Creating LEGO® MINDSTORMS® NXT Software Blocks

Contents

Introduction

You can use the LabVIEW Toolkit for LEGO® MINDSTORMS® NXT with LabVIEW 7.1 to create blocks that you can import into the LEGO MINDSTORMS NXT Software. Blocks are the objects you use to program NXT devices in the MINDSTORMS software. This document contains exercises that you can use to learn how to create and import new MINDSTORMS blocks.

Note Because the LEGO MINDSTORMS NXT software is based on version 7.1 of LabVIEW, you must use LabVIEW 7.1 to save the VIs for the MINDSTORMS blocks you create.

This document assumes that you have LabVIEW 7.1, the LabVIEW Toolkit for LEGO MINDSTORMS NXT, and that you are familiar with intermediate VI development in LabVIEW. Specifically, block creation involves the following LabVIEW concepts.

- Data types: integers, Booleans, strings, arrays, clusters, and variant data
- Basic functions used with the data types above
- Case, loop, sequence, and event structures
- Local variables
- VI Server

This document also assumes that you have the LEGO MINDSTORMS NXT software and are familiar with basic development for the NXT. To import blocks into MINDSTORMS software, you need to install the Dynamic Block Update, available online at `mindstorms.lego.com/support/updates`.

The LabVIEW Toolkit for LEGO MINDSTORMS NXT provides several template blocks with examples of functionality that you can use. These templates are located in the `LabVIEW\vi.lib\addons\NXTToolkit\Block Templates\` directory. To create your own block, you can start with an appropriate template block and modify it to meet your requirements.

Block Architecture

Creating a MINDSTORMS NXT block involves creating several required files and importing them into the MINDSTORMS software. The importation process relies on a strictly defined directory structure and involves string matching. Thus, to successfully import a block into the MINDSTORMS software you must follow the naming conventions specified in this document.

Note Throughout this document, `NewBlock` refers to the name you choose for your block.

Required Files

You must create the following files to import a block into the MINDSTORMS software.

Implementation VI

The Implementation VI is the file that the MINDSTORMS software actually compiles and downloads to the NXT. You must name your Implementation VI as follows.

`NewBlock\NewBlock.vi`

Configuration VI

The Configuration VI is what appears in the configuration pane at the bottom of the MINDSTORMS software. The Configuration VI provides several functions, including updating the default values on your Implementation VI and reading sensor data. You must name your Configuration VI as follows.

`NewBlock\Config NewBlock.vi`

Draw VI

The Draw VI updates your block's image on the MINDSTORMS block diagram. You must name your Draw VI as follows.

`NewBlock\Draw NewBlock.vi.`

Block Icon

The Block Icon is the image for your block that appears in the palette and on the block diagram. You must name your Block icon as follows.

`NewBlock\NewBlock.png`

Block Name File

The Block Name File contains the text that appears when you hover over your block in the MINDSTORMS palette. You must name the Block Name file as follows.

`NewBlock\NewBlock.txt`

Sensor File

Note The Sensor file is required only if you create a sensor block.

The contents of the Sensor File provide information that allows the Wait For, Loop, Switch, and Calibrate blocks to target the sensor block. You must name the Sensor File as follows.

`NewBlock\Sensor.ini`

Refer to the *Sensor File* section of *Additional Considerations* in this document for more information about Sensor files.

Optional Files

The following files are not required to successfully import a block into the MINDSTORMS software. However, you can include them to provide your block with additional functionality.

Version File

The Version File contains the version string for your block. This is a way for users to know what version of the block they have. You must name the Version File as follows.

`NewBlock\Version.txt`

Dragged Icon

The Dragged Icon is the image that appears when you drag your block from the palette to the block diagram. You must name your Dragged icon as follows.

`NewBlock\NewBlock Dragged.png`

Drawers File

The Drawers File describes the order in which MINDSTORMS displays the controls from your Implementation VI in the data hub that extends below the block in the workspace. You must name your Drawers File as follows.

