ECHELON®

Introduction to the Pyxos FT Platform

078-0341-01A

Welcome

The Pyxos™ FT platform is Echelon's embedded control networking platform. The Pyxos FT platform provides a low-cost, high-performance solution for any smart device or controller that interfaces with remote sensors and actuators.

The smart device or controller and the remote sensors and actuators all use a Pyxos FT Chip to communicate over a reliable free-topology twisted-pair network. The Pyxos FT Chip acts as the network transceiver for the Pyxos FT network.

The 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 platform and to develop your own devices that incorporate Pyxos FT technology.

The Pyxos FT platform complements other control network applications, including those that use the LONWORKS® platform. The Pyxos FT platform and the LONWORKS platform use the same data types, which allows seamless communications from a Pyxos FT sensor or actuator to a remote LONWORKS device.

This document introduces the Pyxos FT technology and describes the Pyxos FT platform, including the Pyxos FT Chip and the Pyxos FT EVK.

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

Audience

This manual provides information for hardware engineers, firmware engineers, and managers who want to learn about Pyxos FT technology or evaluate the Pyxos FT platform.

Related Documentation

In addition to the *Introduction to the Pyxos FT Platform*, the following manuals also describe 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 application programming interface (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 manuals describe the Pyxos FT EVK Evaluation Kit:

- *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.
- *Pyxos FT EVK User's Guide*. This manual describes the Pyxos FT EVK hardware and how to use the hardware and software examples that are

included with the Pyxos FT EVK. It also describes how to develop devices that incorporate Pyxos FT technology.

When 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.

All of the Pyxos FT 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.

Table of Contents

1

The Pyxos FT Platform

This chapter introduces the Pyxos FT platform.

Introduction

The Pyxos FT platform is Echelon's embedded control networking platform. You can use the Pyxos FT platform to develop smart devices and controllers that communicate with remote sensors and actuators. You can use a Pyxos FT network as a low-cost, high-performance sensor and actuator I/O bus that extends the reach of any control system or control network to a wide variety of sensors and actuators.

You can also use the Pyxos FT platform to replace existing wiring harnesses, or more expensive solutions, with a simple twisted-pair cable. Embedded control networks based on the Pyxos FT platform can reduce costs for product installation, warranty, and total life-cycle.

The Pyxos FT Chip is a simple, small, low-cost component that enables the development of Pyxos FT embedded control-networking applications. The Pyxos FT Chip acts as the transceiver for managing communications on the Pyxos FT network, and it embeds the communications protocol for the Pyxos FT platform.

This chapter introduces both the Pyxos FT platform and technology.

I/O Bus

A Pyxos FT network is a low-cost, high-performance I/O bus, with a rich set of available data types. An I/O bus is a communications network that allows a controller and its associated I/O devices to communicate without requiring discrete wiring between them. That is, one set of wires connects all of the devices, rather than requiring separate wires from the controller to each of the I/O devices.

An I/O bus generally has a simple control strategy, for example, a single master controller with multiple subordinate sensors and actuators. In addition, the data that is collected by the sensors and actuators and delivered on the I/O bus is typically only directly available to the controller, that is, the sensors and actuators do not communicate directly with each other.

Unlike a peer-to-peer network, an I/O bus generally cannot be expanded to include a large number of devices because of the need for all communications to go through the controller. However, you can create large systems by enabling communications among multiple controllers, each with its own independent I/O bus. Controller-to-controller communications is typically provided by a control network, such as a network based on LONWORKS technology.

At the heart of an I/O bus is the network transceiver, which manages network communications.

Network Transceiver

System developers and installers need the flexibility to choose the I/O bus that suits their needs. That choice is based on balancing design constraints such as:

- Transaction speed
- Determinism
- Installation cost

- Component cost

No single communications technology can fully optimize all of these design constraints. However, an I/O bus can offer a fast transaction speed with determinism, with lower installation and component costs than a peer-to-peer network.

The Pyxos FT platform is based on a network transceiver, the Pyxos FT Chip, which manages the I/O bus network. Although the network type is an I/O bus, the actual network topology for a Pyxos FT network can be either:

- Bus topology, which connects devices in a linear configuration, with devices connected to the bus by a network stub.
- Free topology, which connects devices in any configuration.

The Pyxos FT Chip can be configured to act as the network transceiver for a master controller, known as the *Pilot*, or as the network transceiver for one of the subordinate sensors or actuators, known as a *Point*.

Pyxos Pilot

The master controller of a Pyxos network is the Pyxos Pilot. The Pilot controls all network activity and manages the I/O of the sensors and actuators on the network.

Although the Pyxos FT Chip manages network communications, the Pyxos Pilot requires a host microprocessor and a firmware application program to manage the data and activity of the Pyxos Points on the network. The Pilot also controls the addition and removal of Pyxos Points on the network. **Figure 1** shows the basic design of a Pyxos Pilot.

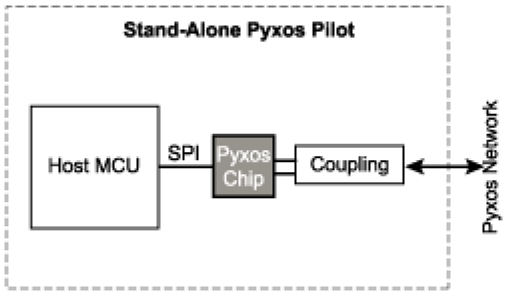

Figure 1. A Pyxos Pilot

A Pyxos Pilot manages all communications with devices on the Pyxos network, and with the addition of another network transceiver, such as an Echelon Smart Transceiver, a Pyxos Pilot can also manage communications with other networks, such as a LONWORKS network, as shown in **Figure 2**.

