I/O Flex 6126
Integration Guide

OEMCtrl®

Table of Contents

What is this document about?

This document provides instructions to integrate the I/O Flex 6126 into the Building Management System (BMS), which is speaking one of the protocols listed below.

Assumption: The controller has been configured by the factory and is functioning correctly. The factory should supply the site integrator with an object listing, which enables the integrator to gather information from the controller.

Protocol Overview

Protocols are the communication languages spoken by control devices. The main purpose of a protocol is to communicate information in the most efficient method possible. Different protocols exist to provide different kinds of information for different applications.

In the BMS application, many different protocols are used, depending on the manufacturer. More and more owners are demanding that their entire facilities be seamlessly linked together and presented in one easy-to-use front end.

All of our controllers have the ability to speak multiple protocols. So, no matter what company your customer chooses for the controls in the rest of their building, our controller will communicate with them without the added cost of a gateway.

What is a protocol?

- A set of formal rules describing how to transmit data, especially across a network
- A low level protocol defines:
  - the electrical and physical standards to be observed
  - bit-and-byte-ordering
  - the transmission, error detection, and correction of the bit stream
- A high level protocol deals with data formatting, including:
  - syntax of messages
  - terminal-to-computer dialogue
  - character sets
  - sequencing of messages, etc.
- It is a language spoken between electronic devices

  NOTE An example is the protocol IP, which stands for Internet Protocol

For two devices to communicate with each other, they must speak the same protocol or have a protocol translator.

Why are there so many protocols?

- Because any two pieces of building automation equipment can vary with application, so can the protocol
- Due to varying applications, protocols are designed with efficiency in mind

What are the benefits of having three of the most widely used protocols built in?

- Manufacturers can now provide a controller with their units that can be seamlessly integrated into a BAS.
- For the building owner, it means upgrade and expansion costs will be competitive.
- Expensive gateways are eliminated.
- Field selection of the protocol requires less up-front coordination, thus reducing manufacturing costs.
- Flexibility and simple configuration allow the customer to make future additions and changes without additional costs.
- More opportunities for sales

What does the site integrator need?

The building owner must supply the following information to the site integrator:

- Protocol Implementation Conformance Statement (See the Appendix)
- Unit-specific object listing (for LonWorks, an XIF file may be required.) See *Appendix* (page 46) for obtaining the file.

The site integrator then supplies the building owner with the:

- Device address
- Network baud rate
- Network Numbers (BACnet only)

The specific site settings are then applied to the controller using this information. Each protocol setting has a unique configuration. See *Configuring protocols* (page 4).

Configuring protocols

BACnet

BACnet, which stands for Building Automation and Controls network, is a protocol developed by ASHRAE. BACnet was developed as a response to industry concerns about increased networking of BAS components using proprietary communications methods. In the past, these proprietary communications severely limited the building owners' choices for system expansion, upgrade, and replacement. Every major controls vendor in North America, as well as academics, end users, consulting engineers, and government groups participated in its development.

BACnet has been accepted as an open standard by the American National Standards Institute (ANSI) and the European CEN standards. It is also being adopted as an international ISO standard.

BACnet is designed to include all building systems, lighting, security, fire, heating, ventilation, and air conditioning. Its purpose is to promote interoperability - sharing data between systems made by different vendors.

It provides the necessary tools to develop a specification for systems that are interoperable. BACnet provides methods and standards for representing information, for requesting and interpreting information, and for transporting information.

BACnet over ARC156

ARCnet is an embedded networking technology well suited for real-time control applications in both the industrial and commercial marketplaces. Its robust performance and the availability of low-cost silicon make it the network of choice in BAS's.

ARC156 is a unique implementation of ARCnet. ARC156 is similar to master slave/token passing (MS/TP). The main difference between the two is speed. ARC156 baud rate is 156K baud whereas MS/TP tops out at 76.8K baud.

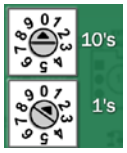
Also, ARC156 uses a separate communications co-processor to handle the network traffic and a separate processor to handle the program execution. This provides faster processing of applications and handling of communications on the network. ARC156 is the standard communications method used by our controllers.

Configuring the I/O Flex 6126 for ARC156

1 Turn **off** the I/O Flex 6126's power.

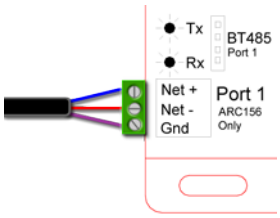
2 Using the rotary switches, set the controller's address. Set the **Tens** (**10's**) switch to the tens digit of the address, and set the **Ones** (**1's**) switch to the ones digit.

EXAMPLE If the controller's address is 01, point the arrow on the **Tens** (**10's**) switch to 0 and the arrow on the **Ones** (**1's**) switch to 1.



3 Connect the communications wiring to Port 1 in the screw terminals labeled **Net +**, **Net -**, and **Gnd**.

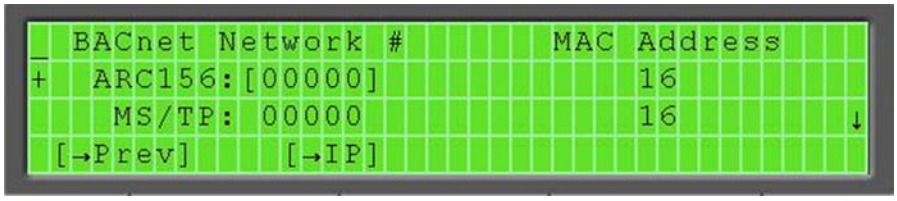
NOTE Use the same polarity throughout the network segment.



4 If the I/O Flex 6126 is at either end of a network segment, connect a BT485 to the I/O Flex 6126.

5 Turn **on** the I/O Flex 6126's power.

6 Set the correct network number to the unique BACnet ARC156 network at the site.



BACnet MS/TP

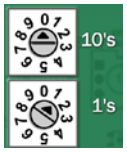
BACnet Master Slave/Token Passing or MS/TP is used for communicating BACnet over a sub-network of BACnet-only controllers. Each controller on the network has the ability to hear the broadcast of any other device on the network. The speed of an MS/TP network ranges from 9600 baud to 76.8K baud.

Configuring the I/O Flex 6126 for BACnet MS/TP

1 Turn **off** the I/O Flex 6126's power.

2 Using the rotary switches, set a unique address. Set the **Tens** (**10's**) switch to the tens digit of the address, and set the **Ones** (**1's**) switch to the ones digit.

EXAMPLE If the controller's address is 01, point the arrow on the **Tens** (**10's**) switch to 0 and the arrow on the **Ones** (**1's**) switch to 1.

