputs R1 and R2.

Available operations are:

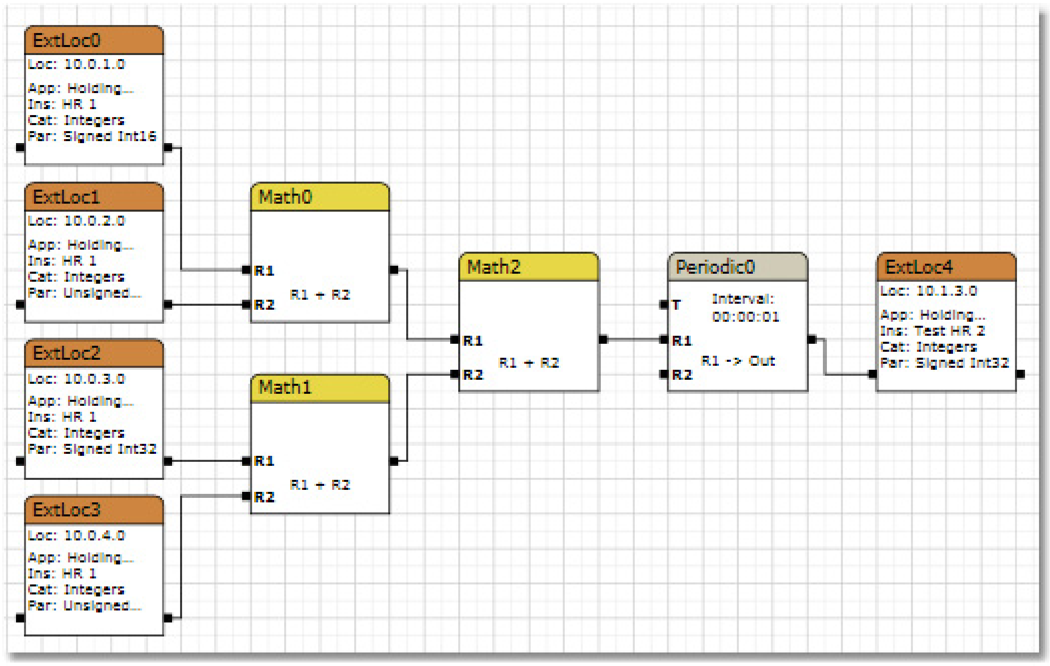
- ***R1 EQ R2*** – The value in R1 is *equal* to the value in R2.
- ***R1 NE R2*** – The value in R1 is *not equal* to the value in R2.
- ***R1 GT R2*** – The value in R1 is *greater than* the value in R2.
- ***R1 GE R2*** – The value in R1 is *greater than or equal to* the value in R2.
- ***R1 LT R2*** – The value in R1 is *less than* the value in R2.
- ***R1 LE R2*** – The value in R1 is *less than or equal to* the value in R2.
- ***(R1 & R2) EQ 0*** – The result of performing an AND operation on the values present in R1 and R2 is *equal to* zero.
- ***(R1 & R2) NE 0*** – The result of performing an AND operation on the values present in R1 and R2 is *not equal to* zero.

- ***(R1 | R2) EQ 0*** – The result of performing an OR operation on the values present in R1 and R2 is *equal to* zero.
- ***(R1 | R2) NE 0*** – The result of performing an OR operation on the values present in R1 and R2 is *not equal to* zero.
- ***(R1 ^| R2) EQ 0*** – The result of performing an exclusive-OR (XOR) operation on the values present in R1 and R2 is *equal to* zero.
- ***(R1 ^| R2) NE 0*** – The result of performing an exclusive-OR (XOR) operation on the values present in R1 and R2 is *not equal to* zero.

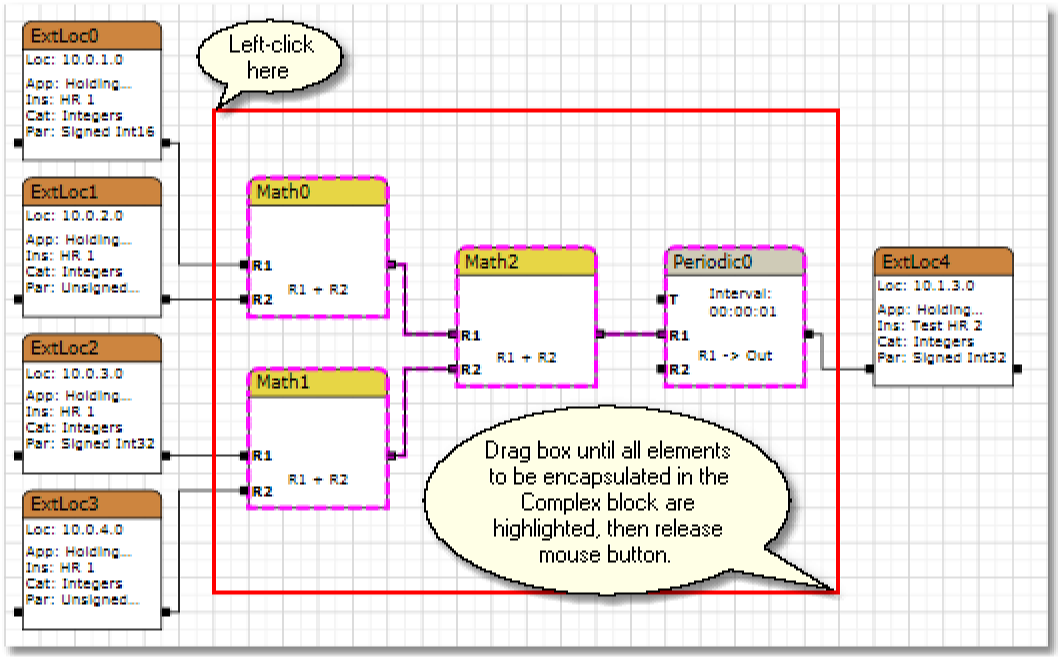
3.3.4 Complex Block

A Complex block is a program element that is used to encapsulate other program elements. Complex blocks do not exist inside X-Series devices; their function is to provide a mechanism to help clean up the visual presentation of large EZ Block Builder programs. Large or common operations may be visualized as a single complex block with multiple inputs and outputs.

An example will be used to explain Complex block creation and usage. In the example image shown below three Math blocks are used to sum four inputs and the output is read by a Periodic function block once per second.

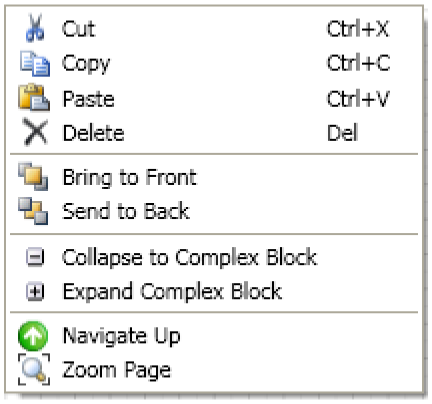


To convert the three Math blocks and the Periodic to a Complex block, left-click in the work area and drag a box around all four program elements and their connecting lines as shown in the image below.

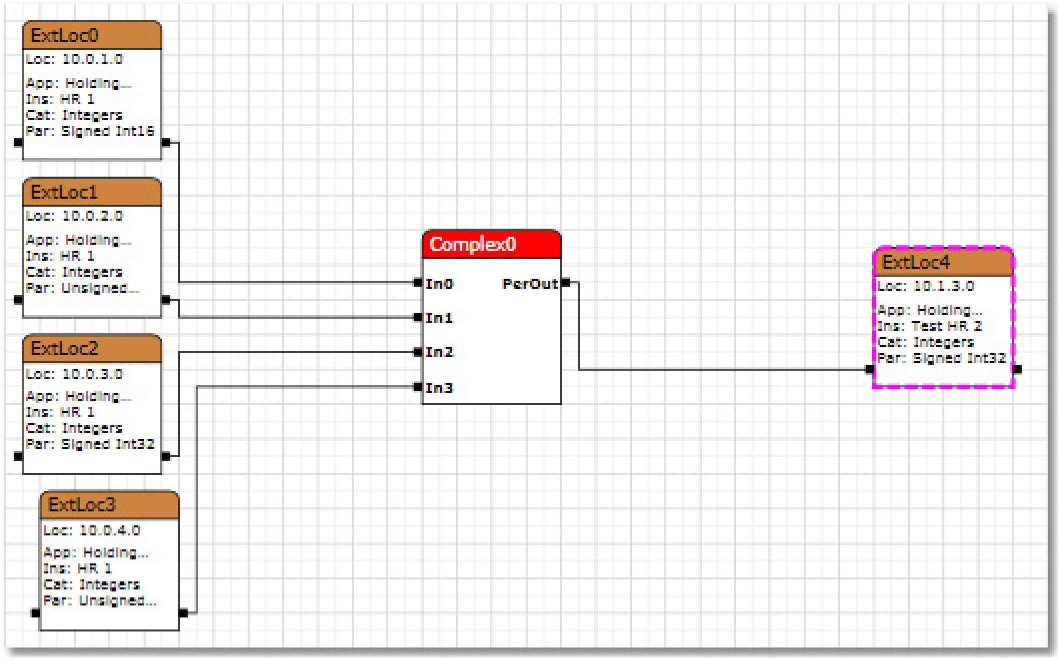


All currently selected items will be outlined with a pink dashed line. In the example shown above we've selected all three Math blocks, the Periodic block, and the three lines that connect them.

To turn this selection into a Complex block, right click on one of the selected blocks to bring up the context menu shown below.

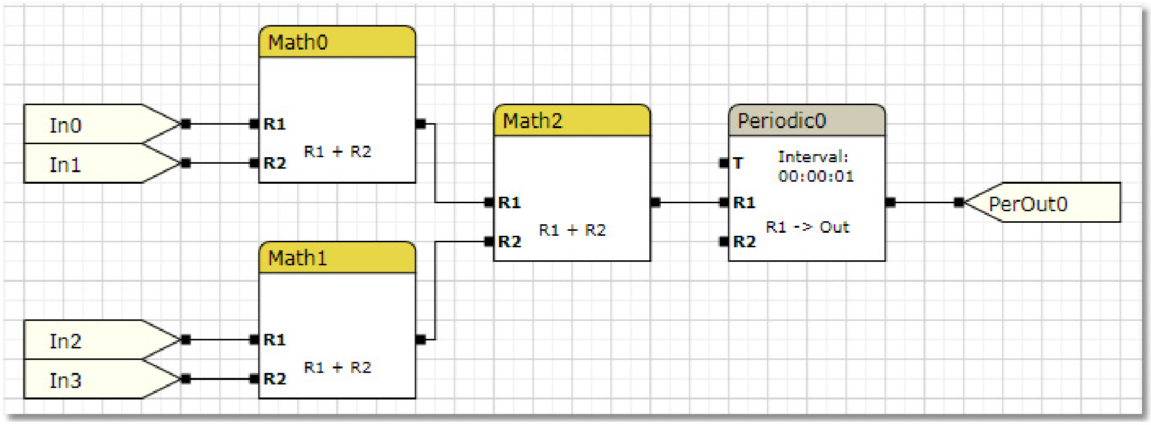


Select "Collapse to Complex Block". The selected items will be placed in a Complex block and the work area will look similar to the following image:



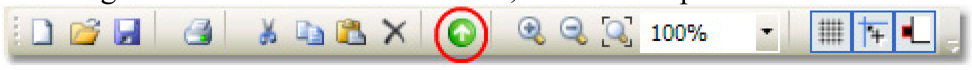
The program in the image shown above has exactly the same functionality as the original program but the visual representation of the functionality provided by the Math blocks and Periodic block has been replaced with a single block with four inputs and one output.

Double-click on the newly created Complex block to drill down into the Complex block and examine its contents.

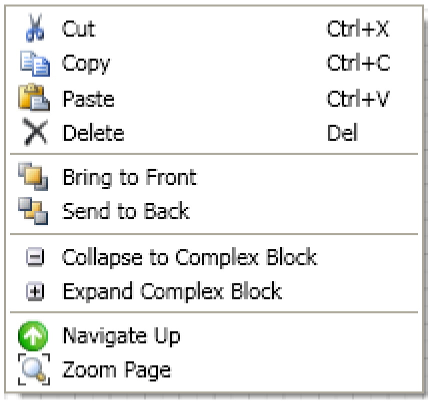


Although the function of the program is the same, a few new program elements have been automatically added inside the Complex block. In0, In1, In2, and In3 represent the four input pins of the Complex block and PerOut0 represents the sole output. These elements are described in detail in the section titled Complex Block Inputs and Outputs but their basic function is to provide the external connection points for Complex blocks.

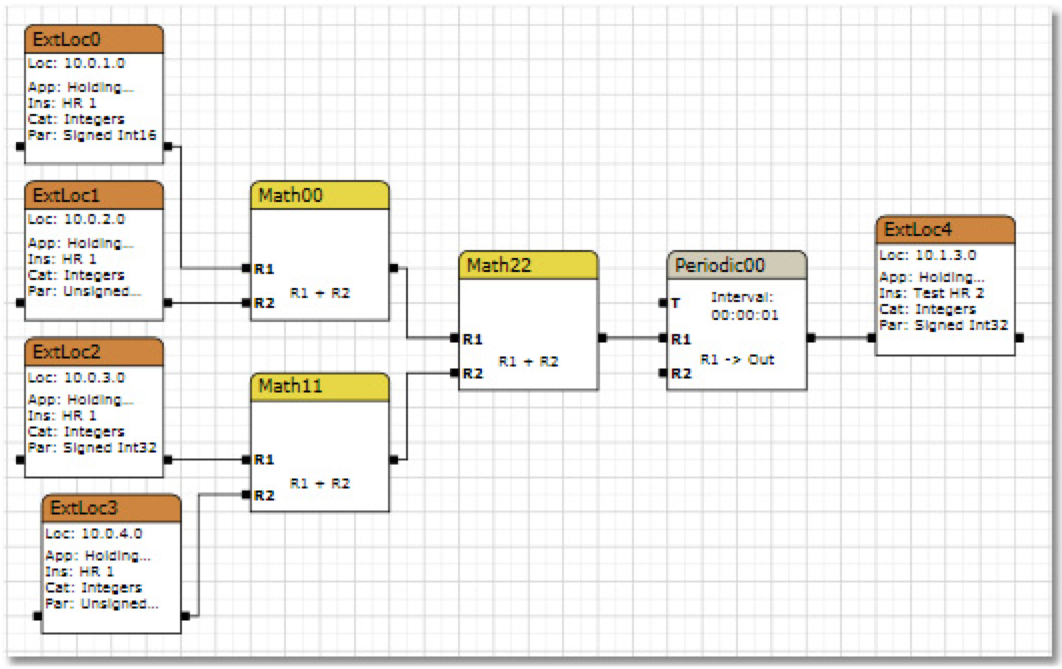
To navigate back to the main work area, click the "Up" button in the main toolbar.



A user can also expand a Complex block. To do so, select the Complex block that was just created and right click it to bring up the context menu.



Select "Expand Complex Block". The Complex block image will be replaced with the program elements contained within the block as shown below:



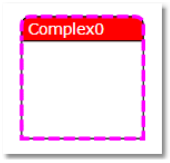
Note: If desired, a blank Complex block can be dragged from the toolbox and populated manually, although the method for creating a Complex block described above will likely be much faster. The procedure for manually creating Complex blocks is explained along with CB Inputs and CB Outputs in the section titled Complex Block Inputs and Outputs.

