FT 6000 Examples Guide

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

078-0505-01C

Printed in the United States of America.
Copyright ©1997–2014 by Echelon Corporation.
Echelon Corporation
www.echelon.com

Contents

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® and IzoT networks. You can browse the Neuron® 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 64-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. 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 theIzoT 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*.

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

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* 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 *lonsupport@echelon.com*
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 *lonsupport@echelon.co.uk*
Japan	Japanese	Echelon Japan Holland Hills Mori Tower, 18F 5-11-2 Toranomon, Minato-ku Tokyo 105-0001 Japan Phone: +81-3-5733-3320 Fax: +81-3-5733-3321 *lonsupport@echelon.co.jp*
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 *lonsupport@echelon.com.cn*

Region	Languages Supported	Contact Information
Other Regions	English Japanese	Phone: +1-408-938-5200 Fax: +1-408-328-3801 *lonsupport@echelon.com*

1

Using the FT 6000 EVB Examples

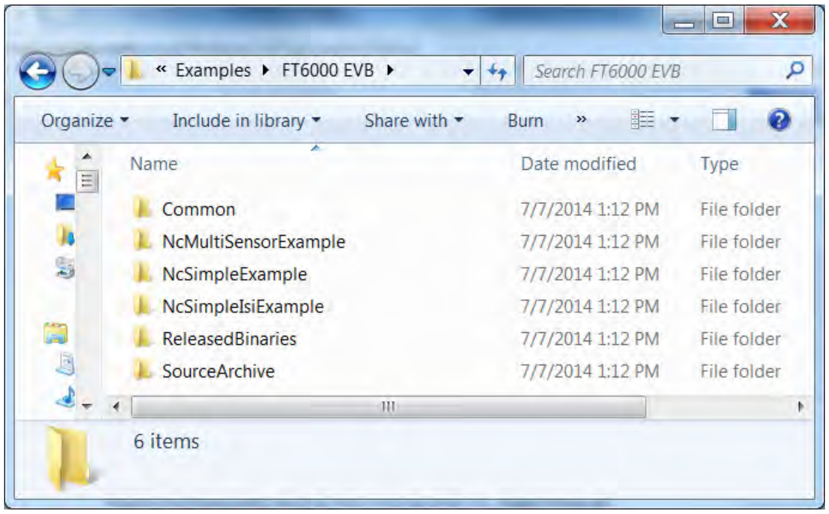
This chapter introduces the three Neuron C example applications that you can run on an FT 6000 EVB. It describes how to load these example applications on an FT 6000 EVB using the IzoT Commissioning Tool which is included with the IzoT FT 6000 EVK. It describes how to bind the example applications in 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 Development Tool includes three Neuron C example applications that you can run on your FT 6000 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.



Name	Date modified	Type
Common	7/7/2014 1:12 PM	File folder
NcMultiSensorExample	7/7/2014 1:12 PM	File folder
NcSimpleExample	7/7/2014 1:12 PM	File folder
NcSimpleIsiExample	7/7/2014 1:12 PM	File folder
ReleasedBinaries	7/7/2014 1:12 PM	File folder
SourceArchive	7/7/2014 1:12 PM	File folder

Each example folder contains the following files and subfolders:

Source	This folder contains the example IzoT NodeBuilder project and all source code files and header files used by the example.
Types	In the **NcMultiSensorExample** folder only, this folder contains definitions for the user-defined functional profiles (UFPTs) developed for the example.

IzoT CT Network Backup (.zip)	An IzoT Commissioning Tool network backup file (**.zip**) that includes an OpenLNS database and IzoT Commissioning Tool 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. 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 (.**APB** extension) that is stored in the **LonWorks\NeuronC\Examples\FT6000 EVB\ReleasedBinaries\** <*Example*> folder. This folder also contains a pre-built text device interface file (**.XIF** extension) that exposes the example application's device interface so that the IzoT Commissioning tool can manage the example application.

See *Using the IzoT Commissioning Tool to Load Example Applications* later in this chapter for how to restore the IzoT CT 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 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 an IzoT CT network backup (IzoT CT drawing and OpenLNS network database) that you can restore to begin running the example applications in a managed network. The IzoT CT 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**	
NcSimpleExample	**Summary**	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	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.
	Installation Mode	You must use IzoT Commissioning Tool or other network tool to commission the **SimpleExample** device and to connect the Switch and Lamp network variables so that pressing the push button illuminates and extinguishes the LED.
	Program ID	9F:FF:FF:05:01:04:04:10
NcSimpleIsiExample	**Summary**	Demonstrates how you can use switch devices to activate lamp devices in a self-installed or managed network. **<u>ISI Mode</u>** In self-installed mode (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. **<u>Managed Mode</u>** 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	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). The Switch and Lamp functional blocks contain **SNVT_switch** input and output network variables.
	Installation Mode	You can run the **SimpleIsiExample** in ISI or managed mode. In managed mode, you must use the IzoT Commissioning Tool or other network tool to commission the **SimpleISIExample** device and to connect the Switch and Lamp network variables so that pressing the push buttons illuminate and extinguish the LEDs.
	Program ID	9F:FF:FF:05:01:04:04:20

Example Application	**Description**	
NcMultiSensorExample	**Summary**	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. A local device refers to one FT 6000 EVB running the *NcMultiSensorExample* application. A remote device refers to another device containing **SNVT_lux** and/or **SNVT_temp_p** output network variables that are connected to the **SNVT_lux** and/or **SNVT_temp_p** input network variables on the local device. A remote device may be a second FT 6000 EVB running the *NcMultiSensorExample* application. You can use a local device to monitor the temperature, light level, and alarm conditions of a remote device. **<u>ISI Mode</u>** 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. **<u>Managed Mode</u>** 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	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 the remote device, and **SNVT_alarm_2** output network variables storing the alarm statuses of the local and remote devices. The Node Object also contains a **SCPTnwrkCnfg** configuration property that stores the current network configuration mode (ISI or managed). • 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.

Example Application		Description
		• 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 on the FT 6000 EVB. The Joystick functional block includes a **SNVT_angle_deg** output network variable that you can change to a **SNVT_switch** type, and a **SCPTnvType** configuration property. **Note**: 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.
	Installation Mode	You can run the **MultiSensorExample** in ISI or managed mode. In managed mode, you must use the IzoT Commissioning Tool or other network tool to commission the **MultiSensorExample** device and to connect the network variables so that the example application responds to the I/O devices on the FT 6000 EVB.
	Program ID	9F:FF:FF:05:01:84:04:30

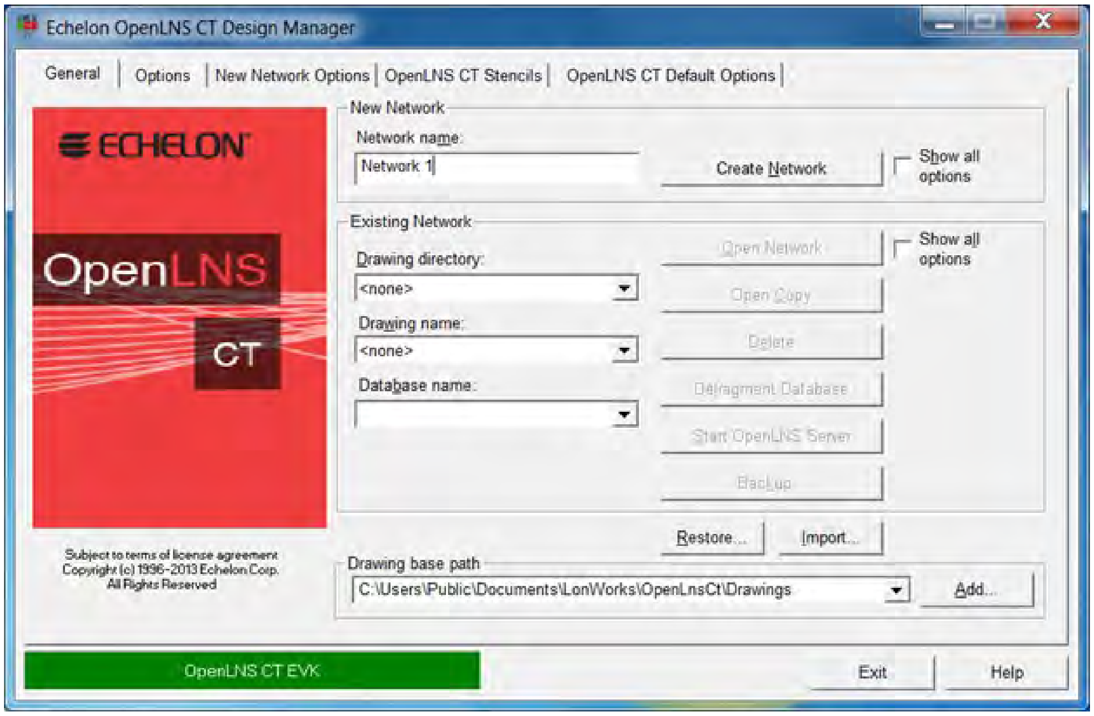
You can download the example applications to the FT 6000 EVBs using the IzoT Commissioning Tool, which is included with the IzoT FT 6000 EVK. After you download an example application to the FT 6000 EVBs, you can install and connect the network variables of the example applications in a self-installed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

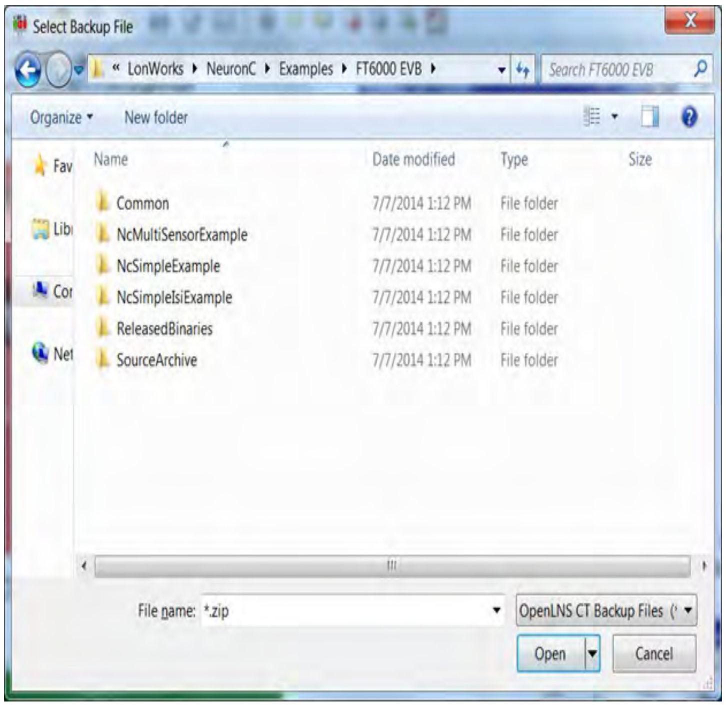
You can use the IzoT Commissioning Tool to download the example applications to the FT 6000 EVBs and install them in a LONWORKS network. To do this, you restore an IzoT Commissioning tool network backup that contains device and functional block shapes for that example application, load the pre-built 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 OpenLNS CT** folder, and then click **OpenLNS Commissioning Tool**. The OpenLNS CT Design Manager opens.



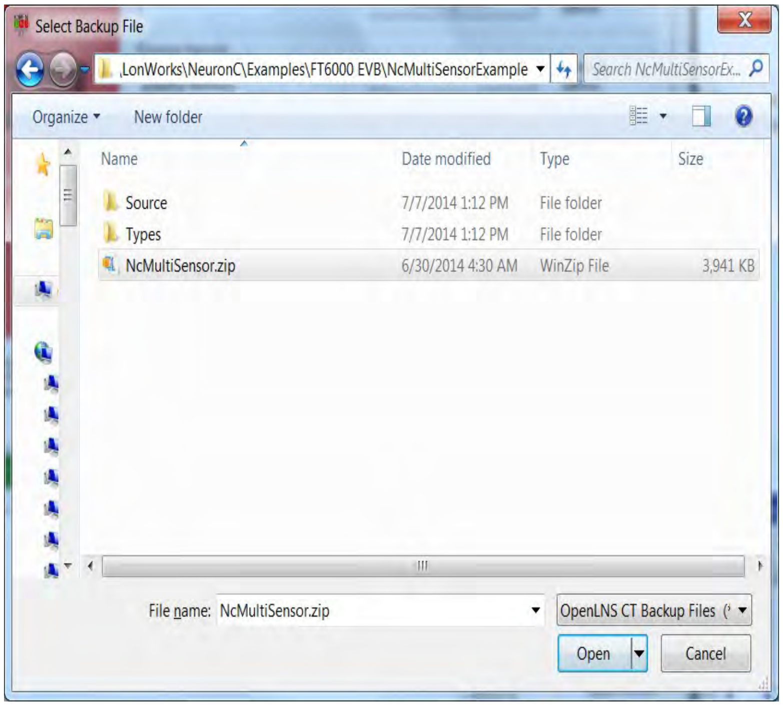
4. Optionally, you can add the folder containing the IzoT Commissioning Tool network backup files for the example applications to the drawing base path. This speeds up the process of restoring the backup files for the other example applications. To do this, click **Add** under **Settings**. The **Browse 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).



Name	Date modified	Type	Size
Common	7/7/2014 1:12 PM	File folder	
NcMultiSensorExample	7/7/2014 1:12 PM	File folder	
NcSimpleExample	7/7/2014 1:12 PM	File folder	
NcSimpleIsiExample	7/7/2014 1:12 PM	File folder	
ReleasedBinaries	7/7/2014 1:12 PM	File folder	
SourceArchive	7/7/2014 1:12 PM	File folder	

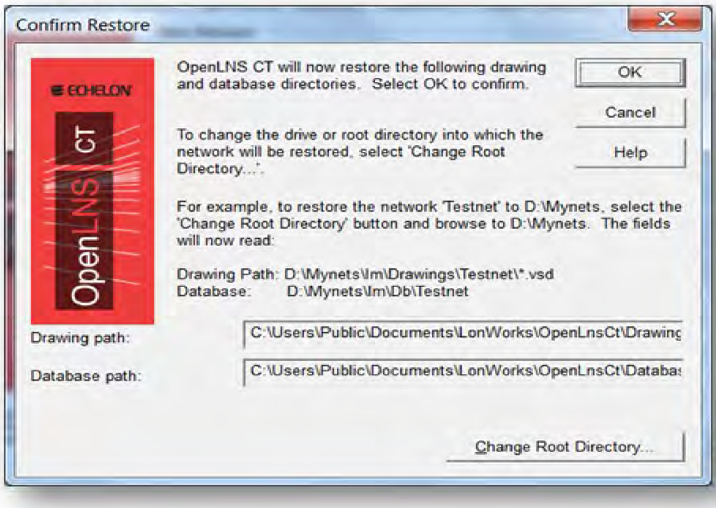
File name: *.zip


Select Backup File
LonWorks\NeuronC\Examples\FT6000 EVB\NcMultiSensorExample
Search NcMultiSensorEx...
Organize ▾
New folder
OpenLNS CT Backup Files
Open
Cancel

Name	Date modified	Type	Size
Source	7/7/2014 1:12 PM	File folder	
Types	7/7/2014 1:12 PM	File folder	
NcMultiSensor.zip	6/30/2014 4:30 AM	WinZip File	3,941 KB

File name: NcMultiSensor.zip

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



OpenLNS CT will now restore the following drawing and database directories. Select OK to confirm.

To change the drive or root directory into which the network will be restored, select 'Change Root Directory...'.

For example, to restore the network 'Testnet' to D:\Mynets, select the 'Change Root Directory' button and browse to D:\Mynets. The fields will now read:

Drawing Path: D:\Mynets\lm\Drawings\Testnet\*.vsd
Database: D:\Mynets\lm\Db\Testnet

Drawing path: C:\Users\Public\Documents\LonWorks\OpenLnsCt\Drawing

Database path: C:\Users\Public\Documents\LonWorks\OpenLnsCt\Databa

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® 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.

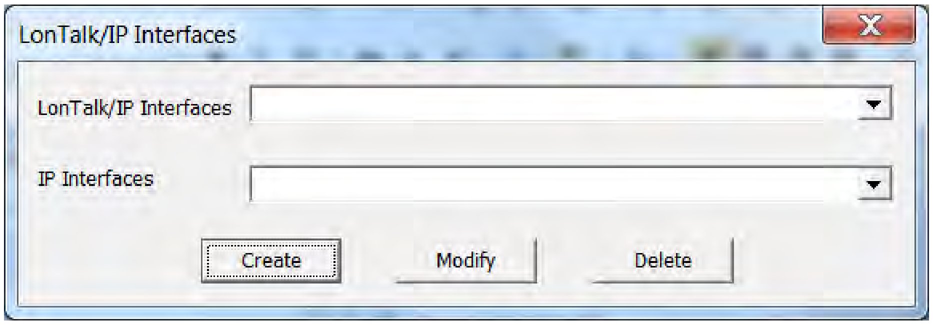
9. A message appears informing you that the network restore operation has been completed, and prompting you to select whether to open the IzoT CT network in order to recommission devices that have changed since the network was backed up. Click **Yes**.

   - 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**.

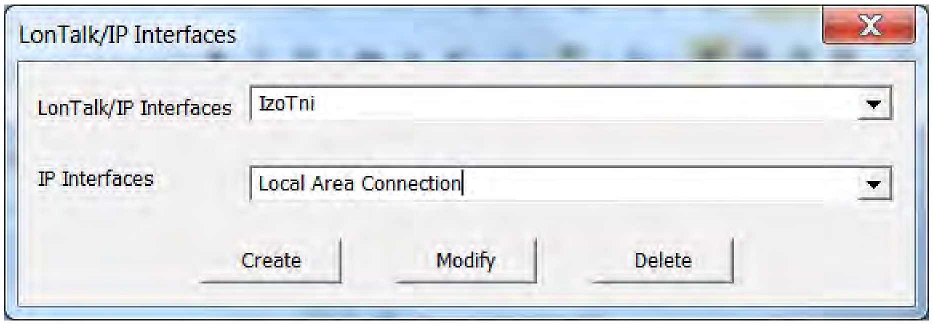
   - A warning may appear asking you if you want to enable macros. You must enable macros for the IzoT Commissioning tool to function.

10. Create a LonTalk/IP Network Interface

   - Start the LonTalk/IP Interfaces application from the Start menu under **All Programs** > **Echelon IzoT Network Services Utilities** > **IzoT Network Services LonTalk-IP Interfaces**.



   - Type a name such as **LonTalk IP** in the **LonTalk/IP Interfaces** field. This will be the name you supply when specifying an interface to NodeUtil and in the IzoT Commissioning Tool.



LonTalk/IP Interfaces: IzoTni

IP Interfaces: Local Area Connection

   - Select **Local Area Connection** as an IP network interface for this LonTalk/IP interface to use in the **IP Interface** box.
   - Click **Create**.
   - Close the LonTalk/IP Interfaces application.

