

## FT 6000 Examples Guide

Install and run the FT 6000 EVK example applications using the FT 6000 EVB.

Echelon, LON, LonWorks, Neuron, 3120, 3150, Digital Home, i.LON, FTXL, LonScanner, LonSupport, LNS, LonMaker, LonMark, LonPoint, LonTalk, NodeBuilder, ShortStack, and the Echelon logo are trademarks of Echelon Corporation that may be registered in the United States and other countries.

Other brand and product names are trademarks or registered trademarks of their respective holders.

Neuron Chips and other OEM Products were not designed for use in equipment or systems which involve danger to human health or safety or a risk of property damage and Echelon assumes no responsibility or liability for use of the Neuron Chips or LonPoint Modules in such applications.

Parts manufactured by vendors other than Echelon and referenced in this document have been described for illustrative purposes only, and may not have been tested by Echelon. It is the responsibility of the customer to determine the suitability of these parts for each application.

ECHELON MAKES NO REPRESENTATION, WARRANTY, OR CONDITION OF ANY KIND, EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE OR IN ANY COMMUNICATION WITH YOU, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, FITNESS FOR ANY PARTICULAR PURPOSE, NONINFRINGEMENT, AND THEIR EQUIVALENTS.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Echelon Corporation.

Printed in the United States of America.  
Copyright ©1997–2009 by Echelon Corporation.  
Echelon Corporation  
[www.echelon.com](http://www.echelon.com)

# Contents

<b>Preface .....</b>	<b>v</b>
Purpose .....	vi
Audience .....	vi
System Requirements .....	vi
Content.....	vi
Related Manuals .....	vii
For More Information and Technical Support.....	viii
<b>1 Using the FT 6000 EVB Examples .....</b>	<b>1</b>
Introduction to the FT 6000 EVB Examples .....	2
Using the IzoT Commissioning Tool to Load Example Applications ....	7
Creating Connections in Managed Mode .....	15
Testing Switch and Lamp Devices .....	16
Testing Light, Temperature, LCD, and Joystick Devices .....	21
Creating Connections in ISI Mode .....	27
Getting Started with Developing Device Applications.....	29
Using the IzoT NodeBuilder Tool to Develop Device Applications.....	29
<b>2 Details of the FT 6000 EVB Examples .....</b>	<b>33</b>
Introduction to FT 6000 EVB Example Details .....	34
NcSimpleExample Details .....	34
Device Interface.....	34
Summary.....	34
Details.....	35
Node Object Functional Block .....	35
Switch Functional Block.....	35
Lamp Functional Block .....	36
Device Application Summary .....	37
I/O Interaction .....	38
NcSimpleIsciExample Details .....	39
Device Interface.....	39
Summary.....	39
Details.....	39
Node Object Functional Block .....	39
Switch Functional Blocks.....	40
Lamp Functional Blocks .....	41
Device Application Summary .....	41
I/O Interaction .....	43
ISI Mode .....	43
Managed Mode .....	44
NcMultiSensorExample Details.....	44
Device Interface.....	44
Summary.....	44
Details.....	46
Node Object Functional Block .....	46
Switch Functional Blocks.....	48
Lamp Functional Blocks .....	48

Light Sensor Functional Block .....	49
Temperature Sensor Functional Block.....	49
Joystick Functional Block .....	51
Device Application Summary .....	52
Using the Joystick and LCD .....	55
Welcome Panel.....	55
Local Info Mode Panel.....	55
Remote Info Mode Panel.....	56
Alarm Config Mode Panel .....	57
I/O Interaction .....	57
ISI Mode .....	58
Managed Mode .....	58
<b>Appendix A Glossary .....</b>	<b>61</b>

# Preface

The FT 6000 Evaluation Kit includes three Neuron C example applications that you can run on your FT 6000 EVBs. You can use these examples to test the I/O devices on the FT 6000 EVBs, and create simple managed and self-installed LONWORKS<sup>®</sup> and IzoT networks. You can browse the Neuron<sup>®</sup> C code used by these examples to learn how to develop your own device applications.

---

## Purpose

This document describes how to run the example applications included with the FT 6000 EVB.

---

## Audience

This guide is intended for device and system designers with an understanding of control networks.

---

## System Requirements

Requirements for computers running the FT 6000 EVB examples are listed below:

- Microsoft® Windows. Windows 8 64-bit and 32-bit or Windows 7 54-bit and 32-bit or Windows XP 32-bit. Echelon recommends that you install the latest service pack available from Microsoft for your version of Windows.
- Intel® Pentium® III 600MHz processor or faster, and meeting the minimum Windows requirements for the selected version of Windows.
- 300 to 550 megabytes (MB) free hard-disk space, plus the minimum Windows requirements for the selected version of Windows.
  - The IzoT NodeBuilder tool requires 100 MB of free space. The IzoT Commissioning Tool, which is included with the IzoT FT 6000 EVK software and is required to install the IzoT NodeBuilder tool, requires 172 MB of free space.
  - IzoT Plug-in for WireShark, which explains how to plug into the freely accessible WireShark packet analyzer. Microsoft .NET Framework 3.5 SP1, which is required to run the IzoT NodeBuilder tool, requires 30 MB of free space.
  - If you install Acrobat® Reader 9.1, you need an additional 204 MB of free space.
- 512 MB RAM minimum.
- CD-ROM drive.
- 1024x768 or higher-resolution display with at least 256 colors.
- Mouse or compatible pointing device.
- IzoT NodeBuilder FX tool. Running the FT 6000 EVB examples in managed mode requires the IzoT Commissioning tool or other network tool. The IzoT Commissioning tool is included with the IzoT FT 6000 EVK..
- IzoT router

**Note:** You must run the IzoT NodeBuilder software on the same computer with the IzoT Network Services Server which is installed by the IzoT Commissioning Tool installer. You cannot run the IzoT NodeBuilder tool as a remote client to an IzoT Network Services Server running on another computer.

---

## Content

This guide includes the following content:

- *Using the FT 6000 EVB Examples.* Introduces the three Neuron C example applications that you can run on an FT 6000 EVB. Describes how to use the IzoT Commissioning tool to load these example applications on an FT 6000 EVB. Describes how to bind the example applications in a

- self-installed or managed network. Explains how to browse the Neuron C code used by the example applications so that you can begin developing your own device applications.
- *Details of the FT 6000 EVB Examples.* Illustrates and details the device interfaces, device applications, and I/O devices used by the FT 6000 EVB examples.
  - *Glossary.* Provides definitions for many terms commonly used with the FT 6000 EVB example applications

---

## Related Manuals

The documentation related to the IzoT NodeBuilder tool is provided as Adobe Acrobat PDF files and online help files. The PDF files for the IzoT NodeBuilder tool are installed in the **Echelon NodeBuilder** program folder when you install the IzoT NodeBuilder tool. You can download the latest IzoT NodeBuilder, including the latest version of this guide, from Echelon's Web site at [www.echelon.com/docs](http://www.echelon.com/docs).

<i>FT 6000 EVB Hardware Guide (078-0504-01)</i>	Describes how to connect the FT 6000 EVB boards, and describes the Neuron core, I/O devices, service pin and reset buttons and LEDs, and jumper settings on the FT 6000 EVB hardware.
<i>Introduction to the LONWORKS® Platform</i>	Provides a high-level introduction to LONWORKS networks and the tools and components that are used for developing, installing, operating, and maintaining them.
<i>ISI Programmer's Guide (078-0299-01F)</i>	Describes the ISI protocol, which provides for easy development of devices that do not require installation tools. You can run the <b>NcSimpleIsiExample</b> and <b>NcMultiSensorExample</b> examples on the FT 6000 EVB in ISI mode.
<i>OpenLNS® Plug-in Programmer's Guide</i>	Describes how to write plug-ins using .NET programming languages such as C# and Visual Basic .NET
<i>IzoT Commissioning Tool User's Guide (078-0509-01)</i>	Describes how to use the IzoT CommissioningTool to design, commission, modify, and maintain LONWORKS networks.
<i>LONMARK® SNVT and SCPT Guide</i>	Documents the standard network variable types (SNVTs), standard configuration property types (SCPTs), and standard enumeration types that you can declare in your applications.
<i>Neuron® C Programmer's Guide (078-0002-02I)</i>	Describes how to write programs using the Neuron® C Version 2.2 language.
<i>Neuron® C Reference Guide (078-0140-02G)</i>	Provides reference information for writing programs using the Neuron C language.
<i>Neuron® Tools Error Guide</i>	Provides reference information for Neuron C errors.
<i>IzoT NodeBuilder® FX User's Guide (078-0402-01D)</i>	Describes how to use the IzoT NodeBuilder tool to develop LONWORKS device applications and build and test prototype and production LONWORKS devices
<i>IzoT Resource Editor User's Guide (078-0508-01)</i>	Describes how to use the IzoT Resource Editor to create and edit resource file sets and resources such as functional profile templates, network variable types, and configuration property types.

---

## For More Information and Technical Support

Free e-mail support is available or you can purchase phone support from Echelon or an Echelon support partner. See [www.echelon.com/support](http://www.echelon.com/support) for more information on Echelon support and training services.

You can also view free online training or enroll in training classes at Echelon or an Echelon training center to learn more about developing devices. You can find additional information about device development training at [www.echelon.com/training](http://www.echelon.com/training).

You can obtain technical support via phone, fax, or e-mail from your closest Echelon support center. The contact information is as follows (check [www.echelon.com/support](http://www.echelon.com/support) for updates to this information):

Region	Languages Supported	Contact Information
The Americas	English Japanese	Echelon Corporation Attn. Customer Support 550 Meridian Avenue San Jose, CA 95126 Phone (toll-free): 1-800-258-4LON (258-4566) Phone: +1-408-938-5200 Fax: +1-408-790-3801 <a href="mailto:lonsupport@echelon.com">lonsupport@echelon.com</a>
Europe	English German French Italian	Echelon Europe Ltd. Suite 12 Building 6 Croxley Green Business Park Hatters Lane Watford Hertfordshire WD18 8YH United Kingdom Phone: +44 (0)1923 430200 Fax: +44 (0)1923 430300 <a href="mailto:lonsupport@echelon.co.uk">lonsupport@echelon.co.uk</a>
Japan	Japanese	Echelon Japan Holland Hills Mori Tower, 18F 5-11-2 Toranomom, Minato-ku Tokyo 105-0001 Japan Phone: +81-3-5733-3320 Fax: +81-3-5733-3321 <a href="mailto:lonsupport@echelon.co.jp">lonsupport@echelon.co.jp</a>
China	Chinese English	Echelon Greater China Rm. 1007-1008, IBM Tower Pacific Century Place 2A Gong Ti Bei Lu Chaoyang District Beijing 100027, China Phone: +86-10-6539-3750 Fax: +86-10-6539-3754 <a href="mailto:lonsupport@echelon.com.cn">lonsupport@echelon.com.cn</a>



Region	Languages Supported	Contact Information
Other Regions	English Japanese	Phone: +1-408-938-5200 Fax: +1-408-328-3801 <a href="mailto:lonsupport@echelon.com">lonsupport@echelon.com</a>



# Using the FT 6000 EVB Examples

This chapter introduces the three Neuron C example applications that you can run on an FT 6000 EVB. It describes how to load these example applications on an FT 6000 EVB using the IzoT Commissioning Tool which is included with the IzoT FT 6000 EVK. It describes how to bind the example applications in a self-installed or managed network. It explains how to browse the Neuron C code used by these examples so that you can begin developing your own device applications.

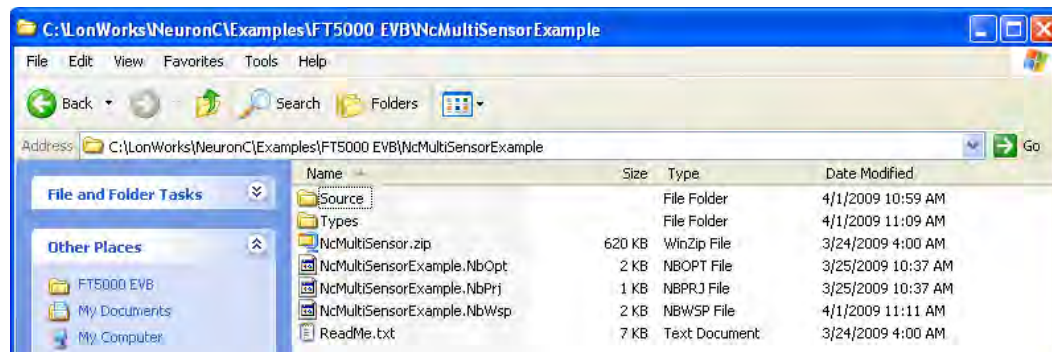
---

## Introduction to the FT 6000 EVB Examples

The IzoT NodeBuilder FX/FT Development Tool includes three Neuron C example applications that you can run on your FT 5000 EVBs: *NcSimpleExample*, *NcSimpleIsiExample*, and *NcMultiSensorExample*. You can use these example applications to test the I/O devices on the FT 6000 EVBs, and create simple managed and self-installed LONWORKS networks.

The *NcMultiSensorExample* application is pre-loaded on the FT 6000 EVBs and runs in Interoperable Self-Installation (ISI) mode by default. This means that out-of-the-box, you can install and connect the network variables of the *NcMultiSensorExample* application using the ISI protocol (see *Creating Connections in ISI Mode* later in this chapter for more information). You can install and bind the *NcMultiSensorExample* example in a managed network by commissioning it with the IzoT Commissioning tool or other network tool (see *Using the IzoT Commissioning Tool to Load Example Applications* later in this chapter for more information).

The FT 6000 EVB examples are stored in separate folders within the **LonWorks\NeuronC\Examples\FT6000 EVB** directory (for example, the *NcMultiSensorExample* application is stored in the **LonWorks\NeuronC\Examples\FT6000 EVB\NcMultiSensorExample** folder). Note that the default LONWORKS folder on your computer is typically C:\LonWorks or C:\Program Files\LonWorks.



Each example folder contains the following files and subfolders:

- |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|---------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Source</b>                         | This folder contains the example IzoT NodeBuilder project and all source code files and header files used by the example.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>Types</b>                          | In the <b>NcMultiSensorExample</b> folder only, this folder contains definitions for the user-defined functional profiles (UFPTs) developed for the example.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <b>LonMaker Network Backup (.zip)</b> | <p>An IzoT Commissioning Tool network backup file (<b>.zip</b>) that includes an OpenLNS database and LonMaker drawing containing the example device and all the functional blocks and network variables in the device's external interface. You can restore this backup file with the OpenLNS tool.</p> <p>After you restore the network, you can use the IzoT Commissioning tool to download the example application to your FT 6000 EVBs. Each example application includes a pre-built binary application image file (<b>.APB</b> extension) that is stored in the <b>LonWorks\NeuronC\Examples\FT6000 EVB\ReleasedBinaries\&lt;Example&gt;</b> folder. This folder also contains a pre-built text device interface file (<b>.XIF</b> extension) that exposes the example application's device interface so that the IzoT Commissioning tool can manage the example application.</p> |

See *Using the IzoT Commissioning Tool to Load Example Applications* later in this chapter for how to restore the LonMaker network backup and download the example applications to the FT 6000 EVBs.

After you restore the backup and load the example application, you can create a simple managed LONWORKS network (see *Creating Connections in Managed Mode* later in this chapter for how to do this).

#### **NodeBuilder Project Files (.NbOpt and .NbPrj)**

Each example includes a NodeBuilder project that you can open with the IzoT NodeBuilder tool in order to browse the example applications and learn how to develop your own device applications. The NodeBuilder project includes the following files:

- **Options File (\*.NbOpt).** Contains the NodeBuilder project options for a project. There is one options file per project.
- **Project File (\*.NbPrj).** Contains a project definition including the project version and a list of the device templates and the hardware templates for a project. There is one project file per project.

#### **ReadMe (.txt)**

A text file (.txt extension) summarizing the functionality of the example application and the device interface and Neuron C code used by it.

You can install, load, commission, and connect the network variables of all three example applications using the IzoT Commissioning Tool or other network tool. Each example application includes a LonMaker network backup (LonMaker drawing and OpenLNS network database) that you can restore to begin running the example applications in a managed network. The LonMaker drawing backup includes a device shape for the example application that you can commission, and functional block and network variable shapes for the functional blocks and network variables defined in the device interface.

In addition, you can install and connect the network variables of the *NcSimpleIsiExample* and *NcMultiSensorExample* applications using the Interoperable Self-Installation (ISI) protocol. ISI is an application-layer protocol that lets you install and connect devices without using a separate network management tool. For more information on ISI and developing an application using the Neuron ISI library, see the *ISI Protocol Specification* and *ISI Programmer's Guide*.

The *NcSimpleIsiExample* and *NcMultiSensorExample* applications start in ISI mode by default. You can run these examples in managed mode by installing them with the IzoT Commissioning Tool or other OpenLNS network management tool. To switch back to ISI mode, you can use the IzoT Commissioning tool to change the **SCPTnwkCnfg** configuration property in the application's **Node Object** functional block to **CFG\_LOCAL** (you can also do this using the **nciNetConfig** configuration property in the application's **Virtual Functional Block**). Alternatively, you can hold down the service pin on the FT 6000 EVB for approximately 10 seconds, which resets the applications to their factory default settings.

The following table summarizes the *NcSimpleExample*, *NcSimpleIsiExample*, and *NcMultiSensorExample* applications:

Example Application	Description	
<i>NcSimpleExample</i>	<b>Summary</b>	Demonstrates how you can use switch devices to activate lamp devices in a managed network. It uses one push button ( <b>SW1</b> ) that represents a switch device, and one LED ( <b>LED1</b> ) that represents a lamp device.
	<b>Device Interface</b>	Includes a Node Object functional block, and Switch and Lamp functional blocks representing the push button and LED I/O objects on the FT 6000 EVB. Both of the Switch and Lamp functional blocks contain <b>SNVT_switch</b> input and output network variables.
	<b>Installation Mode</b>	You must use IzoT Commissioning Tool or other network tool to commission the <b>SimpleExample</b> device and to connect the Switch and Lamp network variables so that pressing the push button illuminates and extinguishes the LED.
	<b>Program ID</b>	9F:FF:FF:05:01:04:04:10
<i>NcSimpleIsiExample</i>	<b>Summary</b>	<p>Demonstrates how you can use switch devices to activate lamp devices in a self-installed or managed network.</p> <p><b><u>ISI Mode</u></b></p> <p>In self-installed mode (ISI mode), this example uses one push button (<b>SW1</b>) that represents a switch device, one LED (<b>LED1</b>) that represents a lamp device, a push button (<b>SW2</b>) to initiate and complete an ISI connection, and an LED (<b>LED2</b>) that indicates the connection status.</p> <p><b><u>Managed Mode</u></b></p> <p>In managed mode, this example uses two push buttons (<b>SW1</b> and <b>SW2</b>) that represent switch devices and two LEDs (<b>LED1</b> and <b>LED2</b>) that represent lamp devices.</p>
	<b>Device Interface</b>	<p>Includes a Node Object functional block, an array of two Switch functional blocks representing the push button I/O objects on the FT 6000 EVB, and an array of two Lamp functional blocks representing the LED I/O objects on the FT 6000 EVB.</p> <p>The Node Object functional block contains a <b>SCPTnwrkCnfg</b> configuration property that stores the current network configuration mode (ISI or managed).</p> <p>The Switch and Lamp functional blocks contain <b>SNVT_switch</b> input and output network variables.</p>
	<b>Installation Mode</b>	You can run the <b>SimpleIsiExample</b> in ISI or managed mode. In managed mode, you must use the IzoT Commissioning Tool or other network tool to commission the <b>SimpleISIExample</b> device and to connect the Switch and Lamp network variables so that pressing the push buttons illuminate and extinguish the LEDs.
	<b>Program ID</b>	9F:FF:FF:05:01:04:04:20

Example Application	Description	
<i>NcMultiSensorExample</i>	<b>Summary</b>	<p>Demonstrates how you can use switch devices to activate lamp devices in a self-installed or managed network. For managed networks, it also demonstrates how you can use light-level sensor, temperature sensor, joystick, and LCD devices to view the current temperature, light level (lux), and alarm conditions of a local or remote device and set light and temperature alarm conditions.</p> <p>A local device refers to one FT 6000 EVB running the <i>NcMultiSensorExample</i> application. A remote device refers to another device containing <b>SNVT_lux</b> and/or <b>SNVT_temp_p</b> output network variables that are connected to the <b>SNVT_lux</b> and/or <b>SNVT_temp_p</b> input network variables on the local device. A remote device may be a second FT 6000 EVB running the <i>NcMultiSensorExample</i> application. You can use a local device to monitor the temperature, light level, and alarm conditions of a remote device.</p> <p><b><u>ISI Mode</u></b></p> <p>In ISI mode, this example is identical to the <i>NcSimpleIsiExample</i> application. It uses one push button (<b>SW1</b>) that represents a switch device, one LED (<b>LED1</b>) that represents a lamp device, a push button (<b>SW2</b>) to initiate and complete an ISI connection, and an LED (<b>LED2</b>) that indicates the connection status.</p> <p><b><u>Managed Mode</u></b></p> <p>In Managed mode, this example uses two push buttons (<b>SW1</b> and <b>SW2</b>) that represent switch devices and two LEDs (<b>LED1</b> and <b>LED2</b>) that represent lamp devices, a temperature sensor, a light level sensor, an LCD display, and a joystick used to toggle the information displayed on the LCD and to enter set points for light and temperature alarms.</p>
	<b>Device Interface</b>	<p>Includes the following functional blocks:</p> <ul style="list-style-type: none"> <li>• A Node Object functional block that contains <b>SNVT_lux</b> and <b>SNVT_temp_p</b> input network variables storing the light and temperature values received from the remote device, and <b>SNVT_alarm_2</b> output network variables storing the alarm statuses of the local and remote devices. The Node Object also contains a <b>SCPTnwrkCnfg</b> configuration property that stores the current network configuration mode (ISI or managed).</li> <li>• An array of two Switch functional blocks representing the push button I/O objects and an array of two Lamp functional blocks representing the LED I/O objects on the FT 6000 EVB. The Switch and Lamp functional blocks contain <b>SNVT_switch</b> input and output network variables.</li> </ul>

Example Application	Description	
		<ul style="list-style-type: none"> <li>• A LightSensor functional block representing the light-level sensor I/O object on the FT 6000 EVB. The LightSensor functional block includes a <b>SNVT_lux</b> output network variable, and a <b>SCPTluxSetpoint</b> configuration property that stores the set point for the light alarm's low limit.</li> <li>• A TempSensor functional block representing the temperature sensor I/O object on the FT 6000 EVB. The TempSensor functional block includes a <b>SNVT_temp_p</b> output network variable, and a <b>SCPThighLimTemp</b> configuration property that stores the set point for the temperature alarm's high limit.</li> <li>• A Joystick functional block representing the joystick I/O object on the FT 6000 EVB. The Joystick functional block includes a <b>SNVT_angle_deg</b> output network variable that you can change to a <b>SNVT_switch</b> type, and a <b>SCPTnvType</b> configuration property.</li> </ul> <p><b>Note:</b> The <b>SCPTnwrkCnfg</b>, <b>SCPThighLimTemp</b>, <b>SCPTluxSetpoint</b>, and <b>SCPTnvType</b> configuration properties are implemented as configuration network variables (CPNVs). As a result, these network variables appear with an “nci” prefix in a Virtual functional block—instead of in their parent functional blocks—when you are using the <i>NcMultiSensorExample</i> device in the IzoT Commissioning tool or other network tool.</p>
	<b>Installation Mode</b>	You can run the <b>MultiSensorExample</b> in ISI or managed mode. In managed mode, you must use the IzoT Commissioning Tool or other network tool to commission the <b>MultiSensorExample</b> device and to connect the network variables so that the example application responds to the I/O devices on the FT 6000 EVB.
	<b>Program ID</b>	9F:FF:FF:05:01:84:04:30

You can download the example applications to the FT 6000 EVBs using the IzoT Commissioning Tool, which is included with the IzoT FT 6000 EVK. After you download an example application to the FT 6000 EVBs, you can install and connect the network variables of the example applications in a self-installed or managed network. The following sections describe how to do the following:

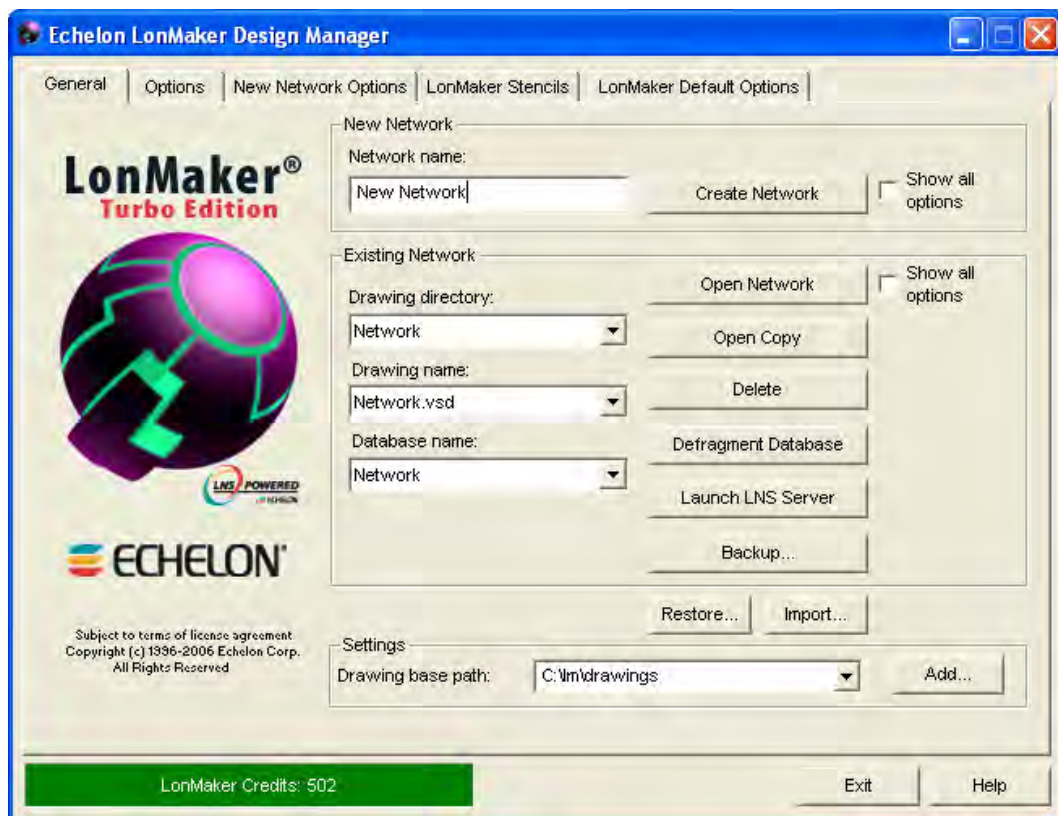
1. Download the example applications with the IzoT Commissioning Tool.
2. Create network variable connections in a managed network.
3. Create network variable connections in a self-installed network.