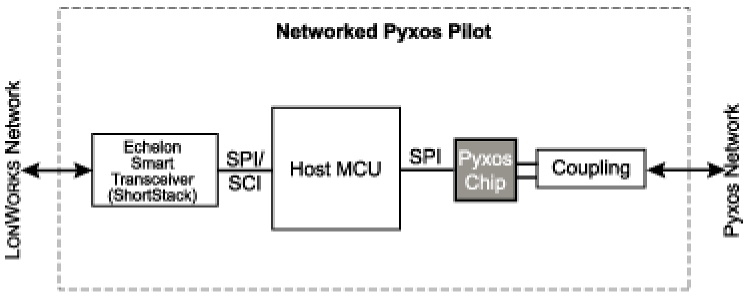

Figure 2. A Networked Pyxos Pilot

Echelon's Pyxos FT EVK provides an example Pilot that is based on a low cost Atmel® ARM7 microprocessor, the Atmel ARM® AT91SAM7S64. A Pilot's host microprocessor can be almost any microprocessor that the networking application requires, even a very low-cost 8- or 16-bit controller.

When you convert an existing application program to use the Pyxos FT platform, you can convert any or all of the existing local I/O to use Pyxos FT network-based I/O. You might also be able to free up code space in the Pilot by performing sensor or actuator linearization in the Points rather than in the Pilot.

Pyxos Point

Each sensor and actuator in a Pyxos FT network is a Pyxos Point. A Point manages its local I/O and communicates with the Pilot. A Point can be as sophisticated or as basic as the application requires.

The simplest type of Point is an *unhosted Point*, that is, a Point with the Pyxos FT Chip and no host microprocessor. An unhosted Pyxos FT Chip can control up to four bidirectional digital I/Os and an additional digital input. Thus, an unhosted Pyxos FT Point can be used for any sensor or actuator that requires a small number of digital I/O points. The Pyxos FT Chip in the unhosted Point sends the digital I/O values over the Pyxos FT network to the Pyxos Pilot. **Figure 3** shows the basic design of an unhosted Pyxos FT Point.

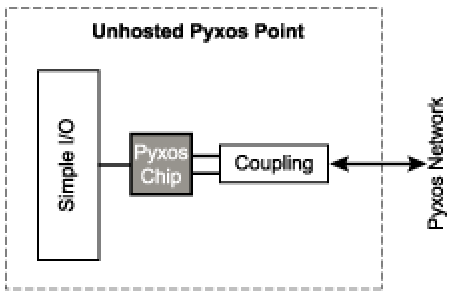

Figure 3. An Unhosted Pyxos Point

Echelon's Pyxos FT EVK provides an example unhosted Point, called the *EV-Nano Point*, that controls two digital inputs and two digital outputs.

For applications that require more complex sensor or actuators, a Point can be *hosted*, that is, a Point with the Pyxos FT Chip and a host microprocessor. The host microprocessor handles all of the digital or analog I/O for the sensor or actuator hardware, optionally processes the I/O values, and passes the raw or processed I/O values to the Pyxos FT Chip, which sends the values over the Pyxos

FT network to the Pyxos Pilot. **Figure 4** shows the basic design of a hosted Pyxos Point.

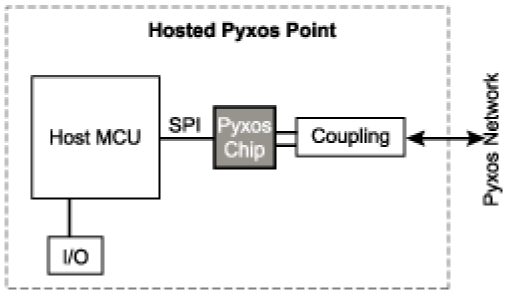

Figure 4. A Hosted Pyxos Point

A hosted Point allows the network application to use and control more complex types of I/O and possibly to perform filtering of input or output data.

Echelon's Pyxos FT EVK provides two example hosted Points, one that is an actuator based on an Atmel ARM7 microprocessor, the Atmel ARM AT91SAM7S64, and another that is a sensor based on an AVR® microprocessor, the Atmel ATtiny13. The actuator is called the *EV-Actuator Point*, and the sensor is called the *EV-Sensor Point*. Because the Pyxos FT Chip contains the complete networking protocol, a Point's host microprocessor needs to be only as powerful as the application requires.

In the figures, there is a box labeled "Coupling" that lies between the Pyxos FT Chip and the Pyxos FT network. This box represents a coupling circuit, which can be as simple as a direct connection to the network or as robust as a transformer-isolated connection. Which coupling circuit a device uses depends on its requirements for protection from common-mode interference.

Pyxos FT Protocol

The Pyxos Pilot and Pyxos Points on a Pyxos FT network communicate using a common communications protocol called the *Pyxos FT protocol*. A communications protocol is a set of standard rules for data representation, signaling, and error detection that enables a device to send information over a communications channel. The Pyxos FT protocol is built into the Pyxos FT Chip.

The Pyxos FT protocol uses time-division multiplexing (TDM) to manage the communications between the Pilot and the Points. TDM is a type of digital multiplexing in which two or more simultaneous bit streams are encoded as sub-channels into a single bit stream by interleaving bits from the different bit streams. The combined bit stream is decoded at the receiving end. For a Pyxos FT network, the different sub-channels are the bit streams from the different Points on the Pyxos FT network, each communicating with the Pilot. Each of the sub-channels has a fixed bit rate, which provides deterministic response for each of the sub-channels. The Pyxos FT Chips on the Pilot and the Points manage the interleaving of the channels.

The Pyxos FT protocol divides the time domain for the bit streams into several recurrent *timeslots* of fixed length. There is one timeslot for each sub-channel. A Pyxos *frame* consists of a window of time that contains all of the timeslots for all of the sub-channels. Each frame includes eight bytes of data sent from the Pilot to each Point, and another eight bytes of data sent from the each Point to the Pilot.

A Pyxos FT network can support up to 32 Points, and the number of timeslots is configurable, from 2 to 32. The Pilot can reserve timeslots for network expansion or it can assign as many timeslots as the current number of Points in the network. However, there must be as many timeslots as there are Points in the network.