`NewBlock\Drawers.dat`

Refer to the *Drawers* section of *Additional Considerations* in this document for more information about Drawers.

Drawer Images

The Drawer Images folder contains any additional drawer images needed for your block. The Drawer Images files must have a `.png` file extension and must reside in the following folder.

`NewBlock\Drawer Images\`

Refer to the *Drawers* section of *Additional Considerations* in this document for more information about Drawers.

Context Help File

The Context Help File is an HTML file where you can include quick reference information about your block. The MINDSTORMS software displays the Context Help File in the lower-right corner of the MINDSTORMS window when you hover the mouse over your block. You must name the Context Help File as follows.

`NewBlock\Help Content\Block_NewBlock.htm`

Any additional files needed for the Context Help File, such as images, must reside in the `NewBlock\Help Content\NewBlock Files\` folder.

Full Help

The Full Help File is an HTML file where you can include detailed information about your block. MINDSTORMS loads the Full Help File in an external web browser when you press F1 while hovering the mouse over your block. You must name your Full Help File as follows.

`NewBlock\Help Content\Full Help\topics\Help_NewBlock.htm`

Any additional files needed for the Full Help File, such as images, must reside in the `NewBlock\Help Content\Full Help\topics\NewBlock Files\` folder.

Global Name Sharing

The MINDSTORMS software, just like LabVIEW, cannot load two VIs with the same name. To avoid name collision, you must name all your VIs uniquely. You also must name all your drawer images uniquely to avoid name collision. Since other developers may produce blocks with the same name as yours, consider prepending your name or company name to ensure that your file names remain unique.

Create a Simple Sensor Block

Complete the following steps to create a simple light sensor block.

Create a New Block Using the New NXT Block Wizard

Note When you use the New NXT Block Wizard, LabVIEW automatically generates all the required files listed above, with appropriate names. The files contain templates based on the type of block you specify in the New NXT Block Wizard.

1. In LabVIEW, select **Tools»NXT Module»New NXT Block Wizard**.
2. Enter the name of your block in the **New Block Name** control.

Note Throughout this document, `NewBlock` refers to the name you choose for your block.

3. Select **Template Simple Sensor Block** from the **Template Block** listbox.
4. Click **Create** and browse to the folder where you want to save your new block.
5. Click **Current Folder** to create your block in the current folder.
6. Click **Close** to exit the New NXT Block Wizard and save your subVIs if prompted to do so.

Note You must choose **Save All SubVIs** if your version of LabVIEW prompts you to do so.

