ECHELON®

Pyxos FT EVK User's Guide

078-0342-01A

Echelon Corporation
www.echelon.com

Welcome

Echelon's Pyxos™ FT EVK Evaluation Kit is a set of hardware and software tools that you can use to demonstrate the functions and capabilities of Pyxos FT technology and to develop your own devices that incorporate Pyxos FT technology. You can also use the Pyxos FT EVK to develop devices that integrate a Pyxos FT network with a control network based on the LONWORKS® platform.

To learn more about the Pyxos FT platform, see the *Introduction to the Pyxos FT Platform*.

The Pyxos FT EVK includes examples devices that you can use to create a Pyxos FT network that includes a Pyxos controller, called a Pyxos *Pilot*, and three Pyxos I/O devices, called Pyxos *Points*. Two of these Pyxos Points demonstrate how you can use a microprocessor with a Pyxos FT Chip; these Points are called *hosted Points*. One of the Pyxos Points demonstrates how you can use a Pyxos FT Chip without a separate microprocessor; this Point is called an *unhosted Point*. The sample devices in the Pyxos FT EVK demonstrate a simple control system with multiple sensors and actuators, plus a controller.

The Pyxos FT EVK includes example software that demonstrates a complete control system using the example devices that are included with the Pyxos FT EVK. This software demonstrates how you can use a Windows® computer to monitor and control a Pyxos FT network through a Pyxos Pilot. The example software is pre-loaded in the Pyxos FT EVK devices, and is included on the Pyxos FT EVK CD in binary form (so that you can re-load the example software if needed) and as C source code that you can use to learn how to develop Pyxos Pilots and Pyxos Points.

The Pyxos FT EVK includes software that you can use to develop applications for Pyxos Pilots and Pyxos Points. This software includes C source code for the Pyxos FT application programming interface (API), and it includes two Windows application programs, the Pyxos FT Interface Developer utility, which simplifies development of Pyxos applications that use the Pyxos API, and the Pyxos Network Example human-machine interface (HMI) application program, which is a visualization tool for monitoring and controlling the Pyxos example software.

This document describes the Pyxos FT EVK hardware, how to assemble the hardware, and how to use the hardware and software that is included with the Pyxos FT EVK to demonstrate Pyxos FT technology and to develop your own Pyxos Pilots and Points.

Except where specified otherwise, the term *Pyxos* in this document refers to Pyxos FT technology.

Audience

This manual provides information for hardware, software, and firmware engineers who are evaluating Pyxos FT technology for use in their products or who are developing devices using Pyxos FT technology.

Related Documentation

The *Pyxos FT EVK User's Guide* describes the Pyxos FT EVK. In addition, the following guide provides a quick summary of how to use the Pyxos FT EVK:

- *Pyxos FT EVK Quick Start Guide*. This guide helps you set up and install the Pyxos FT EVK Evaluation Boards and helps you get started with the Pyxos FT EVK examples. A printed copy of this guide is included with the Pyxos FT EVK.

The following manuals describe the Pyxos FT platform:

- *Introduction to the Pyxos FT Platform*. This manual introduces the concepts, technology, and tools for the Pyxos FT platform.
- *Pyxos FT Chip Data Book*. This manual provides hardware specifications for working with the Pyxos FT Chip.
- *Pyxos FT Programmer's Guide*. This manual describes the API for the Pyxos FT platform, how to use the API to develop applications that use the Pyxos FT platform, and how to program directly to the Pyxos FT Chip when you cannot use the API.

The following manual provides additional information that can help you to develop applications for the Pyxos FT platform:

- *Introduction to the LONWORKS System.* This manual provides an introduction to the ANSI/CEA-709.1 (EN14908) Control Networking Protocol, and provides a high-level introduction to LONWORKS networks and the tools and components that are used for developing, installing, operating, and maintaining them.

After you install the Pyxos FT EVK software, you can view all of these documents from the Windows **Start** menu: select **Programs → Echelon Pyxos FT EVK → Documentation,** then select the document that you want to view.

In addition to the documentation that is included with the Pyxos FT EVK, the following documents are also useful for Pyxos FT development projects and are available from the Echelon Web site (www.echelon.com):

- *ShortStack User's Guide*. This manual describes how to develop applications for LONWORKS devices that use the ShortStack® Micro Server. It also describes the architecture of a ShortStack device and how to develop one.
- *LONWORKS Host Application Programmer's Guide*. This manual describes how to create LONWORKS host applications. Host applications are application programs that run on hosts other than Neuron® Chips and use the LONTALK® protocol to communicate with devices on a LONWORKS network.

All of the Pyxos documentation, and related product documentation, is available in Adobe® PDF format. To view the PDF files, you must have a current version of the Adobe Reader®. The Pyxos FT EVK CD includes the English-language version of the Adobe Reader; you can download other language versions from Adobe at: www.adobe.com/products/acrobat/readstep2.html.

FCC Compliance Statement – Class A

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Title 47 of the Code of Federal Regulations (CFR) Part 15 of the United States Federal Communications Commission (FCC). These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instructions in this manual, can cause interference with radio communications. Operation of this equipment in a residential area is likely to cause harmful interference, in which case the user will be required to correct the interference at his or her own expense.

IC Compliance Statement – Class A

This Class A digital apparatus complies with Canadian ICES-003.

Cet appareil numérique de la classe A est conforme à la norme NMB-003 du Canada.

ECHELON®

DECLARATION OF CONFORMITY

Pyxos™ FT EVK Evaluation Kit

Application of Council Directive:	89/336/EEC, Electromagnetic Compatibility Directive (EMC)
Manufacturer's Name:	Echelon Corporation
Manufacturer's Address:	550 Meridian Avenue San Jose, CA 95126 USA
Manufacturer's Address in Europe:	Echelon BV Printerweg 3 3821 AP Amersfoort The Netherlands
Product Model Numbers:	11000R-10-11, 11000R-10-12, 11000R-10-13, 11000R-10-14
Type of Equipment:	Information Technology Equipment
Standards to which Conformity is Declared:	EN 55022:1998 EN 55024:1998 EN 61000-4-2 EN 61000-4-3 EN 61000-4-4 EN 61000-4-5 EN 61000-4-6 EN 61000-4-11

I, Wim Meijer, hereby declare that the equipment specified above conforms to the above Directives and Standards.

For a signed copy of this Declaration of Conformity, go to the Echelon Web site.

Place: Amersfoort, The Netherlands Date: January 2007 Position: Controller, Echelon Europe

Table of Contents

1

Introduction to the Pyxos FT EVK

This chapter introduces the Pyxos FT EVK, including the hardware contents of the EVK, power considerations, network termination, connecting to a computer using USB or to a LONWORKS network.

Introduction to the Pyxos FT EVK

The Pyxos FT EVK Evaluation Kit provides a set of hardware and software tools that you can use to evaluate the Pyxos FT platform and to develop applications and devices that use Pyxos FT technology. You can also use the Pyxos FT EVK to evaluate the development of control network applications, including those that use the LONWORKS platform.

The Pyxos FT EVK includes the following evaluation boards that demonstrate the features and functions of the Pyxos FT Chip and of Pyxos FT technology, and that you can use to develop your own Pyxos applications:

- The *Pyxos FT EV Pilot Evaluation Board*

  The EV Pilot is the controller for the Pyxos FT EVK network. It includes an example application that controls the Pyxos Network Example. This program demonstrates how to use the Pyxos FT Pilot API to monitor and control a Pyxos FT network.

  The EV Pilot includes an Atmel ARM7 host microprocessor that is connected to, and communicates with, a Pyxos FT Chip. The Pilot is responsible for configuring, maintaining, and communicating with the Pyxos Points in the Pyxos FT network, as well as for receiving information from the network, and distributing that information to the appropriate Pyxos Points. The EV Pilot can also communicate with LONWORKS devices.

- The *Pyxos FT EV-Actuator Point Evaluation Board*

  The EV-Actuator Point is a Pyxos Point that serves as a hosted analog and digital actuator for the Pyxos FT EVK network. The EV-Actuator Point also includes an Atmel ARM7 host microprocessor.

- The *Pyxos FT EV-Sensor Point Evaluation Board*

  The EV-Sensor Point is a Pyxos Point that serves as a hosted multi-sensor for the Pyxos FT EVK network. The EV-Sensor Point includes an Atmel ATtiny13 microprocessor. The sensors measure temperature, light level, and DC voltage.

- The *Pyxos FT EV-Nano Point Evaluation Board*

  The EV-Nano Point is a Pyxos Point that serves as a simple unhosted digital sensor for the Pyxos FT EVK network. The EV-Nano Point shows how a Point can participate in a Pyxos FT network without a host microprocessor.

For more information about the hardware that is included with Pyxos FT EVK, see Chapter 2, *Getting Started with the Pyxos FT EVK*, on page 11, and Chapter 3, *Pyxos FT EVK Hardware Details,* on page 17.

The Pyxos FT EVK includes the following Windows application programs:

- The Pyxos FT Interface Developer utility, which allows you to create Point interface definitions for Pyxos FT network applications. You can use these definitions to simplify Pyxos application development.

For more information about the Pyxos FT Interface Developer utility, see the *Pyxos FT Programmer's Guide*.

- The Pyxos Network Example human-machine interface (HMI) application program, which is a visualization tool for monitoring and controlling the Pyxos Network Example application.

  For more information about the Pyxos Network Example HMI application program, see Chapter 5, *Running the Pyxos Network Example HMI Application Program,* on page 73.

The Pyxos FT EVK also provides source code for:

- The Pyxos FT Pilot API, Pyxos FT Point API, and Pyxos FT Serial API. You can port the APIs to various host microprocessors for your Pilot and Point applications. The Pyxos FT Serial API source code provides an example port for an ARM7-family microprocessor, the Atmel AT91SAM7S64.

  For more information about the Pyxos FT APIs, see the *Pyxos FT Programmer's Guide*.

- The Pyxos Network Example firmware applications that are pre-loaded in the device's host microprocessors. You can modify the applications and use them to learn how to develop your own Pyxos FT network applications.

  For more information about the Pyxos Network Example firmware application, see Chapter 4, *Using the Pyxos Network Example,* on page 63, and Chapter 8, *Exploring the Pyxos Network Example,* on page 105.

- The LONWORKS Neuron C model file that is used for the EV Pilot firmware application and the ShortStack® API that the EV Pilot firmware application uses to communicate with the ShortStack Micro Server on the Pyxos FT EV Pilot Evaluation Board.

  For more information about the LONWORKS Neuron C model file, see *Using the Pyxos-LonWorks Gateway* on page 121. For more information about the ShortStack API, see the *ShortStack User's Guide*.

For a more detailed introduction to the Pyxos FT platform and Pyxos FT technology, see the *Introduction to the Pyxos FT Platform*.

Pyxos FT EVK Hardware Features

The Pyxos FT EVK evaluation boards are fully functional Pyxos FT devices that you can use to demonstrate key features and functions of Pyxos FT technology and to develop your own Pyxos applications. At a system level, you can use the Pyxos FT EVK evaluation boards to demonstrate the following features:

- Link power at 24 VAC. The Pyxos FT network provides power to each of the EV Points as well as providing communications between the EV Points and the EV Pilot.
- LONWORKS connectivity. You can connect the EV Pilot to a LONWORKS network to share Pyxos data with a LONWORKS network, or you can connect multiple EV Pilots and other Pyxos Pilots to a LONWORKS

network to share data among Pyxos Points on multiple Pyxos FT networks and devices on the LONWORKS network.

- Variety of Point designs:
    - High-performance ARM7-based Point (the EV-Actuator Point)
    - Low-cost AVR®-based Point (the EV-Sensor Point)
    - Small, low-cost, unhosted Point (the EV-Nano Point)
- Free-topology communication. You can configure a Pyxos FT network as a bus or as a free-topology network. The Pyxos FT EVK uses a free-topology network with a built-in network terminator on the EV Pilot evaluation board.
- Deterministic response time. The Pyxos FT network provides predictable communications response times. The Pyxos Network Example firmware includes a Performance Demo that you can run from the Network Example HMI application program.
- Switching and linear power supply designs. The Pyxos FT EVK provides examples of two power supply designs for link-powered Points, a switching power supply and a linear power supply.
- Network coupling options. The Pyxos FT EVK provides examples of two different options for coupling to the Pyxos FT network, transformer-isolated coupling and floating non-isolated coupling.

The Pyxos FT EVK includes the necessary power supply and network cables so that you can run the example Pyxos FT network out of the box.

Pyxos FT EVK Hardware Contents

The Pyxos FT EVK includes the hardware listed in **Table 1**.

Table 1. Pyxos FT EVK Contents

Item Description	Quantity
Model 11200R-1P Pyxos FT EV Pilot	1
Model 11220R Pyxos FT EV-Actuator Point	1
Model 11210R Pyxos FT EV-Sensor Point	1
Model 11230R Pyxos FT EV-Nano Point	1
Power supply adapter	1
Pyxos FT network cables	3
I/O cable (used to connect the analog output of the EV-Actuator Point to the analog input of the EV-Sensor Point)	1
Universal Serial Bus (USB) A/B cable	1

Item Description	Quantity
Pyxos FT Chip samples	10
Pyxos FT EVK Quick Start Guide	1
Pyxos FT EVK CD	1
Echelon Technical Documentation CD	1
LonScanner™ Protocol Analyzer Demo CD	1
Echelon Training brochure	1
Pyxos FT EVK product registration card	1

Each of the four evaluation boards includes an integrated Pyxos FT Chip.

The Pyxos FT EVK includes an example application that demonstrates the functions and capability of the Pyxos FT Chip and of the Pyxos FT network. To run the example, connect the four evaluation boards together using the supplied Pyxos FT network cables.

The Pyxos FT EVK also includes an example human-machine interface (HMI) application program that you can use to monitor and control the example Pyxos FT network. To run this application program, connect the EV Pilot evaluation board to a computer that will run the HMI application program. You can connect the EV Pilot Evaluation Board to the computer using the supplied USB cable or using a LONWORKS network cable and LONWORKS network interface (not included with the Pyxos FT EVK).

One of the functions of the HMI application program demonstrates network determinism for the Pyxos FT network. To run this demonstration, connect the EV-Actuator Point evaluation board to the EV-Sensor evaluation board using the supplied I/O cable.

This chapter describes the following main tasks for becoming familiar with the Pyxos FT EVK:

- *Power Considerations for the Pyxos FT EVK* on page 6. This section describes the Pyxos FT EVK's link-power supply.
- *Network Termination* on page 6. This section describes the required network terminators for a Pyxos FT network.
- *Connecting the EV Pilot to a Computer Using USB* on page 7. This section describes how to connect the Pyxos FT EV Pilot to a computer to run the HMI application program or to run other Windows software that will interact with the evaluation boards.
- *Connecting to a LonWorks Network* on page 7. This section describes how to connect the Pyxos FT EV Pilot to a LONWORKS network interface and how to configure that network interface as a layer-5 network interface.
- *Developing a Pyxos FT Application* on page 9. This section introduces using the Pyxos FT EVK to develop Pyxos FT applications.

Power Considerations for the Pyxos FT EVK

The Pyxos FT EVK includes a power supply that converts line power to 24 VAC. This power supply provides power to the EV Pilot, which in turn provides link power to the Pyxos FT network. The EV Pilot uses an inductor-based filter to condition the link power for each of the EV Points. For the Pyxos FT EVK, this filter supports up to 6 link-powered Pyxos FT Points (where each Point requires 35-100 mA of application current). This Point limitation applies only to the Pyxos FT EVK, and is not a general limitation for the Pyxos FT Chip. If you provide local power for the Pyxos FT Points, the Pyxos FT EVK supports up to 32 Points.

To include additional link-powered Points in the Pyxos FT EVK network, see the *Pyxos FT Chip Data Book* for distance limitations for the network cable. If you exceed those limitations, both available power and communication integrity can degrade performance.

Recommendation: To connect additional devices to the Pyxos FT EVK network or extend the distance to each Point, use a 24 VDC power supply with appropriate filter circuit for the Pilot. See the *Pyxos FT Chip Data Book* for more information about designing power supplies and filter circuits for a Pyxos FT network.

Network Termination

A Pyxos FT network must be properly terminated. For the Pyxos FT EVK, the EV Pilot evaluation board includes the proper network terminator for an AC link-powered free-topology network. Thus, you can use the EV Pilot network terminator for a free-topology Pyxos FT network.

You can disable the EV Pilot network terminator and replace it with a different network terminator:

- If you want to use the Pyxos FT EVK in a bus-topology network (which requires two network terminators)
- If you do not want the network terminator to be co-located with EV Pilot (for example, if you want to perform large-network testing)

To disable the EV Pilot network terminator and supply your own network termination, perform the following steps:

1. Dismount jumper **JP306** on the EV Pilot evaluation board to disable the EV Pilot network terminator.
2. For a free-topology Pyxos FT network, attach a single network terminator, as shown in **Figure 1** on page 7, anywhere in the network.
3. For a bus-topology Pyxos FT network, attach two network terminators, as shown in **Figure 2** on page 7, one at each end of the network.

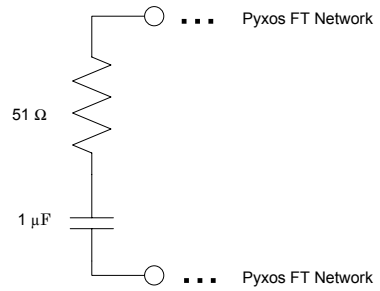


Figure 1. Network Terminator for a Free-Topology Pyxos FT Network

Figure 2. Network Terminators for a Bus-Topology Pyxos FT Network

For details about these network-termination circuits, see the *Pyxos FT Chip Data Book.*

Caution: Do not use Echelon 44100 or Echelon 44101 network terminators for your Pyxos FT network. Because the power transmission characteristics for a Pyxos FT network are different than those of a TP/FT-10 channel, using the Echelon 44100 or Echelon 44101 network terminators in a Pyxos FT network can damage the terminators and damage the network cable.

Connecting the EV Pilot to a Computer Using USB

You can connect the EV Pilot to a USB port on a Windows computer. Connecting the EV Pilot to a computer is not required to evaluate the Pyxos FT technology. However, to use the Pyxos Network Example HMI application program, you must connect the Pyxos FT EV Pilot evaluation board to a Windows computer through either a USB connection or a LONWORKS network interface.

To connect the EV Pilot to a computer using a USB connection, insert the Pyxos FT EVK USB cable into an available USB port on your computer and attach the USB cable to the EV Pilot. The Pyxos FT EVK CD installs a Windows-compatible USB driver that manages the communications between the computer and the EV Pilot.

Connecting to a LONWORKS Network

You can connect the Pyxos FT network to a LONWORKS network, either to connect multiple Pyxos FT networks together or to share Pyxos data with devices on a LONWORKS network. You can also connect the EV Pilot to a LONWORKS network interface that is connected to a Windows computer and run the Network

Example HMI application program. In this configuration, the EV Pilot acts as a LONWORKS device and sends and receives LONWORKS network variable updates to the Pyxos Network Example HMI application program over the LONWORKS network.

Connecting the EV Pilot to a LONWORKS network is not required to evaluate or develop with the Pyxos FT technology.

To connect the EV Pilot to a LONWORKS network, perform the following steps:

1. Ensure that OpenLDV 3.3 or later is installed on your computer. You can install the OpenLDV driver from the Pyxos FT EVK CD, or you can download the latest version from www.echelon.com/openldv. See the OpenLDV ReadMe file for installation instructions.
2. Install an OpenLDV compatible LONWORKS TP/FT-10 network interface in your computer. OpenLDV compatible TP/FT-10 network interfaces include: the *i*.LON 10, *i*.LON 100, *i*.LON 600, PCC-10, PCLTA-20/21, SLTA/2, and U10 network interfaces. The LONWORKS network interface is not included with the Pyxos FT EVK. See the documentation for your network interface for installation instructions.
3. Attach the network interface and your EV-Pilot to a LONWORKS TP/FT-10 twisted pair cable. The LONWORKS cable is not included with the Pyxos FT EVK.

If you are using a PCC-10, a PCLTA-20, or a PCLTA-21 as your network interface, you must configure it to operate as a layer-5 network interface before using it with the EV Pilot. To configure a PCC-10, PCLTA-20, or PCLTA-21 as a layer-5 network interface, perform the following steps:

1. Open the Windows Control Panel and double-click the **LONWORKS Plug 'n Play** icon. The dialog shown in **Figure 3** opens.

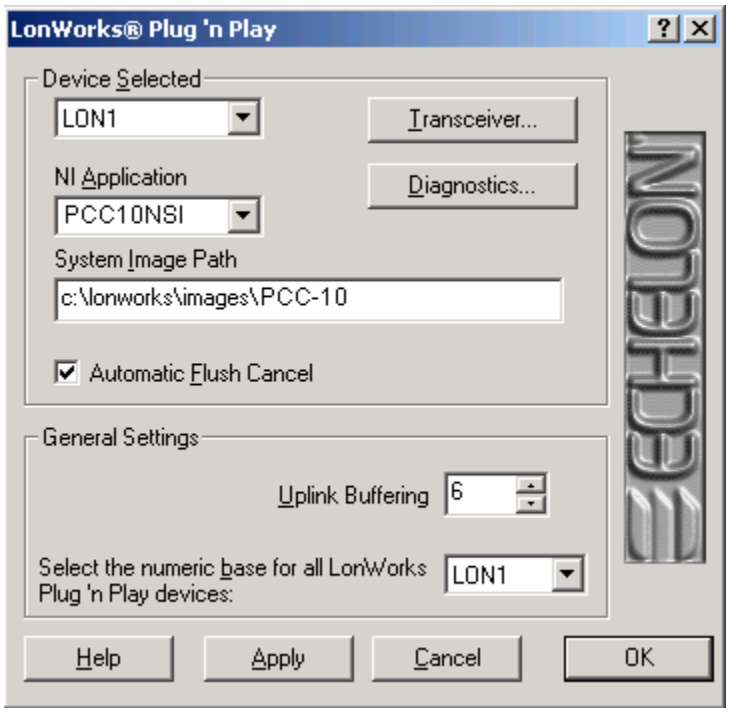


Figure 3. LONWORKS Plug 'n Play Application

2. Select your network interface from the **Device Selected** dropdown list.
3. If you are using a PCC-10, select PCC10NSI from the **NI Application** dropdown list. If you are using a PCLTA-20 or a PCLTA-21, select NSIPCLTA.
4. Click **OK** to save your changes and close the LONWORKS Plug 'n Play application. You can now use your PCC-10, PCLTA-20, or PCLTA-21 with the Network Example HMI application program, or any other OpenLDV application.

Developing a Pyxos FT Application

In addition to providing a showcase for Pyxos FT technology and running the Pyxos FT Network Example application, you can also use the Pyxos FT EVK as part of an open development platform for your own Pyxos FT applications.

The EV Pilot and the EV-Actuator Point evaluation boards both include a header connector that provides external access to the host microprocessor memory. This header complies with the Institute of Electrical and Electronics Engineers (IEEE) Standard Test Access Port and Boundary-Scan Architecture (IEEE 1149.1-1990) of the Joint Test Action Group (JTAG), and can be used for debugging the host microprocessor's program, for an in-circuit emulator (ICE), or to load program firmware into the host microprocessor's memory.

Use the JTAG header connector to load your own Pyxos FT application firmware into the host processors for the EV Pilot or the EV-Actuator Point.

You can also use the Pyxos FT EVK to test or debug applications for the Pyxos FT platform that use host microprocessors that are different than the ones used on the evaluation boards. You can replace the AVR host microprocessor on the EV-Sensor Point evaluation board by using the Serial Programming Interface (SPI) header on the Evaluation Board to connect the new microprocessor to the Pyxos FT Chip. You can also use a microprocessor that has the same 8-lead, 0.300" Wide Body, Plastic Dual In-line Package (PDIP) package and has the same pinout as the Atmel® ATtiny13 microprocessor. For specific information about the host microprocessors for the Pyxos FT EVK, see Chapter 3, *Pyxos FT EVK Hardware Details* on page 17.

For more information about using the Pyxos FT EVK for Pyxos application development, see Chapter 7, *Developing Pyxos FT Applications Using the Pyxos FT EVK*, on page 89.

.

2

Getting Started with the Pyxos FT EVK

This chapter helps you set up the Pyxos FT EVK evaluation boards and install the Pyxos FT software so that you can begin to demonstrate key features and functions of Pyxos FT technology and to develop your own Pyxos applications.

Assembling the Evaluation Boards

Before you begin: The Pyxos FT evaluation boards are shipped in protective anti-static packaging. When assembling the Pyxos FT evaluation boards, you must not subject the boards to high electrostatic potentials. Wear a grounding strap, or similar protective device, while handling the boards. Avoid touching the component pins, or any other metallic components on the evaluation boards.

Unpack the equipment from the shipping carton. Refer to **Table 1** on page 4 to verify that all hardware items are present. Avoid touching areas of integrated circuitry, because electrostatic discharge can damage the circuits.

Figure 4 shows the four Pyxos FT evaluation boards, as they appear when assembled. For more information about each of the Evaluation Boards, see Chapter 3, *Pyxos FT EVK Hardware Details*, on page 17.

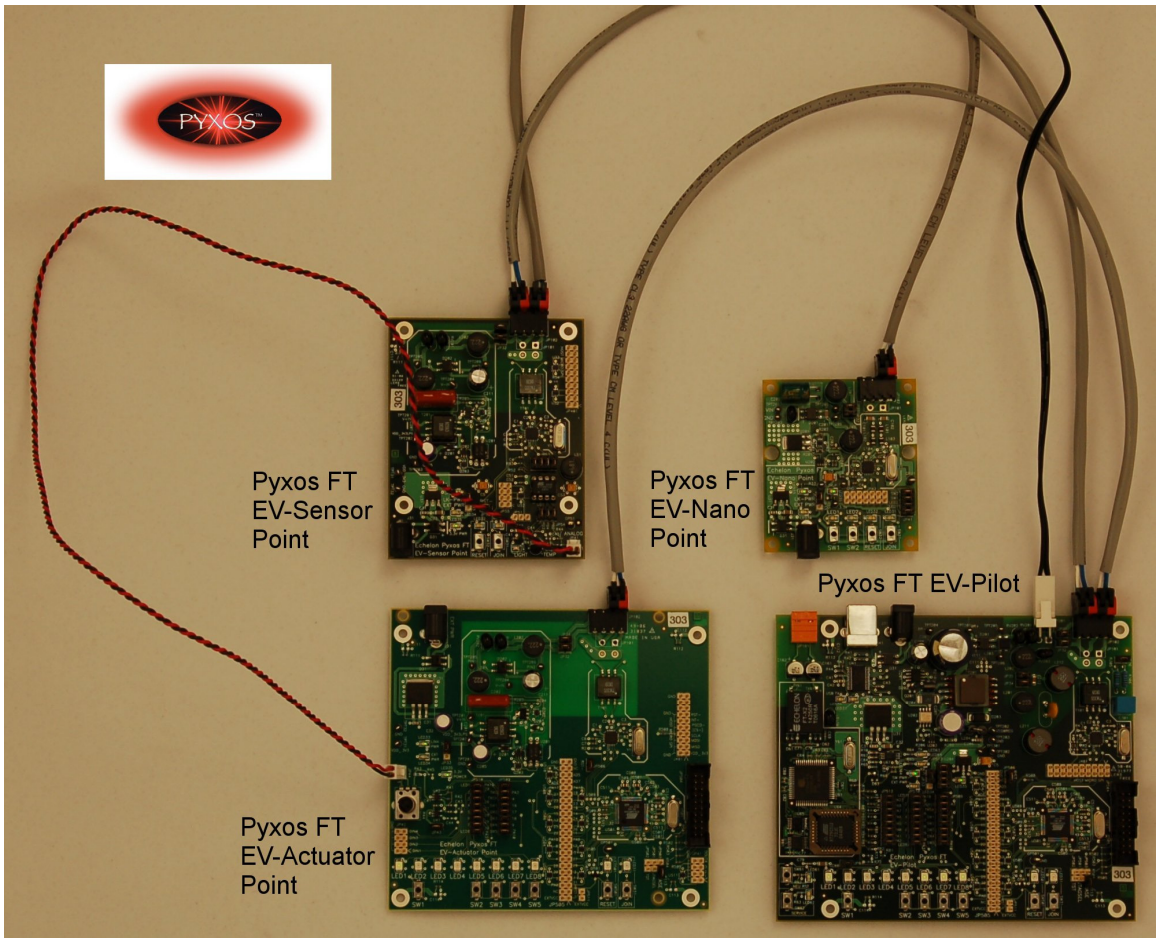


Figure 4. Pyxos FT EVK Evaluation Boards

To assemble the Pyxos FT EVK Evaluation Boards, perform the following steps:

1. Connect the Pyxos FT EVK power supply adapter to the EV Pilot evaluation board, as shown in **Figure 4**, and plug it in. The power LED on the EV Pilot evaluation board illuminates to show that it is running.
2. Connect the Pyxos FT EVK network cables to each of the four evaluation boards, as shown in **Figure 4**. These cables establish the Pyxos FT

network and supply power to each of the EV Point evaluation boards. The power LED on each of the EV Point evaluation boards illuminate to show that they are running.

Note: The Pyxos FT network connector for the EV Pilot evaluation board is the black (rightmost) connector; the orange (leftmost) connector is for an optional LONWORKS network cable (not included with the Pyxos FT EVK).

3. Register the EV-Nano Point. Press the **SW5** button on the EV Pilot to place the Pilot in manual registration mode; verify that **LED8** blinks while the EV Pilot is in this mode. Then, press the Join button on the EV-Nano Point to register the Point. Verify that **LED8** is illuminated and no longer blinking.
4. Verify that **LED6**, **LED7**, and **LED8** on the EV Pilot are illuminated. When these three LEDs are illuminated, the three EV Points are correctly connected to, and communicating with, the EV Pilot.
5. Connect one end of the Pyxos FT EVK I/O cable to the analog output header (**JP44**) of the EV-Actuator Point evaluation board. Connect the other end of the cable to the analog input header (**JP41**) of the EV-Sensor Point evaluation board.
6. Install the Pyxos FT software on a computer. If you plan to use the Network Example HMI application program, or if you plan to write Pyxos FT applications that use the Pyxos FT application programming interface (API), you must install the Pyxos FT software.
7. Connect the EV Pilot to your computer. If you plan to use the Network Example HMI application program, you must connect the EV Pilot to a computer using either the USB cable or a LONWORKS network interface cable.
    - Use the supplied USB cable to connect the EV Pilot to your computer.
    - To demonstrate or develop a LONWORKS interface to your Pyxos FT EV Pilot, install an OpenLDV compatible LONWORKS TP/FT-10 network interface in your computer, and then attach a network interface and the EV Pilot to a LONWORKS TP/FT-10 twisted-pair cable. The LONWORKS network interface and the LONWORKS cable are not included with the Pyxos FT EVK.
8. Run the Pyxos FT Network Example application and the Network Example HMI application program. For more information about running these examples, see Chapter 4, *Using the Pyxos Network Example*, on page 63, and Chapter 5, *Running the Pyxos Network Example HMI Application* Program, on page 73.

Pyxos FT EVK Software System Requirements

This section describes the minimum and recommended hardware and software requirements for the Pyxos FT EVK software.

Hardware Requirements

To run the Pyxos FT EVK Windows software, your computer system must meet the following minimum requirements:

- Intel® Pentium® III 500 MHz processor or AMD™ Athlon® 750 MHz processor
- 128 MB RAM
- 65 MB available hard disk space
- CD-ROM drive
- 1 available Universal Serial Bus (USB) port

The recommended specifications for your computer system include:

- Intel Pentium 4 1.0 GHz processor or AMD Athlon 1.5 GHz processor
- 256 MB RAM
- 100 MB available hard disk space
- CD-ROM or DVD-ROM drive
- 3 available USB ports

In addition, you need a monitor that can support a minimum screen resolution of 1024 x 768 pixels for the Pyxos FT Interface Developer utility and the Pyxos Network Example HMI application program.

Optional hardware, if you need LONWORKS connectivity:

- LONWORKS TP/FT-10 compatible network interface, such as a U10 USB Network Interface or an *i*.LON 100 Internet Server
- LONWORKS TP/FT-10 network cable, with network terminator

Software Requirements

To run the Pyxos FT EVK Windows software, your computer system must meet either one of the following minimum requirements:

- Microsoft® Windows XP, plus Service Pack 2
- Microsoft Windows 2000, plus Service Pack 4

In addition, you must have the following software, all of which is installable from the Pyxos FT EVK CD:

- Microsoft .NET 2.0 (or later) runtime components
- Adobe Reader 7.0.8 or later
- The OpenLDV 3.3 driver, if you need LONWORKS connectivity for LONWORKS network interfaces
- LONMARK® Resource Editor 3.13 or later, if you need to create custom LONMARK resource files and data type definitions

Important: Before you can install the Pyxos FT EVK software, you must log on to your Windows system with a user ID that is a member of the Administrators group or have equivalent administrator privileges.