3.3.5 Complex Block Inputs and Outputs

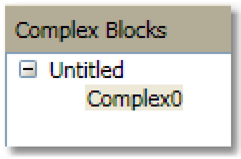
There are four program elements that may be used to define an input or output pin for a complex block. These four elements reside in the "General" collection of the toolbar and are named: *CB Input*, *CB Periodic Input*, *CB Output*, and *CB Periodic Output*. Each of these four elements serve as a proxies for the pins to which they are connected and allow users to define Complex Blocks (CBs) that have any number of inputs and outputs. These program elements bridge the divide between program elements that wish to use a CB and program elements that comprise the internal workings of the CB.

The following example will illustrate the usage of each of these elements. In this example we will go through the process of creating a CB from scratch, populating it with complex I/O elements and then connecting the pins of the CB with other program elements.

1. To create a CB, drag a *Complex* element from the toolbox and drop it on the work area. The CB will receive the default name "Complex0". Notice that there are no input or output pins on this CB.

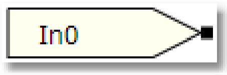


2. Double-click on the CB to navigate inside the block. The work area will appear blank but the *Program treeview* in the bottom right section of the *Main Form* will show that you have navigated inside the CB named "Complex0".



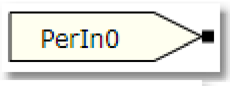
3. Populate the Complex block with Lead, Periodic and External Location program elements.

4. Place a CB Input element next to the input of the Lead block.



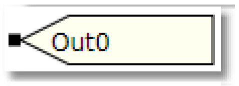
5. CB Inputs are used to provide value data to the blocks that reside inside the complex block. Both the Lead and Periodic blocks accept value data, so connect the pin of the CB Input element to the R1 inputs of both the Lead and Periodic blocks.

6. Place a CB Periodic Input element next to the input of the External location program element.



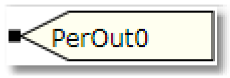
7. CB Periodic Inputs are used whenever it is desired to accept the output pin of a Periodic as an input to the CB. Whereas regular CB Inputs may provide data to many inputs, the CB Periodic Input (like the Periodic output pin it mirrors) must be assigned a single location where the output value of the Periodic will be written. For this example, connect the pin of the CB Periodic Input to the External Location program element.

8. Place a CB Output element next to the output of the Lead block.



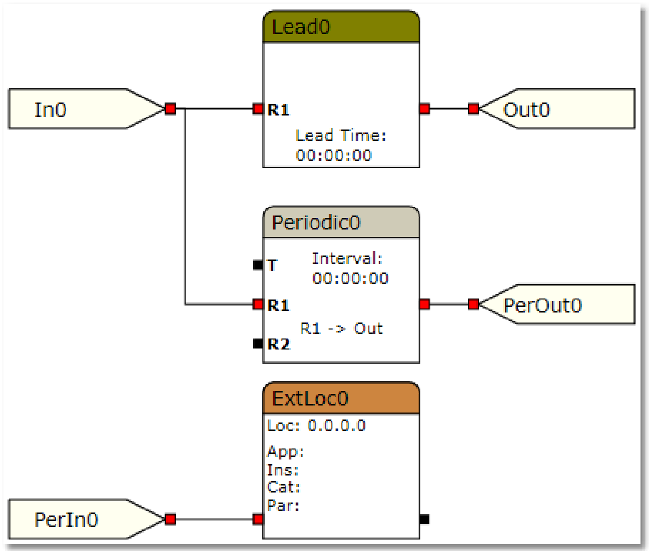
9. CB Outputs are used to source value data to EZ blocks outside the complex block. In this example, we wish to provide the output of the Lead block on an output pin of the CB so connect the Lead block's pin to the CB Output.

10. Place a CB Periodic Output element next to the output of the Periodic block.

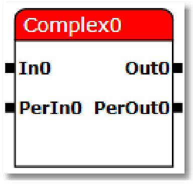


11. CB Periodic Output is used whenever it is desired to provide the output pin of a Periodic as an output pin of the CB. Periodic blocks cannot source their outputs using regular CB Outputs; instead, they must use the CB Periodic outputs. Connect the output pin of the Periodic to the CB Periodic Output to complete the example CB.

12. At this point, the example CB should have a structure similar to the following diagram:



13. Click the Up button on the toolbar to navigate back to the CB's parent and the CB should now look like the image shown to the right. Note that the CB now has 2 input pins and 2 output pins. These pins have the names of the Complex I/O elements that were assigned inside the CB.



14. The CB encapsulates the functionality shown in the image in step 12 but it represents that functionality to its parent as the CB shown in step 13. These newly defined input and output pins may now be connected to appropriate program elements external to the CB. "In0" is a CB Input and will accept a connection from program elements that provide value data such as non-Periodic

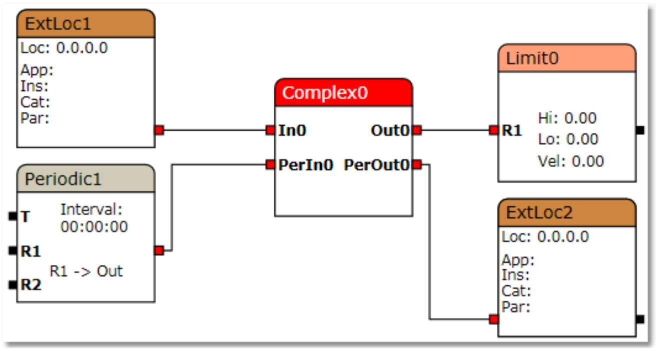
blocks and External locations. For this example, drag an External location onto the work area and connect it to the "In0" pin of the CB.

15. "PerIn0" is a CB Periodic Input pin and it will only accept connections from Periodic blocks. Drag a Periodic block onto the work area and connect its output to the "PerIn0" pin of the CB.

16. "Out0" is a CB Output and it sources value data so drag a Limit block onto the work area and connect the "Out0" pin of the CB to the R1 input of the Limit block. (Note: In this example, the Limit block is being used to represent an EZ block that accepts value data. Several different kinds of EZ blocks could have been used instead.)

17. "PerOut0" is a CB Periodic Output and it mirrors the output pin of a Periodic block located inside the CB. Drag an External Location program element (which is a suitable target for a Periodic output pin) onto the work area and connect the "PerOut0" of the CB to the External Location.

18. At this point the work area should look similar to the picture shown to the right. All Complex inputs and outputs have now been assigned. These Complex I/O elements simply mirror the pins to which they are connected and provide a means for program elements residing inside a CB to communicate with program elements residing outside the CB.

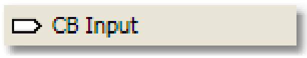


3.3.5.1CB Input

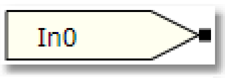
CB Inputs are used to provide value inputs to Complex blocks. EZ blocks external to a CB may provide value data to a CB Input. The CB Input, in turn, provides that value data to EZ blocks internal to the CB.

Refer to the section titled Complex Block Inputs and Outputs for an example of how to use this program element.

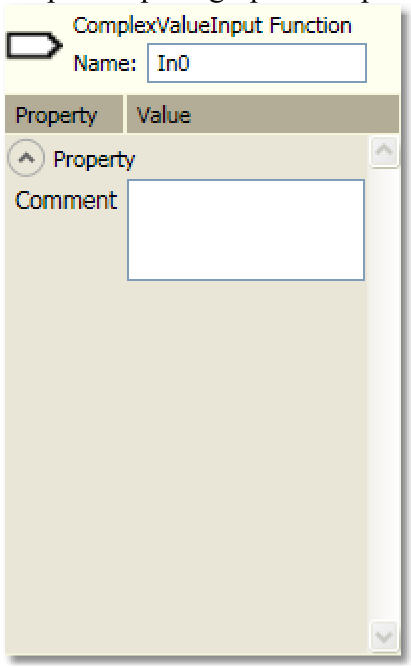
Toolbox graphical representation:



Work area graphical representation (internal to a Complex block):



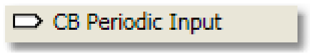
Properties pane graphical representation:

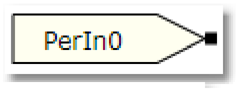


3.3.5.2CB Periodic Input

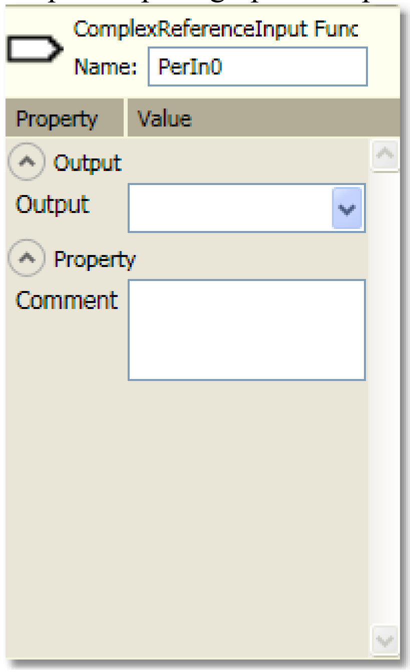
CB Periodic Inputs are used as connection points between Periodic blocks external to the host Complex block and program elements internal to the host Complex block.

Refer to the section titled Complex Block Inputs and Outputs for an example of how to use this program element.

Toolbox graphical representation: 

Work area graphical representation (internal to a Complex block): 

Properties pane graphical representation:



ComplexReferenceInput Func
Name: PerIn0

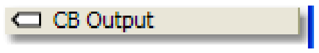
Property	Value
Output	
Output	
Property	
Comment	

3.3.5.3CB Output

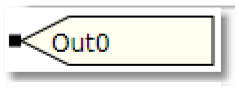
CB Outputs are used to source value outputs from Complex blocks. EZ blocks external to a CB may consume the value data provided by CB Output. The CB Output provides the output value of a (non-Periodic) block internal to the CB.

Refer to the section titled Complex Block Inputs and Outputs for an example of how to use this program element.

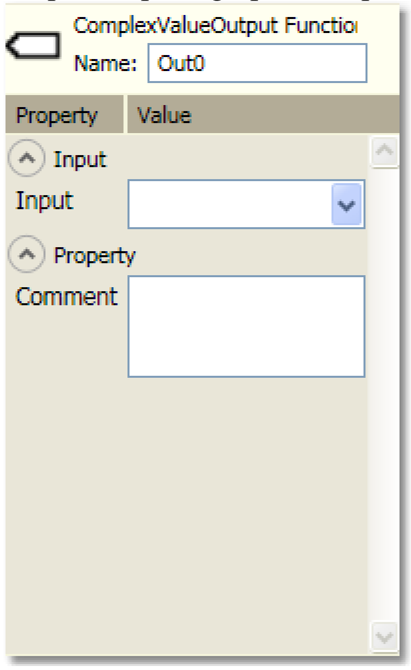
Toolbox graphical representation:



Work area graphical representation (internal to a Complex block):



Properties pane graphical representation:



ComplexValueOutput Functio

Name: Out0

Property	Value
Input	
Input	
Property	
Comment	

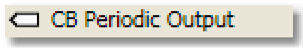
3.3.5.4CB Periodic Output

CB Periodic Outputs are used to source the output pin of a Periodic block as an output pin of a CB. Periodic blocks cannot source their outputs using regular CB Outputs; instead, they must use the CB Periodic Outputs.

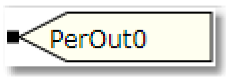
A program element external to a CB that is valid target of a Periodic output (such as an External Location) may connect to a CB's Periodic Output pin to receive the data sourced by a Periodic block internal to the CB.

Refer to the section titled Complex Block Inputs and Outputs for an example of how to use this program element.

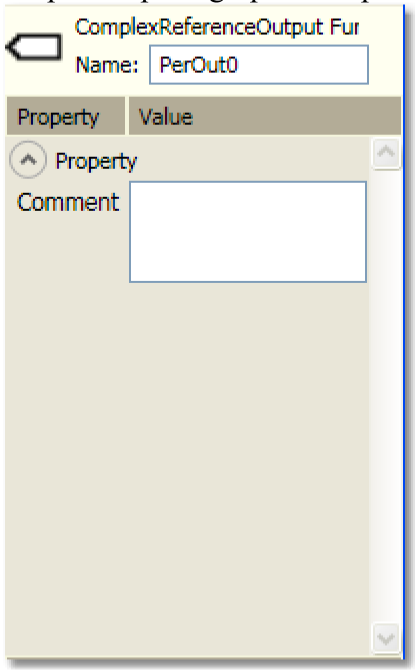
Toolbox graphical representation:



Work area graphical representation (internal to a Complex block):



Properties pane graphical representation:


ComplexReferenceOutput Fur

Name: PerOut0

Property	Value
Property	
Comment	

3.3.6 Connector

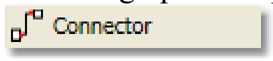
The Connector program element is used to establish a relationship between two other program elements. The square on either end of the connector is called a pin. These pins can be placed on top of the pins of other program elements to establish a connection.

For example, assume a user wishes to assign the input of Scale block to the output of a Math block. To accomplish this using a Connector program element, the user would drag a connector onto the work area, dropping the connector so that the Connector's left pin is on top of the Math block's output pin. The user could then click on the Connector's rightmost pin and drag it onto the R1 input pin of the Scale block.

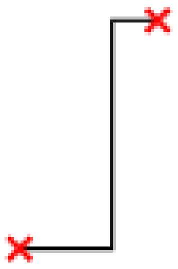
A connector will not attach to any pin when doing so would create an invalid connection such as an input-to-input, output-to-output, or circular reference condition.

Connectors will attempt to automatically route themselves around EZ blocks in the work area as layout of the program elements in the work area are rearranged.

Toolbox graphical representation:



Work area graphical representation:



Properties pane graphical representation:
None

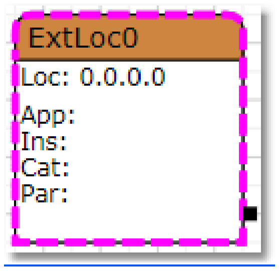
3.3.7 External Location

The `External Location` block is a program element that is used to represent a discrete register address in the device. These external locations may point to any register location in the device, even if the location is external to the current operations app.

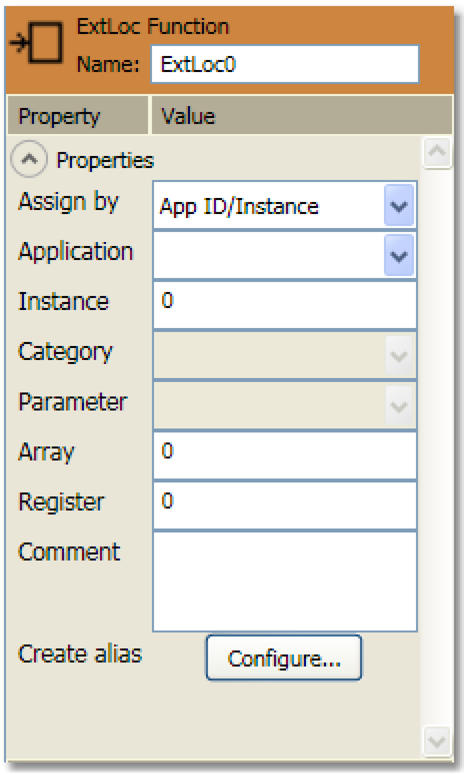
A user may define an external location's address in one of two ways: manually specifying the numeric register location values individually or as text values through the use of Alias Lists and Profiles.