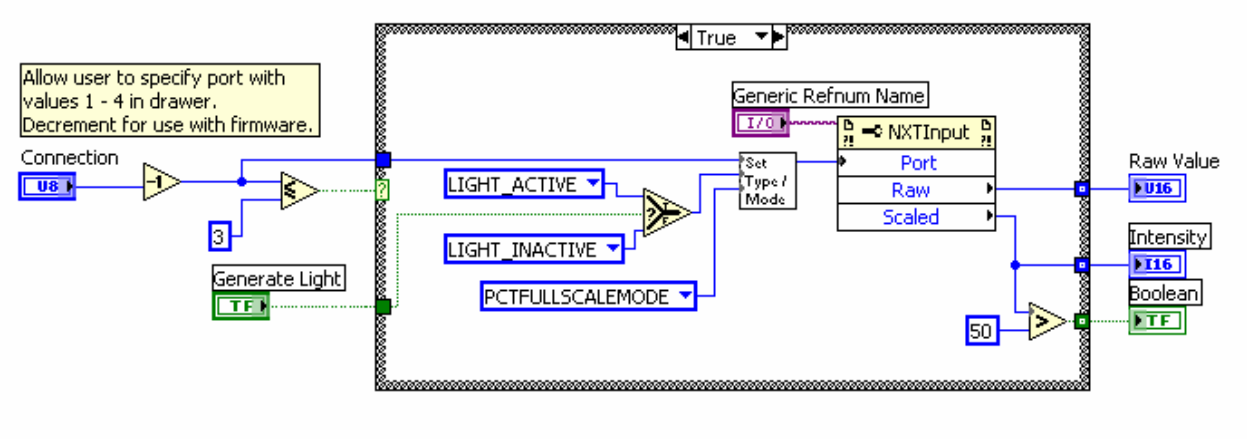
Modify the Implementation VI

Change the Implementation VI so that it targets a light sensor.

1. Open `NewBlock.vi` in LabVIEW.
2. On the block diagram, double-click to open the Simple Sensor Sub VI.
3. Modify the Simple Sensor Sub VI to target a light sensor and include a control that toggles whether the light sensor generates light.
   a. Create a Boolean control and label it `Generate Light`.
   b. Place a Select function on the block diagram.

c. Create a copy of the Sensor Type control, which is currently set to **LIGHT_INACTIVE**. Set the new Sensor Type control to **LIGHT_ACTIVE**. Wire the **LIGHT_ACTIVE** Sensor Type control to the **True** input of the Select function and the **LIGHT_INACTIVE** Sensor Type control to the **False** input of the Select function.

d. Add and remove wires as necessary. When you finish this step, the block diagram should appear similar to the figure below.



e. On the front panel, add the Generate Light control to the connector pane.

f. Save and close the Simple Sensor Sub VI and return to the Implementation VI.

4. Create a Boolean control, label it `Generate Light`, and wire it to the **Generate Light** input of the Simple Sensor Sub VI.

5. On the front panel, add the **Generate Light** control to the connector pane. The controls and indicators on the connector pane of the Implementation VI become the drawers in the data hub of your block in MINDSTORMS. Refer to the *Drawers* section of *Additional Considerations* in this document for more information about Drawers.

Note The MINDSTORMS software supports only numeric, Boolean, and string data types for drawers. To successfully import a block into MINDSTORMS, include on the connector pane of your Implementation VI only controls and indicators of supported data types.

6. Save and close the Implementation VI.

Modify the Configuration VI

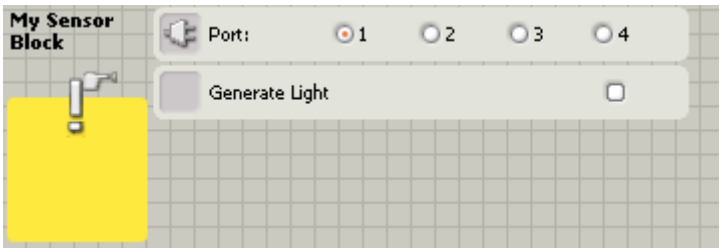
Change the Configuration VI so that it interacts with the **Generate Light** control on the Implementation VI.

1. Open the Configuration VI.
2. Change the text at the top left of the front panel from `Template Sensor Block` to `NewBlock`.
3. Add a Boolean control on the front panel that allows the user to configure whether the light sensor generates light.
   a. From the **All Controls** palette, select **NXTToolkit** to open the **NXTToolkit** palette.
   b. Select **NXTToolkit»NXT Controls»Config Decorations»One Row Background** and place a One Row Background on the front panel below the One Row Background labeled **Port**.
   c. Select **NXTToolkit»NXT Controls»NXT Checkbox** and place a checkbox inside the **Generate Light** One Row Background.

d. Rename the checkbox from `NXT Checkbox` to `Generate Light`.

e. On all block diagram labels, disable **Size to Text** and ensure that the labels contain extra room to accommodate different DPI and font size settings.

f. Ensure that all user interface objects are located in the upper-left corner of the front panel. When you finish this step, the front panel should appear similar to the following figure.



Note When you import your block, only objects in the upper-left corner of the front panel appear in the MINDSTORMS software.