11. Find the IP Address of your IzoT Router
   - To open a Windows command prompt, click the Windows **Start** button, type **cmd** in the search box, and then press the **Enter** key.
   - At the command prompt, type **NodeUtil –d**<*interfaceName*>. Enclose the interface name in quotes if there are any spaces in the name. For example: NodeUtil –d"LonTalk IP"
   - Press the **Service/Connect** button on your IzoT Router. NodeUtil displays a message similar to the following: `Received an ID message from device 1`

- Type **G** to go to a device, type **1** to go to device 1, and then press the **Enter** key.
- Type **S** to report device status and statistics to display status information about the IzoT Router. The last line of the display includes the IzoT Router's IP address.
- Type **N** to preserve the router's statistics. You can type **Y** instead to reset the router's statistics counters.
- Type **E** to exit the device menu.
- Type **E** to exit NodeUtil.
- Close the command prompt.

12. The FT 6000 EVK includes three Neuron C example applications that you can use to test the I/O devices on the FT 6000 EVBs, and to create and manage a network with the FT 6000 EVBs and your IzoT Router. The FT 6000 EVB comes with the NcMultiSensorExample application loaded. This application uses the push buttons, LEDs, temperature sensor, light sensor, LCD, and joystick on the FT 6000 EVB. You can load a different example in the FT 6000 EVB, and you can also load your own custom applications in the FT 6000 EVB.

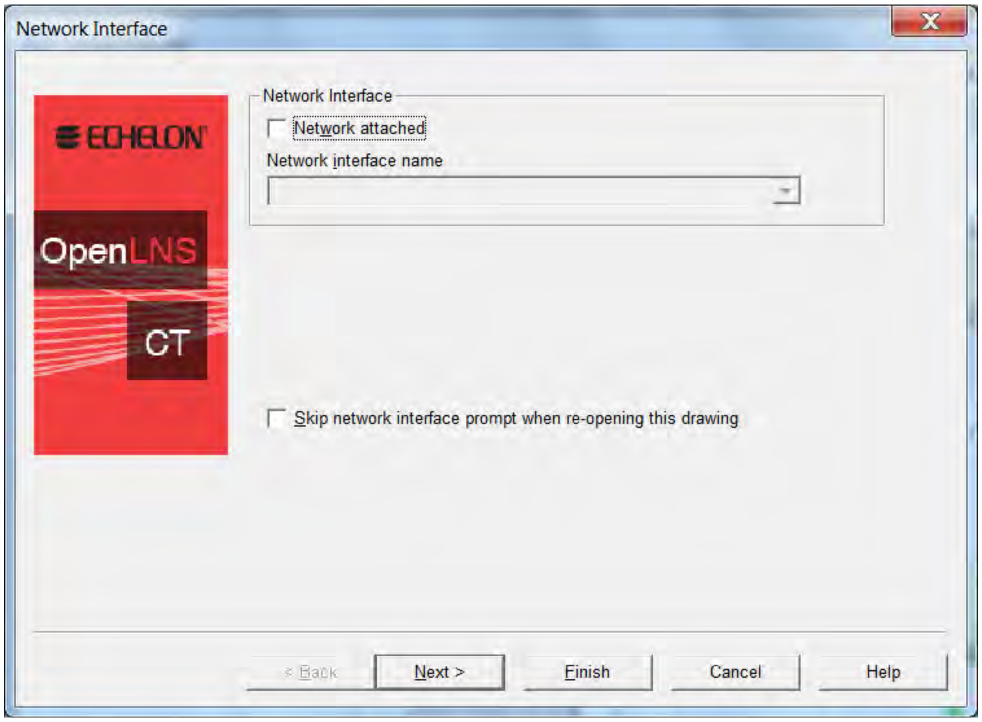
    You can create a network with the FT 6000 EVBs in two ways—you can either use Interoperable Self-Installation (ISI) to enable the EVBs to be connected to each other by pressing Connect buttons on each, or you can use the IzoT Commissioning Tool to specify the devices to be installed in a network and to create connections. To create a network using ISI, see the *FT 6000 EVB Examples Guide* for details. To create a network using the IzoT Commissioning Tool and your FT 6000 EVBs with the NcMultiSensorExample application, follow these steps:

    1. To add an IP route for your LonTalk/IP Ethernet and FT channels, open a Windows command prompt and enter the following commands:

    ```
    route –p add 10.128.1.1 mask 255.255.255.255 <IzoT Router IP address>
    route –p add 10.128.2.0 mask 255.255.255.0 <IzoT Router IP address>
    ```

    The 10.128.1.1 and 10.128.2.0 IP addresses encode the domain ID length (1 byte) and domain ID value of 128 (0x80) used for the FT 6000 EVK network. The first route encodes subnet 1 node 1 for the Ethernet side of the IzoT router, and the second route encodes subnet 2 for the FT side of the router and the EVB devices, as assigned in the IzoT CT network drawing. If you are using a different IzoT CT network you can use the LonTalkIpAddressCalculator application to translate LonTalk addresses into IP addresses. If you are installing your devices onto a network that already uses any IP addresses in the ranges of 10.128.1.0 to 10.128.1.255 or 10.128.2.0 to 10.128.2.255 IP addresses, change the domain for the IzoT CT network as described in the *IzoT Commissioning Tool User's Guide*. This will be the case if you install multiple FT 6000 EVBs on the same IP network. For more information on IP addresses for LonTalk/IP devices, see *How Devices Communicate Using Network Variables* in the *Neuron C Programmer's Guide*. For more information on the LonTalkIpAddressCalculator, see the *IzoT NodeBuilder ReadMe* document.

13. The Network Wizard opens with the Network Interface window displayed.



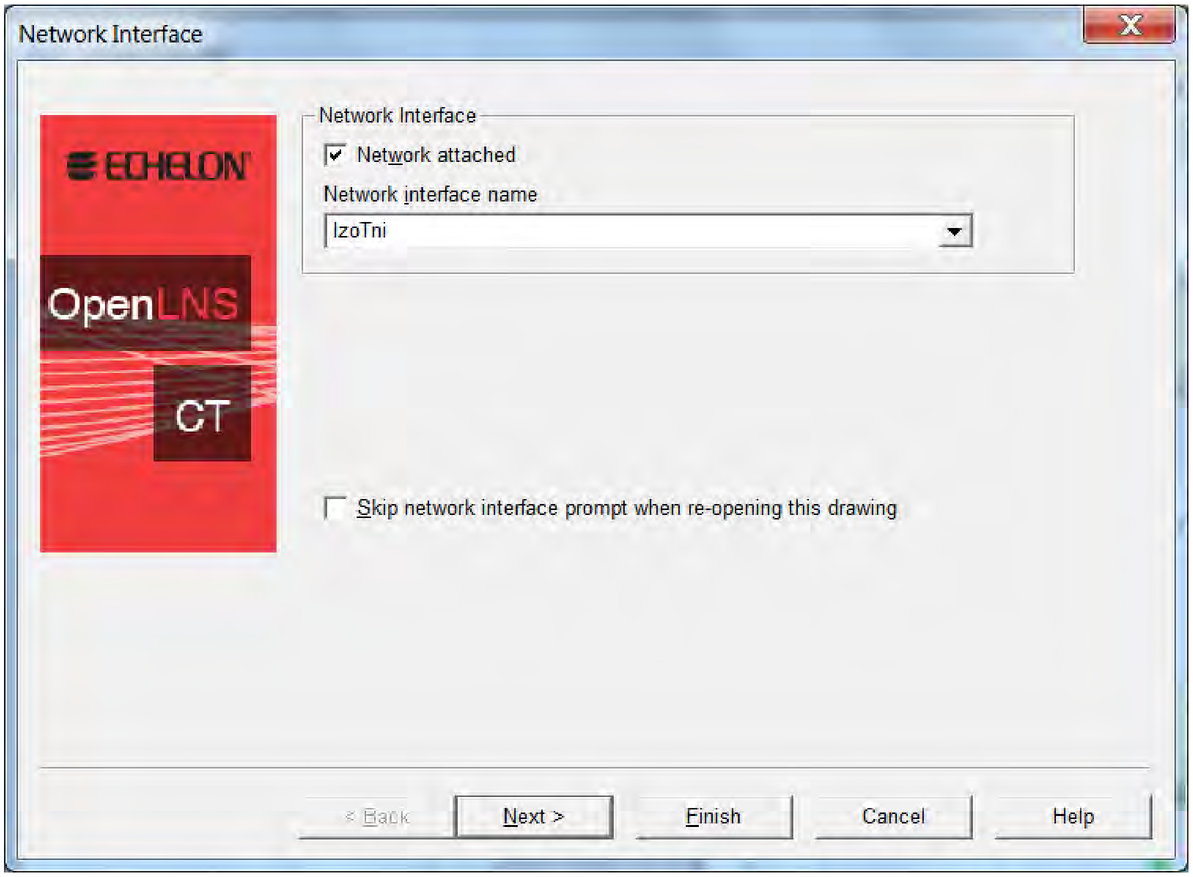
Network Interface

☐ Network attached

Network interface name

☐ Skip network interface prompt when re-opening this drawing

14. 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**.



Network Interface

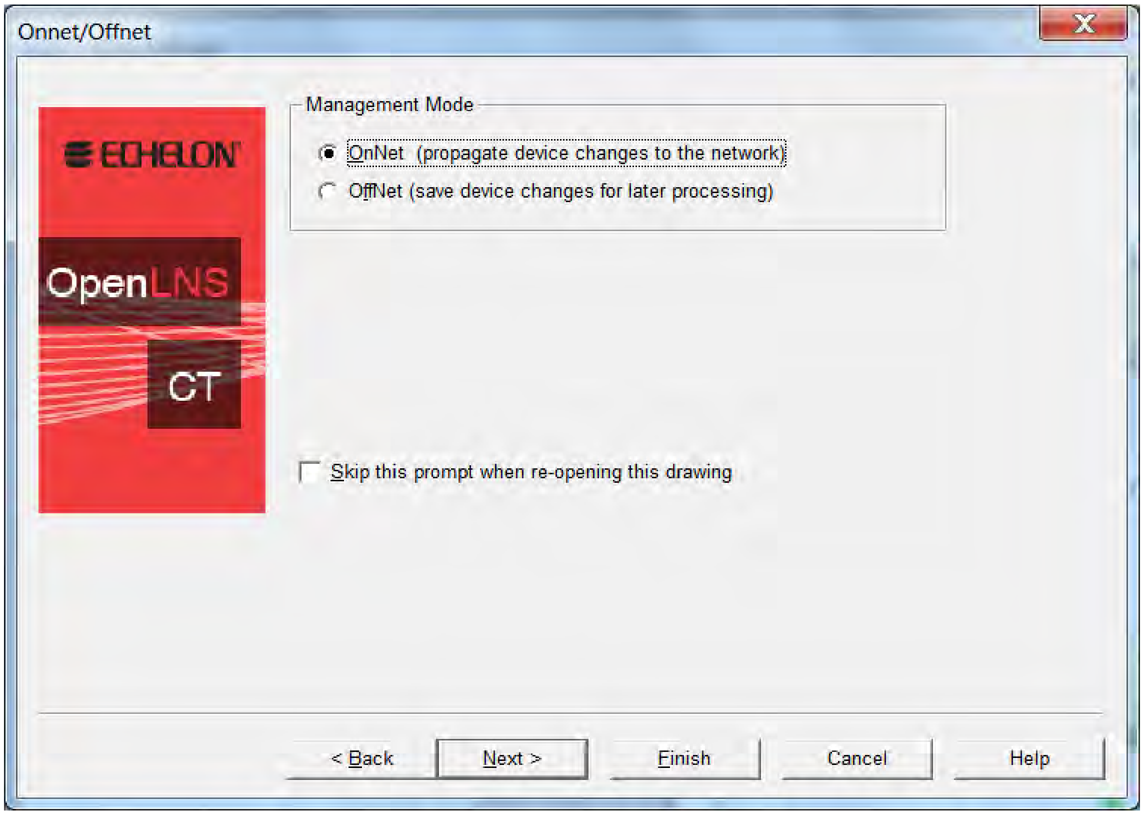
☑ Network attached

Network interface name

IzoTni

☐ Skip network interface prompt when re-opening this drawing

15. The Management Mode window opens.


Onnet/Offnet

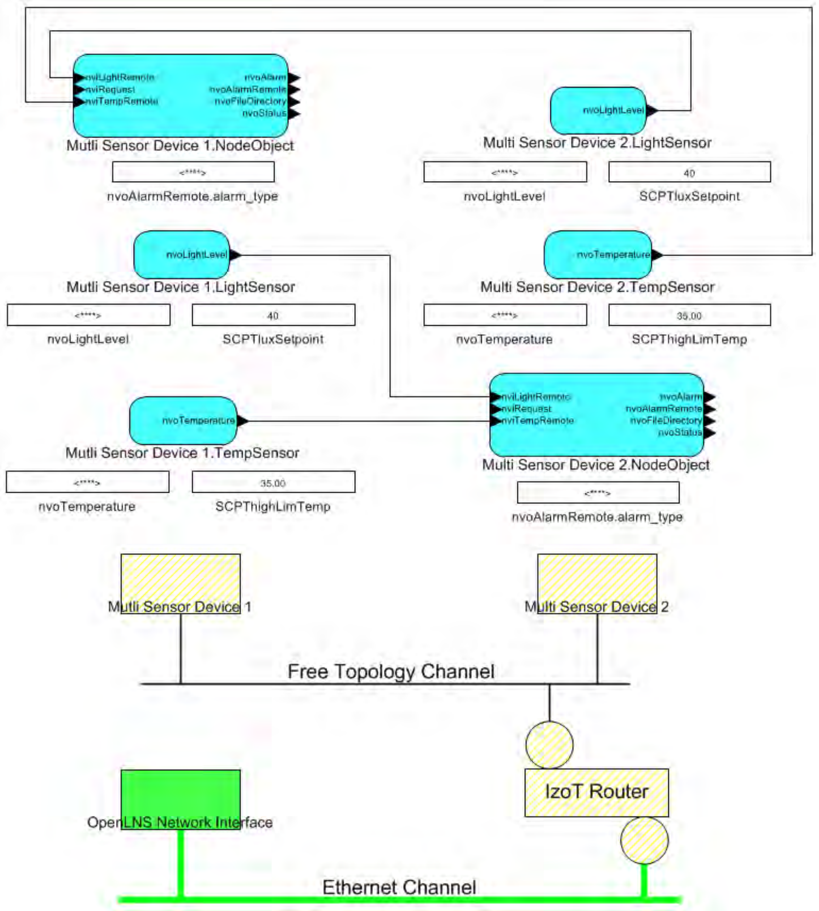
Management Mode

[✓ OnNet (propagate device changes to the network)]

OffNet (save device changes for later processing)

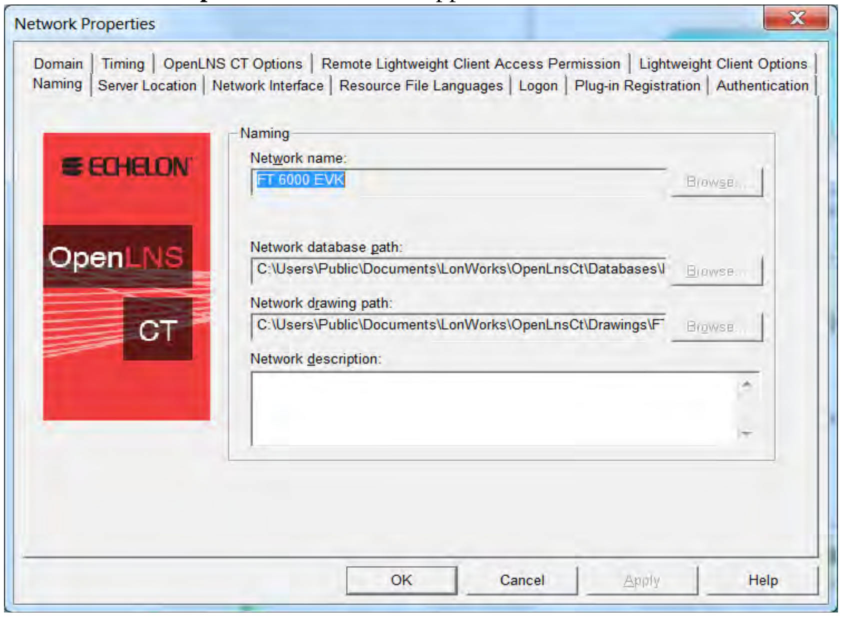
Skip this prompt when re-opening this drawing

16. Select **OnNet** to immediately propagate changes you make to the example device in the IzoT Commissioning Tool drawing to the physical device on the network. Click **Finish**.
17. A message appears recommending that you recommission devices that have changed since the network was backed up. Click **No**.
18. The IzoT CT drawing for the example application opens. The IzoT CT 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.