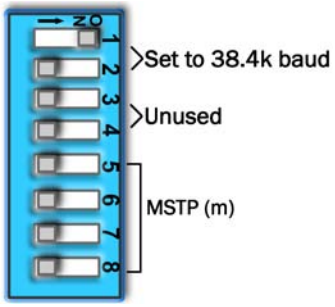


- Set the Comm Selector DIP Switches **1** and **2** for the appropriate communications speed (9600, 19.2k, 38.4k, or 76.8k bps).

  NOTE Use the same baud rate for all devices on the network segment.

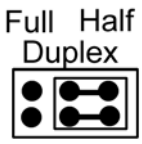
Baud Rate	DIP Switch 1	DIP Switch 2
9600	Off	Off
19.2	Off	On
38.4	On	Off
76.8	On	On

3 Set the Comm Selector DIP Switches **5** through **8** for BACnet MS/TP (m) master or (s) slave. The following example shows the DIP Switches set for 38.4k baud and BACnet MS/TP(m).



NOTE MS/TP (m) is recommended.

4 Set the Duplex for Half (two-wire).



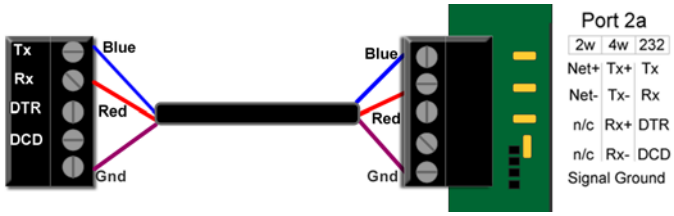
5 Set the Communications Selection jumper to EIA-485.

6 Configure Port 2a for BACnet MS/TP. Connect to **Net+**, **Net-**, and **Gnd**.

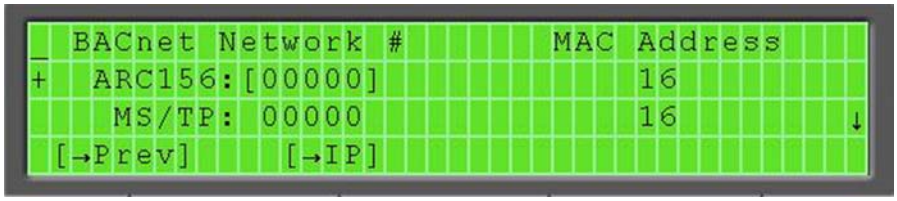
Wire specifications

- A dedicated 24 AWG to 18 AWG twisted pair wire (EIA 485)
- 2000 feet (610 meters) for 76.8 kbps, or
- 3000 feet (914.4 meters) for 9600 bps, 19.2 kbps, or 38.4 kbps, before needing a Repeater.
- Devices should be daisy chained and not star wired.
- If the I/O Flex 6126 is at either end of a network segment, connect a BT485 to the I/O Flex 6126

NOTE Use the same polarity throughout the network segment.



7 Turn **on** the I/O Flex 6126's power.



8 Set the correct network number to the unique BACnet MS/TP network at the site.

BACnet PTP

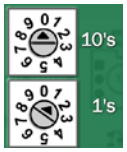
PTP is used to connect two distinct BACnet networks so that information can be shared between the networks. PTP uses an EIA-232 connection between two BACnet half-routers. This connection allows for two different BACnet networks to speak to each other, even at different baud rates.

Configuring the I/O Flex 6126 for BACnet PTP

1 Turn **off** the I/O Flex 6126's power.

2 Using the rotary switches, set a unique address. Set the **Tens** (**10's**) switch to the tens digit of the address, and set the **Ones** (**1's**) switch to the ones digit.

EXAMPLE If the controller's address is 01, point the arrow on the **Tens** (**10's**) switch to 0 and the arrow on the **Ones** (**1's**) switch to 1.

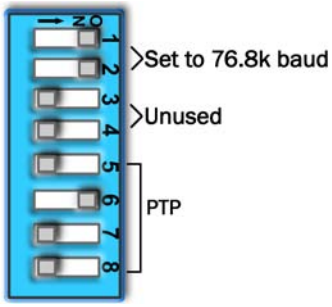


- Set the Comm Selector DIP Switches **1** and **2** for the appropriate communications speed (9600, 19.2k, 38.4k, or 76.8k bps).

  NOTE Use the same baud rate for all devices on the network segment.

Baud Rate	DIP Switch 1	DIP Switch 2
9600	Off	Off
19.2	Off	On
38.4	On	Off
76.8	On	On

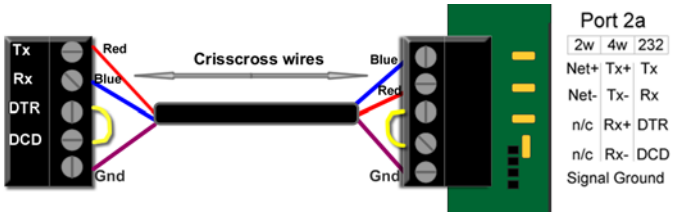
3 Set the Comm Selector DIP switches 5 through 8 for BACnet PTP. The following example shows the DIP switches set for 76.8k baud and BACnet PTP.



4 Set the Communications Selection jumper to **EIA-232**. EIA-232 indicates the controller will be connected to one other device speaking this protocol.

5 To wire I/O Flex 6126 to another device:

- Configure Port 2a for BACnet PTP.
- Connect to **Net+**, **Net-**, and **Gnd**.
- Connect to Tx, Rx, DTR, DCD, and Gnd using three wire termination with pins 3 and 4 jumpered. Wiring must go plus-to-minus and minus-to-plus, **Gnd**-to-**Gnd**.



6 See table below to wire I/O Flex 6126 to a modem.

Modem (25-pin) Null Modem Cable	Null Modem Cable (9-pin)	S2-DB9 (9-pin)	Device (5 pin)
TX - pin 2	TX - pin 3	TX - pin 3	TX - pin 1
RX - pin 3	RX - pin 2	RX - pin 2	RX - pin 2
DTR - pin 20	DTR - pin 4	DTR - pin 4	DTR - pin 3
DCD - pin 8	DCD - pin 1	DCD - pin 1	DCD - pin 4
GND - pin 7	GND - pin 5	GND - pin 5	GND - pin 5

7 Turn **on** the I/O Flex 6126's power.

Modbus

The Modbus protocol is used mostly in the industrial process market to communicate between PLCs (Programmable Logic Controllers). Although there is no official standard, there is extensive documentation on Modbus and most companies who choose to interface using this protocol follow the same format.

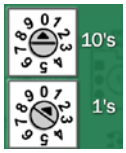
Modbus is not a protocol that is particularly well suited for building management because of its limited master/slave structure, but as it is fairly simple to construct an interface, many companies do offer Modbus as an open protocol solution.

Configuring the I/O Flex 6126 for Modbus RTU or ASCII - 6126

1 Turn **off** the I/O Flex 6126's power.

2 Using the rotary switches, set a unique address. Set the **Tens** (**10's**) switch to the tens digit of the address, and set the **Ones** (**1's**) switch to the ones digit.