Toolbox graphical representation:

Work area graphical representation:


ExtLoc0
Loc: 0.0.0.0
App:
Ins:
Cat:
Par:

Properties pane graphical representation:


ExtLoc Function
Name: ExtLoc0

Property	Value
Properties	
Assign by	App ID/Instance
Application	
Instance	0
Category	
Parameter	
Array	0
Register	0
Comment	
Create alias	Configure...

Assign by – Select the method that will be used to assign the address for this external location.

Available operations are:

- ***App ID/Instance*** – Allows the manual entry of numeric values for the Array and Register fields. Disables the Category and Parameter combo boxes and enables the Array and Register fields and the `Configure…` button.
- ***Alias*** – Allows the selection of alias entries for the register address. Enables the Category and Parameter combo boxes and disables the Array and Register fields and the `Configure…` button.

Application – The available selections and the application ids that they map to are listed below:

IO Subsystem - 1
Communications - 3
AGA3 - 4
AGA7 - 5
Trend - 7
Alarms - 8
Valve Control - 9
Holding Registers - 10
Therms Master - 11
XMV - 12
IEC61131 - 13
Tank - 15
Pump - 16
Operations - 18
SUAGA3 - 20
SUAGA7 - 21
Ultrasonic - 22
VCONE - 23
SUConvert - 26
EnronAGA3 - 27
EnronAGA7 - 28
Plunger - 56
Wireless IO - 57
Coriolis - 58
Safety System - 59
PAD Controller - 62
Spare - 255

These application ids are defined inside the AppTypes.xml that resides in the EZ Block Builder's installation directory. Refer to the section titled File Structure for more information.

Instance – A device may contain multiple applications that share the same type. The instance field defines to which of these applications the external location refers.

When the "Assign By" selection is set to *App ID/Instance*, the instance property should be set to the zero-based instance number of the desired application. This means that the 1st instance of an application will have a value of 0. For example, is a user wishes for an external location to refer to the 2nd instance of a "Tank" application, the user would select "Tank" as the `Application` and enter an `Instance` value of 1.

When the "Assign By" selection is set to *Alias*, the instance property is represented as a combo box and the user should make the appropriate instance selection from the provided selections. The available selections are defined in the user-specified profile described in the Profiles section of this document.

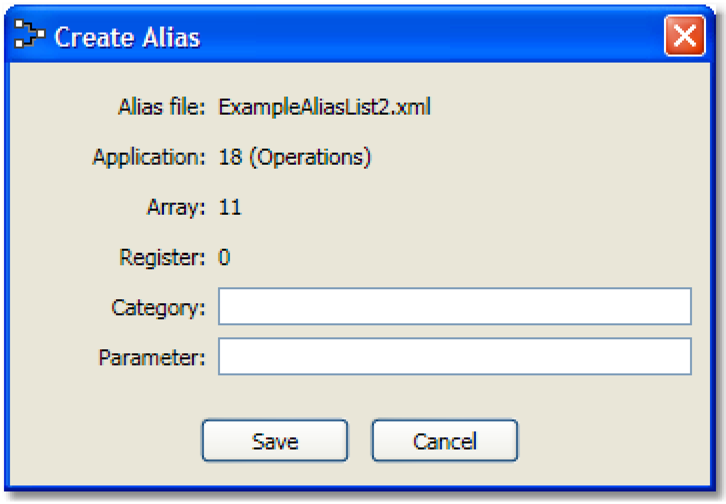
Category – This field is only enabled when the user has set the "Assign By" selection to *Alias*. If no categories are displayed, make sure an alias list has been selected via the options menu. The Category/Parameter pair map to an array/register setting as defined in the selected alias file.

Parameter – This field is only enabled when the user has set the "Assign By" selection to *Alias*. The field values displayed here depend on the Category value so the user will want to select the Parameters field after choosing the Category. The Category/Parameter pair map to an array/register setting as defined in the selected alias file.

Array – This field is only enabled when the user has set the "Assign By" selection to *App ID/Instance.* The numeric value entered here is the array value for the external location.

Register – This field is only enabled when the user has set the "Assign By" selection to *App ID/Instance.* The numeric value entered here is the register value for the external location.

Create Alias – This field is only enabled when the user has set the "Assign By" selection to *App ID/Instance.* Clicking this button displays the dialog show below. Using this dialog, a user may define a Category and Parameter name for the given array/register pair. Selecting "Save" from this dialog would save the Category and Parameter names to the current alias file.


Create Alias

Alias file: ExampleAliasList2.xml

Application: 18 (Operations)

Array: 11

Register: 0

Category:

Parameter:

Refer to the section titled Alias Lists for additional information.

3.3.8 Lag

The output value for the lag block is computed as follows:

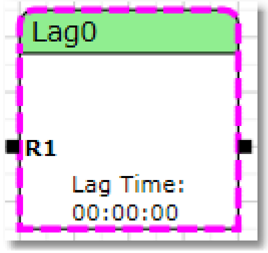
Output Value = Previous Output + (Sample Interval / Lag Time) * (Current Input - Previous Output)

Where:

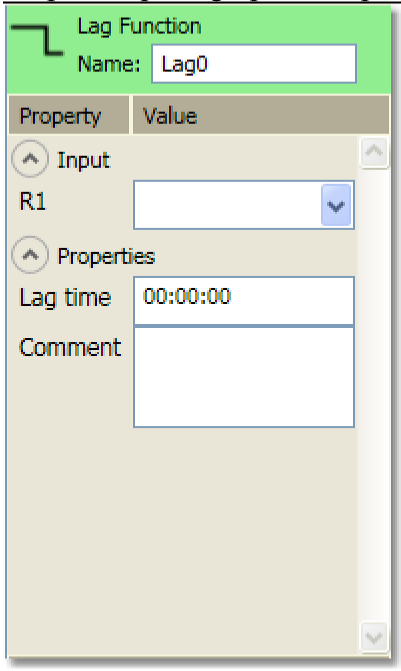
- **Output Value** = Output value calculated by the Lag algorithm.
- **Previous Output** = Previous output value from the Lag algorithm.
- **Sample Interval** = Frequency that the source or input register is read.
- **Lag Time** = Time constant as specified by the *Lag Time* property. The *Lag Time* must be greater than the Sample Interval.
- **Current Input** = The value provided to input *R1*.

Toolbox graphical representation:

Work area graphical representation:



Properties pane graphical representation:


Lag Function
Name: Lag0

Property	Value
Input	
R1	
Properties	
Lag time	00:00:00
Comment	

R1 – Specify the data source for the Lag block.

The input value for this block may be set using multiple methods. From the property pane, the input value may be set by choosing the output of an EZ block from among those listed in the drop-down box. From the work area, the input value may be set by connecting the output of a source block to the input pin.

Lag Time – Enter a lag time value in the format HH:MM:SS

3.3.9 Lead

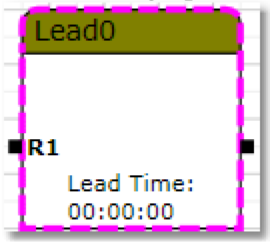
The output value for the lag block is computed as follows:

Output Value = Current Input + (Lead Time / Sample Interval) * (Current Input – Previous Input)

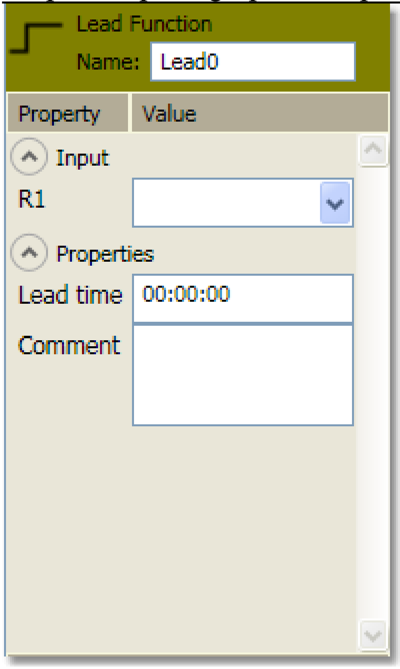
Where:
- **Output Value** = Output value calculated by the Lead algorithm.
- **Current Input** = The value provided to input *R1*.
- **Lead Time** = Time constant as specified in the *Lead Time* property. The *Lead Time* must be greater than the Sample Interval.
- **Sample Interval** = Frequency that the source or input register is read.
- **Previous Input** = Previous input value to the Lead algorithm.

Toolbox graphical representation:

Work area graphical representation:



Properties pane graphical representation:



R1 – Specify the data source for the Lead block.

The input value for this block may be set using multiple methods. From the property pane, the input value may be set by choosing the output of an EZ block from among those listed in the drop-down box. From the work area, the input value may be set by connecting the output of a source block to the input pin.

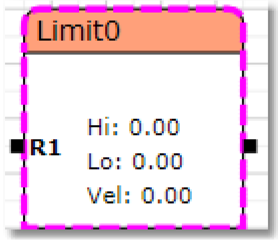
Lead Time – Enter a lead time value in the format HH:MM:SS

3.3.10 Limit

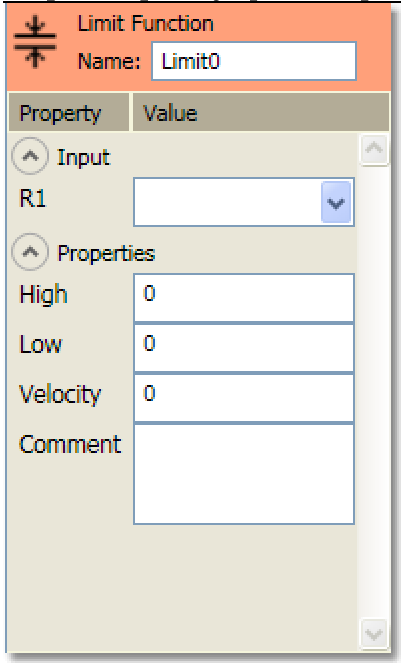
The Limit block limits the value coming from the register specified by the *R1* input to the constraints of the *Low* and *High* properties. The rate of change is limited by the *Velocity* property.

Toolbox graphical representation:

Work area graphical representation:



Properties pane graphical representation:



Property	Value
Name	Limit0
Input	
R1	
Properties	
High	0
Low	0
Velocity	0
Comment	

R1 – Specify the data source for the Limit block.

The input value for this block may be set using multiple methods. From the property pane, the input value may be set by choosing the output of an EZ block from among those listed in the drop-down box. From the work area, the input value may be set by connecting the output of a source block to the input pin.

High – Specify the High limit value for the input data. The output value will not be allowed to go above this value.

Low – Specify the low limit value for the input data. The output value will not be allowed to go below this value.

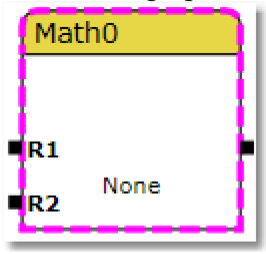
Velocity – Specify the value in engineering units per second that the output is allowed to change in response to the input. If no velocity is specified, the output value will change with the input value.

3.3.11 Math

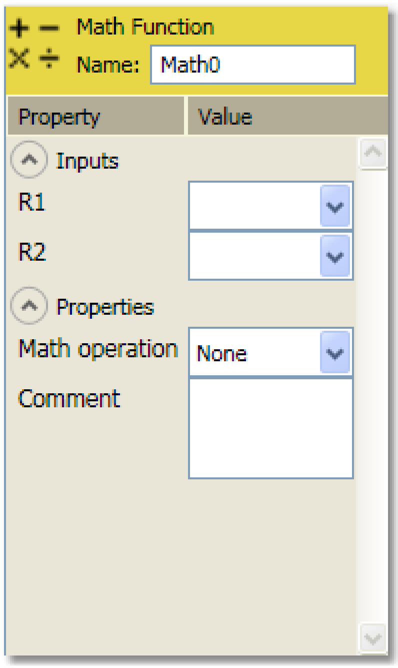
The output of this EZ block is the value resulting from performing the selected math operation on the block's inputs.

Toolbox graphical representation:

Work area graphical representation:



Properties pane graphical representation:



Name: Math0

Property	Value
R1	
R2	
Math operation	None
Comment	

R1, R2 – The input values for this block may be set using multiple methods. From the property pane, an input value may be set by choosing the output of an EZ block from among those listed in the drop-down box. From the work area, an input value may be set by connecting the output of a source block to an input pin.

Math operation – Specifies the operation to be performed on the values provided to inputs *R1* and/or *R2*.

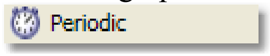
Available operations are:

- ***R1 + R2*** – The value in *R2* is added to the value in *R1*.
- ***R1 – R2*** – The value in *R2* is subtracted from the value in *R1*.
- ***R1 * R2*** – The value in *R1* is multiplied by the value in *R2*.
- ***R1 / R2*** – The value in *R1* is divided by the value in *R2*.
- ***R1 % R2*** – Produces the remainder of *R1* integer divided by *R2* integer. (Modulus operator)
- ***R1 ^R2*** – The value in *R1* is raised to the power of the value in *R2*.
- ***1 / R1*** – The value 1 is divided by the value in *R1*.
- ***SQRT (R1)*** – The square root is taken of the value in *R1*.
- ***ABS (R1)*** – Converts the value in R1 to an absolute number (removes sign).

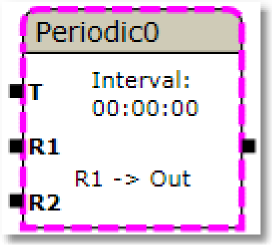
3.3.12 Periodic

The output value of this EZ Block is determined by the selected Operation type. The frequency at which this output value is updated is determined by the selected Trigger mode.

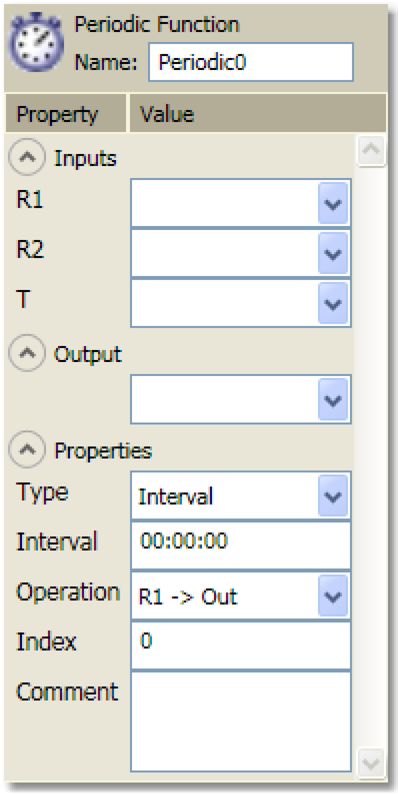
Toolbox graphical representation:



Work area graphical representation:



Properties pane graphical representation:



Periodic Function
Name: Periodic0

Property	Value
Inputs	
R1	
R2	
T	
Output	
Properties	
Type	Interval
Interval	00:00:00
Operation	R1 -> Out
Index	0
Comment	

R1, R2, T – The input values for this block may be set using multiple methods. From the property pane, an input value may be set by choosing the output of an EZ block from among those listed in the drop-down box. From the work area, an input value may be set by connecting the output of a source block to an input pin.

Output – The output location for this block may be set inside the property pane by choosing the input of an external location or CB Periodic Input from among those listed in the drop-down box. The input values may be also be set in the work area by connecting the output of a source block to an external location input pin or a CB Periodic Input pin.