4. Modify the Configuration VI to read the **Generate Light** value from the Implementation VI and populate the configuration pane with the correct values when your block is selected in the MINDSTORMS software.

   a. Open the Configuration VI.

   b. On the block diagram, find the **Get Default Values from Block** Sequence structure. Remove the wires connecting the two Invoke Nodes. Resize the Sequence structure and move the leftmost Invoke Node further left to make room for additional functions.

   c. Place a new Invoke Node between the Property Node and the existing Invoke Node. Wire the **VI Ref** output of the Property Node to the **reference** input of the new Invoke Node. On the new Invoke Node, select the **Get Control Value [Variant]** method.

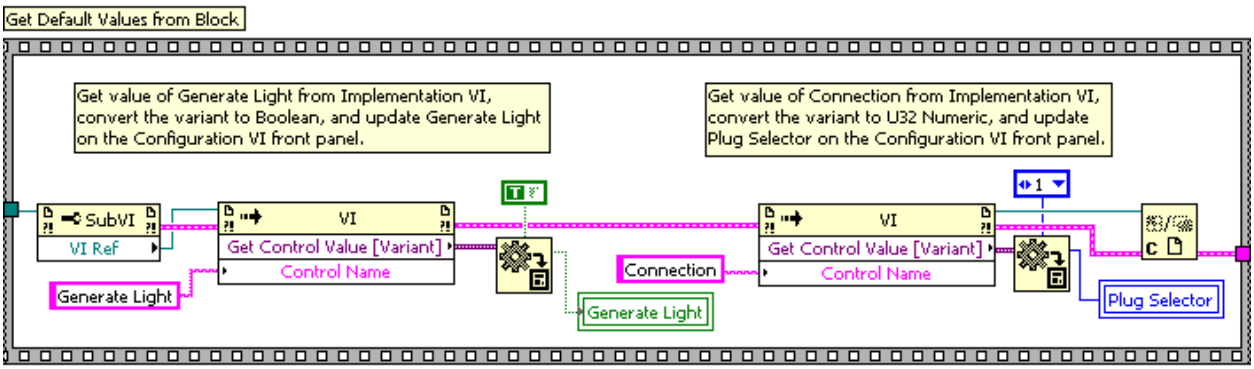
   d. Create a new string constant with the value `Generate Light` and wire it to the **Control Name** input of the new Invoke Node.

   e. Place a new **Variant to Data** function in the **Get Default Values from Block** Sequence structure and wire a **True** constant to the **type** input.

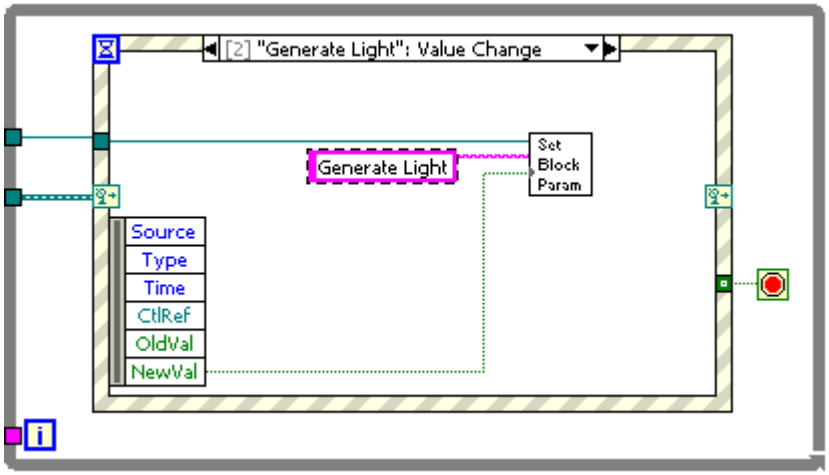
   f. Create a new local variable for the **Generate Light** checkbox control you added to the front panel.

   g. Wire the Variant to Data function's **data** output to the **Generate Light** variable.

   h. Add and delete wires as necessary. When you finish this step, the **Get Default Values from Block** Sequence structure should appear similar to the figure below.