Installing the Pyxos FT EVK Software

To install the Pyxos FT EVK software, perform the following steps:

1. Insert the Pyxos FT EVK CD-ROM into a CD-ROM or DVD drive. The setup application should start automatically. If it does not, use Windows Explorer to open the contents of the CD-ROM drive, and double click the **Setup** icon. The Pyxos FT EVK setup application's main window opens.
2. From the Pyxos FT EVK setup application's main window, click **Install Products**. The Install Products window opens.
3. From the Install Products window, click **Pyxos FT EVK 1.0**. Follow the installation dialogs to install the Pyxos FT EVK software onto your computer.
4. If you do not already have the Microsoft .NET 2.0 (or later) runtime components on your computer, return to the Install Products window and click **Microsoft .NET Framework 2.0** to install the Microsoft .NET Framework. The .NET runtime components are required to run the Pyxos Network Example HMI application program.
5. If you do not already have the Adobe Reader installed on your computer, return to the Install Products window and click **Adobe Reader 7.0.8** to install the Adobe Reader. A current version of the Adobe Reader is required to view the Pyxos product documentation. The Pyxos FT EVK CD includes the English-language version of the Adobe Reader; you can download other language versions from Adobe at: www.adobe.com/products/acrobat/readstep2.html.
6. If you plan to connect the Pyxos FT network to a LONWORKS network, you can install the LONMARK Resource Editor 3.13 and the OpenLDV 3.3 driver from the Install Products window. These products are optional. You can also download the latest version of the OpenLDV driver from www.echelon.com/downloads.

After you install the Pyxos FT EVK Software, you are ready to use the Pyxos Network Example HMI application program and the Pyxos FT Interface Developer utility. However, you might need to restart your computer if you also installed the Microsoft .NET 2.0 runtime components.

Compatibility

The Pyxos FT EVK software is compatible with Echelon's LONWORKS products, such as the ShortStack Developer's Kit and the LonMaker® Integration Tool.

3

Pyxos FT EVK Hardware Details

This chapter describes the Pyxos FT EVK hardware, including descriptions of the LEDs, push buttons, header connectors, jumper settings, and I/O. It also provides information that you will need when using custom applications and I/O devices with the evaluation boards. It includes the following major sections:

Pyxos FT EV Pilot Evaluation Board

The Pyxos FT EV Pilot serves as the controller for the Pyxos FT EV Points on the Pyxos FT network. You can use the EV Pilot as a standalone controller for the Pyxos FT network, or you can connect the EV Pilot to another Pyxos FT network using a LONWORKS network connection. You can also connect the EV Pilot to a Windows computer using a USB connection or a LONWORKS network connection.

The Pyxos FT EV Pilot uses the Atmel ARM® AT91SAM7S64 microprocessor as its host microprocessor, which is an ARM7-family microprocessor. You can use this microprocessor, or another microprocessor that meets your application requirements, as the host microprocessor for any Pyxos FT Pilot applications that you develop.

The Pyxos FT EV Pilot also includes an Echelon FT 3150® Smart Transceiver with a ShortStack 2 Micro Server that provides an interface to an optionally connected LONWORKS network.

This section provides additional details for the Pyxos FT EV Pilot evaluation board, including descriptions of the push buttons, light-emitting diodes (LEDs), jumper settings, and connectors.

The descriptions of programmatic behaviors in this section apply only to the Pyxos Network Example firmware; your custom firmware can provide different behaviors.

The Pyxos FT EVK includes schematics for the circuitry of the evaluation boards. You can view the Pyxos FT EVK schematics from the Windows **Start** menu: select **Programs → Echelon Pyxos FT EVK → Board Schematics**, then select a schematic. These installed schematics are not meant to be used as general reference designs; see the *Pyxos FT Chip Data Book* for reference designs that you can use for designing your own Pyxos FT devices.

Key Features

The EV Pilot evaluation board includes the following key features:

- High-performance ARM7 host microprocessor running at 47.9232 MHz
- Optional USB connectivity
- Optional LONWORKS connectivity using the ShortStack Micro Server and FT 3150 Smart Transceiver
- Link power source power supply and filter that supports up to 6 Pyxos Points that each consume 35-100 mA of application current
- Transformer-isolated coupling circuit

Push Buttons and LEDs

The Pyxos FT EV Pilot evaluation board includes a Join button, a Reset button for the Pyxos FT Chip and host microprocessor, a Reset button for the FT 3150 Smart Transceiver, a Service button for the FT 3150 Smart Transceiver, and five application push buttons. The evaluation board also includes eight application

LEDs that illuminate either when an application push button is pressed or when some condition exists that causes the EV Pilot to illuminate them.

This section describes the default behavior of the application push buttons and LEDs for the Pyxos Network Example firmware application that is pre-installed in the EV Pilot ARM7 host processor.

Figure 5 shows the locations of the various push buttons and LEDs for the Pyxos FT EV Pilot evaluation board.

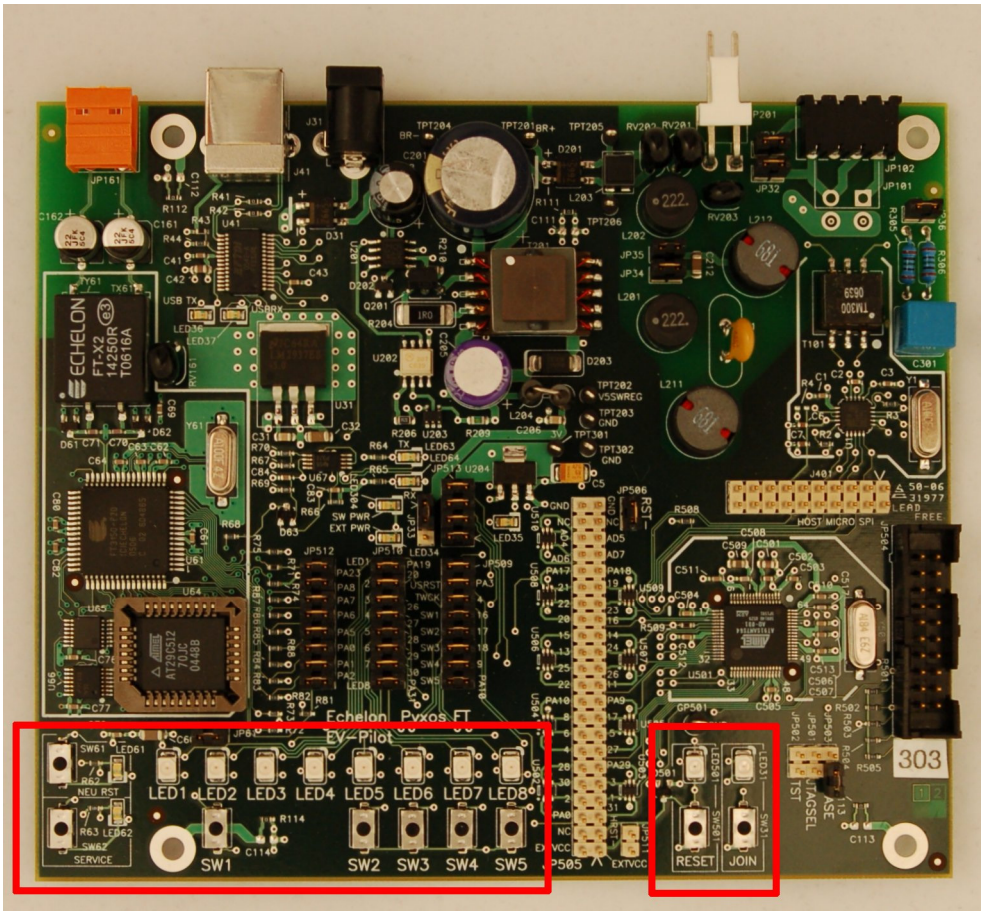


Figure 5. Push Buttons and LEDs for the Pyxos FT EV Pilot

Join and Reset Buttons and LEDs

The Pyxos FT EV Pilot evaluation board includes a Join button (labeled JOIN) and corresponding LED that are not used by the Pyxos FT Network Example application.

The EV Pilot evaluation board also includes a Reset button (labeled RESET) and corresponding LED that you can use to reset all functions on the board, including the ARM7 host microprocessor, the Pyxos FT Chip, and the FT 3150 Smart Transceiver. Restarting the host microprocessor restarts the host firmware application.

Smart Transceiver Reset and Service Buttons

The Pyxos FT EV Pilot evaluation board includes a Reset button (labeled NEU RST) for the FT 3150 Smart Transceiver. You can use this button to reset the connection to a LONWORKS network or test the reset behavior of the LONWORKS functionality of your Pilot application firmware.

The Pyxos FT EV Pilot evaluation board also includes a Service button (labeled SERVICE) that you can use to send a service-pin message from the EV Pilot to the LONWORKS network.

Application Buttons and LEDs

The Pyxos FT EV Pilot evaluation board includes the following push buttons for application use:

SW1	Tests dry-contact functions.
SW2	Provides digital input for a custom application. This button is not used by the Pyxos Network Example application; however, the Pyxos Network Example HMI application program displays the current state of this push button.
SW3	Clears persistent data associated with the Pyxos FT EV-Actuator Point, and causes the EV Pilot to re-register the EV-Actuator Point.
SW4	Clears persistent data associated with the Pyxos FT EV-Sensor Point, and causes the EV Pilot to re-register the EV-Sensor Point.
SW5	Clears persistent data associated with the Pyxos FT EV-Nano Point, and places the EV Pilot in manual registration mode. While the EV-Nano Point is in manual registration mode, **LED8** blinks; to cancel manual registration mode without registering the Point, press **SW5** a second time (**LED8** turns off).

The example firmware for the Pyxos FT EV Pilot uses the following LEDs:

LED1	Indicates network status and network security. • This LED is on when all three EV Points are connected and the network is functioning properly. • This LED blinks for 10 seconds to indicate a security alarm when a Point is added or removed from the network. • This LED is off if any one of the three EV Points is disconnected from the network or is unconfigured.

LED2	Indicates whether the dry-contact push button on the EV Pilot, EV-Actuator Point, or EV-Nano Point is pressed. This LED remains on while a dry-contact push button is pressed.
LED3	Indicates an Over Temperature alarm status. This LED is on when the temperature rises above a threshold value. The default value is 30 ºC (86 ºF). You can configure the threshold value using the Pyxos Network Example HMI application program.
LED4	Indicates a Low Light Level alarm status. This LED is on when the light level falls below a threshold value. The default value is 50 lux. You can configure the threshold value using the Pyxos Network Example HMI application program.
LED5	Indicates that the Performance Demo is running. This LED is on when the demo is active. See *Running the Performance Demo* on page 81 for more information about the Performance Demo.
LED6	Indicates the status of the Pyxos FT EV-Actuator Point. This LED is on when the EV-Actuator Point is connected and online.
LED7	Indicates the status of the Pyxos FT EV-Sensor Point. This LED is on when the EV-Sensor Point is connected and online.
LED8	Indicates the status of the Pyxos FT EV-Nano Point. This LED is on when the EV-Nano Point is connected and online. This LED blinks when the Pilot is in manual registration mode.

Jumper Settings for Controlling Onboard I/O

The Pyxos FT EV Pilot evaluation board contains 12 sets of jumpers, as shown in Figure 6.

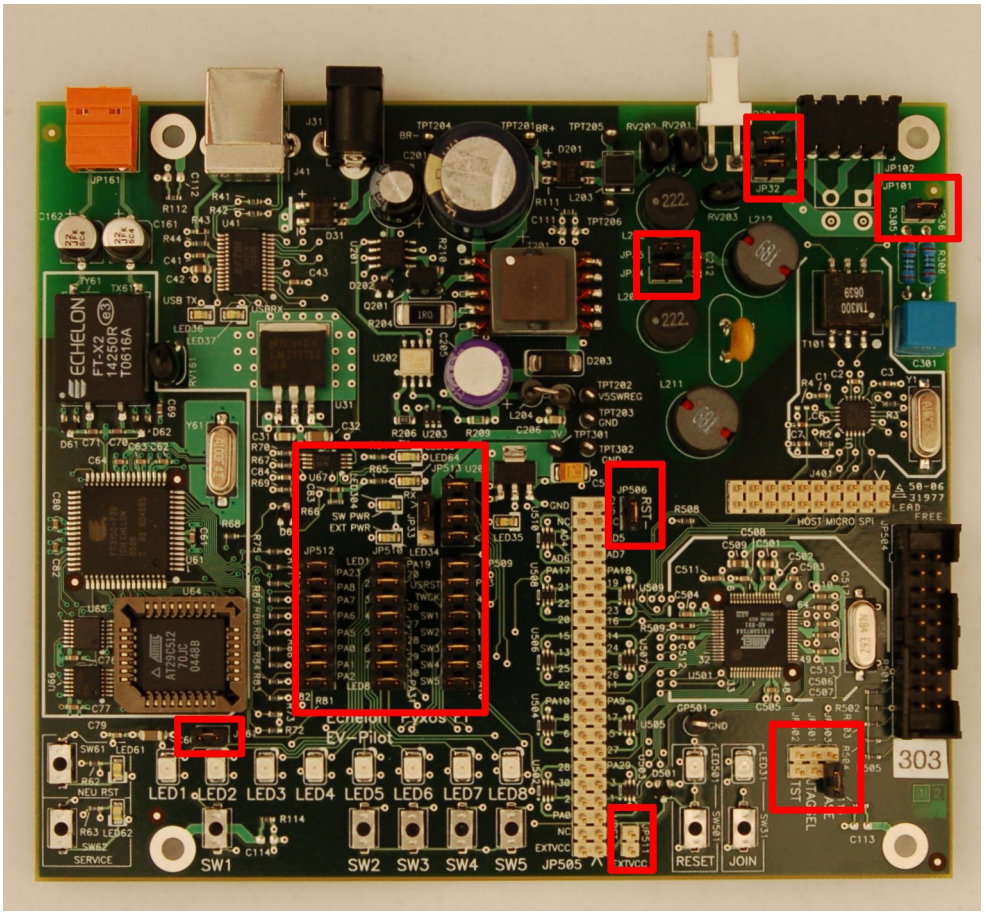
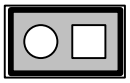
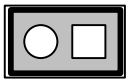
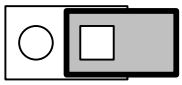
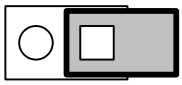
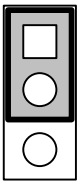
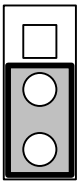
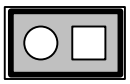
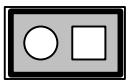
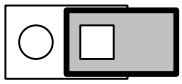
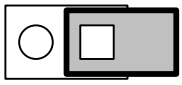
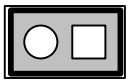
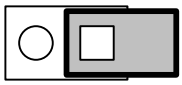
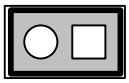
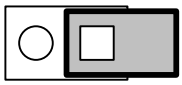
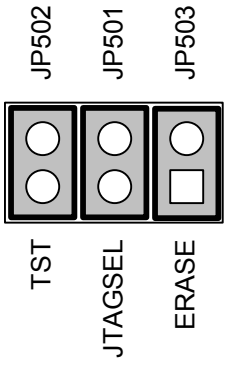


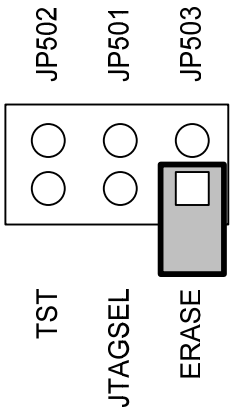
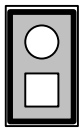
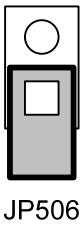
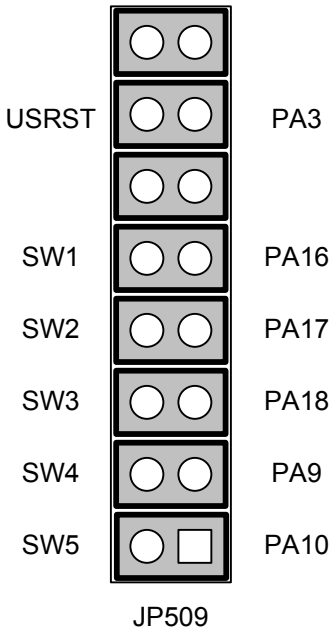
Figure 6. Pyxos FT EV Pilot Evaluation Board Jumper Settings

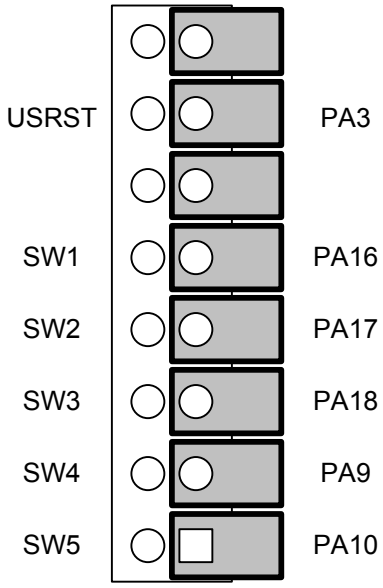
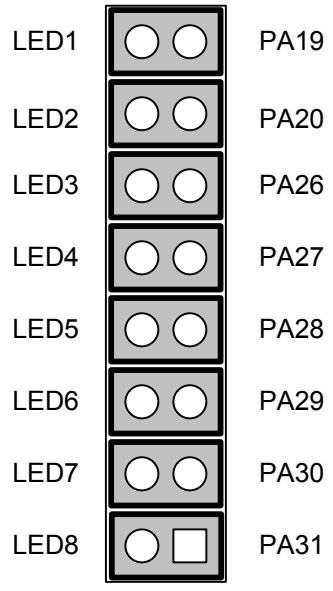
Table 2 describes the Pilot evaluation board jumpers.

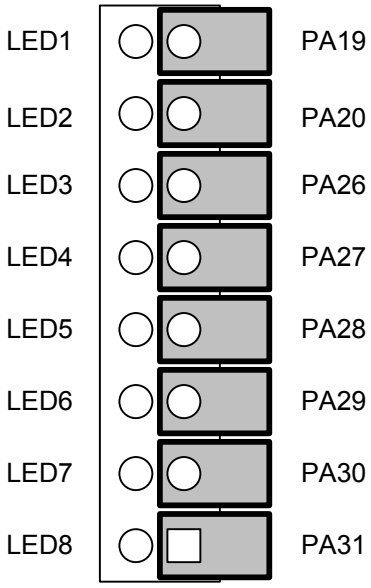
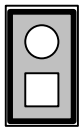
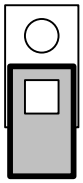
Table 2. Pyxos FT EV Pilot Jumpers

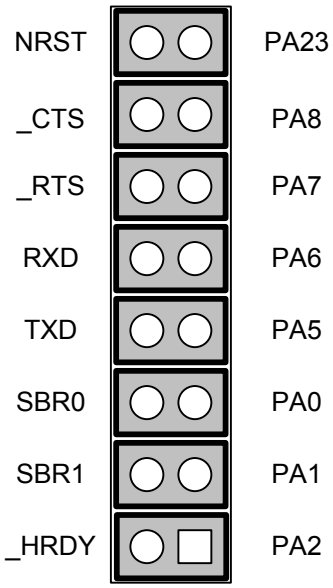
Function	Jumper	Description
Network Power Connect Enable (JP31 and JP32)	 JP31  JP32	Connects the filtered AC power input to the Pyxos FT network. Be sure to mount or dismount these two jumpers at the same time. When both of these jumpers are mounted, you must also mount jumper **JP33** in the upper position to select the onboard 5 VDC switching power supply. This is the factory shipped default setting.
	 JP31  JP32	Disconnects the filtered AC power input from the Pyxos FT network. Be sure to mount or dismount these two jumpers at the same time.
Power Supply Input Selector (JP33)	 JP33	Selects the onboard 5 VDC switching power supply. This setting is required for the EV Pilot to use link power. This is the factory shipped default setting.
	 JP33	Selects the external power supply connected to connector **J31**.
Power Supply Connect Enable (JP34 and JP35)	 JP34  JP35	Connects the transformer side of the onboard switching power supply to the 24 VAC network connector **JP201**. Be sure to mount or dismount these two jumpers at the same time. This is the factory shipped default setting.

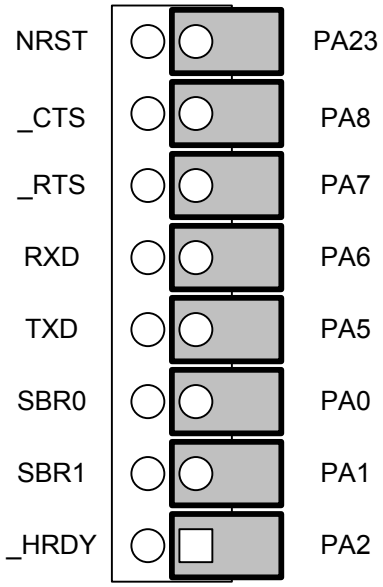
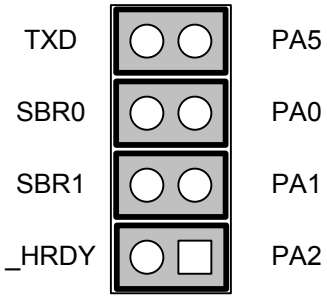
Function	Jumper	Description
	 JP34  JP35	Disconnects the transformer side of the onboard switching power supply from the 24 VAC network connector **JP201**. Be sure to mount or dismount these two jumpers at the same time.
Network Terminator Connect Enable **(JP36)**	 JP36	Connects the onboard network terminator to the Pyxos FT network connector **JP102**. This is the factory shipped default setting.
	 JP36	Disconnects the onboard network terminator from the Pyxos FT network connector **JP102**.
LONWORKS Enable **(JP61)**	 JP61	Connects the onboard 5 VDC power supply to the LONWORKS-connectivity section of the Evaluation Board (including the FT 3150 Smart Transceiver chip). This is the factory shipped default setting.
	 JP61	Disconnects the onboard 5 VDC power supply from the LONWORKS-connectivity section of the Evaluation Board (including the FT 3150 Smart Transceiver chip).
Host Processor Function Enable **(JP501, JP502, JP503)**	 JP502 JP501 JP503 TST JTAGSEL ERASE	Allows external connections to the following ARM7 host microprocessor functions: • The ERASE pin to reinitialize flash memory content and certain non-volatile memory (NVM) bits. • The JTAGSEL pin to select the JTAG boundary scan when asserted at a high level. • The TST pin for manufacturing test, fast programming mode, or SAM-BA Boot Recovery of the AT91SAM7S Series when asserted high.

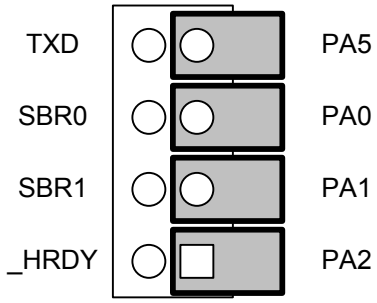
Function	Jumper	Description
		Prevents external connections to the ARM7 host microprocessor functions. This is the factory shipped default setting.
Reset Line Enable (JP506)		Connects the NRST~ pin of the ARM7 host microprocessor to the RST~ pin of the Pyxos FT Chip. This is the factory shipped default setting.
		Disconnects the NRST~ pin of the ARM7 host microprocessor from the RST~ pin of the Pyxos FT Chip.
Example I/O Enable - Switches (JP509)		Connects the ARM7 host microprocessor to the EV Pilot onboard push buttons (**SW1** through **SW5**), and connects the RESET pin of the ARM7 host microprocessor to the reset function of the USB interface (USRST). The PA numbers refer to the ARM7 programmed I/O (PIO) lines. This is the factory shipped default setting.

Function	Jumper	Description
		Disconnects the ARM7 host microprocessor from the EV Pilot onboard push buttons (**SW1** through **SW5**), and disconnects the RESET pin of the ARM7 host microprocessor from the reset function of the USB interface (USRST). The PA numbers refer to the ARM7 PIO lines.
Example I/O Enable - LEDs **(JP510)**		Connects the ARM7 host microprocessor to the EV Pilot onboard LEDs (**LED1** through **LED8**). The PA numbers refer to the ARM7 PIO lines. This is the factory shipped default setting.

Function	Jumper	Description
	 JP510	Disconnects the ARM7 host microprocessor from the EV Pilot onboard LEDs (**LED1** through **LED8**). The PA numbers refer to the ARM7 PIO lines.
Host Microprocessor Header Voltage Enable **(JP511)**	 JP511	Connects the EV Pilot 3.3 V_{DD3} power supply to the ARM7 host microprocessor I/O header **JP505**.
	 JP511	Disconnects the EV Pilot 3.3 V_{DD3} power supply from the ARM7 host microprocessor I/O header **JP505**. This is the factory shipped default setting.

Function	Jumper	Description
ShortStack Micro Server SCI Connect Enable (JP512)	 JP512	Connects the PIO lines of the ARM7 host microprocessor to the Serial Communications Interface (SCI) of the ShortStack Micro Server. • NRST allows the ARM7 host microprocessor to restart the ShortStack Micro Server. • _CTS: Clear. Set by the Micro Server after the host has asserted _RTS. The _CTS output is used to signal that the Micro Server is ready to receive data from the host. • _RTS: Request to Send. Set by the host to signal that it has data to send. The host asserts this signal low and waits for the Micro Server to assert _CTS. • RXD: Receive Data. Used to transfer data from the host to the Micro Server. • TXD: Transmit Data. Used to transfer data from the Micro Server to the host. • SBR0: Serial Bit Rate Bit 0. • SBR1: Serial Bit Rate Bit 1. • _HRDY: Host Ready. The host can use this optional signal to indicate to the Micro Server that it cannot accept any data transfers. The PA numbers refer to the ARM7 PIO lines. This is the factory shipped default setting. See the *ShortStack User's Guide* for more information about the ShortStack Micro Server.

Function	Jumper	Description
	 JP512	Disconnects the PIO lines of the ARM7 host microprocessor from the ShortStack Micro Server. The PA numbers refer to the ARM7 PIO lines. See the *ShortStack User's Guide* for more information about the ShortStack Micro Server.
USB Interface Communications Connect Enable (JP513)	 JP513	Connects the ARM7 host microprocessor to the communications lines of the onboard USB universal asynchronous receiver/transmitter (UART) chip. • CTS1: Clear to Send Control input / Handshake signal. • RTS1: Request To Send Control Output / Handshake signal • TXD1: Transmit Asynchronous Data Output • RXD1: Receive Asynchronous Data Input The PA numbers refer to the ARM7 PIO lines. This is the factory shipped default setting. See the Future Technology Devices International Ltd. Web site (www.ftdichip.com) for more information about the USB UART chip.

Function	Jumper	Description
		Disconnects the ARM7 host microprocessor from the communications lines of the onboard USB UART chip. The PA numbers refer to the ARM7 PIO lines. See the Future Technology Devices International Ltd. Web site (www.ftdichip.com) for more information about the USB UART chip.

Connectors

The Pyxos FT EV Pilot evaluation board includes three header connectors, two network connectors, a USB connector, a 24 VDC power connector, and a 24 VAC power connector, as shown in Figure 7. The header connectors provide access to the host processor, including host power and I/O, and provide an interface for debugging or programming the host processor. The network connectors provide access to the Pyxos FT network and to a LONWORKS network. The USB connector allows the EV Pilot to communicate with a PC that is running the Pyxos Network Example HMI application program.

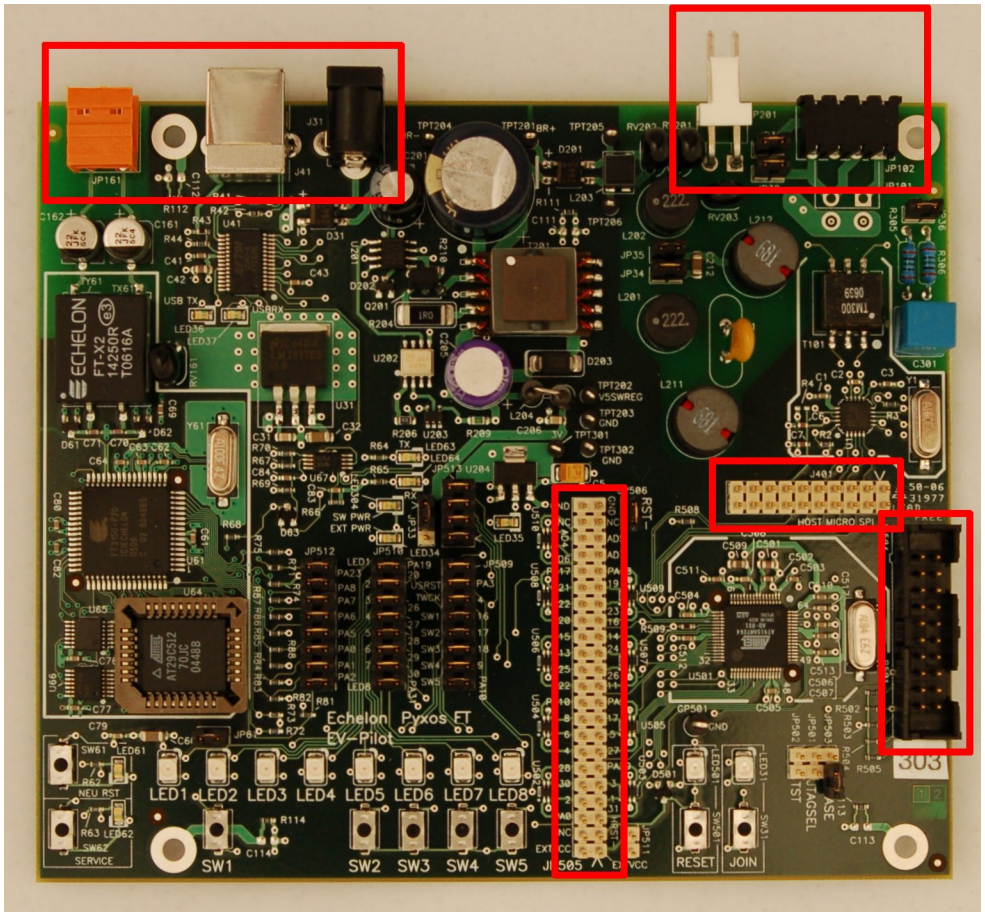


Figure 7. Pyxos FT EV Pilot Evaluation Board Connectors

Header Connector for Accessing Host Power and I/O

The Pyxos FT EV Pilot evaluation board provides a header connector (**JP505**) that provides external access to the ARM7 programmed I/O (PIO) lines, onboard power (+3.3 VDC), and onboard ground.

The connections for this header are shown in **Figure 8**, and match those of the PIO header of the Atmel AT91SAM7S-EK evaluation board for accessing the ARM7 host microprocessor.

GND	44	43	GND
NC	42	41	NC
AD4	40	39	AD5
AD6	38	37	AD7
PA17	36	35	PA18
PA21	34	33	PA19
PA22	32	31	PA23
PA20	30	29	PA16
PA15	28	27	PA14
PA13	26	25	PA24
PA25	24	23	PA26
PA12	22	21	PA11
PA10	20	19	PA9
PA8	18	17	PA7
PA6	16	15	PA5
PA4	14	13	PA27
PA28	12	11	PA29
PA30	10	9	PA3
PA2	8	7	PA1
PA0	6	5	PA31
NC	4	3	HRST
EXTVDD3	2	1	EXTVDD3

Figure 8. Host Power and I/O Header Connector

Header Connector for Accessing Host SPI

The Pyxos FT EV Pilot evaluation board provides a header connector (**J401**) that provides external access to the ARM7 Serial Peripheral Interface (SPI) lines. The connections for this header are shown in **Figure 9**.

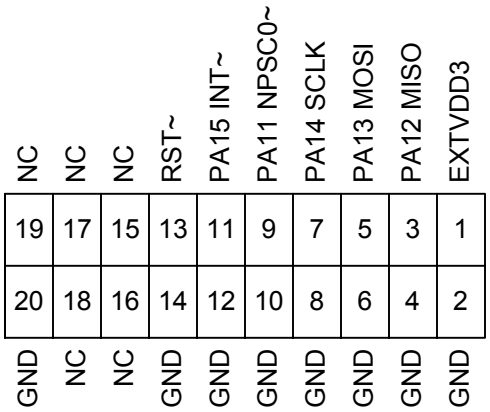


Figure 9. Host SPI Header Connector

Header Connector for Remote Programming and Debugging

The Pyxos FT EV Pilot evaluation board provides a header connector (**JP504**) that provides external access to the ARM7 host processor memory. This header complies with the IEEE Standard Test Access Port and Boundary-Scan Architecture (IEEE 1149.1-1990) of the Joint Test Action Group (JTAG). You can use this header for remote debugging of the ARM7 application program, for an in-circuit emulator (ICE), or to load programs into the ARM7 memory.

Pyxos FT Network Connector

The Pyxos FT EV Pilot evaluation board provides a connector (**JP102**) that provides access to the Pyxos FT network. The EV Pilot uses this connector to communicate with Pyxos FT EV Points.

LONWORKS Network Connector

The Pyxos FT EV Pilot evaluation board provides a LONWORKS TP/FT-10 interface connector (**JP161**) that provides access to a LONWORKS network. Use of this network connector is optional.

You can use this network connector to:

- Connect the EV Pilot to a LONWORKS network, either to connect multiple Pyxos FT networks together or to share Pyxos FT data with devices on a LONWORKS network.
- Connect the EV Pilot to a LONWORKS network interface that is connected to a Windows computer so that you can run the Pyxos Network Example HMI application program.