For more information about the Pyxos FT protocol, see *Pyxos FT Protocol* on page 15 or the *Pyxos FT Programmer's Guide*.

Pyxos FT Network Topology

You can configure a Pyxos FT network into either of the following network topologies:

- Bus topology
- Free topology

A bus-topology network consists of a single backbone wire to which all of the devices connect, as shown in Figure 5.

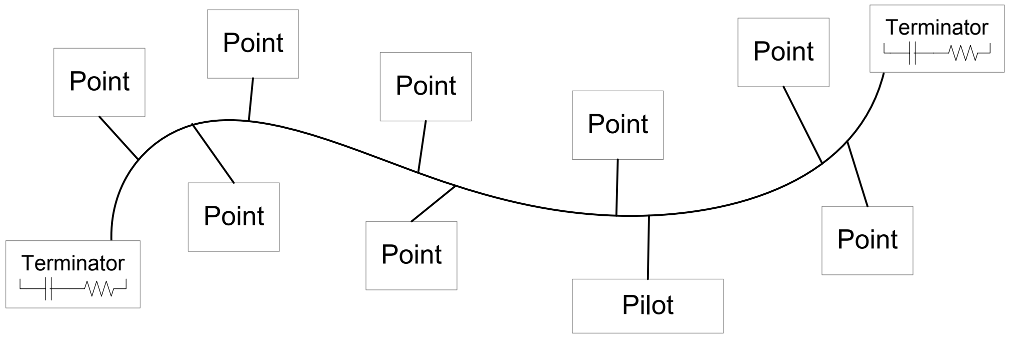

Figure 5. Bus-Topology Network

The following limits apply for a Pyxos FT network that uses bus topology:

- 400 meters maximum wire
- 0.3 meter stubs to each device
- Network terminators required at both ends of the bus
- Up to 32 Points

A free-topology network has no standard configuration; the configuration is dictated by the network application, as shown in Figure 6 on page 7.

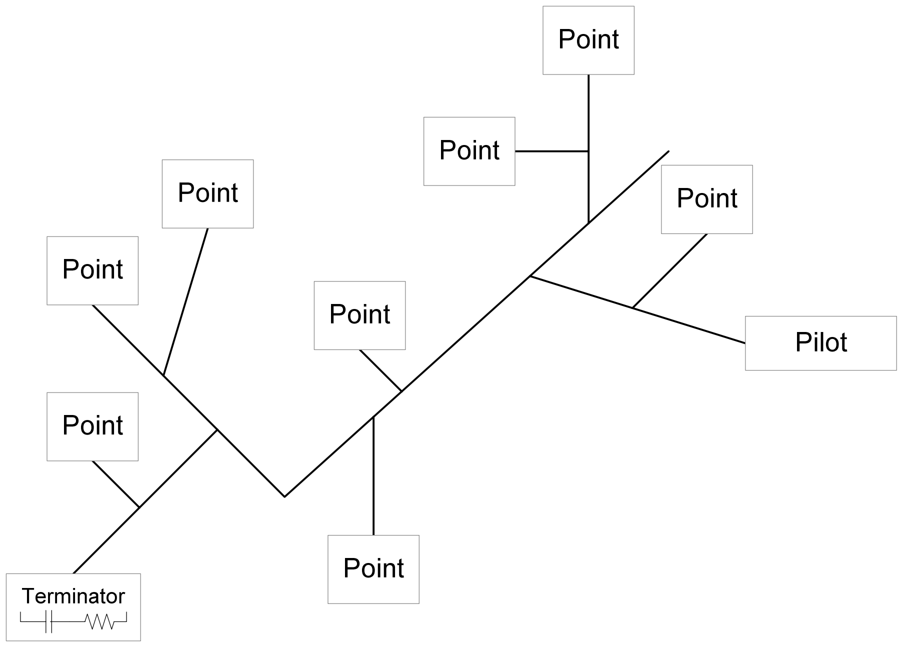

Figure 6. Free-Topology Network

The following limits apply for a Pyxos FT network that uses free topology:

- 100 meters maximum wire
- A single network required terminator anywhere in the network
- Up to 32 Points

For either of these network topologies, the physical network connections are polarity insensitive.

Adding Points to a Pyxos FT Network

During initialization for a Pyxos FT network, the Pilot determines how many timeslots are available for the network and assigns each Point to a timeslot. Each Point is added to the network during a process called *registration*.

At the start of the registration process, the Pilot advertises available timeslots on the network. Then, each Point uses one of the following registration methods to join the Pyxos FT network:

- Automatic
- Manual
- Hardwired

For *automatic* registration, each Point sends a message to the Pilot to request one of the free timeslots. The Pilot assigns a free timeslot to each Point, and sends a message to the Point to inform it of its assigned timeslot. No user interaction is required.

Automatic registration requires a host microprocessor for the Point. Therefore, unhosted Points cannot use automatic registration.

For *manual* registration, a user initiates registration, typically by pressing a button on the device. By pressing this button, called the Join button, the user causes the Point to send a message to the Pilot to request that the Point join the network. When the Pilot receives this message, it assigns a timeslot to the Point, and sends a message to the Point to inform it of its assigned timeslot.

Any Point can use manual registration, but unhosted Points must use manual registration.

For *hardwired* registration, the Point is assigned its timeslot during device manufacturing. The Point can store its timeslot information in its host microprocessor, in a wiring harness, or in another external device identifier. The Pilot must have reserved timeslots for all hardwired Points, and not advertise them as free timeslots for the network. A hardwired Point sends a message to the Pilot to inform it that the Point is using the timeslot.

Hardwired registration requires a host microprocessor for the Point. Therefore, unhosted Points cannot use automatic registration. No user interaction is required for hardwired registration.

All of the registration methods are based on the Pyxos FT protocol that is built into every Pyxos FT Chip. The Pilot firmware application controls the registration process.

Self Organization

The registration process allows Pyxos FT networks to be self organizing. That is, no user interaction is required during device installation to define the Points to the network. And because the Pyxos FT protocol is built into the Pyxos FT Chip, all three of the registration methods require only a limited amount of firmware programming for the Pilot application.