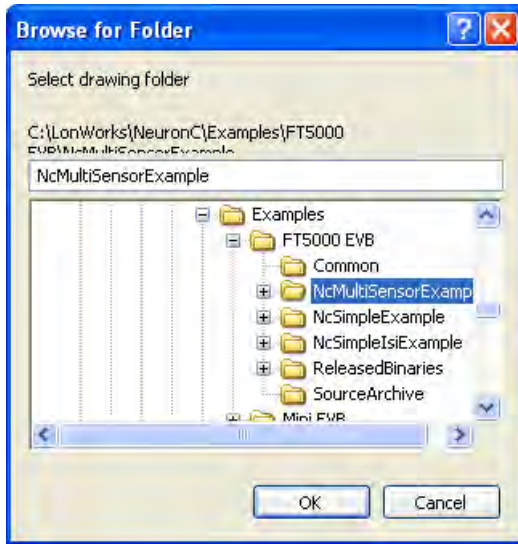
## Using the IzoT Commissioning Tool to Load Example Applications

You can use the IzoT Commissioning Tool to download the example applications to the FT 6000 EVBs and install them in a LONWORKS network. To do this, you restore an IzoT Commissioning tool network backup that contains device and functional block shapes for that example application, load the pre-built binary application image file (.APB extension) for the example application to the device, and then commission the example device following these steps:

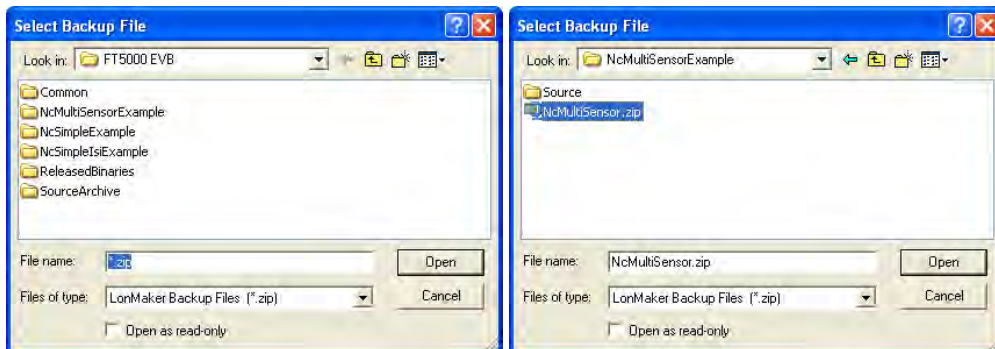
1. Verify that you have installed and activated the IzoT Commissioning Tool following Chapter 2 of the *IzoT Commissioning Tool User's Guide*.
2. Connect your FT 6000 EVBs following Chapter 1 of the *FT 6000 EVB Hardware Guide*.
3. Start the IzoT Commissioning Tool. To do this, click **Start** on the taskbar, point to **Programs**, point to the **Echelon LonMaker** folder, and then click **LonMaker**. The LonMaker Design Manager opens.



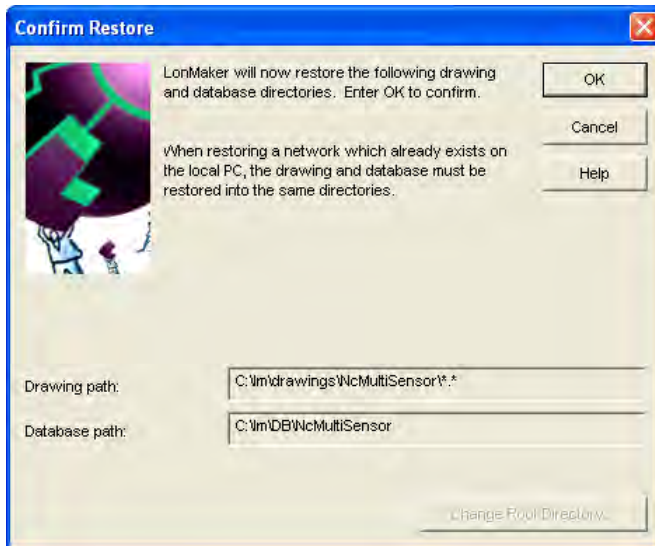
4. Optionally, you can add the folder containing the IzoT Commissioning Tool network backup files for the example applications to the drawing base path. This speeds up the process of restoring the backup files for the other example applications. To do this, click **Add** under **Settings**. The **Browse for Folder** dialog opens. Browse to the LonWorks\NeuronC\Examples\FT6000 EVB folder, click any of the three example folders, which start with "Nc", and then click **OK**.



5. Click **Restore**. The **Select Backup File** dialog opens. Double-click the folder containing the example application to be loaded, and then double-click the IzoT Commissioning Tool backup file (.zip extension).

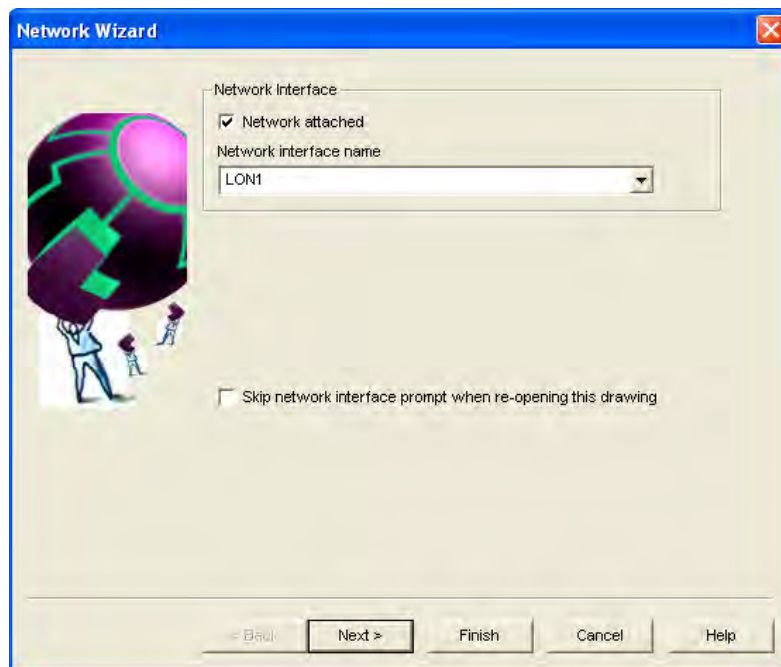


6. The **Confirm Restore** dialog opens.

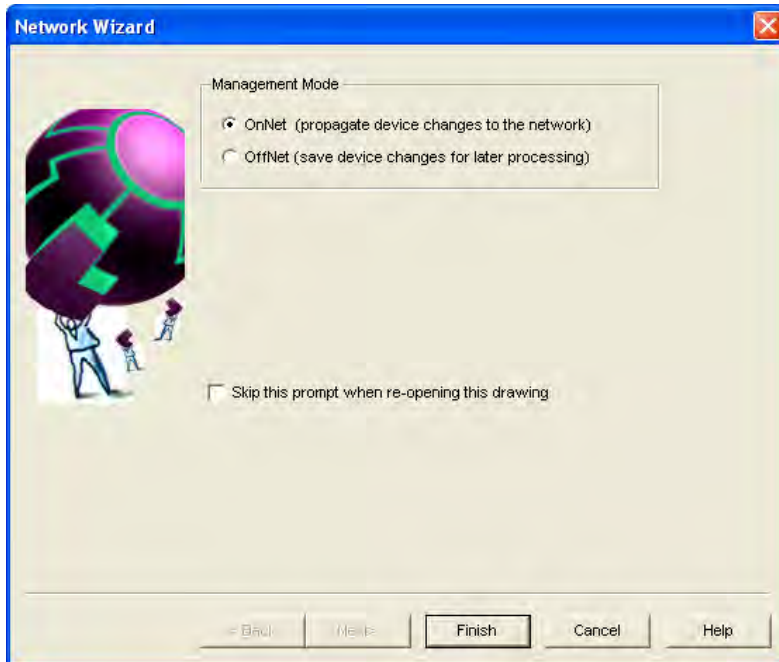


7. Click **OK**.

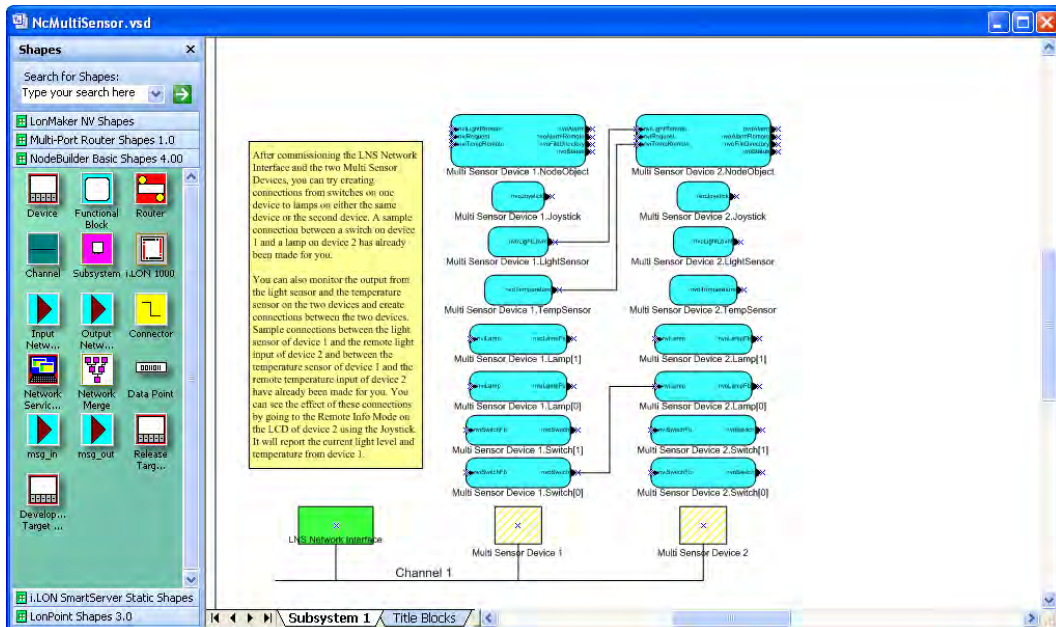
8. By default, the IzoT Commissioning Tool will prompt you to select whether to install any new files in the Import folder (includes LONMARK<sup>®</sup> resource files) and then any new files in the Types folder (includes XIF and application image files [.APB extension]). Click **Yes** to restore the files.
9. A message appears informing you that the network restore operation has been completed, and prompting you to select whether to open the LonMaker network in order to recommission devices that have changed since the network was backed up. Click **Yes**.
  - A message may appear informing you that Visio must be launched and initialized so that it can work with the IzoT Commissioning Tool. Click **OK**.
  - A warning may appear asking you if you want to enable macros. You must enable macros for the IzoT Commissioning tool to function.
10. The Network Wizard opens with the Network Interface window displayed.



11. Select the **Network Attached** check box. In the **Network Interface Name** property, select the network interface to be used for communication between IzoT Commissioning Tool and the FT 6000 EVBs over the LONWORKS channel. Click **Next**.
12. The Management Mode window opens.



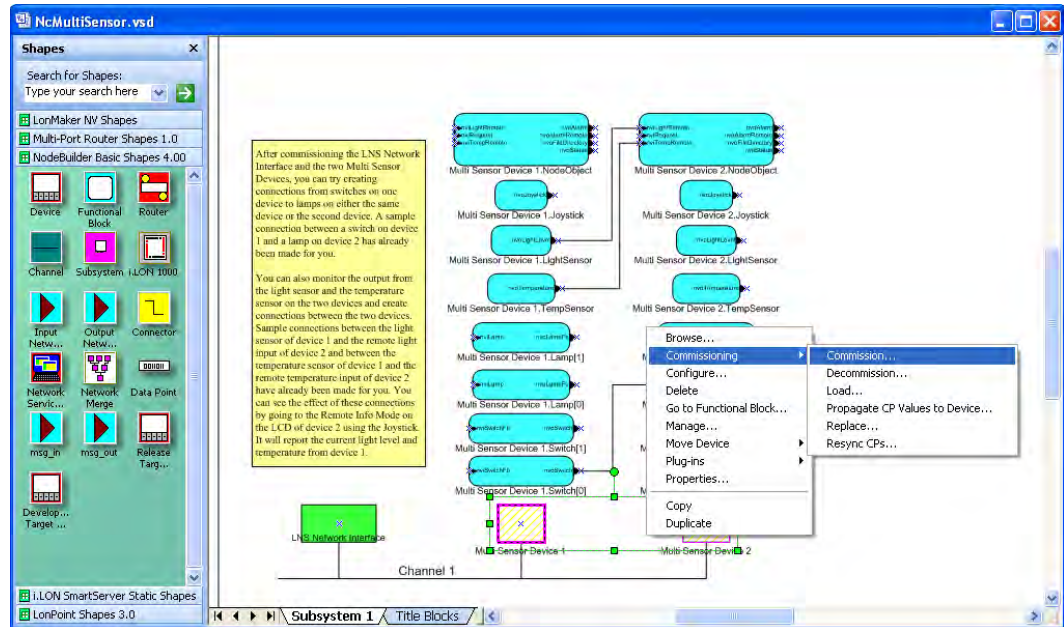
13. Select **OnNet** to immediately propagate changes you make to the example device in the LonMaker drawing to the physical device on the network. Click **Finish**.
14. A message appears recommending that you recommission devices that have changed since the network was backed up. Click **No**.
15. The LonMaker drawing for the example application opens. The LonMaker drawing includes a commissioned OpenLNS Network Interface device shape, two uncommissioned device shapes, and functional block and network variable shapes for all the functional blocks and network variables defined in the device interface.



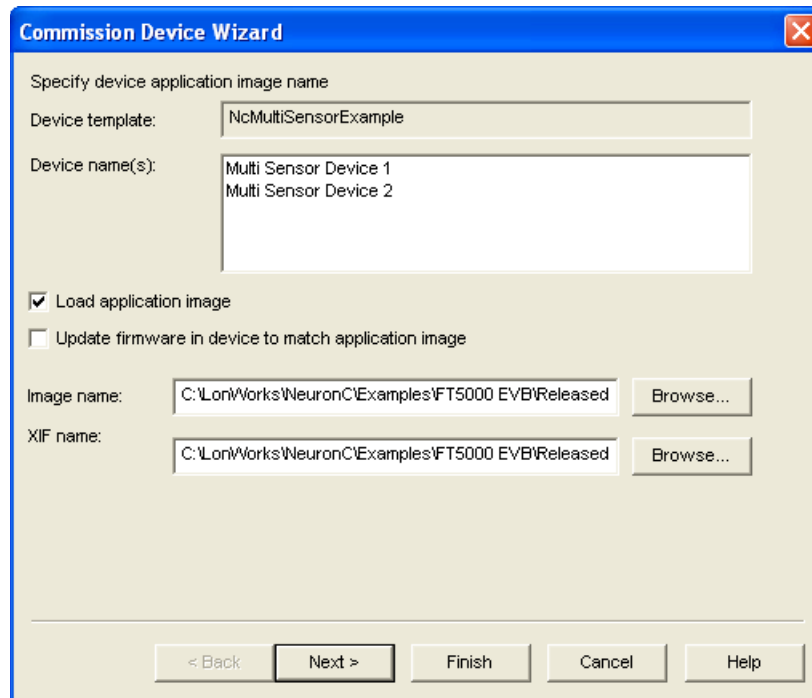
16. Download the example application to the FT 6000 EVBs following these steps:
  - a. Hold down CTRL and click both yellow cross-hatched device shapes representing your uncommissioned example devices (alternatively, you can click an empty space in the drawing



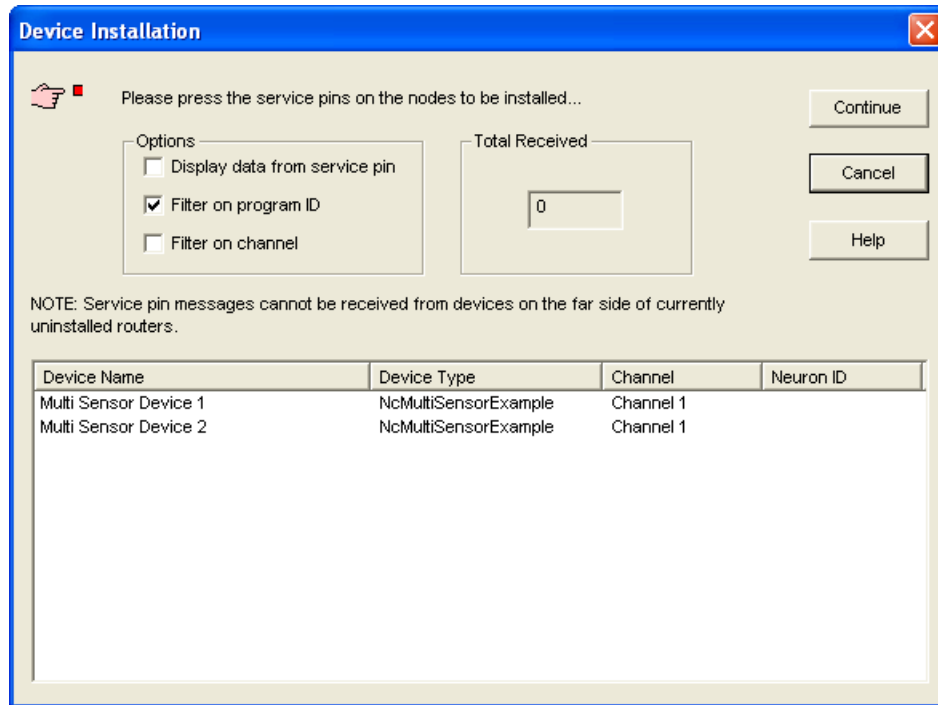
page and drag a selection net around the device shapes), right-click one of the selected devices, point to **Commissioning**, and then click **Commission** on the shortcut menu.



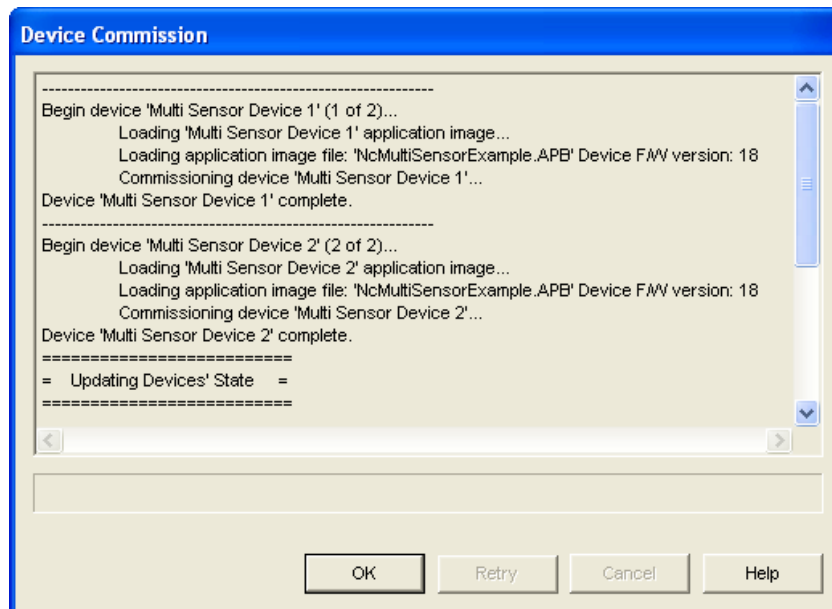
- b. The Commission Device Wizard opens with the Application Image window displayed. Select the **Load Application Image** check box. This specifies that you will download the pre-built binary application image file (.APB extension) for the example application to the device. The pre-built binary application image files for the example applications are stored in the **LonWorks\NeuronC\Examples\FT6000 EVB\ReleasedBinaries\<Example>** folder.



- c. Click **Finish**. The **Device Installation** dialog opens.

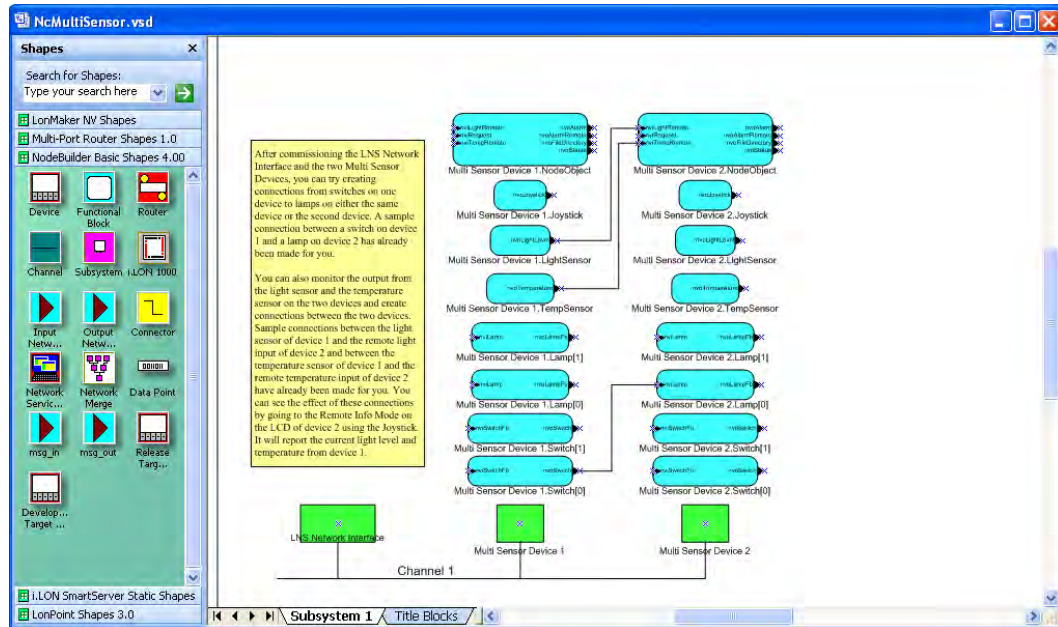


- d. Press the Service button on each FT 6000 EVB. The Service button on the FT 6000 EVB is a black button that is located near the upper right-hand corner of the board and is labeled “Service.”
- e. The **Device Commission** dialog opens. This dialog lists application image loading, commissioning, and device state events in the order they occur.



- f. After both example devices have been commissioned, click **OK** to return to the LonMaker drawing.
17. The device shapes are solid green, indicating that the devices have been commissioned and are online, and the LCD on the FT 6000 EVBs display the name of the example application you

loaded. The device applications will not do anything until you test the devices or connect them to other devices.

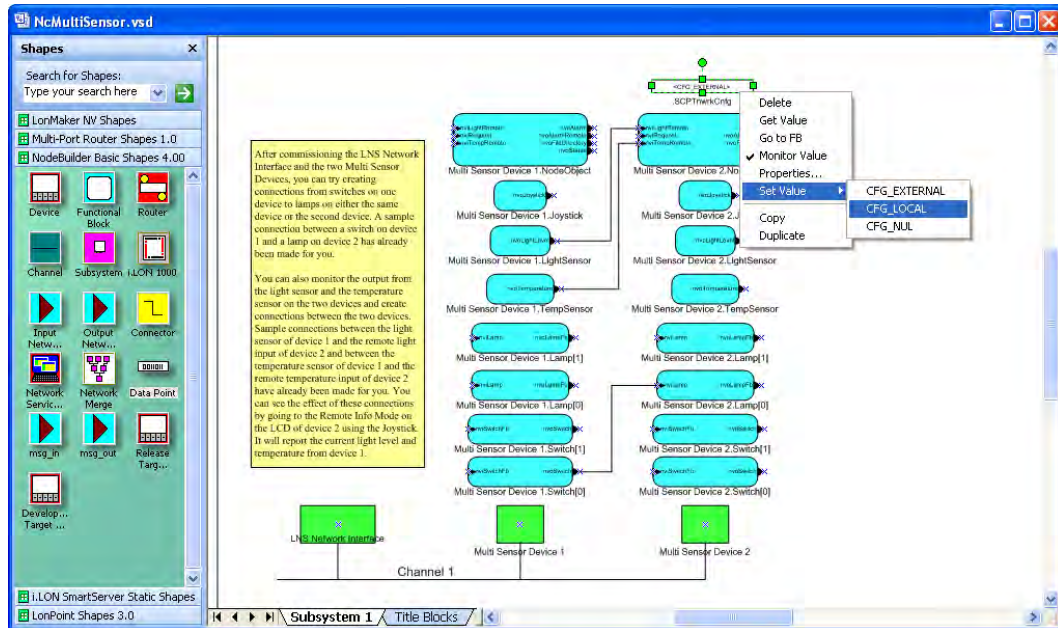


**Note:** Instead of running the same example application on both FT 6000 EVBs, you can create a second device running a different FT 6000 EVB example application or your own device application. In this case, see Chapters 4 and 5 of the *IzoT Commissioning Tool User's Guide* for how to create application devices and functional blocks, and commission application devices.