19. In order to Commission the IzoT Router you must:

- Go to the **Add-Ins** tab of the program where you select **LonWorks Network**, under which you will choose **Network Properties**.
- The **Network Properties** window will appear:


Network Properties
Domain | Timing | OpenLNS CT Options | Remote Lightweight Client Access Permission | Lightweight Client Options
Naming | Server Location | Network Interface | Resource File Languages | Logon | Plug-in Registration | Authentication
ECHELON
OpenLNS
CT
Browse...
OK
Cancel
Apply
Help

Naming

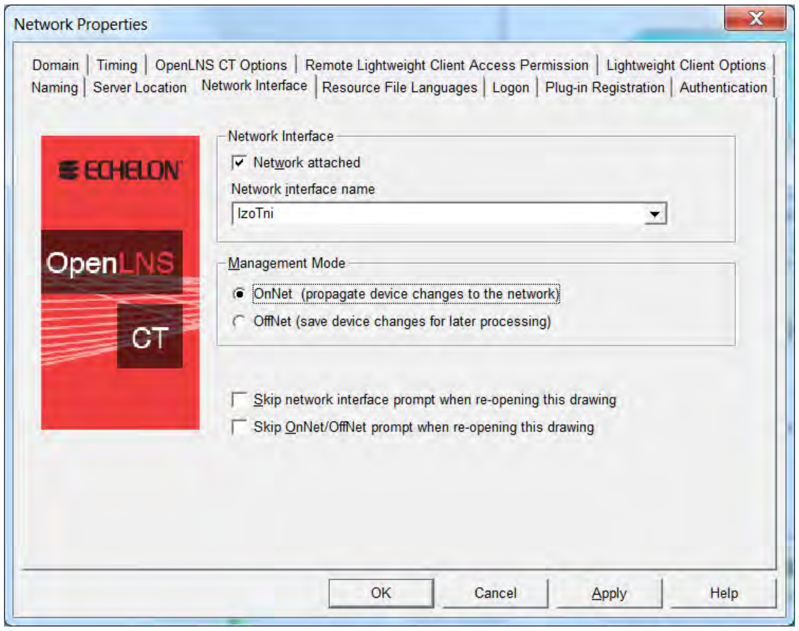
Network name: FT 6000 EVK

Network database path: C:\Users\Public\Documents\LonWorks\OpenLnsCt\Databases\

Network drawing path: C:\Users\Public\Documents\LonWorks\OpenLnsCt\Drawings\F

Network description:

- Select the **Network Interface** tab, click **OnNet :**


Network Properties

Domain | Timing | OpenLNS CT Options | Remote Lightweight Client Access Permission | Lightweight Client Options
Naming | Server Location | Network Interface | Resource File Languages | Logon | Plug-in Registration | Authentication

Network Interface
- [✓] Network attached

Network interface name: IzoTni

Management Mode
- [✓ OnNet (propagate device changes to the network)]
- OffNet (save device changes for later processing)

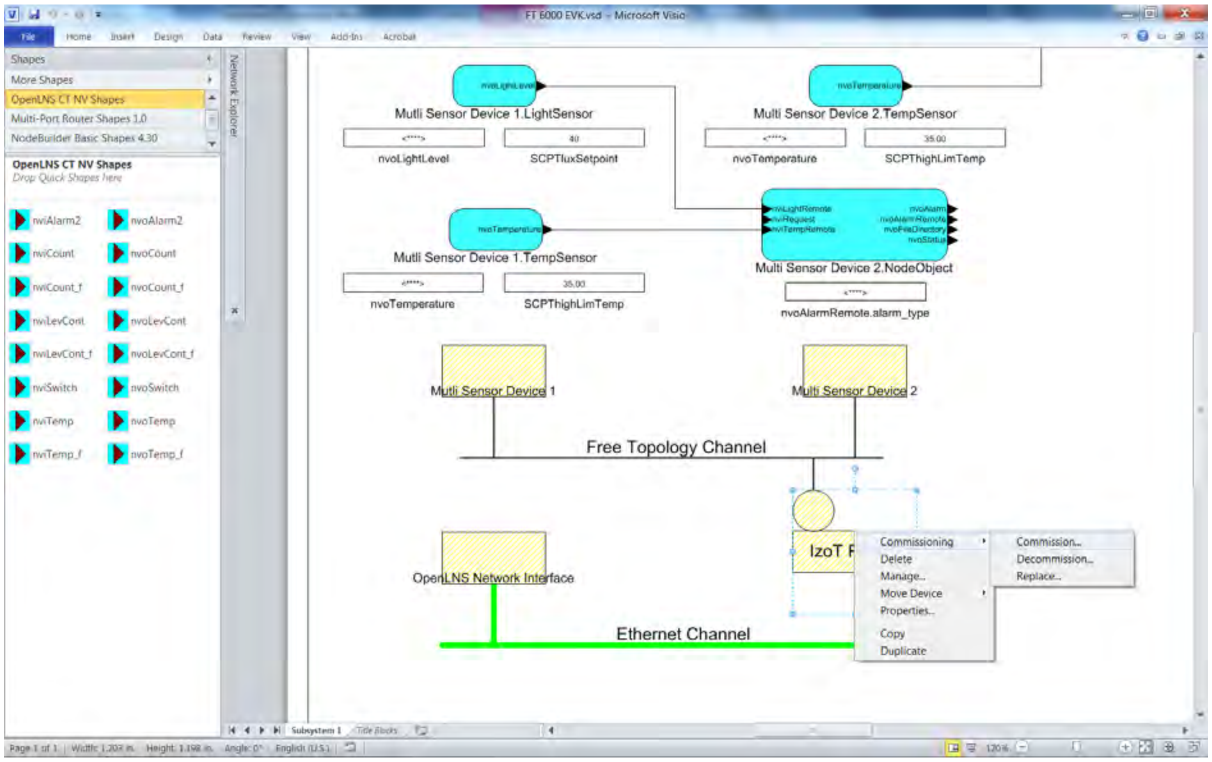
- [ ] Skip network interface prompt when re-opening this drawing
- [ ] Skip OnNet/OffNet prompt when re-opening this drawing

OK | Cancel | Apply | Help

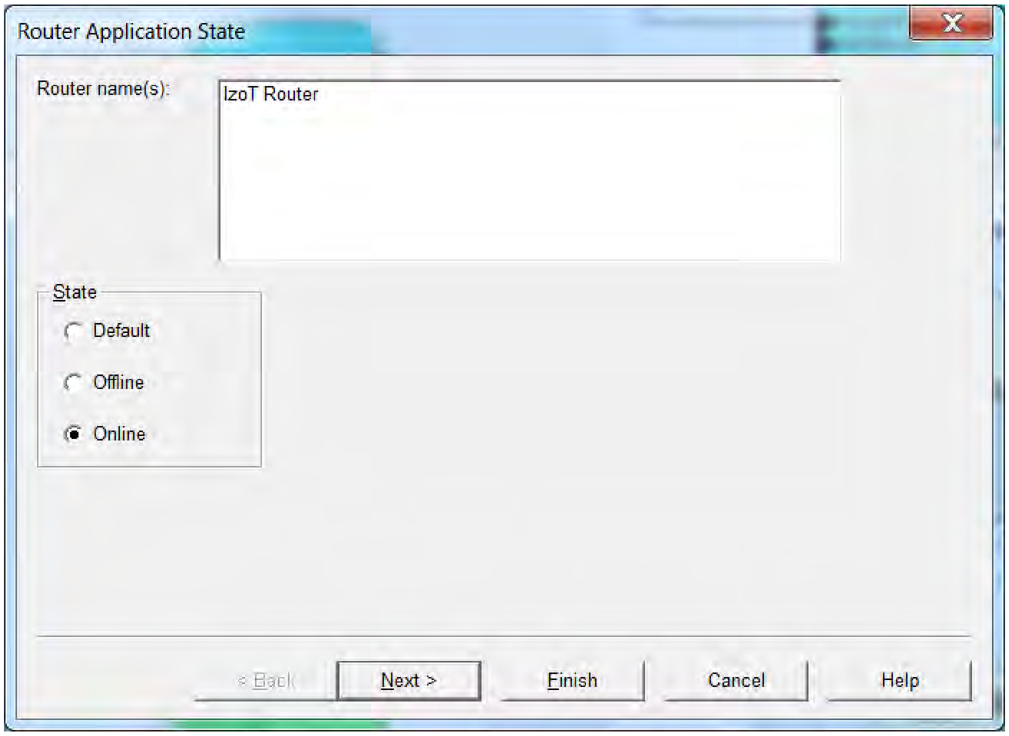
- Click **OK**

20. 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.

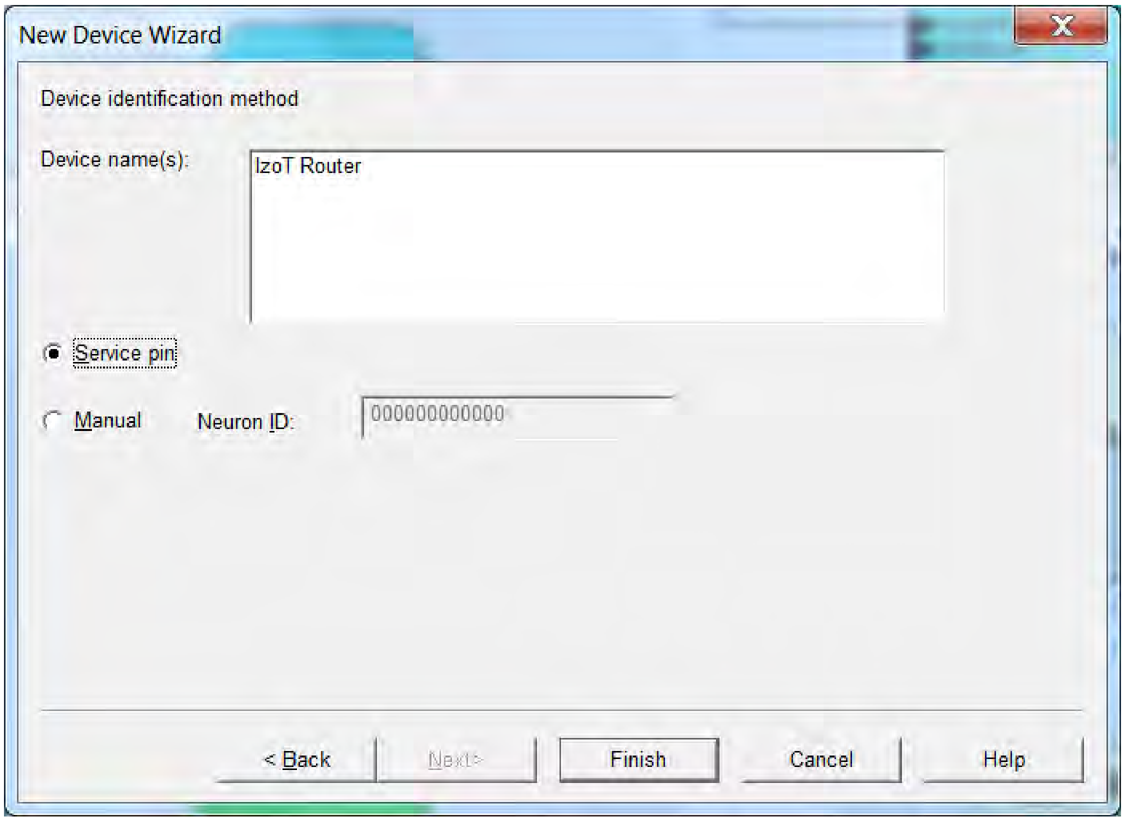


b. After Commissioning the device the following window appears; select the **Online** state, click **Next:**



Router name(s): IzoT Router

State
Default [✓ Online] Offline



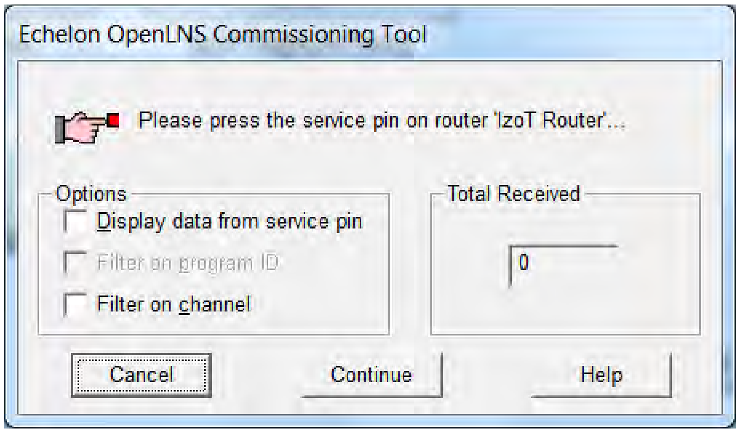
Device identification method

Device name(s): IzoT Router

[✓ Service pin] Manual

Neuron ID: 000000000000

c. Click **Finish**

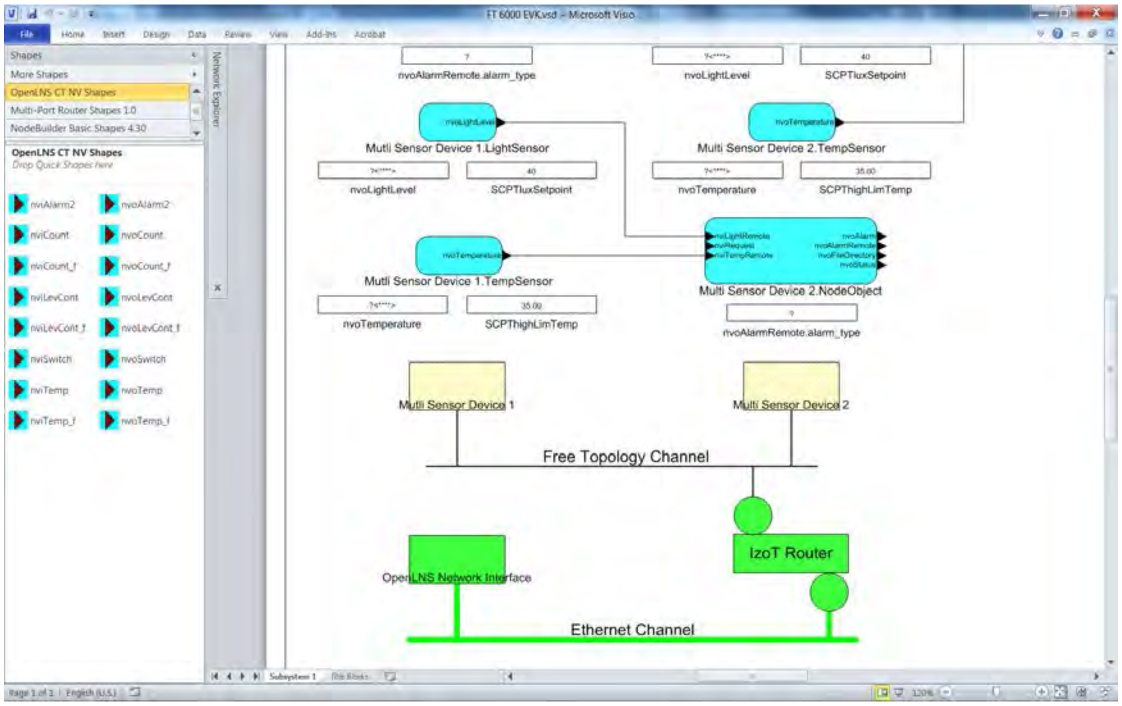

Echelon OpenLNS Commissioning Tool

Please press the service pin on router 'IzoT Router'...

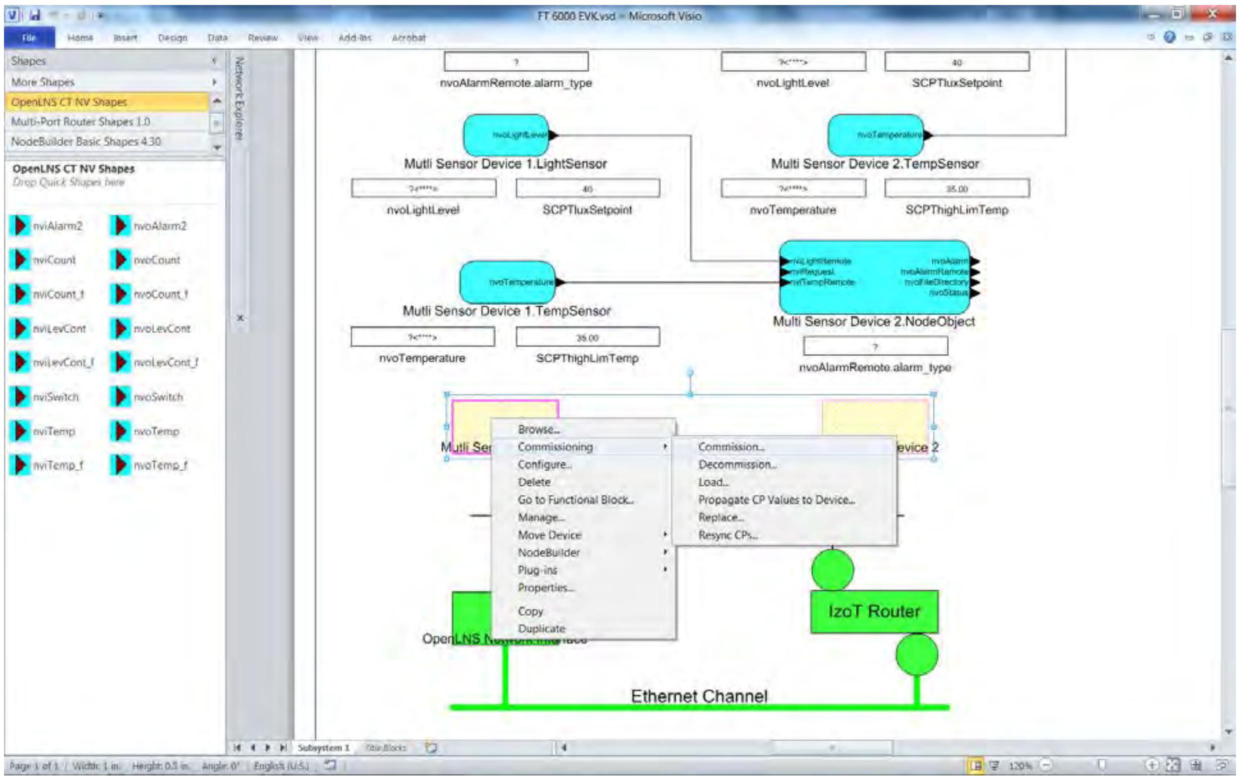
Options
- [ ] Display data from service pin
- [ ] Filter on program ID
- [ ] Filter on channel

Total Received: 0

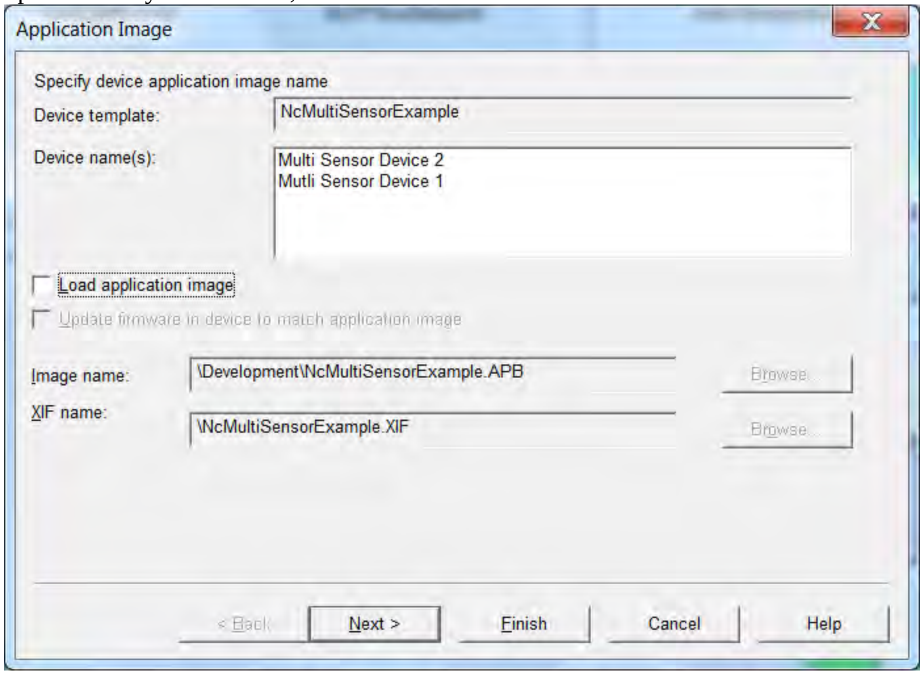
d. The IzoT Router should now be commissioned:


FT 6000 EVK.vsd - Microsoft Visio
OpenLNS CT NV Shapes
Mutli Sensor Device 1.LightSensor
Multi Sensor Device 2.TempSensor
Mutli Sensor Device 1.TempSensor
Multi Sensor Device 2.NodeObject
nvoLightLevel
SCPTluxSetpoint
nvoTemperature
SCPThighLimTemp
nvoAlarmRemote.alarm_type
Mutli Sensor Device 1
Multi Sensor Device 2
Free Topology Channel
IzoT Router
OpenLNS Network Interface
Ethernet Channel

e. To commission MultiSensor Device 1 and MultiSensor Device 2, ctrl-click the two device shapes, right-click one of the selected shapes, point to **Commissioning**, and then click **Commission** on the shortcut menu


FT 6000 EVK.vsd - Microsoft Visio
OpenLNS CT NV Shapes
Multi Sensor Device 1.LightSensor
Multi Sensor Device 2.TempSensor
Mutli Sensor Device 1.TempSensor
Multi Sensor Device 2.NodeObject
Browse...
Commissioning
Configure...
Delete
Go to Functional Block...
Manage...
Move Device
NodeBuilder
Plug-ins
Properties...
Copy
Duplicate
Commission...
Decommission...
Load...
Propagate CP Values to Device...
Replace...
Resync CPs...
IzoT Router
OpenLNS Network Interface
Ethernet Channel

f. The Commission Device Wizard opens with the Application Image window displayed. In the Application Image window, clear the **Load Application Image** and **Update Firmware** options if they are selected, and then click **Next**.