EXAMPLE If the controller's address is 01, point the arrow on the **Tens** (**10's**) switch to 0 and the arrow on the **Ones** (**1's**) switch to 1.

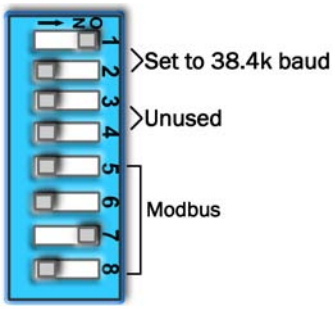


- Set the Comm Selector DIP Switches **1** and **2** for the appropriate communications speed (9600, 19.2k, 38.4k, or 76.8k bps).

  NOTE Use the same baud rate for all devices on the network segment.

Baud Rate	DIP Switch 1	DIP Switch 2
9600	Off	Off
19.2	Off	On
38.4	On	Off
76.8	On	On

3 Set the Comm Selector DIP Switches **5** through **8** for Modbus. The following example shows the DIP Switches set for 38.4k baud, and Modbus.



4 Set the Communications Selection jumper to EIA-232. or EIA 485.

a) If EIA 485, the controller will be daisy-chained to the network of devices.

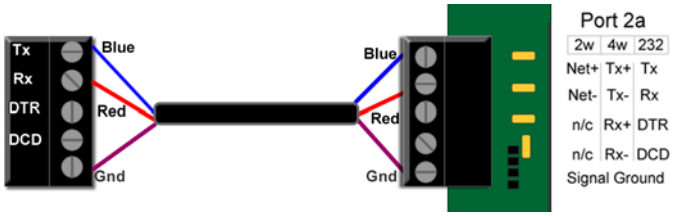
b) If EIA-232, the controller will be connected to one device.

5 Configure Port 2a for Modbus using EIA 485. Connect to Net+, Net-, and Gnd.

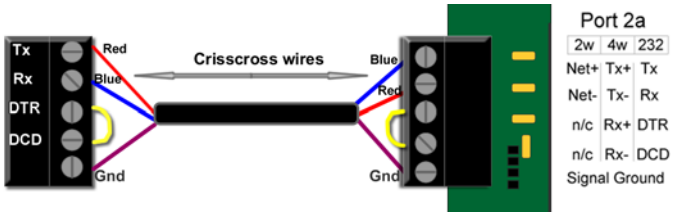
Wire specifications

- A dedicated 22 AWG to 18 AWG twisted pair wire (EIA 485)
- 3000 feet (914.4 meters) for 9600 bps, 19.2 kbps, or 38.4 kbps.
- Devices should be daisy chained and not star wired.

NOTE Use the same polarity throughout the network segment.



6 Configure Port 2a for Modbus using EIA 232. Connect to **Tx**, Rx, DTR, DCD, and Gnd using three wire termination with pins 3 and 4 jumpered. Wiring must go plus-to-minus-to-plus, Gnd to Gnd.and **Gnd**. Connect to Tx, Rx, DTR, DCD, and Gnd using three wire termination with pins 3 and 4 jumpered. Wiring must go plus-to-minus and minus-to-plus, Gnd-to-Gnd.



7 Do not power the device from the same transformer that powers the I/O Flex 6126.

8 Turn **on** the I/O Flex 6126's power.

Johnson Controls (N2)

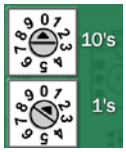
N2 is not a standard protocol, but one that was created by Johnson Controls, Inc. that has been made open and available to the public. Johnson Controls is the only company to use N2 bus as their standard network protocol. Because it is open and still prevalent within the industry, N2 is a standard offering for our controllers.

Configuring the I/O Flex 6126 for N2

1 Turn **off** the I/O Flex 6126's power.

2 Using the rotary switches, set a unique address. Set the **Tens** (**10's**) switch to the tens digit of the address, and set the **Ones** (**1's**) switch to the ones digit.

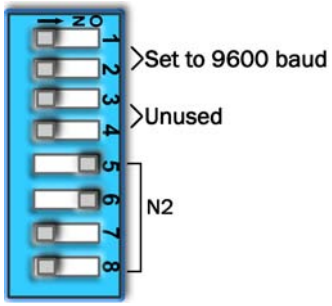
EXAMPLE If the controller's address is 01, point the arrow on the **Tens** (**10's**) switch to 0 and the arrow on the **Ones** (**1's**) switch to 1.



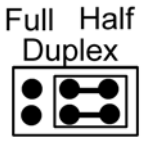
- Set the Comm Selector DIP Switches **1** and **2** for 9600 bps.

NOTE Use the same baud rate for all devices on the network segment.

3 Set the Comm Selector DIP switches **5** through **8** for Johnson Controls N2. The following example shows the DIP switches set for 9600 baud and N2.



4 Set the Duplex for Half (two-wire).



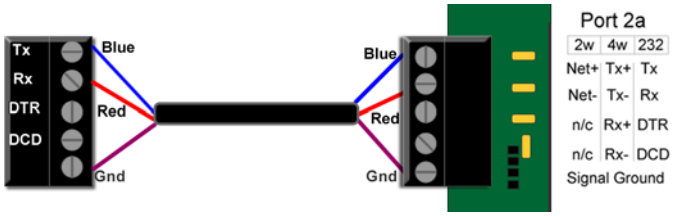
5 Set the Communications Selection jumper to EIA-485.

6 Configure Port 2a for N2. Connect to Net+, Net-, and Gnd.

Wire specifications

- A dedicated 22 AWG to 18 AWG twisted pair wire (EIA 485)
- 3000 feet (914.4 meters) for 9600 bps, 19.2 kbps, or 38.4 kbps.
- Devices should be daisy chained and not star wired.

NOTE Use the same polarity throughout the network segment.



7 Turn **on** the I/O Flex 6126's power.

LonWorks

LonWorks is an open protocol that was originally developed by Echelon Corporation. It is now maintained by Echelon in collaboration with members of the LonMark Interoperability Association. It requires the use of Echelon's Neuron microprocessor to encode and decode the LonWorks packets.

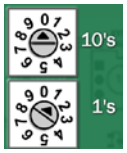
The LonWorks protocol is based on the concept of using standardized functional profiles to control similar pieces of equipment. OEM controllers are LonWorks compatible devices, but are not LonMark devices. A LonMark device has been thoroughly tested by Echelon (LonMark.org) and has been given the LonMark logo indicating compliance with the LonWorks profile specification. All LonMark devices require the use of proprietary hardware manufactured by Echelon Corp. In order to reduce the cost of adding that hardware on every controller, OEM formats the data packets in a manner specified by the LonWorks documentation and hands them off to the LonWorks Option Card.

Refer to the *Appendix* (page 48) for the LonWorks Protocol Implementation Conformance Statement (PICS).