USB Connector

The Pyxos FT EV Pilot evaluation board provides a USB Type B socket connector (**J41**) that provides a USB connection to a Windows computer. Use of this connector is optional.

You can use this network connector to connect the EV Pilot to a Windows computer so that you can run the Pyxos Network Example HMI application program.

DC Power Connector

The Pyxos FT EV Pilot evaluation board provides a 24 VDC power connector (**J31**) that supplies optional DC power to the EV Pilot evaluation board.

AC Power Connector

The Pyxos FT EV Pilot evaluation board provides a 24 VAC power connector (**JP201**) that supplies AC power to the EV Pilot evaluation board.

Pyxos FT EV-Actuator Point Evaluation Board

The Pyxos FT EV-Actuator Point is a Pyxos Point that serves as a hosted analog and digital actuator for the Pyxos FT EVK network. The EV-Actuator Point includes a variable attenuator that you can use with the Performance Demo to demonstrate network determinism.

The Pyxos FT EV-Actuator Point uses the Atmel ARM AT91SAM7S64 microprocessor as its host processor, which is an ARM7-family microprocessor. This microprocessor includes a robust feature set, and contrasts with the reduced feature set of the Atmel ATtiny13 host microprocessor that is used for the Pyxos FT EV-Sensor Point. You can use this microprocessor, or another microprocessor that meets your application requirements, as the host microprocessor for any Pyxos Points that you develop.

This section provides additional details for the Pyxos FT EV-Actuator Point evaluation board, including descriptions of the push buttons, LEDs, jumper settings, and connectors.

The descriptions of programmatic behaviors in this section apply only to the Pyxos Network Example firmware; your custom firmware can provide different behaviors.

Key Features

The EV-Actuator Point Evaluation Board includes the following key features:

- High-performance ARM7 host microprocessor running at 47.9232 MHz
- Digital-to-analog converter (DAC) and LED output
- Link powered using a switching power supply that delivers up to 100 mA application current
- Transformer-isolated coupling circuit

Push Buttons and LEDs

The Pyxos FT EV-Actuator Point evaluation board includes a Join button, a Reset button, and five application push buttons. The evaluation board also includes eight application LEDs that illuminate either when a button is pressed or when some condition exists.

This section describes the default behavior of the application push buttons and LEDs for the Pyxos Network Example firmware that is pre-installed in the EV-Actuator Point ARM7 host processor.

Figure 10 shows the locations of the various push buttons and LEDs for the Pyxos FT EV-Actuator Point evaluation board.

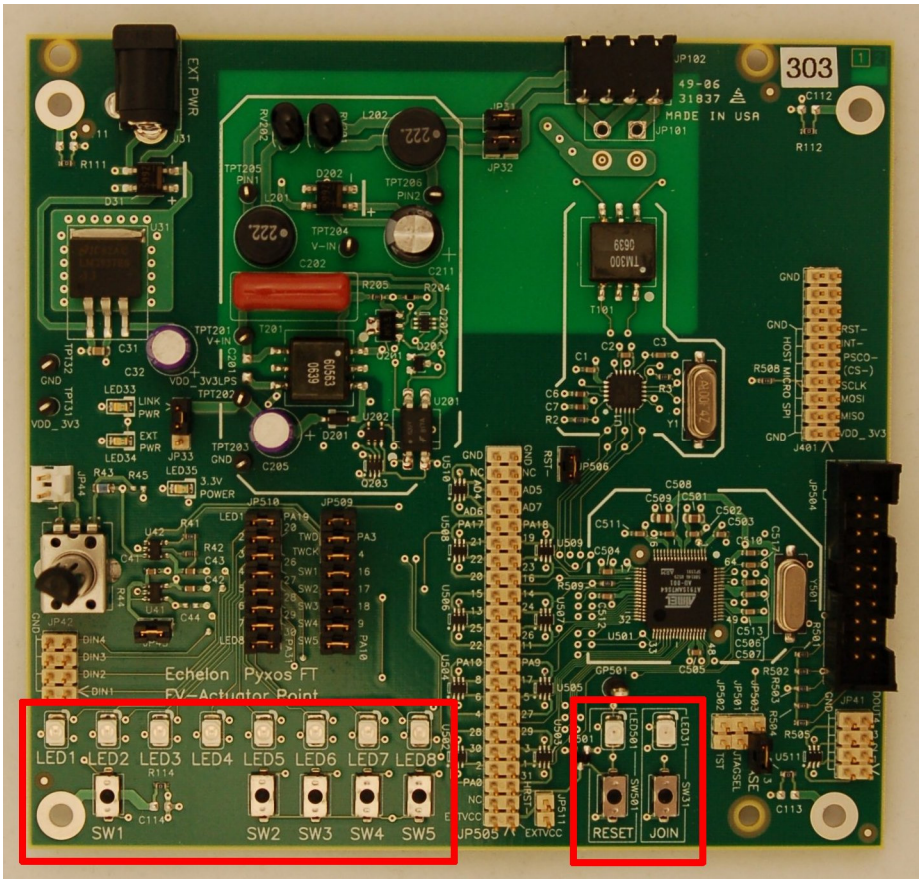


Figure 10. Push Buttons and LEDs for the Pyxos FT EV-Actuator Point

Join and Reset Buttons and LEDs

The Pyxos FT EV-Actuator Point evaluation board includes a Join button (labeled JOIN) and corresponding LED that are not used by the Pyxos Network Example application. This button and LED are included on the EV-Actuator Point evaluation board so that you can perform manual registration for this Point if your custom firmware application should require it.

The EV-Actuator Point evaluation board also includes a Reset button (labeled RESET) and corresponding LED that you can use to reset the ARM7 host microprocessor and Pyxos FT Chip, and restart the host firmware.

Application Buttons and LEDs

The Pyxos FT EV-Actuator Point evaluation board includes the following push buttons for application use:

SW1	Tests dry-contact functions.
SW2	Provides digital input for a custom application. This button is not used by the Pyxos Network Example application; however, the Pyxos Network Example HMI application program displays the current state of this push button.
SW3	Provides digital input for a custom application. This button is not used by the Pyxos Network Example application; however, the Pyxos Network Example HMI application program displays the current state of this push button.
SW4	Provides digital input for a custom application. This button is not used by the Pyxos Network Example application; however, the Pyxos Network Example HMI application program displays the current state of this push button.
SW5	Provides digital input for a custom application. This button is not used by the Pyxos Network Example application; however, the Pyxos Network Example HMI application program displays the current state of this push button. When you connect the Pyxos Network Example HMI application program to the EV Pilot through a LONWORKS network connection and press **SW5**, **LED8** illuminates on the EV-Actuator Point Evaluation Board, and the Network Example HMI shows **LED8** as on.

The example firmware application for the Pyxos FT EV-Actuator Point uses the following LEDs:

LED1	Indicates the status of the Pyxos FT EV-Actuator Point. This LED is on when the EV-Actuator Point is connected and correctly configured.
LED2	Indicates whether the dry-contact push button on the EV Pilot, EV-Actuator Point, or EV-Nano Point is pressed. This LED remains on while a dry-contact push button is pressed.
LED3	Indicates an Over Temperature alarm status. This LED is on when the temperature rises above a threshold value. The default value is 30 ºC (86 ºF). You can configure the threshold value using the Pyxos Network Example HMI application program.

LED4	Indicates a Low Light Level alarm status. This LED is on when the light level falls below a threshold value. The default value is 50 lux. You can configure the threshold value using the Pyxos Network Example HMI application program.
LED5	Indicates that the Performance Demo is running. This LED is on when the demo is active. See *Running the Performance Demo* on page 81 for more information about the Performance Demo.
LED6	Indicates the presence of a digital output. This LED is on when one or more of the digital outputs are active.
LED7	Indicates status from a custom application. This LED is not used by the Pyxos Network Example application.
LED8	Indicates status from a custom application. This LED is not used by the Pyxos Network Example application. When you connect the Pyxos Network Example HMI application program to the EV Pilot through a LONWORKS network connection and press **SW5**, **LED8** illuminates on the EV-Actuator Point Evaluation Board, and the Network Example HMI shows **LED8** as on.

Jumper Settings for Controlling Onboard I/O

The Pyxos FT EV-Actuator Point evaluation board contains eight sets of jumpers, as shown in Figure 11.

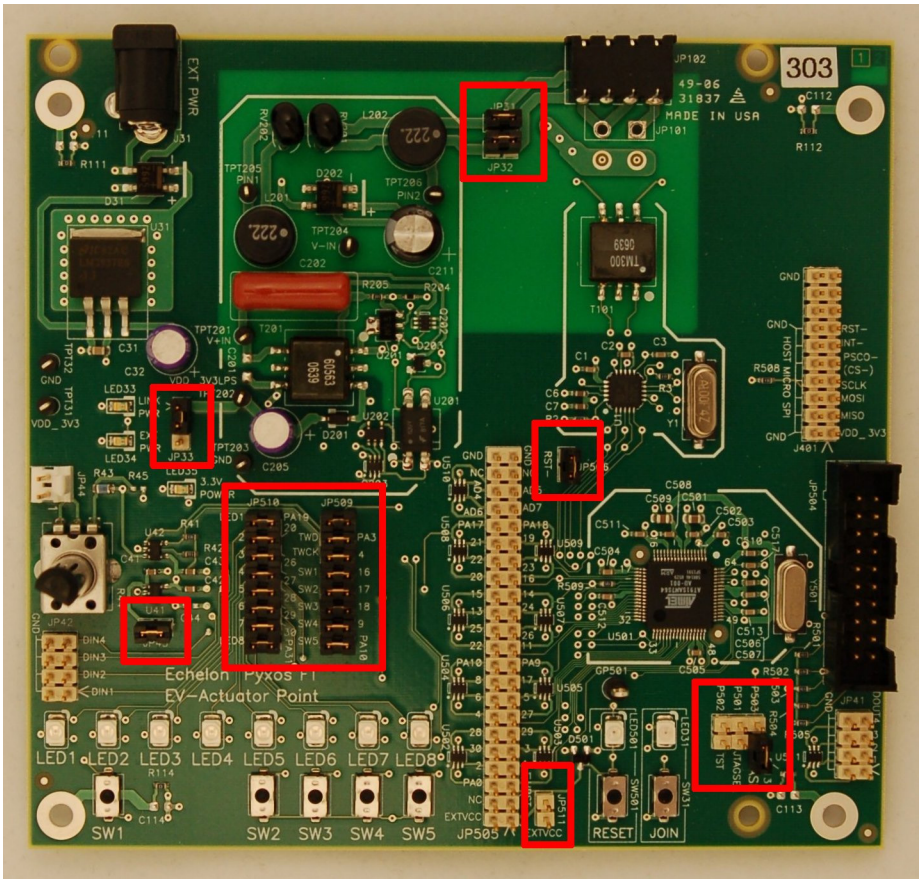
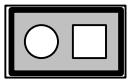
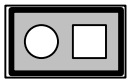
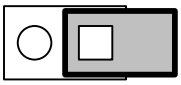
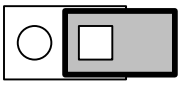
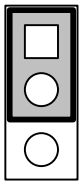
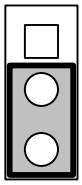
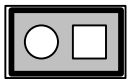
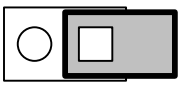
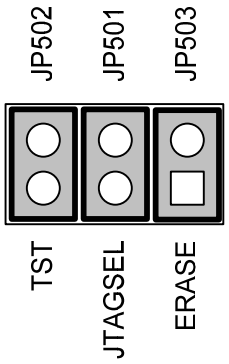
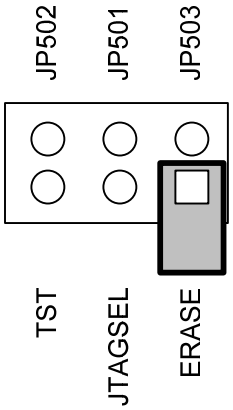
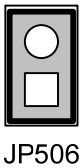
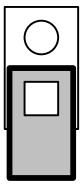


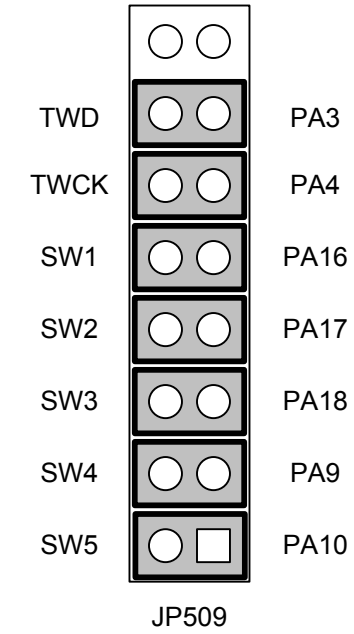
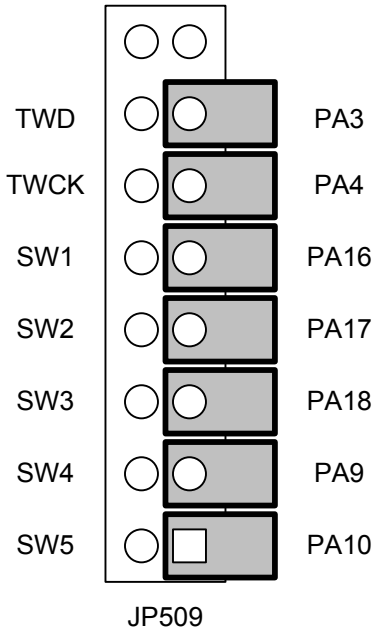
Figure 11. Pyxos FT EV-Actuator Point Evaluation Board Jumper Settings

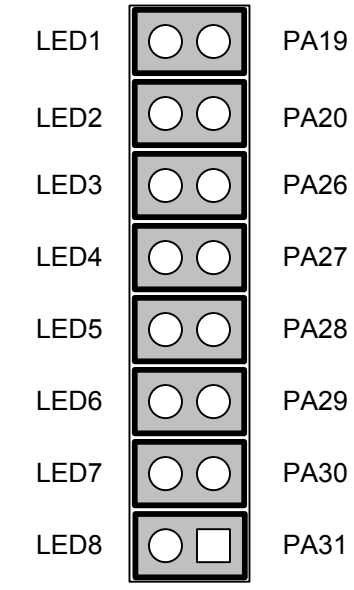
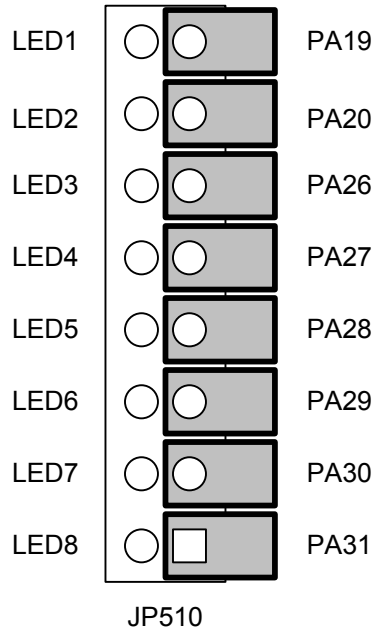
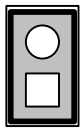
Table 3 describes the EV-Actuator Point jumpers.

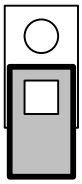
Table 3. Pyxos FT EV-Actuator Point Jumpers

Function	Jumper	Description
Network Power Connect Enable (JP31 and JP32)	 JP31  JP32	Connects the EV-Actuator point to network link power. Be sure to mount or dismount these two jumpers at the same time. When both of these jumpers are mounted, you must also mount jumper **JP33** in the upper position to select the onboard 5 VDC switching power supply. This is the factory shipped default setting.
	 JP31  JP32	Disconnects the EV-Actuator Point from network link power. Be sure to mount or dismount these two jumpers at the same time.
Power Supply Input Selector (JP33)	 JP33	Selects the onboard 5 VDC switching power supply. This setting is required for the EV-Actuator Point to use link power. This is the factory shipped default setting.
	 JP33	Selects the external power supply connected to connector **J31**.
Analog Output Voltage Enable (JP43)	 JP43	Connects the EV-Actuator Point 3.3 V_{DD3} power supply to the analog output header (**JP44**). This is the factory shipped default setting.
	 JP43	Disconnects the EV-Actuator Point 3.3 V_{DD3} power supply from the analog output header (**JP44**), and allows you to use the analog output header for other purposes.

Function	Jumper	Description
Host Processor Function Enable (JP501, JP502, JP503)		Allows external connections to the following ARM7 host microprocessor functions: • The ERASE pin to reinitialize flash memory content and certain non-volatile memory (NVM) bits. • The JTAGSEL pin to select the JTAG boundary scan when asserted at a high level. • The TST pin for manufacturing test, fast programming mode, or SAM-BA Boot Recovery of the AT91SAM7S Series when asserted high.
		Prevents external connections to the ARM7 host microprocessor functions. This is the factory shipped default setting.
Reset Line Enable (JP506)		Connects the NRST~ pin of the ARM7 host microprocessor to the RST~ pin of the Pyxos FT Chip. This is the factory shipped default setting.
		Disconnects the NRST~ pin of the ARM7 host microprocessor from the RST~ pin of the Pyxos FT Chip.

Function	Jumper	Description
Example I/O Enable - Switches (JP509)		Connects the ARM7 host microprocessor to the EV-Actuator Point onboard push buttons (**SW1** through **SW5**). Also connects the ARM7 host microprocessor two-wire interface serial data (TWD) and two-wire interface serial clock (TWCK) functions. The PA numbers refer to the ARM7 PIO lines. This is the factory shipped default setting.
		Disconnects the ARM7 host microprocessor from the EV-Actuator Point onboard push buttons (**SW1** through **SW5**), and disconnects the ARM7 host microprocessor two-wire interface serial data (TWD) and two-wire interface serial clock (TWCK) functions. The PA numbers refer to the ARM7 PIO lines.

Function	Jumper	Description
Example I/O Enable - LEDs (JP510)		Connects the ARM7 host microprocessor to the EV-Actuator Point onboard LEDs (**LED1** through **LED8**). The PA numbers refer to the ARM7 PIO lines. This is the factory shipped default setting.
		Disconnects the ARM7 host microprocessor to the EV-Actuator Point onboard LEDs (**LED1** through **LED8**). The PA numbers refer to the ARM7 PIO lines.
Host Microprocessor Header Voltage Enable (JP511)		Connects the EV-Actuator Point 3.3 V_{DD3} power supply to the ARM7 host microprocessor I/O header (**JP505**).

Function	Jumper	Description
	 JP511	Disconnects the EV-Actuator Point 3.3 V_{DD3} power supply from the ARM7 host microprocessor I/O header (**JP505**). This is the factory shipped default setting.

Connectors

The Pyxos FT EV-Actuator Point evaluation board includes five header connectors, a network connector, and a 24 VDC power connector, as shown in Figure 12. The header connectors provide access to the ARM7 host processor, including host power and I/O, provide access to analog and digital I/O, and provide an interface for debugging or programming the host processor. The network connector provides access to the Pyxos FT network.

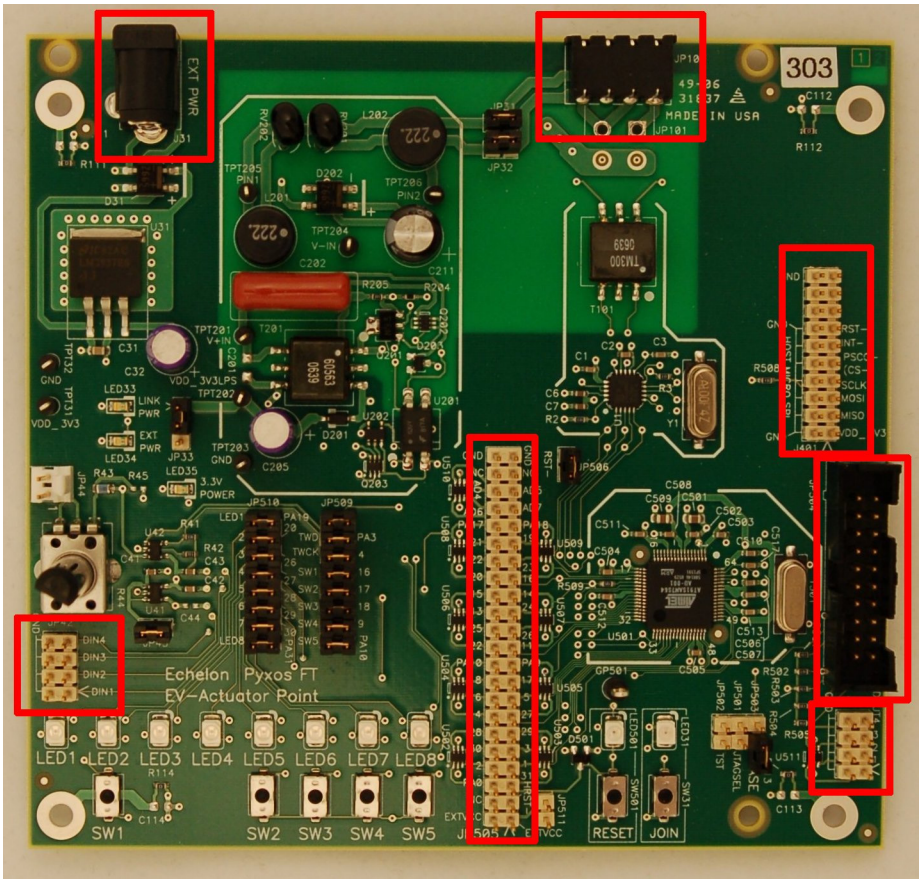


Figure 12. Pyxos FT EV-Actuator Point Evaluation Board Connectors

Header Connector for Accessing Host Power and I/O

The Pyxos FT EV-Actuator Point evaluation board provides a header connector (**JP505**) that provides external access to the ARM7 programmed I/O (PIO) lines, onboard power (+3.3 VDC), and onboard ground.

The connections for this header are shown in **Figure 13**, and match those of the PIO header of the Atmel AT91SAM7S-EK evaluation board for accessing the ARM7 host microprocessor.

GND	44	43	GND
NC	42	41	NC
AD4	40	39	AD5
AD6	38	37	AD7
PA17	36	35	PA18
PA21	34	33	PA19
PA22	32	31	PA23
PA20	30	29	PA16
PA15	28	27	PA14
PA13	26	25	PA24
PA25	24	23	PA26
PA12	22	21	PA11
PA10	20	19	PA9
PA8	18	17	PA7
PA6	16	15	PA5
PA4	14	13	PA27
PA28	12	11	PA29
PA30	10	9	PA3
PA2	8	7	PA1
PA0	6	5	PA31
NC	4	3	HRST
EXTVDD3	2	1	EXTVDD3

Figure 13. Host Power and I/O Header Connector

Header Connector for Accessing Host SPI

The Pyxos FT EV-Actuator Point evaluation board provides a header connector (**J401**) that provides external access to the ARM7 Serial Peripheral Interface (SPI) lines. The connections for this header are shown in **Figure 14**.

GND	20	19	NC
NC	18	17	NC
NC	16	15	NC
GND	14	13	RST~
GND	12	11	PA15 INT~
GND	10	9	PA11 NPSC0~
GND	8	7	PA14 SCLK
GND	6	5	PA13 MOSI
GND	4	3	PA12 MISO
GND	2	1	EXTVDD3

Figure 14. Host SPI Header Connector

Header Connector for Analog Output

The Pyxos FT EV-Actuator Point evaluation board provides a header connector (**JP44**) that provides external access to the onboard digital-to-analog converter (DAC) output and to the attenuator circuit. The analog output ranges from 0 to +3.0 VDC.

You can connect the analog output from the EV-Actuator Point to the analog input on the EV-Sensor Point to allow the Pyxos Network Example HMI application program to run the Performance Demo.

Header Connectors for Accessing Digital I/O

The Pyxos FT EV-Actuator Point evaluation board provides two header connectors that provide external access to the onboard digital I/O. One header (**JP41**) provides access to the four digital outputs. The other header (**JP42**) provides access to the four digital inputs. These headers are shown in **Figure 15**.

You can use either the digital input header or the **SW2** through **SW5** push buttons to provide digital input to the EV-Actuator Point.

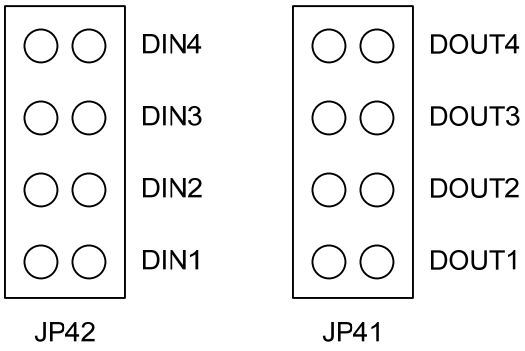


Figure 15. Digital I/O Header Connectors

Header Connector for Remote Programming and Debugging

The Pyxos FT EV-Actuator Point evaluation board provides a header connector (**JP504**) that provides external access to the ARM7 host processor memory. This header complies with the IEEE Standard Test Access Port and Boundary-Scan Architecture (IEEE 1149.1-1990) of the Joint Test Action Group (JTAG). You can use this header for remote debugging of the ARM7 application program, for an in-circuit emulator (ICE), or to load programs into the ARM7 memory.

Pyxos FT Network Connector

The Pyxos FT EV-Actuator Point evaluation board provides a connector (**JP102**) that provides access to the Pyxos FT network. The EV-Actuator Point uses this connector to communicate with Pyxos FT EV Pilot.

Power Connector

The Pyxos FT EV-Actuator Point evaluation board provides a 24 VDC power connector (**J31**) that supplies optional local power to the EV-Actuator Point. Do not use this connector if the EV-Actuator Point evaluation board is powered by link power.

Pyxos FT EV-Sensor Point Evaluation Board

The Pyxos FT EV-Sensor Point is a Pyxos Point that serves as a hosted multi-sensor for the Pyxos FT EVK network. The sensors measure temperature, light level, and DC voltage.

The Pyxos FT EV-Sensor Point uses the Atmel AVR® ATtiny13 microprocessor as its host processor. This microprocessor has a reduced feature set compared with the feature set of the ARM7 host microprocessor that is used for the Pyxos EV-Actuator Point, but is a lower-cost solution. You can use this microprocessor, or another microprocessor that meets your application requirements, as the host microprocessor for any Pyxos Point applications that you develop.

This section provides additional details for the Pyxos FT EV-Sensor Point evaluation board, including descriptions of the push buttons, LEDs, jumper settings, and connectors.

The descriptions of programmatic behaviors in this section apply only to the Pyxos Network Example firmware; your custom firmware can provide different behaviors.

Key Features

The EV-Sensor Point Evaluation Board includes the following key features:

- Low-cost AVR host microprocessor running at 10 MHz
- Three sensor inputs: light-level sensor, temperature sensor, and analog-to-digital converter (ADC) input
- Small size (9.5 cm x 7.75 cm)

- Link powered using a switching power supply that delivers up to 100 mA application current
- Transformer-isolated coupling circuit

Join and Reset Buttons and LEDs

The Pyxos FT EV-Sensor Point evaluation board includes a Join button (labeled JOIN) and corresponding LED that are not used by the Pyxos Network Example application. This button and LED are included on the EV-Sensor Point evaluation board so that you can perform manual registration for this Point if your custom firmware application should require it.

The EV-Sensor Point evaluation board also includes a Reset button (labeled RESET) and corresponding LED that you can use to reset the AVR host microprocessor and Pyxos FT Chip, and restart the host firmware application program.

Figure 16 shows the location of the buttons and LEDs for the Pyxos FT EV-Sensor Point evaluation board.

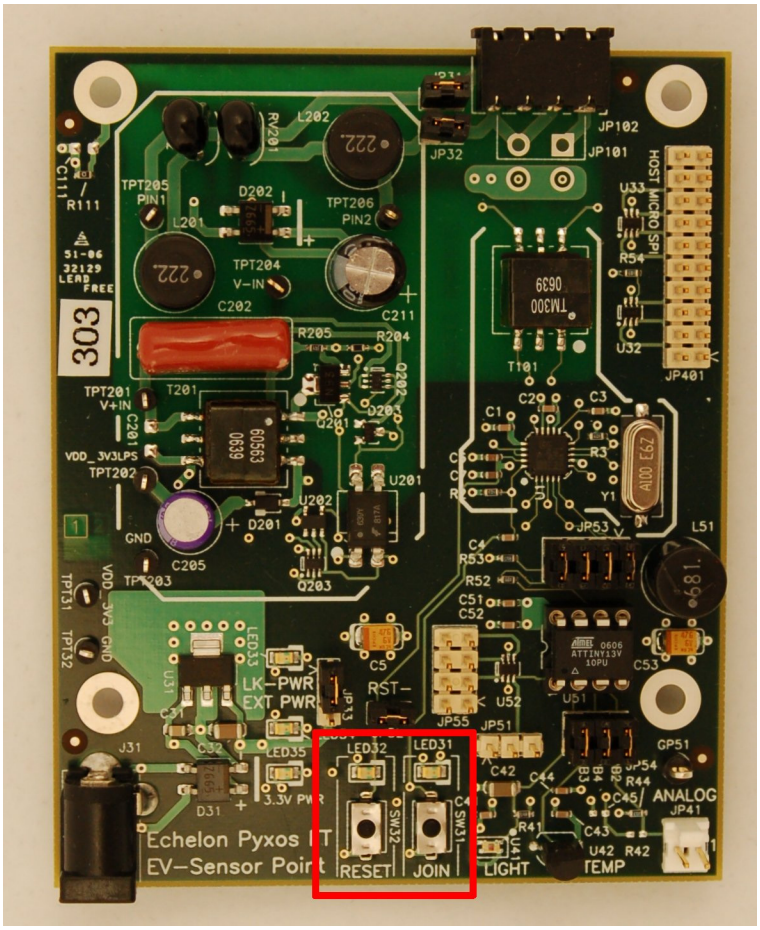


Figure 16. Push Buttons and LEDs for the Pyxos FT EV-Sensor Point

Sensors

The EV-Sensor Point evaluation board includes two onboard sensors, as shown in **Figure 17**:

- An ambient light photo sensor
- A low-voltage temperature sensor

The EV-Sensor Point evaluation board also includes a header (**JP41**) for analog voltage input that is used as an analog voltage sensor.

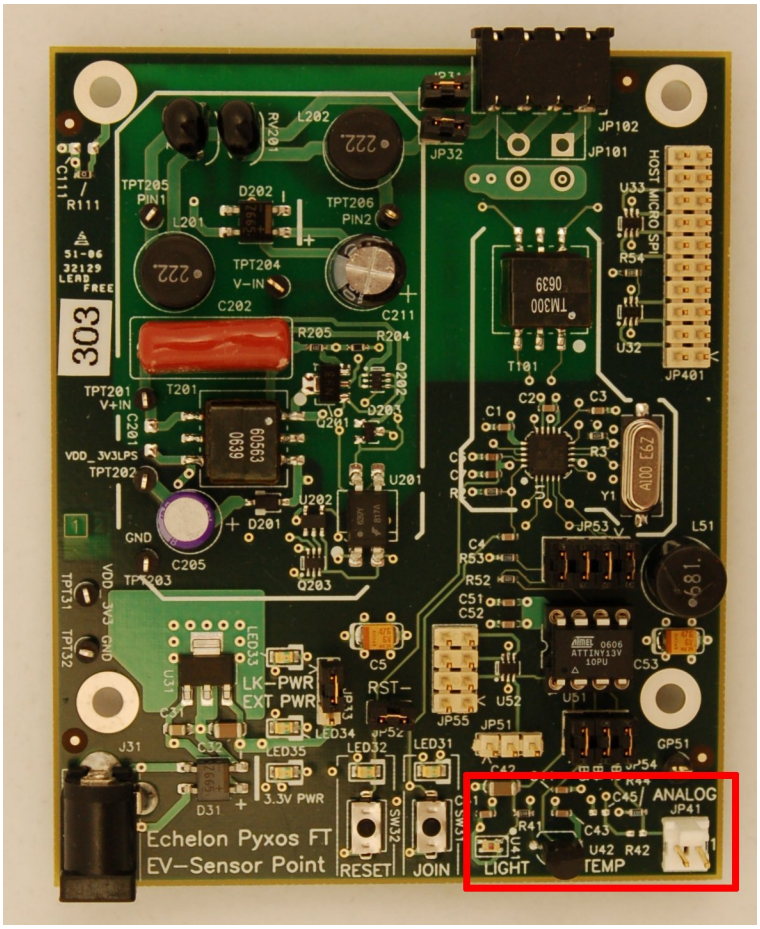


Figure 17. Onboard Sensors for the Pyxos FT EV-Sensor Point

Neither the light sensor nor the temperature sensor has been accurately calibrated because accurate measurements of light levels and temperature are not necessary for the Pyxos Network Example.

The following characteristics define a subset of the device specifications for the light photo sensor:

- Light reception cone: ±35°
- Saturation: approximately 500 lux
- The sensor is less sensitive to fluorescent light, and can report values that show from 100% to 300% variance from equivalent incandescent lux values

- Specified operating temperature: –40 °C to +85 °C

The following characteristics define a subset of the device specifications for the temperature sensor:

- Specified operating temperature: –40 °C to +125 °C
- Specified ambient temperature T_A: +25 °C
- Specified accuracy: ±3.0 °C within specified operating temperature

Jumper Settings for Controlling Onboard I/O

The Pyxos FT EV-Sensor Point evaluation board contains five sets of jumpers, as shown in Figure 18.