5. Modify the Configuration VI to update the **Generate Light** control on your Implementation VI when a user updates the **Generate Light** checkbox.
   a. Find the Event structure to the right of the **Get Default Values from Block** Sequence structure and select the **Plug Selector** event case. Right-click the Event structure, and select **Duplicate Event Case** from the shortcut menu. Select **Generate Light** in the **Event Source** listbox. Then, select **Value Change** in the **Events** listbox. Click **OK** to save your changes and exit the dialog box.
   b. In the frame for the new **Generate Light** case, change the value of the string constant from `Connection` to `Generate Light`.
   c. When you finish this step, the Event structure should appear similar to the figure below.



6. Save and Close the Configuration VI.

Modify the Draw VI

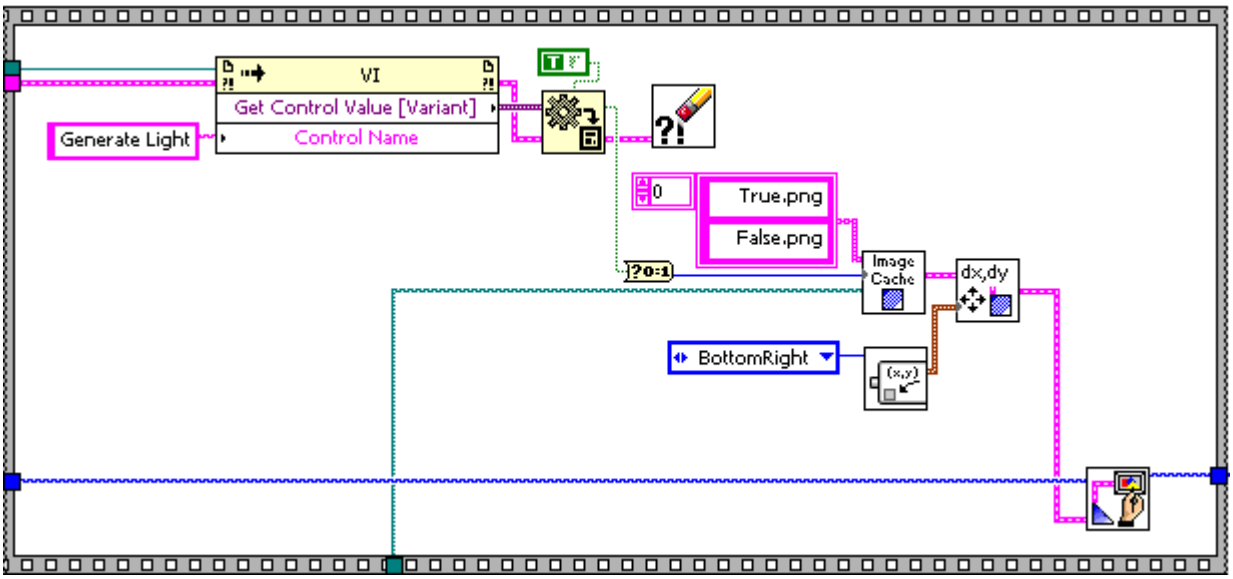
The Draw VI template adds the input port number to the Block Icon. In this exercise, you will modify the Draw VI so that it also adds a True/False image depending on the value of the **Generate Light** control in your Implementation VI.