Application Image

Specify device application image name

Device template: NcMultiSensorExample

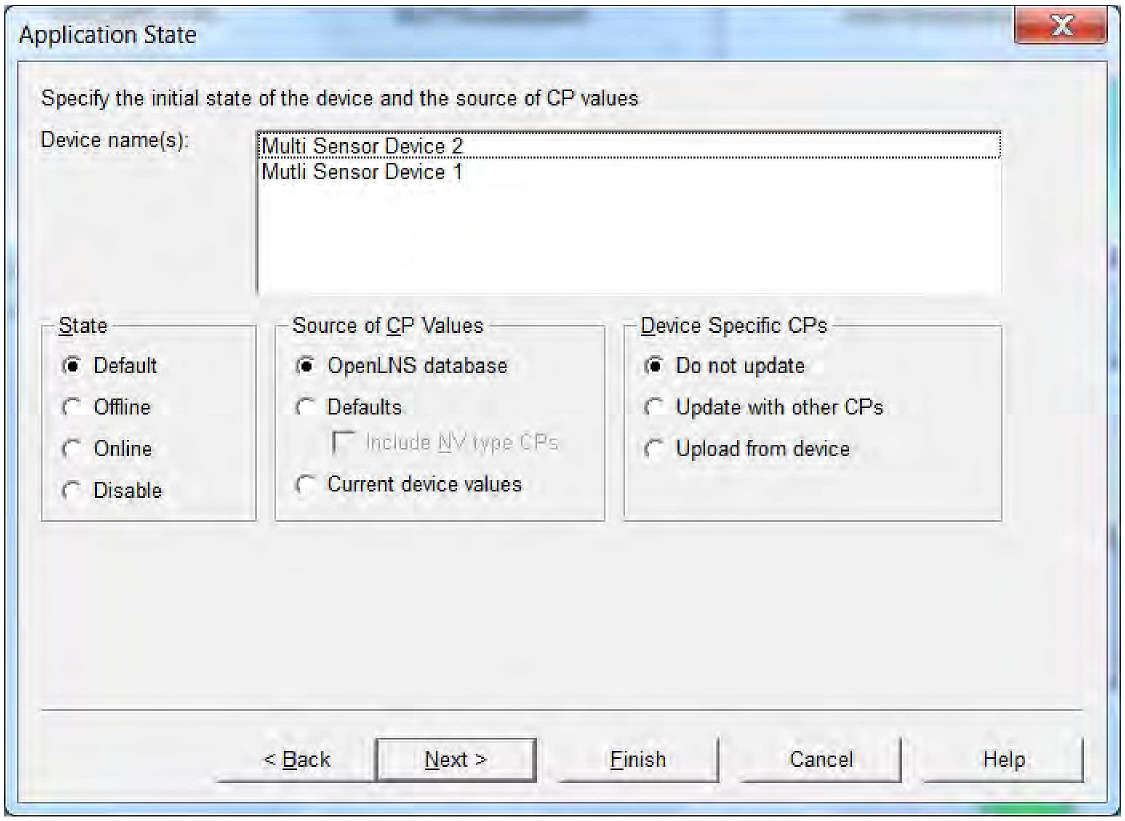
Device name(s): Multi Sensor Device 2
Mutli Sensor Device 1

☐ Load application image

☐ Update firmware in device to match application image

Image name: \Development\NcMultiSensorExample.APB

XIF name: \NcMultiSensorExample.XIF


Application State

Specify the initial state of the device and the source of CP values

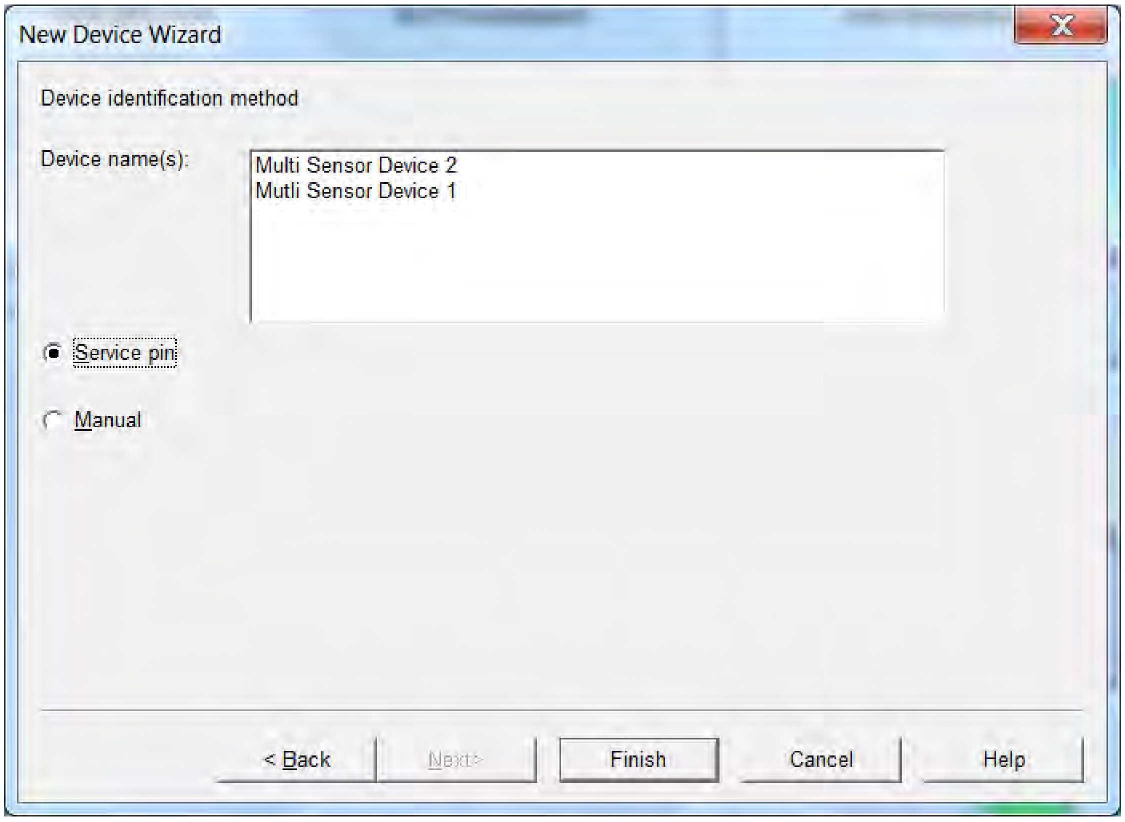
Device name(s):
Multi Sensor Device 2
Mutli Sensor Device 1

State
(● Default) Offline / Online / Disable

Source of CP Values
(● OpenLNS database) Defaults / Include NV type CPs / Current device values

Device Specific CPs
(● Do not update) Update with other CPs / Upload from device

g. Click **Next**

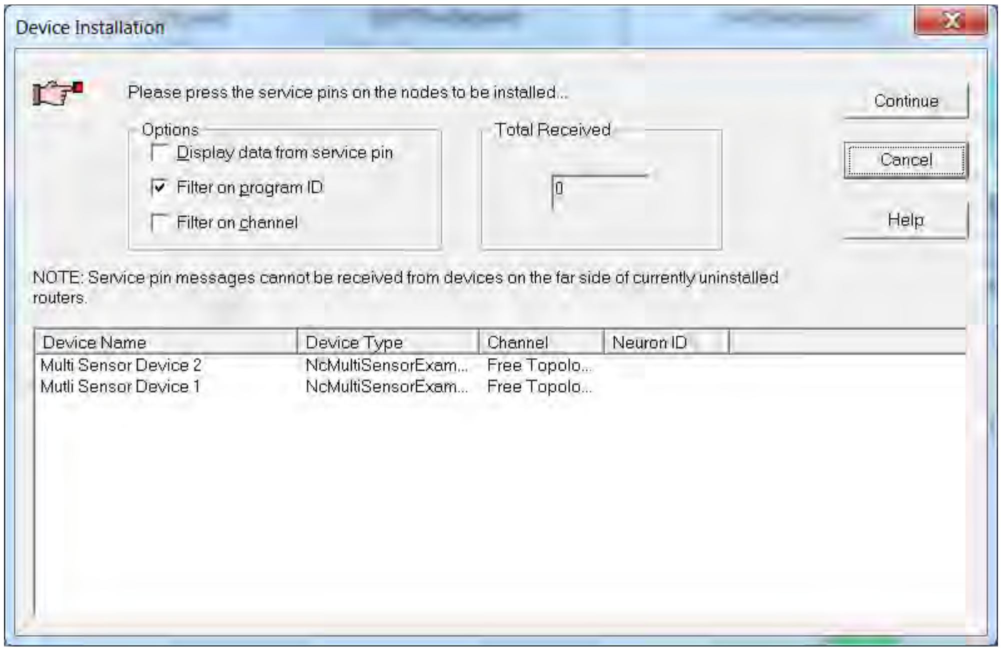

New Device Wizard

Device identification method

Device name(s):
Multi Sensor Device 2
Mutli Sensor Device 1

(● Service pin) Manual

h. Click **Finish**. The **Device Installation** dialog opens.


Device Installation

Please press the service pins on the nodes to be installed...

Options
- [ ] Display data from service pin
- [✓] Filter on program ID
- [ ] Filter on channel

Total Received: 0

NOTE: Service pin messages cannot be received from devices on the far side of currently uninstalled routers.

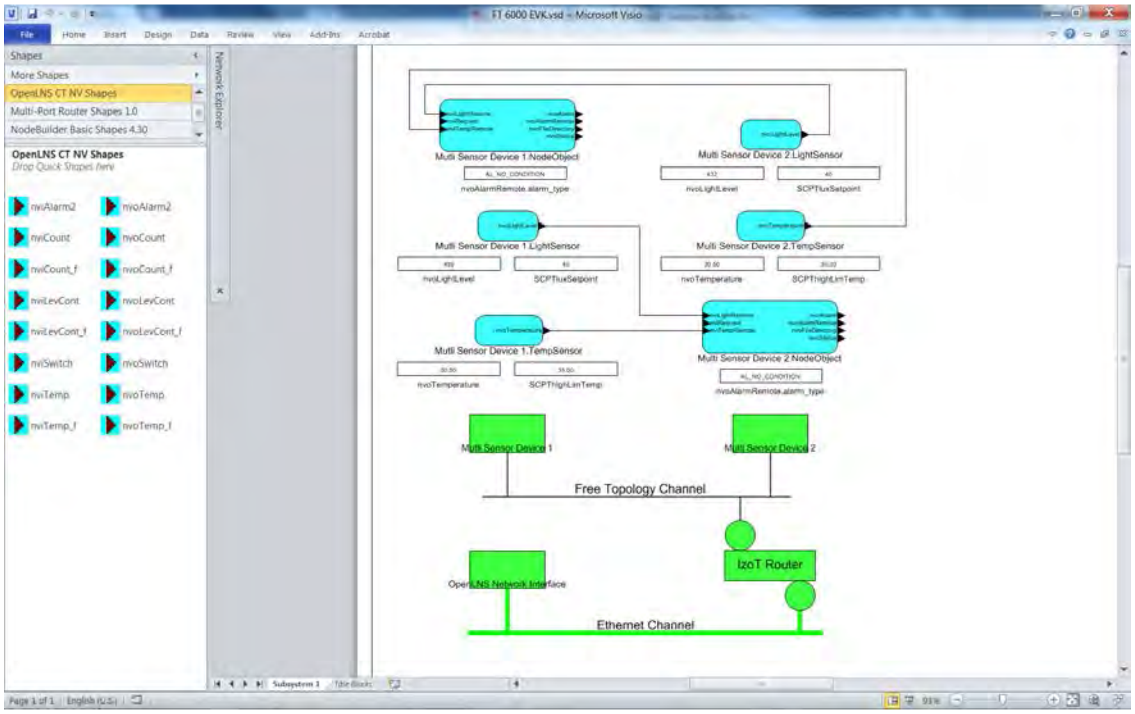
Device Name	Device Type	Channel	Neuron ID
Multi Sensor Device 2	NcMultiSensorExam...	Free Topolo...	
Mutli Sensor Device 1	NcMultiSensorExam...	Free Topolo...	

i. 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**."

j. The **Device Commission** dialog opens. This dialog lists application image loading, commissioning, and device state events in the order they occur.

k. After both example devices have been commissioned, click **OK** to return to the IzoT CT drawing.

l. 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.

21. Test the I/O devices on the FT 6000 EVBs using the sample network variable connections provided in the IzoT Commissioning Tool 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 IzoT CT Browser 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 Izot CT Browser and the Data Point shape in the IzoT Commissioning tool, see Chapter 6 of the *IzoT Commissioning Tool User's Guide*.

22. 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**."

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 IzoT CT 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 IzoT CT 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.*

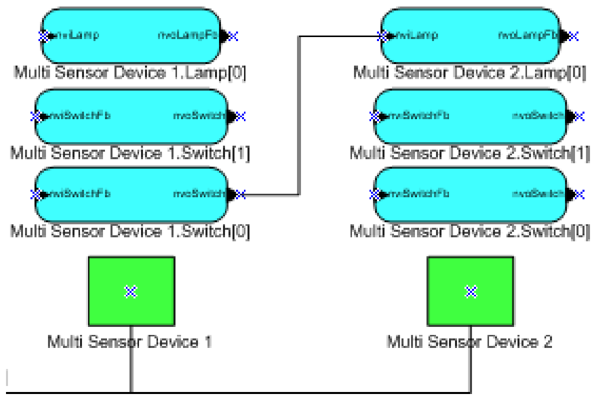
Testing Switch and Lamp Devices

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.

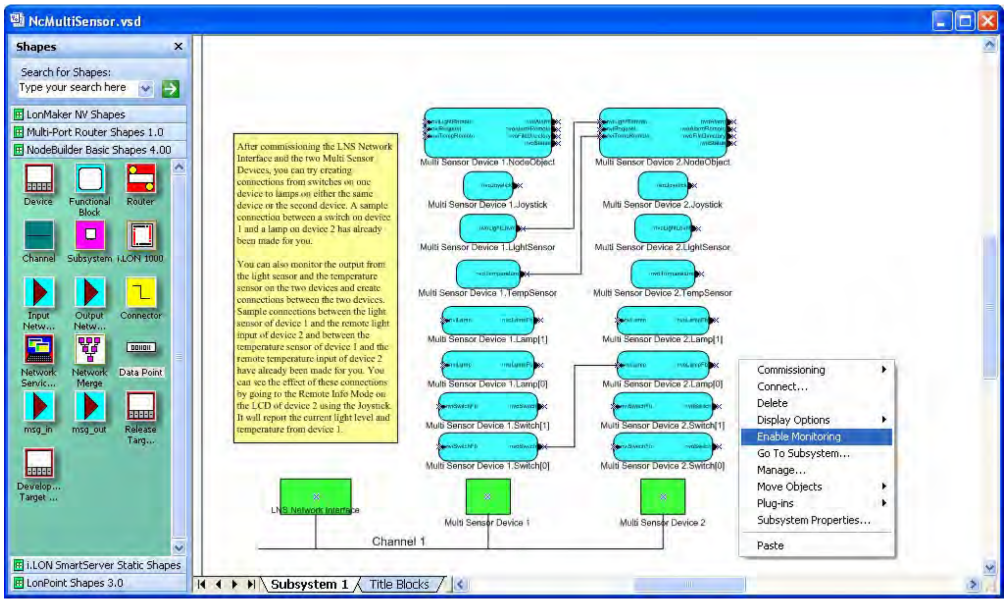
2. Press the **SW1** button on the FT 6000 EVB corresponding to **Multi Sensor Device 1** to turn on **LED1** on the FT 6000 EVB corresponding to **Multi Sensor Device 2**, and then press the **SW1** button again to turn off **LED1**.

3. Click the Echelon OpenLNS Commissioning Tool/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.


nviLamp
nvoLampFb
Multi Sensor Device 1.Lamp[0]
nviSwitchFb
nvoSwitch
Multi Sensor Device 1.Switch[1]
nviSwitchFb
nvoSwitch
Multi Sensor Device 1.Switch[0]
Multi Sensor Device 1
nviLamp
nvoLampFb
Multi Sensor Device 2.Lamp[0]
nviSwitchFb
nvoSwitch
Multi Sensor Device 2.Switch[1]
nviSwitchFb
nvoSwitch
Multi Sensor Device 2.Switch[0]
Multi Sensor Device 2

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:

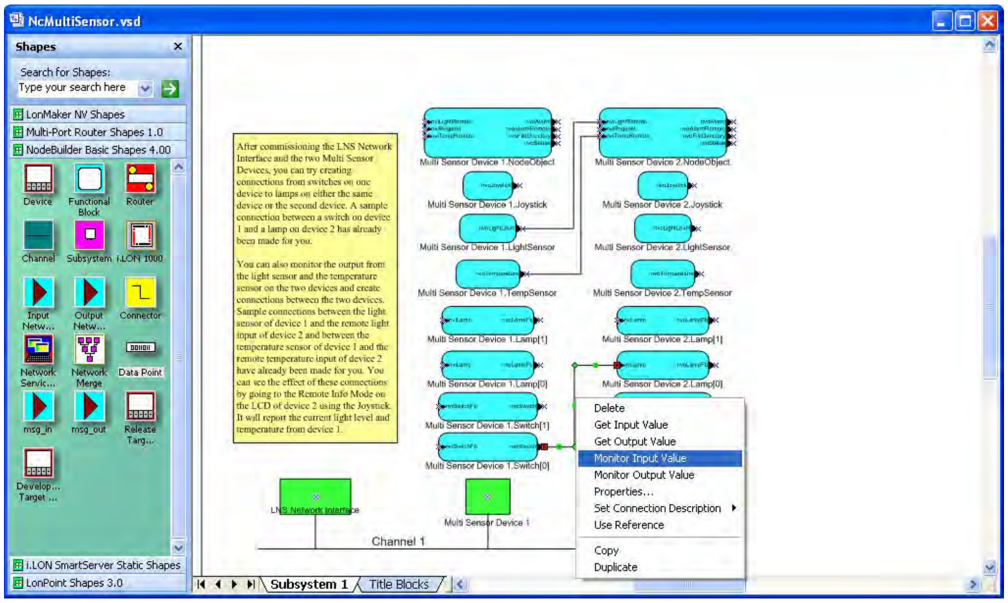
   a. Right-click an empty space in the IzoT Commissioning Tool drawing, and then select **Enable Monitoring** on the shortcut menu.