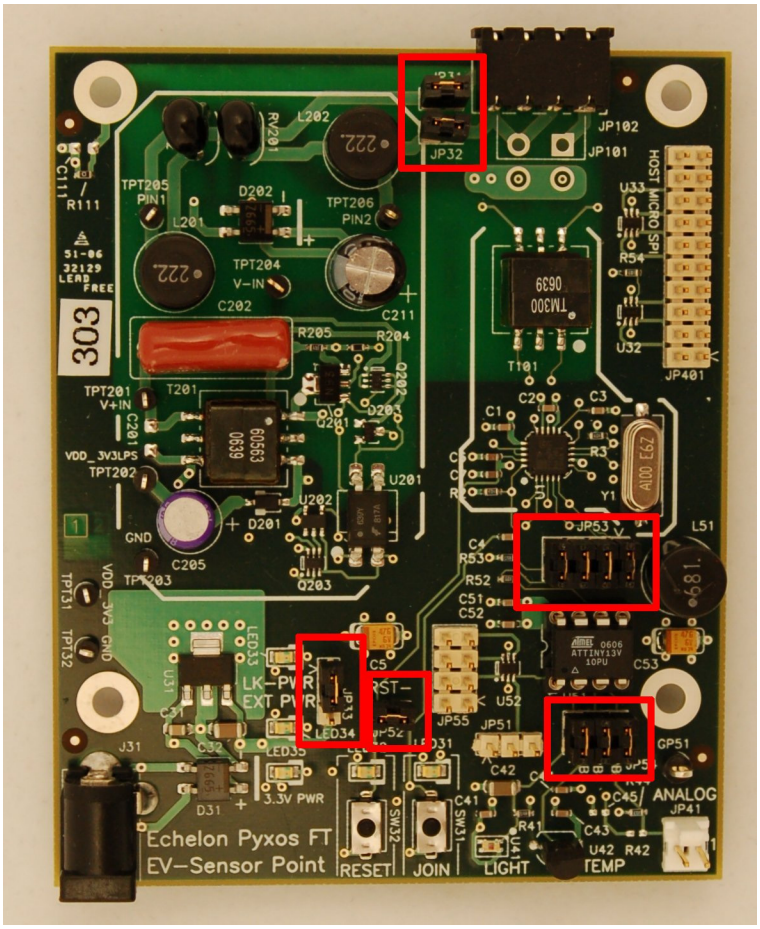
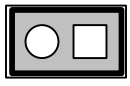
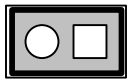
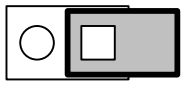
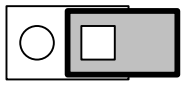
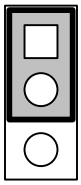
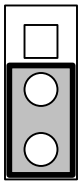
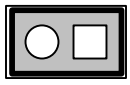
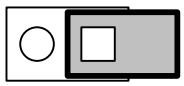
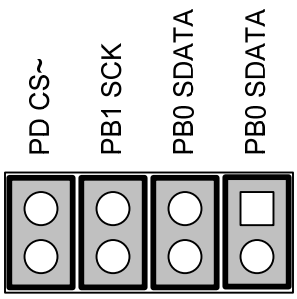
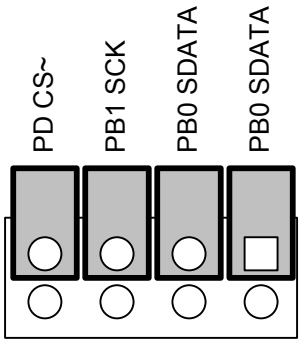
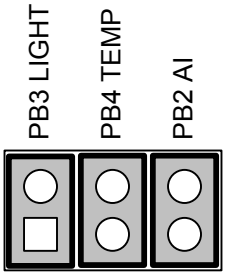


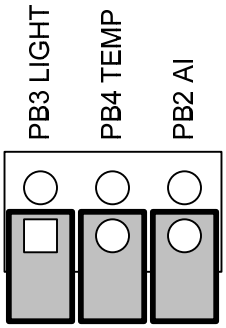
Figure 18. Pyxos FT EV-Sensor Point Evaluation Board Jumper Settings

Table 4 describes the EV-Sensor Point jumpers.

Table 4. Pyxos FT EV-Sensor Point Jumpers

Function	Jumper	Description
Network Power Connect Enable (JP31 and JP32)	 JP31  JP32	Connects the EV-Sensor Point to network link power. Be sure to mount or dismount these two jumpers at the same time. When both of these jumpers are mounted, you must also mount jumper **JP33** in the upper position to select the onboard 5 VDC switching power supply. This is the factory shipped default setting.
	 JP31  JP32	Disconnects the EV-Sensor Point from network link power. Be sure to mount or dismount these two jumpers at the same time.
Power Supply Input Selector (JP33)	 JP33	Selects the onboard 5 VDC switching power supply. This setting is required for the EV-Sensor Point to use link power. This is the factory shipped default setting.
	 JP33	Selects the external power supply connected to connector **J31**.
Reset Line Enable (JP52)	 JP52	Connects the RESET~ pin of the AVR host microprocessor to the RST~ pin of the Pyxos FT Chip. This is the factory shipped default setting.
	 JP52	Disconnects the RESET~ pin of the AVR host microprocessor from the RST~ pin of the Pyxos FT Chip.

Function	Jumper	Description
Host Microprocessor Interface Connect Enable (JP53)	 JP53	Connects the onboard SPI header (**JP401**) to the AVR host microprocessor. This header connects pins 3, 5, 7, and 9 of the SPI header to PB0 and PB1 of the AVR host microprocessor. The CS~ pin is connected to Ground. This is the factory shipped default setting.
	 JP53	Disconnects the onboard SPI header (**JP401**) from the AVR host microprocessor.
Analog I/O Enable (JP54)	 JP54	Connects the AVR host microprocessor to the EV-Sensor Point onboard analog I/O devices (light sensor, temperature sensor, and analog input voltage sensor). This is the factory shipped default setting.

Function	Jumper	Description
	 JP54	Disconnects the AVR host microprocessor from the EV-Sensor Point onboard analog I/O devices (light sensor, temperature sensor, and analog input voltage sensor).

Connectors

The Pyxos FT EV-Sensor Point evaluation board includes four header connectors, a network connector, and a 24 VDC power connector, as shown in Figure 19 on page 53. The header connectors provide access to the AVR host processor, including host power and I/O, provide access to analog I/O, and provide an interface for debugging or programming the host processor. The network connector provides access to the Pyxos FT network.

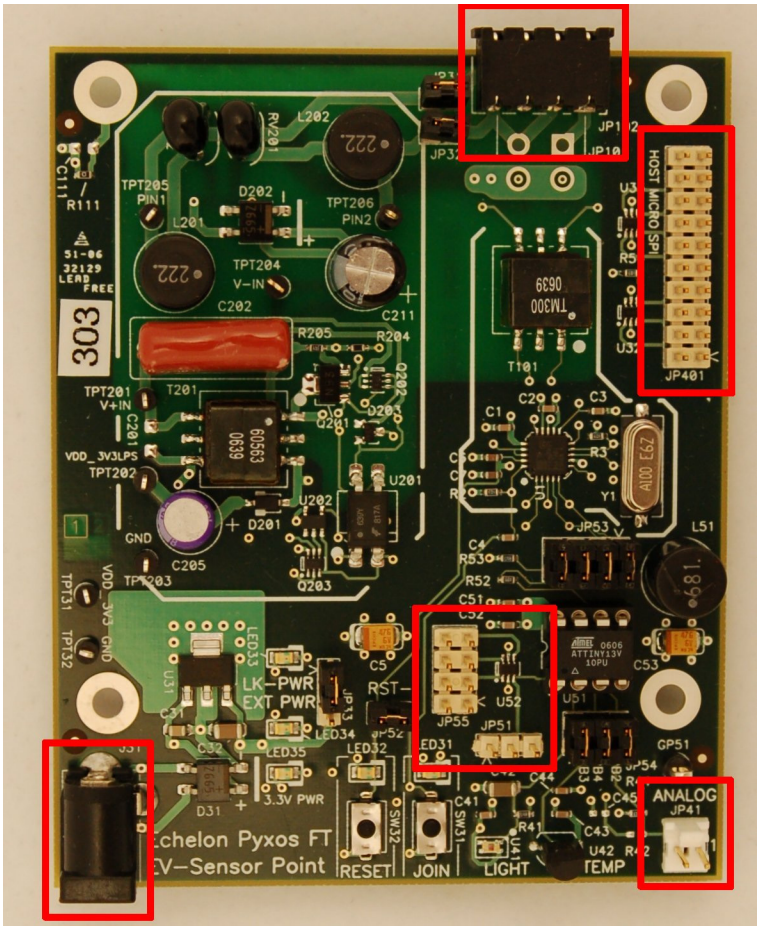


Figure 19. Pyxos FT EV-Sensor Point Evaluation Board Connectors

Header Connector for Accessing Host Power and I/O and for Remote Programming and Debugging

The Pyxos FT EV-Sensor Point evaluation board provides a header connector (**JP51**) that provides external access to the AVR programmed I/O (PIO) line, onboard power (+3.3 VDC), and onboard ground. The PIO line also provides external access to the host processor memory. You can use this header for remote debugging of the AVR application program, for an in-circuit emulator (ICE), or to load programs into the AVR memory.

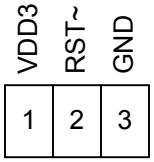


Figure 20. Host Power and I/O Header Connector

The header's three pins are for external supply voltage, reset, and ground, as shown in Figure 20. Pin 2, the Reset pin, allows you to use the AVR host

microprocessor's debugWIRE interface to control program flow and to program the microprocessor's non-volatile memory. For information about how to use this header, see *Debugging for the AVR ATtiny13 Microprocessor* on page 100.

Header Connector for Accessing Host SPI

The Pyxos FT EV-Sensor Point evaluation board provides a header connector (**JP401**) that provides external access to the AVR Serial Peripheral Interface (SPI) lines. The connections for this header are shown in **Figure 21**.

You can use this header together with the Host Microprocessor Interface Connect Enable jumpers (**JP53**) to connect your own host microprocessor to the Pyxos FT Chip on the EV-Sensor Point evaluation board.

GND	20	19	NC
NC	18	17	NC
NC	16	15	NC
GND	14	13	RST~
GND	12	11	PA15 INT~
GND	10	9	PA11 NPSC0~
GND	8	7	PA14 SCLK
GND	6	5	PA13 MOSI
GND	4	3	PA12 MISO
GND	2	1	EXTVDD3

Figure 21. Host SPI Header Connector

Header Connector for Accessing Sensor I/O

The Pyxos FT EV-Sensor Point evaluation board provides a header connector (**JP55**) that provides external access to the sensor output. The connections for this header are shown in **Figure 22**.

GND	8	7	PB2 AI
GND	6	5	PB4 TEMP
GND	4	3	PB3 LIGHT
GND	2	1	EXTVDD3

Figure 22. Sensor I/O Header Connector

Header Connector for Analog Input

The Pyxos FT EV-Sensor Point evaluation board provides a header connector (**JP41**) that provides access to the input to the onboard analog-to-digital converter (ADC). The ADC also receives input from the output of the two onboard sensors.

You can connect the analog output from the EV-Actuator Point to the analog input on the EV-Sensor Point to allow the Pyxos Network Example HMI application program to run the Performance Demo.

Pyxos FT Network Connector

The Pyxos FT EV-Sensor Point evaluation board provides a connector (**JP102**) that provides access to the Pyxos FT network. The EV-Sensor Point uses this connector to communicate with Pyxos FT EV Pilot.

Power Connector

The Pyxos FT EV-Sensor Point evaluation board provides a 24 VDC power connector (**J31**) that supplies optional local power to the EV-Sensor Point. Do not use this connector if the EV-Sensor Point evaluation board is powered by link power.

Pyxos FT EV-Nano Point Evaluation Board

The Pyxos FT EV-Nano Point is a Pyxos Point that serves as a simple digital sensor for the Pyxos FT EVK network. The EV-Nano Point shows how a Pyxos Point can participate in a Pyxos FT network without a host microprocessor.

This section provides additional details for the Pyxos FT EV-Nano Point evaluation board, including descriptions of the push buttons, LEDs, jumper settings, and connectors.

The descriptions of programmatic behaviors in this section apply only to the Pyxos Network Example firmware of the EV Pilot; your custom firmware can provide different behaviors.

Key Features

The EV-Nano Point evaluation board includes the following key features:

- Unhosted Point with four general-purpose digital I/Os
- Small size (6.75 cm x 6.0 cm)
- Link powered using a linear power supply that delivers up to 15 mA application current
- Floating isolation circuit

Power Considerations

The EV-Nano Point evaluation board's voltage is intended to float relative to Earth ground.

Caution: Connecting the onboard ground to Earth (for example, by using an Earth-grounded oscilloscope probe) can damage the EV-Nano Point evaluation board and can impair network communications.

Push Buttons and LEDs

The Pyxos FT EV-Nano Point evaluation board includes a Join button, a Reset button, and two application push buttons. The evaluation board also includes two LEDs that illuminate either when an application push button is pressed or when some condition exists.

Figure 23 shows the locations of the push buttons and LEDs for the Pyxos FT EV-Nano Point evaluation board.

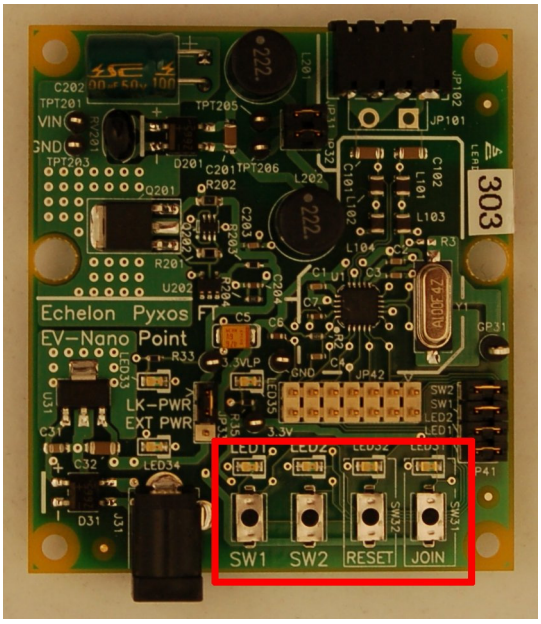


Figure 23. Push Buttons and LEDs for the Pyxos FT EV-Nano Point

Join and Reset Buttons and LEDs

The Pyxos FT EV-Nano Point evaluation board includes a Join button (labeled JOIN) and corresponding LED that is used for joining the Pyxos FT network.

Because the EV-Nano Point is an unhosted Point, it uses manual registration and does not automatically register with the EV Pilot. To register the EV-Nano Point, press the **SW5** push button on the EV Pilot to place the EV Pilot in manual registration mode; in this mode, **LED8** on the EV Pilot blinks. Press the **JOIN** button on the EV-Nano Point to register the Point. When the EV-Nano Point is correctly registered with the EV Pilot and connected to the Pyxos FT network, **LED8** on the EV Pilot is on.

The EV-Nano Point evaluation board also includes a Reset button (labeled RESET) and corresponding LED that you can use to reset the Pyxos FT Chip.

Application Buttons and LEDs

The Pyxos FT EV-Nano Point evaluation board includes the following push buttons for application use:

SW1	Provides digital input for a custom application. This button is not used by the Pyxos Network Example application; however, the Pyxos Network Example HMI application program displays the current state of this push button. When you connect the Pyxos Network Example HMI application program to the EV Pilot through a LONWORKS network connection and press **SW1**, LED1 illuminates on the EV-Nano Point Evaluation Board, and the Network Example HMI shows **LED1** as on.
SW2	Tests dry-contact functions.

The Pyxos EV Pilot example firmware application uses the following LEDs on the EV-Nano Point:

LED1	Indicates status from a custom application. This LED is not used by the Pyxos Network Example application. When you connect the Pyxos Network Example HMI application program to the EV Pilot through a LONWORKS network connection and press **SW1**, LED1 illuminates on the EV-Nano Point Evaluation Board, and the Network Example HMI shows **LED1** as on.
LED2	Indicates whether the dry-contact button on the EV Pilot, EV-Actuator Point, or EV-Nano Point is pressed. This LED remains on while a dry-contact push button is pressed.

Jumper Settings for Controlling Onboard I/O

The Pyxos FT EV-Nano Point evaluation board contains three sets of jumpers, as shown in Figure 24.

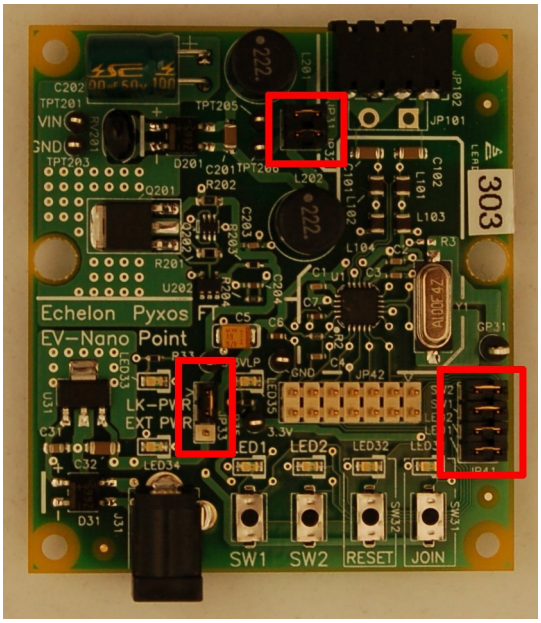
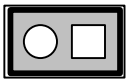
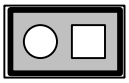
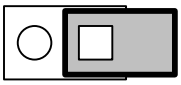
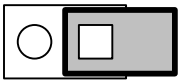
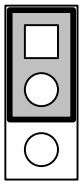
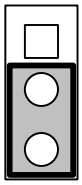
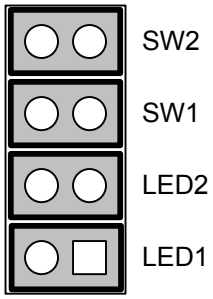
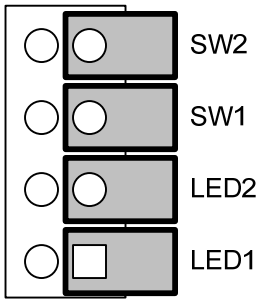


Figure 24. Pyxos FT EV-Nano Point Evaluation Board Jumper Settings

Table 5 describes the EV-Nano Point jumpers.

Table 5. Pyxos FT EV-Nano Point Jumpers

Function	Jumper	Description
Network Power Connect Enable (JP31 and JP32)	 JP31  JP32	Connects the EV-Nano Point to network link power. Be sure to mount or dismount these two jumpers at the same time. When both of these jumpers are mounted, you must also mount jumper **JP33** in the upper position to select the onboard 5 VDC switching power supply. This is the factory shipped default setting.
	 JP31  JP32	Disconnects the EV-Nano Point from network link power. Be sure to mount or dismount these two jumpers at the same time.
Power Supply Input Selector (JP33)	 JP33	Selects the onboard 5 VDC linear power supply. This setting is required for the EV-Nano Point to use link power. This is the factory shipped default setting.
	 JP33	Selects the external power supply connected to connector **J31**.

Function	Jumper	Description
Example I/O Enable (JP41)	 JP41	Connects the Pyxos FT Chip to the EV-Nano Point onboard I/O (**SW1**, **SW2**, **LED1**, and **LED2**). This is the factory shipped default setting.
	 JP41	Disconnects the Pyxos FT Chip from the EV-Nano Point onboard I/O (**SW1**, **SW2**, **LED1**, and **LED2**).

Connectors

The Pyxos FT EV-Nano Point evaluation board includes a header connector, a network connector, and a 24 VDC power connector, as shown in Figure 25 on page 61. The header connector provides access to the Pyxos FT Chip. The network connector provides access to the Pyxos FT network.

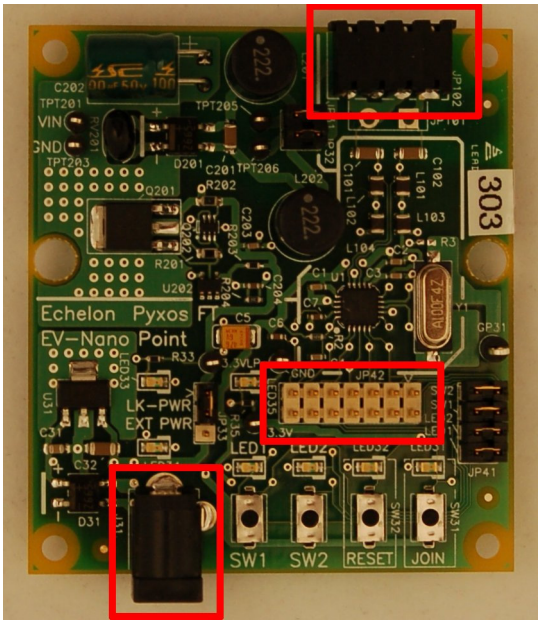


Figure 25. Pyxos FT EV-Nano Point Evaluation Board Connectors

Header Connector for Accessing Power and I/O

The Pyxos FT EV-Nano Point evaluation board provides a header connector (**JP42**) that provides external access to the onboard I/O lines of the Pyxos FT Chip, onboard power (+3.3 VDC), and onboard ground. The connections for this header are shown in **Figure 26**.

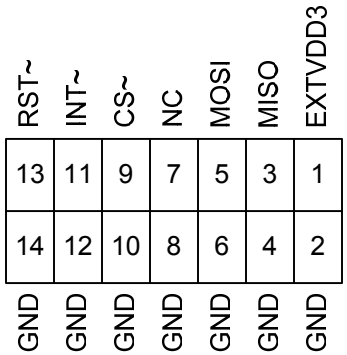


Figure 26. Power and I/O Header Connector

Pyxos FT Network Connector

The Pyxos FT EV-Nano Point evaluation board provides a connector (**JP102**) that provides access to the Pyxos FT network. The EV-Nano Point uses this connector to communicate with Pyxos FT EV Pilot.

Power Connector

The Pyxos FT EV-Nano Point evaluation board provides a 24 VDC power connector (**J31**) that supplies optional local power to the EV-Nano Point. Do not use this connector if the EV-Nano Point evaluation board is powered by link power.

Electromagnetic Compatibility Considerations

Echelon's Pyxos FT technology supports the creation of products that meet a wide variety of regulatory requirements. Chapter 7 of the *Pyxos FT Chip Data Book* describes how to create products with Echelon's Pyxos FT technology that meet electromagnetic compatibility regulations.

The evaluation boards are designed to facilitate testing of Echelon's Pyxos FT technology. As such, they have no enclosure, which provides open access to the I/O connectors, buttons, LEDs, and other I/O components. They have been developed to allow consumer and commercial device OEM suppliers to evaluate the technology quickly, and have not been designed to be installed permanently in homes or commercial buildings. If you work with the evaluation boards in a home environment, operation of other electronic equipment that is sensitive to RF radiated emissions, such as televisions or radios, might be temporarily impaired during the evaluation period.

The standards for RF emissions vary by geographic region. To determine which standards apply in your region, consult the appropriate regulatory agencies. In the European Union, CISPR 22 (or equivalently, EN 55022) applies. In North American, the FCC regulates emissions from unintentional radiators under 47CFR15.109, Subpart A, which allows for substitution of CISPR 22. The evaluation boards comply with CISPR 22 Level A, but not Level B (which is required for deployment in home and commercial environments).

4

Using the Pyxos Network Example

This chapter describes the Pyxos Network Example, including setting it up and operating it.

Overview of the Pyxos Network Example

The Pyxos FT EVK includes an example that simulates a room controller network with analog and digital sensors, analog and digital actuators, and a controller running on a Pyxos Pilot. The example also includes performance demonstration that you can use to demonstrate Pyxos FT network determinism. This example is called the *Pyxos Network Example*. The example demonstrates several of the major features of the Pyxos FT platform, including:

- Automatic registration of Points
- Manual registration of Points
- Seamless replacement of Points within the Pyxos FT network
- Determinism within the Pyxos FT network
- Network monitoring for the Pyxos FT network

You can interact with the Pyxos Network Example directly by pressing the push buttons on the various evaluation boards and by affecting the sensor inputs. You can also interact with the Pyxos Network Example by connecting the EV Pilot to a Windows computer and using the Pyxos Network Example HMI application program. There are a few functions of the Pyxos Network Example application that you can access only by using the Network Example HMI application program (or by using an external connection to the evaluation boards, such as an oscilloscope probe). For example, the Performance Demo that shows network determinism requires the Network Example HMI application program.

Figure 27 on page 65 shows an overview of the Pyxos Network Example.

In addition to using the Pyxos Network Example, you can modify the firmware for the EV Pilot, EV-Actuator Point, or the EV-Sensor Point to create your own Pyxos FT network using the Pyxos FT EVK. For more information about developing your own Pyxos FT applications, see *Developing Pyxos FT Applications Using the Pyxos FT EVK* on page 89, the *Pyxos FT Chip Data Book*, and the *Pyxos FT Programmer's Guide*.

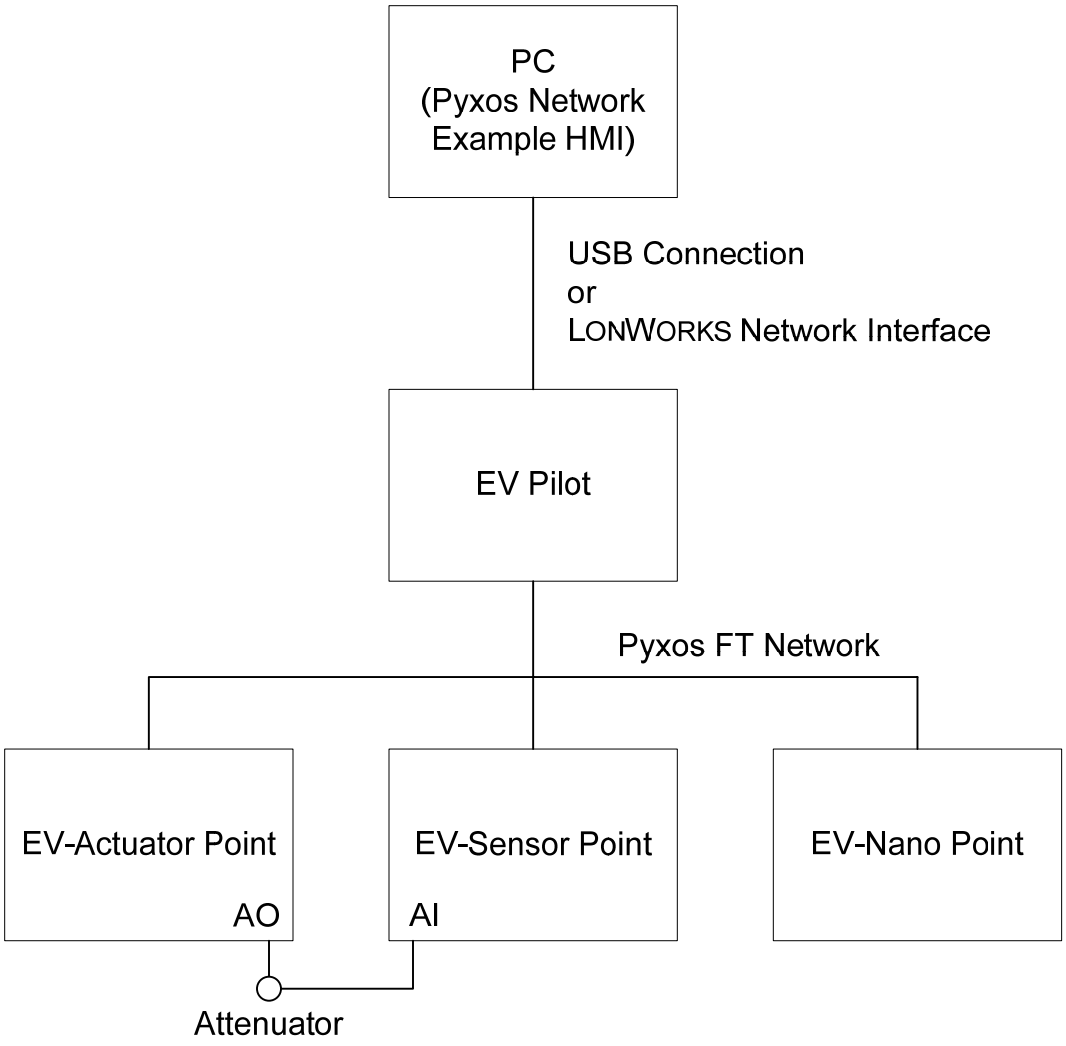


Figure 27. The Pyxos Network Example

The Pyxos FT EV Pilot

The Pyxos Network Example uses the EV Pilot as the network controller. The EV Pilot monitors the status of each of the sensors and controls each of the actuators in the Pyxos FT network. The sensors and actuators are implemented on the three EV Points. The EV Points communicate with the EV Pilot through *Pyxos network variables* (*PNVs*). PNVs are richly-typed data points that facilitate the exchange of data between a Pyxos Pilot and the Points attached to it. When the EV Pilot needs to change the state of one of the EV Points, the Pilot updates a PNV and puts it on the Pyxos FT network for the Point to receive.

The EV Pilot includes a USB connector for communication with a Windows computer. The Pyxos Network Example HMI application program can communicate with the Pyxos FT network using this USB connection.

The EV Pilot also includes an FT 3150 Smart Transceiver with a ShortStack 2 Micro Server and a LONWORKS network connector for optional communication

with a LONWORKS network. The EV Pilot firmware includes a Pyxos LONWORKS Gateway application to facilitate communication with a LONWORKS network.

The EV Pilot includes five push buttons (one for dry-contact input, three for Point registration, and one that is not used for the Pyxos Network Example) and eight status LEDs. In addition, the EV Pilot includes headers that allow you to access I/O and other board functions, most of which are not part of the Pyxos Network Example.

The Pyxos FT EV Points

The Pyxos Network Example uses the following EV Points included with the Pyxos FT EVK for the example sensors and actuators:

- The Pyxos FT EV-Actuator Point is a Pyxos Point that serves as a hosted analog and digital actuator for your Pyxos FT EVK network. The EV-Actuator Point includes an Atmel ARM7 host microprocessor that runs example firmware which uses the Pyxos Point API. It also includes five push buttons (one for dry-contact input and four that are not used for the Pyxos Network Example), eight status LEDs, a header for connecting the analog output of the EV-Actuator Point to the analog input of the EV-Sensor Point, and a potentiometer that controls a voltage attenuator; the header and attenuator are used for the Performance Demo of the Pyxos Network Example HMI application program. In addition, the EV-Actuator Point includes headers that allow you to access I/O and other board functions that are not part of the Pyxos Network Example.
- The Pyxos FT EV-Sensor Point is a Pyxos Point that serves as a hosted multi-sensor for the Pyxos FT EVK network. The EV-Sensor Point includes an Atmel ATtiny13 host microprocessor that runs example firmware which does not use the Pyxos Point API. The sensors measure temperature, light level, and DC voltage. The EV-Sensor Point also includes a header for connecting the analog output of the EV-Actuator Point to the analog input of the EV-Sensor Point. In addition, the EV-Sensor Point includes headers that allow you to access I/O and other board functions that are not part of the Pyxos Network Example.
- The Pyxos FT EV-Nano Point is a Pyxos Point that serves as a simple unhosted digital sensor for the Pyxos FT EVK network. The EV-Nano Point shows how a Point can participate in a Pyxos FT network without a host microprocessor. The EV-Nano Point includes two push buttons, one for dry-contact input and one that is not used for the Pyxos Network Example, and two status LEDs. In addition, the EV-Nano Point includes headers that allow you to access I/O and other board functions that are not part of the Pyxos Network Example.

Starting and Running the Pyxos Network Example

To set up the Pyxos Network Example, complete the following tasks:

1. Connect the Pyxos FT network cables to the Pyxos FT network connectors on each of the four evaluation boards.
2. To control and monitor the Pyxos Network Example from the Network Example HMI application program, perform one of the following tasks:

- Connect the USB cable B end (the square connector) to the EV Pilot and connect the USB cable A end (the flat rectangular connector) to your computer.
- Connect a LONWORKS network cable to the EV Pilot LONWORKS network connector and to a LONWORKS Network Interface device for your computer.

You can create both types of connections, but only one is required and only one can be used at a time.

3. Connect the Pyxos FT EVK I/O cable to the EV-Actuator Point analog I/O header (**JP44**) and to the EV-Sensor Point analog I/O header (**JP41**).
4. Connect the power cable to the EV Pilot. Do not connect power cables to the EV Points if they are set up for link power (the factory shipped default configuration).

The evaluation boards become active as soon as you supply them with power. After a few seconds, after the firmware for the EV Pilot, EV-Actuator Point, and EV-Sensor Point complete their startup processing, the Pyxos Network Example is ready for operation.