1. Open the Draw VI.
2. Add functionality to the Draw VI so that it draws an image on the Block icon that depends on the Generate Light control in your Implementation VI. You can find the subVIs you need for the following steps in the `LabVIEW\vi.lib\addons\NXTToolkit\Block Templates\Support` directory.
   a. Find the sequence structure on the block diagram of the Draw VI. The sequence structure is where you will place your additional code.
   b. Place new Invoke Node and Variant to Data primitives in the sequence structure. Create a string constant and a Boolean constant. Enter the value `Generate Light` for the control you previously created on your Implementation VI. Place a Boolean to (0, 1) function on the block diagram to read the control value from your Implementation VI.
   c. Place NXT_ImageDataCache.vi as a subVI in the sequence structure. Create a constant for the **Image Name Array** input, and add **True.png** as the first element and **False.png** as the second element of the array. NXT_ImageDataCache.vi stores images in memory so that they only need to be read from the file once.

d. Place NXT_BlockParamOffset.vi as a subVI in the sequence structure. Create a constant for the **Pad Selector** input, and select **BottomRight**. The **Pad Selector** input determines the offset to draw the image in the bottom right corner.

e. Place OffsetImageCluster.vi and Draw Flattened Blended Pixmap.vi as subVIs in the sequence structure. OffsetImageCluster.vi shifts the image to the proper location and Draw Flattened Blended Pixmap.vi adds it to the Block Icon.

f. Add and remove wires as shown. When you finish this step, the sequence structure should appear similar to the figure below.



3. Save and close the Draw VI.

Import a Block to MINDSTORMS Software

Note To import blocks into MINDSTORMS software, you need to install the Dynamic Block Update, available online at `mindstorms.lego.com/support/updates`.

1. Launch the MINDSTORMS software and create a new program.
2. Select **Tools»Block Import and Export Wizard**.
3. Click **Browse**, and browse to either your block folder or a folder that contains several blocks.
4. Select the block(s) you want to import in the **Import** listbox.
5. Choose the palette to which you want to add the block(s).
6. Click **Import**.

Note If you include Drawer Images, they will not appear on the data hub of your block until you restart the MINDSTORMS software.

7. Verify that your block appears on the specified palette and behaves properly.

Note If your block is not selectable in the **Import** listbox, a portion of your block is either missing or broken. Make sure your block folder contains all the required files, and that the files are all named properly. Open your block VIs and ensure that they do not have broken run arrows. Also ensure that your block folder includes all the subVIs your block uses.

Additional Considerations

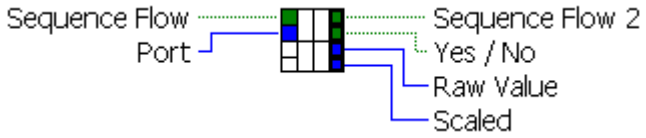
VI Instances

Every time a MINDSTORMS user places your block on the MINDSTORMS workspace, the MINDSTORMS software creates a new instance of the Implementation VI. Each instance of the block creates a dedicated copy of the code in memory. This means that instances of the block can run simultaneously without overwriting each other's data. Thus, when a Configuration VI modifies the values on an instance of the Implementation VI, other instances of that Implementation VI are not affected.

However, any non-reentrant subVIs you use in your Implementation VI can become shared resources among instances of the block. Thus, you reduce the per-instance cost of your block by placing static code in non-reentrant subVIs. Keep in mind that only one block can access the resources of a shared subVI at a time. If you want each instance of a block to have its own instance of a subVI, you must make the subVI reentrant.

Implementation VI Connector Pane

To successfully import your Implementation VI into the MINDSTORMS software, you must use the connector pane pattern shown in the figure below.



Any control or indicator that you want to appear in the drawers for a block must be on the connector pane of your Implementation VI. In addition, you must include `Sequence Flow` and `Sequence Flow 2` Booleans on the top left and top right corners respectively of your connector pane. If you are creating a sensor block, you also must include a `Yes / No` Boolean just below the `Sequence Flow 2` Boolean. The **Wait For**, **Loop**, and **Switch** blocks use the `Yes / No` Boolean to make decisions.

Note When you name the `Sequence Flow`, `Sequence Flow 2`, and `Yes / No` Booleans, you must name them exactly as they appear in this document, including spaces. The import process depends upon exact string matching.