18. Test the I/O devices on the FT 6000 EVBs using the sample network variable connections provided in the LonMaker drawing and network variable connections that you can create. See *Creating Connections in Managed Mode* later in this chapter for how to do this.

**Note:** If you loaded the *NcSimpleIsiExample* or *NcMultiSensorExample* application on your FT 6000 EVBs, you can switch to ISI mode and use the ISI protocol to connect the example applications or connect them to other applications that are compatible with their ISI assemblies. The *NcSimpleIsiExample* and *NcMultiSensorExample* applications are compatible with each other, and other compatible applications include the *MGSwitch*, *MGLight*, and *MGDemo* applications running on an FT 3150 EVB, and the *MGSwitch* or *MGLight* applications running on an FT 3120 EVB.

To switch to ISI mode, use the LonMakerBrowser or a Data Point shape to change the value of the **SCPTnwkCnfg** configuration property in the example device's **Node Object** functional block to **CFG\_LOCAL**. Alternatively, you can use the **nciNetConfig** configuration property in the example device's **Virtual Functional Block**. The IzoT Commissioning tool immediately loses communication with the example device. You can re-commission the example device to return the example application to managed mode. For more information on using the LonMaker Browser and the Data Point shape in the IzoT Commissioning tool, see Chapter 6 of the *IzoT Commissioning Tool User's Guide*.



19. Alternatively, you switch to ISI mode by holding down the service pin on the FT 6000 EVB for approximately 10 seconds. This resets the example applications to their factory default settings. See *Creating Connections in ISI Mode* later in this chapter for more information on connecting the example application running on your FT 6000 EVB via the ISI protocol. Press the Service button on the FT 6000 EVB. The Service button on the FT 6000 EVB is a black button that is located near the upper right-hand corner of the board and is labeled “Service.”
20. The **Service Pin Message** dialog opens. The Neuron ID of the FT 6000 EVB appears in the **Neuron ID** box and its program ID in the **Program ID** box.



**Note:** An FT 6000 EVB running the *NcSimpleExample* must be installed using the IzoT Commissioning tool or other network tool. If you do not have a network tool installed on your development computer, do not select the *NcSimpleExample.ndl* file. You can install an FT 6000

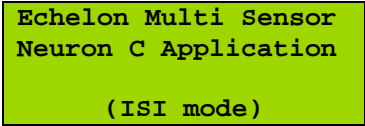


EVB running the *NcSimpleIsiExample* and *NcMultiSensorExample* with the Interoperable Self-Installation (ISI) protocol or with a network tool.

21. Click **Load** to load the selected Neuron application image into the FT 6000 EVB. The **Status** box informs you when the application image has been successfully loaded into the device, and also informs you of any load errors.

```
8/12/2009, 16:32:27]Resetting node
8/12/2009, 16:32:29]Successfully loaded C:\LONWORKS\NEURONC\EXAMPLES\FT5000 EVB\RELEASEDBINARIES\
8/12/2009, 16:32:29]NodeLoad Result: Success; NID=0700000095fe.
8/12/2009, 16:32:30]----- NodeLoad complete -----
8/12/2009, 16:32:30]Send      IsiDomainResourceUsage      DID=495349000000 DidLength=3 NID=04.D8.9E.2F
```

22. When the application image has been downloaded to the FT 6000 EVB, the LCD on the board displays the name of the example application you loaded and the installation mode (ISI Mode).



Echelon Multi Sensor  
Neuron C Application  
  
(ISI mode)

23. If you have a second FT 6000 EVB, repeat steps 6–12 to download an application image to the board.
24. If you downloaded the *NcSimpleIsiExample* or *NcMultiSensorExample* applications to your FT 6000 EVB, you can use the ISI protocol to connect the switch and LED I/O devices on your FT 6000 EVB, and you can connect the example application on your FT 6000 EVB to a compatible application running on an FT 5000, FT 3150, and FT 3120 EVB, or another application that is compatible with the ISI assembly used by these examples. See *Creating Connections in ISI Mode* later in this chapter for more information.

If you loaded any of the three example applications, you can use the IzoT Commissioning tool or other network tool to install your example device (see *Using the IzoT Commissioning Tool to Load Example Applications* earlier in this chapter for more information). After you install the example device, you can create network variable connections with other devices, including another FT 6000 EVB running an example application (see the next section, *Creating Connections in Managed Mode*, for more information).

---

## Creating Connections in Managed Mode

You can use the IzoT Commissioning tool or other network tool to bind your example device and verify its operation within a network. For example, you can bind the switch and lamp devices on the FT 6000 EVBs so that pressing the push buttons turns the LEDs on and off. If you are using the *NcMultiSensorExample* application, you can bind the temperature sensor and light-level sensor devices on one FT EVB 6000 device to the LCD display on a second FT 6000 EVB. This lets you can monitor the temperature, light level, and alarm conditions of the first 6000 FT EVB from the second one.

To bind devices, you connect the network variables within the device's functional blocks in the LonMaker drawing, and verify that the network variable values are updated appropriately when you use the I/O devices on the FT 6000 EVB.

You can connect an output network variable on one example device to compatible input network variables on the second example device. You can also create connections between compatible network variables in the same example device, and between an example device and your own device application or another LONWORKS device. Once you create a connection, the input network variables will receive all updates from the output network variables in the connection.

The LonMaker drawing includes one or more sample network variable connections for your example application. You can use these sample network variable connections and then create your own.

**Note:** Using a network tool lets you create more complex network variable connections that generally perform better compared to connections created with the ISI protocol. In addition, using a network

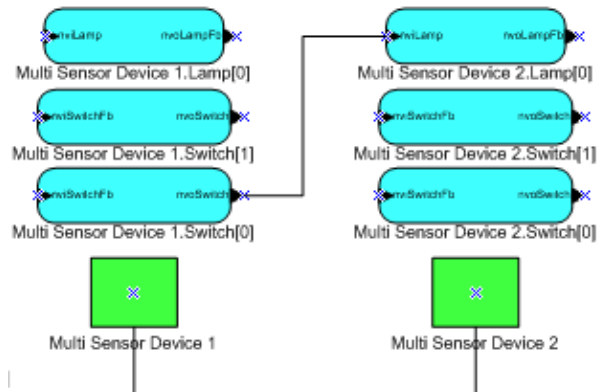
tool lets you design network variable connections offsite without having to access the physical network.

The following sections describe how to use network variable connections to test the switch and lamp devices if you are running any of the three example applications, and how to test the temperature sensor, light-level sensor, LCD, and joystick devices if you are running the *NcMultiSensorExample*.

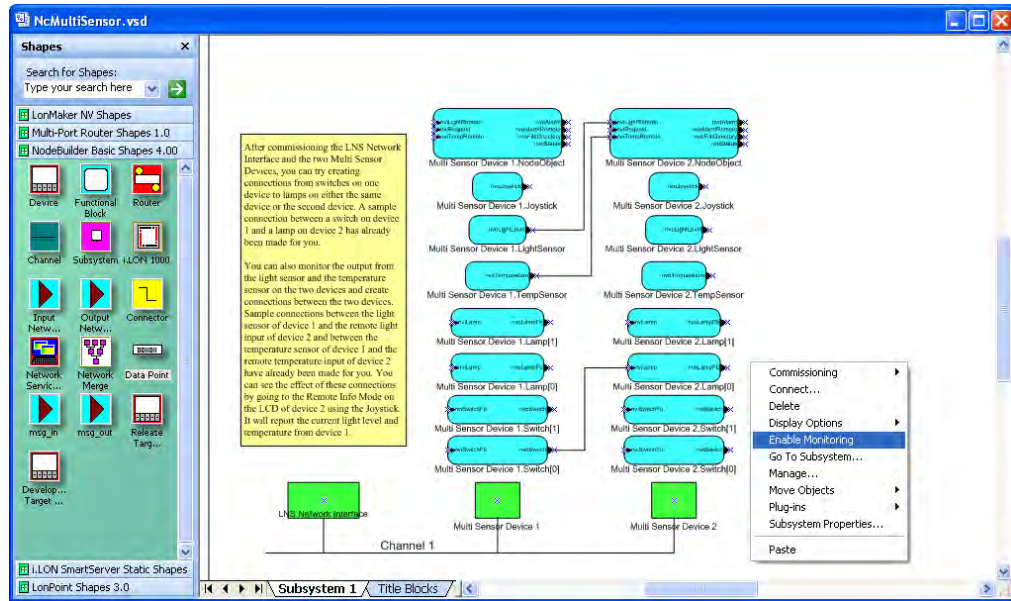
## Testing Switch and Lamp Devices

To test the network variable connections between the switch and lamp devices on the FT 6000 EVBs, follow these steps:

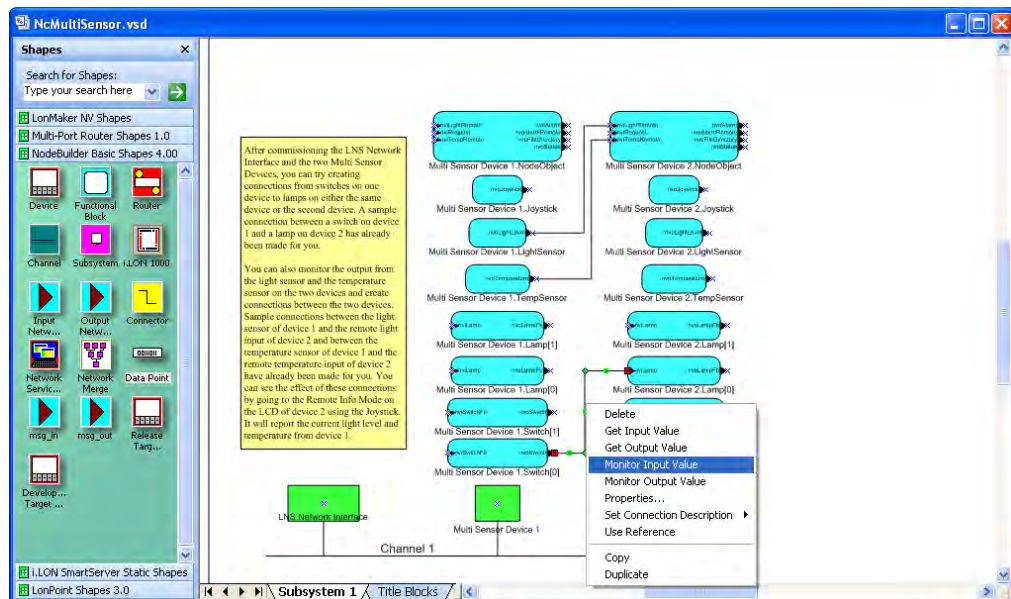
1. Verify that you have downloaded the desired example application to the FT 6000 EVBs following the steps described in *Using the IzoT Commissioning Tool to Load Example Applications* earlier in this chapter.
2. Press the **SW1** button on the FT 6000 EVB corresponding to **Multi Sensor Device 1** to turn on **LED1** on the FT 6000 EVB corresponding to **Multi Sensor Device 2**, and then press the **SW1** button again to turn off **LED1**.
3. Click the Echelon LonMaker/Visio Taskbar button in the Taskbar to switch to the IzoT Commissioning tool, if necessary. Observe that the drawing includes a sample network variable connection between the **nvoSwitch** output network variable in the **Switch[0]** functional block on the **Multi Sensor Device 1** device (**Multi Sensor Device 1.Switch[0].nvoSwitch**) and the **nviLamp** input network variable in the **Lamp[0]** functional block on the **Multi Sensor Device 2** device (**Multi Sensor Device 2.Lamp[0].nviLamp**). This connection enables **LED1** on the **Multi Sensor Device 2** device to be updated when the **SW1** button on the **Multi Sensor Device 1** device is pressed.



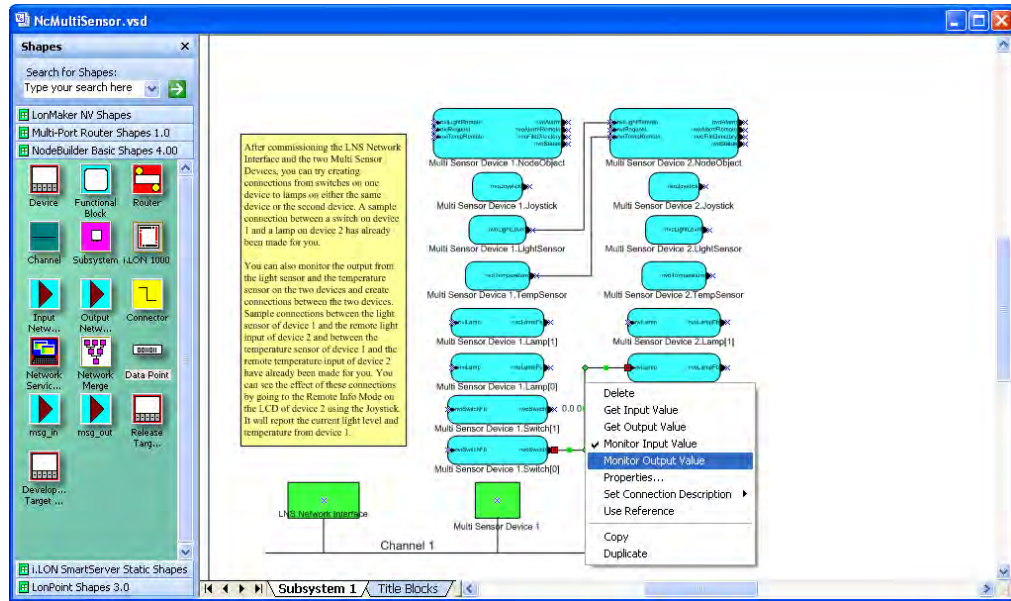
4. Monitor the values of the **Multi Sensor Device 1.Switch[0].nvoSwitch** and **Multi Sensor Device 2.Lamp[0].nviLamp** network variables. To do this, follow these steps:
  - a. Right-click an empty space in the LonMaker drawing, and then select **Enable Monitoring** on the shortcut menu.



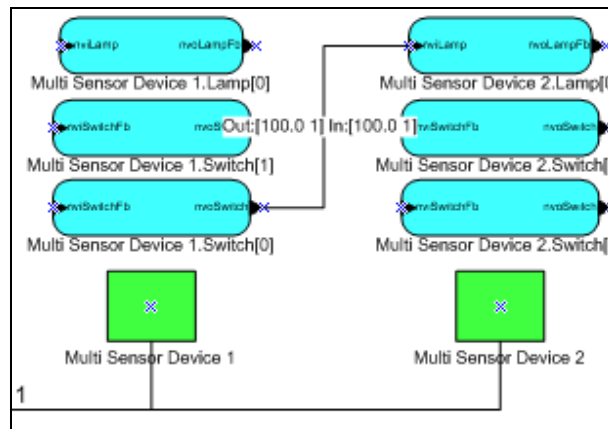
- b. Right-click the sample **Connector** shape and select **Monitor Input Value** to display the current value of the **Multi Sensor Device 1.Switch[0].nvoSwitch** output network variable.



- c. Right-click the **Connector** shape and select **Monitor Output Value** to display the current value of the **Multi Sensor Device 2.Lamp[0].nviLamp** input network variable.

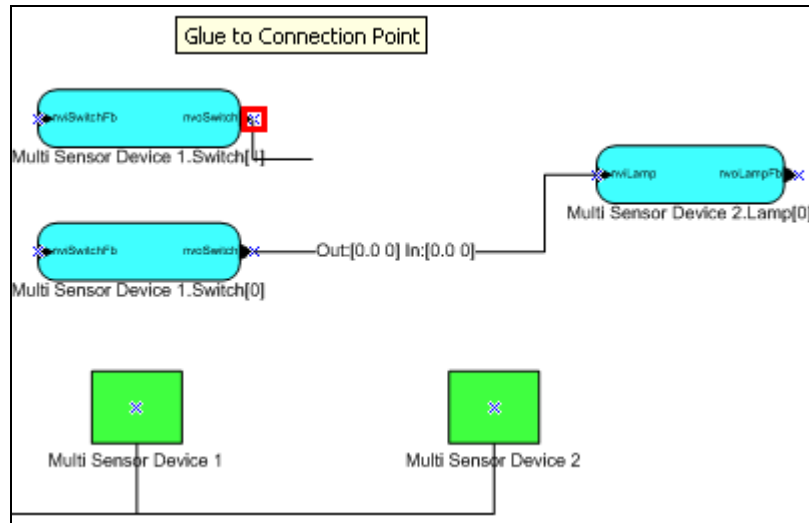


- d. Press the **SW1** button on **Multi Sensor Device 1** multiple times. Observe that **LED1** on **Multi Sensor Device 2** turns on and off each time you press the **SW1** button. In addition, the current values of the output and input network variable on the **Connector** shape toggle between 100.0 1 and 0.0 0 each time you press the button.



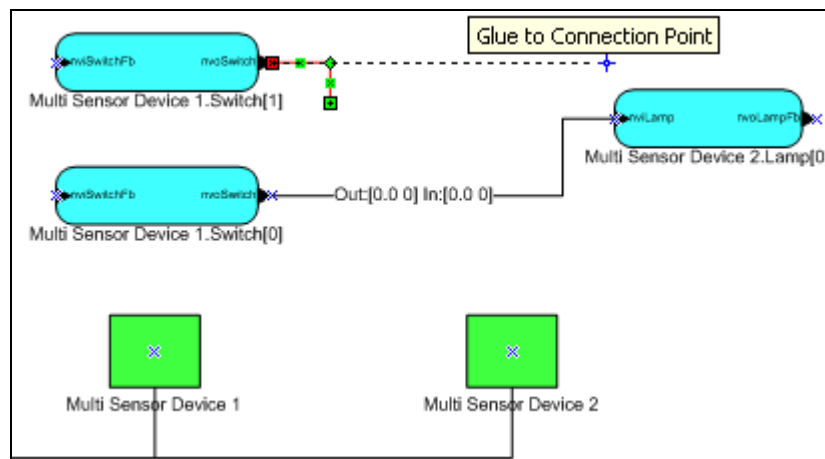
If you are running the *NcSimpleExample* application, you are done. See *Getting Started with Developing Device Applications* in this chapter for more information about the next steps to take to create your own device application, and see *NcSimpleExample Details* in Chapter 2 for more detailed information about the device interface, Neuron C code, and I/O devices used by the *NcSimpleExample* application.


5. If you are running the *NcSimpleIsiExample* or *NcMultiSensorExample*, connect the **Multi Sensor Device 1.Switch[1].nvoSwitch** network variable to the **Multi Sensor Device 2.Lamp[0].nviLamp** network variable. To do this follow these steps:
  - a. Drag the **Connector** shape from the **NodeBuilder Basic Shapes 4.00** stencil to the drawing. Position the left end of the shape over the tip of the **Multi Sensor Device 1.Switch[1].nvoSwitch** output network variable before releasing the mouse button. A red box appears around the end of the **Connector** shape when you have positioned it correctly over the **Network Variable** shape.



**Note:** To simplify the LonMaker drawing in this example, only the subject functional blocks are further displayed.

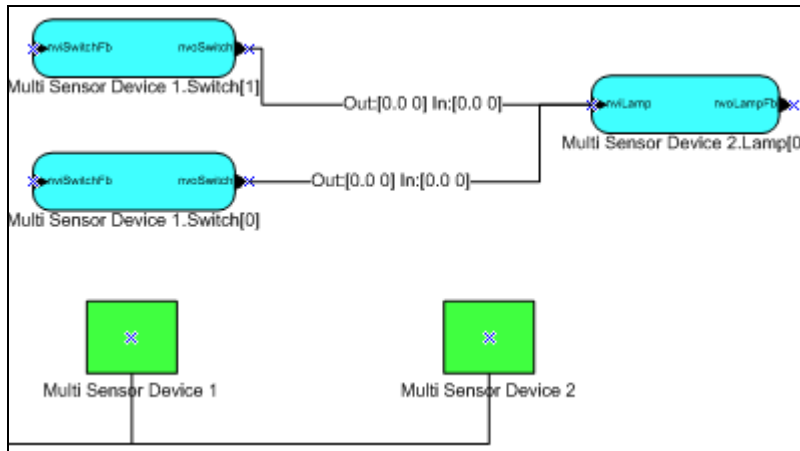
- b. Drag the other end of the **Connector** shape to the **Multi Sensor Device 2.Lamp[0].nviLamp** input network variable until it snaps into place and a square box appears around the end of the **Connector** shape. There is a brief pause as the IzoT Commissioning tool updates the example devices over the network.



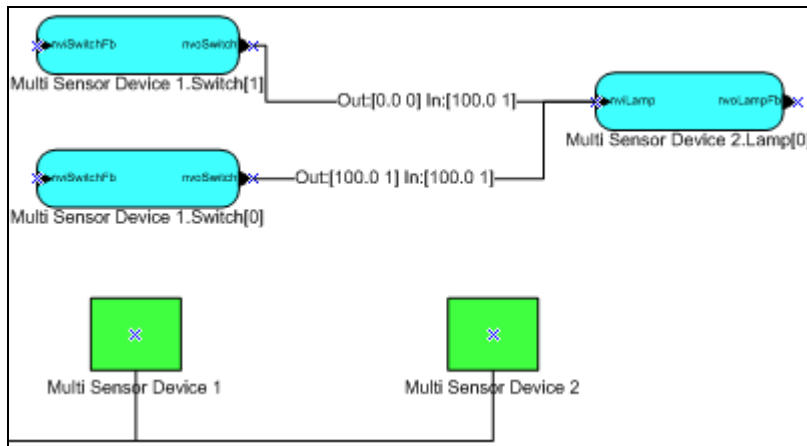
**Note:** You can also create connections using the **Connector** tool () on the Visio **Standard** toolbar or the **Network Variable Connection** dialog box. See Chapter 4 of the *IzoT Commissioning Tool User's Guide* for more information on creating connection using these methods.

6. Follow step 4 to monitor the values of the **Multi Sensor Device 1.Switch[1].nviSwitch** and **Multi Sensor Device 2.Lamp[0].nviLamp** network variables on the connection you just created.

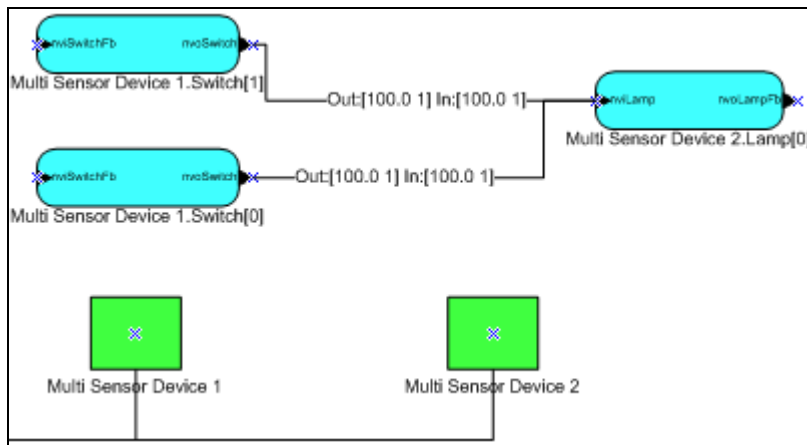




7. If **LED1** on **Multi Sensor Device 2** is off, press the **SW1** button on **Multi Sensor Device 1** to turn it on.



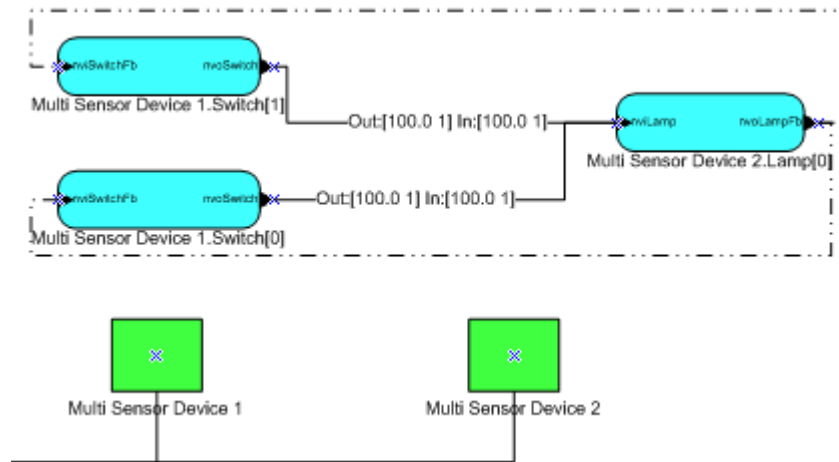
8. Press the **SW2** button on **Multi Sensor Device 1**. Observe that **LED1** on **Multi Sensor Device 2** does not turn off, and observe that the **Multi Sensor Device 2.Lamp[0].nviLamp** network variable is not updated in the LonMaker drawing. This is because the **Multi Sensor Device 1.Switch[1].nvoSwitch** network variable does not know the current state of **LED1**—it is sending an ON value while **LED1** is already on.



Proceed to step 9 to create feedback connections that enable the **nvoSwitch** network variables to get the current state of **LED1** so that pressing either the **SW1** or **SW2** buttons places **LED1** in the

expected condition. Feedback connections are useful for synchronizing devices such as in this example.

9. Create the following two feedback connections. These feedback connections lets the switches know whether the **LED1** is actually turned on or off so that pressing the **SW1** or **SW2** buttons turns **LED1** on and off.
  - Create a connection between the **Multi Sensor Device 2.Lamp[0].nvoLampFB** output network variable and the **Multi Sensor Device 1.Switch[0].nviSwitchFB** input network variable.
  - Create a connection between the **Multi Sensor Device 2.Lamp[0].nvoLampFB** output network variable and the **Multi Sensor Device 1.Switch[1].nviSwitchFB** input network variable.