Type – Select among the following options:

- ***Interval*** – Specifies that the operation will occur at the frequency specified by the *Interval* property
- ***Time*** – Specifies that the operation will occur at the time of day specified by the *Interval* property.
- ***Triggered*** – Specifies that the operation will occur when the value provided to input *T* is non-zero.

Interval – The interval time is used when the periodic is configured to be of type ***Interval*** or ***Time***. Enter the interval value in the format HH:MM:SS.

Operation – Specifies the operation to be performed using the values provided to inputs R1 and R2 with the result placed in the register as specified in the *Output* property.

Available operations are:

- ***R1 -> Out*** – Output the contents of *R1* to the register specified in the Output property.
- ***R1 -> R2 -> Out*** – Output the contents of *R1* to *R2* and to the register specified in the *Output* property.
- ***R1 + R2*** – The value in *R2* is added to the value in *R1*.
- ***R1 – R2*** – The value in *R2* is subtracted from the value in *R1*.
- ***R1 * R2*** – The value in *R1* is multiplied by the value in *R2*.
- ***R1 / R2*** – The value in *R1* is divided by the value in *R2*.
- ***R1 % R2*** – Produces the remainder of *R1* integer divided by *R2* integer. (Modulus operator)
- ***R1 ^R2*** – The value in *R1* is raised to the value in *R2*.
- ***1 / R1*** – The value 1 is divided by the value in *R1*.
- ***SQRT (R1)*** – The square root is taken of the value in *R1*.
- ***ABS (R1)*** – Converts the value in *R1* to an Absolute number.
- ***R1 & R2*** – Performs an AND operation on the values in *R1* and *R2*.
- ***R1 | R2*** – Performs an OR operation on the values in *R1* and *R2*.
- ***R1 ^| R2*** – Performs an exclusive-OR (XOR) operation on the values in *R1* and *R2*.
- ***R1 << R2*** – The value in *R1* is shifted left the number of spaces specified by *R2.*
- ***R1 >> R2*** – The value in *R1* is shifted right the number of spaces specified by *R2*.
- ***~R1*** – Performs a bitwise complement (reverses the state of each bit) of *R1*.
- ***!R1*** – Performs the Not function on the value in *R1*. Converts zero to one and non-zero to a zero.
- ***R1 EQ R2*** – The value in *R1* is *equal* to the value in *R2*.
- ***R1 NE R2*** – The value in *R1* is *not equal* to the value in *R2*.
- ***R1 GT R2*** – The value in *R1* is greater than the value in *R2*.
- ***R1 GE R2*** – The value in *R1* is *greater than or equal to* the value in *R2*.
- ***R1 LT R2*** – The value in *R1* is *less than* the value in *R2*.
- ***R1 LE R2*** – The value in *R1* is *less than or equal to* the value in *R2*.
- ***(R1 & R2) EQ 0*** – The result of performing an AND operation on the values present in R1 and R2 is *equal to* zero.
- ***(R1 & R2) NE 0*** – The result of performing an AND operation on the values present in R1 and R2 is *not equal to* zero.
- ***(R1 | R2) EQ 0*** – The result of performing an OR operation on the values present in R1 and R2 is *equal to* zero.

- ***(R1 | R2) NE 0*** – The result of performing an OR operation on the values present in R1 and R2 is *not equal to* zero.
- ***(R1 ^ R2) EQ 0*** – R1 value raised to the power of R2 value is *equal to* zero.
- ***(R1 ^ R2) NE 0*** – R1 value raised to the power of R2 value is *not equal to* zero.
- ***(R1 EQ 0) R2 -> Out*** – If the *R1* value is *equal to* zero, move the value in *R2* to the register specified in the *Output* property.
- ***(R1 NE 0) R2 -> Out*** – If the *R1* value is *not equal to* zero, move the value in *R2* to the register specified in the *Output* property.
- ***(R1 GT 0) R2 -> Out*** – If the *R1* value is *greater than* zero, move the value in *R2* to the register specified in the *Output* property.
- ***(R1 LT 0) R2 -> Out*** – If the *R1* value is *less than* zero, move the value in *R2* to the register specified in the *Output* property.
- ***Sum(R1[R2])*** – Sums the values in the Array specified by *R1* of which the Array size is specified by *R2*.
- ***Mean(R1[R2])*** – Calculates the Mean average of the values in the Array specified by *R1* of which the Array size is specified by *R2*.
- ***SqRtMean(R1[R2])*** – Calculates the Square Root Mean average of the values in the Array specified by *R1* of which the Array size is specified by *R2*.
- ***Min(R1[R2])*** – Finds the Minimum value in the Array as specified by *R1* of which the Array size is specified in *R2*.
- ***Max(R1[R2])*** – Finds the Maximum value in the Array as specified by *R1* of which the Array size is specified in *R2*.
- ***Copy(R1[R2]) -> Out*** – Copy the Array specified by *R1* to the Array specified by the *Output* register using the array size as specified by *R2*.
- ***Move(R1[R2] -> Out*** – Same as *Copy* above but *R1* is zeroed.

Index – Periodic blocks must execute in a defined order and the "Index" property defines this order. The lower the Index value of the Periodic block the earlier it is processed relative to other Periodic blocks.

Each Periodic block must have a unique value assigned to its Index. By default, the Index value starts at zero and is incremented each time a new Periodic block is dragged from the toolbox and placed on the work area.

When a user changes the Index value of a Periodic to the value that is already in-use by another Periodic block, the Index value of the in-use Periodic will be incremented. This will allow the insertion of Periodic blocks at the requested index value.

For example, assume the following table represents the index values assigned to all Periodic blocks defined in a program:

Periodic	**Index**
A	0
B	1
C	2

If a user creates a new Periodic named “D” and then modifies the Index property to have a value of 1, the resulting index set will be:

Periodic	**Index**
A	0
B	2
C	3
D	1

3.3.13 PID

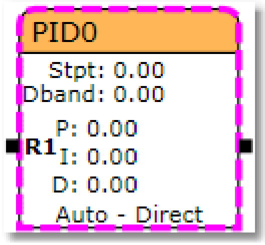
The PID block represents a three-mode (Proportional, Integral, Derivative) controller.

Since each control loop is different, it typically takes trial and error to find the optimum values for each mode. The following table can be used as a guide but only provides ball-park numbers.

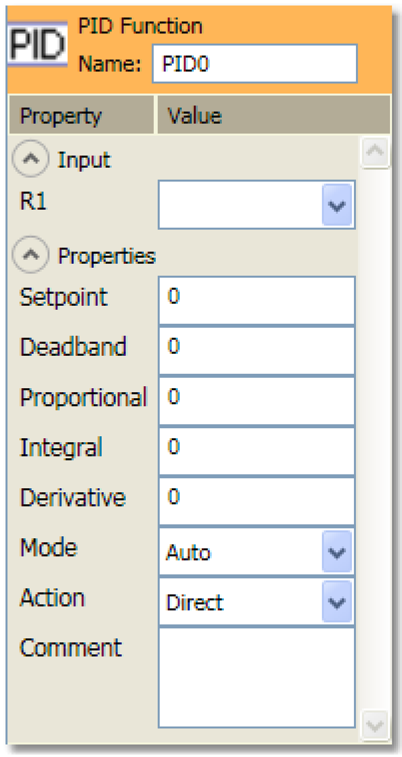
Loop Type	Proportional (%)	Integral (mins)	Derivative (mins)
Liquid	< 100	10	None
Temperature	20 - 60	2 - 15	0.25
Flow	150	0.1	None
Liquid Pressure	50 - 500	0.005 - 0.5	None
Gas Pressure	1 - 50	0.1 - 50	0.02 - 0.1
Chromatograph	100 - 2000	10 - 120	0.1 - 20

Toolbox graphical representation:

Work area graphical representation:



Properties pane graphical representation:



Property	Value
Input	
R1	
Properties	
Setpoint	0
Deadband	0
Proportional	0
Integral	0
Derivative	0
Mode	Auto
Action	Direct
Comment	

R1 – The R1 input is the register that contains the process variable being controlled.

The input value for this block may be set using multiple methods. From the property pane, the input value may be set by choosing the output of an EZ block from among those listed in the drop-down box. From the work area, the input value may be set by connecting the output of a source block to the input pin.

Setpoint – Enter a value at which the process variable is to be maintained by the controller. *Setpoint* has the same engineering units as the process variable.

Deadband – Specifies a range around the *Setpoint* within which the controller will take no action. The value entered is either side of the *Setpoint* so that the total *Deadband* range is twice the value entered. *Deadband* has the same engineering units as the process variable.

Proportional – Proportional is the gain of the controller. At a minimum, you will need a Proportional value whether or not the Integral and Derivative modes are used. Proportional control reduces the error but typically does not eliminate it unless the process has a natural offset.

Integral – With Integral action, the controller output is proportional to the amount and duration of the error. Integral is typically used to correct the offset between the process variable and the Setpoint that remains when using Proportional mode only. This value can be zero if Integral mode is not wanted.

Derivative – With Derivative action, the controller output is proportional to the rate of change of the process variable or error. Derivative helps with overshoot and ringing or oscillations. This value can be zero if Derivative mode is not wanted.

Mode – Select among:

- ***Manual*** – The user enters an output value
- ***Auto*** – The PID function automatically calculates an output.

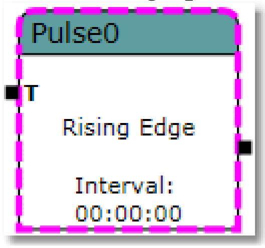
Action – Select among:

- ***Direct*** – Controller output increases when the process variable is less than the Setpoint and decreases when the process variable is greater than the Setpoint.
- ***Reverse*** – Controller output increases when the process variable is greater than the Setpoint and decreases when the process variable is less than the Setpoint.

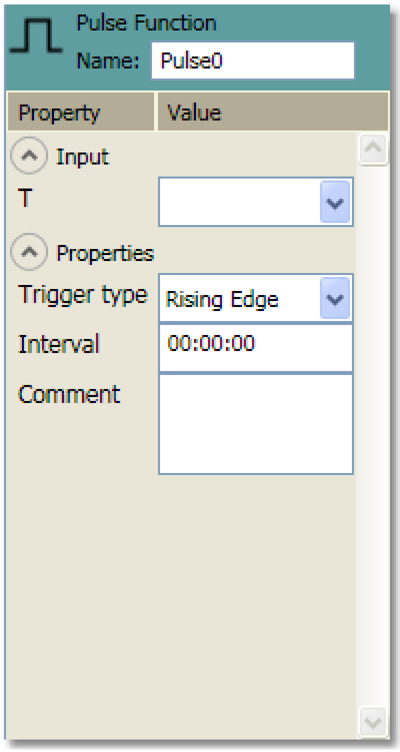
3.3.14 Pulse

Toolbox graphical representation:

Work area graphical representation:



Properties pane graphical representation:


Pulse Function
Name: Pulse0

Property	Value
Input	
T	
Properties	
Trigger type	Rising Edge
Interval	00:00:00
Comment	

T – The input register that is used to trigger the start of the pulse.

The input value for this block may be set using multiple methods. From the property pane, the input value may be set by choosing the output of an EZ block from among those listed in the drop-down box. From the work area, the input value may be set by connecting the output of a source block to the input pin.

Trigger type – Select among:

- ***Rising Edge*** – Trigger (*T*) goes from a 0 to a 1.
- ***Falling Edge*** – Trigger (*T*) goes from a 1 to a 0.
- ***Change of State*** – Trigger (*T*) undergoes either transition; 0 to 1 or 1 to 0.
- ***High Level*** – Trigger (*T*) is a 1.
- ***Auto Reset*** – Attempts to write a 0 to Trigger (*T*).

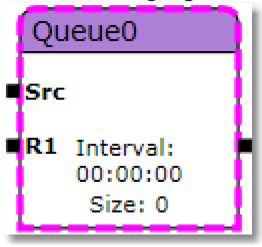
Interval – Specify a time for the duration of the pulse or time the output value remains a one. The interval value should be specified in the format HH:MM:SS

3.3.15 Queue

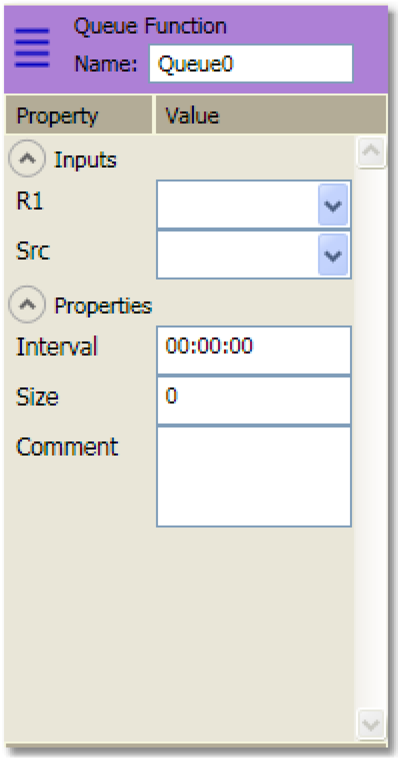
The Queue block provides the capability of delaying the value of a specified variable for a period of time controlled by the Interval property and the size of an array. At the same time the array is updated, the last value in the array becomes the Queue block's output value.

Toolbox graphical representation:

Work area graphical representation:



Properties pane graphical representation:



Name: Queue0

Property	Value
Inputs	
R1	
Src	
Properties	
Interval	00:00:00
Size	0
Comment	

R1 – Specify the element that will provide the source data or input data for the array.

Src – Specify the starting element for the array. Arrays can be defined by the *Holding Registers* application.

The input values for this block may be set using multiple methods. From the property pane, an input value may be set by choosing the output of an EZ block from among those listed in the drop-down box. From the work area, an input value may be set by connecting the output of a source block to an input pin.

Interval – Enter an interval at which the value in the register specified by the *R1* input is read and placed in the first slot of the array shifting all other slots down as in a FIFO. The data in the last slot becomes the Queue block's output value. Although the *Interval* controls the frequency, a read operation of the source data register must occur to trigger the operation. The read operation can be equal to or greater, but not less than the Interval. Enter a queue time interval value in the format HH:MM:SS

Size – Enter a numeric value to define the number of slots in the array.

3.3.16 Ramp

The Ramp block provides an analog ramp that produces a ramp change every calculation cycle. The magnitude is dependent upon the specified rate of change (Slope) and the sample interval (*R1* input read frequency). The direction of the ramp change is determined by the *R1* input value and by the previous output value. The change is applied to the output value in the same direction of the input value.

Algorithm:

if: **Current Input - Previous Output = Ramp Value**
then: **Output Value = Current Input**

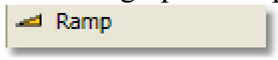
if: **Current Input - Previous Output > Ramp Value**
and: **Current Input > Previous Output**
then: **Output Value = Previous Output + Ramp Value**

if: **Current Input - Previous Output > Ramp Value**
and: **Current Input < Previous Output**
then: **Output Value = Previous Output - Ramp Value**

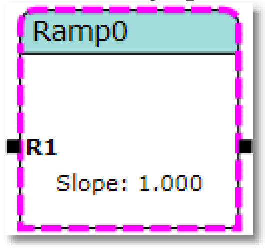
Where:

- **Current Input** = Current input value (*R1*) to the Ramp algorithm.
- **Previous Output** = Previous output value calculated by the Ramp algorithm.
- **Ramp Value** = Slope value x number of seconds since *R1* was read.
- **Output Value** = Current output value calculated by the Ramp algorithm.