Drawers

Drawers in the MINDSTORMS software are the connectors for a block. In the MINDSTORMS software, connectors are called plugs and are contained in the data hub that extends below a block in the workspace. The MINDSTORMS software supports only numeric, Boolean, and string data types for drawers. To successfully import a block into MINDSTORMS, include on the connector pane of your Implementation VI only controls and indicators of supported data types.

When you create a MINDSTORMS block, you can use the Drawers File to specify the functionality of its data. If you do not include a Drawers File in your block folder, the MINDSTORMS software automatically orders the controls and indicators on the connector pane of your Implementation VI in the block's data hub, but opens none of them when a user places the block. To specify functionality that differs from this default state, you must include a Drawers File in your block folder.

When you use the New NXT Block Wizard to create your block, the wizard places a Drawers File template file in your block folder. You can modify the Drawers File in a text editor to specify your own functionality. Under **[Drawers]** in the Drawers File, list the controls and indicators on the connector pane of your Implementation VI in the order you want them to appear in the MINDSTORMS software. Under

[OpenOnDrop], list the labels of the controls and indicators you want MINDSTORMS to automatically open when a user drops your block onto the diagram, in the order you want them to appear.

If you want to create custom images for your plugs, you must include them in the `NewBlock\Drawer Images\` directory. You also must name them exactly the same as the corresponding control or indicator label, with a `.png` file extension. For example, if you have a control labeled `MyControl`, you must name the corresponding image as follows.

```
NewBlock\Drawer Images\MyControl.png
```

If you do not include your own image files, MINDSTORMS displays the default image for the data type of the control. The text MINDSTORMS displays for the control or indicator in the drawer comes from the caption of the control, as opposed to the label, allowing you to provide text that differs from the required drawer image name.

Note MINDSTORMS Blocks share drawer images based on name. To avoid name collision, you must give your drawer images unique names.

Sensor File

If you create a sensor block and want it to interface with the Loop/Wait for/Switch sensor ring, you must include a Sensor File in your block's directory. The Sensor File must at least contain the following lines:

```
[Sensor]
Group=Custom
```

MINDSTORMS uses the above lines to populate your sensor into the ring controls on the **Wait For**, **Switch**, and **Loop** blocks. If your sensor is an analog sensor, and you want it to be listed in the **Calibrate** block and the **Calibrate Sensors** tool, you need to include the following additional lines in the Sensor File:

```
Calibrate=TRUE
SensorMode=128
SensorType=5
DefaultPort=3
PropIconFilename=CalLightSensor.png
```

You can specify your own values for **SensorMode**, **SensorType**, **DefaultPort**, and **PropIconFilename**.

Invalidate Host Structure in Sensor Blocks

The MINDSTORMS software uses the **TruePicture** and **FalsePicture** values from your Draw VI when drawing a Switch block that targets your sensor block. If you change the values of **TruePicture** and **FalsePicture**, you need to inform the Switch block to update its appearance. To do this, you need to modify your Configuration VI. Whenever you update a value on the Implementation VI in the event structure that affects the **TruePicture** and **FalsePicture** values, you should make a call to **NXT_Invalidate.vi** that passes the **HostStructRef**. You can find an example of this behavior in the Template Advanced Sensor Block.

True and False Images for the Switch Block

When you create a non-sensor block, you do not need to wire **TruePicture** and **FalsePicture** in the Draw VI. However, when you create a sensor block, you need to specify images for the **TruePicture** and **FalsePicture** indicators. When the Case structure depends upon a sensor reading in the MINDSTORMS software, the Draw VI draws **TruePicture** for the true case and **FalsePicture** for the false case.

Note Refer to the implementation of the Draw VI for the **Template Advanced Sensor** Block if you want to update the **TruePicture** and **FalsePicture** indicators based on control values.