Self organization is possible because every Pyxos FT Chip has a 48-bit factory-defined unique ID, which ensures that every Pyxos FT device is unique. And, every hosted Pyxos Point has a 64-bit program ID that can be unique within the particular Pyxos FT network. Thus, when a Point requests a free timeslot from the Pilot, the Pilot can know exactly which Pyxos FT device is claiming the timeslot, and what kinds of data the Point uses.

For hardwired registration, system designers must take care that each Point's timeslot assignment is unique for the network.

Unhosted Points do not have a program ID that is unique within the network (all unhosted Points use a program ID of zero). However, the manual registration process allows the registration of multiple, otherwise identical, Points to have an order and some semantics associated with them. The user interaction defines the Point to the Pilot, similar to service pin messages used for LONWORKS devices.

System Initialization

During registration, each Point is assigned its own timeslot (from the Pilot, for automatic or manual registration, or from the Point, for hardwired registration). The timeslot acts as the Point's network address, so that the Pilot can always communicate with the Point.

In addition to the Point's unique ID, the Pilot needs to know the Point's interface so that the Pilot can know how to process the data values that the Point sends, and the Point can know how to process the data values that the Pilot sends. For example, if the Point sends a value to the Pilot, the Pilot must be able to determine whether that value represents a temperature, a pressure, a digital input, or some other type of data. The Pilot's firmware application needs to know the interfaces of all possible Points that could connect to its Pyxos FT network. When the Point sends its program ID during registration, the Pilot associates the program ID with the interface for that type of Point, and thus can properly process the Point's data.

The Pyxos FT EVK provides a utility program that allows Point developers to create unique program IDs for their Points and to create Point interface files that Pilots use to map the program ID to the interface definition. This utility program is called the Pyxos FT Interface Developer utility.

Link Power

There are two ways that the Pilot and Points within a Pyxos FT network can receive power:

- From a local power supply and power source attached to the Pilot or Point
- Over the Pyxos FT network cable from a power source, typically co-located with the Pilot

When a Pilot or Point receives power over the Pyxos FT network cable, it is receiving *link power*. In this case, the Pilot or Point receives both communications data and power over the same physical wires, which eliminates the need for separate wiring. A Pyxos FT network can include a mixture of devices that are locally powered and link powered.

For a link-power network, one device in the Pyxos FT network connects to the AC mains power and contains the network's source power supply. Which device contains the source power supply depends on your application: it could be the Pilot, one of the Points, or a separate device (such as a plug-in power adapter). Each of the other devices in the Pyxos FT network receives its power from the network.

Pyxos link power is not the same as Echelon's PL-20 power line technology or Echelon's TP/FT-10 link power technology. You cannot connect the Pyxos FT network directly to the AC power mains or directly to a LONWORKS TP/FT-10 channel (with or without link power). The Pyxos FT network is connected to each Pyxos FT device, and the link power is delivered over the network from the Pyxos FT power source.

Key Features of the Pyxos FT Platform

The Pyxos FT platform includes the following key features:

- Bus or free topology, using polarity-insensitive twisted pair wiring.
- Long network distance – up to 100 m for free topology, and up to 400 m for bus topology (with 0.3 m maximum stub length).
- High speed, deterministic communications – the Pilot can read and write 8 bytes of data for each of 32 Points at a rate of 40 times per second.
- Multiple power options – local power, AC link power, or DC link power.
- Low overhead for Pilot and Point host microprocessors – you can select a host based on application I/O and processing requirements, not based on the scan rate of the I/O.
- No host microprocessor is required for simple digital I/O.
- Self-organizing network does not require special tools or training for installers.
- Data types shared with LONWORKS networks provides seamless integration of Pyxos Points with LONWORKS devices, and enables multiple Pyxos FT networks to use a LONWORKS network as the backbone.
- Multiple low-cost isolation options available to meet a range of noise immunity requirements.
- Reliable communications – every message is protected by a CRC, and messages are automatically re-sent by the Pyxos FT Chip until they are received.
- Simple programming model – Pyxos FT Chip users see an interface similar to serial memory with an SPI interface. Pyxos data appears in this memory, where Points can store their data values.

Specification Summary

Table 1. Pyxos FT Specification Summary

Description	Specification
Communication bit rate	312.5 kbps
Maximum number of Points	32
Response time (2 Points)	1.8 ms
Response time (32 Points)	25 ms
Bus topology distance	400 m (with 0.3 m stubs)
Free topology distance	100 m
Link power	24 V AC or DC
Voltage for Pyxos FT Chip	3.3 V ±10%
Operating temperature	-40 ºC to +85 ºC
Polarity insensitive	Yes
Network wire types	Belden® 8471 or Category 5
Electrostatic discharge (ESD) protection for Pyxos FT Chip	8 kV contact 15 kV air
Package for Pyxos FT Chip	20-pin QFN
RoHS compliant	Yes

2

Pyxos FT Communications

This chapter describes the Pyxos FT communication model, the Pyxos FT protocol, and programming for Pyxos FT devices.

Introduction

The Pyxos FT platform provides a communication protocol, called the *Pyxos FT protocol*, that is built in to the Pyxos FT Chip. This protocol provides all of the communication semantics needed to ensure delivery and receipt of all network messages, including message acknowledgement and automatic retry for messages.

This chapter describes the communication model and introduces the Pyxos FT protocol. It also provides an overview of programming for Pyxos FT devices. Additional information is contained in the *Pyxos FT Programmer's Guide*.

Pyxos FT Communication Model

Communications within a Pyxos FT network are described from the perspective of a Point. Thus, input data is data that is an input for the Point (for example, the Pilot sends input data for the Point to illuminate an LED). And output data is data that is an output for the Point (for example, sensor data that the Point sends to the Pilot).

From a firmware point of view, the Point application views the network as shared local memory that it accesses through the Pyxos FT Chip's serial peripheral interface (SPI) port.