10. Press the **SW1** and **SW2** buttons on **Multi Sensor Device 1**. Observe that **LED1** on **Multi Sensor Device 2** turns on and off each time you press either button. Also, observe that the **nvoSwitch** network variables are updated simultaneously.

If you are running the *NcSimpleIsiExample* application, you are done. See *Getting Started with Developing Device Applications* in this chapter for more information about the next steps to take to create your own device application, and see *NcSimpleIsiExample Details* in Chapter 2 for more detailed information about the device interface, Neuron C code, and I/O devices used by the *NcSimpleIsiExample* application.

If you are running the *NcMultiSensorExample* application, proceed to the next section, to monitor the temperature, light level (lux), and alarm conditions, and set alarm limits for the temperature and light levels.

## Testing Light, Temperature, LCD, and Joystick Devices

You can use the light-level sensor, temperature sensor, LCD, and joystick devices on the FT 6000 EVBs to monitor the light level (lux), temperature, and alarm conditions and to set alarm limits for the light level and temperature of an FT 6000 EVB running the *NcMultiSensorExample* application.

You can also create connections between the network variables in the **LightSensor** and **TempSensor** functional blocks of **Multi Sensor Device 1** to the network variables in the **Node Object** functional block of **Multi Sensor Device 2**, and vice versa. This enables you to view the light, temperature, and alarm conditions of one FT EVB 6000 from the LCD of the other FT EVB 6000.

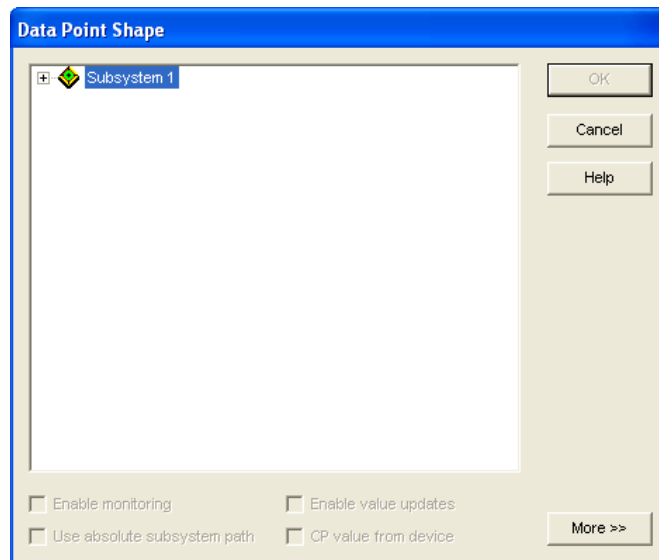
The LonMaker drawing includes one set of sample network variable connections between the network variables in the **LightSensor** and **TempSensor** functional blocks of **Multi Sensor Device 1** to the network variables in the **Node Object** functional block of **Multi Sensor Device 2**. You can create your own network variable connections between the **LightSensor** and **TempSensor** functional blocks

of **Multi Sensor Device 2** to the network variables in the **Node Object** functional block of **Multi Sensor Device 1**.

To test the light-level sensor, temperature sensor, and LCD devices on the FT 6000 EVBs, follow these steps:

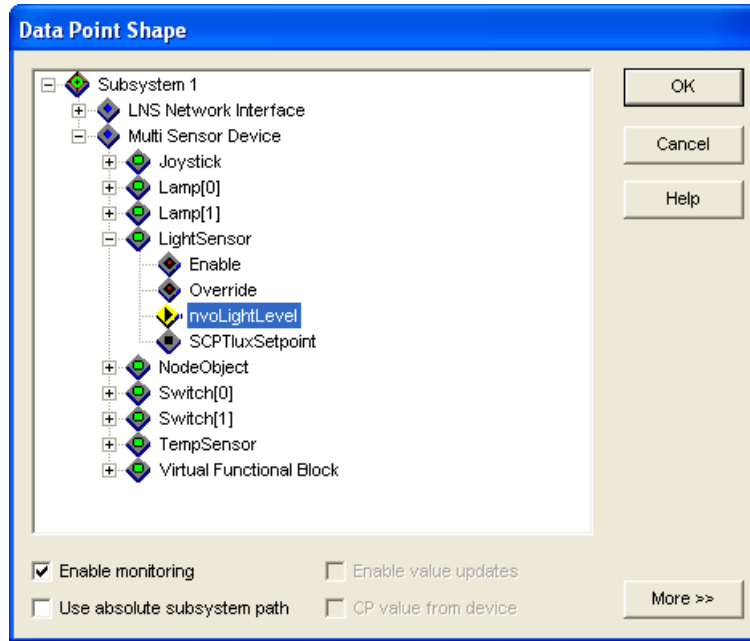
1. Use the LCD on the FT 6000 EVB corresponding to **Multi Sensor Device 1** and the LonMaker drawing to monitor the light level and temperature of the board, set alarm limits for the light level and temperature, and observe the affects of changing the light level and light alarm limits. To do this, follow these steps:
  - a. Drag a **Data Point** shape from the **NodeBuilder Basic Shapes 4.00** stencil to the drawing. The Data Point shape lets you monitor and control a single network variable or configuration property value from the current drawing page. It is ideal for testing smaller device interfaces with few network variables and configuration properties. For more information on using the Data Point shape in the IzoT Commissioning tool, see Chapter 6 of the *IzoT Commissioning Tool User's Guide*.

You can place the Data Point shape anywhere, but a good place is directly above or below the functional block containing the data point to be monitored and controlled. The **Data Point Shape** dialog opens.



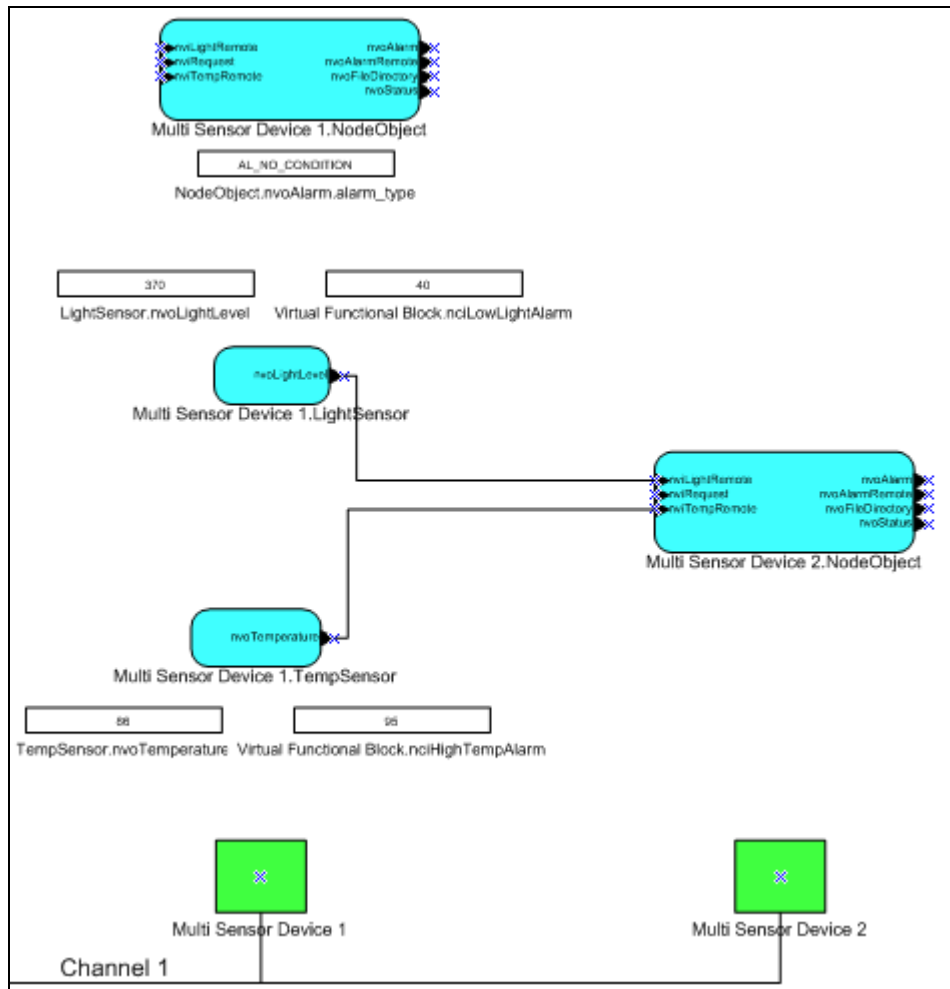
Expand **Subsystem 1**, expand **Multi Sensor Device 1**, expand the **LightSensor** functional block, and then select the **nvoLightLevel** output network variable; select the **Enable Monitoring** check box; and then click **OK**.





The Data Point shape is added to your LonMaker drawing.

- b. Repeat step a for the following data points:
- the **Multi Sensor Device 1.TempSensor.nvoTemperature** output network variable.
  - the **Multi Sensor Device 1.Virtual Functional Block.nciHighTempAlarm** and **Multi Sensor Device 1.Virtual Functional Block.nciLowLightAlarm** configuration properties. When you add these configuration properties, select the **Enable Value Updates** check box in the **Data Point Shape** dialog. This enables you to write values to these configuration properties.
  - the **Multi Sensor Device 1.NodeObject.nvoAlarm** output network variable. When you add this network variable, click **More**, and then select **alarm\_type** in the **Field Name** property in the **Data Point Shape** dialog. This enables you to view whether the temperature or light level has exceeded their respective alarm settings.



**Note:** To simplify the LonMaker drawing in this example, only the subject functional blocks are further displayed.

- c. Toggle the joystick on the FT 6000 EVB corresponding to **Multi Sensor Device 1** down once to display the **Local Info Mode** on the board's LCD (the joystick is located in the bottom center of the board between the **SW1** and **SW2** buttons). Observe that the **Light:** and **Temp:** properties in the LCD reflect the current lux and temperature, and the **Alarms:** property in the LCD displays "None".

```

Local Info Mode
Light : 370 Lux
Temp : 30.0 C
Alarms: Light

```

This means the following:

- The current lux is more than the lower limit defined in the **Multi Sensor Device 1.Virtual Functional Block.nciLowLightAlarm** configuration property, which is **40** lux by default.
- The current temperature is less than the upper limit defined in the **Multi Sensor Device 1.Virtual Functional Block.nciHighTempAlarm** configuration property, which is **35.0°C** by default.

**Note:** You can display the current temperature in Fahrenheit by toggling the joystick sideways. You can change it back to Celsius by toggling the joystick sideways again. Changing the temperature format in the **Local Info Mode** panel also changes the format used for the high temperature alarm setpoint in the **Alarm Config Mode** panel.

- d. On the FT 6000 EVB corresponding to **Multi Sensor Device 1**, cover the light sensor I/O device on the bottom right side of the board. Observe that the **Light:** property in the LCD reflects the decreased lux, and the **Alarms:** property in the LCD displays “Light”.

Local Info Mode		
Light :	12	Lux
Temp :	30.0	C
Alarms:	Light	

This means that the current lux (12 in this example) is less than the lower limit defined in the **Multi Sensor Device 1.Virtual Functional Block.nciLowLightAlarm** configuration property (40 lux by default).

- e. In the LonMaker drawing, double-click the Data Point shape for the **Multi Sensor Device 1.Virtual Functional Block.nciLowLightAlarm** configuration property, enter 0, and then click anywhere outside the Data Point shape. Cover the light sensor I/O device again on the FT 5000 EVB corresponding to **Multi Sensor Device 1**.

Observe that the **Alarms:** property in the LCD now displays “None”. This is because the current lux is now higher than the lower limit defined in the **Multi Sensor Device 1.Virtual Functional Block.nciLowLightAlarm** configuration property.

Local Info Mode		
Light :	12	Lux
Temp :	30.0	C
Alarms:	None	

- f. Toggle the joystick down two more times on the FT 6000 EVB corresponding to **Multi Sensor Device 1** to display the **Alarm Config Mode** on the board’s LCD. Toggle the joystick up two times to move the pointer to the **Light:** alarm property. Toggle the joystick right eight times to increase the light alarm level from 0 to 40 (returning to its default level), toggle the joystick down two times to point to the **Ok** option, and then press the joystick center button to save the new light alarm level.

Alarm Config Mode		
Light :	40	Lux
Temp :	30.0	C
Cancel Ok<-		

**Note:** When you set the light alarm level, toggling the joystick left decreases the lux by 5, and toggling the joystick right increases the lux by 5. When you set the temperature alarm level, toggling the joystick left decreases the temperature by 0.5°C, and toggling the joystick right increases the temperature by 0.5°C.

- g. In the LonMaker drawing, observe that the value in the data point shape for the **Multi Sensor Device 1.Virtual Functional Block.nciLowLightAlarm** configuration property has been updated to reflect the current light alarm level.
2. Use the sample network variable connections in the LonMaker drawing to monitor the light level, temperature, and alarm conditions of the FT 6000 EVB corresponding to **Multi Sensor Device 1** from the Remote Info Mode panel on the LCD of the FT 6000 EVB corresponding to **Multi Sensor Device 2**.

**Note:** If you only have one FT 6000 EVB, you can still observe the application behavior described in this step by connecting the light level and temperature output network variables in the

**LightSensor** and **TempSensor** functional blocks to the remote light level and temperature input network variables in the **NodeObject** functional block on the same device.

To use the sample network variable connections in the LonMaker drawing to monitor the light level, temperature, and alarm conditions, follow these steps:

- a. Toggle the joystick down on the FT 6000 EVB corresponding to **Multi Sensor Device 2** (the local device) two times to display the **Remote Info Mode** on the board's LCD. This mode displays the current temperature, lux, and alarm conditions of the FT 6000 EVB corresponding to **Multi Sensor Device 1** (the remote device).

Remote Info Mode		
Light :	501	Lux
Temp :	85.1	F
Alarms:	None	

**Note:** You can display the temperature measured by the remote device in Fahrenheit by toggling the joystick sideways. You can change it back to Celsius by toggling the joystick sideways again.

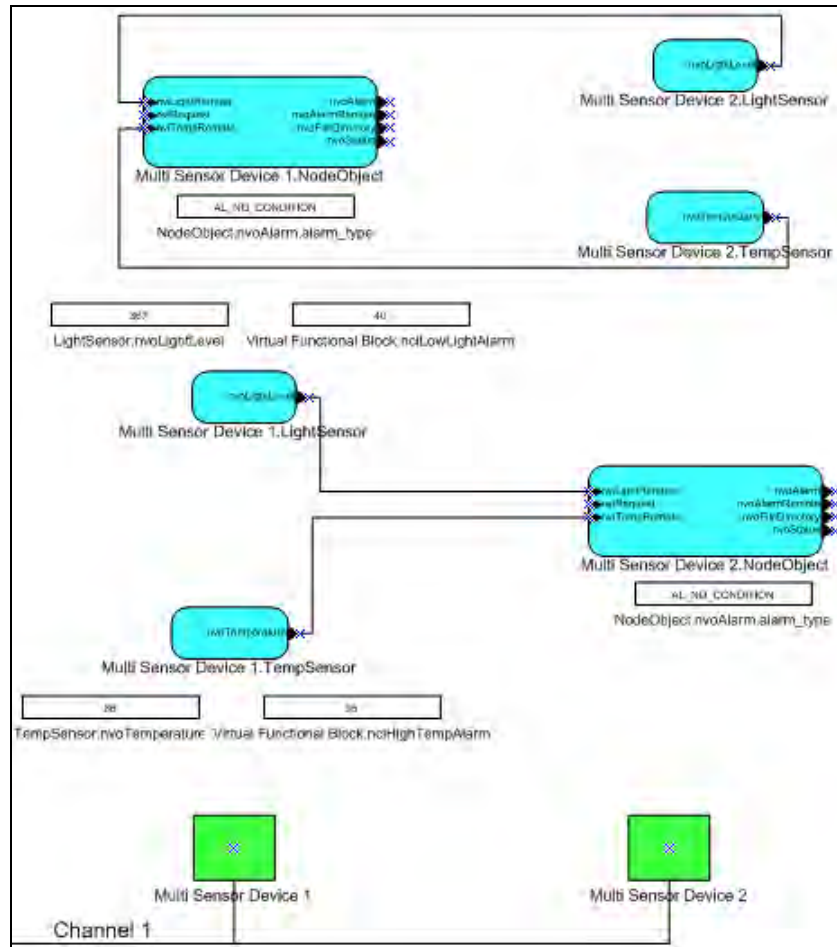
- b. On the FT 6000 EVB corresponding to **Multi Sensor Device 1**, cover the light sensor I/O device on the bottom right side of the board.
- c. On the LCD of the FT 6000 EVB corresponding to **Multi Sensor Device 2**, observe that the **Light:** property in the **Remote Info Mode** panel reflects the decreased lux of the remote device and the **Alarms:** property displays "Light". This means that that current lux is less than the lower limit defined in the Light property in the **Alarm Config Mode** panel on the remote device.

Remote Info Mode		
Light :	18	Lux
Temp :	85.1	F
Alarms:	Light	

- d. In the LonMaker drawing, observe that the value in the data point shape for the **Multi Sensor Device 2. Node Object. nvoAlarmRemote** output network variable is now **AL\_ALM\_CONDITION**.
3. Optionally, you can use the IzoT Commissioning tool to create network variable connections between the **LightSensor** and **TempSensor** functional blocks of **Multi Sensor Device 2** to the network variables in the **Node Object** functional block of **Multi Sensor Device 1**. This is simply the opposite of the sample network variable connections used in step 2.

You can then monitor the light level, temperature, and alarm conditions of the FT 6000 EVB corresponding to **Multi Sensor Device 2** from the Remote Info Mode panel on the LCD of the FT 6000 EVB corresponding to **Multi Sensor Device 1**.

- a. Connect the **Multi Sensor Device 2.LightSensor. nvoLightLevel** output network variable to the **Multi Sensor Device 1.NodeObject.nviLightRemote** input network variable.
- b. Connect the **Multi Sensor Device 2.LightSensor.nvoTemperature** output network variable to the **Multi Sensor Device 1. NodeObject.nviTempRemote** input network variable.
- c. Create a Data Point shape for the **alarm\_type** field in the **Multi Sensor Device 1.NodeObject.nvoAlarmRemote** output network variable.



4. See *Getting Started with Developing Device Applications* in this chapter for more information about the next steps to take to create your own device application, and see *NcMultiSensorExample Details* in Chapter 2 for more detailed information about the device interface, Neuron C code, and I/O devices used by the *NcMultiSensorExample* application.

## Creating Connections in ISI Mode

You can use the ISI protocol to create and maintain network variable connections without using a network management tool. To create network variable connections with the ISI protocol, the devices require some input that indicates to which devices to connect. One method is for the user to press Connect buttons on the devices to initiate, accept, and confirm the connection, and use Connect lights to indicate the connection status. For the FT 6000 EVB, the Connect button is the **SW2** button on the right side of the board, and the Connect light is **LED2**, which is located directly above the **SW2** button.

If you downloaded the *NcSimpleIsiExample* or *NcMultiSensorExample* application to your FT 6000 EVB, you can use the ISI protocol to connect the switch and LED I/O devices on your FT 6000 EVB to another device running an application that is compatible with the ISI assembly used by these examples. Applications that have compatible Light and Switch assemblies are the *NcSimpleIsiExample* and *NcMultiSensorExample* applications running on another FT 6000 EVB, the *MGSwitch*, *MGLight*, and *MGDemo* applications running on an FT 3150 EVB, and the *MGSwitch* or *MGLight* applications running on an FT 3120 EVB.

To create connections between the *NcSimpleIsiExample* or *NcMultiSensorExample* application running on your FT 6000 EVB to a compatible application running on another FT 6000, FT 3150, or FT 3120 EVB, follow these steps:

1. Verify that one or both of the FT 6000 EVBs are running the *NcSimpleIsiExample* or *NcMultiSensorExample* application, and verify that the applications are in ISI mode. The name of the application is displayed at the top of the LCD on the board.

**Note:** The *NcSimpleExample* does not support ISI connections. Download the *NcSimpleIsiExample* or *NcMultiSensorExample* application to the board with the IzoT NodeBuilder tool. See *Using the IzoT Commissioning Tool to Load Example Applications* for more information on how to do this.

2. If you are connecting an FT 6000 EVB to an FT 3150 EVB or FT 3120 EVB, verify that the FT 3150 EVB or FT 3120 EVB is running the appropriate example application.
  - An FT 3150 EVB must be running the *MGSwitch*, *MGLight*, or *MGDemo* application, or another application that is compatible with the ISI assembly used by the *NcSimpleIsiExample* and *NcMultiSensorExample* applications.
  - An FT 3120 EVB must be running the *MGSwitch* or *MGLight* application, or another application that is compatible with the ISI assembly used by the *NcSimpleIsiExample* and *NcMultiSensorExample* applications.
3. Press the Connect button on one evaluation board (this is the connection host). The Connect button on an FT 6000 EVB running the *NcSimpleIsiExample* or *NcMultiSensorExample* application is the **SW2** button on the right side of the board. The Connect buttons on the MiniGizmo board, which is attached to an FT 3150 or FT 3120 EVB board, are **SW5**, **SW6**, **SW7**, or **SW8** if running the *MGDemo* application, or **SW8** if running the *MGSwitch* or *MGLight* application. This step is called opening enrollment or initiating the connection.
4. The Connect lights on the other evaluation boards that can join the connection and the Connect light on the connection host start blinking. The Connect light on an FT 6000 EVB running the *NcSimpleIsiExample* or *NcMultiSensorExample* application is **LED2**, which is located directly above the **SW2** button. The Connect lights on the MiniGizmo board are **LED5**, **LED6**, **LED7**, or **LED8** if running the *MGDemo* application, or **LED8** if running the *MGSwitch* or *MGLight* application. The LEDs are located directly above their respectively numbered buttons.
5. Press the Connect button on another evaluation board (the connection member) to add that device to the connection. The Connect lights on both the device and the connection host will illuminate without flashing, indicating they are ready to join the connection. This step is called accepting the connection invitation.
6. Press the Connect button used in step 4 on the connection host to complete the connection. The Connect lights on both the connection host and the device will extinguish, indicating that the devices are connected. This step is called confirming the connection.
7. Press the I/O button on either evaluation board to illuminate the I/O LEDs in the connection. Press the I/O button on either evaluation board to extinguish the I/O LEDs in the connection.
  - For an FT 6000 EVB, the I/O button and LED in an ISI connection are **SW1** and **LED1**, respectively.
  - For the *MGSwitch*, *MGLight*, or *MGDemo* applications running on a FT 3150 or 3120 EVB, the I/O button and LED in an ISI connection depends on the application and the connect button used

**Note:** You can remove the device from the ISI connection by pressing and holding the Connect button on the connection member (not the connection host) for approximately 8 seconds. This step is called cancelling the connection. You can confirm that the device has left the connection by pressing its I/O button and observing that the I/O LED of the connection host does not illuminate. You can delete an

ISI connection to all connection members by pressing and holding the Connect button on the Connection host for approximately 8 seconds.

See Chapter 2 for more information on how the FT 6000 EVBs exchange data in ISI mode.

---

## Getting Started with Developing Device Applications

The FT 6000 EVB example applications were developed using Neuron C (Version 2.2), which is a programming language based on ANSI C that you can use to develop applications for Neuron Chips and Smart Transceivers. It includes network communication, I/O, and event-handling extensions to ANSI C, which make it a powerful tool for the development of LONWORKS device applications. For more information on the Neuron C programming language, see the *Neuron C Programmer's Guide* and the *Neuron C Reference Guide*.

You can view the Neuron C code used by the FT 6000 EVB example applications to learn how to develop your own device applications (see the *Device Application Summary* sections in Chapter 2 for descriptions of the Neuron C code used by the example applications). IzoT NodeBuilder tool users can view the Neuron C source files (.nc extension) of an example application by opening the example's NodeBuilder project, as described in the next section, *Using the IzoT NodeBuilder Tool to Develop Device Applications*. IzoT NodeBuilder tool users can view the Neuron C source files by browsing to the **C:\LonWorks\NeuronC\Examples\FT6000 EVB\<example>\Source** folder, and then opening the file with a text editor such as Notepad. Alternatively IzoT NodeBuilder tool users can access an example application's source file by clicking **Start**, pointing to **Programs**, pointing to **Echelon NodeBuilder FX** or **Echelon Mini**, pointing to **Examples**, pointing to **FT 6000 EVB**, clicking the desired **Example Source Code** folder, and then clicking the **Source** folder.

After you view the Neuron C code in the example applications, you can create a new device application by modifying the existing example applications or by developing the device application from scratch. You can then use the IzoT NodeBuilder tool to build the device applications and download them to an FT 6000 EVB or other LONWORKS device based on a Neuron 6000 Processor or FT 6000 Smart Transceiver. You can create a simple device application from scratch by following the quick-start exercise in Chapter 3 of your development tool's user's guide (*IzoT NodeBuilder FX Tool User's Guide*).

The following sections describe how to view the Neuron C source files in the FT 6000 EVB example applications based on the IzoT NodeBuilder tool, and they describe the next steps for developing a device with your development tool.