NcMultiSensor.vsd
Shapes
Search for Shapes:
Type your search here
LonMaker NV Shapes
Multi-Port Router Shapes 1.0
NodeBuilder Basic Shapes 4.00
Device
Functional Block
Router
Channel
Subsystem
i.LON 1000
Input Netw...
Output Netw...
Connector
Network Servic...
Network Merge
Data Point
msg_in
msg_out
Release Targ...
Develop... Target ...
i.LON SmartServer Static Shapes
LonPoint Shapes 3.0
Multi Sensor Device 1.NodeObject
Multi Sensor Device 2.NodeObject
Multi Sensor Device 1.Joystick
Multi Sensor Device 2.Joystick
Multi Sensor Device 1.LightSensor
Multi Sensor Device 2.LightSensor
Multi Sensor Device 1.TempSensor
Multi Sensor Device 2.TempSensor
Multi Sensor Device 1.Lamp[1]
Multi Sensor Device 2.Lamp[1]
Multi Sensor Device 1.Lamp[0]
Multi Sensor Device 2.Lamp[0]
Multi Sensor Device 1.Switch[1]
Multi Sensor Device 2.Switch[1]
Multi Sensor Device 1.Switch[0]
Multi Sensor Device 2.Switch[0]
LNS Network Interface
Multi Sensor Device 1
Multi Sensor Device 2
Channel 1
Commissioning
Connect...
Delete
Display Options
Enable Monitoring
Go To Subsystem...
Manage...
Move Objects
Plug-ins
Subsystem Properties...
Paste
Subsystem 1
Title Blocks

After commissioning the LNS Network Interface and the two Multi Sensor Devices, you can try creating connections from switches on one device to lamps on either the same device or the second device. A sample connection between a switch on device 1 and a lamp on device 2 has already been made for you.

You can also monitor the output from the light sensor and the temperature sensor on the two devices and create connections between the two devices. Sample connections between the light sensor of device 1 and the remote light input of device 2 and between the temperature sensor of device 1 and the remote temperature input of device 2 have already been made for you. You can see the effect of these connections by going to the Remote Info Mode on the LCD of device 2 using the Joystick. It will report the current light level and temperature from device 1.

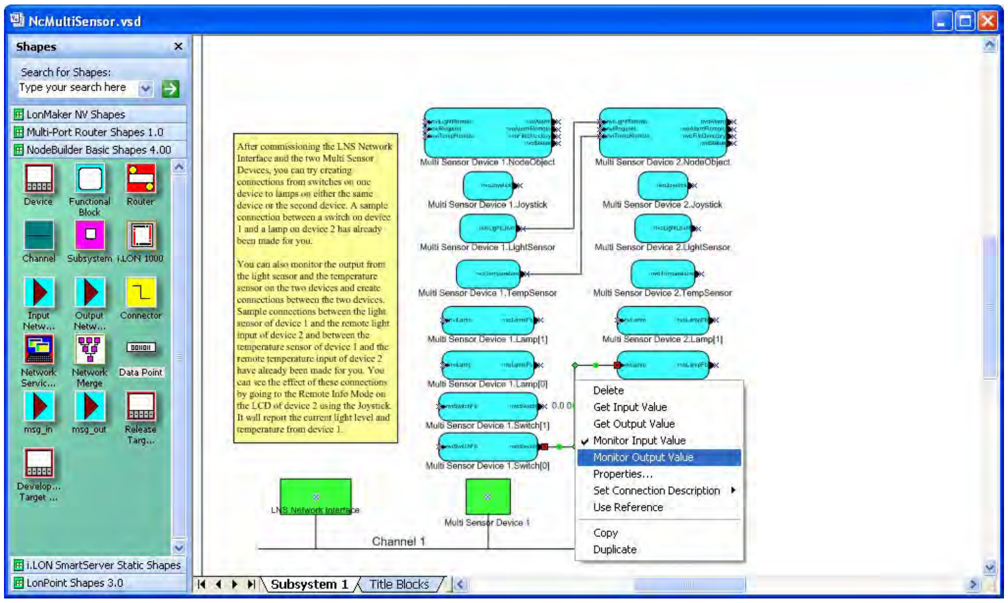
b. Right-click the sample **Connector** shape and select **Monitor Input Value** to display the current value of the **Multi Sensor Device 1.Switch[0].nvoSwitch** output network variable.


NcMultiSensor.vsd
Shapes
Search for Shapes:
Type your search here
LonMaker NV Shapes
Multi-Port Router Shapes 1.0
NodeBuilder Basic Shapes 4.00
Device
Functional Block
Router
Channel
Subsystem
i.LON 1000
Input Netw...
Output Netw...
Connector
Network Servic...
Network Merge
Data Point
msg_in
msg_out
Release Targ...
Develop... Target ...
i.LON SmartServer Static Shapes
LonPoint Shapes 3.0
Multi Sensor Device 1.NodeObject
Multi Sensor Device 2.NodeObject
Multi Sensor Device 1.Joystick
Multi Sensor Device 2.Joystick
Multi Sensor Device 1.LightSensor
Multi Sensor Device 2.LightSensor
Multi Sensor Device 1.TempSensor
Multi Sensor Device 2.TempSensor
Multi Sensor Device 1.Lamp[1]
Multi Sensor Device 2.Lamp[1]
Multi Sensor Device 1.Lamp[0]
Multi Sensor Device 2.Lamp[0]
Multi Sensor Device 1.Switch[1]
Multi Sensor Device 1.Switch[0]
LNS Network Interface
Multi Sensor Device 1
Channel 1
Delete
Get Input Value
Get Output Value
Monitor Input Value
Monitor Output Value
Properties...
Set Connection Description
Use Reference
Copy
Duplicate
Subsystem 1
Title Blocks

After commissioning the LNS Network Interface and the two Multi Sensor Devices, you can try creating connections from switches on one device to lamps on either the same device or the second device. A sample connection between a switch on device 1 and a lamp on device 2 has already been made for you.

You can also monitor the output from the light sensor and the temperature sensor on the two devices and create connections between the two devices. Sample connections between the light sensor of device 1 and the remote light input of device 2 and between the temperature sensor of device 1 and the remote temperature input of device 2 have already been made for you. You can see the effect of these connections by going to the Remote Info Mode on the LCD of device 2 using the Joystick. It will report the current light level and temperature from device 1.

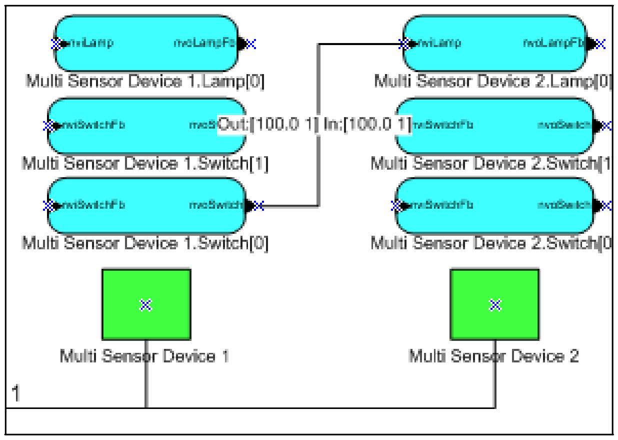
c. Right-click the **Connector** shape and select **Monitor Output Value** to display the current value of the **Multi Sensor Device 2.Lamp[0].nviLamp** input network variable.


After commissioning the LNS Network Interface and the two Multi Sensor Devices, you can try creating connections from switches on one device to lamps on either the same device or the second device. A sample connection between a switch on device 1 and a lamp on device 2 has already been made for you.

You can also monitor the output from the light sensor and the temperature sensor on the two devices and create connections between the two devices. Sample connections between the light sensor of device 1 and the remote light input of device 2 and between the temperature sensor of device 1 and the remote temperature input of device 2 have already been made for you. You can see the effect of these connections by going to the Remote Info Mode on the LCD of device 2 using the Joystick. It wall report the current light level and temperature from device 1.

NcMultiSensor.vsd
Shapes
Search for Shapes:
Type your search here
LonMaker NV Shapes
Multi-Port Router Shapes 1.0
NodeBuilder Basic Shapes 4.00
Device
Functional Block
Router
Channel
Subsystem
i.LON 1000
Input Netw...
Output Netw...
Connector
Network Servic...
Network Merge
Data Point
msg_in
msg_out
Release Targ...
Develop... Target ...
i.LON SmartServer Static Shapes
LonPoint Shapes 3.0
Multi Sensor Device 1.NodeObject
Multi Sensor Device 2.NodeObject
Multi Sensor Device 1.Joystick
Multi Sensor Device 2.Joystick
Multi Sensor Device 1.LightSensor
Multi Sensor Device 2.LightSensor
Multi Sensor Device 1.TempSensor
Multi Sensor Device 2.TempSensor
Multi Sensor Device 1.Lamp[1]
Multi Sensor Device 2.Lamp[1]
Multi Sensor Device 1.Lamp[0]
Multi Sensor Device 1.Switch[1]
Multi Sensor Device 1.Switch[0]
0.0 0
LNS Network Interface
Multi Sensor Device 1
Channel 1
Delete
Get Input Value
Get Output Value
Monitor Input Value
Monitor Output Value
Properties...
Set Connection Description
Use Reference
Copy
Duplicate
Subsystem 1
Title Blocks

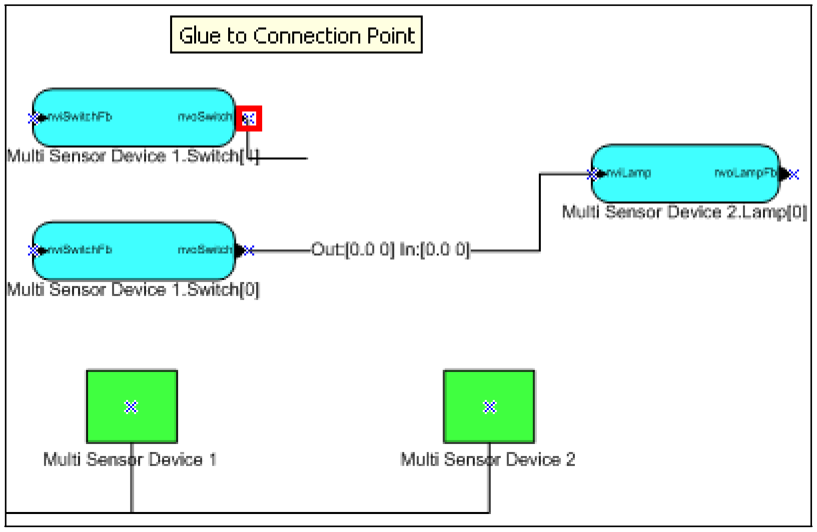
d. Press the **SW1** button on **Multi Sensor Device 1** multiple times. Observe that **LED1** on **Multi Sensor Device 2** turns on and off each time you press the **SW1** button. In addition, the current values of the output and input network variable on the **Connector** shape toggle between 100.0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


nviLamp
nvoLampFb
Multi Sensor Device 1.Lamp[0]
nviLamp
nvoLampFb
Multi Sensor Device 2.Lamp[0]
nviSwitchFb
nvoS
Out:[100.0 1]
In:[100.0 1]
nviSwitchFb
nvoSwitch
Multi Sensor Device 1.Switch[1]
Multi Sensor Device 2.Switch[1
nviSwitchFb
nvoSwitch
Multi Sensor Device 1.Switch[0]
nviSwitchFb
nvoSwitch
Multi Sensor Device 2.Switch[0
Multi Sensor Device 1
Multi Sensor Device 2
1

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.

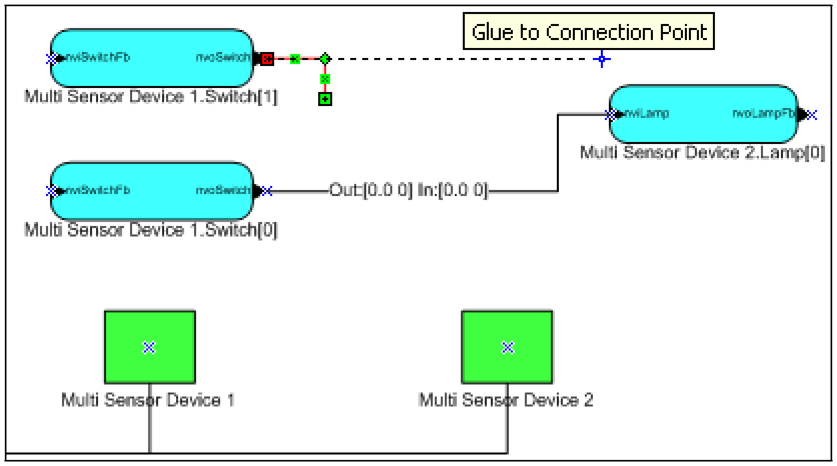
5. If you are running the *NcSimpleIsiExample* or *NcMultiSensorExample*, connect the **Multi Sensor Device 1.Switch[1].nvoSwitch** network variable to the **Multi Sensor Device 2.Lamp[0].nviLamp** network variable. To do this follow these steps:

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.



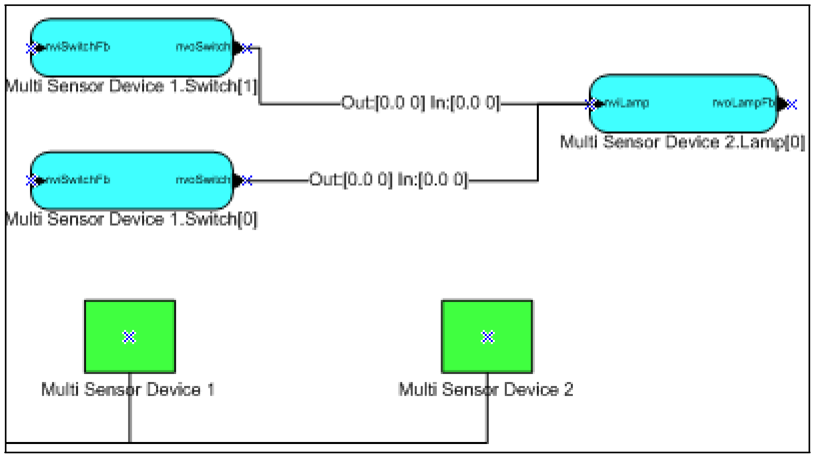
Note: To simplify the IzoT CT drawing in this example, only the subject functional blocks are further displayed.

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.

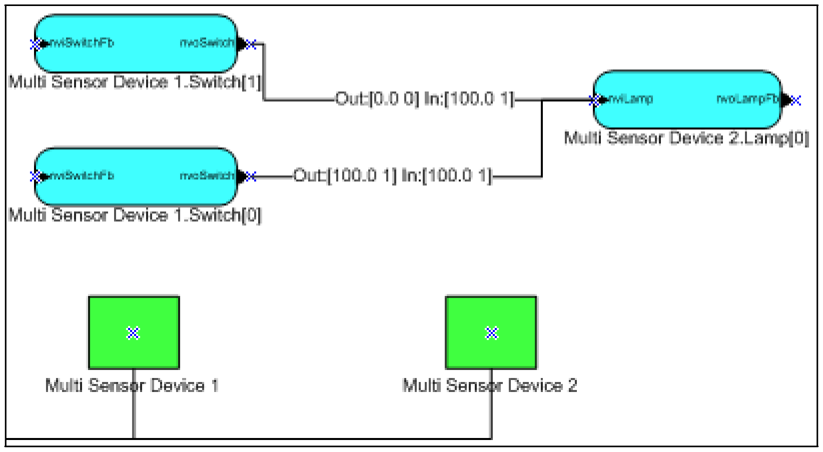


Note: You can also create connections using the **Connector** tool ( ) on the Visio **Standard** toolbar or the **Network Variable Connection** dialog box. See Chapter 4 of the *IzoT Commissioning Tool User's Guide* for more information on creating connection using these methods.

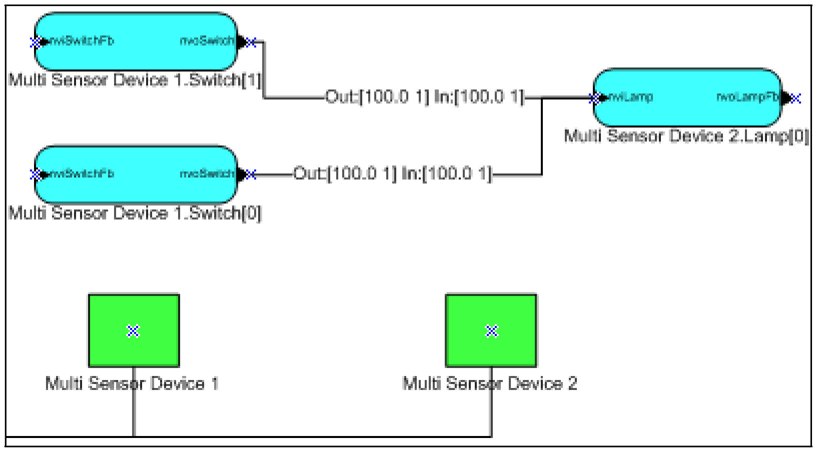
6. Follow step 4 to monitor the values of the **Multi Sensor Device 1.Switch[1].nvoSwitch** and **Multi Sensor Device 2.Lamp[0].nviLamp** network variables on the connection you just created.



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



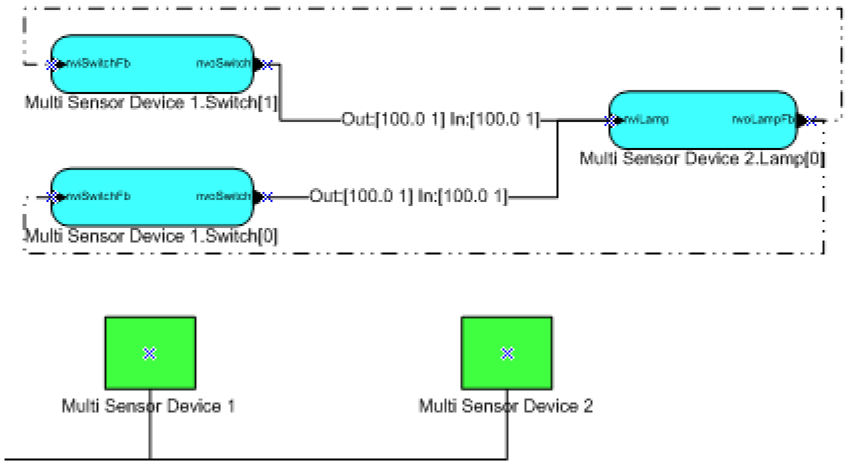
8. Press the **SW2** button on **Multi Sensor Device 1**. Observe that **LED1** on **Multi Sensor Device 2** does not turns off, and observe that the **Multi Sensor Device 2.Lamp[0].nviLamp** network variable is not updated in the IzoT CT drawing. This is because the **Multi Sensor Device 1.Switch[1].nvoSwitch**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


nviSwitchFb
nvoSwitch
Multi Sensor Device 1.Switch[1]
Out:[100.0 1] In:[100.0 1]
nviLamp
nvoLampFb
Multi Sensor Device 2.Lamp[0]
nviSwitchFb
nvoSwitch
Out:[100.0 1] In:[100.0 1]
Multi Sensor Device 1.Switch[0]
Multi Sensor Device 1
Multi Sensor Device 2

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.


nviSwitchFb
nvoSwitch
Multi Sensor Device 1.Switch[1]
Out:[100.0 1] In:[100.0 1]
nviLamp
nvoLampFb
Multi Sensor Device 2.Lamp[0]
nviSwitchFb
nvoSwitch
Out:[100.0 1] In:[100.0 1]
Multi Sensor Device 1.Switch[0]
Multi Sensor Device 1
Multi Sensor Device 2

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.