This shared local memory is organized as 32-bit words when data is sent on the Pyxos FT network. There are 128 of these 32-bit words (512 bytes) available for network data in a Pyxos FT Chip when it is configured as a Point.

The Point accesses the memory on the network by index value. This index value is called the *Pyxos Chip index* (PCI). A PCI value can point to either a data value (a *Pyxos Chip value*) or to a command register.

The Pyxos FT Chip contains reserved memory space for control registers that are mapped into the memory space. These control registers contain the chip's unique ID, its program ID, configuration information, error statistics, and error communication locations.

A Point sends a message to the network by writing data to one of the 32-bit words in the Pyxos FT Chip's memory. And a Point is notified when it receives new data from the Pilot through its control registers.

The Pilot's view of the network is different than the Point's view. The Pilot sees the network as a collection of up to 32 timeslots. Each timeslot is physically divided into a write timeslot and a read timeslot, but logically are together considered a single timeslot.

Figure 7 on page 15 shows the Pilot's and the Point's view of the Pyxos FT network. For the Pilot, the network is a set of read and write timeslots and control registers, and for the Point it is a set of Pyxos Chip values and control registers. In the figure, low memory addresses are at the bottom.

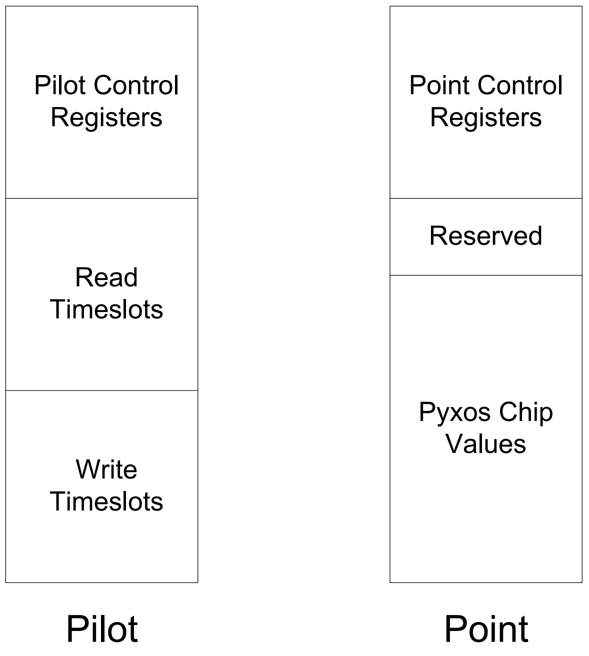

Figure 7. Pilot's View of the Network and Point's View of the Network

The Pilot addresses individual Points by timeslot number. A hosted Point sends its data to the Pilot automatically whenever its data values are updated. An unhosted Point can send its updates when the data values are updated or can wait for the Pilot to poll the Point's Pyxos FT Chip I/O registers.

The Pilot and Points can either run the Pyxos FT network in continuous mode or on-demand mode. Continuous mode means that data is continually flowing through the network. On-demand mode means that the Pilot or Points send or receive data from the network at predetermined intervals. On-demand mode is useful for battery-powered systems, such as fire or life safety systems.

Pyxos FT Protocol

The Pyxos FT protocol is a time-division multiplexing (TDM) link layer protocol, which has many advantages for a wired I/O bus:

- Deterministic – all devices are ensured regular and predictable access to the network
- Low Overhead – network packets contain only data, rather than a Media Access Control (MAC) address plus the data
- High speed – long message preambles are not required for synchronization because the network is always running
- Continuously supervised – there is no need for heartbeats because healthy devices transmit something in every timeslot

However, TDM differs from a peer-to-peer network:

- Architecture is strictly master / subordinate
- Maximum number of devices is limited to minimize MAC delays
- Message routing between channels is not allowed

- Packets are short, and packets have fixed length

The Pyxos FT protocol divides each repeated window of the TDM bit stream into a frame. Each frame is divided into a number of timeslots that represent segments of the TDM frame that are dedicated to a single Point.

The Pilot assigns a unique timeslot to each Point when the Point joins the network. Timeslot assignment is not necessarily sequential, but is randomized across the Points in the network. However, once a Point is assigned a timeslot, it uses that same timeslot until the Point leaves the network.

A Point sends data to the Pilot and receives data from the Pilot only during its assigned timeslot. The Pyxos FT Chip physically writes to a “write” timeslot and reads from a “read” timeslot, but a Pilot application sees both the write and read timeslots as a single timeslot assigned to the Point. Each timeslot contains 16 bytes of data: 8 bytes in the read timeslot and 8 bytes in the write timeslot.

Figure 8 shows a frame divided into four timeslots; the figure shows the physical write and read timeslots. Two of the timeslots are allocated to Points on the network, and the remaining timeslots are unused and are available for future Points that might join the network.