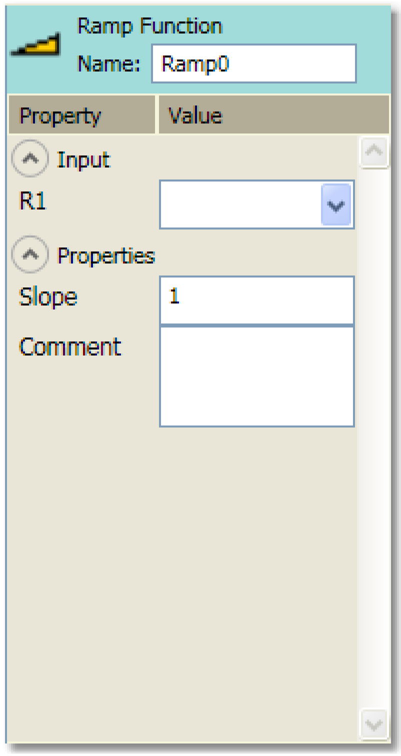
Toolbox graphical representation:



Work area graphical representation:



Properties pane graphical representation:


Ramp Function

Name: Ramp0

Property	Value
Input	
R1	
Properties	
Slope	1
Comment	

R1 – Specify the data source for the Ramp block.

The input value for this block may be set using multiple methods. From the property pane, the input value may be set by choosing the output of an EZ block from among those listed in the drop-down box. From the work area, the input value may be set by connecting the output of a source block to the input pin.

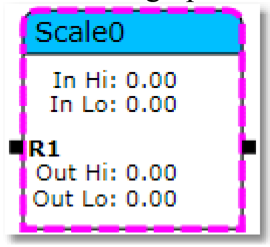
Slope – Enter a value for the slope that when multiplied times the number of seconds since *R1* was last read, provides the maximum ramp value.

3.3.17 Scale

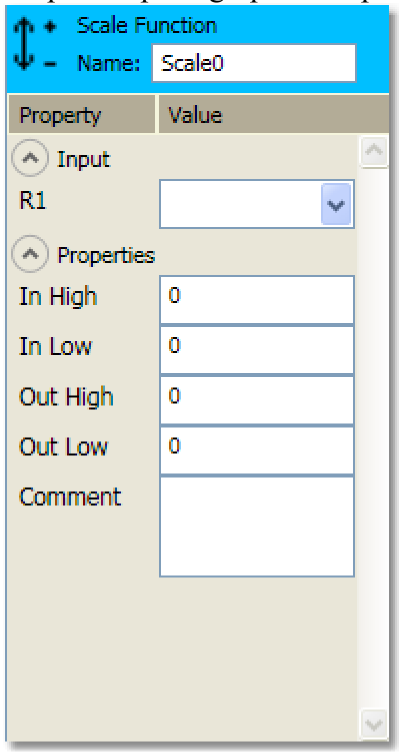
The Scale block allows the user to specify an input range and an output range. The input value will scale proportionally based on the ratio of the input and output ranges.

Toolbox graphical representation:

Work area graphical representation:


Scale0
In Hi: 0.00
In Lo: 0.00
R1
Out Hi: 0.00
Out Lo: 0.00

Properties pane graphical representation:


Scale Function
Name: Scale0

Property	Value
Input	
R1	
Properties	
In High	0
In Low	0
Out High	0
Out Low	0
Comment	

R1 – The input location that provides the source data for the scale block.

The input value for this block may be set using multiple methods. From the property pane, the input value may be set by choosing the output of an EZ block from among those listed in the drop-down box. From the work area, the input value may be set by connecting the output of a source block to the input pin.

In High – Specify the High value for the input range.

In Low –Specify the Low value for the input range.

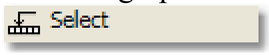
Out High –Specify the High value for the output range.

Out Low – Specify the Low value for the output range.

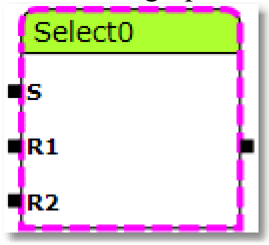
3.3.18 Select

The *Select* operation looks in the register specified by the Select input (*S*). If the value is zero, the value in the register specified by input *R1* becomes the output value. If the value is a non-zero value (negative or positive; one or greater), the value in register specified by input *R2* becomes the output value.

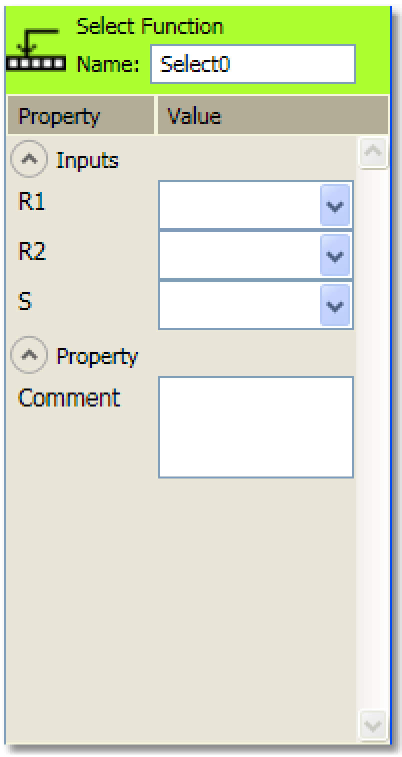
Toolbox graphical representation:



Work area graphical representation:



Properties pane graphical representation:



R1 – Specify a register in which its value will be selected when input *S* is set to zero.

R2 – Specify a register in which its value will be selected when the input S is set to a non-zero value (negative or positive; one or greater).

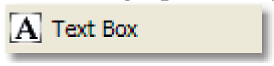
S – Specify an input such that when its value is a zero, the value in *R1* becomes the selected value.

The input values for this block may be set using multiple methods. From the property pane, an input value may be set by choosing the output of an EZ block from among those listed in the drop-down box. From the work area, an input value may be set by connecting the output of a source block to an input pin.

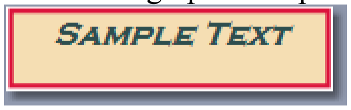
3.3.19 Text Box

`Text Box` elements are user interface elements that serve as containers for any comment text used in an EZ Block Builder application. These blocks may be placed anywhere in the work area but they do not have any impact on the functionality of the program. Several dropdown lists appear in the properties pane for the text box that allow the user to set font, size, color and other style options.

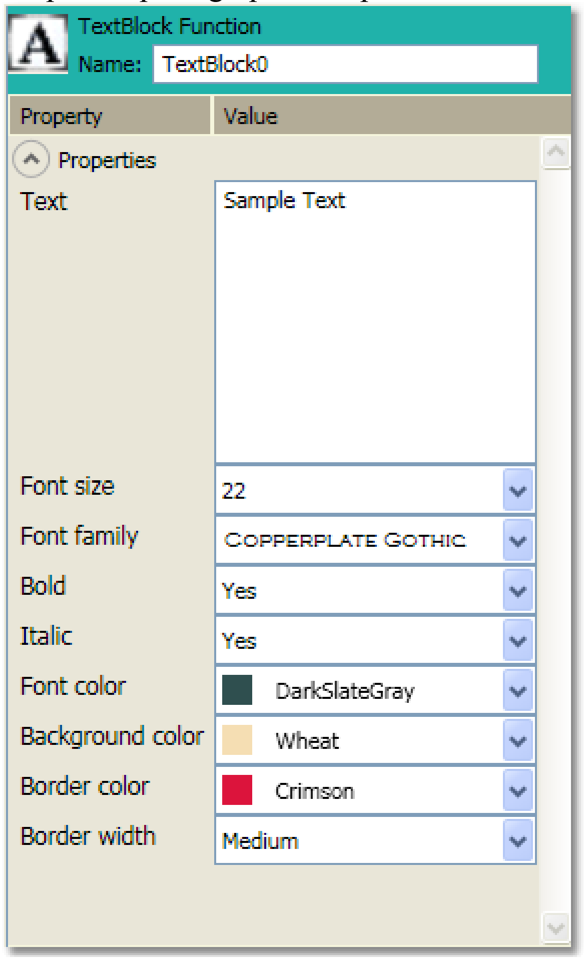
Toolbox graphical representation:



Work area graphical representation:



Properties pane graphical representation:


TextBlock Function

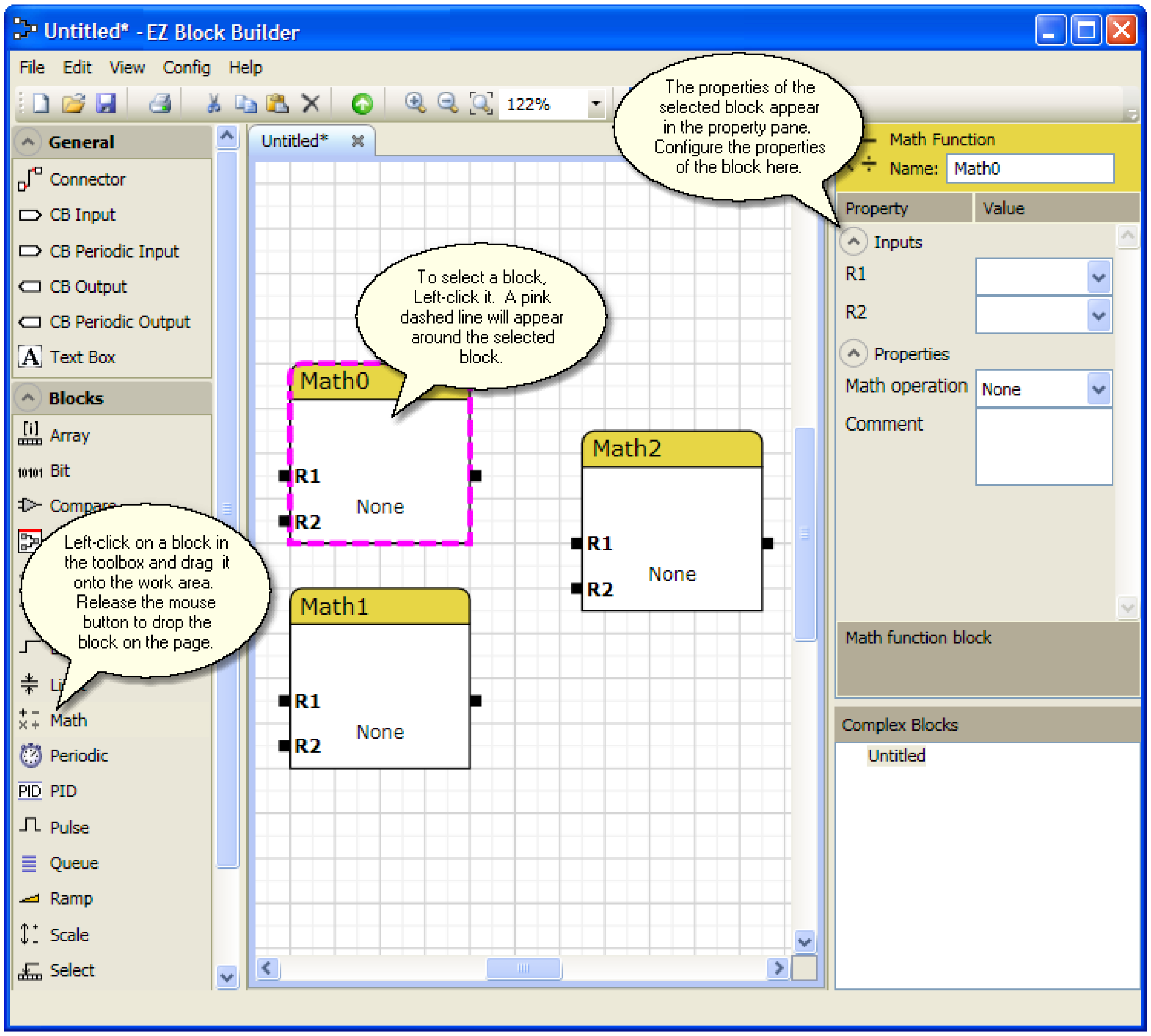
Name: TextBlock0

Property	Value
Properties	
Text	Sample Text
Font size	22
Font family	COPPERPLATE GOTHIC
Bold	Yes
Italic	Yes
Font color	⬛ DarkSlateGray
Background color	🟨 Wheat
Border color	🟥 Crimson
Border width	Medium

4 Creating a Program

To create a new program with the EZ Block Builder, select File|New from the main menu.

A blank work area will be displayed. Drag EZ Blocks from the toolbox to the work area as shown below.


Untitled* - EZ Block Builder
File Edit View Config Help
122%
General
Connector
CB Input
CB Periodic Input
CB Output
CB Periodic Output
Text Box
Blocks
Array
Bit
Compare
Math
Periodic
PID
Pulse
Queue
Ramp
Scale
Select
Untitled*
Math0
R1
R2
None
Math1
Math2
Left-click on a block in the toolbox and drag it onto the work area. Release the mouse button to drop the block on the page.
To select a block, Left-click it. A pink dashed line will appear around the selected block.
The properties of the selected block appear in the property pane. Configure the properties of the block here.
Math Function
Name: Math0
Property
Value
Inputs
Properties
Math operation
Comment
Math function block
Complex Blocks
Untitled

To configure an EZ block, left-click it. The block's properties will appear in the property pane on the right side of the screen. Modify the property settings as appropriate. Repeat this process for each block of your program.

The EZ Builder supports three methods of connecting blocks:

1) Drag the "Connector" program element from the toolbox to the work area. Drop the connector ends on the input and output pins of the blocks you wish to connect.

2) Hover the mouse over the pin of a block in the work area. The mouse icon appearance will change to look like a pen. Left-click on the pin and drag the resulting line to a pin on a different EZ block to establish a connection.
3) Pins that accept a single input or that can push a result to a single output may be configured via the property pane. Select the object that has the pin you wish to configure and make the appropriate selection from the dropdown lists provided.

Once all the blocks that comprise the program have been placed in the work area and configured, a user may save the file to disk or download it directly to the device.

To validate that the program is properly configured, choose **`Config|Validate`** Program from the main menu. This will cause EZ Builder to check if the program contains blocks that have input pins that are needed to perform a function but that are unassigned. If this check detects any needed but unassigned input pins, a dialog will be displayed that contains a list of EZ Blocks that require modification.

5 Register Assignment Strategies

The application can handle register assignments to X-series devices in several different ways.

Using the default register assignment methods, each successive EZ block of a particular type that is added to the application will be assigned the next available register value. Register values are integers that have a minimum value of zero.

For example, the first Math block added will be assigned to register address 0, the second to register 1, etc. If a PID block is added to a program containing only two Math blocks, it will be assigned to register address 0, since it will be the first PID block.

The issue of register assignments becomes more complicated when editing a program that is already in use in a device. The reason for the extra complexity stems from the fact that other applications within the device may be referencing outputs of particular EZ blocks within the program. This makes it necessary for the EZBB application to minimize changes to EZ block register assignments; otherwise these external references will be broken.

As a result, the application supports the following three operational strategies with regard to EZ block register assignments:

- **Optimize** – This is the default strategy. The application will change EZ block register assignments as necessary to ensure that no holes in memory exist; i.e., all EZ blocks of a particular type will be assigned to contiguous registers starting at register 0.
- **Preserve** - The application will not change EZ block register assignments.
- **Prompt** - The tool will prompt the user before changing a EZ block register assignment, proceeding only if the user answers ‘Yes’.