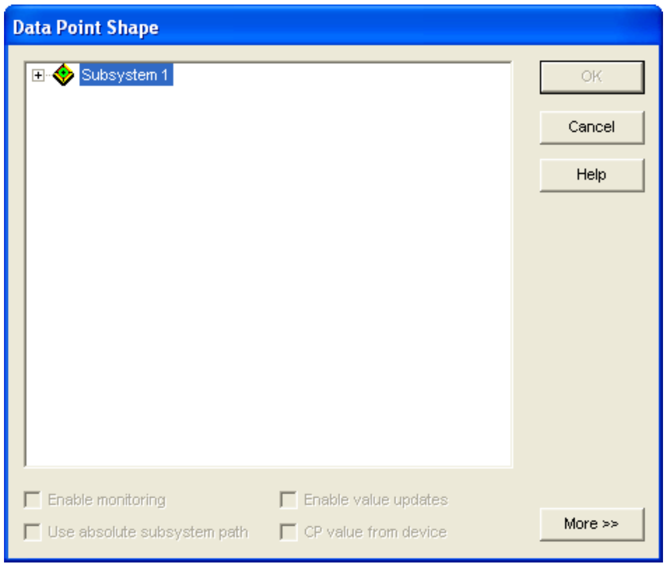
The IzoT CT drawing includes one set of sample network variable connections between the network variables in the **LightSensor** and **TempSensor** functional blocks of **Multi Sensor Device 1** to the network variables in the **Node Object** functional block of **Multi Sensor Device 2**. You can create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

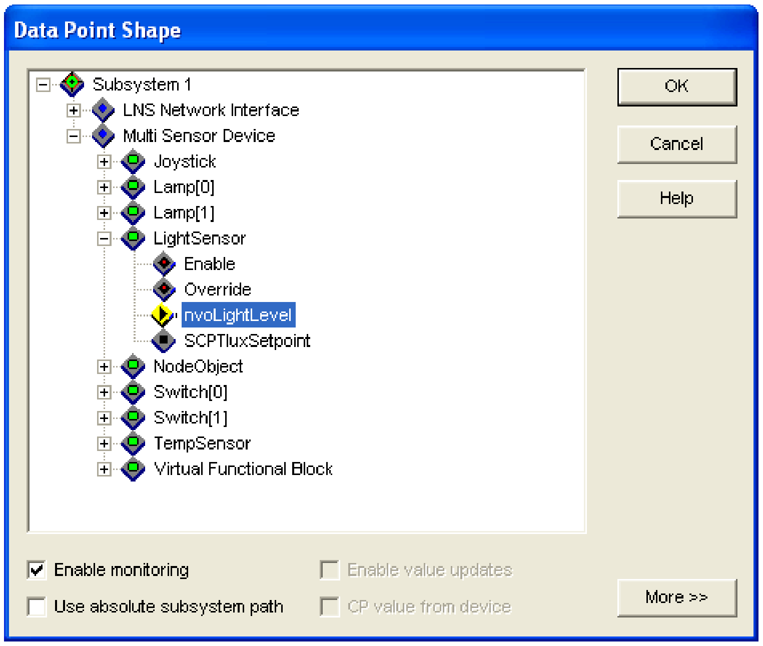
1. Use the LCD on the FT 6000 EVB corresponding to **Multi Sensor Device 1** and the IzoT CT drawing to monitor the light level and temperature of the board, set alarm limits for the light level and temperature, and observe the affects of changing the light level and light alarm limits. To do this, follow 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.



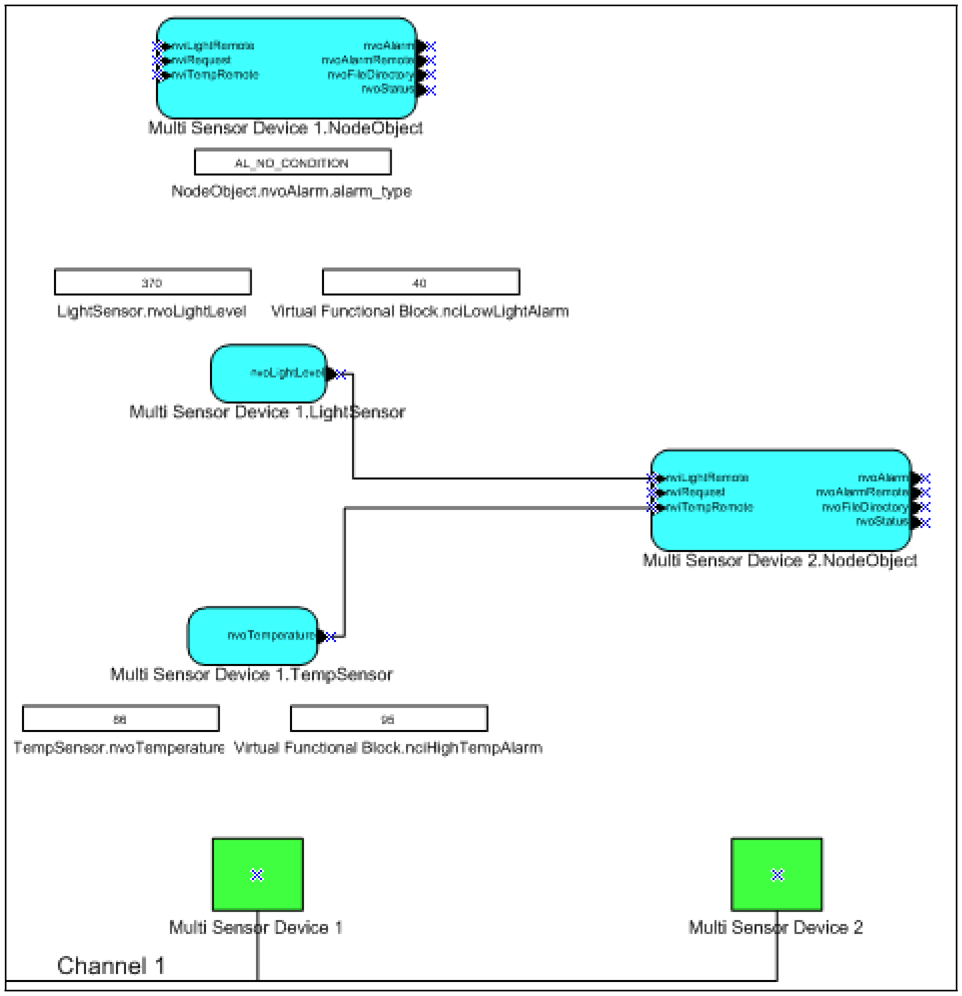
       Expand **Subsystem 1**, expand **Multi Sensor Device 1**, expand the **LightSensor** functional block, and then select the **nvoLightLevel** output network variable; select the **Enable Monitoring** check box; and then click **OK**.



The Data Point shape is added to your IzoT CT drawing.

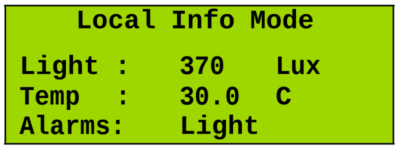
b. Repeat step a for the following data points:

- 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 Update**s 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.


nviLightRemote
nviRequest
nviTempRemote
nvoAlarm
nvoAlarmRemote
nvoFileDirectory
nvoStatus
Multi Sensor Device 1.NodeObject
AL_NO_CONDITION
NodeObject.nvoAlarm.alarm_type
370
LightSensor.nvoLightLevel
40
Virtual Functional Block.nciLowLightAlarm
nvoLightLevel
Multi Sensor Device 1.LightSensor
Multi Sensor Device 2.NodeObject
nvoTemperature
Multi Sensor Device 1.TempSensor
86
TempSensor.nvoTemperature
95
Virtual Functional Block.nciHighTempAlarm
Multi Sensor Device 1
Multi Sensor Device 2
Channel 1

Note: To simplify the IzoT CT 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".



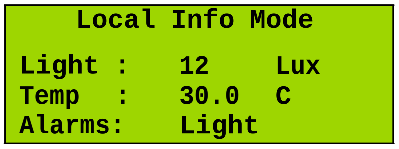
```
    Local Info Mode

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".

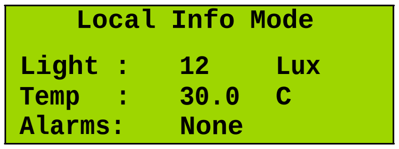


```
    Local Info Mode

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).

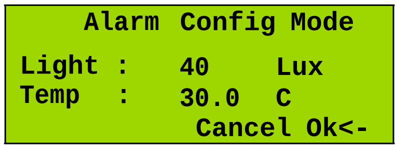
e. In the IzoT CT drawing, double-click the Data Point shape for the **Multi Sensor Device 1.Virtual Functional Block.nciLowLightAlarm** configuration property, enter **0**, and then click anywhere outside the Data Point shape. Cover the light senor I/O device again on the FT 5000 EVB corresponding to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

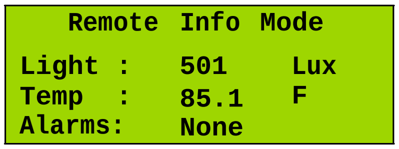
2. Use the sample network variable connections in the IzoT CT drawing to monitor the light level, temperature, and alarm conditions of the FT 6000 EVB corresponding to **Multi Sensor Device 1** from the Remote Info Mode panel on the LCD of the FT 6000 EVB corresponding to **Multi Sensor Device 2**.

   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 IzoT CT 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).

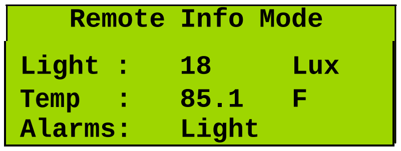


```
    Remote Info Mode

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 senor 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 IzoT CT 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**.

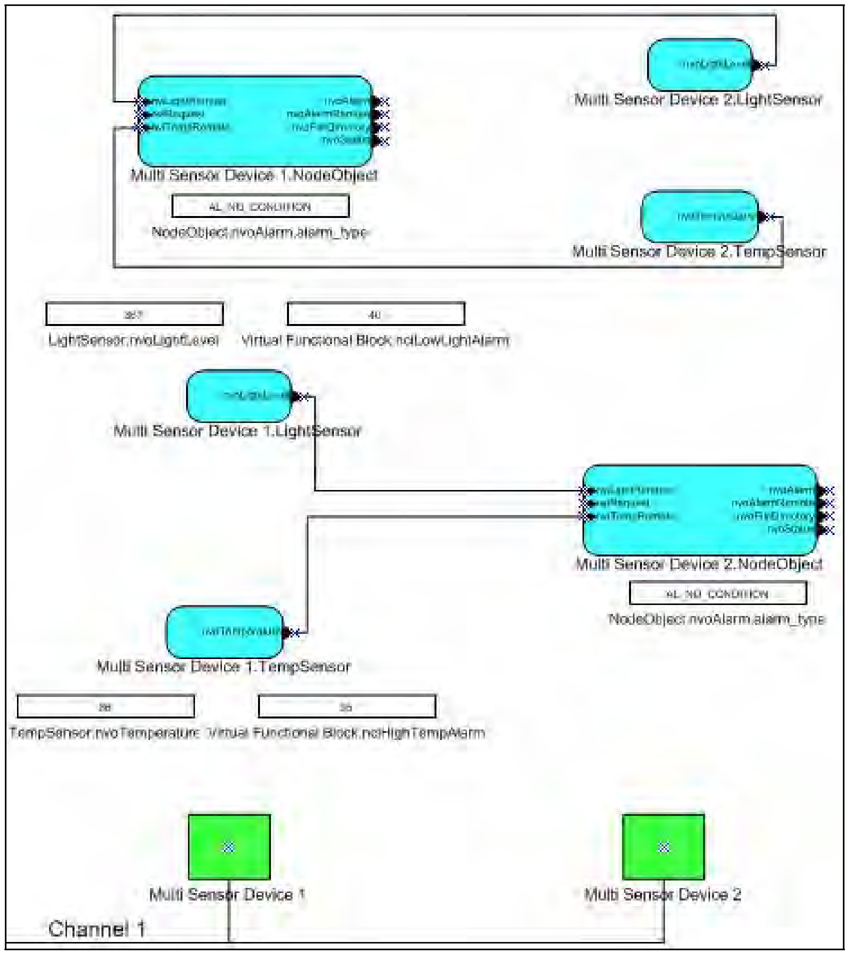
3. Optionally, you can use the IzoT Commissioning tool to create network variable connections between the **LightSensor** and **TempSensor** functional blocks of **Multi Sensor Device 2** to the network variables in the **Node Object** functional block of **Multi Sensor Device 1**. This is simply the 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.


Multi Sensor Device 1.NodeObject
AL_NO_CONDITION
NodeObject.nvoAlarm.alarm_type
Multi Sensor Device 2.LightSensor
Multi Sensor Device 2.TempSensor
LightSensor.nvoLightLevel
Virtual Functional Block.nciLowLightAlarm
Multi Sensor Device 1.LightSensor
Multi Sensor Device 2.NodeObject
AL_NO_CONDITION
NodeObject.nvoAlarm.alarm_type
Multi Sensor Device 1.TempSensor
TempSensor.nvoTemperature
Virtual Functional Block.nciHighTempAlarm
Multi Sensor Device 1
Multi Sensor Device 2
Channel 1

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*).

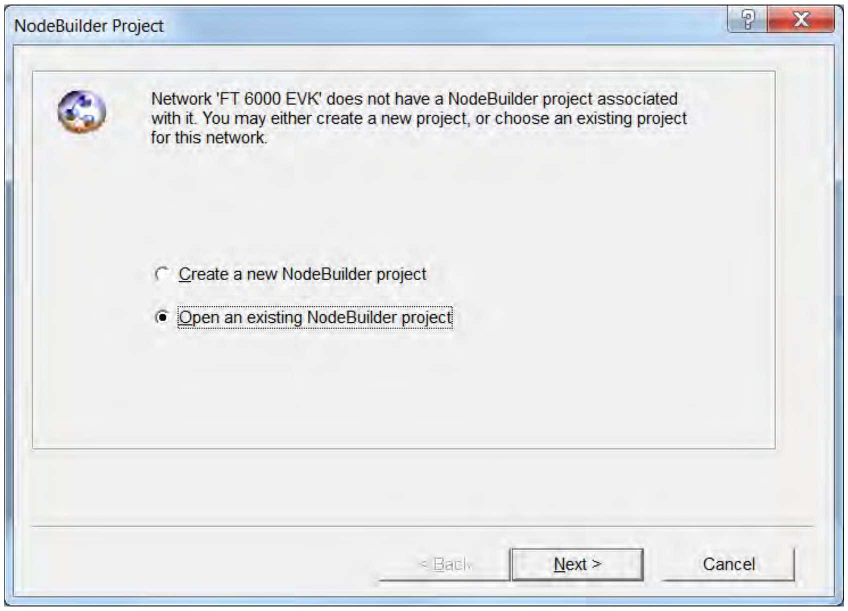
The following sections describe how to view the Neuron C source files in the FT 6000 EVB example applications based on the IzoT NodeBuilder tool, and they describe the next steps for developing a device with your development tool.

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 the view the Neuron C source files with IzoT NodeBuilder tool, follow these steps:

1. Restore the IzoT CT 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 IzoT CT drawing, click **Add-Ins,** then click OpenLNS CT 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**.

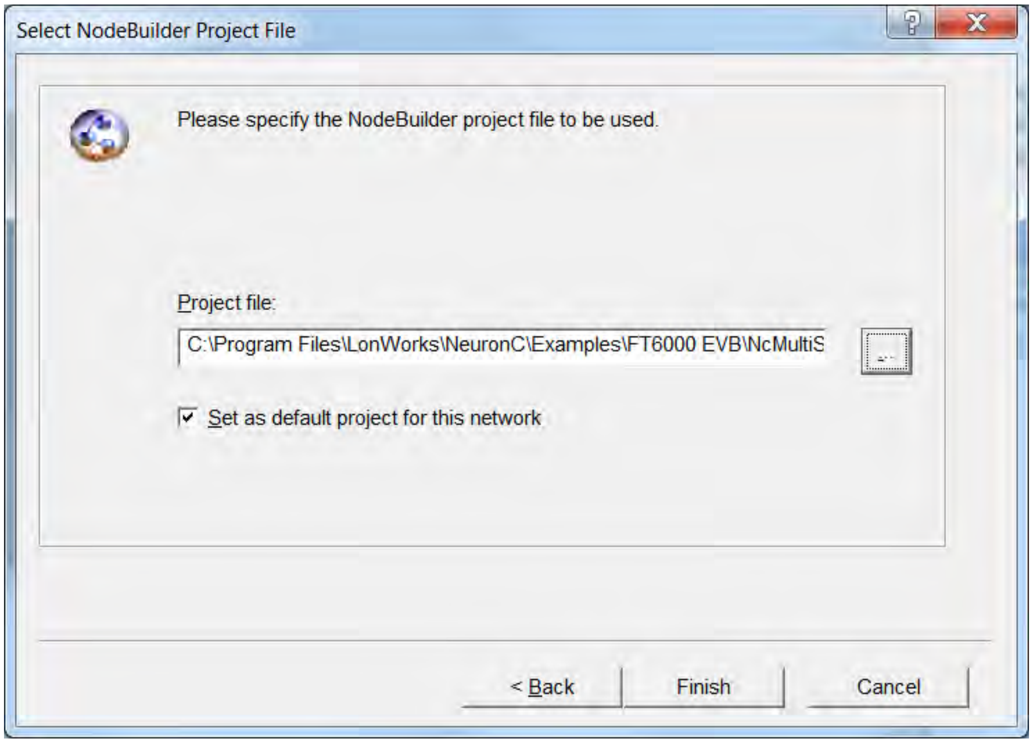

NodeBuilder Project

Network 'FT 6000 EVK' does not have a NodeBuilder project associated with it. You may either create a new project, or choose an existing project for this network.

( ) Create a new NodeBuilder project

(●) Open an existing NodeBuilder project

4. Click **Next**
5. The **Select NodeBuilder Project File** dialog opens. Click the browse 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. Click the **Open** button.


Select NodeBuilder Project File

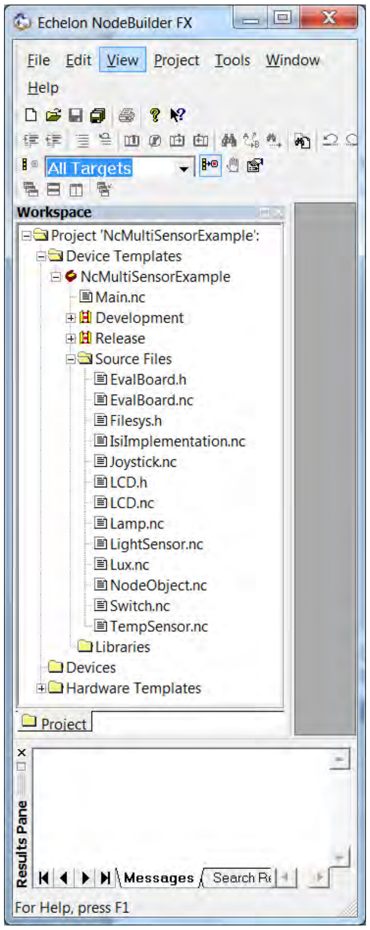
Please specify the NodeBuilder project file to be used.

Project file:
C:\Program Files\LonWorks\NeuronC\Examples\FT6000 EVB\NcMultiS

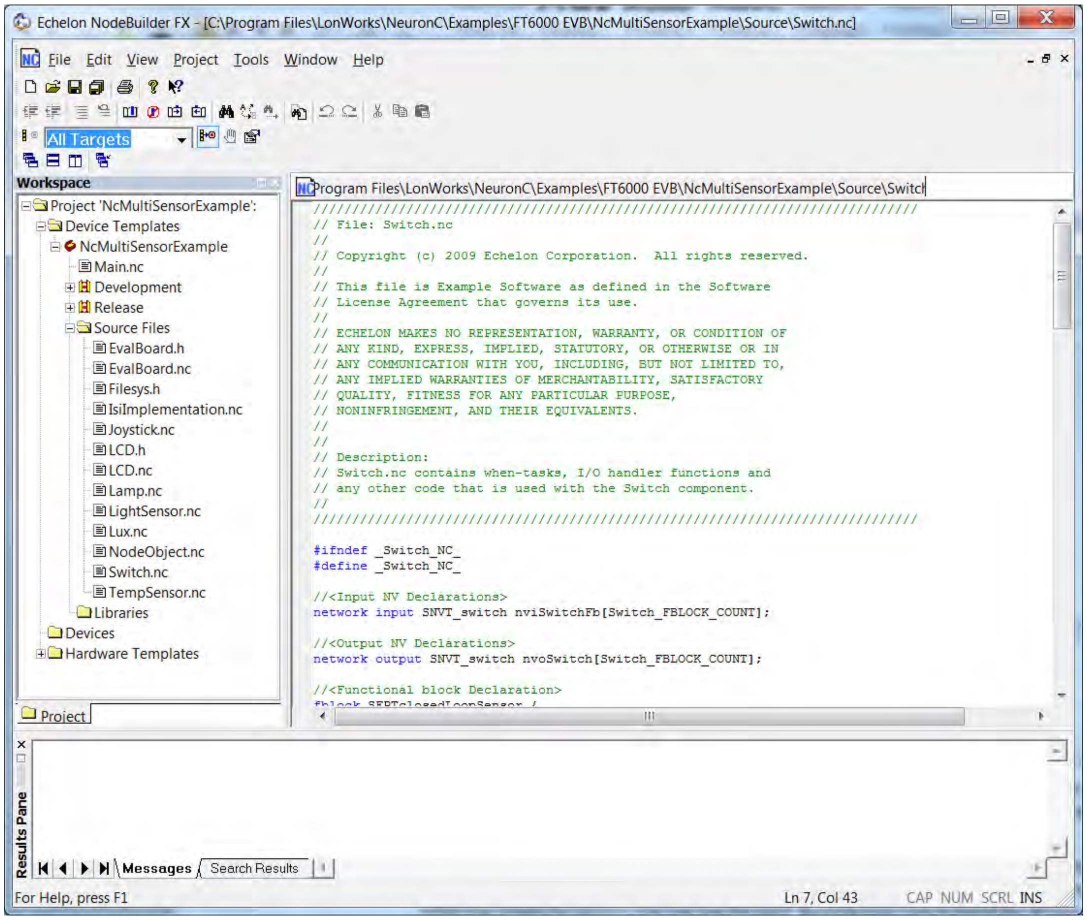
[✓] Set as default project for this network

6. Click **Finish**. The NodeBuilder Project Manager opens.
7. 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.



8. 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.