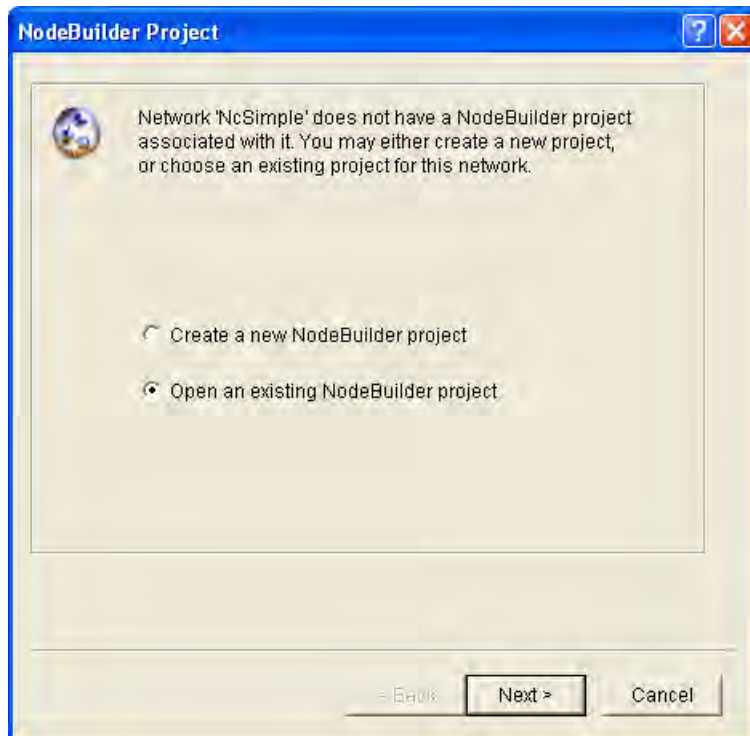
---

### Using the IzoT NodeBuilder Tool to Develop Device Applications

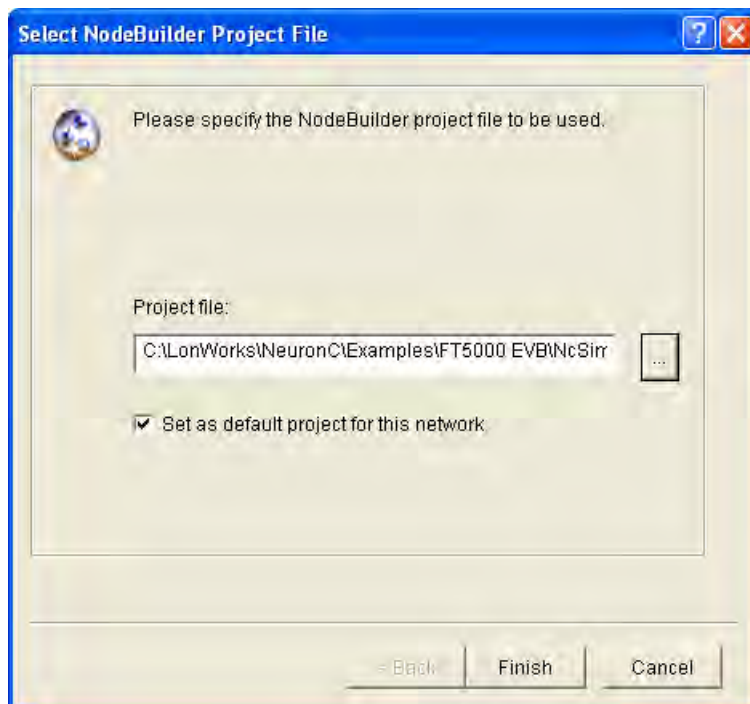
If you are using the IzoT NodeBuilder tool, you can open the pre-built NodeBuilder projects created for each example application and then view the Neuron C source (.nc) files and header (.h) files that comprise the device application. See the *Device Application Summary* sections in Chapter 2 for descriptions of the functionality provided by these files. After you view the Neuron C code in the example applications, you can create a new device application, or modify the existing example applications.

To open the NodeBuilder project for an example application and then view the Neuron C source files with IzoT NodeBuilder tool, follow these steps:

1. Restore the LonMaker backup for the example application as described by steps 1–15 in *Using the IzoT Commissioning Tool to Load Example Applications* earlier in this chapter.
2. In the LonMaker drawing, click **LonMaker** and then click **NodeBuilder**. The NodeBuilder Project Manager starts. If you have not previously created a NodeBuilder project for this network, the New Project wizard automatically starts with the **NodeBuilder Project** dialog displayed.
3. In the **NodeBuilder Project** dialog, select the **Open an Existing NodeBuilder Project** option and then click **Next**.



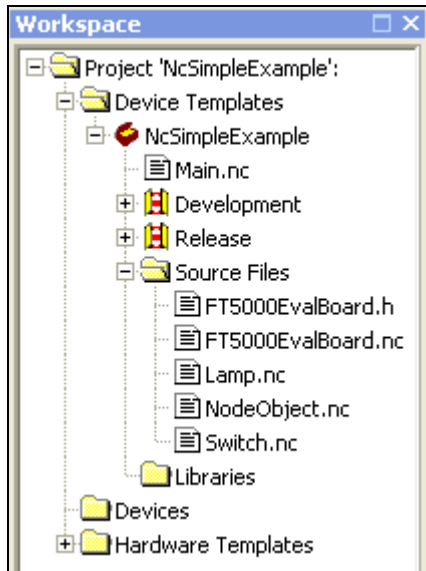
4. The **Select Project File** dialog opens. Click the button to the right of the **Project File** property, browse to the LonWorks\NeuronC\Examples\FT6000 EVB\<Example> folder, and then select the project file (.NbPrj extension) in the project folder.



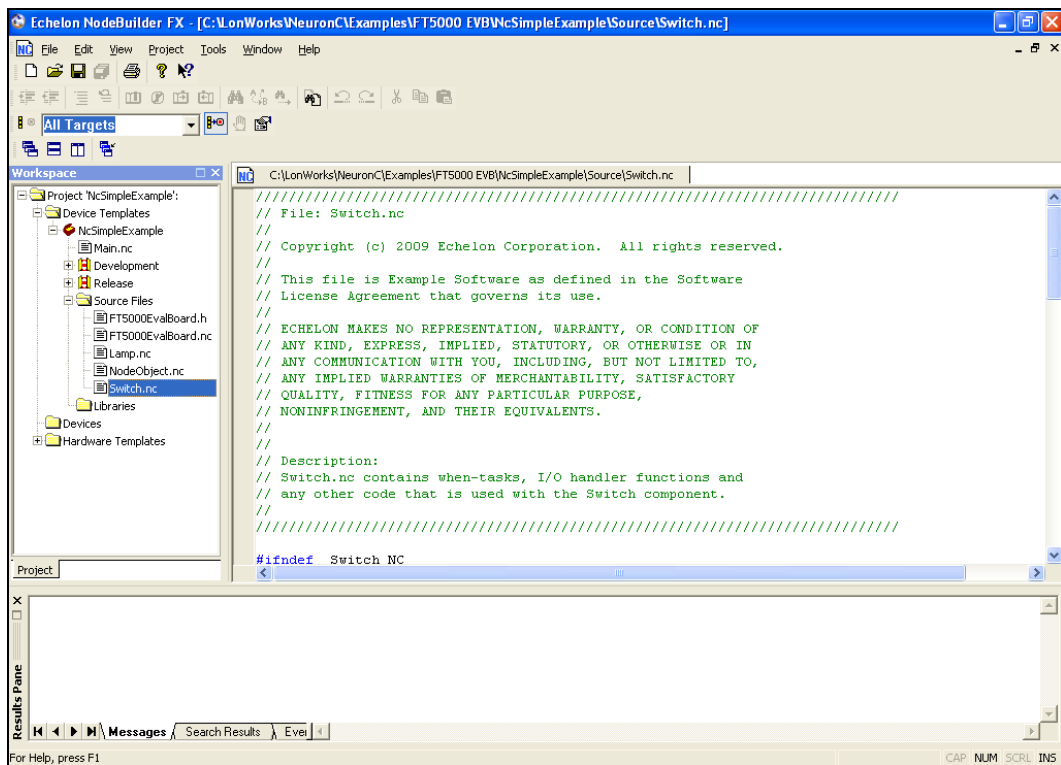
5. Click **Finish**. The NodeBuilder Project Manager opens.
6. In the Project pane on the left side of the NodeBuilder Project Manager, expand the **Device Templates** folder under the Project folder, expand the Device Template (🔌), and then expand the



Source Files folder (📁). The Neuron C source files (.nc) and header files (.h) in the device application are displayed.



7. Double-click the Neuron C source files (.nc) and header files (.h) to view them in the Edit pane on the right side of the NodeBuilder Project Manager.



8. Read the example application's *Device Application Summary* section in Chapter 2 for a summary of the functionality provided by each Neuron C source (.nc) file and header (.h) file in the device application.
9. After you view the Neuron C code in the example applications, you can create a new device application by modifying the existing example applications or by creating the device from scratch. After you create the device, you can use the IzoT NodeBuilder tool to build the device application

and download it to an FT 6000 EVB or other LONWORKS device based on a Neuron 6000 Processor and FT 6000 Smart Transceiver. For more information on creating a device with the IzoT NodeBuilder tool, see the *IzoT NodeBuilder FX User's Guide*.

You can develop a simple device application by following the quick-start exercise in Chapter 3 of the *IzoT NodeBuilder FX User's Guide*. In the quick-start exercise, you will develop a device with one sensor and one actuator. The sensor is a simple sensor that monitors a push button on the FT 6000 EVB and toggles a network variable output each time the button is pressed. The actuator drives the state of an LED on the FT 6000 EVB based on the state of a network variable input. The quick-start guides you through all the steps of creating a device with the IzoT NodeBuilder tool, including creating the NodeBuilder project, the device template, the device interface, and the Neuron C code that implements your device interface; implementing device functionality in the Neuron C code; building and downloading the device application; testing the device in a LONWORKS network; and debugging the device application.

**Note:** You can also open a NodeBuilder project directly from the NodeBuilder Project Manager. To do this, see *Opening a NodeBuilder Project from the NodeBuilder Project Manager* in Chapter 4 of the *IzoT NodeBuilder FX User's Guide*.

## Details of the FT 6000 EVB Examples

This chapter illustrates and details the device interfaces, device applications, and I/O devices used by the FT 6000 EVB examples.

---

## Introduction to FT 6000 EVB Example Details

The FT 6000 EVB examples demonstrate three major components of a LONWORKS device: the device interface, the device application, and the I/O devices on the device hardware.

The device interface includes all the functional blocks, network variables, and configuration properties defined for the example application. This chapter summarizes the device interfaces used by the FT 6000 EVB example applications, and it details the functional blocks, network variables, and configuration properties in the device interfaces.

The device application consists of Neuron C code that implements the external interface of the example device and handles interaction between the device interface and the I/O devices on the FT 6000 EVB. For example, the Neuron C source file for the **Switch** functional block used by all the three examples (**Switch.nc**) includes code that toggles the value of the **nvoSwitch** network variable when you press a push button on the on the FT 6000 EVB. In addition, the Neuron C source file for the **Lamp** functional block used by all the three examples (**Lamp.nc**) includes code that sets the state of an LED on the FT 6000 EVB when the **nviLamp** network variable is updated. This chapter summarizes the functionality of the Neuron C source (**.nc**) files and header (**.h**) files that comprise the example device applications. See *Getting Started with Developing Device Applications* in Chapter 1 for more information about using the IzoT NodeBuilder tool to view the Neuron C source and header files.

The FT 6000 EVB includes the following I/O devices: two push buttons, two LEDs, a temperature sensor, a light-level sensor, an LCD display, and a 5-way joystick. The example applications use some to all of the I/O devices. This chapter lists the I/O pins used by each example application for the I/O devices on the FT 6000 EVB and illustrates the interaction between the I/O devices and the device interface used for each example. An I/O pin is used to connect a Neuron Chip or Smart Transceiver to one or more physical I/O devices. The Neuron firmware implements numerous *I/O objects* that manage the interface to these devices for a Neuron C application. In your Neuron C code, you will declare the I/O objects that monitor and control the Neuron 6000 Processor or FT 6000 Smart Transceiver I/O pins. For more information on using and declaring I/O objects, see Chapter 2 of the *Neuron C Programmer's Guide* and Chapter 8 of the *Neuron C Reference Guide*.

---

## NcSimpleExample Details

The *NcSimpleExample* application demonstrates how you can use switch devices to activate lamp devices in a managed network. It uses one push button (**SW1**) that represents a switch device, and one LED (**LED1**) that represents a lamp device.

---

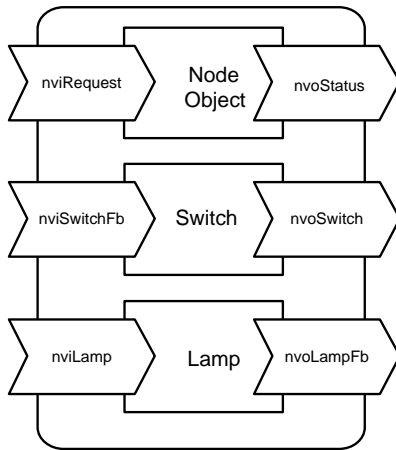
### Device Interface

The following section summarizes and then details the *NcSimpleExample* application.

#### Summary

The *NcSimpleExample* application includes a **Node Object** functional block, and Switch and Lamp functional blocks representing the push button and LED I/O objects on the FT 6000 EVB. Both of the **Switch** and **Lamp** functional blocks contain **SNVT\_switch** input and output network variables.

The following diagram displays the functional blocks and network variables in the device interface used by the *NcSimpleExample* application.

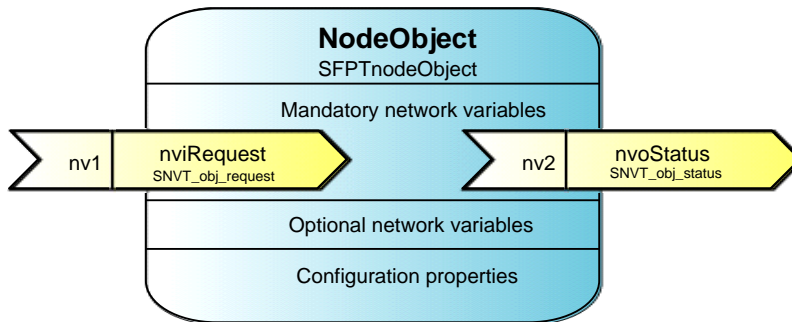


## Details

The following sections detail the **Node Object**, **Switch**, and **Lamp** functional blocks in the *NcSimpleExample* device interface and their network variables.

### Node Object Functional Block

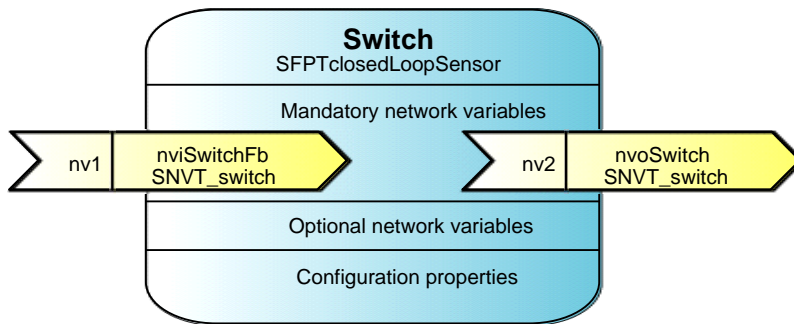
The **NodeObject** functional block returns invalid request for all requests that are not implemented. It returns an invalid ID for all IDs that are not implemented.



FB/NV Name	Type	Direction	Description
NodeObject	SFPTnodeObject		Lets you monitor the functional blocks within a device.
nviRequest	SNVT_obj_request	Input	Lets you place a functional block in the device interface in a specific state. Handles RQ_NORMAL, RQ_UPDATE_STATUS, RQ_REPORT_MASK and RQ_ENABLE requests.
nvoStatus	SNVT_obj_status	Output	Reports the status of a functional block in the device interface.

### Switch Functional Block

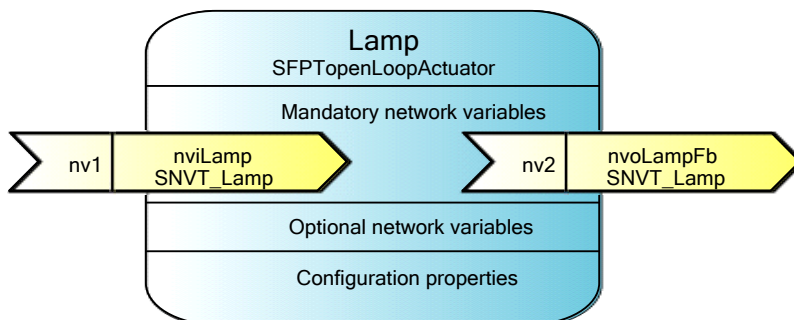
The **Switch** functional block represents the **SW1** push button on the FT 6000 EVB. The **Switch** functional block contains **SNVT\_switch** input and output network variables.



FB/NV Name	Type	Direction	Description
Switch	SFPTclosedLoopSensor		Represents the <b>SW1</b> push button on the FT 6000 EVB.
nvoSwitch	SNVT_switch	Output	Transmits the value and state of the <b>SW1</b> button on the FT 6000 EVB.
nviSwitchFb	SNVT_switch	Input	Lets you create a feedback connection between the <b>SW1</b> button and <b>LED1</b> .

### Lamp Functional Block

The **Lamp** functional block represents **LED1** on the FT 6000 EVB. The **Lamp** functional block contains **SNVT\_switch** input and output network variables.



FB/NV Name	Type	Direction	Description
Lamp	SFPTopenLoopActuator		Represents <b>LED1</b> on the FT 6000 EVB.
nviLamp	SNVT_switch	Output	Receives the value and state of a <b>SNVT_switch</b> network variable and updates <b>LED1</b> on the FT 6000 EVB accordingly.
nviSwitchFb	SNVT_switch	Input	Lets you send the current value and state of <b>LED1</b> to the <b>Switch</b> functional block.

---

## Device Application Summary

The following table summarizes the functionality provided by the Neuron C source (.nc) files and header (.h) files that comprise the *NcSimpleExample* application. See *Getting Started with Developing Device Applications* in Chapter 1 for more information about using the IzoT NodeBuilder tool to view these Neuron C source and header files.

*Main.nc* This source file includes a *when*-task that resets the FT 6000 EVB, and a *when*-task that uses a timer to check the state of the switches on the FT 6000 EVB every millisecond.

### **when(reset)**

Power cycles the LCD on the FT 6000 EVB.

This function also outputs debugging information that you can view on your computer. All instances in the example application where debugging information is output to your computer are indicated with **EvalBoardPrintDebug(string)** functions.

To view this debugging output, connect your FT 6000 EVB to your computer via the RS-232 or USB interface on the board and then use a terminal emulation program on your computer. For more information on connecting the FT 6000 EVB to a computer for debugging purposes, see the *FT 6000 EVB Hardware Guide*.

### **when(timer\_expires(tTick))**

This *when*-task uses a timer to check the state of the switches and joystick on the FT 6000 EVB every millisecond, and to check the state of the temperature sensor and light-level sensor every second.

**Note:** Instead of using a timer to evaluate changes or updates to hardware I/O, you could use the **when (io\_changes)** or **when (io\_update\_occurs)** *when*-clauses or an *interrupt*-task. See Chapter 2 of the *Neuron C Programmer's Guide* for more information on these *when*-clauses and Chapter 7 for more information on *interrupt*-tasks.

*FT6000EvalBoard.h* This header file includes the I/O object and function declarations for the I/O devices on the FT 6000 EVB.

*FT6000EvalBoard.nc* This source file includes the following functions that get and set values for the I/O devices on the FT 6000 EVB:

### **void EvalBoardSetLed(Leds whichLed, boolean bOn)**

Sets the state of a specific LED.

### **boolean EvalBoardGetSwitch(Switches whichSwitch)**

Determines which switch has been updated.

**Note:** This file also includes functions for getting values from the light-level sensor, temperature sensor, and joystick on the FT 6000 EVB; however, the *SimpleIsiExample* application does not use these functions.

*Lamp.nc*

This source file implements the **Lamp** functional block. It contains a single *when*-task that illuminates or extinguishes an LED on the FT 6000 EVB when the **nviLamp** network variable in the **Lamp** functional block is updated.

This *when*-task also copies the updated value to the **nvoLampFb** network variable in the **Lamp** functional block. This value can then be passed to any **SNVT\_switch** network variables bound to the **nvoLampFb** network variable in a feedback connection. See *Testing Switch and Lamp Devices* in Chapter 1 for more information on creating and using feedback connections.

*NodeObject.nc*

This source file implements the **NodeObject** functional block. It includes a *when*-task for processing requests received from the **nviRequest** network variable. This function routes **RQ\_NORMAL**, **RQ\_REPORT\_MASK**, **RQ\_UPDATE\_STATUS**, **RQ\_DISABLED**, and **RQ\_ENABLE** requests to the subject functional block or functional blocks in the device application.

**Note:** You must implement the **RQ\_NORMAL**, **RQ\_REPORT\_MASK**, **RQ\_UPDATE\_STATUS** requests in your device application. The **RQ\_DISABLED** and **RQ\_ENABLE** requests are optional.

*Switch.nc*

This source file implements the **Switch** functional block. It includes **ProcessSwitch1()** and **ProcessSwitch2()** functions that get the current state of their respective switches, evaluate whether the state has changed, and if the state of the switch has changed, toggles the value and state of the **nvoSwitch** network variable and copies the updated value and state to the **nviSwitchFb** network variable.

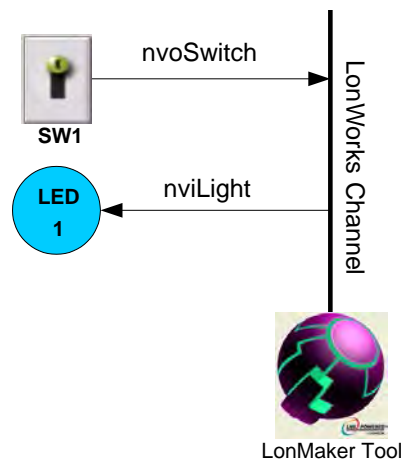
---

## I/O Interaction

The following table lists the I/O pins used for the **SW1** button and **LED1** on the FT 6000 EVB.

I/O	Pin Used	I/O Model
LED1	IO2	Bit I/O
SW1	IO9	Bit I/O

The following graphic illustrates the interaction between the I/O devices on the FT 6000 EVB and the device interface used by the *NcSimpleExample* application.





---

## NcSimpleIsiExample Details

The *NcSimpleIsiExample* application demonstrates how you can use switch devices to activate lamp devices in a self-installed or managed network.

In ISI mode, this example uses one push button (**SW1**) that represents a switch device, one LED (**LED1**) that represents a lamp device, a push button (**SW2**) to initiate and complete an ISI connection, and an LED (**LED2**) that indicates the connection status.

In managed mode, this example uses two push buttons (**SW1** and **SW2**) that represent switch devices and two LEDs (**LED1** and **LED2**) that represent lamp devices.

---

## Device Interface

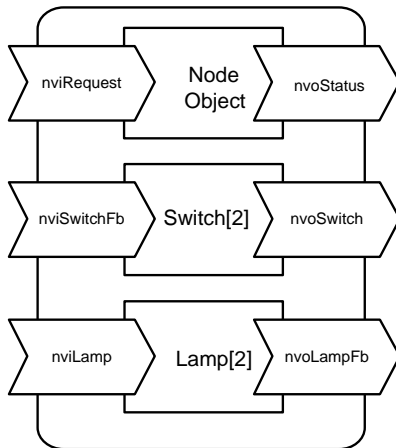
The following section summarizes and then details the *NcSimpleIsiExample* application.

### Summary

The *NcSimpleIsiExample* application includes a Node Object functional block, an array of two Switch functional blocks representing the push button I/O objects on the FT 6000 EVB, and an array of two Lamp functional blocks representing the LED I/O objects on the FT 6000 EVB.

The **Node Object** functional block contains a **SCPTnwrkCnfg** configuration property that stores the current network configuration mode (ISI or managed). This configuration property is implemented as a configuration network variable (CPNV), and it is declared as a non-const device-specific configuration property (a configuration property that can be changed using the device hardware or an LNS network management tool such as the IzoT Commissioning tool). The **Switch** and **Lamp** functional blocks contain **SNVT\_switch** input and output network variables.

The following diagram displays the functional blocks and network variables in the device interface used by the *NcSimpleIsiExample* application

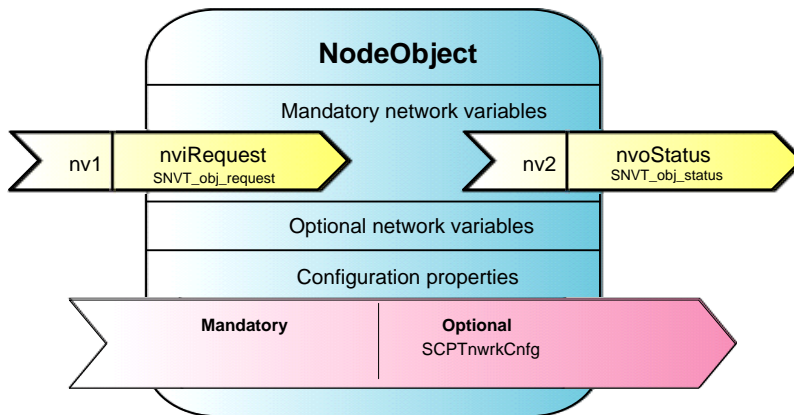


### Details

The following sections detail the **Node Object**, **Switch**, and **Lamp** functional blocks in the *NcSimpleIsiExample* device interface and their network variables and configuration properties.

#### Node Object Functional Block

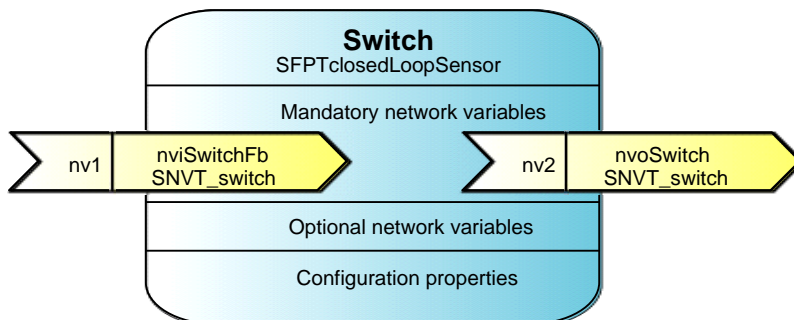
The **NodeObject** functional block returns invalid request for all requests that are not implemented, returns an invalid ID for all IDs that are not implemented, and indicates the current network configuration mode (ISI or managed).