Users will likely want to choose the ‘Optimize’ strategy when creating a new program and the ‘Prompt’ strategy when modifying an existing program.

The register assignment strategy can be modified in the `Options` dialog by choosing **`Config->Options`** from the main menu.

6 Alias Lists

Alias lists allow a user to map an X-Series array/register pair to a user-defined text string. This helps the user assign External Locations using human-readable XML text instead of the numeric notation required by the device.

6.1 Alias List file structure

To help illustrate the hierarchy of the alias list, an example alias list file is shown below:

```
<?xml version="1.0" encoding="utf-8"?>
<AliasList Version="1.0">
  <AppType Name="Holding Registers">
    <Category Name="Byte">
      <Parameter Alias="0x00">5.0</Parameter>
      <Parameter Alias="0x01">5.1</Parameter>
      <Parameter Alias="0x02">5.2</Parameter>
    </Category>
    <Category Name="Integers">
      <Parameter Alias="Signed Int16">1.0</Parameter>
      <Parameter Alias="Unsigned Int16">2.0</Parameter>
      <Parameter Alias="Signed Int32">3.0</Parameter>
      <Parameter Alias="Unsigned Int32">4.0</Parameter>
    </Category>
  </AppType>
  <AppType Name="Operations">
    <Category Name="Outputs of Math and Compare">
      <Parameter Alias="Output of Math0 FB">7.0</Parameter>
      <Parameter Alias="Output of Math1 FB">7.1</Parameter>
      <Parameter Alias="Output of Math2 FB">7.2</Parameter>
    </Category>
    <Category Name="Outputs of Select and Lag">
      <Parameter Alias="Output of Select0 FB">32.0</Parameter>
      <Parameter Alias="Output of Lag0 FB">36.0</Parameter>
    </Category>
  </AppType>
</AliasList>
```

The *AliasList* tag encapsulates the entire alias list definition.

One or more *AppType* tags may appear as children of the *AliasList* tag. The *Name* attribute defines the text that the user will see when selecting an application type from the EZ Block Builder.

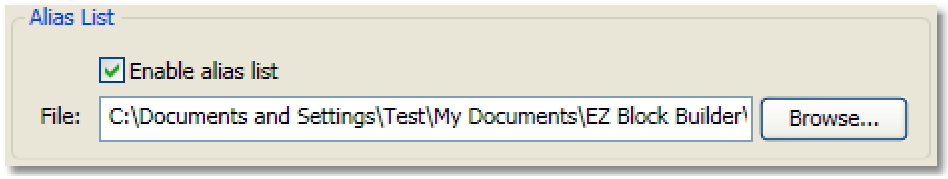
One or more *Category* tags may appear as children of the *AppType* tag. The *Name* attribute defines the text that the user will see when selecting a category entry from the EZ Block Builder.

Finally, one or more *Parameter* tags may appear as children of the *Category* tag. Each *Parameter* tag maps to an array/register pair that must be expressed in the format “array.register” where “array” and “register” are numeric values. The *Category* tag’s

Alias attribute defines the alias text that the user will see when selecting a Parameter entry from the EZ Block Builder.

6.2 Loading Alias Lists

In order to enable alias list processing, choose **`Config|Options`** from the main menu to bring up the application options dialog and configure the "Alias List" section by checking the "Enable alias list" checkbox and selecting the alias list file to load. By default, the application will look for alias list files in the user's ***My Documents\EZ Block Builder\AliasLists*** directory.



File: C:\Documents and Settings\Test\My Documents\EZ Block Builder\

Users may add categories and parameters to the alias list by configuring External Locations and selecting the `Create Alias…` option. This procedure is described in the External Location section of this document.

7 Profiles

The graphical programming tool supports application profiles, which allows the user to place limits on the number of instances of a particular application type that an EZ Block Builder program will support. These instance values are used when configuring External Location blocks.

For example, if an X-Series device is known to support only 3 instances of the Communications application type, a profile can be used to set the instance limit for the Communications application type to 3 thereby preventing a user of EZ Block Builder from selecting instance values that are out of range.

7.1 Profile file structure

Profiles are XML files and an example file that illustrates the profile's structure is shown below:

```xml
<?xml version="1.0" encoding="utf-8"?>
<Profile Version="1.0">
  <AppType Name="Operations">
    <Instance Name="Ops A" />
    <Instance Name="Ops B" />
    <Instance Name="Demo" />
  </AppType>
  <AppType Name="Holding Registers">
    <Instance Name="HR A" />
    <Instance Name="HR B" />
  </AppType>
</Profile>
```

In the example above, the profile defines the "Operations" application type to have 3 instances. Instance 0 is named "Ops A", instance 1 is named "Ops B", and instance 2 is named "Demo". The "Holding Registers" application type has 2 instances. Instance 0 is named "HR A" and instance 1 is named "HR B".

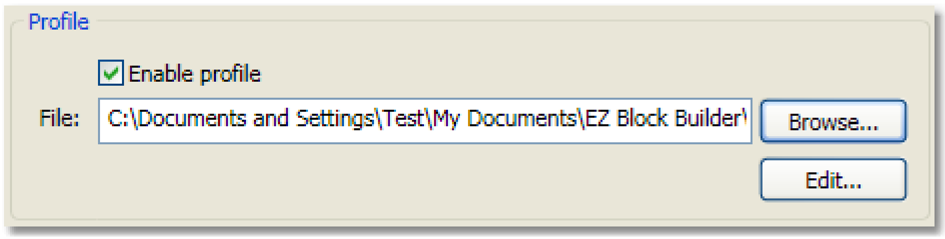
A single *Profile* tag encapsulates the profile definition.

One or more *AppType* tags appear as children of the *Profile* tag. The *Name* attribute matches an application type present in EZ Block Builder's `AppTypes.xml` file. Refer to the section titled File Structure for more information on `AppTypes.xml`.

One or more *Instance* tags appear as children of the *AppType* tag. Each *Instance* tag child defines a single instance and the *Instance* tag's Name attribute defines the name that the EZ Block Builder UI will report for the instance.

7.2 Loading Profiles

To load a profile, select **`Config|Options…`** from the main menu to display the options dialog. In the section of the dialog titled "Profile", check the Enable profile button and choose the profile to load.

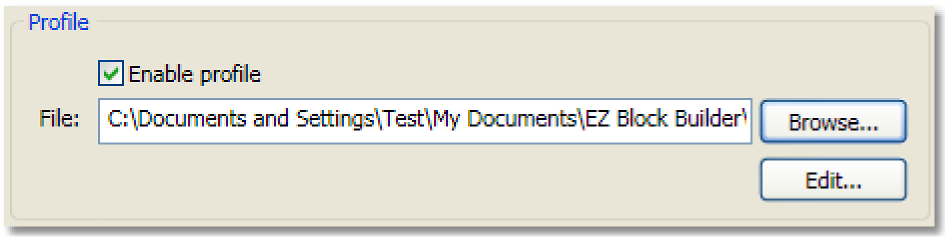


Enable profile: ✓
File: C:\Documents and Settings\Test\My Documents\EZ Block Builder\

By default, EZ Builder will look for application profiles in the user's "My Documents\EZ Block Builder\Profiles" directory although any directory may be specified.

7.3 Editing Profiles

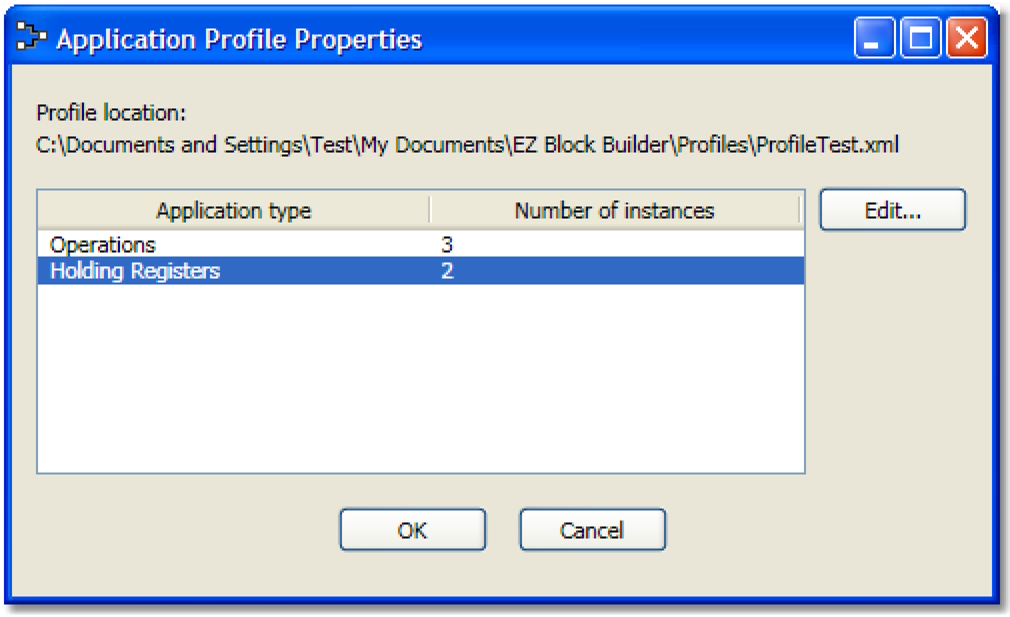
To edit a profile from within EZ Block Builder, select **`Config|Options…`** from the main menu to display the options dialog. In the section of the dialog titled "Profile", check the Enable profile button and browse to the profile to be edited.



Enable profile: ✓
File: C:\Documents and Settings\Test\My Documents\EZ Block Builder\

Using the example profile discussed in the section titled Profile file structure , select the Holding Registers row and click `Edit…`

Click the `Edit…` button to bring up the `Application Profile Properties` dialog.

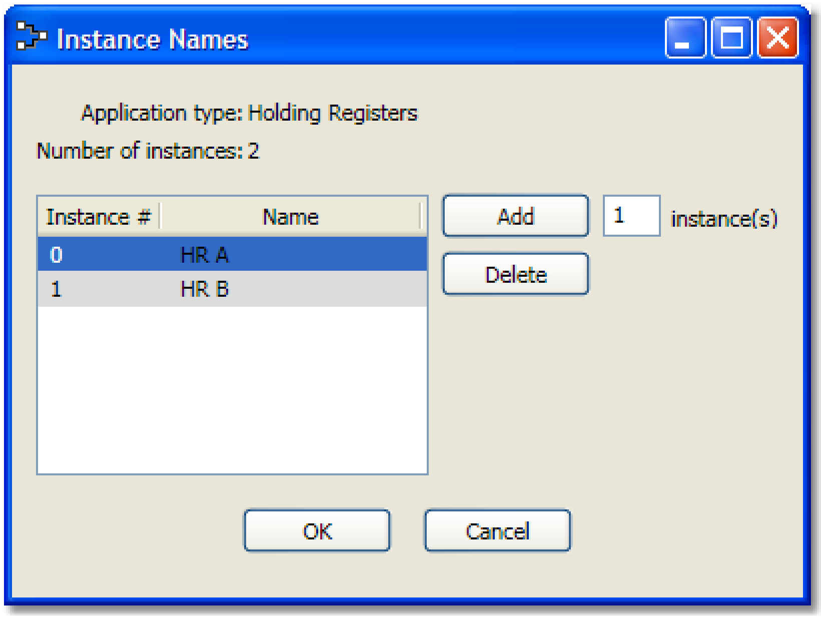


Profile location:
C:\Documents and Settings\Test\My Documents\EZ Block Builder\Profiles\ProfileTest.xml

Application type	Number of instances
Operations	3
Holding Registers	2

The `Application Profile Properties` dialog display each Application type defined in the currently loaded profile and the number of instances each application type supports. The location of the currently loaded profile is displayed at the top of the dialog. In the image above, the profile being targeted is the example profile described in the section titled Profile file structure.

To edit a profile entry, select the row of the row of the application type to be modified and click the Edit… button to bring up the `Instance Names` dialog.



Application type: Holding Registers
Number of instances: 2

Instance #	Name
0	HR A
1	HR B

From this dialog a user may add instances by entering the number of instances to be added in the box provided on the form and clicking the `Add` button.

To edit the name of an instance, click inside the Name of the instance to be changed and edit the text.

An instance may be deleted by highlighting a row and pressing the `Delete` button.

None of the changes made using this dialog are saved to the profile until the user clicks the `OK` button on this form.

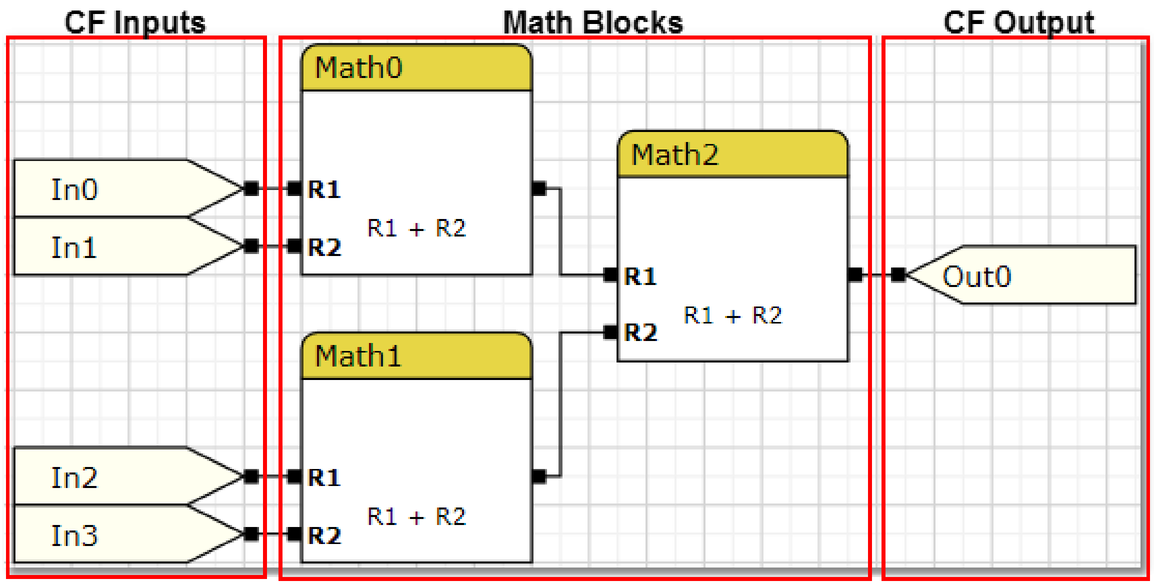
8 Templates

A template is an interconnected set of program elements which are 'pre-wired' internally to a specific number of inputs and outputs. A user may wish to consider creating templates of commonly used program functions so that they may be easily re-used in future projects. The following sections discuss the creation and usage of EZ Block Builder templates.

8.1 Creating templates

In the following example, a template will be created that accepts 4 input values and provides an output that is the sum of the 4 values.

To create a program template with the EZ Block Builder, select **`File|New`** from the main menu. Configure the work area as shown below:



In the example above, Math0 produces the sum of In0 and In1, Math1 produces the sum of In2 and In3, and Math 2 produces the sum of Math0 and Math1. The result of the Math2 operation is then made available on the CF Output pin labeled Out0.