Registering Points Automatically

To set up a Pyxos FT network, each Pyxos Point must be *registered* with a Pyxos Pilot. The registration process enables communication between a Pyxos Pilot and a Pyxos Point. There are three types of registration: automatic, manual, and hardwired. The example firmware for EV-Actuator Point and the EV-Sensor Point use automatic registration. The EV-Nano Point always uses manual registration because it is an unhosted Point. The Pyxos Network Example does not use hardwired registration.

For automatic registration in the Pyxos Network Example, when either the EV-Actuator Point or the EV-Sensor Point is initially connected to the Pyxos FT network, the EV Point's firmware writes the EV Point's unique ID (UID) into a free timeslot that is advertised by the EV Pilot. The EV Pilot firmware receives the EV Point's UID and assigns a permanent timeslot to the Point, then the EV Pilot reads the EV Point's program ID and identifies the program interface for the Point based on its type. After the EV Pilot completes this processing, the EV Point is functional and can send and receive network updates.

To force the EV Pilot to re-register the EV-Actuator Point or the EV-Sensor Point, press the appropriate registration push button on the EV Pilot (**SW3** for the EV-Actuator Point or **SW4** for the EV-Sensor Point). Pressing one of these registration push buttons causes the following actions:

- The EV Pilot deletes persistent data that is associated with the EV Point
- The EV Pilot deletes dynamic (PNV) data that is associated with the EV Point
- The EV Pilot frees the timeslot that is associated with the EV Point
- The EV Point (if it is still connected to the network) sends its UID to the EV Pilot to initiate registration
- The EV Pilot receives the EV Point's UID and completes registration for the Point

During the registration of any EV Point, the Network Security LED on the EV Pilot (**LED1**) blinks for ten seconds to indicate that the Pyxos FT network status has changed. When registration is complete, the Network Security LED illuminates to indicate that all three EV Points are connected and registered and that the Pyxos FT network is functioning properly.

When registration for an EV Point is complete, the appropriate Point Status LED on the EV Pilot (**LED6** for the EV-Actuator Point and **LED7** for the EV-Sensor Point) illuminates to indicate that the EV Point is registered and active.

After initial registration, each EV Point's timeslot information persists across resets and power cycles because this information is stored as persistent data in the EV Pilot's host microprocessor. You can clear the persistent data for the EV-Actuator Point or the EV-Sensor Point by pressing the appropriate registration push button on the EV Pilot (**SW3** for the EV-Actuator Point and **SW4** for the EV-Sensor Point).

Registering Points Manually

Hosted Points can use any of the three registration methods, but unhosted Points must use manual registration. Therefore, the EV-Nano Point uses manual registration. The EV-Nano Point is non-operational when initially connected to the network.

To manually register the EV-Nano Point, perform the following steps:

1. Press the **SW5** push button on the EV Pilot to place the Pilot in manual registration mode. While it is in this mode, the EV Pilot's Nano Point Status LED (**LED8**) blinks. If the EV-Nano Point was previously registered, pressing this push button causes the EV Pilot to:
   - Delete persistent data that is associated with the EV-Nano Point
   - Delete dynamic (PNV) data that is associated with the EV-Nano Point
   - Free the timeslot that is associated with the EV-Nano Point
2. Press the Join push button on the EV-Nano Point to initiate manual registration. If you press the EV-Nano Point's Join push button when the EV Pilot is not in manual registration mode, the EV Pilot ignores the registration request.

Note: If you accidentally press the **SW5** push button on the EV Pilot, press the **SW5** push button a second time to ensure that the EV Pilot is not in manual registration mode. Because pressing this push button causes the EV-Nano Point to be no longer registered, **LED8** should be off.

During manual registration, pressing the Join push button on the EV-Nano Point causes the Pyxos FT Chip on the Point to send the Point's unique ID (UID) to the EV Pilot. The EV Pilot firmware receives the Point's UID and assigns a permanent timeslot to the EV Point, then identifies the program interface for the Point based on its type. After the EV Pilot completes this processing, the EV-Nano Point is functional and can send and receive network updates.

During registration of any Point, the Network Security LED on the EV Pilot (**LED1**) blinks for ten seconds to indicate that the Pyxos FT network status has changed. When registration is complete, the Network Security LED illuminates

to indicate that all three EV Points are connected and registered and that the Pyxos network is functioning properly.

When registration for the EV Point is complete, the EV-Nano Point Status LED on the EV Pilot (**LED8**) illuminates to indicate that the Point is registered and active.

After initial registration, each EV Point's timeslot information persists across resets and power cycles because this information is stored as persistent data in the EV Pilot's host microprocessor. You can clear the persistent data for the EV-Nano Point by pressing **SW5** on the EV Pilot to place the EV Pilot in manual registration mode. If you do not want to re-register the Point, press **SW5** a second time to cause the EV Pilot to leave manual registration mode.

Monitoring Activity within the Pyxos FT Network

Whenever the state changes for some data point within the Pyxos Network Example, the originator of that change encapsulates the changed data value in a Pyxos network variable (PNV) update. Each PNV is associated with a Point and has a direction that is determined relative to its Point, so that a PNV is either an input PNV or an output PNV relative to the Point. Thus, a Point sends output PNVs to the Pilot, and the Pilot sends input PNVs to the Point. For example, the EV Pilot updates an input PNV to set an LED on an EV Point, and an EV Point sends an output PNV to the EV Pilot when you press a push button on the EV Point.

The Pyxos Network Example implements a number of PNVs that you can monitor and control through the LEDs and push buttons on the EV Pilot and EV Points. There are also a few other PNVs that you can monitor and control only through the Pyxos Network Example HMI application program. This section describes the Pyxos FT network activity that you can monitor and control.

Monitoring Sensor Data and Dry-Contact Input

The EV-Sensor Point includes three sensors: a light-level sensor, a temperature sensor, and an analog voltage-level sensor. As the sensor values change, the EV-Sensor Point firmware encapsulates the changes in PNVs and sends them to the EV Pilot over the Pyxos FT network. The EV Pilot firmware monitors the updated PNVs, and takes appropriate action as necessary, such as updating a PNV to illuminate an LED.

Monitoring Sensor Data

To monitor sensor data:

- The EV-Sensor Point reads the light level from the light-level sensor and sends the light-level value as a PNV to the EV Pilot. If the light-level value is less than the Low Light Level alarm threshold value (by default, 50 lux), the EV Pilot updates a PNV to illuminate the Low Light Level LED on the EV-Actuator Point (**LED4**). The EV Pilot also illuminates its own Low Light Level LED (**LED4**).

  If the EV Pilot is connected to the Pyxos Network Example HMI application program, the Pilot also sends it the updated light-level PNV.

Using the Network Example HMI application program, you can override the default Low Light Level alarm threshold value.

- The EV-Sensor Point reads the temperature level from the temperature sensor and sends the temperature value as a PNV to the EV Pilot. If the temperature value is greater than the Over Temperature alarm threshold value (by default, 30 °C or 86 °F), the EV Pilot updates a PNV to illuminate the Over Temperature LED on the EV-Actuator Point (**LED3**). The EV Pilot also illuminates its own Over Temperature LED (**LED3**).

  If the EV Pilot is connected to the Pyxos Network Example HMI application program, the Pilot also sends it the updated temperature PNV. Using the Network Example HMI application program, you can override the default Over Temperature alarm threshold value.

- The EV-Sensor Point reads the analog voltage-level from the analog voltage-level sensor and sends the voltage value as a PNV to the EV Pilot. The EV Pilot takes no action based on the voltage level unless it is connected to the Network Example HMI application program or to a LONWORKS network.

  When the EV Pilot is connected to the Network Example HMI application program, the EV Pilot sends it the voltage-level PNV. Using the Pyxos Network Example HMI application program, you can override the analog output value or run the Performance Demo. When the performance demo is active, the EV Pilot updates a PNV to illuminate the Performance Demo LED (**LED5**) on the EV-Actuator Point. The EV Pilot also illuminates its own Performance Demo LED (**LED5**).

Monitoring Dry-Contact Input

The evaluation boards for the EV-Actuator Point, EV-Nano Point, and the EV Pilot each include a push button that represents a dry-contact button. A *dry-contact* button is a push button with an electrical contact that does not make or break a circuit, that is, no current flows through it. It represents a simple digital input.

The evaluation boards include digital input circuits that read the state of the dry-contact input:

- The EV-Actuator Point reads the state of its dry-contact push button (**SW1**) and sends the changed state as a PNV to the EV Pilot.
- The EV-Nano Point reads the state of its dry-contact push button (**SW2**) and sends the changed state as an I/O update to the EV Pilot.
- If the dry-contact push button (**SW1**) on the EV Pilot itself changes state, the EV Pilot does not update a PNV for this change, but acts on the change directly.

If the dry-contact push button state is on (that is, someone is pressing the button), the EV Pilot updates a PNV to illuminate the Dry-contact LED on the EV-Actuator Point (**LED2**) and the EV-Nano Point (**LED2**). The EV Pilot also illuminates its own Dry Contact LED (**LED2**).

If the EV Pilot is connected to the Pyxos Network Example HMI application program, the Pilot also sends it the updated dry-contact PNV.

Monitoring Network Integrity and Security

When an EV Point is connected to the Pyxos FT network, or when an EV Point is disconnected from the Pyxos FT network, the EV Pilot causes its Network Security LED (**LED1**) to blink for ten seconds to indicate a network security alarm condition. If the EV Pilot is connected to the Pyxos Network Example HMI application program, the Pilot also sends an updated network-security PNV value to it.

When the Pyxos FT network is functioning properly, that is, when all of the EV Points are configured properly and connected to the Pyxos FT network, the Network Security LED is on.

Monitoring and Controlling Analog I/O

The EV-Actuator Point includes an analog output header, and the EV-Sensor Point includes an analog input header. You can use these two headers to monitor and control analog I/O for the Pyxos FT EVK.

When the Performance Demo of the Pyxos Network Example HMI application program is not running, you can use the Pyxos Network Example HMI application program to monitor user analog input from the EV-Sensor Point analog input header and control user analog output on the EV-Actuator Point analog output header.

When the Performance Demo is running, the EV Pilot continuously sends updated voltage-level PNVs to the Network Example HMI application program. In this case, you cannot manually control the analog I/O; however, you can still monitor it.

Monitoring and Controlling Digital I/O

The EV-Actuator Point includes four digital outputs on one header, and four digital inputs that are shared with the **SW2** through **SW5** push buttons on another header. You can use these headers to monitor and control digital I/O.

If the EV Pilot is connected to the Pyxos Network Example HMI application program, the Pilot sends it the updated output PNVs that represent the digital-input values from the EV-Actuator Point. You can use the Network Example HMI application program to monitor and control the digital input and output values, although the Pyxos Network Example does not use these values.

Replacing Points and Simulating Network Failure

The EV Pilot saves the timeslot number, unique ID, and program ID (if any) of each Pyxos Point on the network. When you replace a Point, if the Point has a program ID that matches a previously removed Point, the EV Pilot attempts to replace the Point in the previously used timeslot. For Point replacement, the EV Pilot performs the following tasks:

1. When the EV Pilot receives a registration request from an Point, the EV Pilot initially allocates an available timeslot for the Point.
2. As part of the registration process, the Pyxos FT Pilot API automatically reads the Point's program ID. The EV Pilot searches its persistent data

to determine if there is any data available for this program ID. The EV Pilot also determines if it can communicate with the old Point using this timeslot.

3. If there is already a timeslot for the program ID, and the EV Pilot cannot communicate with the old Point using this timeslot, the EV Pilot releases the current timeslot and assigns the previously used timeslot to the new Point.

For unhosted Points, user input is required to indicate that the Point needs to be replaced. For the EV-Nano Point, press the **SW5** push button on the EV Pilot to enter manual registration mode, and then press the Join push button on the EV-Nano Point to initiate the replacement process. If an unused timeslot exists that was previously used by this type of Point, the Pilot will assign it to this newly attached Point.

To simulate network failure, detach the Pyxos FT network cable from one of the EV Points or from the EV Pilot. The Pilot detects that a Point is non-responsive and enters recovery mode for that Point. When you reconnect the network cable, the Pilot performs recovery processing for the Point and attempts to reconfigure it.

Stopping the Pyxos Network Example

To stop the Pyxos Network Example, remove power from the EV Pilot. The EV Pilot firmware periodically saves necessary data in persistent memory in the Pilot's host microprocessor, so no registration information is lost when you remove power from the EV Pilot. However, the EV Pilot does not save current PNV values in persistent memory, so all PNVs are updated the next time you supply power to the EV-Pilot.

5

Running the Pyxos Network Example HMI Application Program

This chapter describes the Pyxos Network Example HMI software application program that controls and interacts with the Pyxos Network Example application firmware for the Pyxos FT EVK.

Starting the Pyxos Network Example HMI Application Program

Before you run the Pyxos Network Example HMI application program, follow the instructions described in *Installing the Pyxos FT EVK Software* on page 15 to install it on your computer.

To start the Pyxos Network Example HMI application program, go to the Windows **Start** menu and select **Programs → Echelon Pyxos FT EVK → Pyxos Network Example HMI**.

Important: Set your Windows screen resolution (the display screen area) to 1024 by 768 pixels or higher before running the Pyxos Network Example HMI application program. You can set your color settings to either high color (16-bit) or true color (32-bit) for the Pyxos Network Example HMI application program.

Figure 28 shows the main window of the Pyxos Network Example HMI application program.

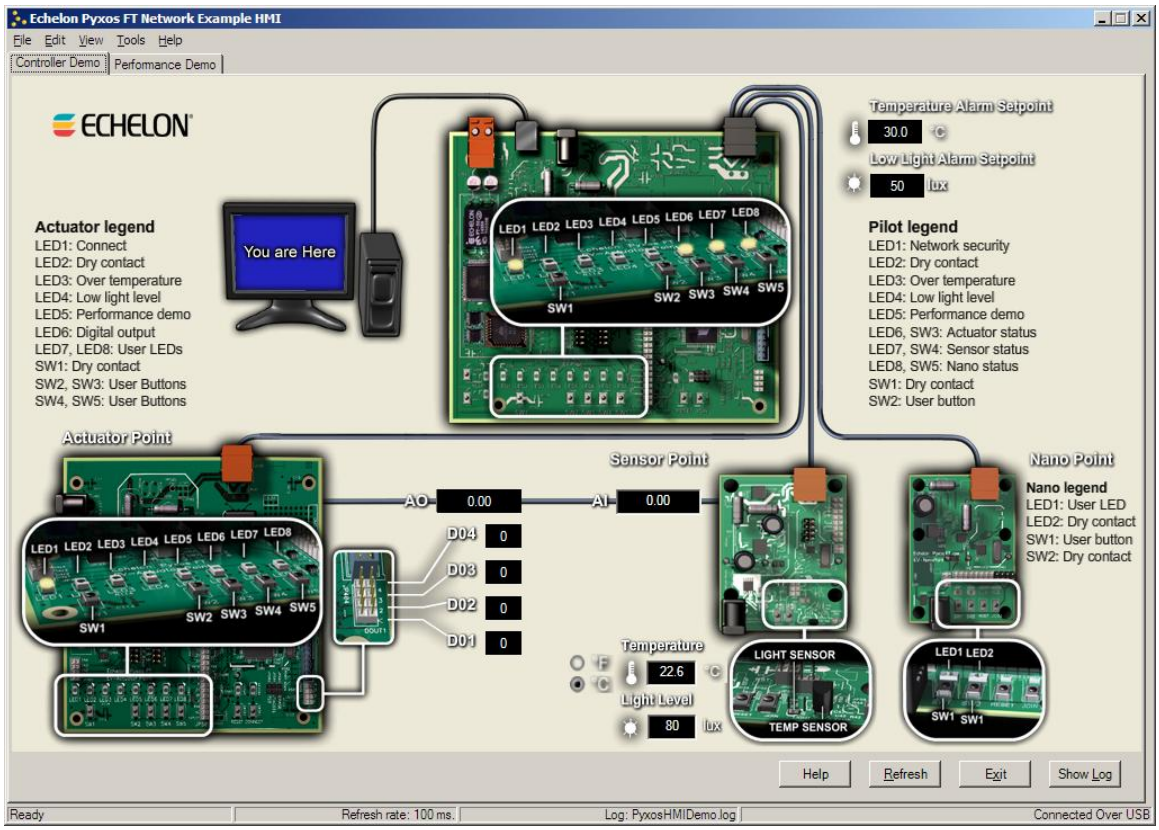


Figure 28. Pyxos Network Example HMI Application Program

Connecting to the Pyxos FT Network

When you start the Pyxos Network Example HMI application program, you see the Connect To Network dialog. You can connect your computer to the Pyxos FT network using either a direct USB connection or using a LONWORKS network

interface. In the Connect To Network dialog, perform either of the following steps:

- For USB, select **Direct**. Click **Connect**.
- For LONWORKS, select **LonWorks** then select a LONWORKS network interface from the dropdown list. Click **Connect**. The Connect To Pyxos Pilot dialog opens, in which you must enter the Neuron ID for the EV Pilot. Press the Neuron Chip's Service pin button (labeled SERVICE) on the EV Pilot to send the Neuron ID to the Network Example HMI application program. Click **OK**.

You can switch between these two connection methods at any time. When the Network Example HMI application program is running, you can connect or reconnect your computer to the Pyxos FT network by selecting **Tools → Connect to Network**.

LONWORKS Functionality

When the EV Pilot sends updates to the Network Example HMI application program over a LONWORKS network, it sends them as LONWORKS network variable (NV) updates. When the Network Example HMI application program receives an updated output LONWORKS NV, it updates the display for that NV. When you update a control within the Network Example HMI application program, it sends the update to the EV Pilot by updating one of the Pilot's input LONWORKS NVs.

The basic functionality of the Network Example HMI application program is the same whether you connect to it by USB or by a LONWORKS network. However, the Network Example HMI application program exhibits the following differences in functionality when connected by a LONWORKS network:

- The **Performance Demo** tab is not available, and you cannot run the performance demo. You must connect by USB to run the performance demo. You can, however, still control the analog output from the EV-Actuator Point and monitor the analog input from the EV-Sensor Point.
- You cannot reset EV Points or clear timeslot information for the EV Points. You also cannot control the refresh rate of the information displayed in the main window.
- When you press the **SW5** push button on the EV-Actuator Point, the Network Example HMI application program updates an input NV for the EV Pilot. This input NV instructs the EV Pilot to illuminate **LED8** on the EV-Actuator Point. The Network Example HMI application program displays the current state of **LED8** as well as the state of **SW5**. This LED does not illuminate when connected by USB.

  This function is provided to illustrate how the functionality of a Pyxos FT network can be extended by providing control over a LONWORKS network.
- When you press the **SW1** push button on the EV-Nano Point, the Network Example HMI application program updates an input NV for the EV Pilot. This input NV instructs the EV Pilot to illuminate **LED1** on the EV-Nano Point. The Network Example HMI application program displays the current state of **LED1** as well as the state of **SW1**. This LED does not illuminate when connected by USB.

This function is provided to illustrate how the functionality of a Pyxos FT network can be extended by providing control over a LONWORKS network.

- The log window area includes slightly different information for alarm events when connected by a LONWORKS network interface than it displays when connected by USB.
- Overall performance of the Network Example HMI application program can be slower than when connected by USB.

In the rest of this chapter, the functionality described assumes that you are connected to the Network Example HMI application program by USB, unless stated otherwise.

Controlling the Pilot and Points in the Network

The **Controller Demo** tab displays a graphical representation of the Pyxos FT network as configured for the Network Example application, including the EV Pilot, EV-Actuator Point, EV-Sensor Point, and EV-Nano Point. Whenever a PNV is updated within the Network Example, the EV Pilot sends the updated PNV to the Network Example HMI application program. Thus, the Network Example HMI application program shows the current states of the various LEDs, push buttons, and sensor values for the EV Pilot and each of the EV Points.

When the EV Pilot or an EV Point in the Pyxos FT network becomes unavailable or is disconnected from the network, the controls for that device are disabled and the display shows it as not connected to the Pyxos FT network. After you make the EV Pilot or EV Point available again or reconnect it to the network, the controls for that device are re-enabled and the display shows it as connected to the Pyxos FT network.

The information that is displayed on the **Controller Demo** tab is refreshed periodically. You can also refresh the displayed information manually, as described in *Refreshing the Display* on page 84.

The following sections describe how to control the EV Pilot and EV Points for the Pyxos Network Example application.

Setting the Alarm Temperature

To set the temperature value at which the EV-Actuator Point and the EV Pilot register an alarm condition and illuminate the Over Temperature LEDs, perform either of the following steps from the **Controller Demo** tab:

- Click the temperature value displayed for the Temperature Alarm set point, as shown in **Figure 29**, to display the Set Alarm Temperature dialog. This alarm set point appears to the right of the display for the Pilot.
- Select **Edit** → **EV-Pilot** → **Set Alarm Temperature** to display the Set Alarm Temperature dialog.

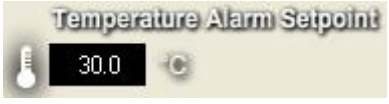


Figure 29. Temperature Alarm Set Point

In the Set Alarm Temperature dialog, specify a valid temperature value in the **Value** field and click **OK**. Valid temperature values are 0 to 100 °C or 32 to 212 °F.

You can switch between ºC (degrees Centigrade) and ºF (degrees Fahrenheit) by clicking the radio buttons that appear next to the Temperature output display, as shown in **Figure 30**. This display appears to the left of the Sensor Point.

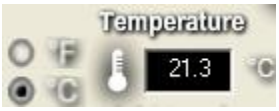


Figure 30. Temperature Scale Radio Buttons

Setting the Light-Level Threshold

To set the lux value at which the EV-Actuator Point and the EV Pilot register an alarm condition and illuminate the Low Light level LEDs, perform either of the following steps from the **Controller Demo** tab:

- Click the lux value displayed for the Low Light Alarm set point, as shown in **Figure 31**, to display the Set Light Level Threshold dialog. This alarm set point appears to the right of the display for the Pilot.
- Select **Edit** → **EV-Pilot** → **Set Light Level Threshold** to display the Set Light Level Threshold dialog.

Figure 31. Low Light Alarm Set Point

In the Set Light Level Threshold dialog, specify a valid lux value in the **Value** field and click **OK**. Valid values are 0 to 2000 lux.

The lux is a measure of illuminance per unit area, and 1 lux is approximately equivalent to 1/10 footcandle. A value of 400 lux represents a brightly lit office.

Setting the Outputs for the EV-Actuator Point

You can set analog and digital voltage output values for the EV-Actuator Point. The analog output value is used for the Performance Demo (see *Running the Performance Demo* on page 81). The digital output values are not used by the Pyxos Network Example application.

Setting the Analog Output

To set the analog output value for the EV-Actuator Point, perform either of the following steps from the **Controller Demo** tab:

- Click the analog output value set point labeled **AO**, as shown in **Figure 32** on page 78, to display the Set Analog Output dialog. This set point appears to the right of the EV-Actuator Point.
- Select **Edit** → **EV-Actuator** → **Set Analog Output** to display the Set Analog Output dialog.

Figure 32. Analog Output Set Point

In the Set Analog Output dialog, specify a valid analog output value in the **Value** field and click **OK**. Valid values are 0 to 3.0 volts.

Note: You cannot set the value for the analog output while the Performance Demo is running.

If the Pyxos FT EVK I/O cable is connected to the EV-Actuator Point and the EV-Sensor Point, the analog input to the EV-Sensor Point monitors the analog output from the EV-Actuator Point, and the measured value for the analog input is displayed in the box labeled AI, which is to the left of the EV-Sensor Point.

Setting a Digital Output

To set any of the four digital output values for the EV-Actuator Point, perform either of the following steps from the **Controller Demo** tab:

- Click a digital output set point value labeled **DO1** through **DO4**, as shown in **Figure 33**. This set point appears to the right of the EV-Actuator Point. Each click toggles the current value, so that if the current value is 1, clicking it changes the value to 0.
- Select **Edit** → **EV-Actuator** → **Set Digital Outputs** to display the Set Digital Outputs dialog. In the Set Digital Outputs dialog, click the digital value displayed for one of the digital outputs labeled **DO1** through **DO4**. Each click toggles the current value, so that if the current value is 1, clicking it changes the value to 0. Click **OK** to set the values and close the dialog.

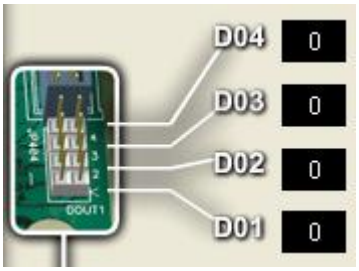


Figure 33. Digital Output Set Points

When any of the four digital outputs is active (set to 1), **LED6** is illuminated on the EV-Actuator Point and in the Network Example HMI application program's main window.

Resetting Points

You can reset any or of all the EV Points from the Pyxos Network Example HMI application program. Resetting an EV Point causes the following actions:

- The Pyxos FT Chip resets, which causes the EV Point to become unconfigured
- For hosted EV Points, the host microprocessor resets

- The EV Pilot reconfigures the EV Point
- The EV Pilot resends cached input PNV values to the EV Point

Resetting an EV Point from the Network Example HMI application program is similar to pressing the Reset button on the evaluation board for that EV Point:

- When you press a Point's Reset button, the EV Point loses its configuration, the EV Pilot detects a communication error for that Point, and the Pilot initiates recovery for the EV Point.
- When you reset an EV Point from the Network Example HMI application program, it sends a reset message to the EV Pilot, which reconfigures the Point without initiating recovery.

Note: You cannot reset the EV Pilot from the Network Example HMI application program. To reset the Pilot, press the Reset button on the EV Pilot evaluation board.

Within the Network Example HMI application program, resetting an EV Point does not affect the set-point values that you might have set for the EV-Actuator Point or the EV-Sensor Point.

To reset any or all of the EV Points from the Network Example HMI application program, perform any of the following steps from either the **Controller Demo** tab or the **Performance Demo** tab:

- Select **Tools** → **Reset** → **EV-Sensor**.
- Select **Tools** → **Reset** → **EV-Actuator**.
- Select **Tools** → **Reset** → **EV-Nano**.
- Select **Tools** → **Reset All Points**.

Monitoring Pilot and Point Status

You can monitor status for the EV Pilot and EV Points from the **Controller Demo** tab of the Pyxos Network Example HMI application program. As shown in **Figure 28** on page 74, the **Controller Demo** tab displays a logical representation of the EV Pilot, EV-Actuator Point, EV-Sensor Point, and EV-Nano Point, along with a logical representation of the Pyxos FT network connections.

For the Pilot and for each EV Point, the Network Example HMI application program displays the current status of the LEDs and push buttons, just as they are on the evaluation boards themselves. The Network Example HMI application program also shows the current temperature, light levels, and analog voltage level measured by the EV-Sensor Point.

The Network Example HMI also shows disruptions to the Pyxos FT network. If any of the EV Points loses its connection to the EV Pilot (and when the EV-Nano Point is not yet registered), the Network Example HMI application program changes the display for the connection and displays a red X on the line that represents the network connection to the disconnected point.

Clearing Timeslot Information

You can clear the timeslot information for the Pyxos FT network by selecting **Tools** → **Clear Timeslot Information** from either the **Controller Demo** tab or the

Performance Demo tab. Clearing the timeslot information for the Pyxos FT network causes the EV Pilot to delete all EV Point information from its memory and to reset each EV Point. Because the EV-Actuator Point and EV-Sensor Point use automatic registration, if the EV Point is connected to the network, the EV Pilot automatically re-registers the EV Point and assigns it a new Pyxos FT network timeslot.

After you clear timeslot information, you must manually re-register the EV-Nano Point. To register the EV-Nano Point, press the **SW5** button on the EV Pilot to place the Pilot in manual registration mode; then press the Join button on the EV-Nano Point to register the Point.

Logging Pyxos FT Network Events

The Pyxos Network Example HMI application program can log events that occur on the Pyxos FT network, such as when EV Points are registered, when an EV Point is disconnected from the Pyxos FT network, when a dry-contact button is pressed, when the Performance Demo is running, and when alarm conditions exist for the temperature and light-level sensors.

The Network Example HMI application program maintains log information in a file named **PyxosHMIDemo.log** in the [*Pyxos FT EVK*]\Bin directory, where [*Pyxos FT EVK*] is directory in which you installed the Pyxos FT EVK software, usually **C:\Program Files\Echelon\Pyxos FT EVK**.

Displaying the Log Window Area

To display the log window area, perform one of the following steps from either the **Controller Demo** tab or the **Performance Demo** tab:

- Click **Show Log**.
- Select **View** → **Show Log**.

To hide the log window area, perform one of the following steps from either the **Controller Demo** tab or the **Performance Demo** tab:

- Click **Hide Log**.
- Select **View** → **Hide Log**.

Log information is not lost while the log window area is hidden, however, if you display the log after it has been hidden, you might need to scroll the contents of the log window area to see current log entries.

Copying Log Information

To copy information from the log window area to the Windows system clipboard, use the mouse to highlight the text that you want to copy, then perform one of the following steps:

- Select **Edit** → **Copy** from either the **Controller Demo** tab or the **Performance Demo** tab.
- Right-click within the log window area and select **Copy** from the popup menu.
- Press Ctrl-C on the keyboard.

You can also select all of the text within the log window area by clicking within the log window area and selecting **Edit → Select All** from either the **Controller Demo** tab or the **Performance Demo** tab (or by pressing Ctrl-A on the keyboard).

Controlling Logging

To clear the log window area and discard current log information, right-click within the log window area and select **Clear Log** from the popup menu.

To temporarily disable logging, right-click within the log window area and select **Pause Logging** from the popup menu. To re-enable logging, right-click within the log window area and select **Resume Logging** from the popup menu.

To control what types of events are included in the log, right-click within the log window area and select **Filter** from the popup menu. You can selectively include or exclude the following event types in the log:

- Registration events
- PNV updates
- Alarms
- Errors and resets

Running the Performance Demo

The Pyxos Network Example HMI application program includes a performance demonstration that shows network determinism for the Pyxos FT network by simultaneously generating and monitoring a time-varying analog voltage signal, which can be optionally modified by an attenuator.

Important: To view and control the Performance Demo, you must use a direct USB connection between the computer that is running the Network Example HMI application program and the Pyxos FT network.

The **Performance Demo** tab displays the output from the Performance Demo in real time. The display for the Performance Demo is a pair of sine curves:

- The blue curve shows the generated output from the EV Pilot for the specified frequency and amplitude.
- The red curve shows the output signal measured by the EV-Sensor Point. If the measured signal is attenuated, the red curve's amplitude is less than the blue curve's amplitude, although they share the same zero point.

When the Performance Demo is active, it performs the following actions, as shown in **Figure 34** on page 82:

1. The EV Pilot sends a stream of Pyxos network variable (PNV) updates to the EV-Actuator Point to generate a time-varying sinusoidal DC analog signal.

   The EV Pilot also sends these values directly to the Network Example HMI application program, which uses the values as the basis for its generated output (the blue curve on the display).

2. The EV-Actuator Point passes that signal through its attenuator (the potentiometer on the EV-Actuator Point), and makes the signal available

on its **AO** header.

You connect to the **AO** header of the EV-Actuator Point to the **AI** header of the EV-Sensor Point using the Pyxos FT EVK I/O cable.

3. The EV-Sensor Point reads the signal on its **AI** header.
4. The EV-Sensor Point sends the current signal value as a PNV to the EV Pilot.
5. The EV Pilot sends the value to the Network Example HMI application program.
6. The Network Example HMI application program displays the measured value, along with the generated output value. You can view the raw values from the **Controller Demo** tab in the AO and AI displays, and you can view the continuous values as sine curves from the **Performance Demo** tab.

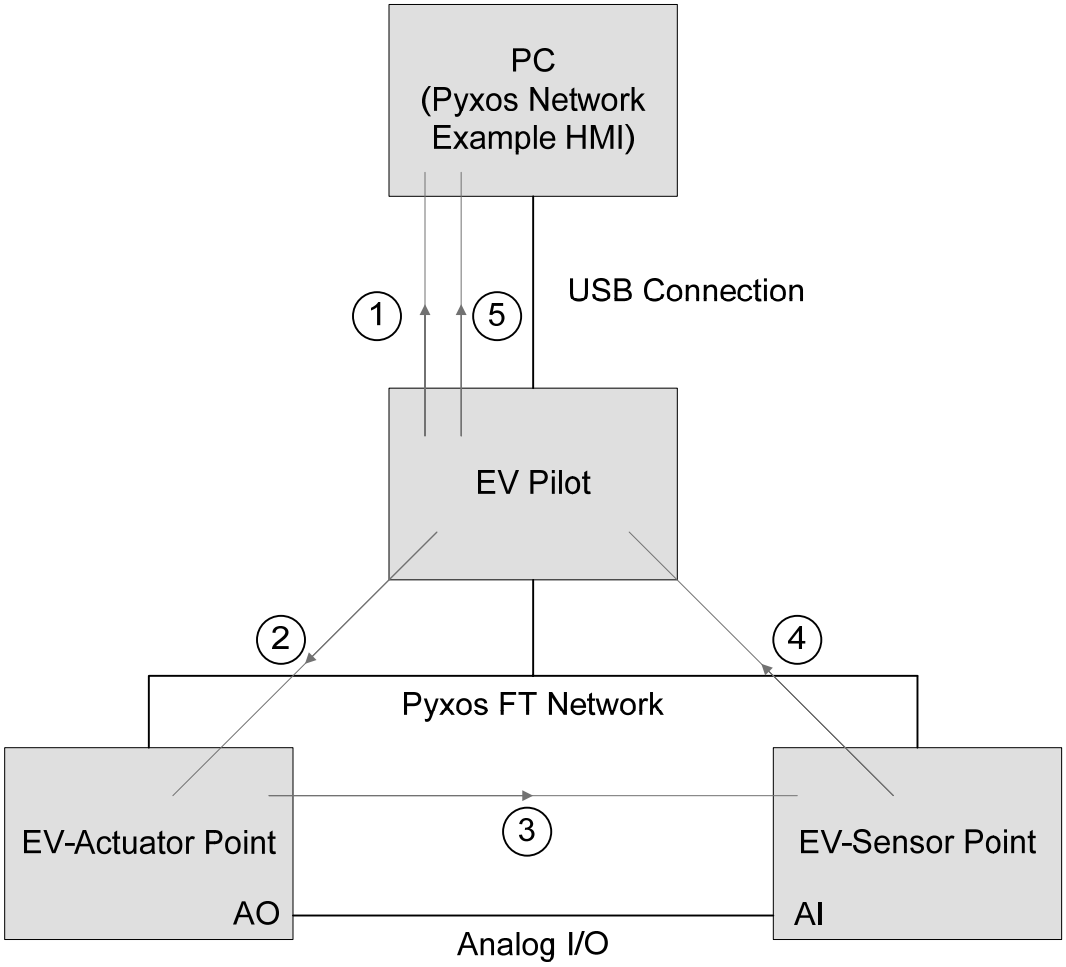


Figure 34. Overview of the Performance Demo

The following sections describe how to control the Performance Demo.