Configuring the I/O Flex 6126 for the LonWorks Option Card

1. Turn **off** the I/O Flex 6126's power.
2. Using the rotary switches, set a unique address. Set the **Tens** (**10's**) switch to the tens digit of the address, and set the **Ones** (**1's**) switch to the ones digit.

   EXAMPLE If the controller's address is 01, point the arrow on the **Tens** (**10's**) switch to 0 and the arrow on the **Ones** (**1's**) switch to 1.

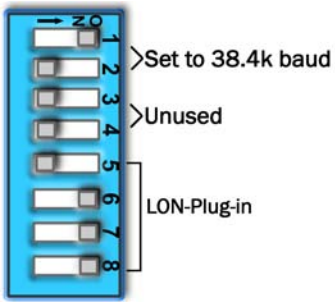
   

   - Set the Comm Selector DIP Switches **1** and **2** for the appropriate communications speed (9600, 19.2k, 38.4k, or 76.8k bps).

     NOTE Use the same baud rate for all devices on the network segment.

Baud Rate	DIP Switch 1	DIP Switch 2
9600	Off	Off
19.2	Off	On
38.4	On	Off
76.8	On	On

3 Set the Comm Selector DIP switches 5 through 8 for LonWorks Option Card. The following example shows the DIP switches set for 38.4k baud and LonWorks Option Card.

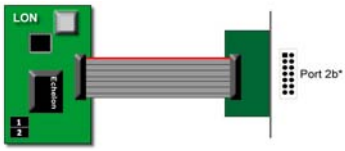


4 Set the Communications Selection jumper to EIA-485.

5 Connect **Port 2b** to the LonWorks Option Card with the supplied ribbon cable.

6 Plug the LonWorks Option Card's ribbon cable (9.75 in.) into the <ALCProduct>'s Option Card port.

CAUTION! the controller must be **OFF** before being connected.



7 Turn **on** the I/O Flex 6126's power.

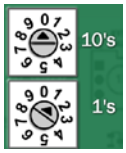
Configuring the I/O Flex 6126 for LonTalk via SLTA

The I/O Flex 6126 can be configured as an application node in a LonWorks system. An Echelon Serial-To-LonTalk Adapter (SLTA-10) provides the network interface between each I/O Flex 6126 and the LonWorks network. An additional SLTA-10 is required if connecting a host computer running LonWorks network management software to the LonWorks network.

1 Turn **off** the I/O Flex 6126's power.

2 Using the rotary switches, set a unique address. Set the **Tens** (**10's**) switch to the tens digit of the address, and set the **Ones** (**1's**) switch to the ones digit.

EXAMPLE If the controller's address is 01, point the arrow on the **Tens** (**10's**) switch to 0 and the arrow on the **Ones** (**1's**) switch to 1.

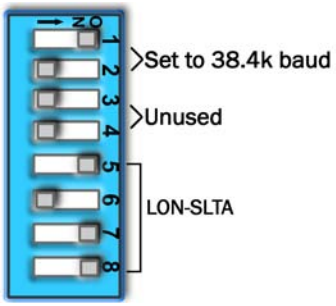


- Set the Comm Selector DIP Switches **1** and **2** for the appropriate communications speed (9600, 19.2k, 38.4k, or 76.8k bps).

NOTE Use the same baud rate for all devices on the network segment.

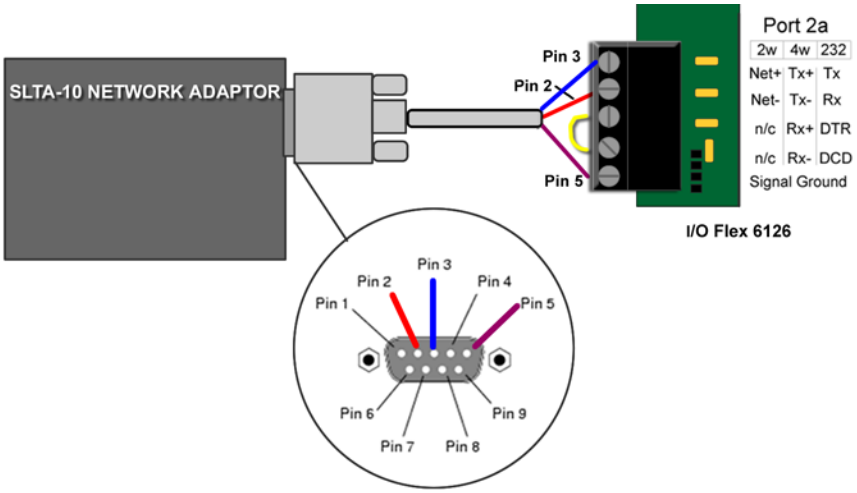
Baud Rate	DIP Switch 1	DIP Switch 2
9600	Off	Off
19.2	Off	On
38.4	On	Off
76.8	On	On

3 Set the Comm Selector DIP switches SW5 through SW8 for LON. The following example shows the DIP switches set for 38.4k baud and LON- SLTA.



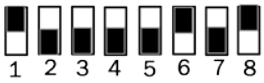
4 Set the Communications Selection jumper to **EIA-232**. EIA-232 indicates the controller will be connected to one other device speaking this protocol.

5 Configure Port 2a for LON using EIA 232. Connect to **Tx, Rx, DTR**, **DCD**, and **Gnd** using three wire termination with pins 3 and 4 jumpered. Wiring must go plus-to-minus and minus-to-plus, Gnd_to_Gnd.



6 Do not power the SLTA-10 from the same transformer that powers the I/O Flex 6126.

7 Set the following SLTA-10 dipswitch settings.



8 Turn **on** the I/O Flex 6126's power.

Troubleshooting

Most common communication problems

Wiring termination

- If wiring an EIA-485 connection, the wire is terminated plus (+) to plus (+) and minus (-) to minus (-). If the receive LED is solid, this means you have the connection incorrectly terminated.
- If the wiring is an EIA-232 connection, the wire must connect plus (+) to minus (-) and minus (-) to plus (+). The GND must be connected to GND.

Jumper selection

- Make sure the jumper for the communication port is set to the communication networks wiring type EIA-485 or EIA-232.

Dipswitch selection

- Make sure the correct protocol is chosen - Comm Selector DIP switches **5, 6, 7, and 8**
- Make sure the correct baud rate is chosen - Comm Selector DIP switches **1 and 2.**

NOTE: These settings are defined at controller start-up. Power must be cycled to make a settings change.

Addressing -

- The rotary address switches define the controller's individuality on the network. Each device must have a unique address.

BACnet MSTP

Verify BMS and controller settings:

1 Both set to speak BACnet MS/TP.

- Comm Selector DIP Switches 3, 4, 5, and 6
- By getting a Modstat of the controller through the BACview. Click and hold the **FN** key and then click the **.** key at the same time. Scroll to the bottom of the page to the section entitled **Network Communications** to view the active protocol and baud rate.