```
///////////////////////////////////////////////////////////////////////////////
// File: Switch.nc
//
// Copyright (c) 2009 Echelon Corporation.  All rights reserved.
//
// This file is Example Software as defined in the Software
// License Agreement that governs its use.
//
// 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.
//
//
// Description:
// Switch.nc contains when-tasks, I/O handler functions and
// any other code that is used with the Switch component.
//
///////////////////////////////////////////////////////////////////////////////

#ifndef _Switch_NC_
#define _Switch_NC_

//<Input NV Declarations>
network input SNVT_switch nviSwitchFb[Switch_FBLOCK_COUNT];

//<Output NV Declarations>
network output SNVT_switch nvoSwitch[Switch_FBLOCK_COUNT];

//<Functional block Declaration>
```

9. 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.

10. 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*.

2

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.

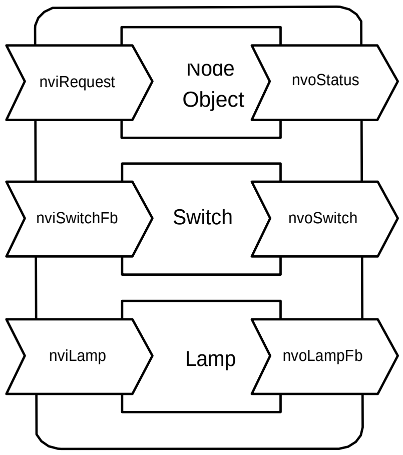
Device Interface

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.

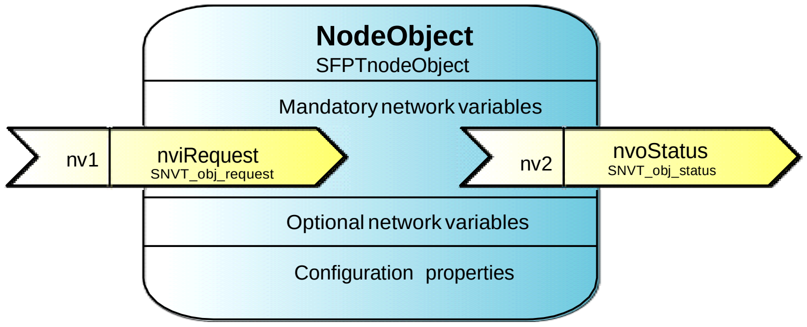


Details

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

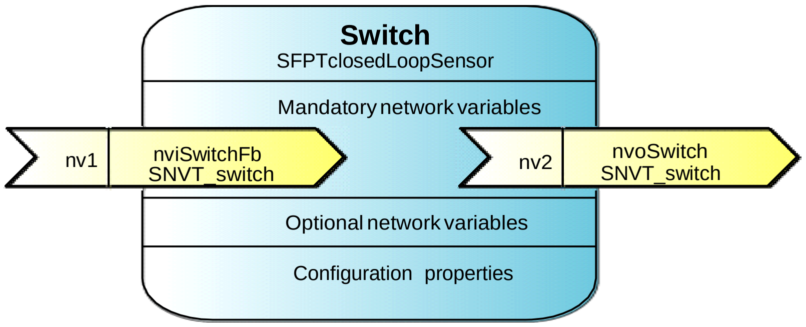
The **NodeObject** functional block returns invalid request for all requests that are not implemented. It returns an invalid ID for all IDs that are not implemented.



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

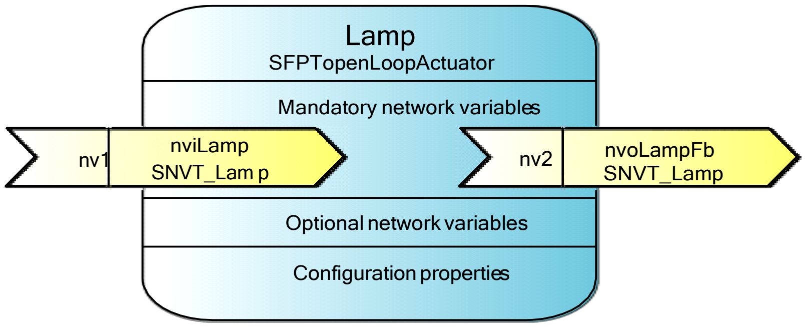
The **Switch** functional block represents the **SW1** push button on the FT 6000 EVB. The **Switch** functional block contains **SNVT_switch** input and output network variables.



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

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 **LED1** on the FT 6000 EVB.
nviLamp	SNVT_switch	Output	Receives the value and state of a **SNVT_switch** network variable and updates **LED1** on the FT 6000 EVB accordingly.
nviSwitchFb	SNVT_switch	Input	Lets you send the current value and state of **LED1** to the **Switch** 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 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. 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.

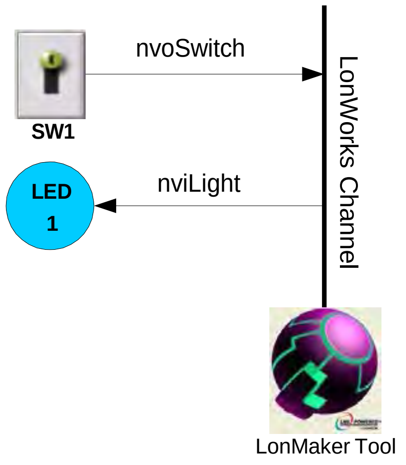
Lamp.nc	This source file implements the **Lamp** functional block. It contains a single *when*-task that illuminates or extinguishes an LED on the FT 6000 EVB when the **nviLamp** network variable in the **Lamp** functional block is updated. This *when*-task also copies the updated value to the **nvoLampFb**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

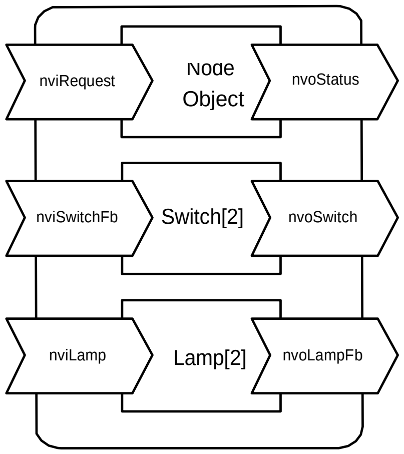
The following section summarizes and then details the *NcSimpleIsiExample* application.

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

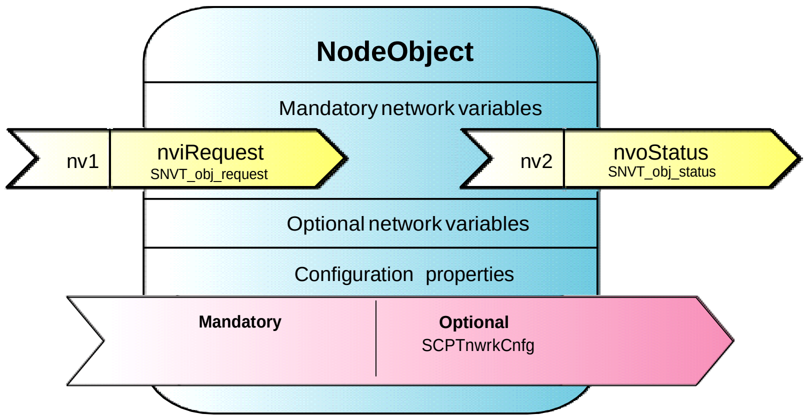


Details

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

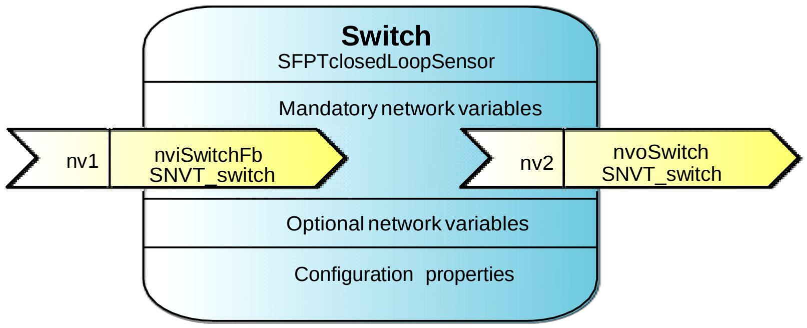
The **NodeObject** functional block returns invalid request for all requests that are not implemented, returns an invalid ID for all IDs that are not implemented, and indicates the current network configuration mode (ISI or managed).



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	SCPTnwrkCnfg	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.

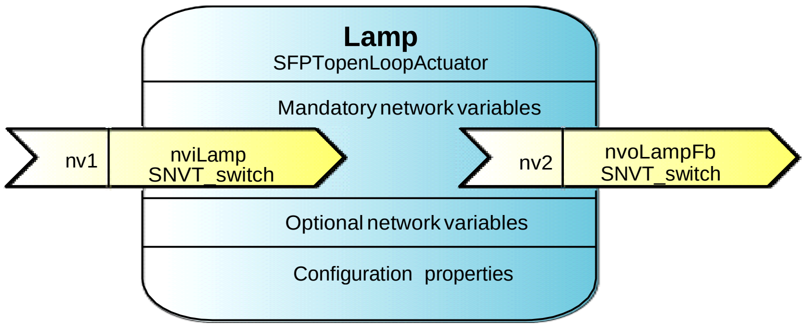


FB/NV Name	Type	Direction	Description
Switch[2]	SFPTclosedLoopSensor		Represents the **SW1** and **SW2** push button on the FT 6000 EVB.
nvoSwitch	SNVT_switch	Output	Transmits the values and states of the **SW1** and **SW2** buttons on the FT 6000 EVB.

nviSwitchFb	SNVT_switch	Input	Lets you create feedback connections between the **SW1** and **SW2** push buttons and **SNVT_switch** 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 **LED1** and **LED2** on the FT 6000 EVB.
nviLamp	SNVT_switch	Output	Receives the value and state of **SNVT_switch** network variables and updates **LED1** and **LED2** on the FT 6000 EVB accordingly.
nviLampFb	SNVT_switch	Input	Lets you send the current value and state of **LED1** and **LED2** to **SNVT_switch** 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.

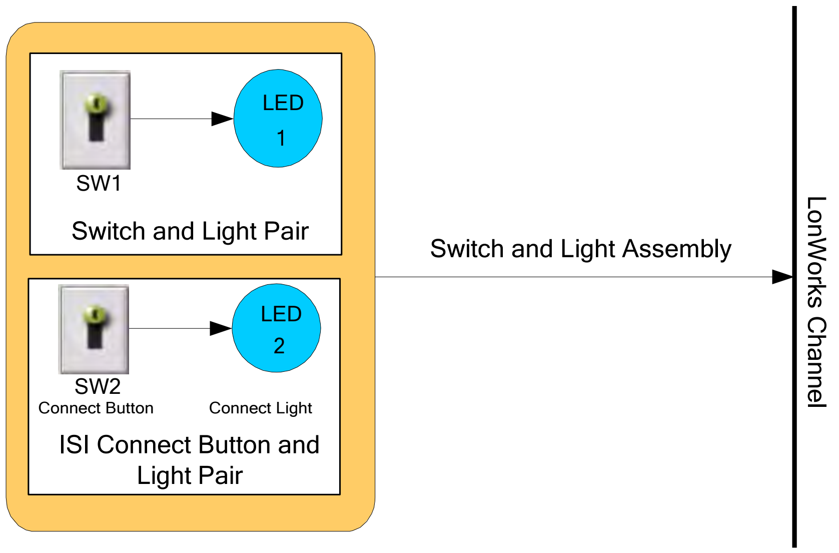
I/O Interaction

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.

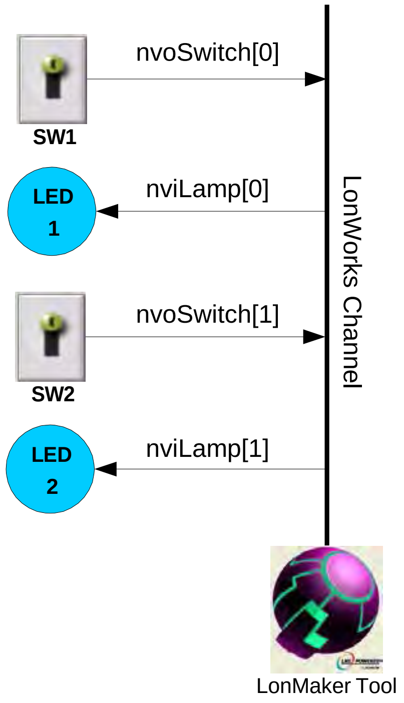
ISI Mode

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.

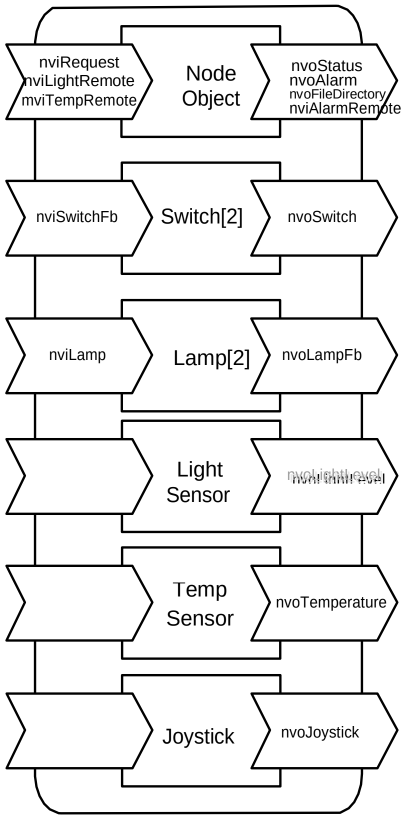
Notes:

- 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

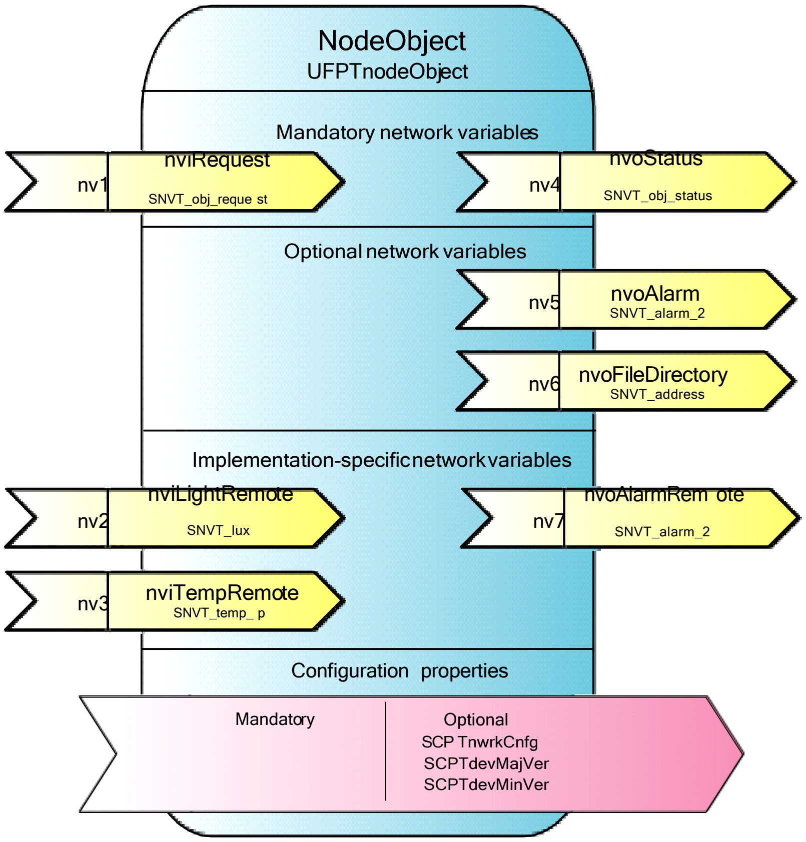
The following sections detail the **Node Object**, **Switch**, **Lamp**, **LightSensor**, **TempSensor**, and **Joystick** functional blocks in the *NcMultiSensorExample* device interface and their network variables and configuration properties.

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.

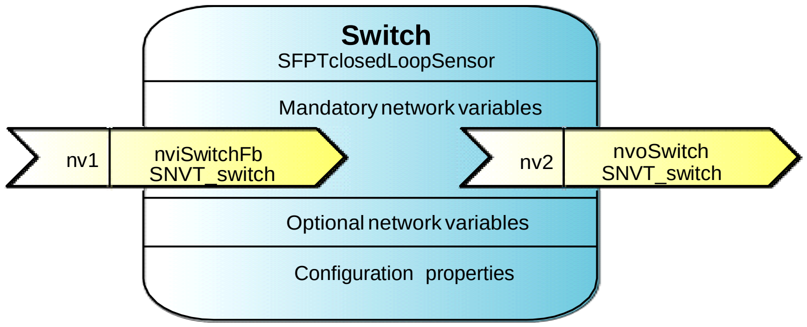


FB/NV/CP Name	**Type**	**Direction**	**Description**
NodeObject	UFPTnodeObject		Lets you monitor the functional blocks within a device.
nviRequest	SNVT_obj_request	Input	Lets you place a functional block in the device interface in a specific state. Handles RQ_NORMAL, RQ_UPDATE_STATUS, RQ_REPORT_MASK 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

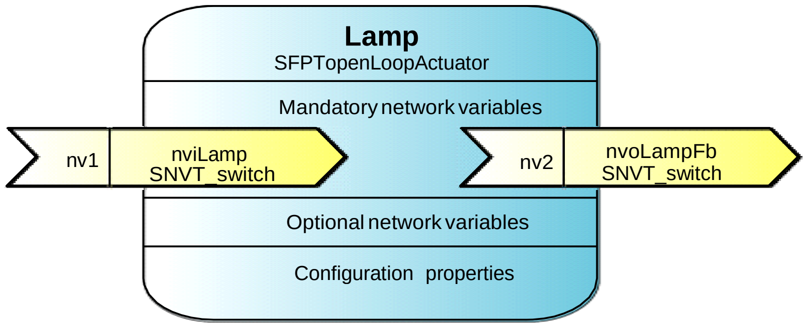
The **Switch** functional blocks represent the **SW1** and **SW2** push buttons on the FT 6000 EVB. They contain **SNVT_switch** input and output network variables.



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

Lamp Functional Blocks

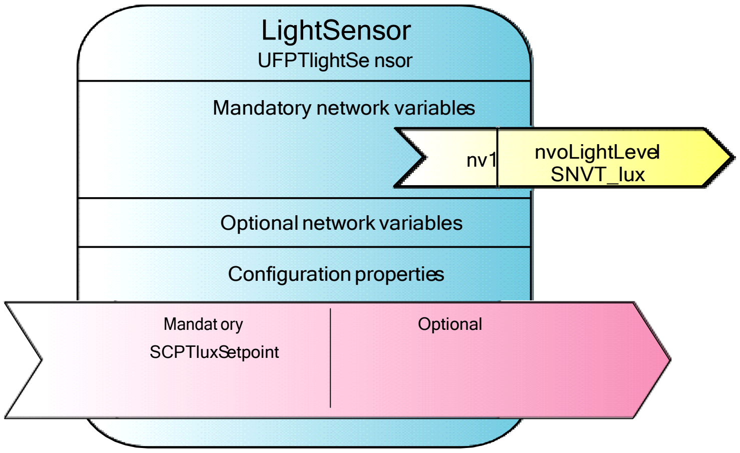
The **Lamp** functional blocks represent **LED1** and **LED2** on the FT 6000 EVB. They contain **SNVT_switch** input and output network variables.



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

Light Sensor Functional Block

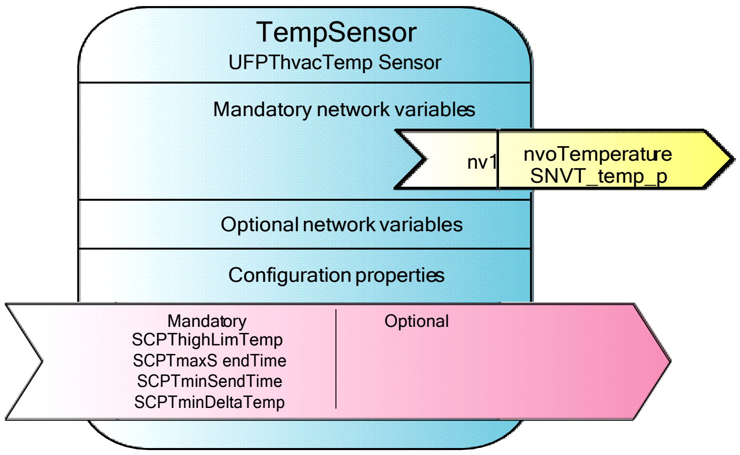
The **LightSensor** functional block represents the light-level sensor I/O object on the FT 6000 EVB. It includes a **SNVT_lux** output network variable, and a mandatory **SCPTluxSetpoint** configuration property that stores the set point for the light alarm's low limit.



FB/NV/CP Name	Type	Direction	Description
LightSensor	UFPTlightSensor		Represents the light-level sensor I/O object (**LIGHT**) 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 **40** lux.

Temperature Sensor Functional Block

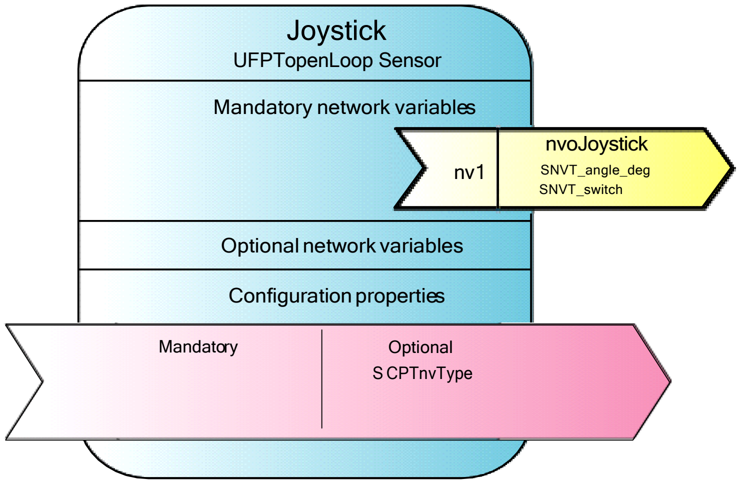
The **TempSensor** functional block represents the temperature sensor I/O object on the FT 6000 EVB. It includes a **SNVT_temp_p** output network variable, and a mandatory **SCPThighLimTemp** configuration property that stores the set point for the temperature alarm's high limit.



FB/NV/CP Name	Type	Direction	Description
TempSensor	UFPThvacTempSensor		Represents the temperature sensor I/O object (**TEMP**) 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	SCPThighLimTemp	Input	Stores the set point for the temperature alarm's high limit. Implemented as a configuration network variable (CPNV) with a default value of **35**^{o} C.
nciMaxSendTime (heartbeat)	SCPTmaxSendTime	File CP	The maximum period of time that can elapse without an update to **nvoTemperature** being transmitted. Implemented as a file configuration property with a default value of **60**s.
nciMinSendTime (throttle)	SCPTminSendTime	File CP	The minimum period of time that must elapse before updates to **nvoTemperature** are transmitted. Implemented as a file configuration property with a default value of **1**s.
nciMinDelta	SCPTminDeltaTemp	File CP	The minimum change to the value of **nvoTemperature** required to transmit an update. Implemented as a file configuration property with a default value of **0.5**^{o} C. Applies to the **nvoTemperature** 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	Stores the value of the last button pressed on the joystick. Has a changeable network variable type. By default, it uses a **SNVT_angle_deg** type, but you can change it to a **SNVT_switch** type. The possible values for **nvoJoystick** are as follows:
nciNvType	SCPTnvType	Input	Stores the current data type used by **nvoJoystick** (**SNVT_angle_deg** or **SNVT_switch**. Implemented as a configuration network variable. Applies to the **nvoJoystick** network variable.

	SNVT_angle_deg	SNVT_switch
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

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).

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.

unsigned long EvalBoardGetTemperature(void)

Gets the current temperature from the temperature sensor on the FT 6000 EVB and scales the raw value to the resolution required by **SNVT_temp_p**.

unsigned long EvalBoardGetLightLevel(void)

Gets the current light level from the light-level sensor on the FT 6000 EVB and passes the value to the **CalculateLux()** function in the **lux.nc** file. The **CalculateLux()** function converts the light level from a raw value to lux.

JoystickDirection EvalBoardGetJoystick(void)

Determines in which direction the joystick on the FT 6000 EVB was last pressed.

filesys.h

This header file contains information and function declarations for configuration properties that have been implemented as configuration files.

The **TempSensor** functional block includes **SCPTminDeltaTemp**, **SCPTminSendTime**, **SCPTmaxSendTime** configuration properties that are implemented as configuration files.

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.

Joystick.nc

This source file implements the **Joystick** functional block, and it includes the following functions that process type changes to the **nvoJoystick** network variable and process changes in the direction of the joystick on the FT 6000 EVB.

when (nv_update_occurs(nciNvType))

Calls the **ProcessTypeChange()** function if the **Joystick** functional block is enabled. The **ProcessTypeChange()** function is used to change the type of the **nvoJoystick** changeable-type network variable. The **nvoJoystick** network variable uses a **SNVT_angle_deg** type by default, but it can be changed to a **SNVT_switch** type.

void ProcessJoystick(void)

Stores the current direction of the joystick.

void ProcessTypeChange(void)

Changes the type of the **nvoJoystick** network variable to a

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.

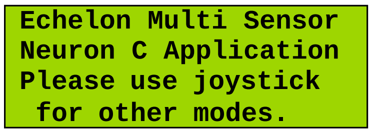
The **ProcessTemperatureSensor()** function also passes the temperature to a function in the **LCD.nc** file for display on the LCD, and evaluates temperature alarm conditions.

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.

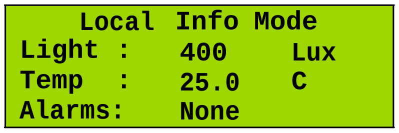