Setting the Frequency for the Performance Demo

To change the frequency for the Performance Demo, perform either of the following steps from the **Performance Demo** tab:

- Click **Change Frequency** to display the Change Frequency for Performance Demo dialog.
- Select **Edit** → **Change Frequency** to display the Change Frequency for Performance Demo dialog.

In the Change Frequency for Performance Demo dialog, you can set how often the Network Example HMI application program refreshes the display output for the Performance Demo and how often the EV Pilot sends a new voltage value for the EV-Actuator Point to make available on its **AO** header, which controls the frequency of the output curve.

In the Change Frequency for Performance Demo dialog:

- To set the refresh rate for drawing the output curves to the screen, select a value from the **Refresh the display every *n* ms** dropdown list. Larger values introduce delay in drawing the output to the screen, but do not affect the data generated by the EV Pilot and measured by the EV-Sensor Point.
- To set the frequency for how often the EV Pilot sends updated values to the Network Example HMI application program for the Performance Demo, enter a valid value in the **Send a new value every *n* frames** field. Valid values are 1 to 32 Pyxos FT network frames. This frame rate represents a range of approximately 297 frames per second to 9 frames per second, or approximately 0.825 Hz to 0.026 Hz for the curves.

To change the amplitude of the output curve, adjust the potentiometer on the EV-Actuator Point.

Starting and Stopping the Performance Demo

To start or stop the Performance Demo, perform either of the following steps from the **Performance Demo** tab:

- Click **Start** to start the demo; click **Stop** to stop the demo.
- Select **Tools** → **Start Performance Demo** to start the demo; select **Tools** → **Stop Performance Demo** to stop the demo.

You can return to the **Controller Demo** tab while the Performance Demo is running to interact with other parts of the Pyxos Network Example HMI application program, but you can interact with the Performance Demo only from the **Performance Demo** tab.

Important: If the Performance Demo is running and you open the Connect To Network dialog to connect the Network Example HMI application to the Pyxos FT network using a LONWORKS network interface, the Performance Demo continues to run. Because you cannot stop the Performance Demo while you are connected by a LONWORKS network interface, you cannot modify the analog output value for the EV-Actuator Point.

Recommendation: Stop the Performance Demo before changing the host connection type.

Refreshing the Display

You can control how often the Pyxos Network Example HMI application program refreshes its display of the information that it collects from the EV Pilot.

To refresh the displayed information on the **Controller Demo** tab, perform either of the following steps:

- Click **Refresh**.
- Select **Tools** → **Refresh Now**.

Select **Tools** → **Set Refresh Rate** to control how often the Network Example HMI application program refreshes the displayed information. In the Set Refresh Rate dialog, select a value from the **Refresh the display every *n* ms** dropdown list. Larger values introduce delay in refreshing the information to the screen, but do not affect the data generated by the EV Pilot or any of the EV Points.

To set the refresh rate for information on the **Performance Demo** tab, click **Change Frequency**, as described in *Setting the Frequency for the Performance Demo* on page 83.

Shutting Down the Pyxos Network Example HMI Application Program

To shut down the Pyxos Network Example HMI application program, perform either of the following steps from either the **Controller Demo** tab or the **Performance Demo** tab:

- Click **Exit**.
- Select **File** → **Exit**.

Recommendation: After you shut down the program, remove power from the EV Pilot evaluation board while you are not using the Pyxos FT EVK.

6

Troubleshooting the Pyxos FT Network Example

This chapter describes basic troubleshooting information for the hardware and software components that are included with the Pyxos FT EVK.

Troubleshooting

This chapter describes some common problems that you might experience while using the Pyxos FT EVK, along with suggested actions that can either fix the problem or help to diagnose the problem. For further assistance in troubleshooting the Pyxos FT EVK, contact Echelon Support at www.echelon.com/support.

1. My EV Pilot evaluation board has no power (the power LED is off).

- Check that the power supply is properly plugged into the EV Pilot power connector and to the AC mains power.
- Check that power-related jumper settings (**JP31** through **JP35**) are correct.

2. My EV Point evaluation board has no power.

- If you are using link power:
  - Ensure that the EV Pilot evaluation board is powered.
  - Ensure that the Pyxos FT network cable is properly connected to the EV Point and to the EV Pilot.
  - Check that the Network Power Connect Enable jumpers (**JP31** and **JP32**) are properly mounted on the EV Pilot and the EV Point.
  - Check that the Power Supply Input Selector jumper (**JP33**) is properly mounted to select the onboard power supply.
- If you are using local power, ensure that the external power supply is working, and that power-related jumper settings (**JP31** through **JP35**) are correct.

3. My EV Point is not working.

- Is the EV Pilot's Point Status LED (**LED6**, **LED7**, or **LED8**) for the affected EV Point illuminated? If not, check EV Point's power and network connectors. For the EV-Nano Point, register it by pressing **SW5** on the EV Pilot and then the Join button on the EV-Nano Point.
- Check that the LEDs are enabled by ensuring that the appropriate Example I/O Enable jumpers (**JP510**) are properly mounted.
- Try manually resetting point by pressing its Reset button.

4. My EV Pilot or EV Point is not working and its Reset LED is on or blinking; the Network Example HMI shows it as not connected.

- For the EV Pilot or EV-Actuator Point evaluation boards, if you have a JTAG probe connected to the board's JTAG header (**JP504**), ensure that the JTAG Emulator is properly connected to its USB cable and that its USB port is working properly and supplying power to it. Alternately, unplug the JTAG probe from the evaluation board's JTAG header.

5. The application LEDs for the EV-Actuator Point flash continuously in the restart pattern and the Reset LED is blinking.

- There is likely a configuration error for the EV Point, such as a mismatch in the firmware versions between the EV-Actuator Point and the EV Pilot. Try reloading the EV Point's firmware with the binary image found in the [*Pyxos FT EVK*]\Pyxos FT EVK\Pyxos Network Example\ImagesArchive directory, where [*Pyxos FT EVK*] is the directory in which you installed the Pyxos FT EVK software, usually C:\Program Files\Echelon\Pyxos FT EVK. See *Loading Your Application into a Host Processor* on page 95 for information about loading these firmware images.

6. The Reset LED is blinking on the EV-Sensor Point.

- There is likely a configuration error for the EV Point, such as a mismatch in the firmware versions between the EV-Sensor Point and the EV Pilot. Try reloading the EV Point's firmware with the binary image found in the [*Pyxos FT EVK*]\Pyxos FT EVK\Pyxos Network Example\ImagesArchive directory. See *Loading Your Application into a Host Processor* on page 95 for information about loading these firmware images.

7. The Pyxos FT network is behaving oddly, or not at all.

- Verify that the Pyxos FT network cable is properly terminated. If you are using the Pyxos FT network cable that is supplied with the Pyxos FT EVK, the terminator is located on the EV Pilot evaluation board. Ensure that the Network Terminator Connect Enable jumper (**JP36**) is properly mounted.
- Ensure that the Reset Line Enable jumper (**JP506**) is properly mounted.

8. How can I verify communications between the Pyxos FT Chip and its host microprocessor?

- You can connect a Serial Peripheral Interface (SPI) connector to the header connector for accessing host SPI (**J401**) and use a SPI protocol analyzer to monitor communications between the Pyxos FT Chip and the host microprocessor.

9. The Network Example HMI application program cannot connect over USB.

- Check that the EV Pilot is connected to the USB cable and that the USB cable is connected to a working USB port on your computer.
- Check that Windows can communicate with the USB port by checking the port's status in the Device Manger. You can open the Device Manager by opening the Windows Control Panel and double clicking the **System** icon; from the Hardware tab of the System Properties window, click **Device Manager**.
- Check that the USB Interface Communications Connect Enable jumpers (**JP513**) on the EV Pilot are properly mounted.

10. The Network Example HMI application program can't find my LONWORKS network interface.

- Ensure that the network interface is properly installed. For most U10 or *i*.LON network interfaces, double click the **LonWorks Interfaces** icon in the Windows Control Panel to configure and test the network interface.
- For a PCLTA-10, PCLTA-20, or PCC-10 device, ensure that it is configured as a layer-5 network interface. See *Connecting to a LonWorks Network* on page 7 for information about configuring network interfaces as layer-5 interfaces.
- Ensure that no other application is currently using the network interface. If another LONWORKS application is running, either detach it from the network interface or shut down the application.

11. The Network Example HMI application program cannot communicate with the EV Pilot over a LONWORKS connection.

- Check that the LONWORKS Enable jumper (**JP61**) and the ShortStack Micro Server SCI Connect Enable jumpers (**JP512**) are properly mounted.

12. The Network Example HMI application program does not recognize the Service Pin message from the EV Pilot.

- Ensure that you are using a LONWORKS TP/FT-10 network interface, such as an *i*.LON, U10, or a PCLTA-10, PCCLTA-20, or PCC-10 network interface. The Network Example HMI might not recognize Service Pin messages from other types of LONWORKS network interfaces.

7

Developing Pyxos FT Applications Using the Pyxos FT EVK

This chapter describes how to use the Pyxos FT EVK in your development environment for developing custom firmware applications for the Pyxos FT platform.

Setting Up Your Development Environment

Application development for the Pyxos FT platform is similar to application development for other embedded program environments. You need:

- A set of development tools that are specific to the host processors that you are using.
- You will likely need to use the Pyxos FT API and the Pyxos FT Serial API.
- You will use the Pyxos FT Interface Developer utility to maintain your Pyxos projects.
- If your Pyxos application requires USB or LONWORKS communications, you need to include code for such communications in your program.
- Finally, you need to load your application into the host microprocessor to run, test, and debug it.

This section describes these general tasks for developing your Pyxos FT application. See the *Pyxos FT Programmer's Guide* for more information about developing Pyxos FT applications.

Working with Development Tools for Host Processors

To develop custom firmware for your Pyxos FT Pilot and hosted Pyxos FT Points, you need a software development environment for the host microprocessor, including a code editor or Integrated Development Environment (IDE), a C compiler, an assembler, a linker, and a debugger. You should also have an In-Circuit Emulator (ICE) for hardware emulation and program simulation.

This section describes some of the development and debugging tools that are available for the host processors that are used by the Pyxos FT EVK evaluation boards. These development tools are not included with the Pyxos FT EVK.

Tools for the ARM7 Processor

The EV Pilot and the EV-Actuator Point each use an Atmel ARM AT91SAM7S64 microprocessor as their host microprocessor. The AT91SAM7S64 is an ARM7-family microprocessor.

You can use any of the many available tools that support ARM7 firmware development for the microprocessor that you plan to use for your Pyxos Pilot or Points, such as one of the following:

- ARM RealView® Development Suite
  The RealView Development Suite is a set of development tools that support all ARM processors and ARM debug technology. It includes a C/C++ compiler, assembler, linker, virtual platforms, text editor, and debugger. It also allows you to choose an IDE for code development. For more information about ARM RealView, see www.arm.com/products/DevTools/.
- IAR Embedded Workbench®
  The IAR Embedded Workbench is a set of development tools for

programming embedded applications. It includes a C/C++ compiler, assembler, linker, librarian, text editor, project manager, and debugger. For more information about the IAR Embedded Workbench, see www.iar.com/ewarm.

To debug ARM7 firmware applications, you should use a hardware emulator and debugger that supports the microprocessor that you plan to use for your Pyxos Pilot or Points, such as one of the following:

- AT91SAM-ICE JTAG Emulator
  The SAM-ICE is a JTAG emulator designed for Atmel ARM processors. You can get software support for the SAM-ICE from www.segger.com (www.segger.com/downloads.html, under "J-Link ARM / J-Trace / MidasLink"). For more information about the SAM-ICE, see www.atmel.com/dyn/products/tools_card.asp?tool_id=3892 or www.atmel.com/products/AT91/.
- IAR J-Link
  IAR J-Link is a small ARM JTAG hardware debug probe that connects to a Windows computer by a USB connection. You can use IAR J-Link with the IAR Embedded Workbench. For more information about the J-Link, see www.iar.com/jlinkarm.

To load program images (such as the images for the Pyxos Network Example firmware applications that you can find in the [*Pyxos FT EVK*]\Pyxos Network Example\ImagesArchive directory), you can use an In-System Programmer (ISP) such as:

- Atmel SAM-PROG
  The SAM-PROG is included with the AT91 In-System Programmer (ISP), an open set of tools for programming the AT91SAM7 and AT91SAM9 ARM-based microcontrollers. SAM-PROG allows you to directly program your application through a SAM-ICE or a J-Link JTAG Probe. For more information about the AT91ISP and SAM-PROG, see www.atmel.com/dyn/products/tools_card.asp?tool_id=3883.

The Pyxos Network Example firmware applications for the EV Pilot and EV-Actuator Point were developed using the IAR Embedded Workbench. The IAR Embedded Workbench project files for the example firmware applications are included with the firmware source code; for more information on the firmware source code, see *Exploring the Pyxos Network Example* on page 105.

Tools for the AVR Processor

The EV-Sensor Point uses the Atmel AVR ATtiny13 microprocessor as its host processor.

You can use any of the many available tools that support AVR firmware development for the microprocessor that you plan to use for your Pyxos Points, such as one of the following:

- Atmel AVR Studio®
  The AVR Studio is a set of development tools for programming embedded applications in Windows environments. It includes a project management tool, source file editor, and chip simulator. For more information about the AVR Studio, see

www.atmel.com/dyn/products/tools_card.asp?tool_id=2725 or www.atmel.com/products/avr/.

- WinAVR
  WinAVR is a suite of Windows-based open source software development tools for the Atmel AVR series of RISC microprocessors. It includes the GCC C/C++ compiler, assembler, linker, and debugger. For more information about WinAVR, see sourceforge.net/projects/winavr/.

To debug AVR firmware applications, you should use a hardware emulator and debugger that supports the microprocessor that you plan to use for your Pyxos Points, such as:

- Atmel AVR JTAGICE mkII
  The JTAGICE mkII is a development tool for On-chip Debugging of AVR 8-bit RISC microcontrollers. It includes both a JTAG interface and a debugWIRE interface. You can use the JTAGICE mkII with the AVR Studio to control the internal resources of the microcontroller. For more information about the JTAGICE mkII, see www.atmel.com/dyn/products/tools_card.asp?tool_id=3353.

You might also need additional hardware to facilitate loading applications into your AVR microprocessor, such as:

- Atmel AVR STK® 500 Flash Microcontroller Starter Kit
  The AVR STK500 is a starter kit and development system for Atmel's AVR Flash microcontrollers. The STK500 interfaces with AVR Studio, Atmel's Integrated Development Environment (IDE) for code writing and debugging. For more information about the STK500, see www.atmel.com/dyn/products/tools_card.asp?tool_id=2735.

The Pyxos Network Example firmware application for the EV-Sensor Point was developed using the AVR Studio. The AVR Studio project file for the EV-Sensor Point example firmware application is included with the firmware source code; for more information on the firmware source code, see *Exploring the Pyxos Network Example* on page 105.

Working with the Pyxos FT API

The Pyxos FT EVK includes C source code for the Pyxos FT application programming interface (API) which you can use with your applications to access a Pyxos FT network from a Pyxos Pilot, or to interact with a Pyxos Pilot from a Pyxos Point. You can develop simple applications for microprocessors with limited memory space, such as the AVR microprocessor used in the EV-Sensor Point, without the Pyxos FT API; instead you add the minimum required API functionality into the application. For microprocessors with sufficient memory to support the API, the Pyxos FT API can reduce the required application development time for a Pyxos Pilot or Pyxos Point application. To port the Pyxos FT API to a new processor, perform the following steps:

1. Customize the **platform.h** file in the Pyxos FT API include-file directory ([*Pyxos FT EVK*]\Pyxos FT API\Include) to include a section for each of the host microprocessors that you intend to support.
2. Implement the Pyxos FT serial driver for your host platform. See *The ARM7 PS API for the EV-Actuator Point and EV Pilot Examples* on page

133 for information about how the Pyxos Network Example firmware applications implement the Pyxos FT serial driver.

When you create Pyxos FT applications, whether they use the Pyxos FT API or not, you can define the Pyxos FT network interface for your Pyxos FT application with the Pyxos FT Interface Developer utility. Use this utility to specify how your Pilot application interoperates with the Pyxos FT network, and to define a Pyxos Point interface definition for each type of Pyxos Point in your network.

To create Pyxos FT applications that use either the Pilot API or the Point API, perform the following general steps:

1. Create applications for any custom hosted Pyxos Points in your network with the Pyxos FT Point API. A Pyxos Point application typically performs the following tasks:
    a. Initializes the Pyxos FT Point and initialize the Pyxos FT API.
    b. If the Pyxos FT Point uses the hardwired installation mode, announces its timeslot after completing initialization.
    c. Periodically calls the **PyxosPointEventHandler()** function to send and receive updates for the Point to the network.
    d. Reads and writes PNVs, as your application requires.
2. Create the Pilot application for your network if you are developing a custom Pilot. A Pilot application should perform the following tasks:
    a. Initialize the Pyxos Pilot and initialize the Pyxos FT API. The Pilot must complete this task before it performs any other tasks.
    b. Configure the Pyxos Points in the network: receive registration requests from the Pyxos Points, allocate timeslots for them, read their program IDs, and specify their interfaces.
    c. Periodically call the Pyxos event handler to handle Pyxos events.
    d. Receive updates for PNVs from the Pyxos Points on the network, and send PNV updates onto the network.
    e. Monitor the health of the network, and recover from network errors by resetting, reconfiguring, or replacing Pyxos Points when necessary.
3. Load the Pyxos Pilot and Point applications into the host microprocessors that are attached to the Pyxos FT Chips in the network, and begin operating the network.

For more information about using the Pyxos FT API, see the *Pyxos FT Programmer's Guide*.

Using the Pyxos FT Interface Developer Utility

You can use the Pyxos FT Interface Developer utility to configure the Pilot application for your Pyxos FT network, and to create Point interface definitions for the Pyxos Points in the network. To perform either of these tasks, perform the following steps:

1. Create a network project. Set the Pilot options that determine how the Pilot will interact with and manage the Pyxos FT network.
2. Create or import the point interface definitions for the Pyxos FT Points in your network. Create a separate Point interface definition for each type of custom Pyxos Point that exists in your network. For each Point interface definition, define the Pyxos network variables that you want each Pyxos Point to contain, and select the installation mode for the Pyxos Points to use. Pyxos Point manufacturers should provide Point interface definitions for their Points.
3. Set advanced options for the Pyxos FT network, including the project settings and Pilot options.
4. Generate the C header files for the network. These files contain macros that control the implementation of the Pyxos Point and Pilot interface definitions. Include these header files in your custom Pyxos Point and Pilot applications.

For more information about using the Pyxos FT Interface Developer utility, see the *Pyxos FT Programmer's Guide*.

Including LONWORKS Support in Your Pyxos FT Applications

You can develop a device that is both a Pyxos Pilot and a LONWORKS device. Pyxos FT networks and LONWORKS networks use the same data types, which simplifies integration of data points on Pyxos Points with other LONWORKS devices, or with other Pyxos FT networks that also include LONWORKS integration. To connect your Pyxos Pilot to a LONWORKS network, complete the following general tasks:

- Create a Neuron C model file of the network-variable data for Pyxos objects. You can use the NodeBuilder® 3.1 Development Tool Code Wizard to generate the Neuron C model file, or you can create one manually.
- Generate network-variable data files for the ShortStack Micro Server. You can use the ShortStack Developer's Kit to generate these files.
- Develop a serial driver for your host application that communicates with the ShortStack Micro Server firmware in a Smart Transceiver.
- Develop a program that handles the communications between the Pyxos FT network and the LONWORKS network. This program works with the serial driver to enable communications.

See *Using the Pyxos-LonWorks Gateway* on page 121 for more information about the example Pyxos LONWORKS Gateway application that is included with the Pyxos Network Example.

Because a Pyxos FT network can often send small amounts of data more quickly than a LONWORKS network can, your networking program must send data to the LONWORKS network at an appropriate rate, that is, you might need to impose a throttle on the propagation of LONWORKS network variables. One approach for imposing such a throttle is to store network-variable updates in a data structure, and to allow your networking program to send the updates to the network in a

round-robin fashion, while ensuring that only one update at a time is sent to the ShortStack Micro Server.

When you use the ShortStack Developer's Kit to generate NV data for Pyxos objects, you might need to make certain manual changes to the generated files for correct operation in your environment; see *ShortStack Interface and Serial Driver* on page 126 for examples of the necessary changes for the EV Pilot Examples firmware.

The ShortStack Developer's Kit includes the ShortStack API, the ShortStack MicroServer library, documentation, and examples. Download the ShortStack Developer's Kit from www.echelon.com/shortstack.

Important: The Pyxos FT EVK has been developed and tested with the ShortStack 2 Micro Server; changes may be required for different releases of the ShortStack firmware.

If you need to reload the ShortStack Micro Server, you can use the NodeLoad utility (available from the Echelon Web site, www.echelon.com) to load the system images from the [*Pyxos FT EVK*]\Pyxos FT EVK\Pyxos Network Example\ImagesArchive\Pilot ShortStack MicroServer directory.

See the *ShortStack User's Guide* for more information about developing ShortStack applications. See the *LONWORKS Host Application Programmer's Guide* for more information about developing LONWORKS networking applications.

Loading Your Application into a Host Processor

The EV Pilot and EV-Actuator Point each include a JTAG-compliant header connector that provides external access to the ARM7 host microprocessor's functions and memory. You can use a hardware emulator, such as Atmel's AT91SAM-ICE JTAG Emulator to connect to this header and load your programs into the ARM7 memory. You can also use the JTAG header for remote debugging of the ARM7 application program or for an in-circuit emulator (ICE).

Because the ATtiny13 microprocessor does not support the JTAG interface, the EV-Sensor Point does not include a JTAG-compliant header connector. You can load your programs into the host microprocessor using a flash programming board or the ATtiny13 microprocessor's debugWIRE interface.

If your Pyxos Point application uses a different host microprocessor, you can connect it to the EV-Sensor Point using the board's External Host SPI header connector (**JP401**).

Loading the ARM AT91SAM7S64 Microprocessor

To load your programs into the AT91SAM7S64 host microprocessor in the EV Pilot and EV-Actuator Point, perform the following general steps:

1. Connect a hardware emulator and debugger, such as the AT91SAM-ICE JTAG Emulator, to the evaluation board's JTAG header connector (**JP504**).
2. Load the program from your software development environment, such as the IAR Embedded Workbench.

3. Disconnect the hardware emulator and debugger from the evaluation board.

You can also use an In-System Programmer (ISP) to load program images (such as the images for the Pyxos Network Example firmware applications that you can find in the [*Pyxos FT EVK*]\Pyxos Network Example\ImagesArchive directory). Atmel's SAM-PROG allows you to directly program your application through a SAM-ICE or a J-Link JTAG Probe. To obtain SAM-PROG, download the latest version of the Atmel AT91ISP (which includes SAM-PROG) from www.atmel.com/dyn/products/tools_card.asp?tool_id=3883.

Loading the AVR ATtiny13 Microprocessor

You can load your custom firmware application (or the image for the EV-Sensor Point example firmware application that you can find in the [*Pyxos FT EVK*]\Pyxos Network Example\ImagesArchive directory) into the EV-Sensor Point ATtiny13 microprocessor using either of the following methods:

- Use the ATtiny13 microprocessor's 6-wire programming interface (the In-System Programming (ISP) interface) by moving the microprocessor to a flash programming board.
- Use the ATtiny13 microprocessor's 3-wire programming interface (the debugWIRE interface) by connecting a hardware emulator and debugger to the EV-Sensor Point Remote Programming header (**JP51**).

Using a Flash Programming Board

To load your programs into the ATtiny13 host microprocessor using a flash programming board, perform the following general steps:

1. Remove the ATtiny13 chip from the EV-Sensor Point evaluation board using a Plastic Leadless Chip Carrier (PLCC) extraction tool (or similar device).
2. Insert the ATtiny13 chip into a flash programming board, such as the Atmel AVR STK500 Flash Microcontroller Starter Kit, using an integrated-circuit inserter tool (or similar device).
3. Connect the flash programming board to a hardware emulator and debugger, such as the Atmel AVR JTAGICE mkII.
4. Load the program from your software development environment, such as the Atmel AVR Studio.
5. Remove the ATtiny13 chip from the flash programming board.
6. Insert the ATtiny13 chip into the EV-Sensor Point evaluation board.

Ensure that the flash programming board is properly configured for programming the ATtiny13 microprocessor. See *Configuring the AVR STK500* for information about configuring the Atmel AVR STK500 Flash Microcontroller Starter Kit flash programming board for the ATtiny13 microprocessor and the Atmel AVR JTAGICE mkII hardware emulator and debugger.

Important: When programming the ATtiny13 microprocessor with AVR Studio connected to a JTAGICE mkII emulator and debugger, set the ISP frequency to 256 kHz.

Using the debugWIRE Interface

To load your programs into the ATtiny13 host microprocessor using the debugWIRE interface, perform the following steps:

1. Enable the debugWIRE interface by programming the debugWIRE Enable (DWEN) Fuse bit on the ATtiny13 chip. You must use a flash programming board, such as the Atmel AVR STK500 Flash Microcontroller Starter Kit, to program this fuse bit.
2. Connect a hardware emulator and debugger, such as the Atmel AVR JTAGICE mkII, to the EV-Sensor Point Remote Programming header (**JP51**).
3. Dismount the EV-Sensor Point Reset Line Enable jumper (**JP52**) to disconnect the RESET~ pin of the ATtiny13 chip from the RST~ pin of the Pyxos FT Chip.
4. Load the program from your software development environment, such as the Atmel AVR Studio.
5. Disable the debugWIRE interface by programming the debugWIRE Enable (DWEN) Fuse bit on the ATtiny13 chip. You must use a flash programming board, such as the Atmel AVR STK500 Flash Microcontroller Starter Kit, to program this fuse bit.
6. Mount the EV-Sensor Point Reset Line Enable jumper (**JP52**) to reconnect the RESET~ pin of the ATtiny13 chip to the RST~ pin of the Pyxos FT Chip.

It is possible to run your firmware application on the ATtiny13 microprocessor while the debugWIRE interface is enabled, but Atmel recommends that the debugWIRE interface be disabled when not in use. In addition, while the debugWIRE interface is enabled, the External Reset function for the ATtiny13 chip is disabled; thus, when you reset the EV-Sensor Point, the ATtiny13 microprocessor will not reset. In this case, communications with the EV Pilot can be lost, and it might be necessary to reset the ATtiny13 microprocessor manually to re-establish communications.

Recommendation: Because you must enable and disable the debugWIRE interface before you can program the ATtiny13 microprocessor, use a flash programming board to load your firmware applications into the ATtiny13 microprocessor, as described in *Using a Flash Programming Board* on page 96.

Configuring the AVR STK500

If you plan to use the Atmel AVR STK500 Flash Microcontroller Starter Kit flash programming board with the Atmel AVR JTAGICE mkII hardware emulator and debugger to program the ATtiny13 microprocessor for the EV-Sensor Point, you must ensure that the STK500 flash programming board is properly configured.

Some of the configuration described in this section configures the board for Atmel's Serial High-Voltage Programming of the ATtiny13 microprocessor, and some of it configures the board for use with the JTAGICE mkII hardware emulator and debugger.

To configure the STK500 flash programming board for programming the ATtiny13 microprocessor with the JTAGICE mkII hardware emulator and debugger, perform the following steps:

1. Connect the wires of the included squid cable to the blue SPROG1 target In-System Programming (ISP) header as described in **Table 6**.

Table 6. JTAGICE mkII Probe and STK500 ISP Header Connections

Squid Cable Wire	JTAGICE mkII Probe Pins	STK500 ISP Header Pins
Black	Pin 1 Test clock (TCK)	Pin 3 Serial clock (SCK)
White	Pin 2 Ground (GND)	Pin 6 Ground (GND)
Grey	Pin 3 Test data output (TDO)	Pin 1 Master In Slave Out (MISO) serial output
Purple	Pin 4 Target reference voltage (VTref)	Pin 2 Supply voltage (VCC)
Green	Pin 6 Target system reset I/O (nSRST)	Pin 5 Reset input (RESET)
Red	Pin 9 Test data input (TDI)	Pin 4 Master Out Slave In (MOSI) serial input
Note: You will not use the blue (pin 5), yellow (pin 7), orange (pin 8), or brown (pin 10) wires of the squid cable connector.		

2. Mount the onboard oscillator (OSCSEL) jumper to pins 2 and 3. This setting enables the onboard crystal signal. Pin 1 is the right-most pin.
3. Mount the onboard system clock (XTAL1) jumper to pins 1 and 2 (the default setting). This setting routes the oscillator signal to the device.
4. Mount the VTARGET jumper to pins 1 and 2 (the default setting). This setting provides reference voltage to the AVR device, and can be controlled from the AVR Studio program.
5. Dismount the RESET jumper. This setting is necessary when using the JTAGICE mkII hardware emulator and debugger.
6. Dismount the Byte Select 2 (BSEL2) jumper. This setting is not used for the ATtiny13 microprocessor.
7. Connect PORTB header pins to PORTE/AUX header pins, as described in **Table 7**.

Table 7. PORTB to PORTE/AUX Header Connections

PORTB Header Pin	PORTE/AUX Header Pin	Function
PB3 Pin 4	XT1 Pin 7	Connect the system clock to the AVR device.
PB5 Pin 6	RST Pin 4	Connect the onboard reset system to the AVR device.

8. Connect the header connector of the squid cable to the JTAGICE mkII probe.

Insert the ATtiny13 chip into the blue target socket (SCKT3400D1) for programming.

Important: Ensure that that ATtiny13 chip is the only chip mounted into any of the sockets on the STK500 flash programming board. Having another chip (such as the AT90S8515-8PC microcontroller that comes with the STK500) mounted into any of the programming sockets can interfere with the software development environment's ability to program the ATtiny13 chip.

When the STK500 flash programming board is properly configured, you can use the AVR Studio program to load your firmware program into the ATtiny13 microprocessor.

See the *AVR STK500 User Guide* for more information about configuring and using the STK500 flash programming board.

See the *AVR JTAG ICE User Guide* and the *JTAGICE mkII Quick Start Guide* for more information about the JTAGICE mkII. You can find additional information for the JTAGICE mkII in the online help for the AVR Studio.

Debugging Your Application

The process of debugging Pyxos FT applications is similar to the process of debugging other embedded applications. In general, you test the application using simulated hardware or prototype hardware, then you test the application with the final target hardware. For Pyxos FT applications, you also need to test the application while it is connected to the Pyxos FT network to ensure that the application correctly uses the Pyxos FT API.

Certain host microprocessors might impose specific requirements or restrictions for program debugging. For example, the host microprocessor might use the debugWIRE interface in addition to, or instead of, the JTAG interface, or the microprocessor might have restrictions for setting multiple program break points.

Debugging for the ARM AT91SAM7S64 Microprocessor

To debug programs running on the AT91SAM7S64 host microprocessor that is included with the EV Pilot and EV-Actuator Point, perform the following general steps:

1. Connect a hardware emulator and debugger, such as the AT91SAM-ICE JTAG Emulator, to the evaluation board's JTAG header connector (**JP504**).
2. Run and debug the program using your software development environment, such as the IAR Embedded Workbench.
3. Disconnect the hardware emulator and debugger from the evaluation board.