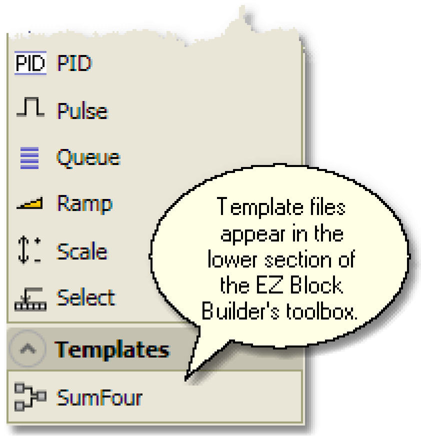
To save the above example as a template file, select **`File| Create Template…`** from the main menu. A dialog box will be displayed that will allow you to name the template file. The name you choose for the file will be the name that appears when the template is loaded into the EZ Block Builder's toolbox.

In order for EZ Block Builder to find this template, it must be stored in the "Templates" directory. The Templates direct can be found in the user's "My Documents\EZ Block Builder\Templates" directory. The default directory provided by the template's "Save File" dialog set to the proper directory so all that is required is to provide a name for the Template. For this example we will specify the template file name to be "SumFour".

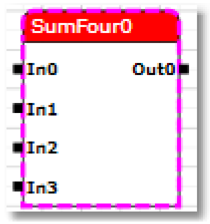
A description of how to use the template that was just created is described in the following section, Using templates .

8.2 Using templates

Template files are available for use during design-time just like any other program element in the toolbox.


PID
Pulse
Queue
Ramp
Scale
Select
Templates
SumFour
Template files appear in the lower section of the EZ Block Builder's toolbox.

To use the template that was created in the previous section, Creating templates , restart EZ Block Builder. EZ Block Builder initializes the templates found in the "My Documents\EZ Block Builder\Templates" directory on start-up and the new template file named "SumFour" will be discovered and loaded into the toolbox.

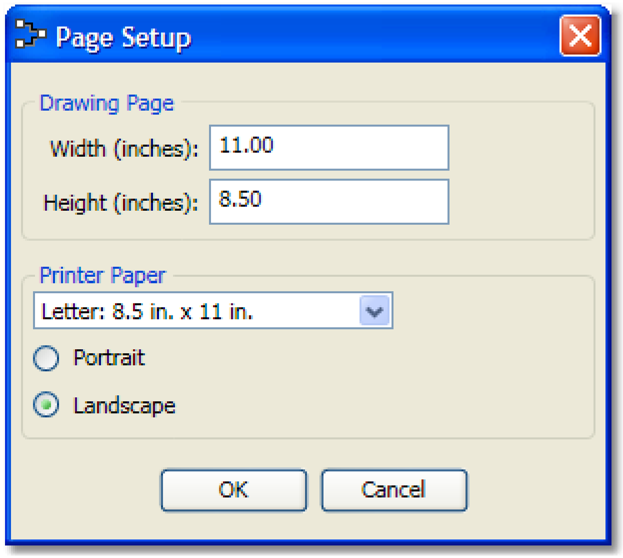


Dragging the example "SumFour" template from the toolbox to the work area causes a complex block to be added to the program and given the name "SumFour0". This complex block, however, is pre-configured to provide the functionality that was defined in the "SumFour" template.

At this point, the SumFour0 block may be treated as any other EZ block and the user may configure it or connect it with other EZ blocks as needed.

9 Page Layout and Printing

To configure the page layout for an EZ Block Builder program, choose **File|Page Setup…** from the application's main menu to display the Page Setup dialog.



Drawing Page
Width (inches): 11.00
Height (inches): 8.50

Printer Paper
Letter: 8.5 in. x 11 in.
Portrait [✓ Landscape]

The drawing page size determines the shape and size of the work area shown in EZ Builder. The printer paper settings determine the paper size as well as the page orientation used when printing. The printer page size does not necessarily have to match the drawing page size. If the drawing page size is larger than the printer page size, the drawing will be printed across multiple pages.

Alter the settings in this dialog to match the page size of the printer. Click OK once the page settings are configured.

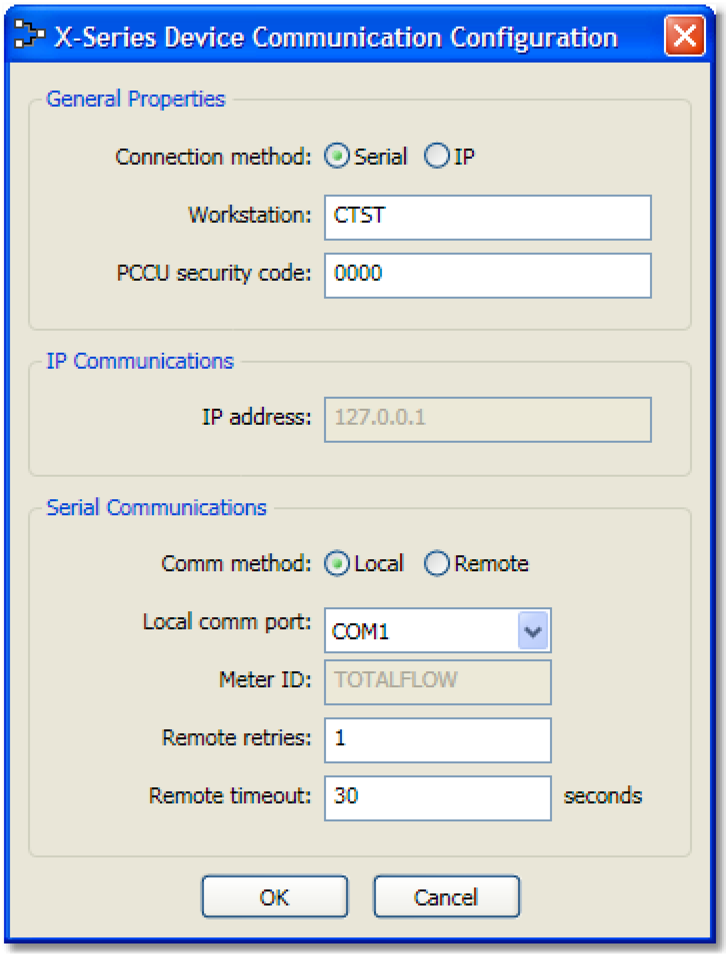
To print an EZ Block Builder program, choose **File|Print…** from the application's main menu to bring up the Print dialog.

10 Device Operations

EZ Block Builder allows a user to both download and upload operations application programs to and from an X-Series device. Uploaded operations applications may be viewed in EZ Block Builder's editor and X-Series devices are programmed by downloading EZ Block Builder programs to the device. This section discusses how to configure EZ Block Builder to talk with an X-Series device and how to use EZ Block Builder's device upload and download capabilities.

10.1 Communications Configuration

Select **`Config->Communications Setup…`** from the main menu to access the `X-Series Device Communication Configuration` dialog.



Connection method: [✓ Serial] IP
Workstation: CTST
PCCU security code: 0000
IP address: 127.0.0.1
Comm method: [✓ Local] Remote
Local comm port: COM1
Meter ID: TOTALFLOW
Remote retries: 1
Remote timeout: 30 seconds

Using this dialog a user may configure the application to connect to an X-Series device using either serial or IP protocols.

The user should enter the device workstation name and PCCU security code no matter which communication method is desired.

To connect to a device via a TCP/IP network choose "IP" as the connection method in the dialog box shown.

After selecting "IP" the IP address control will be enabled and the user may enter the IP address for the device.

To connect to a device via USB or RS-232 choose "Serial" as the connection method.

Once the "Serial" radio button is selected the controls in the "Serial Communication" section of the dialog will be enabled.

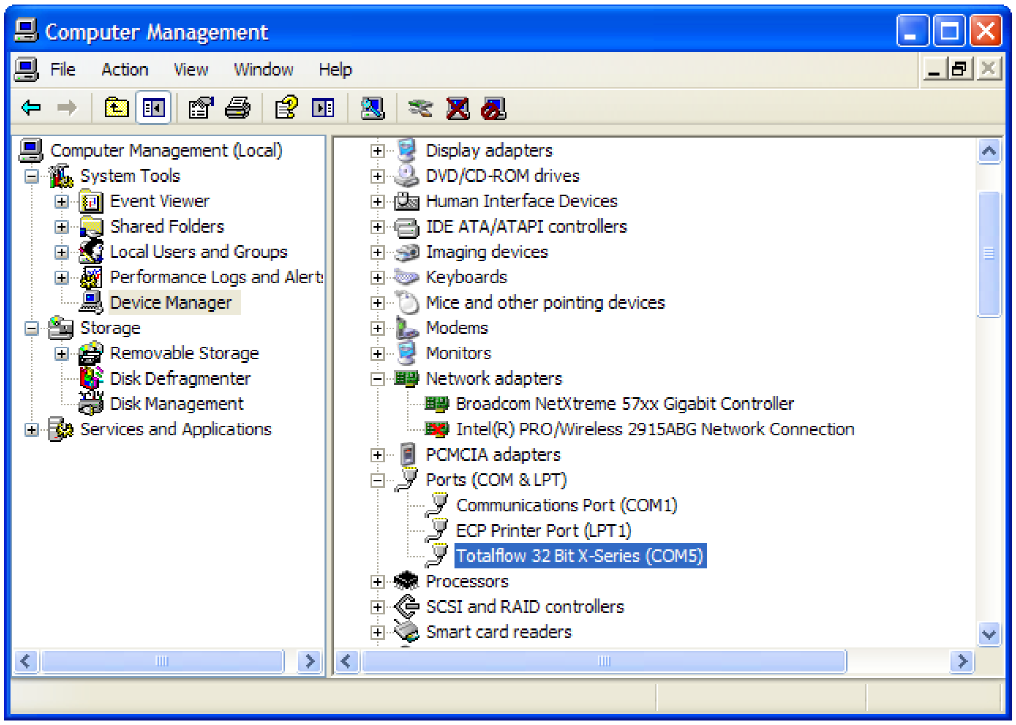
Choose local or remote mode as required. For local mode, choose the COM port connected to the device; for local mode, enter the Meter ID of the target device.

For USB connections, use a connection method of “Serial” and a Comm method of “Local”. If there is difficulty in connecting to the device using USB it is possible that additional setup may be required on your PC to enable USB communications. Refer to the next section (USB Connectivity) for additional information.

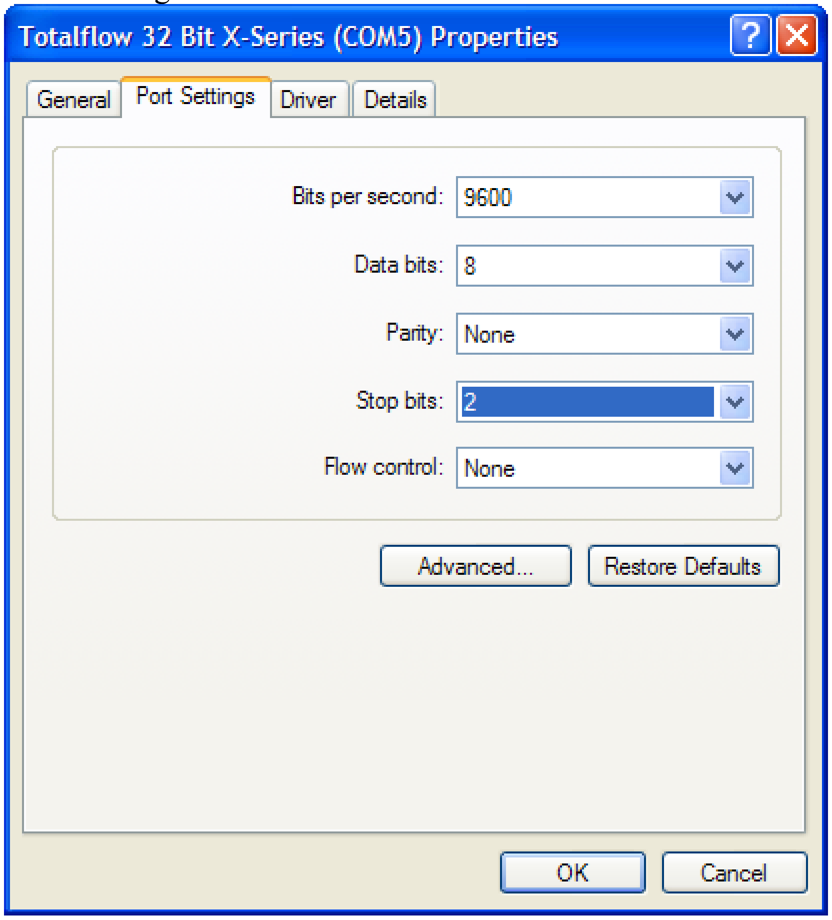
10.1.1 USB Connectivity

If a problem is encountered while attempting to connect to an X-Series device via USB, it may be necessary to perform the following procedure. This procedure describes how to configure Windows XP to allow the EZ Block Builder to communicate with an X-Series device via a USB cable.

2. Connect your PC to the device via the USB cable.
3. Open the Computer management console. Click **`Start->Run`** and type “`compmgmt.msc`” in text box (without quotes) and hit OK. The Computer Management window will be displayed.
4. Choose the “Device Manager” from the pane on the left and expand the node titled “Ports (COM & LPT)” from the pane on the right.
5. Double-click the entry labeled “Totalflow 32 Bit X-Series” as shown below:



6. The Properties window for this COM port will be displayed. Navigate to the "Port Settings" tab as shown below:

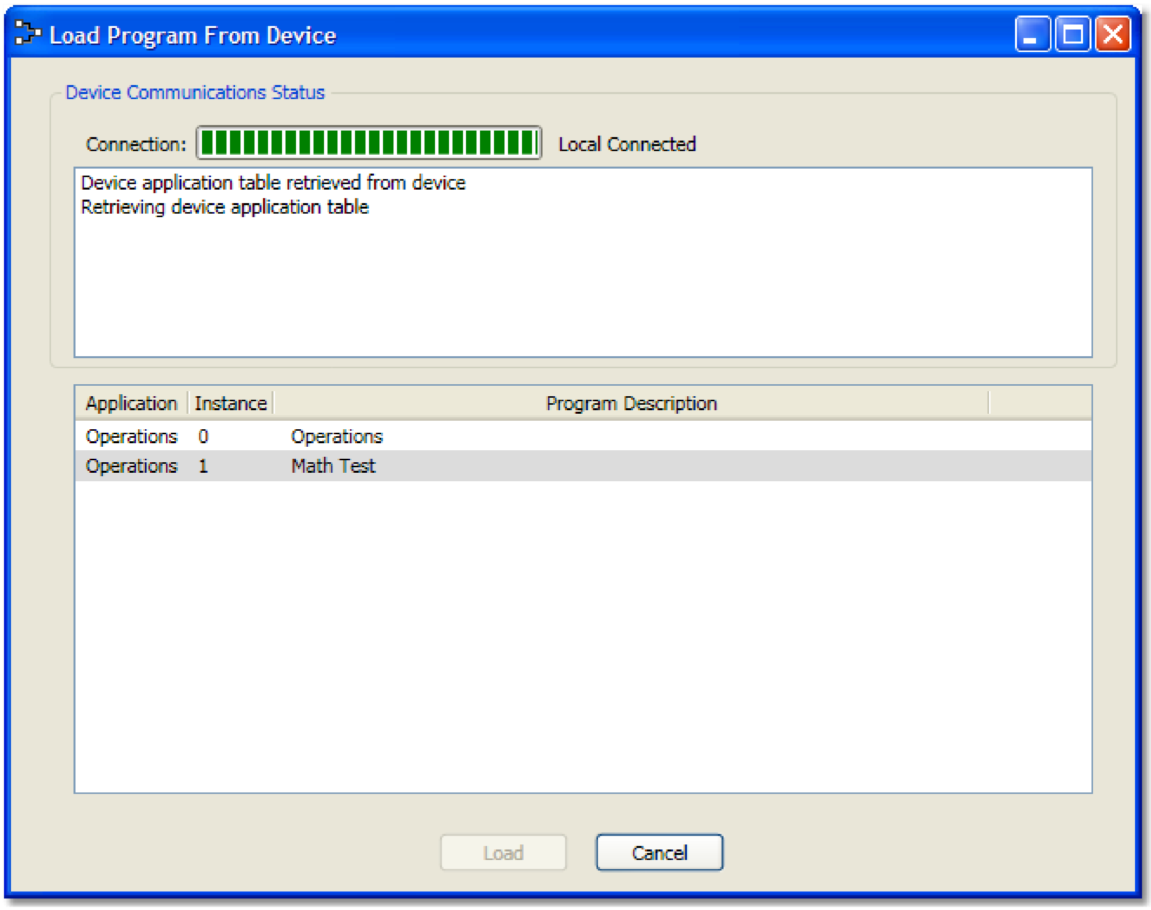

Totalflow 32 Bit X-Series (COM5) Properties

Bits per second: 9600
Data bits: 8
Parity: None
Stop bits: 2
Flow control: None

7. Configure the port settings as show above: 9600 bps, 8 data bits, no parity, 2 stop bits, and no flow control.
8. Click OK to complete the operation.
9. The USB COM port shown in the title bar of the dialog (COM5 in this example) is now configured to talk to the X-Series device.