FB/NV/CP Name	Type	Direction	Description
NodeObject	SFPTnodeObject		Lets you monitor the functional blocks within a device.
nviRequest	SNVT_obj_request	Input	Lets you place a functional block in the device interface in a specific state. Handles RQ_NORMAL, RQ_UPDATE_STATUS, RQ_REPORT_MASK and RQ_ENABLE requests.
nvoStatus	SNVT_obj_status	Output	Reports the status of a functional block in the device interface.
nciNetConfig	SCPTnrkCnfg	Input	Stores the current network configuration mode (ISI or managed).

## Switch Functional Blocks

The **Switch** functional blocks represent the **SW1** and **SW2** push buttons on the FT 6000 EVB. They contain **SNVT\_switch** input and output network variables.

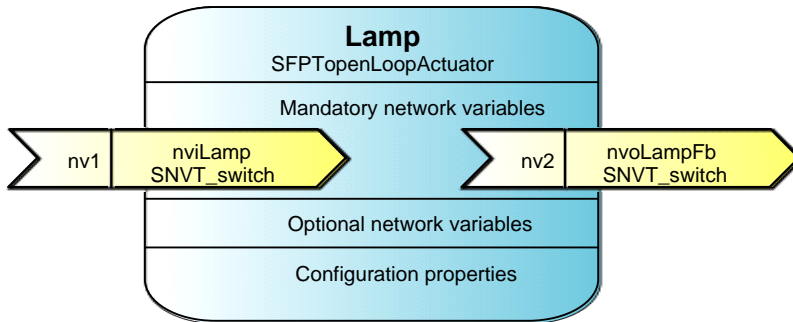


FB/NV Name	Type	Direction	Description
Switch[2]	SFPTclosedLoopSensor		Represents the <b>SW1</b> and <b>SW2</b> push button on the FT 6000 EVB.
nvoSwitch	SNVT_switch	Output	Transmits the values and states of the <b>SW1</b> and <b>SW2</b> buttons on the FT 6000 EVB.

nviSwitchFb	SNVT_switch	Input	Lets you create feedback connections between the <b>SW1</b> and <b>SW2</b> push buttons and <b>SNVT_switch</b> network variables.
-------------	-------------	-------	-----------------------------------------------------------------------------------------------------------------------------------

## Lamp Functional Blocks

The **Lamp** functional blocks represent **LED1** and **LED2** on the FT 6000 EVB. They contain **SNVT\_switch** input and output network variables.



FB/NV Name	Type	Direction	Description
Lamp[2]	SFPTopenLoopActuator		Represents <b>LED1</b> and <b>LED2</b> on the FT 6000 EVB.
nviLamp	SNVT_switch	Output	Receives the value and state of <b>SNVT_switch</b> network variables and updates <b>LED1</b> and <b>LED2</b> on the FT 6000 EVB accordingly.
nviLampFb	SNVT_switch	Input	Lets you send the current value and state of <b>LED1</b> and <b>LED2</b> to <b>SNVT_switch</b> network variables in feedback connections.

## Device Application Summary

The following table summarizes the functionality provided by the Neuron C source (.nc) files and header (.h) files that comprise the *NcSimpleIsiExample* application. See *Getting Started with Developing Device Applications* in Chapter 1 for more information about using the IzoT NodeBuilder tool to view these Neuron C source and header files.

### *Main.nc*

This source file includes a *when*-task that resets the FT 6000 EVB, and a *when*-task that user a timer to check the state of the switches on the FT 6000 EVB every millisecond.

### **when(reset)**

Power cycles the LCD on the FT 6000 EVB, and puts the example application into ISI mode if the **NodeObject.nciNetConfig** configuration property is set to CFG\_LOCAL or CFG\_NUL (it is set to CFG\_NUL the first time the example application is reset so that the example application initially starts in ISI mode).

This function also outputs debugging information that you can view on your computer. All instances in the example application where debugging information is output to your computer are indicated with

**EvalBoardPrintDebug(string)** functions.

To view this debugging output, connect your FT 6000 EVB to your computer via the RS-232 or USB interface on the board and then use a terminal emulation program on your computer.

**when(timer\_expires(tTick))**

This *when*-task uses a timer to check the state of the switches and joystick on the FT 6000 EVB every millisecond, and to check the state of the temperature sensor and light-level sensor every second.

**Note:** Instead of using a timer to evaluate changes or updates to hardware I/O, you could use the **when (io\_changes)** or **when (io\_update\_occurs)** *when*-clauses or an *interrupt*-task. See Chapter 2 of the *Neuron C Programmer's Guide* for more information on these *when*-clauses and Chapter 7 for more information on *interrupt*-tasks.

*FT5000EvalBoard.h*

This header file includes the I/O object and function declarations for the I/O devices on the FT 6000 EVB.

*FT5000EvalBoard.nc*

This source file includes the following functions that get and set values for the I/O devices on the FT 6000 EVB:

**void EvalBoardSetLed(Leds whichLed, boolean bOn)**

Sets the state of a specific LED.

**boolean EvalBoardGetSwitch(Switches whichSwitch)**

Determines which switch has been updated.

**Note:** This file also includes functions for getting values from the light-level sensor, temperature sensor, and joystick on the FT 6000 EVB; however, the *SimpleIsiExample* application does not use these functions.

*IsiImplementation.nc*

This source file contains the device application's implementation of the ISI protocol. See the *ISI Protocol Specification* and *ISI Programmer's Guide* for more information on ISI and developing an application using the Neuron ISI library.

*Lamp.nc*

This source file implements the **Lamp** functional block. It contains a single *when*-task that illuminates or extinguishes an LED on the FT 6000 EVB when the **nviLamp** network variable in the **Lamp** functional block is updated.

This *when*-task also copies the updated value to the **nvoLampFb** network variable in the **Lamp** functional block. This value can then be passed to any **SNVT\_switch** network variables bound to the **nvoLampFb** network variable in a feedback connection. See *Testing Switch and Lamp Devices* in Chapter 1 for more information on creating and using feedback connections.

*NodeObject.nc*

This source file implements the **NodeObject** functional block. It includes a *when*-task for processing requests received from the **nviRequest** network variable. This function routes **RQ\_NORMAL**, **RQ\_REPORT\_MASK**, **RQ\_UPDATE\_STATUS**, **RQ\_DISABLED**, and **RQ\_ENABLE** requests to the subject functional block or functional blocks in the device application.

**Note:** You must implement the **RQ\_NORMAL**, **RQ\_REPORT\_MASK**, **RQ\_UPDATE\_STATUS** requests in your device application. The **RQ\_DISABLED** and **RQ\_ENABLE** requests are optional.

*Switch.nc*

This source file implements the **Switch** functional block. It contains a *when*-task that handles the switch-light connections if the *NcSimpleIsiExample* is running in ISI mode.

It includes **ProcessSwitch1()** and **ProcessSwitch2()** functions that get the current state of their respective switches, evaluate whether the state has changed, and if the state of the switch has changed, toggles the value and state of the **nvoSwitch** network variable and copies the updated value and state to the **nviSwitchFb** network variable.

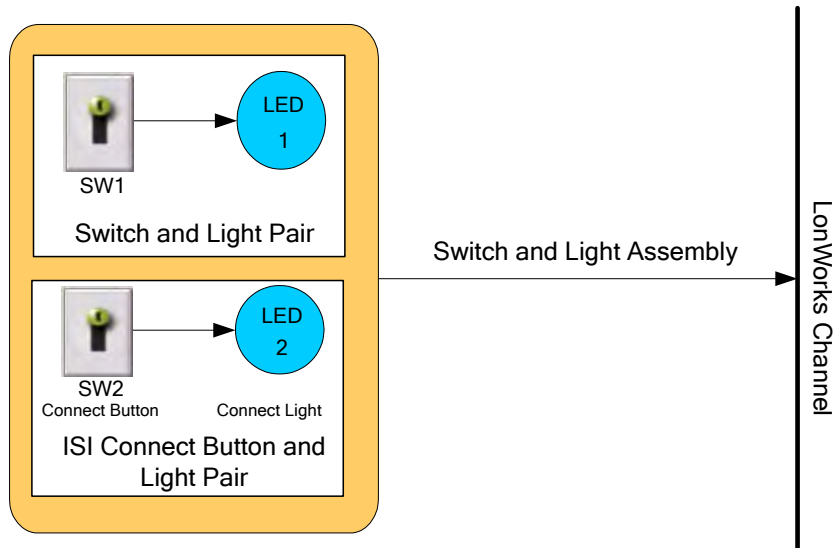
## I/O Interaction

The following table lists the I/O pins used for the **SW1** button, **SW2** button, **LED1**, and **LED2** on the FT 5000 EVB.

I/O	Pin Used	I/O Model
LED1	IO2	Bit I/O
LED2	IO3	Bit I/O
SW1	IO9	Bit I/O
SW2	IO4, IO5, IO6	Bitshift I/O. These pins are also used for input from the Joystick on the FT 6000 EVB.

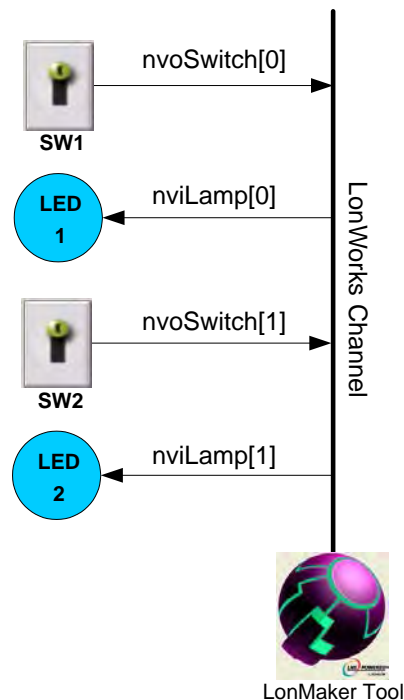
## ISI Mode

The following graphic illustrates the interaction between the I/O devices on the FT 6000 EVB and the device interface used by the *NcSimpleIsiExample* application in ISI mode.



## Managed Mode

The following graphic illustrates the interaction between the I/O devices on the FT 6000 EVB and the device interface used by the *NcSimpleIsiExample* application in managed mode.



---

## NcMultiSensorExample Details

The *NcMultiSensorExample* application demonstrates how you can use switch devices to activate lamp devices in a self-installed or managed network. For managed networks, it also demonstrates how you can use light-level sensor, temperature sensor, joystick, and display devices to view the current temperature, light level (lux), and alarm conditions and set light and temperature alarm conditions for a local or remote device. The FT 6000 EVB ships with the *NcMultiSensorExample* application loaded on it.

In ISI mode, this example is identical to the *NcSimpleIsiExample* application. It uses one push button (**SW1**) that represents a switch device, one LED (**LED1**) that represents a lamp device, a push button (**SW2**) to initiate and complete an ISI connection, and an LED (**LED2**) that indicates the connection status.

In Managed mode, this example uses two push buttons (**SW1** and **SW2**) that represent switch devices and two LEDs (**LED1** and **LED2**) that represent lamp devices, a temperature sensor, a light level sensor, an LCD display, and a joystick used to toggle the information displayed on the LCD and to enter set points for light and temperature alarms.

---

## Device Interface

The following section summarizes and then details the *NcMultiSensorExample* application.

### Summary

The *NcMultiSensorExample* application includes the following functional blocks:

- A Node Object functional block that contains **SNVT\_lux** and **SNVT\_temp\_p** input network variables storing the light and temperature values received from a remote device, and

**SNVT\_alarm\_2** output network variables storing the alarm statuses of the local and remote devices. The **Node Object** functional block also contains a **SCPTnwrkCnfg** configuration property that stores the current network configuration mode (ISI or managed). This configuration property is implemented as a configuration network variable (CPNV), and it is declared as a non-const device-specific configuration property (a configuration property that can be changed using the device hardware or an OpenLNS network management tool such as the IzoT Commissioning tool).

- An array of two **Switch** functional blocks representing the push button I/O objects and an array of two **Lamp** functional blocks representing the LED I/O objects on the FT 6000 EVB. The **Switch** and **Lamp** functional blocks contain **SNVT\_switch** input and output network variables.
- A **LightSensor** functional block representing the light-level sensor I/O object on the FT 6000 EVB. The **LightSensor** functional block includes a **SNVT\_lux** output network variable, and a **SCPTluxSetpoint** configuration property that stores the set point for the light alarm's low limit.
- A **TempSensor** functional block representing the temperature sensor I/O object on the FT 6000 EVB. The **TempSensor** functional block includes a **SNVT\_temp\_p** output network variable, and a **SCPThighLimTemp** configuration property that stores the set point for the temperature alarm's high limit.
- A **Joystick** functional block representing the joystick I/O object (**SW3**) on the FT 6000 EVB. The **Joystick** functional block includes a **SNVT\_angle\_deg** (or **SNVT\_switch**) output network variable and a **SCPTnvType** configuration network variable.

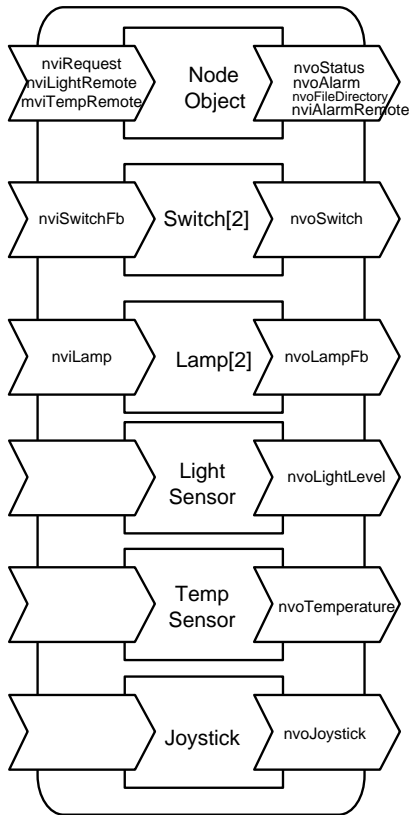
#### Notes:

- The **SCPTnwrkCnfg**, **SCPThighLimTemp**, **SCPTluxSetpoint**, and **SCPTnvType** configuration properties are implemented as configuration network variables (CPNVs). As a result, these network variables appear with an “nci” prefix in a Virtual functional block—instead of in their parent functional blocks—when you are using the *NcMultiSensorExample* device in the IzoT Commissioning tool or other network tool.
- The **Node Object**, **LightSensor**, **TempSensor** and **Joystick** functional blocks implement user-defined functional profile templates (UFPTs) that inherit from standard functional profile templates (SFPTs).

UFPTs are used for the **Node Object**, **LightSensor**, **TempSensor** functional blocks because no SFPT includes the configuration properties required by the example application for setting alarm limits and viewing alarm conditions. A UFPT is used for the **Joystick** functional block because it uses a changeable-type network variable that is not used by the SFPT from which it inherits.

Using UFPTs that inherit from SFPTs—as opposed to using implementation-specific network variables and configuration properties—enables the device application to comply with interoperability guidelines version 3.4 (or better) and pass LONMARK certification. A device application that includes implementation-specific network variables does not comply with interoperability guidelines version 3.4 (or better) and therefore cannot be certified by LONMARK.

The following diagram displays the functional blocks and network variables in the device interface used by the *NcMultiSensorExample* application.



## Details

The following sections detail the **Node Object**, **Switch**, **Lamp**, **LightSensor**, **TempSensor**, and **Joystick** functional blocks in the *NcMultiSensorExample* device interface and their network variables and configuration properties.

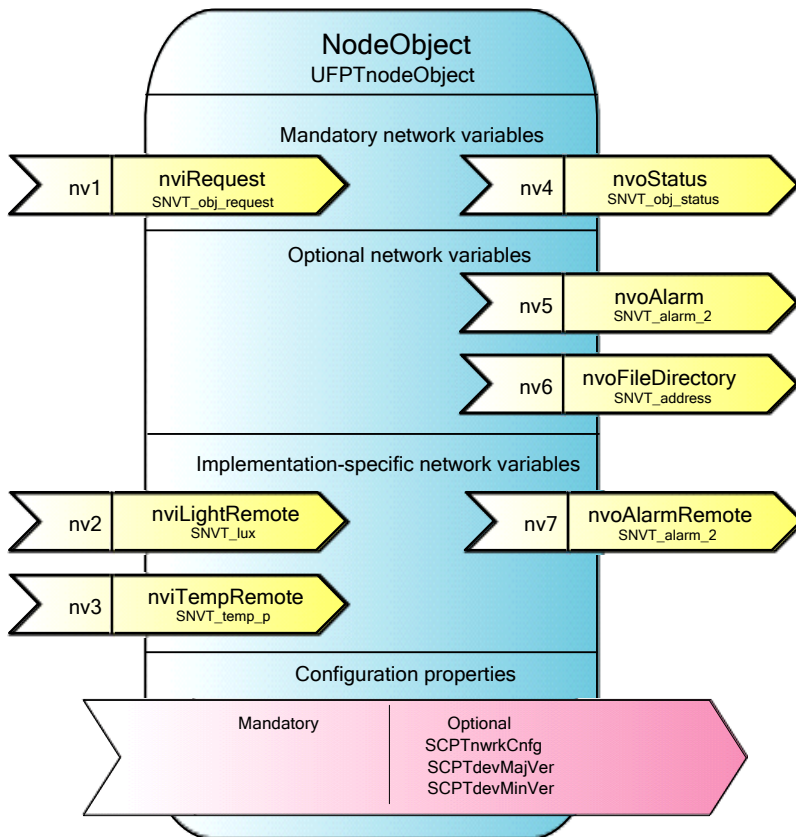
### Node Object Functional Block

The **NodeObject** functional block returns invalid request for all requests that are not implemented, returns an invalid ID for all IDs that are not implemented, and indicates the current network configuration mode (ISI or managed).

The **NodeObject** functional block contains **SNVT\_lux** and **SNVT\_temp\_p** input network variables. These network variables store the light and temperature values received from a remote device. A remote device is another device containing **SNVT\_lux** and/or **SNVT\_temp\_p** output network variables. Remote devices include a second FT 6000 EVB running the *NcMultiSensorExample* application.

The **NodeObject** functional block also contains **SNVT\_alarm\_2** output network variables storing the alarm statuses of a local device (an FT 6000 EVB running the *NcMultiSensorExample* application) and a remote device. You can monitor the light level, temperature, and alarm conditions of local and remote devices from the LCD of the local device.



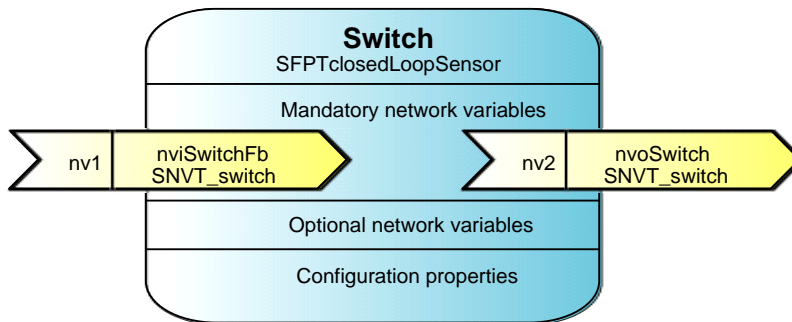


FB/NV/CP Name	Type	Direction	Description
NodeObject	UFPTnodeObject		Lets you monitor the functional blocks within a device.
nviRequest	SNVT_obj_request	Input	Lets you place a functional block in the device interface in a specific state. Handles RQ_NORMAL, RQ_UPDATE_STATUS, RQ_REPORT_MASK and RQ_ENABLE requests.
nvoStatus	SNVT_obj_status	Output	Reports the status of a functional block in the device interface.
nvoAlarm	SNVT_alarm_2	Output	Stores the alarm status of the local device.
nvoFileDirectory	SNVT_address	Output	Address for file directory containing descriptors for configuration parameter files.
nviLightRemote	SNVT_lux	Input	Stores the light value received from a remote device.
nviTempRemote	SNVT_temp_p	Input	Stores the temperature value received from a remote device.
nvoAlarmRemote	SNVT_alarm_2	Output	Stores the alarm status of a remote device.

nciNetConfig	SCPTnwrkCnfg	Input	Stores the current network configuration mode (ISI or managed).
nciDevMajVer	SCPTdevMajVer	Input	Sets the major version number for the example application.
nciDevMinVer	SCPTdevMinVer	Input	Sets the minor version number for the example application.

### Switch Functional Blocks

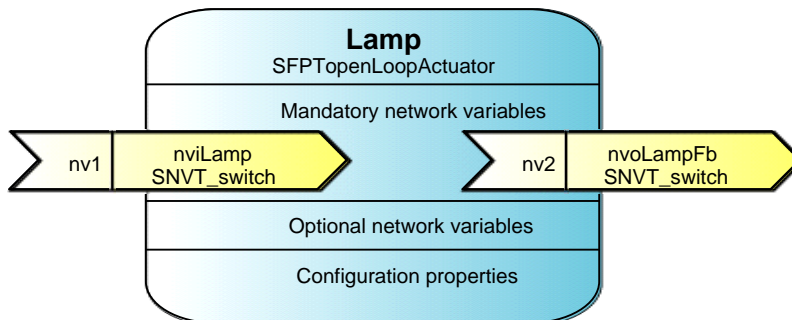
The **Switch** functional blocks represent the **SW1** and **SW2** push buttons on the FT 6000 EVB. They contain **SNVT\_switch** input and output network variables.



FB/NV Name	Type	Direction	Description
Switch[2]	SFPTclosedLoopSensor		Represents the <b>SW1</b> and <b>SW2</b> push buttons on the FT 6000 EVB.
nvoSwitch	SNVT_switch	Output	Transmits the values and states of the <b>SW1</b> and <b>SW2</b> buttons on the FT 6000 EVB.
nviSwitchFb	SNVT_switch	Input	Lets you create feedback connections between the <b>SW1</b> and <b>SW2</b> push buttons and <b>SNVT_switch</b> network variables.

### Lamp Functional Blocks

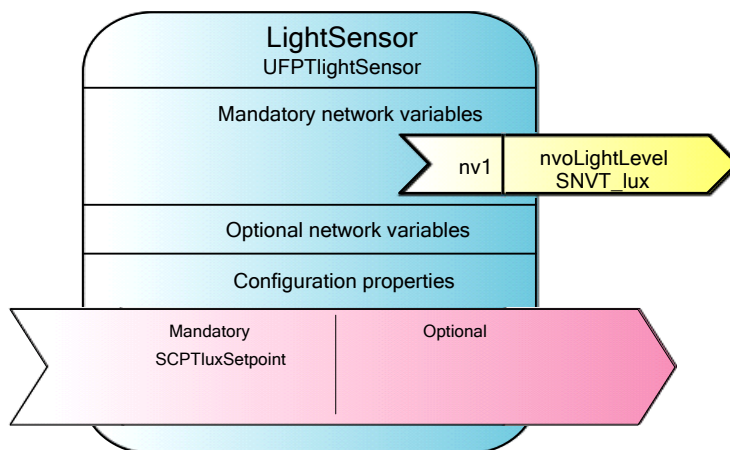
The **Lamp** functional blocks represent **LED1** and **LED2** on the FT 6000 EVB. They contain **SNVT\_switch** input and output network variables.



FB/NV Name	Type	Direction	Description
Lamp[2]	SFPTopenLoopActuator		Represents <b>LED1</b> and <b>LED2</b> on the FT 6000 EVB.
nviLamp	SNVT_switch	Output	Receives the value and state of <b>SNVT_switch</b> network variables and updates <b>LED1</b> and <b>LED2</b> on the FT 6000 EVB accordingly.
nvoLampFb	SNVT_switch	Input	Lets you send the current value and state of <b>LED1</b> and <b>LED2</b> to <b>SNVT_switch</b> network variables in feedback connections.

### Light Sensor Functional Block

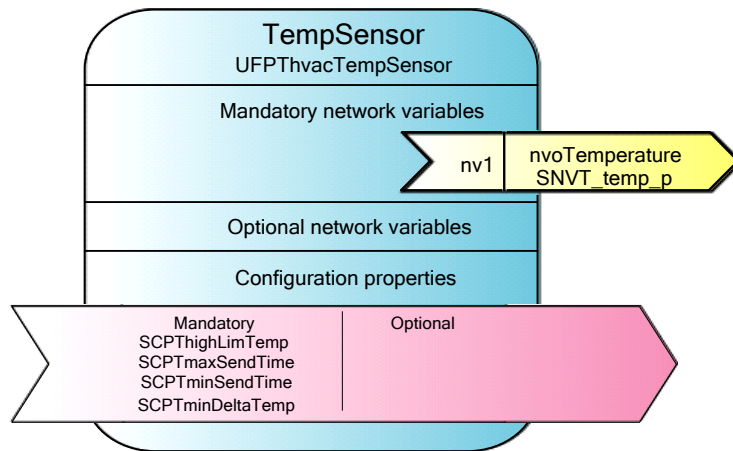
The **LightSensor** functional block represents the light-level sensor I/O object on the FT 6000 EVB. It includes a **SNVT\_lux** output network variable, and a mandatory **SCPTluxSetpoint** configuration property that stores the set point for the light alarm's low limit.



FB/NV/CP Name	Type	Direction	Description
LightSensor	UFPTlightSensor		Represents the light-level sensor I/O object ( <b>LIGHT</b> ) on the right side of the FT 6000 EVB.
nvoLightLevel	SNVT_lux	Output	Transmits the light level (lux) measured by the light-level sensor on the FT 6000 EVB.
nciLowLightAlarm	SCPTluxSetPoint	Input	Stores the set point for the light alarm's low limit.  Implemented as a configuration network variable (CPNV) with a default value of <b>40</b> lux.