2 Both set for the same baud rate.

- Comm Selector DIP switches **1 and 2.**
- By getting a Modstat of the controller through the BACview. Click and hold the **FN** key and then click the **.** key at the same time. Scroll to the bottom of the page to the section entitled **Network Communications** to view the active protocol and baud rate.

3 BMS configured to speak 2-wire EIA-485 to the controller.

4 Both set to 8 data bits, No Parity, and 1 stop bit.

5 Rotary address switches set for the controller's unique slave address.

6 Proper connection wiring.

7 BMS reading or writing to the proper BACnet objects in the controller - download the latest points list for the controller to verify.

8 BMS is sending requests to the proper MS/TP MAC address of the controller.

9 Present the BMS company with a copy of your controller's BACnet PICS so that they know which BACnet commands are supported.

NOTE See Appendix.

It may be necessary to adjust the following two MS/TP Protocol timing settings through the BACview6.

Max Masters - defines the highest MS/TP Master MAC address on the MS/TP network.

For example, if there are 3 master nodes on an MS/TP network, and their MAC addresses are 1, 8, and 16, then Max Masters would be set to 16 (since this is the highest MS/TP MAC address on the network).

This property optimizes MS/TP network communications by preventing token passes and "poll for master" requests to non—existent Master nodes.

In the above example, MAC address 16 would know to pass the token back to MAC address 1 instead of counting up to MAC address 127). Each MS/TP master node on the network must have their Max Masters set to this same value. The default is 127.

Max Info Frames - defines the maximum number of responses that will be sent when the I/O Flex 6126 receives the token.

Any positive integer is a valid number. The default is 10 and should be ideal for the majority of applications. In cases where the I/O Flex 6126 is the target of many requests, this number could be increased as high as 100 or 200.

NOTES

- MS/TP networks can be comprised of both Master and Slave nodes. Valid MAC addresses for Master nodes are 0 – 127 and valid addresses for Slave nodes are 0 - 254.

  If the third party attempts to communicate to the controller but does not get a response, make sure the controller is set as a BACnet MS/TP (m) master. The BACnet software asks the controllers, "Who Is?" This is to auto-locate devices on the network. Only controllers set as masters will answer this request.

BACnet PTP

Verify BMS and controller settings:

1 Both set to speak N2.

- Comm Selector DIP Switches 3, 4, 5, and 6
- By getting a Modstat of the controller through the BACview. Click and hold the **FN** key and then click the **.** key at the same time. Scroll to the bottom of the page to the section entitled **Network Communications** to view the active protocol and baud rate.

2 Both set for the same baud rate.
   - Comm Selector DIP switches **1 and 2.**
   - By getting a Modstat of the controller through the BACview. Click and hold the **FN** key and then click the **.** key at the same time. Scroll to the bottom of the page to the section entitled **Network Communications** to view the active protocol and baud rate.

3 BMS set to speak 2-wire EIA-232 to the controller.

4 Both set to 8 data bits, No Parity, and 1 stop bit.

5 Rotary address switches set for the controller's unique slave address.

6 Proper connection wiring.

7 BMS reading or writing to the proper BACnet objects in the controller - download the latest points list for the controller to verify.

8 BMS is sending requests to the proper BACnet device instance of the controller.

9 Present the BMS company with a copy of your controller's BACnet PICS so that they know which BACnet commands are supported.

NOTE See Appendix (B) for the N2 Protocol Conformance Statement

Modbus

Verify BMS and controller settings:

1 Both set to speak Modbus RTU.
   - Comm Selector DIP Switches 3, 4, 5, and 6
   - By getting a Modstat of the controller through the BACview. Click and hold the **FN** key and then click the **.** key at the same time. Scroll to the bottom of the page to the section entitled **Network Communications** to view the active protocol and baud rate.

2 Both set for the same baud rate.
   - Comm Selector DIP switches **1 and 2.**
   - By getting a Modstat of the controller through the BACview. Click and hold the **FN** key and then click the **.** key at the same time. Scroll to the bottom of the page to the section entitled **Network Communications** to view the active protocol and baud rate.

3 BMS configured to speak 2-wire EIA-485 to the controller.

4 Both set to 8 data bits, No Parity, and 1 stop bit.

5 Rotary address switches set for the controller's unique slave address.

6 Proper connection wiring.

7 BMS must be reading or writing to the proper Modbus register numbers on the controller. Download the latest points list for the controller to verify.

8 BAS is sending requests to the proper slave address of the controller.

NOTE See Appendix (B) for Modbus Protocol Conformance Statement

Modbus Exception Codes that might be returned from this controller

Codes	Name	Description
01	Illegal Function	The Modbus function code used in the query is not supported by the controller.
02	Illegal Data Address	The register address used in the query is not supported by the controller.
04	Slave Device Failure	The Modbus Master has attempted to write to a non-existent register or a read-only register in the controller.

N2

Verify BMS and controller settings:

1 Both set to speak N2.
   - Comm Selector DIP Switches 3, 4, 5, and 6
   - By getting a Modstat of the controller through the BACview. Click and hold the **FN** key and then click the **.** key at the same time. Scroll to the bottom of the page to the section entitled **Network Communications** to view the active protocol and baud rate.

2 Both set for 9600 baud rate.
   - Comm Selector DIP switches **1 and 2.**
   - By getting a Modstat of the controller through the BACview. Click and hold the **FN** key and then click the **.** key at the same time. Scroll to the bottom of the page to the section entitled **Network Communications** to view the active protocol and baud rate.

3 BMS configured to speak 2-wire EIA-485 to the controller.

4 Both set to 8 data bits, No Parity, and 1 stop bit.

5 Rotary address switches set for the controller's unique slave address.

6 Proper connection wiring.

7 BMS reading or writing to the proper network point addresses in the controller - download the latest points list for the controller to verify.

8 BAS is sending requests to the proper slave address of the controller.

NOTE See Appendix (C) for N2 Protocol Conformance Statement

LonWorks

Verify BMS and controller settings:

1 Both set to speak LonWorks protocol.

- Comm Selector DIP Switches 3, 4, 5, and 6
- By getting a Modstat of the controller through the BACview. Click and hold the **FN** key and then click the **.** key at the same time. Scroll to the bottom of the page to the section entitled **Network Communications** to view the active protocol and baud rate.

2 Both set for 9600 baud rate.

- Comm Selector DIP switches **1 and 2.**
- By getting a Modstat of the controller through the BACview. Click and hold the **FN** key and then click the **.** key at the same time. Scroll to the bottom of the page to the section entitled **Network Communications** to view the active protocol and baud rate.

3 Configure BMS to speak 2—wire EIA-485 to the SLTA or LonWorks Plug-in.

4 Proper connection wiring.

5 BMS reading or writing to the proper network point addresses in the controller - download the latest points list for the controller to verify.