Debugging for the AVR ATtiny13 Microprocessor

To debug programs running on the EV-Sensor Point ATtiny13 host microprocessor, perform the following general steps:

1. Enable the ATtiny13 microprocessor's debugWIRE interface by programming the debugWIRE Enable (DWEN) Fuse bit on the ATtiny13 chip. You must use a flash programming board, such as the Atmel AVR STK500 Flash Microcontroller Starter Kit, to program this fuse bit.
2. Connect a hardware emulator and debugger, such as the Atmel AVR JTAGICE mkII, to the EV-Sensor Point Remote Programming header (**JP51**).
3. Dismount the EV-Sensor Point Reset Line Enable jumper (**JP52**) to disconnect the RESET~ pin of the ATtiny13 chip from the RST~ pin of the Pyxos FT Chip.
4. Run and debug the program from your software development environment, such as the Atmel AVR Studio.
5. Disable the debugWIRE interface by programming the debugWIRE Enable (DWEN) Fuse bit on the ATtiny13 chip. You must use a flash programming board, such as the Atmel AVR STK500 Flash Microcontroller Starter Kit, to program this fuse bit.
6. Mount the EV-Sensor Point Reset Line Enable jumper (**JP52**) to reconnect the RESET~ pin of the ATtiny13 chip to the RST~ pin of the Pyxos FT Chip.

It is possible to run your firmware application on the ATtiny13 microprocessor while the debugWIRE interface is enabled, but Atmel recommends that the debugWIRE interface be disabled when not in use. In some situations, the EV Pilot can lose communications with the Point and be unable to reconfigure the Point. In this case, manually reset the ATtiny13 microprocessor to re-establish communications with the Pilot.

Enabling the debugWIRE Interface

The debugWIRE interface provides a one-wire, bi-directional interface that allows you to control program flow and to program the ATtiny13 microprocessor's non-volatile memory.

To enable the ATiny13 microprocessor's debugWIRE interface, perform the following steps:

1. Remove the ATtiny13 chip from the EV-Sensor Point evaluation board using a Plastic Leadless Chip Carrier (PLCC) extraction tool (or similar device).
2. Insert the ATtiny13 chip into a flash programming board, such as the Atmel AVR STK500 Flash Microcontroller Starter Kit, using an integrated-circuit inserter tool (or similar device).
3. Connect the flash programming board to a hardware emulator and debugger, such as the Atmel AVR JTAGICE mkII.
4. Enable the debugWIRE interface by programming the debugWIRE Enable (DWEN) Fuse bit on the ATtiny13 chip. The interface is enabled when DWEN=0.

   You can program this fuse bit from your software development environment, such as the Atmel AVR Studio.
5. Remove the ATtiny13 chip from the flash programming board.
6. Insert the ATtiny13 chip into the EV-Sensor Point evaluation board.

While the debugWIRE interface is enabled, and the ATtiny13 microprocessor is installed on the EV-Sensor Point evaluation board, you can connect a hardware emulator and debugger to the evaluation board and use your software development environment, such as the Atmel AVR Studio, to load, run, and debug the program.

To disable the debugWIRE interface, following the same steps as those described above. The interface is disabled when DWEN=1.

Important: Because the debugWIRE communication pin (dW) on the ATtiny13 microprocessor is physically located on pin 1, which is the same pin as its External Reset (RESET) function, you cannot use the External Reset function while the debugWIRE interface is enabled. For the EV-Sensor Point, the ATtiny13 microprocessor's pin 1 (the External Reset (RESET~) pin) is connected to the Pyxos FT Chip's pin 19 (the Reset (RST~) pin), so that while the debugWIRE interface is enabled, when you press the Reset button on the EV-Sensor Point, the Pyxos FT Chip resets, but the ATtiny13 microprocessor does not.

Additionally, when the EV Pilot sends a reset message (for example, during an error-recovery scenario) to the EV-Sensor Point while the debugWIRE interface is enabled, the EV-Sensor Point successfully resets the Pyxos FT Chip, but does not reset the ATtiny13 microprocessor. Thus, your firmware program must perform the reset function for the ATtiny13 microprocessor while the debugWIRE interface is enabled.

Important: Disconnect the Pyxos FT Chip's Reset pin from the ATtiny13 microprocessor's External Reset pin while you are using the debugWIRE

interface. To disconnect these two pins, dismount the EV-Sensor Point Reset Line Enable jumper (**JP52**).

Be sure to re-mount the Reset Line Enable jumper (**JP52**) after you disable the debugWIRE interface to ensure that your firmware program operates correctly.

The debugWIRE system accurately emulates all I/O functions while the processor is running. However, when the microprocessor is stopped, if you access the I/O registers through a debugger (such as the AVR Studio), you should verify your results while the microprocessor is running.

See the debugWIRE documentation for a description of other programming and debugging considerations.

Connecting a Hardware Emulator and Debugger to the EV-Sensor Point Evaluation Board

You can connect a hardware emulator and debugger to the EV-Sensor Point evaluation board to debug your custom firmware application while the ATtiny13 microprocessor has its debugWIRE interface enabled.

To connect the Atmel AVR JTAGICE mkII to the EV-Sensor Point evaluation board, perform the following steps:

1. Connect the wires of a squid cable to the EV-Sensor Point Remote Programming header (**JP51**), as described in **Table 8**. The colors of the squid wire correspond to those of the squid wire cable that is included with the AVR JTAGICE mkII hardware emulator and debugger.
2. Connect the header connector of the squid cable to the JTAGICE mkII probe.
3. Dismount the EV-Sensor Point Reset Line Enable jumper (**JP52**) to disconnect the Pyxos FT Chip's Reset pin from the ATtiny13 microprocessor's External Reset pin while you are using the debugWIRE interface.

Be sure to re-mount the EV-Sensor Point Reset Line Enable jumper (**JP52**) after you disable the debugWIRE interface to ensure that your firmware program operates correctly.

Table 8. JTAGICE mkII Probe and EV-Sensor Point Remote Programming Header (**JP51**) Connections

Squid Cable Wire	JTAGICE mkII Probe Pins	EV-Sensor Point Remote Programming Header Pins
White	Pin 2 Ground (GND)	Pin 3 Ground (GND)
Purple	Pin 4 Target reference voltage (VTref)	Pin 1 Supply voltage (VCC)

Squid Cable Wire	JTAGICE mkII Probe Pins	EV-Sensor Point Remote Programming Header Pins
Green	Pin 6 Target system reset I/O (nSRST)	Pin 2 Reset (RST)
Note: You will not use any of the other wires of the squid cable connector.		

8

Exploring the Pyxos Network Example

This chapter describes the C code that comprises the firmware example applications that are loaded in the host microprocessors for the EV Pilot and the hosted Points (the EV-Actuator Point and EV-Sensor Point).

Design Overview for the Pyxos Network Example

The Pyxos Network Example defines a Pyxos FT network with a Pilot, two hosted Points, and one unhosted Point, as described in Chapter 4, *Using the Pyxos Network Example*, on page 63. This chapter describes the firmware implementation for the Network Example.

The firmware application programs for the Pyxos Network Example are written in standard ANSI C, with some additional programming framework to support the host processors for the Pyxos Pilot and Points. If you are familiar with Echelon's Neuron C language, certain programming concepts (such as network variables) and certain aspects of the communication model might already be familiar to you. However, you do not need to know Neuron C to develop Pyxos FT applications.

The overall design for the Pyxos Network Example firmware applications encompasses the following programming models:

- Master/subordinate: Each of the EV Points performs a specific set of functions and acts independently of each other. The Points send their data to the Pilot and receive their data from the Pilot. Thus, the Pilot acts as the master, communicating with all of the Points and managing their actions.
- Event-driven: When an event occurs on one of the EV Points, its firmware application program detects that event and sends the data for the event to the EV Pilot. When the EV Pilot application program receives notification of an event from one of the EV Points, or when an event occurs on the EV Pilot itself, the Pilot's firmware application program performs some action; often, that action is to illuminate an LED on the EV Pilot evaluation board and to send a message to one of the EV Points to perform some similar action.
- Time-division multiplexing: The devices in the Pyxos FT network share the communications channel by dividing the communications signal into discrete timeslot intervals. Each device in the network is assigned a unique timeslot ID, and each device sends and receives data only during its assigned timeslot. The EV Pilot application program is responsible for managing the timeslots and for the communication with the EV Points in the network.

The Pyxos Network Example firmware programs are pre-loaded into the host microprocessors for the EV Pilot, EV-Actuator Point, and EV-Sensor Point. The source files for the example firmware programs are installed into a set of directories on your computer when you install the Pyxos FT software. You can use this source code to learn how to write your own Pyxos Pilot firmware or Pyxos Point firmware, or you can modify the source code to change the behavior of the Pyxos Network Example application.

The source files for the Pyxos Network Example are contained in the following top-level directory:

[*Pyxos FT EVK*]\Pyxos Network Example

where [*Pyxos FT EVK*] is the directory in which you installed the Pyxos FT EVK software, usually **C:\Program Files\Echelon\Pyxos FT EVK**. Table 9 on page 107 lists the top-level directories for the example firmware and describes the

contents of these directories. The rest of this chapter describes the example firmware and the files that make up the Pyxos Network Example.

Table 9. Directory Structure and Contents for Network Example

Directory	Contents
\Pyxos Network Example	• Overall project file (pyxos network example.PxPrj) for the Pyxos Network Example, created with the Pyxos FT Interface Developer utility. • Project files for each Point (*.PxIntf), created with the Pyxos FT Interface Developer utility. • C header files (*.h), generated from the project files.
\Pyxos Network Example\ActuatorPoint	• C source and header files for the EV-Actuator Point example. • Project files for the IAR Embedded Workbench development environment. • Options file for the Pyxos FT Interface Developer utility to define implementation-specific options for the EV-Actuator Point example.
\Pyxos Network Example\AT91SAM7S64	Directories that contain the shared files for the ARM7 development environment, including the modified Pyxos FT Serial API (see Chapter 9, *The ARM7 PS API for the EV-Actuator Point and EV Pilot Examples* on page 133).
\Pyxos Network Example\ImagesArchive	Compiled binary executable files of the factory-shipped example firmware programs, which you can use to reload the example firmware on the evaluation boards.
\Pyxos Network Example\Pilot	• C source and header files for the EV Pilot example. • Project files for the IAR Embedded Workbench development environment. • Options file for the Pyxos FT Interface Developer utility to define implementation specific options for the EV Pilot example. • A directory for LONWORKS support.

Directory	Contents
\Pyxos Network Example\SensorPoint	• C source and header file for the EV-Sensor Point example. • Project file for the AVR Studio development environment.

The Pyxos FT EVK example firmware applications were developed using the following tools: IAR Embedded Workbench for the ARM7 firmware, and AVR Studio for the ATtiny13 firmware. You can use any development tools that meet the requirements of your host microprocessor, but to use the included development-environment project files, you must use these tools.

The Pyxos FT EV-Actuator Point Example

The EV-Actuator Point example firmware application program controls digital and analog output for the Pyxos Network Example application.

The EV-Actuator Point example firmware uses the Pyxos FT Point API to manage communications with the Pyxos FT Chip.

Design

The EV-Actuator Point example firmware has a simple design. It includes a **main()** function, an implementation of the Pyxos network variable (PNV) update function that is required by the Pyxos FT API, and I/O definitions and functions for interfacing with the ARM7 host microprocessor.

The EV-Actuator Point Example firmware's **main()** function performs the following tasks:

1. Initializes the digital inputs.
2. Initializes the Pyxos FT API.
3. Runs an infinite loop to repeatedly perform the following tasks:
    a. Read each of the digital input values.
    b. If the value of an input has changed since the last time through the loop:
        i. Convert the value to a **SNVT_switch** data structure.
        ii. Call the Pyxos FT Point API to update the value of the PNV in the Pyxos FT Chip.
    c. Call the Pyxos FT Point API event handler to send any updated PNV values to the EV Pilot and receive any updated PNV values from the EV Pilot.

The EV-Actuator Point example firmware does not interpret any of the digital input values, nor does it take independent action based on the values. The EV Pilot example firmware interprets the data values and instructs the EV-Actuator Point example to take any necessary actions, such as activating an LED.

The EV-Actuator Point example firmware also does not handle the analog output setting directly, but sets the current analog output value based on PNV values received from the EV Pilot example firmware.

The EV-Actuator Point example firmware I/O definitions and functions for the ARM7 microprocessor include:

- Pin definitions and I/O mappings
- Get and set functions for the digital and analog I/O (including the push buttons, LEDs, and analog output value)
- Utility functions that are called by the EV-Actuator Point firmware's **main()** function and by the EV Pilot firmware

Interface

The EV-Actuator Point supports eight LEDs, four additional digital outputs (available on a header), and five switches. The PNVs are declared in the order of the physical switches and LEDs on the evaluation board, from left to right. The example firmware uses the Pyxos FT Point API.

The [*Pyxos FT EVK*]\Pyxos Network Example**EchelonEx_ActuatorPointInterface.h** file (which is generated from [*Pyxos FT EVK*]\Pyxos Network Example**EchelonEx_Actuator.PxIntf** by the Pyxos FT Interface Developer utility) defines the PNVs for the EV-Actuator Point. The PNVs include:

- Eight input PNVs for the LEDs:
  - CONNECTED_LED
  - DRY_CONTACT_LED
  - OVER_TEMP_LED
  - LOW_LIGHT_LED
  - PERFORMANCE_DEMO_LED
  - DO_STATUS_LED
  - USER_LED_1
  - USER_LED_2
- Four input PNVs for the digital outputs:
  - DO_1
  - DO_2
  - DO_3
  - DO_4
- One input PNV for the analog output:
  - AO
- Five output PNVs for the push buttons:
  - DRY_CONTACT_BUTTON
  - USER_BUTTON_1

    - USER_BUTTON_2
    - USER_BUTTON_3
    - USER_BUTTON_4
- One output PNV for the version number of the EV-Actuator Point firmware:
    - VERSION

The interface file also defines the program ID (PID) for the EV-Actuator Point.

In **myPyxosApplication.h**, the registration mode for the Actuator Point is defined as automatic. This header file is included in the [*Pyxos FT EVK*]\Pyxos API\Include**Pyxos.h** Pyxos FT API header file.

Source Files and Project Files

In addition to the files in the [*Pyxos FT EVK*]\Pyxos Network Example directory that are listed in Table 9 on page 107 (specifically, the **EchelonEx_ActuatorPointInterface.h** file, which is generated from **EchelonEx_Actuator.PxIntf** by the Pyxos FT Interface Developer utility), the EV-Actuator Point example uses the files in the [*Pyxos FT EVK*]\Pyxos Network Example\ActuatorPoint directory that are listed in Table 10.

Table 10. File Contents for EV-Actuator Point Example

File	Contents
ActuatorPointApp.c ActuatorPointApp.h	C source and header files for the main application of the EV-Actuator Point example, including main control loop and event handlers.
ActuatorPointIo.c ActuatorPointIo.h	C source and header files for the code used by the EV-Actuator Point example to access digital inputs, digital outputs, and analog output supported by the EV-Actuator Point.
EvkActuatorPoint.ewd EvkActuatorPoint.ewp EvkActuatorPoint.eww	Project files for the IAR Embedded Workbench development environment.
MyPyxosApplication.h	C header file for the EV-Actuator Point example; generated by the Pyxos FT Interface Developer utility.
MyPyxosApplication.PxOpts	Application options file for the EV-Actuator Point example, generated by the Pyxos FT Interface Developer utility.

File	Contents
resources.h	C header file for the LONWORKS definitions that are used by the EV-Actuator Point example; generated by the Pyxos FT Interface Developer utility.

The project files for the IAR Embedded Workbench development environment for the EV-Actuator Point include two configurations to control conditional compilation for different environments: Flash Debug and Flash Release. These configurations allow you to control whether to include debug control code.

The Pyxos FT EV-Sensor Point Example

The EV-Sensor Point example firmware monitors three onboard sensor devices for the Pyxos Network Example application. The sensors measure temperature, light levels, and analog input.

The EV-Sensor Point example firmware does not use the Pyxos FT Point API to manage communications with the Pyxos FT Chip, but instead manages the communication with the Pyxos FT Chip directly by using a software bit-bang technique.

Design

Because the EV-Sensor Point example firmware does not use the Pyxos FT Point API, it might appear more complex than the EV-Actuator Point example firmware, but the application design is still relatively simple. It includes a **main()** function and a number of implementations of functions that are required by the AVR Serial Peripheral Interface (SPI), including I/O definitions and functions for interfacing with the AVR host microprocessor. It also calls functions that provide an interface with the Pyxos FT Chip.

The EV-Sensor Point example firmware's **main()** function performs the following tasks:

1. Configures the **PORTB** pins on the ATtiny13 microprocessor for I/O, and waits for the SPI to complete initialization.
2. Calls an SPI function to write the Sensor Point's program ID to the Pyxos FT Chip.
3. Calls an SPI function to write an initial value (to specify that the EV-Sensor Point is a hosted Point, that it uses the automatic registration mode, and that it is ready) to the Pyxos FT Chip configuration register, and waits until the Pyxos FT Chip confirms that the Point is configured.
4. Calls an SPI function to write the unique ID (UID) to the Pyxos FT Chip, and waits for the Pyxos FT Chip to acknowledge receipt of the UID. Sending the UID as the first data from the EV-Sensor Point to the EV Pilot ensures that the EV-Sensor Point UID synchronizes with the EV Pilot's transaction ID.
5. Calls an SPI function to write a value to the Pyxos FT Chip **POINT_READY** register, and waits for it to be acknowledged. Writing to

this register signifies that the Point is configured and ready to accept PNV updates.

6. Calls an SPI function to check that the Pilot has sent a value to the Pyxos FT Chip **SET_POINT_ONLINE** register. The Pilot writes to this register when it is ready for the Point to begin sending PNV updates.
7. Calls an SPI function to write the firmware version number PNV to the Pyxos FT Chip.
8. Runs an infinite loop to repeatedly perform the following tasks:
    a. Determine whether the sensor values have been updated recently.
    b. If the sensor values have been updated, determine whether to read the lux sensor value or the temperature sensor value. Each iteration through this infinite loop alternates between reading either the lux value or the temperature value.
    c. If the lux sensor value is the one to process, read the lux value from the analog-to-digital converter (ADC) on the ATtiny13 microprocessor input for the light sensor. Then, convert this value to a SNVT format.
    d. If the temperature sensor value is the one to process, read the temperature value from the ADC converter on the ATtiny13 microprocessor input for the temperature sensor. Then, convert this value to a SNVT format.
    e. Read the analog-input voltage value from the ADC converter on the ATtiny13 microprocessor input for the analog input line. Then, convert this value to a SNVT format. Each iteration through the infinite loop reads the analog-input voltage value so that the value is always current for the Pyxos Network Example HMI application program's Performance Demo.
    f. Call an SPI function to determine whether there are pending updates, that is, updates that the EV-Sensor Point example firmware has written to the Pyxos FT Chip but that the EV Pilot has not yet acknowledged.
    g. If there are no pending updates, write either the lux or temperature SNVT value to the Pyxos FT Chip and write the analog-input SNVT value to the Pyxos FT Chip.

The EV-Sensor Point example firmware does not interpret any of the sensor values, nor does it take independent action based on the values. The EV Pilot example firmware interprets the data values and instructs the EV-Sensor Point example to take any necessary actions.

Interface

The EV-Sensor Point evaluation board includes three sensors: a temperature sensor, a light sensor, and a voltage sensor.

The example firmware defines a 2-byte output PNV for each of the three sensors. The firmware reads the sensor values, converts them to standard formats, and

then sends them out onto the Pyxos FT network. The application does not use the Pyxos FT Point API.

The [*Pyxos FT EVK*]\Pyxos Network Example**EchelonEx_SensorPointInterface.h** file (which is generated from [*Pyxos FT EVK*]\Pyxos Network Example**EchelonEx_Sensor.PxIntf** by the Pyxos FT Interface Developer utility) defines the PNVs for the EV-Sensor Point. The PNVs include:

- Three output PNVs for the sensors:
  - LUX
  - TEMP
  - AI
- One output PNV for the version number of the EV-Sensor Point firmware:
  - VERSION

The interface file also defines the program ID (PID) for the EV-Sensor Point.

Source Files and Project Files

In addition to the files in the [*Pyxos FT EVK*]\Pyxos Network Example directory that are listed in Table 9 on page 107 (specifically, the **EchelonEx_SensorPointInterface.h** file, which is generated from **EchelonEx_Sensor.PxIntf** by the Pyxos FT Interface Developer utility), the Sensor Example uses the files in the [*Pyxos FT EVK*]\Pyxos Network Example\SensorPoint directory that are listed in Table 11.

Table 11. File Contents for EV-Sensor Example

File	Contents
resources.h	C header file for the LONWORKS definitions that are used by the EV-Sensor Point example; generated by the Pyxos FT Interface Developer utility.
EvkSensorPoint.aps	Project file for the AVR Studio development environment.
SensorPoint.c SensorPoint.h	C source and header files for the main application for the EV-Sensor Point example, including main control loop, drivers, event handlers, and definitions for the SPI interface.

The project files for the AVR Studio development environment for the EV-Sensor Point includes one configuration: default. You can create additional configurations to control whether to include debug control code.

Note: Because the EV-Sensor Point's example firmware does not use the Pyxos FT Point API, there is no Point application options file (***.PxOpts** file) generated by the Pyxos FT Interface Developer utility for it. If you open the overall project

file for the Network Example (**pyxos network example.PxPrj** file in the [*Pyxos FT EVK*]\Pyxos Network Example directory), select the Ex_Sensor Point, and click the **Point Application Options** button, the settings displayed in the Point Application Options dialog represent default settings for the hosted Points rather than the actual settings used by the EV-Sensor Point example firmware.

The Pyxos FT EV-Nano Point Example

Because the EV-Nano Point is an unhosted Point, it does not have a host microprocessor or a firmware application. The EV Pilot controls all of the functions of the EV-Nano Point. See the [*Pyxos FT EVK*]\Pyxos Network Example\Pilot**PilotNanoPoint.c** file for an example of the actions that the EV Pilot performs.

The [*Pyxos FT EVK*]\Pyxos Network Example**EchelonEx_NanoPointInterface.h** file (which is generated from [*Pyxos FT EVK*]\Pyxos Network Example**EchelonEx_Nano.PxIntf** by the Pyxos FT Interface Developer utility) defines the interface for the EV-Nano Point. This interface includes I/O masks that the EV Pilot uses to monitor and control the I/O on the EV-Nano Point. These I/O masks include:

- Two input masks for the buttons:
  - DRY_CONTACT_BUTTON
  - USER_BUTTON
- Two output masks for the LEDs:
  - DRY_CONTACT_LED
  - USER_LED

The EV-Nano Point has no program ID (PID); the PID for unhosted Points is always zero.

The Pyxos FT EV Pilot Example

The EV Pilot controls the Pyxos FT network, including handling communications with the Pyxos Points in the Pyxos FT network and handling communications with a Windows computer or with a LONWORKS network. The EV Pilot receives all updates to Pyxos network variables (PNVs) from the EV Points, and takes appropriate action for each update, such as activating an LED or sending the updated value to a computer or to a LONWORKS network.

The EV Pilot also handles registration for each EV Point in the Pyxos FT network, and maintains current status for each Point. If the status for an EV Point becomes unacceptable, the EV Pilot tries to reestablish communications with the Point, and if it cannot, the EV Pilot instructs the EV Point to reset.

The EV Pilot's example firmware also includes a number of utility functions for the Pilot and the Points, as well as functions for interacting with its ARM7 host microprocessor.

The EV Pilot example firmware uses the Pyxos FT Pilot API to manage communications with the Pyxos FT Chip.

Design

The EV Pilot example firmware is more complex than the EV Point examples because the Pilot acts as the controller for all of the EV Points in the Pyxos FT network. It includes several files that encapsulate functionality into discrete areas, such as processing for the EV Pilot itself, handling I/O functions for the EV Pilot and EV Points, managing registration for the EV Points, processing for each type of Point, interfacing with the ARM7 microprocessor, handling USB connections, and managing the LONWORKS Gateway. The EV Pilot uses the Pyxos FT Pilot API for interfacing with the Pyxos FT Chip.

The EV Pilot example firmware's **main()** function performs the following tasks:

1. Initializes the ARM7 host microprocessor.
2. Reads the nonvolatile data from the ARM7 host microprocessor. This data helps the EV Pilot restore the Pyxos FT network to the state it had before the EV Pilot was powered off.
3. Activates the LEDs on the EV Pilot evaluation board in a pattern to give a visual indication that the EV Pilot is active.
4. Initializes the Pyxos FT Pilot API.
5. Calls a function to inform the Pyxos Network Example HMI application program that the EV Pilot firmware has reset.
6. Reallocates timeslots for any EV Points that the EV Pilot had previously assigned and stored in the ARM7 nonvolatile memory.
7. Initializes the LONWORKS Gateway.
8. Runs an infinite loop to repeatedly perform the following tasks:
    a. Call a function to process events from a connected computer, such as updates from the Pyxos Network Example HMI application program.
    b. Call a function to process registration push buttons for any EV Points in the Pyxos network, and send them to a connected computer and to a connected LONWORKS network.
    c. Call the Pyxos FT Pilot API to process Pyxos events, such as PNV updates from any of the EV Points in the Pyxos FT network.
    d. Call a function to process state changes for any of the dry-contact push buttons on the EV Pilot or on any of the EV Points in the Pyxos FT network.
    e. Call a function to check the current status of each EV Point in the Pyxos FT network, and set an alarm condition for any EV Points that do not have an acceptable status.
    f. Call the LONWORKS Gateway to process LONWORKS events, such as network variable updates from LONWORKS devices or PNV updates from another Pyxos FT network.
    g. Call a function to send an updated voltage-level value to the EV-Actuator Point for the Pyxos Network Example HMI application program's Performance Demo, if it is running.

h. Call a function to write current data to the ARM7 nonvolatile memory, if necessary.

i. Call a function to check the current status of the EV Pilot itself, in case the software or hardware state should have an unacceptable status.

The following sections describe other functions performed by the EV Pilot firmware.

Maintaining EV Point Data

The EV Pilot firmware maintains general information about the EV Points in two arrays, one for the EV Point's static properties and another for the EV Point's current status.

The **pointDefinitions** array describes each EV Point's static properties. Each array element has the following structure:

- A pointer to the EV Point's program interface. An EV Point's interface includes the Point's program ID, whether it is a hosted or unhosted Point, the number of PNVs and remote registers that are associated with the Point, and an array that describes the PNVs and remote registers.
- A variable that identifies which LED the EV Point uses to indicate that it is registered.
- A variable that identifies which switch the EV Point uses for registration.
- The name of the function that the EV Point uses to initialize its inputs. This function name can be NULL.
- The name of the function that the EV Point uses to clear cached Point data. This function name can be NULL.

The **pointInfo** array describes the current status of each EV Point. Each array element has the following structure:

- A variable that contains the timeslot that the EV Pilot assigns to the EV Point (or, if there were hardwired registrations, the timeslot that the EV Point returns to the Pilot).
- A variable that contains the EV Point's unique ID (UID). This UID is uniquely assigned to each Pyxos FT Chip during manufacturing.
- A variable that identifies whether the EV Point is correctly configured.
- A variable that contains the frame count of the last time that the EV Pilot checked the EV Point. When this frame count is approximately 250 milliseconds older than the current write frame, the EV Pilot will check up on the EV Point.
- A variable that contains the frame count of the last time that the EV Pilot received a value from the EV Point. If this frame count is approximately 500 milliseconds older than the current write frame, the EV Pilot assumes that the EV Point has a problem.
- A variable that identifies the current status of the EV Point's registration button.

EV Point Registration

There are three Point-registration methods: automatic, manual, and hardwired. For automatic registration, a Point sends its unique ID to the Pilot and the Pilot assigns a timeslot to the Point. For manual registration, a Point does not communicate with the Pilot until a user presses the Join button at the Point. For hardwired registration, a Point sends its timeslot to the Pilot, in addition to sending its unique ID, and the Pilot records that timeslot for that Point. The EV-Actuator and EV-Sensor Points use automatic registration, and the EV-Nano Point uses manual registration. The Pyxos Network Example does not use hardwired registration.

The following sections describe the registration process as implemented in the Pyxos Network Example firmware applications.

Initial Registration

The initial registration of a hosted EV Point includes the following general steps:

1. The Point's Pyxos FT Chip sends its unique ID (UID) to the Pilot.

   The EV-Actuator Point calls the **PyxosPointInit()** function to initialize the Pyxos FT Chip. It also sends its unique ID (UID) to the EV Pilot.

   The EV-Sensor Point calls an implementation of an SPI function to write to the Pyxos FT Chip's configuration register to set the Point's registration mode. It also sends its UID to the EV Pilot.

   The EV-Nano Point waits for external confirmation of registration (that is, the user's pressing the Join button), before it sends its UID to the EV Pilot.

   The EV-Sensor Point and the EV-Actuator Point both send their UIDs automatically when they start up or are reset. The EV-Nano Point sends its UID only during registration, that is, when you push the JOIN button.

2. The EV Pilot receives the UID.

   The Pyxos FT Pilot API calls the **PyxosPilotRegistrationRequestReceived()** function to assign a timeslot to the EV Point, if it has not already.

3. The EV Point is configured.

   After the EV Point is successfully configured, the Pyxos FT Pilot API calls the **PyxosPilotPointConfigured()** function.

4. The EV Pilot sets the EV Point's interface.

   The EV Pilot calls the **PyxosPilotSetPointInterface()** function based on the EV Point's program ID to inform the Pyxos FT API which interface the EV Point implements, and to allocate a cache of PNVs for the EV Point based on that interface.

5. The Pilot updates the Point's **SET_POINT_ONLINE** register.

The EV Pilot calls the **PyxosPilotSetPointOnline()** function to signify that the EV Pilot is ready to receive updates from the EV Point.

Installation after EV Pilot Reset

After a reset, the EV Pilot determines if there are any EV Points defined in its non-volatile memory. If there are, the EV Pilot reallocates the timeslot to the EV Points. After the allocation completes, the EV Pilot calls the **PyxosPilotSetPointInterface()** function to set each EV Point's interface. If an EV Point is not successfully configured, the EV Pilot attempts to reset the EV Point and reconfigure the EV Point.

Resetting a Point

After the EV Pilot resets an EV Point, the Pyxos FT Pilot API calls the **PyxosPilotResetPointCompleted()** function. The EV Pilot firmware always attempts to reconfigure the EV Point after the reset completes, regardless of whether the reset is successful.

Reconfiguration

After an EV Point is reconfigured or replaced, the Pyxos FT Pilot API calls either the **PyxosPilotPointConfigured()** or the **PyxosPilotPointConfigurationFailed()** function:

- If the reconfiguration is successful, the EV Pilot puts the EV Point online and updates all of the EV Point's inputs with the values that the EV Pilot had previously cached.
- If the reconfiguration is unsuccessful, the EV Pilot attempts to reset the EV Point, which, when it completes, restarts the reconfiguration process. This process continues until the EV Point is correctly configured or the EV Point is deleted.

Replacement

During the registration process, if a Point is discovered that has the same program ID as a currently installed Point, and that currently installed Point is not responsive, the EV Pilot replaces the old Point with the new Point and polls its outputs and updates its inputs.

Detection and Recovery of Non-Responsive Points

The EV Pilot polls the configuration register of each EV Point's Pyxos FT Chip. If this poll request fails, or if any update fails, or if the EV Pilot does not receive any updates from an EV Point for an extended period of time, the EV Pilot assumes that the Point has become unconfigured. The EV Pilot attempts to recover an unconfigured Point by resetting it and reconfiguring it.

The EV Pilot also periodically checks its own configuration by calling the **PyxosPilotCheckConfiguration()** function. If the EV Pilot is not correctly configured, it attempts to reconfigure itself by calling the **PyxosPilotReInitPyxosInterface()** function. If this attempt fails, the EV Pilot resets its host processor.

Processing for the EV-Actuator Point

The EV Pilot example firmware's processing for the EV-Actuator Point consists of three principal functions:

- Processing PNV updates from the EV-Actuator Point.

  During its main control loop, the EV Pilot firmware calls the Pyxos FT Pilot API to process Pyxos events, such as PNV updates. When the EV-Actuator Point updates a PNV (for example, when you press the dry-contact push button on the EV-Actuator Point evaluation board):

  1. The Pyxos FT Pilot API calls the **PyxosPilotPnvUpdateOccurred()** function of the EV Pilot firmware to process the PNV update. This function determines which of the Points sent the updated PNV, and calls the appropriate function, in this case, **ActuatorPnvUpdateOccurred()**.
  2. The **ActuatorPnvUpdateOccurred()** function sends the event to a connected computer and a connected LONWORKS network, then records the dry-contact push button event.
  3. Later in its main control loop, the EV Pilot firmware calls the **ProcessDryContacts()** function to activate the dry contact LED on the EV Pilot (**LED2**), on the EV-Actuator Point (**LED2**), and on the EV-Nano Point (**LED2**).

- Updating the EV-Actuator Point's analog output for the Pyxos Network Example HMI application program's Performance Demo.

  During its main control loop, the EV Pilot firmware calls the **SendPerfValue()** function to update the EV-Actuator Point's analog output that is used for the Performance Demo. This function checks whether the Performance Demo is active and whether it needs to be updated, and if so, it updates the EV-Actuator Point's AO PNV with the next appropriate voltage-level value for the demo.

  If the Performance Demo is not active, the EV Pilot updates the EV-Actuator Point's AO PNV when that PNV is updated by the Network Example HMI application program (whether connected by USB or by a LONWORKS network interface) or by a LONWORKS device that is connected to the EV Pilot.