```
Echelon Multi Sensor
Neuron C Application
Please use joystick
 for other modes.
```

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

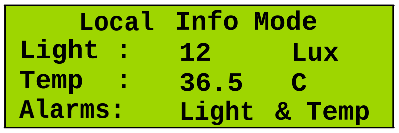
Local Info Mode Panel

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



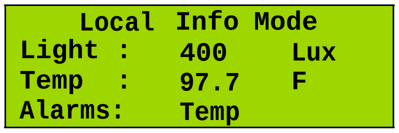
```
    Local Info Mode
Light :   400    Lux
Temp  :   25.0   C
Alarms:   None
```

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.



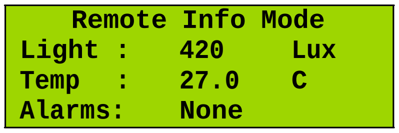
```
    Local Info Mode
Light :   400    Lux
Temp  :   97.7   F
Alarms:   Temp
```

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.

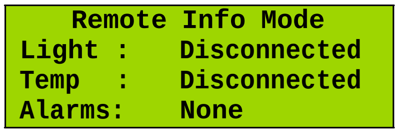
You can connect the network variables of two FT 5000 EVBs running the *NcMultiSensorExample* application in managed mode using the IzoT Commissioning tool or other network tool. For more information how to do this, see *Creating Connections in Managed Mode* in Chapter 1.



```
   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
```

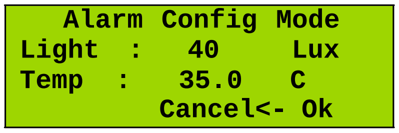
You can toggle the joystick down or press the center button to display the **Alarm Config Mode** panel. You can toggle the joystick up to return to the **Local Info Mode** panel.

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

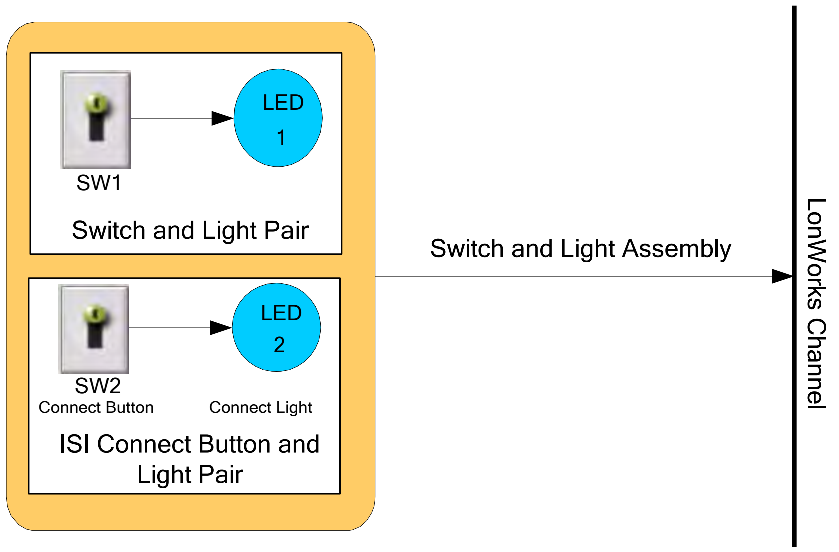
The following table lists the I/O pins used for the **SW1** button, **SW2** button, **LED1**, **LED2**, the **LIGHT** level sensor, the **TEMP** sensor, and joystick (**SW3**) on the FT 5000 EVB.

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 LIGHT	IO7 IO0, IO1	Touch I/O. The temperature sensor uses a 1-Wire Dallas DS18S20 digital thermometer device. I^2C. The I^2C 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 **SW2** 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^2C. The I^2C address for the LCD is 0x28.

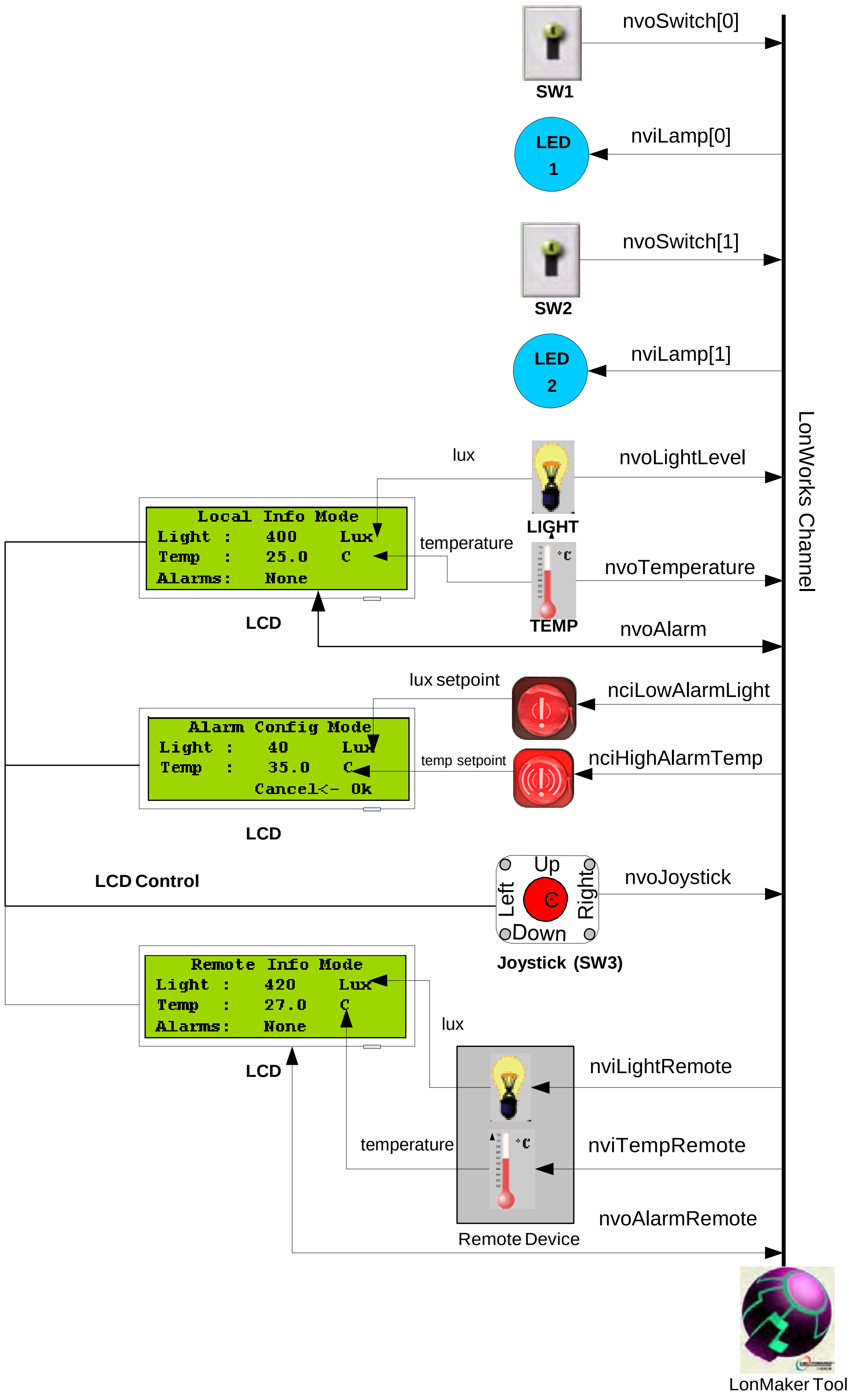
ISI Mode

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.


nvoSwitch[0]
SW1
LED 1
nviLamp[0]
nvoSwitch[1]
SW2
LED 2
nviLamp[1]
LonWorks Channel
lux
nvoLightLevel
Local Info Mode
Light : 400 Lux
Temp : 25.0 C
Alarms: None
LIGHT
temperature
nvoTemperature
TEMP
LCD
nvoAlarm
lux setpoint
nciLowAlarmLight
Alarm Config Mode
Light : 40 Lux
Temp : 35.0 C
Cancel<- Ok
temp setpoint
nciHighAlarmTemp
LCD
LCD Control
Up
Left
Right
Down
nvoJoystick
Joystick (SW3)
Remote Info Mode
Light : 420 Lux
Temp : 27.0 C
Alarms: None
lux
LCD
nviLightRemote
temperature
nviTempRemote
nvoAlarmRemote
Remote Device
LonMaker Tool