6 BMS is sending requests to the proper slave address of the controller from the SLTA-10 or LonWorks Plug-in card.

NOTE See Appendix (D) for the LonWorks Protocol Conformance Statement

Commissioning the I/O Flex 6126 for LonWorks

Before a device can communicate on a LonWorks network, it must be commissioned. Commissioning allows the system integrator to associate the device hardware with the LonWorks system's network layout diagram. Together, the I/O Flex 6126 and its LonWorks Option Card serve as a single LonWorks device or node. This is done using the device's unique Neuron ID.

A network management tool such as Echelon's LonMaker is used to commission each device, as well as to assign addressing. Specific instructions regarding the commissioning of LonWorks devices should be obtained from documentation supplied with the LonWorks Network Management Tool.

When a new device is first commissioned onto the LonWorks network, the system integrator must upload the device's External Interface File (XIF) information. LonWorks uses the XIF to determine the points (network variables) that are available from a device. The I/O Flex 6126 has a set of predefined network variables. These variables can be bound or accessed by the network management tool.

The network variables defined on the I/O Flex 6126 Network Variables Property pages determine its XIF information. If any information is changed, added, or deleted on the Network Variable Property pages, the I/O Flex 6126 must be removed from the network management tool's database and recommissioned, including uploading the XIF information again.

There are some issues with LonWorks that should be considered when using the I/O Flex 6126:

Device Configuration Information (XIF)

- When members of the object cache are modified, you must modify the device configuration information (XIF) from that originally imported into the LonWorks Network Management Tool. The new information will not be recognized by the Network Management Tool until it is imported again from the I/O Flex 6126.
- The user must first undefine all of the network variable bindings and the device, recommission the device, and establish the network variable bindings again.
- Modifications to the object cache should be avoided once the device is fully commissioned and operational. Any modifications to the addressing schemes should also be avoided once the I/O Flex 6126 is commissioned.

Address parameters

- If the address parameters are modified, the LonWorks Option Card will be set to **Node Offline**, and **Unconfigured**, which means it no longer communicates with the LonWorks network.
- This does not require deletion or importing the device configuration information again, but does require the device to be recommissioned by the Network Management Tool.

Point configuration

- When the I/O Flex 6126 is first commissioned onto the LonWorks network, the system integrator should use the **Browse** features of the network Management Tool to check the data that is available from the controller.
- Any changes in point count and point configuration should be made prior to performing any further system integration.
- I/O Flex 6126 may be deleted and re-imported as many times as necessary to ensure that the points are correct.

NOTE For these reasons, all parameters on the module driver parameter page should be configured prior to connecting this device to a LonWorks network.

The **Browse** feature of the Network Management Tool also allows you to read real-time values from the I/O Flex 6126. The Tool allows you to test integration prior to binding the I/O Flex 6126's network variables to other LonWorks nodes.

See *Appendix E* (page 46) for instructions on obtaining your LonWorks object mapping (XIF) file.

Communication LED's

The LED's indicate if the controller is speaking to the devices on the network. The LED's should reflect communication traffic based on the baud rate set. The higher the baud rate the more solid the LED's become.

LEDs	Status
Power	Lights when power is being supplied to the controller. **NOTE** The I/O Flex 6126 is protected by internal solid state Polyswitches on the incoming power and network connections. These Polyswitches are not replaceable, but they will reset themselves if the condition that caused the fault returns to normal.

LEDs	Status
Rx	Lights when the controller receives data from the network segment; there is an Rx LED for Ports 1 and 2.
Tx	Lights when the controller transits data from the network segment; there is an Rx LED for Ports 1 and 2.
Run	Lights based on controller health.
Error	Lights based on controller health.

The **Run** and **Error** LED's indicate controller and network status.

If Run LED shows...	And Error LED shows...	Status is...
2 flashes per second	Off	Normal
2 flashes per second	2 flashes, alternating with **Run** LED	Five minute auto-restart delay after system error
2 flashes per second	3 flashes, then off	The controller has just been formatted
Alternating flashes with the **Error** LED	Alternating flashes with the **Run** LED	The controller files have been archived
2 flashes per second	4 flashes, then pause	Two or more devices on this network have the same ARC156 network address
2 flashes per second	1 flash per second	The controller is alone on the network
2 flashes per second	On	Exec halted after frequent system errors or control programs halted
5 flashes per second	On	Exec start-up aborted, Boot is running
5 flashes per second	Off	Firmware transfer in progress, Boot is running
7 flashes per second	7 flashes per second, alternating with **Run** LED	Ten second recovery period after brownout
14 flashes per second	14 flashes per second, alternating with **Run** LED	Brownout
On	On	Failure. Try the following solutions: • Turn the I/O Flex 6126 off, then on. • Format the I/O Flex 6126. • Download memory to the I/O Flex 6126. • Replace the I/O Flex 6126.

Compliance

FCC Compliance

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. 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 instruction manual, may cause harmful interference to 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 own expense.

CAUTION Changes or modifications not expressly approved by the responsible party for compliance could void the user's authority to operate the equipment.

CE Compliance

WARNING This is a Class A product. In a domestic environment, this product may cause radio interference in which case the user may be required to take adequate measures.

BACnet Compliance

BACnet® is a registered trademark of ASHRAE. ASHRAE does not endorse, approve or test products for compliance with ASHRAE standards. Compliance of listed products to requirements of ASHRAE Standard 135 is the responsibility of the BACnet manufacturers Association (BMA). BTL® is a registered trademark of the BMA.

Appendix (A) BACnet Protocol Implementation Conformance Statement

Date: 11/22/05

Vendor Name: **OEM**

Product Names: **I/O Flex 6126**

Product Model Number: **IOF6126**

Applications Software Version: **HW_Exec_B DRV_IOFLEX** Firmware Revision: **2.03**

BACnet Protocol Revision: **3**

Product Description:

The I/O Flex 6126 is a general purpose building management controller with programmable functionality, designed for controlling various pieces of equipment. BACnet objects are spawned within the device as a result of downloading graphical control programs. It supports one PTP connection with dial-up modem support. The I/O Flex 6126 can operate as an MS/TP master or slave node.

BACnet Standardize Device Profile (Annex L): B-AAC

List of all BACnet Interoperability Building Blocks Supported (Annex K):

DS-RP-A*	AE-N-I-B	SCHED-I-B	T-VMT-I-B	DM-DDB-A*
DS-RP-B	AE-ACK-B	SCHED-E-B	T-ATR-B	DM-DDB-B
DS-RPM-A*	AE-ASUM-B			DM-DOB-A*
DS-RPM-B	AE-INFO-B			DM-DOB-B
DS-WP-A*	AE-ESUM-B			DM-DCC-B
DS-WP-B				DM-PT-A*
DS-WPM-A*				DM-PT-B
DS-WPM-B				DM-TS-B
DS–COV-A*				DM-UTC-B
DS–COV-B				DM-RD-B
DS–COVU-A*				DM-LM-B
DS–COVU-B				DM-BR-B
				NM-CE-A*

* Dynamic Binding is not supported when MS/TP is configured in slave mode.