  When the EV-Actuator Point receives this updated PNV, it sets the analog output to the updated voltage-level value by writing to the digital-to-analog converter (DAC) on the EV-Actuator Point.

- Processing digital I/O for the EV-Actuator Point.

  Because the Pyxos Network Example does not make use of the digital input or output that is available on the EV-Actuator Point, the EV Pilot firmware does not control the digital I/O for the EV-Actuator Point.

  However, the Pyxos Network Example HMI application program does allow you to set the digital output values for the EV-Actuator Point. The EV Pilot firmware simply passes any PNV updates for the digital outputs

from the Pyxos Network Example HMI application program to the EV-Actuator Point, and updates the DO_STATUS_LED PNV to illuminate the digital-output status LED (**LED6** on the EV-Actuator Point Evaluation Board) as necessary. This LED is on if any of the four digital outputs is active.

Processing for the EV-Sensor Point

During its main control loop, the EV Pilot example firmware calls the Pyxos FT Pilot API to process Pyxos events, such as PNV updates. When the EV-Sensor Point updates a PNV (for example, when the lux or temperature values change, or when the analog input value for the Pyxos Network Example HMI application program's Performance Demo changes):

1. The Pyxos FT Pilot API calls the **PyxosPilotPnvUpdateOccurred()** function of the EV Pilot firmware to process the PNV update. This function determines which of the Points sent the updated PNV, and calls the appropriate function, in this case, **SensorPnvUpdateOccurred()**.
2. The **SensorPnvUpdateOccurred()** function determines which of the EV-Sensor Point's PNVs was updated, LUX, TEMP, or AI:
   - For the LUX PNV: If the value has changed by 5 lux or more, the function saves the updated value and calls the **CheckLowLightLevel()** function to determine if the new lux value is below the light-level-alarm limit. If the value is below the limit, the EV Pilot example firmware updates a PNV to activate the EV-Actuator Point's Low Light LED (**LED4**) and calls a function to activate the EV Pilot's Low Light LED (**LED4**).
   - For the TEMP PNV: If the value has changed by 0.5 ºC or more, the function saves the updated value and calls the **CheckHighTempLevel()** function to determine if the new temperature value is over the temperature-alarm limit. If the value is over the limit, the EV Pilot example firmware updates a PNV to activate the EV-Actuator Point's Over Temperature LED (**LED3**) and calls a function to activate the EV Pilot's Over Temperature LED (**LED3**).
   - For the AI PNV: If the value has changed by 20 millivolts or more, the function saves the updated value.
3. For all PNV updates, if the value meets the PNV's change threshold, the EV Pilot example firmware sends the updated PNV to a connected computer and to a connected LONWORKS network.

Processing for the EV-Nano Point

During its main control loop, the EV Pilot example firmware calls the Pyxos FT Pilot API to process Pyxos events, such as PNV updates. When the EV-Nano Point updates a PNV (for example, when you press the dry-contact push button on the EV-Nano Point evaluation board):

1. The Pyxos FT Pilot API calls the **PyxosPilotPnvUpdateOccurred()** function of the EV Pilot firmware to process the PNV update. This function determines which of the Points sent the updated PNV, and calls the appropriate function, in this case, **NanoPnvUpdateOccurred()**.

2. The **NanoPnvUpdateOccurred()** function sends the updated PNV to a connected computer and a connected LONWORKS network and records the state of the dry-contact push button.
3. Later in its main control loop, the EV Pilot example firmware calls the **ProcessDryContacts()** function to activate the dry contact LED on the EV Pilot (**LED2**), on the EV-Actuator Point (**LED2**), and on the EV-Nano Point (**LED2**).

Connecting to a Computer Using a USB Connection

You can connect the EV Pilot to a Windows computer using a USB connection. Table 12 lists the files that provide the USB support for the Pyxos Network Example. The USB connection uses a standard USB A/B cable.

Table 12. Files for USB Support

File	Description
\USB Driver EchPyxosEVK.inf FTD2XX.dll FTIBUS.sys	The Windows USB driver provided by Future Technology Devices International (FTDI). This driver is automatically installed and registered with Windows during installation of the Pyxos FT EVK software.
\Pyxos Network Example\AT91SAM7S64\USB UART SCI uart.c uart.h	Universal asynchronous receiver-transmitter (UART) driver for the ARM7 host microprocessor. This driver interfaces with the FTDI USB UART chip used on the EV Pilot.
\Pyxos Network Example\Pilot PilotUsb.c PilotUsb.h	Control program that allows the EV Pilot example firmware to communicate with the computer using the USB UART driver.

Using the Pyxos-LONWORKS Gateway

The Pyxos-LONWORKS Gateway is part of the EV Pilot example firmware application, and it allows the EV Pilot to communicate with a LONWORKS network through the FT 3150 Smart Transceiver with ShortStack Micro Server on the EV Pilot evaluation board. The Gateway source code includes a Neuron C model file that maps the PNVs to LONWORKS functional blocks and standard network variable types (SNVTs). The Gateway also includes a serial driver that handles communication with the FT Smart Transceiver.

The EV Pilot is a fully functional LONWORKS device. You can use a network management tool, such as the LonMaker Integration tool, to configure and manage the EV Pilot and integrate it into a LONWORKS network. The Pyxos Network Example HMI application program is also a simple network management tool that manages only one device in a LONWORKS network, the EV-Pilot. To prevent management conflicts, do not manage the EV-Pilot with any

other network management tools, such as the LonMaker tool, when you are running the Pyxos Network Example HMI application program through a LONWORKS network interface.

The EV Pilot example firmware implements several LONWORKS functional blocks that allow any LONWORKS devices in the LONWORKS network to monitor and control the activity within the Pyxos FT network. Table 13 shows the LONWORKS functional blocks, configuration properties, and network variables that are available. In the table, the standard type for each functional block, network variable, and configuration property is given in parenthesis following the name used in the EV Pilot firmware.

Table 13. LONWORKS Functional Blocks Implemented by the EV Pilot

Functional Block	Network Variables and Configuration Properties	Description
NodeObject (SFPTnodeObject)		The standard node object required for all LONMARK devices with more than one functional block. For more information about this functional block, see *LONMARK® Functional Profile: Node Object SFPTnodeObject*, available from www.lonmark.org.
	nviRequest (SNVT_obj_request)	Input network variable used to request information or control functional blocks.
	nvoStatus (SNVT_obj_status)	Output network variable used to report status information concerning functional blocks.
	nvoAlarm2 (SNVT_alarm_2)	Output network variable used to report alarm information. Alarms are generated when Points are connected or disconnected, when the light level drops below a specified limit, or when the temperature exceeds a specified limit.
nanoDcSw (SFPTopenLoopSensor)	nvoNanoDcSw (SNVT_switch)	Output network variable that represents the current state of the dry-contact switch (**SW2**) on the EV-Nano Point.

Functional Block	Network Variables and Configuration Properties	Description
nanoDcLed (SFPTclosedLoopActuator)	nviNanoDcLed (SNVT_switch)	Input network variable used to control the dry-contact LED (**LED2**) on the EV-Nano Point.
	nvoNanoDcLedFb (SNVT_switch)	Output network variable that represents the current state of the dry-contact LED (**LED2**) on the EV-Nano Point.
NVIDX_nvoNanoSwitch (SFPTopenLoopSensor)	nvoNanoSwitch (SNVT_switch)	Output network variable that represents the current state of the user switch (**SW1**) on the EV-Nano Point.
nanoLed (SFPTclosedLoopActuator)	nviNanoLed (SNVT_switch)	Input network variable used to control the user LED (**LED1**) on the EV-Nano Point.
	nvoNanoLedFb (SNVT_switch)	Output network variable that represents the current state of the user LED (**LED1**) on the EV-Nano Point.
lightSensor (SFPTopenLoopSensor)	nvoLightValue (SNVT_lux)	Output network variable that represents the light value on the EV-Sensor Point.
	nciLowLightLimit (SCPTLowLimit1)	Configuration property that sets the low light limit. If **nvoLightValue** drops below the value of this CP, the low light LEDs on the EV Pilot and EV-Actuator Point are illuminated, the **under_range** status for this functional block is set, and an **under-range** alarm is generated.
tempSensor (SFPTopenLoopSensor)	nvoTempValue (SNVT_temp_p)	Output network variable that represents the temperature measured by the EV-Sensor Point.

Functional Block	Network Variables and Configuration Properties	Description
	nciHighTempLimit (SCPThighLimit1)	Configuration Property that sets the high temperature limit. If **nvoTempValue** exceeds the value of this CP, the high temperature LEDs on the EV Pilot and EV-Actuator Points are illuminated, the **over_range** status for this functional block is set, and an **over-range** alarm is generated.
openLoopAi (SFPTopenLoopSensor)	nvoAi (SNVT_volt_mil)	Output network variable that represents the voltage read by the EV-Sensor Point.
actuatorDcSw (SFPTopenLoopSensor)	nvoActDcSw (SNVT_switch)	Output network variable that represents the current state of the dry-contact switch (**SW1**) on the EV-Actuator Point.
actuatorDcLed (SFPTclosedLoopActuator)	nviActDcLed (SNVT_switch)	Input network variable used to control the dry-contact LED (**LED2**) on the EV-Actuator Point.
	nvoActDcLedFb (SNVT_switch)	Output network variable that represents the current state of the dry-contact LED (**LED2**) on the EV-Actuator Point.
actuatorSwitch_1 through actuatorSwitch_4 (SFPTopenLoopSensor)	nvoActuatorSw_1 through nvoActuatorSw_4 (SNVT_switch)	Array of output network variables that represent the current state of switches **SW2** through **SW5** on the EV-Actuator Point.
actuatorLed_1 through actuatorLed_2 (SFPTclosedLoopActuator)	nviActuatorLed_1 through nviActuatorLed_2 (SNVT_switch)	Input network variables used to control the user LEDs (**LED7** and **LED8**) on the EV-Actuator Point.
	nvoActuatorLedFb_1 through nvoActuatorLedFb_2 (SNVT_switch)	Output network variables that represent the current state of user LEDs (**LED7** and **LED8**) on the EV-Actuator Point.

Functional Block	Network Variables and Configuration Properties	Description
openLoopDo_1 through openLoopDo_4 (SFPTopenLoopActuator)	nviDoValue_1 through nviDoValue_4 (SNVT_switch)	Array of output network variables used to control the four digital outputs on the EV-Actuator Point header (**DOUT1** through **DOUT4**).
openLoopAo (SFPTopenLoopActuator)	nviAo (SNVT_volt_mil)	Input network variable used to set the analog output on the EV-Actuator Point.
PilotDcSw (SFPTopenLoopSensor)	nvoPilotDcSw (SNVT_switch)	Output network variable that represents the current state of the dry-contact switch (**SW1**) on the EV Pilot.
pilotDcLed (SFPTclosedLoopActuator)	nviPilotDcLed (SNVT_switch)	Input network variable used to control the dry-contact LED (**LED2**) on the EV Pilot.
	nvoPilotDcLedFb (SNVT_switch)	Output network variable that represents the current state of the dry-contact LED (**LED2**) on the EV Pilot.
pilotSwitch (SFPTopenLoopSensor)	nvoPilotSwitch (SNVT_switch)	Output network variable that represents the current state of the user switch (**SW2**) on the EV Pilot.

LONWORKS Interface

The Pyxos-LONWORKS Gateway is implemented in the **PilotLonWorksApp.c** file in the [*Pyxos FT EVK*]\Pyxos Network Example\Pilot\LonWorks directory. The Gateway maps Pyxos network variables (PNVs) to LONWORKS functional blocks, and defines LONWORKS network variable self-documentation data. It also includes several utility functions for handling the LONWORKS network data and alarm conditions.

When the EV Pilot sends updated PNVs to the network, the main EV Pilot application program (**PilotApp.c**) calls the **SendExternalEvent()** function, which calls the **SendToPc()** function to send data to the Windows computer. This function also calls the **lonProcessPyxosEvent()** function (in the Gateway, **PilotLonWorksApp.c**) if a LONWORKS network is defined.

When the computer or the LONWORKS network sends data to the EV Pilot, the main EV Pilot application program (**PilotApp.c**) receives the data from the **ProcessExternalPilotEvent()** function.

The Neuron C model file for the Pyxos interface was generated with the NodeBuilder Code Wizard. The generated files for the Pyxos FT EVK Examples

firmware application are in the [*Pyxos FT EVK*]\Pyxos Network Example\Pilot\LonWorks\Neuron Model directory.

Because a Pyxos FT network can often send small amounts of data more quickly than a LONWORKS network can, the Gateway must ensure that data is sent to the LONWORKS network at an appropriate rate, so the Gateway imposes a throttle on the propagation of network variables. As updates come in from the Pyxos FT network, the EV Pilot ensures that the network variables are updated, but it does not propagate them. Instead, the EV Pilot updates a data structure (**pendingNvPropagationTable**) to record the fact that the network variable needs to be propagated. The **LonPilotEventHandler()** function of the Gateway (**PilotLonWorksApp.c**) is responsible for propagating these updates onto the network in a round-robin fashion and to ensure that only one update at a time is sent to the ShortStack Micro Server.

ShortStack Interface and Serial Driver

The general architecture for communicating with a LONWORKS network includes:

- A host application (in this case, the EV Pilot example firmware)
- The ShortStack API
- A serial driver
- The ShortStack Micro Server firmware
- A transceiver that connects to the LONWORKS network

For the EV Pilot firmware, the serial driver uses the Serial Communication Interface (SCI) and is implemented for the ARM7 processor in the **ldvsci.c** file, which is in the [*Pyxos FT EVK*]\Pyxos Network Example\AT91SAM7S64\ShortStack SCI directory. This file defines the interface with the ShortStack Micro Server firmware in the FT 3150 Smart Transceiver.

To generate LONWORKS network-variable data for Pyxos FT data, use the ShortStack Developer's Kit. To modify the ShortStack interfaces for the Pyxos FT EVK examples, you must manually edit a few of the generated files. Table 14 lists some of the changes that you need to make. The generated files for the Pyxos FT EVK Examples firmware application are in the [*Pyxos FT EVK*]\Pyxos Network Example\Pilot\LonWorks\ShortStack directory.

Table 14. Changes to Files Generated by the ShortStack Wizard

File	Changes Needed
LonDev.c	In the **nvtable[]** network variable table, change the data type for the **sizeof()** function for the following variables: • For **&nciLowLightLimit**, change the type from **sizeof(SCPTlowLimit1)** to **sizeof(SNVT_lux)**. • For **&nciHighTempLimit**, change the type from **sizeof(SCPThighLimit1)** to **sizeof(SNVT_temp_p)**.

File	Changes Needed
LonDev.h	In the **NVIndex** enumeration, the NV array names contain a tilde (~), which does not compile. Change the tilde to another character, such as an underscore (_).
NvTypes.h	Enforce a packing of 1 and a padding of 0 when compiling the type definitions: • Add **#pragma pack(push, 1)** to the top of the file, after the **#include** statements. • Add **#pragma pack(pop)** to the end of the file, before the final **#endif** statement.

Important: The Pyxos FT EVK has been developed and tested with the ShortStack 2 Micro Server; changes might be required for different releases of the ShortStack firmware.

If you need to reload the ShortStack Micro Server, you can use the NodeLoad utility (available from the Echelon Web site, www.echelon.com) to load the system images from the [*Pyxos FT EVK*]\Pyxos FT EVK\Pyxos Network Example\ImagesArchive\Pilot ShortStack MicroServer directory.

Source Files and Project Files

In addition to the files in the [*Pyxos FT EVK*]\Pyxos Network Example directory that are listed in Table 9 on page 107, the Pilot Example uses the files in the [*Pyxos FT EVK*]\Pyxos Network Example\Pilot directory that are listed in Table 15.

Table 15. File Contents for EV Pilot Example

File	Contents
EvkPilot.ewd EvkPilot.ewp EvkPilot.eww	Project files for the IAR Embedded Workbench development environment.
MyPyxosApplication.h	C header file for the EV Pilot example firmware, generated by the Pyxos FT Interface Developer utility. This file also includes information about the EV Point interfaces that the EV Pilot supports.
MyPyxosApplication.PxOpts	Application options file for the EV Pilot example firmware, generated by the Pyxos FT Interface Developer utility.
PcPilotEvkComm.h	C header file used by the EV Pilot example application and the Pyxos Network Example HMI application program to define the interfaces to communicate with each other.

File	Contents
PilotActuatorPoint.c PilotActuatorPoint.h	C source and header files used by the EV Pilot to monitor and control the EV-Actuator Point.
PilotApp.c PilotApp.h	C source and header files for the main application for the EV Pilot example firmware, including main control loop, I/O processing, PNV handling, and networking.
PilotFlash.c PilotFlash.h	C source file used by the EV Pilot to manage non-volatile flash memory on the ARM7 host microprocessor, where the EV Pilot stores data pertaining to the Pyxos FT network.
PilotIo.c PilotIo.h	C source and header files used by the EV Pilot to access the LEDs and push buttons on the EV Pilot evaluation board.
PilotManagePoints.c PilotManagePoints.h	C source and header files used by the EV Pilot to install, replace, delete, and reconfigure the EV Points, including verifying communication between the EV Pilot and each of the EV Points.
PilotNanoPoint.c PilotNanoPoint.h	C source and header files used by the EV Pilot to monitor and control the EV-Nano Point.
PilotSensorPoint.c PilotSensorPoint.h	C source and header files used by the EV Pilot to monitor and control the EV-Sensor Point.
PilotUsb.c PilotUsb.h	C source and header files used by the EV Pilot to communicate with a PC using the USB interface.
Resources.h	C header file for the LONWORKS definitions that are used by the EV Pilot example firmware; generated by the Pyxos FT Interface Developer utility.

In addition to the files listed in Table 15, the Pilot requires the device-driver code that is contained in the following files:

- **uart.c** and **uart.h** in the [*Pyxos FT EVK*]\Pyxos Network Example\AT91SAM7S64\USB UART SCI directory. These files provide the USB drivers for the Pilot to communicate with a computer using the USB interface.
- **Flash.c** and **Flash.h** in the [*Pyxos FT EVK*]\Pyxos Network Example\AT91SAM7S64\SrcIAR directory. These files provide the flash

drivers for the EV Pilot to manage non-volatile flash memory on the ARM7 host microprocessor.

Both the EV Pilot and EV-Actuator Point use the ARM7 host processor, and share the following files for ARM7 driver support:

- **init.c** and **init.h** in the [*Pyxos FT EVK*]\Pyxos Network Example\AT91SAM7S64\Initialization directory. These files initialize low-level I/O with the ARM7 microprocessor, and call functions contained in other files within the [*Pyxos FT EVK*]\Pyxos Network Example\AT91SAM7S64 directory tree. You can customize these files by modifying macros that are defined at the project level.
- **psImpl.c** in the [*Pyxos FT EVK*]\Pyxos Network Example\AT91SAM7S64\Pyxos SPI directory and **psUtilImpl.c** in the [*Pyxos FT EVK*]\Pyxos API\Serial API directory. These files contain the ARM7 implementation of the Pyxos FT Serial API.
- **Cstartup.s79** and **Cstartup_SAM7.c** in the [*Pyxos FT EVK*]\Pyxos Network Example\AT91SAM7S64\SrcIAR directory. These files are the low-level initialization files that allow the ARM7 processor to run the C code for the EV-Pilot and EV-Actuator Point; they are adapted from samples provided with the IAR Embedded Workbench.

To provide the networking support for the Pyxos-LONWORKS Gateway, the EV Pilot uses many of the files contained in the [*Pyxos FT EVK*]\Pyxos Network Example\Pilot\LonWorks directory. The following files represent the more important files used by the EV Pilot:

- **PilotLonWorksApp.c** and **PilotLonWorksApp.h**. These files define the Pyxos-LONWORKS Gateway, and communicate with the other EV Pilot functions to send and receive PNVs from the EV Points and receive EV Pilot events. The Pyxos-LONWORKS Gateway uses the ShortStack API.
- **ldvsci.c** and **ldvsci.h** in the [*Pyxos FT EVK*]\Pyxos Network Example\AT91SAM7S64\ShortStack SCI directory. These files contain the ShortStack driver for the ARM7 microprocessor.
- **LonDev.c**, **LonDev.h**, and **NvTypes.h**. These files are generated by the ShortStack wizard.
- **lonapi.c** and **lonapi.h**. These files contain the LONWORKS API that is used to communicate with other LONWORKS devices.

The ShortStack interfaces use the **PilotShortStack.swprj** ShortStack Wizard project file in the [*Pyxos FT EVK*]\Pyxos Network Example\Pilot\LonWorks\ShortStack directory, and this file uses the Neuron C model files contained in the [*Pyxos FT EVK*]\Pyxos Network Example\Pilot\LonWorks\Neuron Model directory.

The [*Pyxos FT EVK*]\Pyxos Network Example\Pilot\LonWorks\ShortStack directory also contains the device interface file (**.XIF** file) that was used for developing the Pyxos Network Example firmware application.

The project files for the IAR Embedded Workbench development environment for the EV Pilot include two configurations to control conditional compilation for different environments: Flash Debug and Flash Release. These configurations allow you to control whether to include debug control code.

Data Flow Scenario for the Network Example

This section describes a scenario that shows the flow of program control and of data through the Pyxos Network Example firmware applications, including parts of the Pyxos FT API and some of the hardware interactions. The scenario follows a below-light-level event from the EV-Sensor Point to the EV Pilot and then to the EV-Actuator Point. The scenario begins with the light sensor on the EV-Sensor Point evaluation board:

1. The light sensor hardware processes incoming light levels and converts them to analog DC voltage levels, and passes them to pin 2 of the ATtiny13 chip.
2. To process the input data from the light sensor hardware, the EV-Sensor Point example firmware (**SensorPoint.c**) reads data from the analog-to-digital converter (ADC) on the ATtiny13 processor by calling the **readA2D()** function.

   This function enables the specified port B pin on the ATtiny13 chip for analog-to-digital conversion, reads the two ADC input registers from the port B pin, and performs the conversion. Four of the port B pins can be defined as ADC input channels.
3. The EV-Sensor Point example firmware passes the **ADMUX_LUX** constant to the **readA2D()** function to specify that it should read data from port B pin 2 (**PB3**).
4. The EV-Sensor Point example firmware then converts the value from the **readA2D(ADMUX_LUX)** function to a PNV named **lux**, and converts the byte order of the **lux** PNV to big-endian form, the form that is used by PNVs.

   In a big-endian byte order, the high-byte value is located in a lower-memory address, and the bits are numbered from left to right. Big-endian order contrasts with little-endian byte order, for which the high-byte value is located in a higher-memory address, and the bits are numbered from right to left.
5. The EV-Sensor Point example firmware checks the **POINT_SEND_BITS** register to determine if there are updates pending. If not, the EV-Sensor Point example firmware publishes the updated **lux** PNV.
6. The EV-Sensor Point example firmware writes the updated **lux** PNV to the Pyxos FT Chip. It writes the PNV value to its assigned Pyxos Chip Index (PCI), 0x0 in this case.
7. The Pyxos FT Chip on the EV-Sensor Point writes the PNV data to the Pyxos FT network during its assigned timeslot, so that the data is made available for the EV Pilot to read and process.
8. During its **main()** loop processing, the EV Pilot example firmware program (**PilotApp.c**) calls the **PyxosPilotEventHandler()** function of the Pyxos FT API (**PyxosPilot.c**) to ensure that network events are processed.
9. The Pyxos FT API (**PyxosPilotProcessInputs.c**) calls the **PyxosPilotPnvUpdateOccurred()** function of the EV Pilot example firmware program (**PilotApp.c**) to notify it that there is an updated PNV on the network for it to process.

10. The EV Pilot example firmware (**PilotApp.c**) calls the **SensorPnvUpdateOccurred()** function in **PilotSensorPoint.c** to process the updated PNV. This function in turn calls the **CheckLowLightLevel()** function.

11. The **CheckLowLightLevel()** function of the EV Pilot firmware program (**PilotSensorPoint.c**) checks whether the **lux** value that it received from the EV-Sensor Point is below the low-light threshold. If the **lux** value is below the low-light threshold, the firmware calls the **UpdateActuatorSnvtSwitch()** function to activate **LED4** on the EV-Actuator Point. It also calls the **PilotUpdateDigitalOutput()** function to activate **LED4** on the EV Pilot.

12. The **UpdateActuatorSnvtSwitch()** function of the EV Pilot example firmware (**PilotActuatorPoint.c**) converts the Boolean value of the **luxUnderLimit** variable to a PNV and passes it to the **UpdatePnv()** function in **PilotApp.c** to update the PNV and write it to the Pyxos FT Chip.

13. The **UpdatePnv()** function of the EV Pilot example firmware (**PilotApp.c**) calls the **PyxosPilotUpdatePnv()** function of the Pyxos FT API (**PyxosPilot.c**) to have the Pyxos FT Chip write the PNV value to the Pyxos FT network so that it gets to the EV-Actuator Point.

    The **UpdatePnv()** function also calls the **SendExternalEvent()** function to send the updated PNV to the computer (if the computer is connected), and to the Pyxos LONWORKS Gateway (if it is active).

14. During its **main()** loop processing, the EV-Actuator Point example firmware (**ActuatorPointApp.c**) calls the **PyxosPointEventHandler()** function of the Pyxos FT API (**PyxosPoint.c**) to ensure that network events are processed.

15. The Pyxos FT API (**PyxosPoint.c**) calls the **PyxosPointPnvUpdateOccurred()** function of the EV-Actuator Point example firmware (**ActuatorPointApp.c**) to notify it that there is an updated PNV on the network for it to process.

16. The EV-Actuator Point example firmware (**ActuatorPointApp.c**) calls the **ActuatorPointSetDigitalOutput()** function of **ActuatorPointIo.c** to activate **LED4** on the EV-Actuator Point.

17. The **ActuatorPointSetDigitalOutput()** function of the EV-Actuator Point example firmware (**ActuatorPointIo.c**) calls the **AT91F_PIO_SetOutput()** function of the ARM7 microprocessor library (**lib_AT91SAM7S64.h**) to set the output data register for line 27 of the Parallel Input/Output Controller A (**PA27**) in the ARM7 microprocessor. Setting this register to high (1) has the effect of activating **LED4** on the EV-Actuator Point.

Figure 35 shows the basic program and data flow for this scenario.

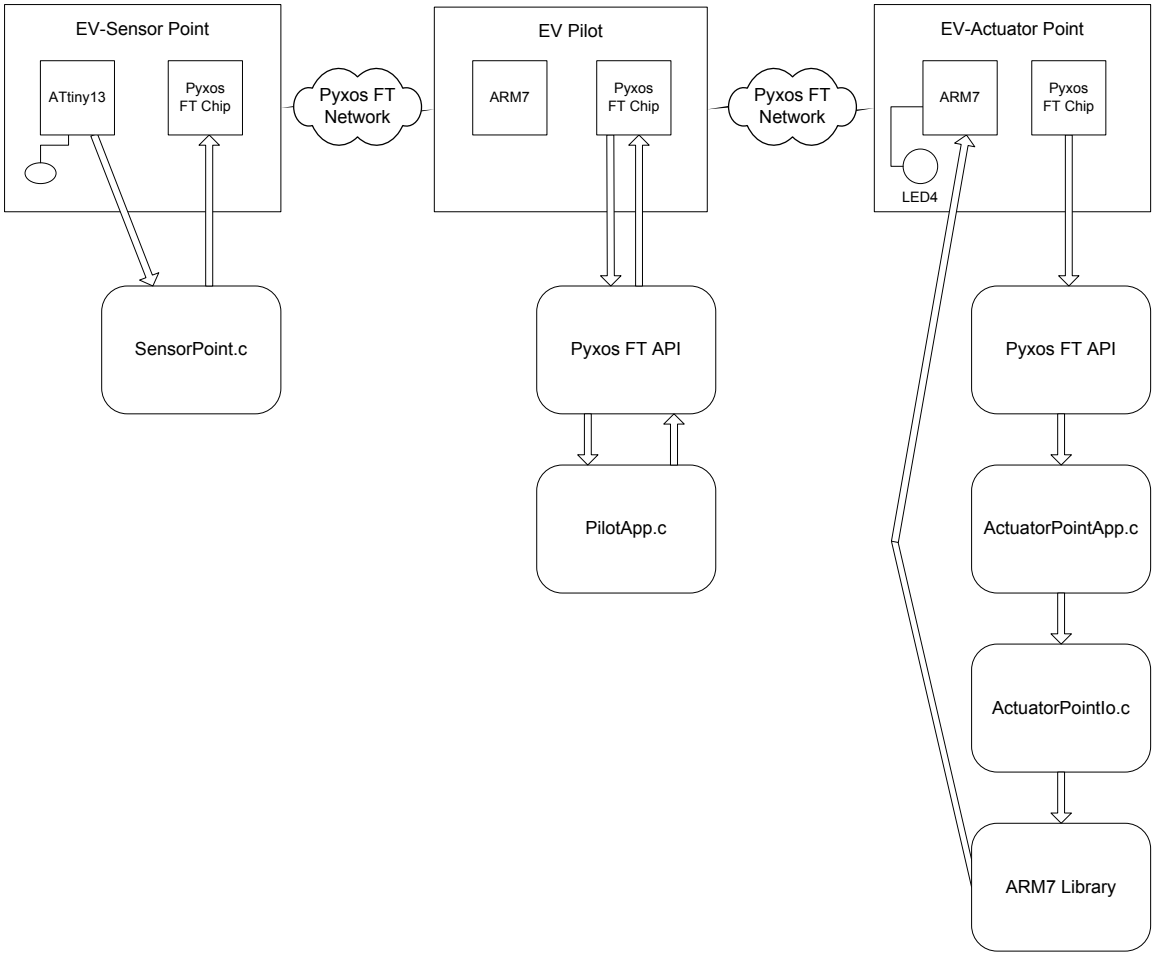


Figure 35. Program Flow Scenario for the Pyxos Network Example

9

The ARM7 PS API for the EV-Actuator Point and EV Pilot Examples

This chapter describes the Pyxos FT Serial API that is used by the example firmware applications in the EV-Actuator Point and the EV Pilot.

Overview of the API

The Pyxos FT Serial API (PS API) provides functions to read from and write to the Pyxos FT Chip synchronously, to detect interrupts, and to initialize the Pyxos FT Chip. The implementation of these functions is entirely dependent on the host microprocessor used.

The basic PS API is defined in the **psApi.h** file in the [*Pyxos FT EVK*]\Pyxos API\Include\Internal directory, where [*Pyxos FT EVK*] is the directory in which you installed the Pyxos FT EVK software (usually **C:\Program Files\Echelon\Pyxos FT EVK**). The PS API defines the following functions for interacting with the Pyxos FT Chip:

- **psInit()** – This function initializes the PS API driver.
- **psRead()** – This function reads data from the Pyxos FT Chip and stores it at a predefined memory location (indicated by the **PS_PBUFFER** pointer) in the ARM7 memory.
- **psWrite()** – This function writes data from a predefined memory location (indicated by the **PS_PBUFFER** pointer) in the ARM7 memory to the Pyxos FT Chip.
- **psIsInterruptSet()** – This function checks whether the interrupt line of the Pyxos FT Chip is currently set.

The implementation of the PS API for the ARM7 microprocessor that is used for both the EV Pilot and the EV-Actuator Point is contained in the **psImpl.c** file in the [*Pyxos FT EVK*]\Pyxos Network Example\AT91SAM7S64\Pyxos SPI directory.

The ARM7 implementation of the PS API uses the ARM7 Serial Peripheral Interface (SPI) functions. To enable the ARM7 microprocessor to interact with the Pyxos FT Chip, the ARM7 implementation of the PS API functions passes the address of the Pyxos FT protocol operation codes to the ARM7 SPI Receive Pointer Register (RPR) and Transmit Pointer Register (TPR). These registers are part of the Peripheral Direct Memory Access (DMA) Controller (PDC) of the ARM7 microprocessor. The PDC transfers data between on-chip serial peripherals and the on- and off-chip memories, in this case, the Pyxos FT Chip.

In addition to the implementation of the PS API functions, the example firmware also includes implementations of ARM7 low-level I/O functions to enable control and management of the ARM7 microprocessor. The firmware also includes an implementation of bootstrap code to load the C-based firmware into the ARM7 microprocessor to run.

For more information about the PS API, see chapter 6 of the *Pyxos FT Programmer's Guide*.

Files Used for the Examples

The Pyxos Network Example firmware uses the files listed in Table 16 on page 135 to define the ARM7 implementation of the PS API.

Table 16. Files for PS API Implementation

File	Contents
\Pyxos Network Example\AT91SAM7S64\Initialization init.c init.h	C source and header files that initialize low-level I/O for the ARM7 microprocessor.
\Pyxos Network Example\AT91SAM7S64\Pyxos SPI psImpl.c	C source file for implementation of the ARM7 PS API to interface with a Pyxos FT Chip.
\Pyxos Network Example\AT91SAM7S64\SrcIAR Cstartup.s79 Cstartup_SAM7.c	Assembler and C source files for starting the example firmware on the ARM7 microprocessor. These files are adapted from those provided by the IAR Embedded Workbench.

Index

F

H

I

J

L

M

N

O

P

R

S

T

ECHELON®

www.echelon.com

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Adesto Technologies:

11000R-10-11 11000R-10-14 11000R-10-13 11000R-10-12