10.2 Loading a file from a device

Operations applications stored in X-Series devices can be uploaded from the device into EZ Block Builder. After configuring the device communication settings as described in the section titled Communications Configuration, select **`File|Open From Device…`** to bring up the `Load Program from Device` dialog.



Device application table retrieved from device
Retrieving device application table

Application	Instance	Program Description
Operations	0	Operations
Operations	1	Math Test

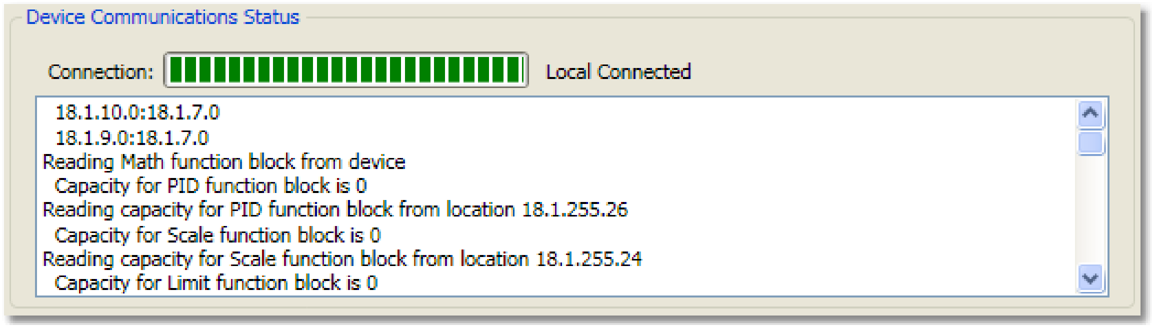
The Device Communications Status section of the dialog will provide detailed information about the connection status and the messages that are passing between the EZ Block Builder and the X-Series device.

After a connection is successfully established, the Connection progress bar will turn green and the Operations application present on the device will be displayed in the lower section of the dialog.

In the example image shown above, two Operations applications are present on the device. The first is named “Operations” and the second is named “Math Test”.

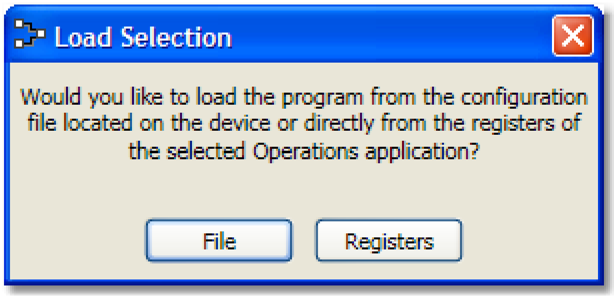
To begin the process of uploading an Operations application, click on the row that contains the Operations application you wish to upload. After selecting an Operations application, the `Load` button on the dialog will become enabled. Click `Load` to begin the upload procedure.

As the upload operation procedure progresses, information read from the X-Series device will be displayed in the Device Communications Status window.



```
  18.1.10.0:18.1.7.0
  18.1.9.0:18.1.7.0
Reading Math function block from device
  Capacity for PID function block is 0
Reading capacity for PID function block from location 18.1.255.26
  Capacity for Scale function block is 0
Reading capacity for Scale function block from location 18.1.255.24
  Capacity for Limit function block is 0
```

If the Operations application being uploaded has an associated EZ Block Builder layout file, the following dialog will be displayed after the upload is complete.



Would you like to load the program from the configuration file located on the device or directly from the registers of the selected Operations application?

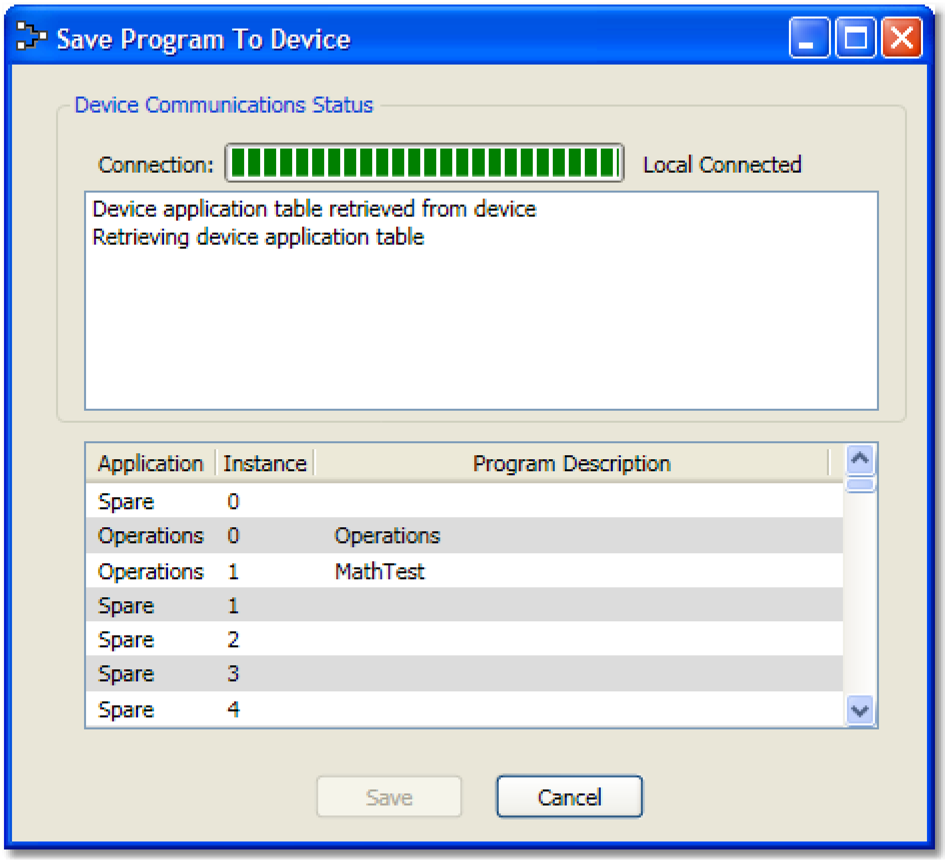
This dialog allows the user to choose to load the program with the layout information found in the file or directly from the registers of the X-Series device without the layout information.

If an EZ Block Builder layout file is not found, then the above dialog will not be displayed and the program will be loaded into EZ Block Builder directly from the device's registers.

Programs loaded from the device registers will not have layout information. EZ Block Builder will create EZ blocks for each program element and connect them with each other as required but the user may want to reposition the EZ Blocks inside the work area to produce a more human-readable picture of the program.

10.3 Saving a file to a device

To program an X-Series device, use EZ Block Builder to download an Operations application file to the device. After either creating a program in the EZ Block Builder work area or loading an EZ Block Builder file from disk and after configuring the device communication settings as described in the section titled Communications Configuration, select **`File|Save to Device…`** to bring up the `Save Program to Device` dialog.



Device application table retrieved from device
Retrieving device application table

Application	Instance	Program Description
Spare	0	
Operations	0	Operations
Operations	1	MathTest
Spare	1	
Spare	2	
Spare	3	
Spare	4	

The Device Communications Status section of the dialog will provide detailed information about the connection status and the messages that are passing between the EZ Block Builder and the X-Series device.

After a connection is successfully established, the Connection progress bar will turn green and the Operations application present on the device will be displayed in the lower section of the dialog.

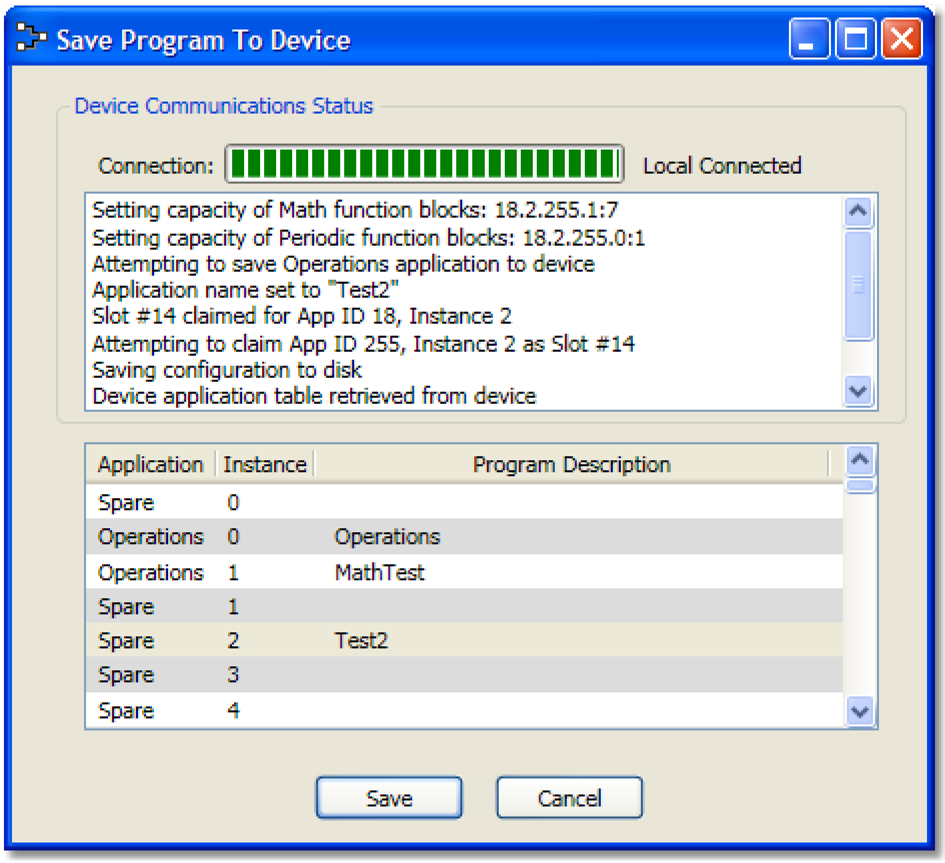
Two types of Applications may be displayed in the lower section. “Spare” rows represent empty locations in the device that may be used to store Operations applications. “Operations” rows represent locations that already have Operations applications present.

In the example image shown above, two Operations applications are present on the device. The first is named “Operations” and the second is named “Math Test”.

To begin the process of dowloading and saving an Operations application to the XSeries device, click on either a Spare or Operations row in the lower portion of the dialog. You may optionally set the name of the Operations application by clicking in the Description field of the row and entering a name.

Once a row has been selected the Save button will be enabled on the dialog. Click Save to begin the download operation.

As the download operation progresses, information transmitted to the X-Series device will be displayed in the Device Communications Status window as shown below.



```
Setting capacity of Math function blocks: 18.2.255.1:7
Setting capacity of Periodic function blocks: 18.2.255.0:1
Attempting to save Operations application to device
Application name set to "Test2"
Slot #14 claimed for App ID 18, Instance 2
Attempting to claim App ID 255, Instance 2 as Slot #14
Saving configuration to disk
Device application table retrieved from device
```

Application	Instance	Program Description
Spare	0	
Operations	0	Operations
Operations	1	MathTest
Spare	1	
Spare	2	Test2
Spare	3	
Spare	4	

Once the Operation application has been downloaded and saved to the X-Series device, close the dialog box.

11 Appendix

11.1 Logical, bitwise, and mathematical operations

Several EZ blocks allow the user to define either logical or bitwise operations on one or more inputs. The shorthand used to denote these are listed below:

&	AND
|	OR
^|	exclusive-OR
<<	Shift left
>>	Shift right
~	Bitwise complement
!	Not function (performs bitwise conversion of ones to zeros and zeros to ones)
%	Modulus
ABS	Absolute value
EQ	Is equal to
GE	Is greater than or equal to
GT	Is greater than
LE	Is less than or equal to
LT	Is less than
NE	Is not equal to
SQRT	Square root

11.2 File structure

EZ Block Builder utilizes four different work directories. Each of these four directories is a subdirectory of the user's "My Documents\EZ Block Builder" directory.

By default, EZ Builder program files are stored in the "My Documents\EZ Block Builder\Configurations" directory.

By default, Alias list XML files are stored in "My Documents\EZ Block Builder\AliasLists" directory.

By default, Profile XML files are stored in "My Documents\EZ Block Builder\Profiles" directory.

Template XML files must reside in the "My Documents\EZ Block Builder\Templates" directory in order for EZ Block Builder to detect and load them into the toolbox on start-up.

The default installation directory will be the "Program Files\ABB\ABB EZ Block Builder" directory on the system drive although the user can override this and choose any directory name he wishes during program installation. The targeted installation directory will contain the EZ Block Builder's EXE file and supporting libraries. It will also contain a file named "AppTypes.xml". The AppTypes.xml file maps the names used for App Types in the EZ Builder with the application IDs used in the device. If required, this XML file may be hand-edited by the user.

11.3 Keyboard and Mouse Interaction

The following list describes keyboard and mouse functionality in the EZ Block Builder application:

- Keypad plus key (+) – Zooms in
- Keypad minus key (-) – Zooms out
- Ctrl-Z – Zooms page
- Delete key – Deletes the selected function block
- Left mouse button click on function block – Selects the function block
- Left mouse button click on page or work area background – De-selects currently selected function block
- Left mouse button double-click on Complex block – Drills-down into Complex block
- Left mouse button double-click on page or work area background – Zooms to page
- Left mouse button drag and drop – Drags function blocks from the toolbox and drops them on the work area. This is also used to move existing function blocks on the page.
- Right mouse button click – Displays context menu
- Middle (or mouse wheel) button drag and drop – Moves page on work area
- Mouse wheel roll forward – Zooms in while centered on mouse location
- Mouse wheel roll backward – Zooms out while centered on mouse location