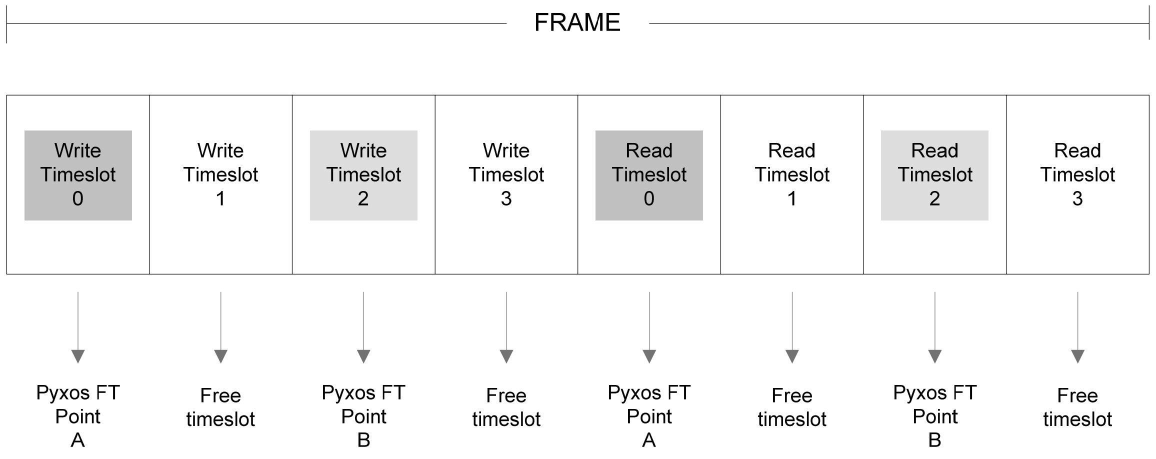

Figure 8. Timeslots in Each Network Frame

A Pilot initiates communications with the Points by including a “start of frame” (SOF) bit pattern at the beginning of each frame (not shown in the figure). This SOF pattern synchronizes communications with the Points.

The Pilot initiates writing immediately after the SOF by writing data to each timeslot. After it completes writing to the timeslots, the Pilot reads data from the timeslots.

The number of timeslots within each frame directly determines overall network latency and response time. For a network with two timeslots (two write plus two read timeslots), that is, a network with two Points, response time is less than 2 ms; for a network with a full 32 timeslots (32 write plus 32 read timeslots for 32 Points), response time is about 25 ms. It does not matter if a timeslot is allocated to a device or is a free timeslot – the network response time is the same.

A Pilot can run the network continuously or “on-demand”. For on-demand operation, the Pilot sends a single frame (one write and one read), and then stops the network. Whereas, in continuous operation, the Pyxos FT Chip automatically initiates another SOF immediately after the last read timeslot in the frame.

If the Pilot or a Point has no new data to report during a frame, it writes a special value in the timeslot to indicate that there is no new data. This value serves as a heartbeat for the device to indicate that the device is operating.

When a Point successfully reads data that is sent to it by the Pilot, the Point's Pyxos FT Chip sends the Pilot an indication that the Point is ready for the Pilot to send more data. The Pilot can cancel a write to a Point that does not seem to be receiving data correctly.

When the Pilot successfully reads data from a Point, the Pilot's Pyxos FT Chip sends the Point an indication that the Pilot is ready to receive more data from the Point.

Although it is generally not necessary, a Point can poll data from the Pilot.

Pyxos FT Messages

Although a single timeslot can hold only 16 bytes of data (8 bytes in the read timeslot and 8 bytes in the write timeslot), it is possible to send longer messages in a Pyxos FT network. The messages are sent in segments of 8 bytes, where each segment is sent in a separate network frame.

For example, a Point can send a 32-byte message to the Pilot by sending the data in four segments, or four frames. If the network includes 32 timeslots, the 32-byte value will be sent from the Point to the Pilot within 100 ms. Because the network latency is proportional to the number of timeslots, you can increase the transfer speed for longer messages by decreasing the number of timeslots in the network. Therefore, data point values longer than 8 bytes take proportionally longer to transmit that data point values of 8 bytes or less.

Pyxos Network Variables

A Pilot and its Points exchange data using a data abstraction called a *Pyxos network variable* (PNV). Each PNV has:

- Direction (input or output, from the Point's perspective)
- Size, in bytes
- Format

In addition to these attributes, the data encoding for each PNV is defined by a *network variable type*. The network variable type specifies the encoding, units, resolution, scaling, and structure for the PNV value. For example, the **SNVT_temp_p** type specifies a temperature value encoded as a signed big-endian two-byte Celsius value with 0.01 °C resolution. Standard formats are defined to display a **SNVT_temp_p** value as a Celsius or Fahrenheit value.

A rich set of more than 180 *standard network variable types* (SNVTs) is defined for the Pyxos FT platform. These types are shared with the LONWORKS platform, thus providing seamless interfaces from Pyxos sensors and actuators to LONWORKS devices. If a suitable SNVT is not available for your application, you can define your own network variable types, called *user network variable types*.

The PNVs for a Point are defined by the Point, and the collection of PNVs defines a Point's interface. The Pyxos FT EVK includes the Pyxos FT Interface Developer utility that you can use to define Point interfaces.

A Pilot's interface is comprised of all of the interfaces of all of the Points that the Pilot manages (those that are currently in the Pyxos FT network and any Points that might join the network in the future).

A Point sends output PNVs to the Pilot, and the Pilot updates input PNVs for a Point, as shown in **Figure 9**.

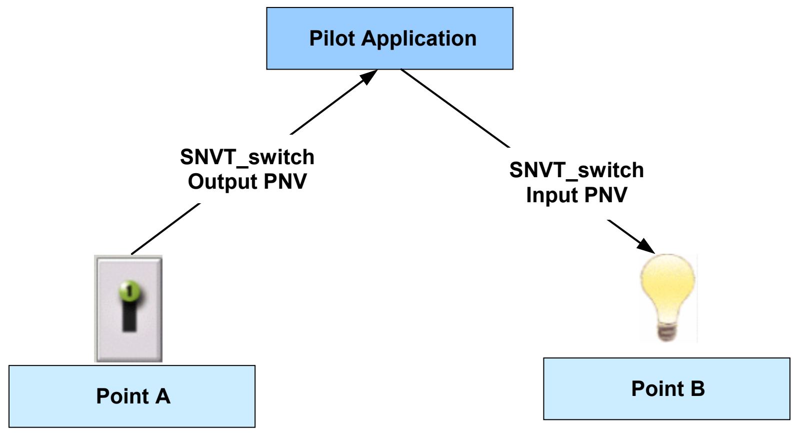

Figure 9. Input and Output PNVs

Pyxos FT Application Programming Interfaces

The Pyxos FT platform provides several application programming interfaces (APIs) to facilitate programming for Pyxos FT devices. These APIs include:

- Pyxos FT Pilot API: The API for Pilot applications
- Pyxos FT Point API: The API for Point applications
- Pyxos FT Serial Driver API: The API for a host microprocessor to communicate with the Pyxos FT Chip through its serial port.