### Temperature Sensor Functional Block

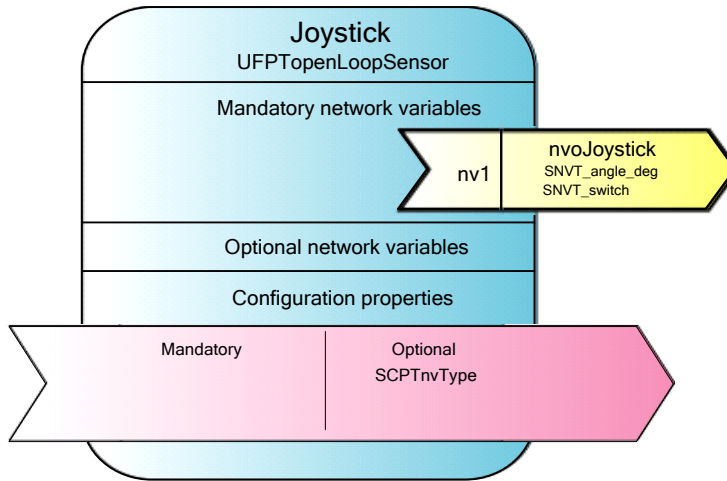
The **TempSensor** functional block represents the temperature sensor I/O object on the FT 6000 EVB. It includes a **SNVT\_temp\_p** output network variable, and a mandatory **SCPTHighLimTemp** configuration property that stores the set point for the temperature alarm's high limit.



FB/NV/CP Name	Type	Direction	Description
TempSensor	UFPTHvacTempSensor		Represents the temperature sensor I/O object ( <b>TEMP</b> ) on the left side of the FT 6000 EVB.
nvoTemperature	SNVT_temp_p	Output	Transmits the temperature (in degrees Celsius) measured by the temperature sensor on the FT 6000 EVB.
nciHighTempAlarm	SCPTHhighLimTemp	Input	Stores the set point for the temperature alarm's high limit. Implemented as a configuration network variable (CPNV) with a default value of <b>35° C</b> .
nciMaxSendTime (heartbeat)	SCPTmaxSendTime	File CP	The maximum period of time that can elapse without an update to <b>nvoTemperature</b> being transmitted.  Implemented as a file configuration property with a default value of <b>60s</b> .
nciMinSendTime (throttle)	SCPTminSendTime	File CP	The minimum period of time that must elapse before updates to <b>nvoTemperature</b> are transmitted.  Implemented as a file configuration property with a default value of <b>1s</b> .
nciMinDelta	SCPTminDeltaTemp	File CP	The minimum change to the value of <b>nvoTemperature</b> required to transmit an update.  Implemented as a file configuration property with a default value of <b>0.5° C</b> .  Applies to the <b>nvoTemperature</b> network variable.

## Joystick Functional Block

The **Joystick** functional block represents the 5-way joystick I/O object (**SW3**) on the FT 6000 EVB. The Joystick functional block includes an output network variable with a changeable type (**SNVT\_angle\_deg** or **SNVT\_switch**) that stores the value of the last button pressed on the joystick. It has a **SCPTnvType** configuration network variable that stores the network variable type used by the joystick output network variable.



FB/NV/CP Name	Type	Direction	Description																					
Joystick	UFPTopenLoopSensor		Represents the joystick I/O object on the bottom center of the FT 6000 EVB.																					
nvoJoystick	SNVT_angle_deg (default) or SNVT_switch	Output	<p>Stores the value of the last button pressed on the joystick.</p> <p>Has a changeable network variable type. By default, it uses a <b>SNVT_angle_deg</b> type, but you can change it to a <b>SNVT_switch</b> type.</p> <p>The possible values for <b>nvoJoystick</b> are as follows:</p> <table><tr><td></td><td><b>SNVT_angle_deg</b></td><td><b>SNVT_switch</b></td></tr><tr><td>Center</td><td>0</td><td>0.0, 1</td></tr><tr><td>Up</td><td>90</td><td>1.0, 1</td></tr><tr><td>Left</td><td>180</td><td>2.0, 1</td></tr><tr><td>Down</td><td>270</td><td>3.0, 1</td></tr><tr><td>Right</td><td>360</td><td>4.0, 1</td></tr><tr><td>Nothing</td><td>-1</td><td>0.0, 0</td></tr></table>		<b>SNVT_angle_deg</b>	<b>SNVT_switch</b>	Center	0	0.0, 1	Up	90	1.0, 1	Left	180	2.0, 1	Down	270	3.0, 1	Right	360	4.0, 1	Nothing	-1	0.0, 0
	<b>SNVT_angle_deg</b>	<b>SNVT_switch</b>																						
Center	0	0.0, 1																						
Up	90	1.0, 1																						
Left	180	2.0, 1																						
Down	270	3.0, 1																						
Right	360	4.0, 1																						
Nothing	-1	0.0, 0																						
nciNvType	SCPTnvType	Input	<p>Stores the current data type used by <b>nvoJoystick</b> (<b>SNVT_angle_deg</b> or <b>SNVT_switch</b>).</p> <p>Implemented as a configuration network variable. Applies to the <b>nvoJoystick</b> network variable.</p>																					

---

## Device Application Summary

The following table summarizes the functionality provided by the Neuron C source (.nc) files and header (.h) files that comprise the *NcMultiSensorExample* application. See *Getting Started with Developing Device Applications* in Chapter 1 for more information about using the IzoT NodeBuilder tool to view these Neuron C source and header files.

### *Main.nc*

This source file includes a *when*-task that resets the FT 6000 EVB, and functions that check the current length of the **nvoJoystick** network variable and check the state of the I/O devices on the FT 6000 EVB:

#### **when(reset)**

Power cycles the LCD on the FT 6000 EVB, starts the light-level sensor on the FT 6000 EVB, and puts the example application into ISI mode if the **NodeObject.nciNetConfig** configuration property is set to CFG\_LOCAL or CFG\_NUL (it is set to CFG\_NUL the first time the example application is reset so that the example application initially starts in ISI mode).

This function also outputs debugging information that you can view on your computer. All instances in the example application where debugging information is output to your computer are indicated with **EvalBoardPrintDebug(string)** functions.

To view this debugging output, connect your FT 6000 EVB to your computer via the RS-232 or USB interface on the board and then use a terminal emulation program on your computer. For more information on connecting the FT 6000 EVB to a computer for debugging purposes, see the *FT 6000 EVB Hardware Guide*.

#### **unsigned get\_nv\_length\_override(unsigned nvIndex)**

Returns the length of the **nvoJoystick** changeable-type network variable in the **Joystick** functional block. This function is required to define the device application behavior when a request to change the network variable type is received.

#### **when(timer\_expires(tTick))**

This *when*-task uses a timer to check the state of the switches and joystick on the FT 5000 EVB every millisecond, and to check the state of the temperature sensor and light-level sensor every second.

**Note:** Instead of using a timer to evaluate changes or updates to hardware I/O, you could use the **when (io\_changes)** or **when (io\_update\_occurs)** clauses or an *interrupt*-task. See Chapter 2 of the *Neuron C Programmer's Guide* for more information on these *when*-clauses and Chapter 7 for more information on *interrupt*-tasks.

#### **void CheckForLocalAlarms(void)**

Checks whether the temperature sensor or alarm sensor on the FT 6000 EVB are in an alarm condition, and reports any alarms to the Local Info Mode panel on the LCD of the local FT 6000 EVB.

#### **void CheckForRemoteAlarms(void)**

Checks whether the **SNVT\_temp\_p** or **SNVT\_lux** network variables on a device bound to the FT 6000 EVB (a remote device) are in an alarm condition, and reports any alarms to the Remote Info Mode panel on the LCD of the FT 6000 EVB (the local device).

<i>FT6000EvalBoard.h</i>	<p>This header file includes the I/O object and function declarations for the I/O devices on the FT 6000 EVB.</p>
<i>FT6000EvalBoard.nc</i>	<p>This source file includes the following functions that get and set values for the I/O devices on the FT 6000 EVB:</p> <p><b>void EvalBoardSetLed(Leds whichLed, boolean bOn)</b></p> <p>Sets the state of a specific LED.</p> <p><b>boolean EvalBoardGetSwitch(Switches whichSwitch)</b></p> <p>Determines which switch has been updated.</p> <p><b>unsigned long EvalBoardGetTemperature(void)</b></p> <p>Gets the current temperature from the temperature sensor on the FT 6000 EVB and scales the raw value to the resolution required by <b>SNVT_temp_p</b>.</p> <p><b>unsigned long EvalBoardGetLightLevel(void)</b></p> <p>Gets the current light level from the light-level sensor on the FT 6000 EVB and passes the value to the <b>CalculateLux()</b> function in the <b>lux.nc</b> file. The <b>CalculateLux()</b> function converts the light level from a raw value to lux.</p> <p><b>JoystickDirection EvalBoardGetJoystick(void)</b></p> <p>Determines in which direction the joystick on the FT 6000 EVB was last pressed.</p>
<i>filesys.h</i>	<p>This header file contains information and function declarations for configuration properties that have been implemented as configuration files.</p> <p>The <b>TempSensor</b> functional block includes <b>SCPTminDeltaTemp</b>, <b>SCPTminSendTime</b>, <b>SCPTmaxSendTime</b> configuration properties that are implemented as configuration files.</p>
<i>IsiImplementation.nc</i>	<p>This source file contains the device application's implementation of the ISI protocol. See the <i>ISI Protocol Specification</i> and <i>ISI Programmer's Guide</i> for more information on ISI and developing an application using the Neuron ISI library.</p>
<i>Joystick.nc</i>	<p>This source file implements the <b>Joystick</b> functional block, and it includes the following functions that process type changes to the <b>nvoJoystick</b> network variable and process changes in the direction of the joystick on the FT 6000 EVB.</p> <p><b>when (nv_update_occurs(nciNvType))</b></p> <p>Calls the <b>ProcessTypeChange()</b> function if the <b>Joystick</b> functional block is enabled. The <b>ProcessTypeChange()</b> function is used to change the type of the <b>nvoJoystick</b> changeable-type network variable. The <b>nvoJoystick</b> network variable uses a <b>SNVT_angle_deg</b> type by default, but it can be changed to a <b>SNVT_switch</b> type.</p> <p><b>void ProcessJoystick(void)</b></p> <p>Stores the current direction of the joystick.</p> <p><b>void ProcessTypeChange(void)</b></p> <p>Changes the type of the <b>nvoJoystick</b> network variable to a</p>

**SNVT\_angle\_deg** or **SNVT\_switch** type. This function first checks whether the type of the **nvoJoystick** network variable is legal processes the change if it is or disables the **Joystick** functional block if it is not, and then calls a function to propagate the appropriate value to the **nvoJoystick** network variable.

*LCD.h* This header file contains the declarations required for and associated with the implementation of the LCD. It includes definitions for the modes used on the LCD, the states used in the Alarm Config Mode, the types of data displayed on the LCD, and various helper functions.

*LCD.nc* This source file contains the functions required for using the LCD. It includes functions that drive the LCD through various modes and display the light, temperature, and alarm condition values on the LCD.

*Lamp.nc* This source file implements the **Lamp** functional block. It contains a single *when*-task that illuminates or extinguishes an LED on the FT 6000 EVB when the **nviLamp** network variable in the **Lamp** functional block is updated.

This *when*-task also copies the updated value to the **nvoLampFb** network variable in the **Lamp** functional block. This value can then be passed to any **SNVT\_switch** network variables bound to the **nvoLampFb** network variable in a feedback connection. See *Testing Switch and Lamp Devices* in Chapter 1 for more information on creating and using feedback connections.

*LightSensor.nc* This source file implements the **LightSensor** functional block. It contains a single *when*-task that sets the value of the **nvoLightLevel** network variable, passes the value to a function in the **LCD.nc** file for display on the LCD, and evaluates light level alarm conditions.

*Lux.nc* This source file contains a single **CalculateLux()** function that converts the light level from a raw value to lux. The raw value is received from the **EvalBoardGetLightLevel()** function in the **FT6000EvalBoard.nc** file.

*NodeObject.nc* This source file implements the **NodeObject** functional block. It includes a *when*-task for processing requests received from the **nviRequest** network variable, and *when*-tasks for processing updates to light level and temperature input network variables values received from a remote device.

**when (nv\_update\_occurs(nviRequest))**

This function routes **RQ\_NORMAL**, **RQ\_REPORT\_MASK**, **RQ\_UPDATE\_STATUS**, **RQ\_DISABLED**, and **RQ\_ENABLE** requests to the subject functional block or functional blocks in the device application.

**Note:** You must implement the **RQ\_NORMAL**, **RQ\_REPORT\_MASK**, **RQ\_UPDATE\_STATUS** requests in your device application. The **RQ\_DISABLED** and **RQ\_ENABLE** requests are optional.

**when (nv\_update\_occurs(nviTempRemote))**

Outputs updates to the **NodeObject.nviTempRemote** input network variable to the LCD on the FT 5000 EVB, and checks the alarm condition of the **nviTempRemote** network variable and outputs it to the LCD.



#### **when (nv\_update\_occurs(nviLightRemote))**

Outputs updates to the **NodeObject**. **nviLightRemote** input network variable to the LCD on the FT 6000 EVB, and checks the alarm condition of the **nviTempRemote** network variable and outputs it to the LCD.

#### **when (nv\_update\_occurs(nciNetConfig))**

Puts the example application in ISI or managed mode when the value of the **NodeObject**. **nciNetConfig** input network variable changes.

#### *Switch.nc*

This source file implements the **Switch** functional block. It contains a *when*-task that handles the switch-light connections if the *MultiSensorExample* is running in ISI mode.

It includes **ProcessSwitch1()** and **ProcessSwitch2()** functions that get the current state of their respective switches, evaluate whether the state has changed, and if the state of the switch has changed, toggles the value and state of the **nvoSwitch** network variable and copies the updated value and state to the **nviSwitchFb** network variable.

#### *TempSensor.nc*

This source file implements the **TempSensor** functional block. It contains a **ProcessTemperatureSensor()** that gets the current temperature, evaluates whether the temperature change is greater than the value defined in the **nciMinDelta** configuration property, which is 0.5°C, and if so sets the **nvoTemperature** network variable to the current temperature.

The function propagates the value of the **nvoTemperature** network variable on the network as long as the **nciMaxSendTime** (heartbeat) configuration property is greater than the **nciSendTime** (throttle) configuration property. These configuration properties control how often updates to the **nvoTemperature** network variable are propagated.

The **ProcessTemperatureSensor()** function also passes the temperature to a function in the **LCD.nc** file for display on the LCD, and evaluates temperature alarm conditions.

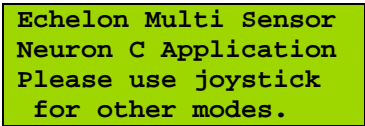
---

## **Using the Joystick and LCD**

When you run the *NcMultiSensorExample* application in managed mode, the Joystick lets you navigate the panels in the LCD and set the light and temperature alarm levels. The following sections describe how to use the joystick and LCD on the FT 6000 EVB.

### *Welcome Panel*

When you start the *NcMultiSensorExample* application, the LCD displays the **Welcome Mode** panel.



Echelon Multi Sensor  
Neuron C Application  
Please use joystick  
for other modes.

You can toggle the joystick down or press the center button to display the **Local Info Mode** panel.

### *Local Info Mode Panel*

The **Local Info Mode** panel displays the current temperature, light level, and alarm conditions of the local FT 6000 EVB running the *NcMultiSensorExample* application.

Local Info Mode		
Light :	400	Lux
Temp :	25.0	C
Alarms:	None	

If the current light level or temperature reaches or exceeds their limits set in the **Alarm Config Mode** panel, that alarm will appear in the **Alarms** property. For example, if both the light level and temperature exceed their alarm set points, the **Alarms** property in the **Local Info Mode** panel would appear as follows:

Local Info Mode		
Light :	12	Lux
Temp :	36.5	C
Alarms:	Light & Temp	

You can display the current temperature in Fahrenheit by toggling the joystick sideways. You can change it back to Celsius by toggling the joystick sideways again. Changing the temperature format in the **Local Info Mode** panel also changes the format used for the high temperature alarm setpoint in the **Alarm Config Mode** panel.

Local Info Mode		
Light :	400	Lux
Temp :	97.7	F
Alarms:	Temp	

You can toggle the joystick down or press the center button to display the **Remote Info Mode** panel. You can toggle the joystick up to return to the **Welcome Mode** panel.

### Remote Info Mode Panel

The **Remote Info Mode** panel displays the current temperature, light level, and alarm conditions of a remote device that you have connected to an FT 6000 EVB running the *NcMultiSensorExample* application in managed mode. A remote device may be any device containing **SNVT\_lux** and/or **SNVT\_temp\_p** output network variables, including a second FT 6000 EVB running the *NcMultiSensorExample* application.

You can connect the network variables of two FT 5000 EVBs running the *NcMultiSensorExample* application in managed mode using the IzoT Commissioning tool or other network tool. For more information how to do this, see *Creating Connections in Managed Mode* in Chapter 1.

Remote Info Mode		
Light :	420	Lux
Temp :	27.0	C
Alarms:	None	

You can display the current temperature of the remote device in Fahrenheit by toggling the joystick sideways. You can change it back to Celsius by toggling the joystick sideways again.

If you have not connected your FT 6000 EVB to a remote device, “Disconnected” appears in the **Light** and **Temp** properties.

Remote Info Mode		
Light :	Disconnected	
Temp :	Disconnected	
Alarms:	None	

You can toggle the joystick down or press the center button to display the **Alarm Config Mode** panel. You can toggle the joystick up to return to the **Local Info Mode** panel.

## Alarm Config Mode Panel

This panel displays the set points for the light and temperature alarm conditions and lets you change them.

The **Light** property displays the lower alarm limit for the lux of the local device. If the current lux is less than or equal to this value, an alarm condition for the light will be displayed in the **Local Info Mode** panel. The default light lower alarm limit is **40** lux. This alarm setpoint is stored in the **nciLowLightAlarm** configuration property in the **LightSensor** functional block.

The **Temp** property displays the upper alarm limit for the temperature of the local device. If the current temperature is greater than or equal to this value, an alarm condition for the light will be displayed in the **Local Info Mode** panel. The default temperature upper alarm limit is **35.0°C (95.0°F)**. This alarm setpoint is stored in the **nciHighTempAlarm** configuration property in the **TempSensor** functional block.

```
Alarm Config Mode
Light :   40      Lux
Temp  :   35.0    C
Cancel<- Ok
```

To change the light and temperature alarm conditions, follow these steps:

1. Toggle the joystick up once to change the setpoint for the temperature alarm. Press the joystick left button to decrease the temperature 0.5°C, and press the joystick right button to increase the temperature 0.5°C.
2. Toggle the joystick up again to change the setpoint for the light alarm. Press the joystick left button to decrease the lux by 5, and press the joystick right button to increase the lux by 5.
3. To save your changes and return to the **Welcome Mode** panel, Toggle the joystick down once or twice to point to **Ok** (once if the pointer is at the **Temp** property, and twice if the pointer is at the **Light** property), and then press the center button on the joystick.

To cancel all changes and return to the **Welcome Mode** panel, Toggle the joystick down once or twice (once if the pointer is at the **Temp** property, and twice if the pointer is at the **Light** property), press the joystick left button to point to **Cancel**, and then press the center button on the joystick.

---

## I/O Interaction

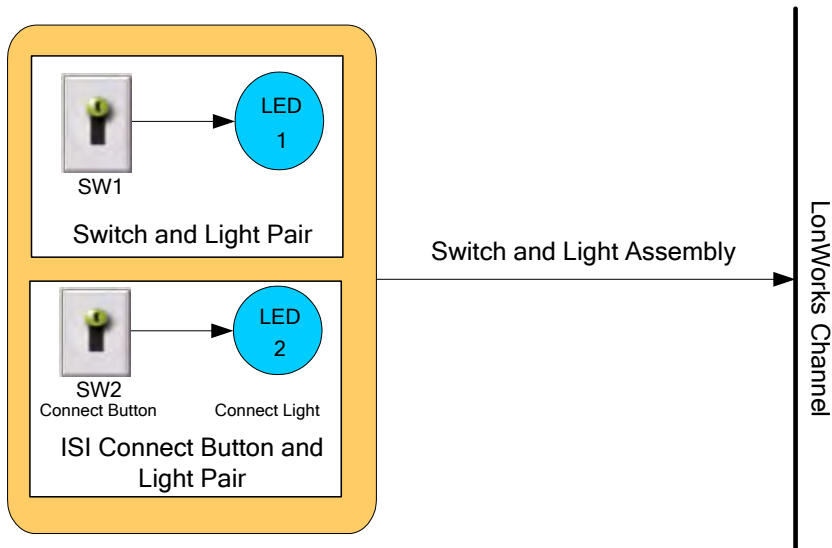
The following table lists the I/O pins used for the **SW1** button, **SW2** button, **LED1**, **LED2**, the **LIGHT** level sensor, the **TEMP** sensor, and joystick (**SW3**) on the FT 5000 EVB.

I/O	Pin Used	I/O Model
LED1	IO2	Bit I/O
LED2	IO3	Bit I/O
SW1	IO9	Bit I/O
SW2	IO4, IO5, IO6	Bitshift I/O. These pins are also used for input from the Joystick on the FT 6000 EVB.
TEMP	IO7	Touch I/O. The temperature sensor uses a 1-Wire Dallas DS18S20 digital thermometer device.
LIGHT	IO0, IO1	I <sup>2</sup> C. The I <sup>2</sup> C address for the light level sensor is 0x39.
SW3 (Joystick)	IO4, IO5, IO6	Bitshift I/O. These pins are also used for input from the <b>SW2</b> button. A total of 6 bits are used in each shift

I/O	Pin Used	I/O Model
		operation (5 joystick inputs and 1 switch input).
LCD	IO0, IO1	I <sup>2</sup> C. The I <sup>2</sup> C address for the LCD is 0x28.

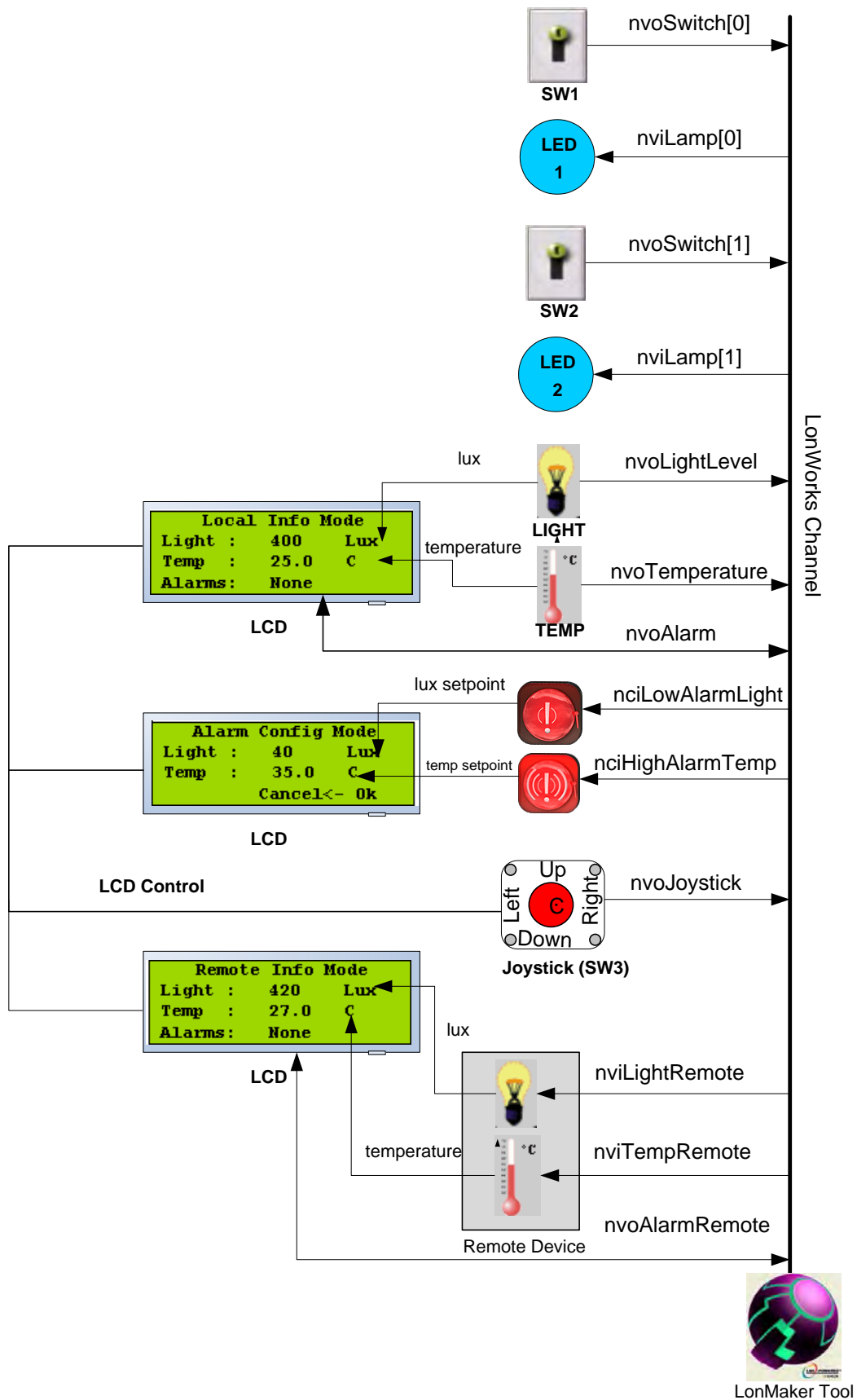
### ISI Mode

The following graphic illustrates the interaction between the I/O devices on the FT 6000 EVB and the device interface used by the *NcMultiSensorExample* application in ISI mode. This is the same as the *NcSimpleIsiExample*.



### Managed Mode

The following graphic illustrates the interaction between the I/O devices on the FT 6000 EVB and the device interface used by the *NcMultiSensorExample* application in managed mode.





# Appendix A

## Glossary

This appendix provides definitions for many terms commonly used with the FT 6000 EVB example applications.