Segmentation Capability:

Able to transmit segmented messages: (NO) Window Size:

Able to receive segmented messages: (NO) Window Size:

Standard Object Types Supported:

On a separate page, please list each standard Object Type supported (i.e., an object of this type may be present in the product). For each standard Object Type supported provide the following data:

1. Whether objects of this type are dynamically creatable using BACnet's CreateObject service
2. Whether objects of this type are dynamically deletable using BACnet's CreateObject service
3. List of all optional properties supported
4. List of all properties that are writable where not otherwise required by this standard
5. List of proprietary properties and for each its property identifier, datatype, and meaning
6. List of any property range restrictions

BACnet Data Link Layer Options

Data Link Layer Options:

☐☐☐BACnet IP, (Annex J)

☐☐☐Able to register as a Foreign Device

☐☐☐ISO 8802-3, Ethernet (Clause 7)

☐☐☐ANSI/ ATA 878.1, 2.5 Mb ARCNET (Clause 8)

XX ANSI/ATA 878.1, RS-485 ARCNET (Clause 8) baud rate(s) 156k baud

XX MS/TP master (Clause 9), baud rate(s): 9600, 19200, 38400, 76800

XX MS/TP slave (Clause 9), baud rate(s): 9600, 19200, 38400, 76800

XX Point-To-Point, EIA 232 (Clause 10), baud rate(s): 9600, 19200, 38400, 76800

XX Point-To-Point, modem, (Clause 10), baud rate(s): 9600, 19200, 38400, 76800

☐☐☐LonTalk, (Clause 11), medium: _____

☐☐☐Other:

Device Address Binding Methods Supported:

XX Send Who-Is, receive I-Am (BIBB DM-DDB-A)*

XX Receive Who-Is, send I-Am (BIBB DM-DDB-B)

XX Send Who-Has, receive I-Have (BIBB DM-DOB-A)*

XX Receive Who-Has, send I-Have (BIBB DM-DOB-B)

XX Manual configuration of recipient device's network number and MAC address.

☐☐None of the above

* Dynamic Binding is not supported when MS/TP is configured as a slave node.

***Networking Options:**

XX Router, Clause 6 - List all routing configurations, e.g., ARCNET-Ethernet, Ethernet-MS/TP, etc.

ARCNET-PTP, MS/TP-PTP, ARCNET-MS/TP, ARCNET-MS/TP-UDP/IP, ARCNET-MS/TP-PTP.

☐☐☐Annex H.3, BACnet Tunneling Router over UDP/ IP

☐☐☐BACnet/ IP Broadcast Messaging Device (BBMD)

Does the BBMD support registrations by Foreign Devices? ☐☐Yes ☐☐No

Character Sets Supported:

Indicating support for multiple character sets does not imply that they can all be supported simultaneously.

XX ANSI X3.4

XX IBM™/Microsoft™☐DBCS

XX ISO 8859-1

XX ISO 10646 (UCS-2)

XX ISO 10646 (ICS-4)

XX JIS C 6226

If this product is a communication gateway, describe the non-BACnet equipment/networks(s) that the gateway supports:

Various protocols, depending on which firmware is loaded.

Analog Input

Analog Input, Analog Input (TLO), Analog Input (PTA), Analog Input (RS):			
1. Creatable?	NO		
2. Deletable?	NO		
3. Optional Properties Supported:	acked_transitions cov_increment deadband description device_type event_enable	event_time_stamps high_limit limit_enable low_limit max_pres_value min_pres_value	notification_class notify_type reliability resolution time_delay update_interval
4. Writeable Properties:	cov_increment deadband description device_type event_enable	high_limit limit_enable low_limit notification_class	notify_type out_of_service time_delay units
5. Proprietary properties:	None		
6. Range Restrictions:			
	description	limited to 50 octets in length	
	device_type	imited to 50 octets in length	
	present_value	limited by min_pres_value and max_pres_value properties	
	notification_class	must be valid notification_class	
	time_delay	0 to 4294967295	

Analog Output

Analog Output, Analog Output (FM), and Analog Output (PWM):			
1. Creatable?	NO		
2. Deletable?	NO		
3. Optional Properties Supported:	acked_transitions cov_increment deadband description device_type event_enable	event_time_stamps high_limit limit_enable low_limit max_pres_value	min_pres_value notification_class notify_type reliability resolution time_delay
4. Writeable Properties:	cov_increment deadband description device_type event_enable	high_limit limit_enable low_limit notification_class notify_type	out_of_service present_value relinquish_default time_delay units
5. Proprietary properties:	None		
6. Range Restrictions:			
	description	limited to 50 octets in length	
	device_type	limited to 50 octets in length	
	present_value	**Priority 16 is under the control of a local control program, so writes at this priority are ineffective**.	
	relinquish_default	limited by min_pres_value and max_pres_value properties	
	notification_class	must be valid notification_class	
	time_delay	0 to 4294967295	

Analog Value (PAR)

Analog Value (PAR):			
1. Creatable?	NO		
2. Deletable?	NO		
3. Optional Properties Supported:	acked_transitions cov_increment deadband description device_type event_enable	event_time_stamps high_limit limit_enable low_limit notification_class	notify_type priority_array reliability relinquish_default time_delay

Analog Value (PAR):			
4. Writeable Properties:	cov_increment deadband description event_enable high_limit	limit_enable low_limit notification_class notify_type	out_of_service present_value relinquish_default time_delay units
5. Proprietary properties:	None		

6. Range Restrictions:		
	description	limited to 50 octets in length
	present_value	limited by min_pres_value and max_pres_value properties
	relinquish_default	limited by min_pres_value and max_pres_value properties
	notification_class	must be valid notification_class
	time_delay	0 to 4294967295

Analog Value (RS)

Analog Value (RS):	
1. Creatable?	NO
2. Deletable?	NO
3. Optional Properties Supported:	reliability
4. Writeable Properties:	out_of_service present_value units
5. Proprietary properties:	None
6. Range Restrictions:	None

Analog Value (STAT)

Analog Value (STAT):	
1. Creatable?	NO
2. Deletable?	NO

Analog Value (STAT):			
3. Optional Properties Supported:	acked_transitions cov_increment deadband description device_type	event_enable event_time_stamps high_limit limit_enable	low_limit notification_class notify_type reliability time_delay
4. Writeable Properties:	cov_increment deadband description event_enable	high_limit limit_enable low_limit notification_class	notify_type out_of_service time_delay units
5. Proprietary properties:	None		
6. Range Restrictions:			
	description	limited to 50 octets in length	
	notification_class	must be valid notification_class	
	time_delay	0 to 4294967295	