The first two APIs are written in portable, ANSI C. The Serial Driver API is the non-portable part of the Pilot and Point APIs, and must be written specifically for each host microprocessor. The Pyxos FT EVK includes source code for the Pyxos FT APIs, which you can modify as necessary for your applications.

Network Determinism

Communications in a Pyxos FT network are deterministic, that is, each network TDM cycle has the same frame characteristics and has a predictable latency. However, a Pyxos FT application is only deterministic if its hardware and software are designed to be deterministic. In addition, the Point and Pilot applications must process PNV data deterministically.

The Pyxos FT EVK includes a performance demonstration that showcases deterministic network behavior: the Pilot sends a PNV value to a Point, which converts the value to a voltage level that another Point senses and sends back to the Pilot. You can observe the deterministic behavior using the Pyxos FT EVK Performance Demo.

Connecting to LONWORKS Networks

You can design a Pyxos FT Pilot so that it can connect to a LONWORKS network. The main two reasons why you might want to include LONWORKS connectivity in your Pyxos FT Pilot are:

- To connect multiple Pyxos FT networks together
- To communicate with LONWORKS devices

To implement a Pilot as a LONWORKS device, you can use an Echelon Smart Transceiver with Echelon's ShortStack® Micro Server and ShortStack API. For more complex Pilots, you can use a Smart Transceiver with Echelon's Microprocessor Interface Program (MIP) firmware and host API.

When you design the Pilot application, you need to decide which data points and conditions of the Pyxos FT network to expose to the LONWORKS network. Not all data points and conditions need to be exposed, but important data points and conditions and alarm conditions should be exposed. Data points can represent PNVs within the Pyxos FT network, or they can be values that are calculated by the Pilot based on PNV values.

You expose your selected data points and conditions as LONWORKS network variables. LONWORKS network variables that represent PNV values can use the same network variable types as the PNVs that they represent. LONWORKS network variables that represent calculated values or other conditions can use different network variable types. For example, a Pyxos Pilot could implement a Node Object functional block with a **SNVT_alarm_2** output variable to report alarm conditions that are detected by the Pilot application.

You install the LONWORKS interface of a Pyxos Pilot in a LONWORKS network like any other LONWORKS device. For example, you can use the LonMaker® Integration Tool to install the LONWORKS interface of a Pyxos Pilot. See the *Introduction to the LONWORKS Platform* for an overview of LONWORKS installation options.

3

The Pyxos FT Product Family

This chapter introduces the family of products that are available from Echelon for the Pyxos FT platform.

Introduction

The Pyxos FT product family includes the following products from Echelon:

- The Pyxos FT Chip
- The Pyxos FT EVK Evaluation Kit

The Pyxos FT Chip is available in quantities of 500 or 3000. The Pyxos FT EVK includes 10 sample chips that you can use for your initial prototyping and development.

The Pyxos FT EVK includes example hardware and software, and includes development hardware, software, and tools that you can use with the tools that you already use for your host microprocessor applications.

There are also a few hardware parts and software tools that Echelon either requires or recommends from other vendors.

Pyxos FT Chip

The Pyxos FT Chip is the network transceiver for a Pyxos FT Pilot or Point. It is a simple, cost-effective component that enables the development of Pyxos FT remote I/O systems for embedded control-network applications.

Because the Pyxos FT Chip embeds and fully implements the Pyxos FT protocol, Pyxos FT system designers can concentrate on the needs of the application rather than on networking.

The Pyxos FT Chip is small, 5 mm by 5 mm, and can fit into virtually any sensor or actuator device. The Pyxos FT Chip uses a 20-pin quad flat package no leads (QFN) package.

A Pyxos FT Chip can operate as either a Pilot transceiver or as a Point transceiver. To specify whether the chip should be a Pilot or a Point transceiver, the Pilot or Point application updates a value in the Pyxos FT Chip Configuration register, typically by calling a Pyxos FT API function.

A Pyxos FT Point can be hosted or unhosted. A simple circuit connected to one of the pins of the Pyxos FT Chip specifies whether the chip is hosted or unhosted. When the chip is hosted, its serial peripheral interface (SPI) port is available for communication with the host microprocessor. When the chip is unhosted, the SPI pins are used for digital I/O: four bidirectional I/O pins and one additional input-only I/O pin.

For more information about the specifications for the Pyxos FT Chip, see the data sheet for the Pyxos FT Chip or the *Pyxos FT Chip Data Book*.

Pyxos FT EVK Evaluation Kit

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 the 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.

Pyxos FT EVK Hardware

The Pyxos FT EVK includes the following evaluation boards that demonstrate the features and functions of the Pyxos FT Chip and of Pyxos FT network 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.

These four boards are preloaded with example firmware programs that simulate a room controller network, with a set of switches, a light sensor, a temperature sensor, and LEDs to provide user feedback.

The Pyxos FT EVK Evaluation Boards demonstrate key features and functions of Pyxos FT technology. At a system level, the Pyxos FT EVK demonstrates the following features:

- Link power at 24 VAC. The Pyxos FT network provides link 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 Point 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 demonstration 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 all of the necessary power supply and network cables so that you can run the example Pyxos FT network out of the box.

For more information about the hardware that is included with Pyxos FT EVK, see the *Pyxos FT EVK User's Guide*.

Pyxos FT EVK Software

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

The Pyxos FT EVK 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.
- 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.
- 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 Pyxos FT Interface Developer utility and the Pyxos FT APIs, see the *Pyxos FT Programmer's Guide*. For more information about the Pyxos Network Example HMI application program, the firmware source code, and the Neuron C model file that is used by the firmware, see the *Pyxos FT EVK User's Guide*. For more information about the ShortStack API, see the *ShortStack User's Guide*.

Third-Party Components

The Pyxos FT platform requires the use of certain third-party components, and recommends the use of other third-party components, for the design and development of Pyxos FT devices.