## **Application Device**

A LONWORKS device that runs an ISO/IEC 14908-1 application (OSI Layer 7). The application may run on a Neuron Chip or Smart Transceiver, in which case the device is called a “Neuron hosted” device.

## **Application Image**

Device firmware that consists of the object code generated by the Neuron C compiler from the user’s application program and other application-specific parameters, including the following:

- Network variable fixed and self-identification data
- Network variable device interface data
- Program ID string
- Optional self-identification and self-documentation data
- Number of address table entries
- Number of domain table entries
- Number and size of network buffers
- Number and size of application buffers
- Number of receive transaction records
- Input clock speed of target Neuron Chip or Smart Transceiver
- Transceiver type and bit rate

## **Application Program**

The software code in a LONWORKS device that defines how it functions. The application program, also referred to as the *application*, may be in the device when you purchase it, or you may load it into the device from application image files (.APB, .NDL, and .NXE extensions) using the IzoT Commissioning tool or other network management tool. The application program interfaces with the ISO/IEC 14908-1 firmware to communicate over the network. It may reside completely in the Neuron Chip or Smart Transceiver, or it may reside on an attached host processor (in a host-based device).

## **Binding**

Process of connecting network variables. Binding creates logical connections (virtual wires) between LONWORKS devices. Connections define the data that devices share with one another. Tables containing binding information are stored in the device’s non-volatile memory, and may be updated by the LonMaker tool or the ISI protocol.

## **Changeable-Type Network Variable**

A network variable that has a type and length that can be changed to that of another network variable type of equal or smaller size. You can use changeable-type network variables to implement generic functional blocks that work with different types of inputs and outputs.

## **Channel**

The physical media between devices upon which the devices communicate. The ISO/IEC 14908-1 protocol is media independent; therefore, numerous types of media can be used for channels: twisted pair, power line, fiber optics, IP, and RF, and other types.

## **Commissioning**

The process in which the IzoT Commissioning tool or other network management tool downloads network and application configuration data into a physical device. For devices whose application programs are not contained in ROM, the network management tool also downloads the application program into non-volatile RAM in the device. Devices are usually either commissioned and tested one at a time, or commissioned and then brought online and tested incrementally.

## **Configuration Properties (CPs)**

Configuration properties are data values that define the behavior of an application device by determining the manner in which device application data is manipulated and when device application



data is transmitted. Configuration properties can be applied to the device, functional block, or network variable level. Configuration properties can determine the functions to be performed on the values stored in network variables. For example, a configuration property may specify a minimum change that must occur on a physical input to a device before the corresponding output network variable is updated.

### **Configured**

A device state where the device has both an application image and a configured network image. This indicates that the device is ready for network operation.

### **Connection**

The implicit addressing established during binding. A connection links one or more logical outputs (network variables or message tags) to one or more logical inputs. A connection may be represented with a connector shape or a reference connection.

### **Connect Button**

A button on an ISI device that the user can press to create a connection. The Connect button on an FT 6000 EVB running the *NcSimpleIsiExample* or *NcMultiSensorExample* application is the **SW2** button on the right side of the board.

### **Connect Light**

An LED on an ISI device that provides feedback related to the status of an ISI connection. The Connect light on an FT 6000 EVB running the *NcSimpleIsiExample* or *NcMultiSensorExample* application is **LED2**, which is located directly above the **SW2** button.

### **Connection Host**

A device that initiates the enrollment process by sending a connection invitation specifying a connection assembly.

### **Connection Member**

A device that has joined an ISI connection, but is not the connection host.

### **Connector Shape**

A single connector used to connect a pair of network variables within the same subsystem.

### **Control Network Protocol (CNP)**

The ISO/IEC 14908-1 Control Network Protocol. The CNP is a complete seven-layer communications protocol, with each layer optimized to the needs of control applications. The seven layers follow the reference model for open systems interconnection (OSI) developed by the International Standard Organization (ISO).

### **Data Point**

A network variable, configuration property, or functional block state (enabled or in override) that the IzoT Commissioning tool can monitor and/or control.

### **Data Point Shape**

A shape in the LonMaker Basic Stencil of the IzoT Commissioning tool that you can use to monitor and control the values of network variables and configuration properties, and the states of functional blocks (enabled or in override).

### **Device**

A device that communicates on a LONWORKS network using CNP. A device may be an application device, network service device, or a router. Devices are sometimes referred to as nodes in LONWORKS documentation.

## **Device Interface**

The logical interface to a device, abbreviated as *XIF*. A device's interface specifies the number and types of functional blocks; number, types, directions, and connection attributes of network variables; and the number of message tags. The program ID for a device is used as the key to identify each device interface. Each program ID uniquely defines the static portion of the interface. However, two devices with identical static portions may differ if dynamic network variables are added or removed, or if the types of changeable network variables are changed. Thus it is possible to have devices with the same program ID but different device interfaces.

### **Device Interface File (XIF)**

A file that documents a device's interface with a network. The file can be a text file (**.XIF** extension), or it can be a binary file (**.XFB** extension).

### **Device-Specific Configuration Property**

A configuration property that has values that can be modified independent of the network database. Changes made to a device-specific configuration property are not updated in the network database.

### **Device Template**

A device template contains all the attributes of a given device type, such as its functional blocks, network variables, and configuration properties. You can create a device template by importing a device interface (XIF) file supplied by the device manufacturer, or by uploading the device interface definition from the physical device. A device template is identified by its name and its program ID. Both must be unique within a network—you cannot have two device templates with the same name or the same program ID in a single network.

### **Download**

An installation process in which data, such as the application program, network configuration, and/or application configuration, is transferred over the network into a device.

### **Free Topology**

A connection scheme for the communication bus that eases traditional transmission line restrictions of trunks and drops of specified lengths and at specified distances, and terminations at both ends. Free topology allows wire to be strung from any point to any other, in bus, daisy chained, star, ring, or loop topologies, or combinations thereof. It only requires one termination anywhere in the network. This can reduce the cost of wiring significantly.

### **FT 6000 EVB**

A LONWORKS evaluation board that uses Echelon's FT 6000 Smart Transceiver. It features a compact design that includes the following I/O devices that you can use to develop prototype devices and run the FT 6000 EVB examples: 4 x 20 character LCD display, 4-way joystick with center push button, 2 push-button inputs, 2 LED outputs, light-level sensor, and temperature sensor.

### **FT 6000 Smart Transceiver**

A chip that integrates a high-performance Neuron 6000 processor core and a TP/FT-10 transceiver. The FT 6000 Smart Transceiver, combined with an FT-X3 Communications Transformer and inexpensive serial memories, provides a lower-cost, higher-performance alternative to the previous generation LONWORKS TP/FT-10 solution.

### **FT 3150 EVB**

A LONWORKS evaluation board that uses Echelon's FT 3150 Smart Transceiver. It is connected to a MiniGizmo board that includes eight push buttons, eight LEDs, a temperature sensor, and a piezo buzzer. In a managed network, you can bind compatible network variables in applications running on the FT 3150 EVB and FT 5000 EVBs. In a self-installed network, you can use the ISI protocol to connect the FT 3150 EVB running the *MGSwitch*, *MGLight*, or *MGDemo* applications to an FT 5000 EVB running the *NcSimpleIsiExample* or *NcMultiSensorExample* applications.

## **FT 3120 EVB**

A LONWORKS evaluation board that uses Echelon's FT 3120 Smart Transceiver. It is connected to a MiniGizmo board that includes eight push buttons, eight LEDs, a temperature sensor, and a piezo buzzer. In a managed network, you can bind compatible network variables in applications running on the FT 3120 EVB and FT 5000 EVBs. In a self-installed network, you can use the ISI protocol to connect the FT 3120 EVB running the *MGSwitch* or *MGLight* applications to an FT 5000 EVB running the *NcSimpleIsiExample* or *NcMultiSensorExample* applications.

## **Functional Block (FB)**

A collection of network variables, configuration properties, and associated behavior that defines a specific system functionality. Functional blocks define standard formats and semantics for how information is exchanged between devices on a network. Each functional block implements a functional profile.

## **Functional Block Array**

A set of identical functional blocks. A functional block array is useful if your device contains two or more identical switches, lights, dials, controllers, or other I/O devices that will each have an identical external interface. In addition, a functional block array saves code space and reduces the number of when-tasks in your code.

## **Functional Profile**

A template for a functional block that enables equipment specifiers to select the functionality they need for a system. Each functional profile defines mandatory and optional network variable and configuration property members along with their intended usage. A number of generic standard functional profiles are available for generic devices such as simple sensor and actuators. Many industry-specific standard functional profiles are available for industry-specific applications. Industry-specific standard profiles are developed through a review and approval process, including a cross-functional review to ensure the profile will interoperate within an individual subsystem and also provide interoperability with other subsystems in the network.

User-defined functional profiles can be created if no appropriate standard profiles are available.

## **I/O Object**

An instantiation of an I/O model. An I/O object consists of a specific I/O model, and its pin assignment, modifiers, and name.

## **i.LON IP-852 Router**

An i.LON IP-852 router forwards ISO/IEC 14908-2 packets enveloped in ISO/IEC 14908-4 packets over an IP-852 channel. i.LON IP-852 routers include the i.LON SmartServer with IP-852 routing, i.LON 100 e3 plus Internet Server with IP-852 routing, and the i.LON 600 LONWORKS-IP Server.

## **IP-852 Channel**

Also known as an ISO/IEC 14908-4 channel or an ANSI/CEA-852 LONWORKS/IP channel, an IP-852 channel carries ISO/IEC 14908-1 packets enveloped in ISO/IEC 14908-4 packets. An IP-852 channel is a LONWORKS channel that uses a shared IP network to connect IP-852 devices and is defined by a group of IP addresses. These IP addresses form virtual wires that connect IP-852 devices so they can communicate with each other. IP-852 devices include the LNS Server computers, LNS client computers, LonMaker computers, and i.LON IP-852 routers. An IP-852 channel enables a client computer to connect directly to a LONWORKS network and perform monitoring and control tasks.

## **IP-852 Network Interface**

An IP-852 network interface enables IP-852 devices such as LNS Server computers, LNS client computers, LonMaker computers, and i.LON IP-852 routers to be attached to IP-852 channels. An IP-852 network interface requires that the LONWORKS-IP Configuration Server be configured before trying to communicate with remote devices or remote computers.

## **Interoperable Self-Installation (ISI) Protocol**

The standard protocol for performing self-installation in LONWORKS networks. ISI is an application-layer protocol that lets you install and connect devices without using a separate network management tool. It is typically used in home networks, and may be used in any network with less than 200 devices with simple connection and configuration requirements.

## **ISI Mode**

An installation scenario in which the ISI protocol is used (instead of the IzoT Commissioning tool or other network tool) to install devices and create network variables connections.

## **IzoT Commissioning Tool**

The commissioning network tool that uses Visio as its graphical user interface. The IzoT Commissioning tool is used to design, commission, maintain, and document distributed control networks comprised of both LONMARK and other LONWORKS devices. The IzoT Commissioning Tool works with a LonMaker Browser and LonMaker Drawings to provide a graphical representation of your network and a table view of network variables and configuration properties.

## **IzoT NodeBuilder Tool**

A hardware and software platform that is used to develop applications for Neuron Chips and Echelon Smart Transceivers. The IzoT NodeBuilder tool provides complete support for creating, debugging, testing, and maintaining LONWORKS devices. You can use the IzoT NodeBuilder tool to create many types of devices, including VAV controllers, thermostats, washing machines, card-access readers, refrigerators, lighting ballasts, blinds, and pumps. You can use these devices in a variety of systems including building controls, factory automation, and transportation.

## **IzoT Router**

The router located on the FT 6000 EVK is an IzoT Router with FT and Ethernet interfaces. It includes the IzoT ServerStack and the FT terminators. The IzoT router can be used to connect FT 6000 EVBs to a host that is running the IzoT NodeBuilder software and the IzoT Commissioning Tool.

## **Local Client**

An LNS application running on the same computer as the LNS Server.

## **Local Device**

An FT 5000 EVB board running the *NcMultiSensorExample* application that receives **SNVT\_lux** and/or **SNVT\_temp\_p** output network variable updates from another device (a remote device). The local device displays the temperature and light level values received from the remote device in the Remote Info Mode panel on its LCD. A remote device may be another FT 5000 EVB board running the *NcMultiSensorExample* application.

## **LonMaker Browser**

An LNS plug-in that provides a table view of the network variables and configuration properties of selected devices and/or functional blocks. The LonMaker Browser can be used to monitor and control the network variables and configuration properties in a network.

## **LonMaker Drawing**

A LonMaker drawing contains the graphical representation of a LONWORKS network.

## **LonMaker Network Design**

A LonMaker network design consists of an LNS network database and a LonMaker drawing.

## **LonMaker Shape**

A reusable drawing object related specifically to a LONWORKS device.

**LonMark**

A distinctive logo applied to LONWORKS devices that have been certified to the interoperability standards of the LONMARK Interoperability International.

**LonTalk Protocol**

Echelon's implementation of the ISO/IEC 14908-1 Control Network Protocol (CNP). The LonTalk protocol provides a standard method for devices on a LONWORKS network to exchange data. The LonTalk protocol defines the format of the messages being transmitted between devices, and it defines the actions expected when one device sends a message to another. The protocol normally takes the form of embedded software or firmware code in each device on the network.

**LONWORKS 2.0 Platform**

The next generation of LONWORKS products designed to both increase the power and capability of LONWORKS devices, and to decrease the costs of device development and devices.

**LONWORKS Network**

A network of intelligent devices (such as sensors, actuators, and controllers) that communicate with each other using a common protocol over one or more communications channels.

**LONWORKS Technology**

The technology that allows for the creation of open, interoperable control networks that communicate with the ISO/IEC 14908-1 Control Network Protocol. LONWORKS technology consists of the tools and components required to build intelligent device and to install them in control networks.

**Managed Network**

A network where a shared network management server, such as OpenLNS, is used to perform network installation.

**Mandatory Network Variable/Configuration Property**

A network variable/configuration property that must be implemented by the functional block, as specified by the functional profile that the functional block is instantiating.

**Monitored Connection**

A network variable connection in which the current values are being monitored, typically by an HMI. The connector shape and reference connection in a LonMaker drawing demonstrate monitored connections.

**Network Interface**

A LONWORKS device that provides a layer 2 or layer 5 LonTalk interface to an external host computer such as a computer or a handheld maintenance tool. Network interfaces include IP-852 interfaces.

**Network Variable (NV)**

Network variables allow a device to send and receive data over the network to and from other devices. Network variables are data items (such as temperature, the state of a switch, or actuator position setting) that a particular device application program expects to receive from other devices on the network (an *input network variable*) or expects to make available to other devices on the network (an *output network variable*).

**Network Variable/Configuration Property Types**

A network variable or configuration property type defines the structure and contents of the data object. A network variable type can be either a standard network variable type (SNVT) or a user-defined network variable type (UNVT). A configuration property type can be a standard configuration property type (SCPT) or a user-defined configuration property type (UCPT).

## Neuron 5000 Processor

Echelon's next-generation Neuron chip designed for the LONWORKS 2.0 platform. The Neuron 5000 processor is faster, smaller, and cheaper than previous-generation Neuron chips. The Neuron 5000 processor includes a fourth processor for interrupt service routine (ISR) processing.

The Neuron 5000 processor supports an internal system clock speed of 5 MHz to 80 MHz (using a 10 MHz external crystal). The Neuron 5000 processor includes 16KB of on-chip ROM to store the Neuron firmware image and 64 KB on-chip RAM (44 KB is user-accessible). The Neuron 5000 processor requires at least 2KB of off-chip EEPROM to store configuration data, and you can use a larger capacity EEPROM device or an additional flash device (up to 64KB) to store your application code, configuration data, and an upgradable Neuron firmware image. The Neuron 5000 processor supports the mapping of external non-volatile memory from 0x4000 to 0xDFFF in the Neuron address space (a maximum of 42KB).

## Neuron C

A programming language based on ANSI C that you can use to develop applications for Neuron Chips and Smart Transceivers. It includes network communication, I/O, interrupt-handling, and event-handling extensions to ANSI C, which make it a powerful tool for the development of LONWORKS device applications.

## Neuron Chip

A semiconductor component specifically designed for providing intelligence and networking capabilities to low-cost control devices. The Neuron Chip includes a communication port for connections to various network types.

## Neuron Core

The Neuron core includes up to four processors that provide both communication and application processing capabilities. Two processors execute the layer 2 through 6 implementation of the ISO/IEC 14908-1 Control Network Protocol and the third executes layer 7 and the application code. Series 5000 chips include a fourth processor for interrupt service routine (ISR) processing.

## Neuron Firmware

A complete operating system including an implementation of the ISO/IEC 14908-1 protocol used by a Neuron Chip or Smart Transceiver. The Neuron firmware is a program that is inserted into memory of a Neuron Chip or Smart Transceiver.

## Neuron ID

A 48-bit number assigned to each Neuron core at manufacture time. Each Neuron Chip has a unique Neuron ID, making it like a serial number.

## Node Object

A functional block that monitors the status of all functional blocks in a device and makes the status information available for monitoring by the LonMaker tool. A LONMARK-compliant device that has more than one functional block must have a node object.

## Non-const Device-specific Configuration Property

A configuration property that can be changed by the device application, an LNS network tool such as the LonMaker tool, or another tool not based on LNS. An example of a non-const device-specific configuration property is the **SCPTnwrkCnfg** configuration property in the **Node Object** functional block of the *NcMultiSensorExample* and *NcSimpleIsiExample* applications. This configuration property stores the current network configuration mode (ISI or managed) of the example application.

## OffNet

A management mode in which network configuration changes are stored in the network database, but not propagated to the devices on the network. To send the changes to the devices, you place the

LonMaker tool OnNet. If the LonMaker tool is OffNet and attached to the network, you can still perform read operations on the network.

### **OnNet**

A management mode in which network configuration changes are propagated immediately to the devices on the network.

### **OpenLNS**

A network operating system that provides services for interoperable LONWORKS installation, maintenance, monitoring, and control tools such as the IzoT Commissioning tool. Using the services provided by the OpenLNS client/server architecture, tools from multiple vendors can work together to install, maintain, monitor, and control LONWORKS networks. The OpenLNS architecture consists of the following elements:

1. OpenLNS client applications, which can be used to develop, monitor and control LONWORKS networks.
2. The OpenLNS Object Server ActiveX Control, which is a language-independent programming interface for OpenLNS client applications to access the LONWORKS network.
3. The OpenLNS Server, which manages the network and maintains a database containing the network configuration.

### **OpenLNS Network Database**

Each LONWORKS network has its own OpenLNS network database (also referred to as the network database) that is managed and maintained by an OpenLNS Server. The network database includes the network and device configuration data for that network. The network database also contains extension records, which are user-defined records for storing application data.

### **OpenLNS Server Computer**

A computer running the OpenLNS Server software. The OpenLNS Server computer contains the OpenLNS global database, which includes the group of LONWORKS networks being managed by the OpenLNS Server, plus a network database for each network managed by the server.

### **Optional Network Variable/Configuration Property**

A network variable or configuration property listed as an optional component in a functional profile. Functional blocks can elect not to implement optional network variables or configuration properties specified by the functional profile that the functional block is instantiating.

### **PCC-10**

A type II PC (formerly PCMCIA) card network interface that includes an integral TP/FT-10 transceiver. Other transceiver types can be connected to the PCC-10 via external transceiver “pods”.

### **PCLTA-20/21**

A ½ size ISA card network interface.

### **Peer-To-Peer**

A control strategy in which independent intelligent devices share information directly with each other and make their own control decisions without the need or delay of using an intermediate, central, or master controller. Because of the enhanced system reliability introduced by eliminating the master (a single point of failure) and the reduced installation and configuration cost inherent in peer-to-peer designs, LONWORKS technology is intended to implement a peer-to-peer control strategy.

### **Program ID**

A unique, 16-hex digit ID that uniquely identifies the device application.

### **Remote Client**

An LNS application that communicates with the LNS Server (running on a separate computer) over a LonWorks channel (an IP-852 or TP/XF-1250 channel) or over an LNS/IP interface. The NodeBuilder tool cannot be run on a remote client, but the LonMaker tool and other LNS client software can.

### **Remote Device**

A device that sends **SNVT\_lux** and/or **SNVT\_temp\_p** output network variables updates to an FT 5000 EVB running the *NcMultiSensorExample* application (the local device). The temperature and light level values are displayed in the Remote Info Mode panel on the LCD of the local device. A remote device may be another FT 5000 EVB running the *NcMultiSensorExample* application.

### **Remote Network Interface (RNI)**

A network interface that enables you to connect an LNS or OpenLDV-based application to a LONWORKS network via a TCP/IP connection. RNIs include the *i.LON SmartServer*, *i.LON 100 e3* plus Internet Server, and *i.LON 600 LONWORKS-IP* Server.

### **Resource File**

A file included with a LONWORKS device that defines the components of the device interface to be used by integration and development tools. Defined components include network variable types, configuration property types, and functional profiles implemented by the device application. Resource files hold definitions of standard and user-defined resources, including network variable and configuration property types, functional profiles, enumerations, and formatting rules to display network variable and configuration properties in a readable form. Resource files are used during device development, installation and management. Standard resource files are distributed by LONMARK International. User-defined resource files are created and managed during device development.

### **SLTA-10**

A serial network interface with built-in twisted pair transceiver that connects to any host with an EIA-232 (formerly RS232) port. It can also connect to the host remotely using a modem.

The SLTA-10 network interface is supported, but not recommended unless dial-up operation through a modem and a serial connection is required. You should use a PCC-10 or U10 USB network interface instead. For accessing remote networks, you can use an RNI such as the *i.LON SmartServer*, *i.LON 100 e3* plus Internet Server, or *i.LON 600 LONWORKS-IP* Server.

### **Self-Installed Network**

A network that has network addresses and connections created without the use of a network management tool. In a self-installed network, each device contains code (the Neuron C ISI library, which implements the ISI protocol) that replaces parts of the network management server's functionality, resulting in a network that no longer requires a special tool or server to establish network communication or to change the configuration of the network.

### **Service Button**

A push button or other actuator on a LONWORKS device that is used during installation to acquire the device's Neuron ID. For a Neuron hosted device, the button is connected to the service pin of the Neuron Chip or Smart Transceiver. When this pin is activated, the Neuron core sends a broadcast message containing its Neuron ID and program ID, which is called service pin message or packet. The method used to implement the Service button varies from device to device. Examples of mechanical methods include grounding via a push button or using a magnetic reed switch. By attaching one of the device's I/O pins to the service pin, the service pin can also be put under software control as long as the application code is being executed. For example, the device can ground the pin when the device is moved or when a predefined series of I/O occurs. The service pin can also be used to drive an LED that indicates the device's state. The service LED is solid on when the device is applicationless, blinks slowly when the device has an application and is unconfigured, is off when the device has an application and is configured. Some applications also implement additional service pin blink patterns.



### **Standard Configuration Property Type (SCPT)**

A standard configuration property type defined by LONMARK International to facilitate interoperability. SCPTs are defined for a wide range of configuration properties used in many kinds of functional profiles, such as hysteresis bands, default values, minimum and maximum limits, gain settings, and delay times. SCPTs should be used in a LONWORKS network wherever applicable. In situations where there is not an appropriate SCPT available, manufacturers may define UCPTs for configuring their devices.

In addition to standard or user-defined network variable types, which define the data type, formatting rules, limits and units, SCPT also define semantics. For example, the **SNVT\_time\_sec** standard network variable type defines a data type for exchanging durations of time, in seconds. The **SCPTmaxSentTime** standard configuration property type references **SNVT\_time\_sec**, but adds semantics by clarifying that this configuration property defines the maximum period of time between consecutive transmissions of the current value. See [types.lonmark.org](http://types.lonmark.org) for a current list and documentation.

### **Standard Functional Profile**

A standard set of functional profiles defined by LONMARK International. See [types.lonmark.org](http://types.lonmark.org) for a current list and documentation. See *Functional Profile* for more information about functional profiles.

### **Standard Network Variable Type (SNVT)**

A standard set of network variable types defined by LONMARK International to facilitate interoperability by providing a well-defined interface for communication between devices made by different manufacturers. See [types.lonmark.org](http://types.lonmark.org) for a current list and documentation.

### **Stencil**

A collection of master shapes that can be reused in Visio.

### **TP/FT-10**

The free topology twisted pair LONWORKS channel type, 78Kbps bit rate.

### **U10 USB Network Interface.**

A low-cost, high-performance LONWORKS network interface with a built-in TP/FT-10 transceiver that can be used with USB-enabled computers and controllers.

### **User-defined Configuration Property Type (UCPT)**

A non-standard data structure used for configuration of the application program in a LONMARK device. UCPTs should be used only when there is no appropriate standard configuration property type (SCPT) defined. LONMARK-certified devices must have UCPTs documented in resource files according to a standard format, in order to allow the devices to be configured without the need for proprietary configuration tools. See *Standard Configuration Property Type (SCPT)* for more information on configuration property types.

### **User-defined Functional Profile**

A non-standard functional profile defined by a device manufacturer. A user-defined functional profile should be used only when there is no appropriate standard functional profile defined. See *Functional Profile* for more information about functional profile templates.

The *NcMultSensor* example uses four UFPTs that inherit from existing SFPTs. Three of the UFPTs are required because no SFPT includes the configuration properties required by the example application for setting alarm limits and viewing alarm conditions. Another UFPT is required because it uses a changeable-type network variable that is not used by the SFPT from which it inherits.

### **User-defined Network Variable Type (UNVT)**

A non-standard network variable type defined by the manufacturer of a device. UNVTs should be used only when there is no appropriate standard network variable type (SNVT) defined.

LONMARK-certified devices must have UNVTs documented in resource files according to a standard format, in order to allow the devices to be interoperable.

**Virtual Functional Block**

A static functional block that contains the network inputs and outputs for a device that are not part of other functional blocks on the device.