Required components include:

- Twisted-pair wire from Belden Wire & Cable. Belden® 8471 unshielded cable is preferred for link-powered Pyxos FT networks that use AWG 16 (1.3 mm) wire. For networks that use Category 5 wire, you can use any wire that meets the ANSI/TIA/EIA 568-B specification.
- A transformer from Transtek Magnetics. This transformer is required when the Pyxos FT Chip couples to the Pyxos FT network using transformer-isolated coupling.
- A one-stage line filter from Schaffner EMV. This filter is required for AC source power supplies.

Recommended components include:

- Embedded software development tools. You can use any development tools, but the Pyxos FT EVK examples were developed using tools from IAR Systems and from Atmel. The Pyxos Network Example firmware applications for the EV Pilot and EV-Actuator Point were developed using the IAR Embedded Workbench®, and the Pyxos Network Example firmware application for the EV-Sensor Point was developed using the Atmel® AVR® Studio.
- Application loading tools. For example, for loading system images into ARM7- and ARM9-family microprocessors, a programming tool such as the Atmel AT91 In-System Programmer (ISP), and for loading system

images into AVR microprocessors, a set of programming tools such as the Atmel AVR JTAGICE mkII and the Atmel AVR STK® 500 Flash Microcontroller Starter Kit.

- A hardware emulator and debugger. You can use any hardware emulator and debugger, but the Pyxos FT EVK ARM7 examples were tested with the Atmel AT91SAM-ICE JTAG Emulator and the IAR J-Link.

For more information about these required and recommended components, see the *Pyxos FT Chip Data Book* and the *Pyxos FT EVK User's Guide*.

A

Glossary

This appendix defines the commonly used terms for the Pyxos FT platform and the Pyxos FT family of products.

F

Frame

A packet of bits that contains all of the data for all sub-channels on a Pyxos FT network. Every frame includes one set of data values sent by the Pilot to each Point and one set of data values sent from each Point to the Pilot.

H

Hosted Pyxos Point

A Pyxos Point that includes a host microprocessor in addition to a Pyxos FT Chip. Contrasts with an unhosted Pyxos Point.

L

Link Power

Delivery of power over the Pyxos FT network cable; a link-powered Pilot or Point receives both communications data and power over the same physical wires.

LONWORKS Pyxos Pilot

A Pyxos Pilot that is attached to a LONWORKS network.

N

Networked Pyxos Pilot

A controller that is attached to an external network, such as a LONWORKS network or an IP network.

P

Pyxos

See Pyxos FT platform.

Pyxos Chip Index

The address of a four-byte Pyxos Chip value.

Pyxos Chip Value

A four-byte value or register in a Pyxos FT Chip. Each Pyxos Chip value is indexed by a Pyxos Chip index, and each Pyxos network variable and Pyxos FT Chip register is comprised of one or more Pyxos Chip values.

Pyxos FT API

A set of portable C language modules that provide access to a Pyxos FT network for a Pyxos Pilot or a Pyxos Point; includes the Pyxos FT Pilot API, the Pyxos FT Point API, and the Pyxos FT Serial API.

Pyxos FT Chip

An integrated circuit that can be used as a transceiver to attach a Pyxos Pilot or Pyxos Point to a Pyxos FT network. The Pyxos FT Chip can be used with or without a host microprocessor. The Pyxos FT Chip can optionally connect directly to digital I/O hardware when not used with a host microprocessor.

Pyxos FT Network

An embedded control network consisting of a Pyxos Pilot, one or more Pyxos Points, and the communications channel connecting them.

Pyxos FT Pilot API

A set of portable C language modules that provide access to a Pyxos FT network for a Pyxos Pilot; part of the Pyxos FT API.

Pyxos FT Platform

A low-cost, high-performance embedded control networking technology from Echelon.

Pyxos FT Point API

A set of portable C language modules that provide access to a Pyxos FT network for a Pyxos Point; part of the Pyxos FT API.

Pyxos FT Serial API

A set of portable C language modules that encapsulate platform-dependent code for transferring data between a host microprocessor and a Pyxos FT Chip, and for optional interrupt polling of the host microprocessor.

Pyxos Pilot

The controller for a Pyxos FT network. The Pyxos Pilot is the device that monitors and controls the Pyxos Points on a Pyxos FT network. A Pilot can optionally include a LONWORKS interface or an interface to another network, such as an IP network. A Pyxos Pilot can be implemented on virtually any host microprocessor that meets your application's requirements.

Pyxos Point

A subordinate device attached to a Pyxos FT network that is not a Pyxos Pilot; typically a sensor or actuator.

Pyxos Network Variable

A 1- to 31-byte data value maintained by a Pyxos Pilot or Pyxos Point that is transmitted on a Pyxos FT network. The units, resolution, scaling, and format of each network variable are described by a network variable type in the configuration for a Pyxos Point.

R

Read Timeslot

A timeslot that carries a data packet from a Pyxos Point to a Pyxos Pilot.

Register

An area of memory within the Pyxos FT Chip that contains a data value that is used to configure, control, or report the state of the Pyxos FT Chip.

Registration

The method that the Pyxos Pilot uses to assign a timeslot to a Pyxos Point and acquire its interface, allowing the Point to join the network that is hosted by the Pilot.

S

Standalone Pyxos FT Network

A Pyxos FT network that is attached to a standalone Pyxos Pilot.

Standalone Pyxos Pilot

A Pyxos Pilot that is not attached to an external network, such as a LONWORKS network.

Sub-Channel

A dedicated bit stream between a Pyxos Pilot and Pyxos Point.

T

Timeslot

A part of a Pyxos FT network frame that is allocated to a single Pyxos Point; can be a read timeslot or a write timeslot.

U

Unhosted Pyxos Point

A Pyxos Point that does not include a host microprocessor.

W

Write Timeslot

A timeslot that carries a data packet from a Pyxos Pilot to a Pyxos Point.

ECHELON®

www.echelon.com