Binary Input

Binary Input:			
1. Creatable?	NO		
2. Deletable?	NO		
3. Optional Properties Supported:	acked_transitions active_text alarm_value change_of_state_count change_of_state_time description	device_type elapsed_active_time event_enable event_time_stamps inactive_text notification_class	notify_type reliability time_delay time_of_active_time_reset time_of_state_count_reset
4. Writeable Properties:	active_text alarm_value change_of_state_count description device_type	elapsed_active_time event_enable inactive_text notification_class	notify_type out_of_service polarity time_delay
5. Proprietary properties:	None		
6. Range Restrictions:			
	active_text	limited to 50 octets in length	
	change_of_state_count	0 to 4294967295	
	description	limited to 50 octets in length	
	device_type	limited to 50 octets in length	
	elapsed_active_time	0 to 4294967295	
	inactive_text	limited to 50 octets in length	

Binary Input:		
	notification_class	must be valid notification_class
	time_delay	0 to 4294967295

Binary Output

Binary Output:			
1. Creatable?	NO		
2. Deletable?	NO		
3. Optional Properties Supported:	acked_transitions active_text change_of_state_count change_of_state_time description device_type elapsed_active_time	event_enable event_time_stamps feedback_value inactive_text minimum_off_time minimum_on_time	notification_class notify_type reliability time_delay time_of_active_time_reset time_of_state_count_reset
4. Writeable Properties:	active_text change_of_state_count description device_type elapsed_active_time	event_enable inactive_text minimum_off_time minimum_on_time notification_class	notify_type out_of_service polarity relinquish_default time_delay
5. Proprietary properties:	None		
6. Range Restrictions:			
	active_text	limited to 50 octets in length	
	change_of_state_count	0 to 4294967295	
	description	limited to 50 octets in length	
	device_type	limited to 50 octets in length	
	elapsed_active_time	0 to 4294967295	
	inactive_text	limited to 50 octets in length	
	minimum_off_time	0 to 4294967295	
	minimum_on_time	0 to 4294967295	
	notification_class	must be valid notification_class	
	time_delay	0 to 4294967295	
	present_value	**Priority 16 is under the control of a local control program, so writes at this priority are ineffective.**	

Binary Value (PAR), (CLOCK), and (STAT)

Binary Value (PAR) and Binary Value (CLOCK) and Binary Value (STAT):			
1. Creatable?	NO		
2. Deletable?	NO		
3. Optional Properties Supported:	acked_transitions active_text alarm_value change_of_state_count change_of_state_time description elapsed_active_time	event_enable event_time_stamps inactive_text minimum_off_time minimum_on_time notification_class notify_type	priority_array reliability relinquish_default time_delay time_of_active_time_reset time_of_state_count_reset
4. Writeable Properties:	active_text alarm_value change_of_state_count description elapsed_active_time	event_enable inactive_text minimum_off_time minimum_on_time notification_class	notify_type out_of_service present_value relinquish_default time_delay
5. Proprietary properties:	None		
6. Range Restrictions:			
	active_text	limited to 50 octets in length	
	change_of_state_count	0 to 4294967295	
	description	limited to 50 octets in length	
	elapsed_active_time	0 to 4294967295	
	inactive_text	limited to 50 octets in length	
	minimum_off_time	0 to 4294967295	
	minimum_on_time	0 to 4294967295	
	notification_class	must be valid notification_class	
	time_delay	0 to 4294967295	

Binary Value (MODULE ALARM)

Binary Value (MODULE ALARM):			
1. Creatable?	NO		
2. Deletable?	NO		
3. Optional Properties Supported:	acked_transitions alarm_value description	event_enable event_time_stamps notification_class	notify_type time_delay

Binary Value (MODULE ALARM):		
4. Writeable Properties:	alarm_value event_enable notification_class	notify_type out_of_service time_delay
5. Proprietary properties:	None	
6. Range Restrictions:		
	notification_class	must be valid notification_class
	time_delay	0 to 4294967295

Calendar

Calendar:		
1. Creatable?	NO	
2. Deletable?	NO	
3. Optional Properties Supported:	description	
4. Writeable Properties:	date_list description	
5. Proprietary properties:	None	
6. Range Restrictions:		
	date_list	limited to 30 BACnetCalendarEntrys
	description	limited to 50 octets in length

Device

Device:			
1. Creatable?	NO		
2. Deletable?	NO		
3. Optional Properties Supported:	active_cov_subscriptions active_vt_sessions apdu_segment_timeout backup_failure_timeout configuration_files database_revision daylight_savings_status	description last_restore_time list_of_session_keys local_date local_time location	max_info_frames max_master time_synchronization_recipients utc_offset vt_classes_supported

Device:			
4. Writeable Properties:	apdu_segment_timeout apdu_timeout backup_failure_timeout description local_date	local_time location max_info_frames max_master number_of_apdu_retries	object_identifier object_name time_synchronization_recipients utc_offset
5. Proprietary properties:	None		
6. Range Restrictions:			
	description	limited to 50 octets in length	
	location	limited to 50 octets in length	
	max_master	1 to 127	
	object_identifier	must be valid device identifier	
	object_name	limited to 50 octets in length	
	utc_offset	_780 to 780	

File

	File:		
1. Creatable?	NO		
2. Deletable?	NO		
3. Optional Properties Supported:	description		
4. Writeable Properties:	archive description	file_type modification_date	read_only
5. Proprietary properties:	None		
6. Range Restrictions:			
	description	limited to 50 octets in length	
	file_type	limited to 50 octets in length	

Multi_State Value

Multi_State Value (PAR): **Multi_State Value (CLOCK):**			
1. Creatable?	NO		
2. Deletable?	NO		
3. Optional Properties Supported:	acked_transitions alarm_values description event_enable event_time_stamps	fault_values notification_class notify_type priority_array	reliability relinquish_default state_text time_delay
4. Writeable Properties:	alarm_values description event_enable fault_values	notification_class notify_type out_of_service	present_value relinquish_default time_delay
5. Proprietary properties:	None		
6. Range Restrictions:			
	description	limited to 50 octets in length	
	notification_class	must be valid notification_class	
	present_value	must be valid state	
	relinquish_default	must be valid state	
	time_delay	0 to 4294967295	

Multi_State Value (STAT)

Multi_State Value (STAT):			
1. Creatable?	NO		
2. Deletable?	NO		
3. Optional Properties Supported:	acked_transitions alarm_values description event_enable	event_time_stamps fault_values notification_class notify_type	reliability state_text time_delay
4. Writeable Properties:	alarm_values description event_enable	fault_values notification_class notify_type	out_of_service time_delay
5. Proprietary properties:	None		
6. Range Restrictions:			
	description	limited to 50 octets in length	
	notification_class	must be valid notification_class	

Multi_State Value (STAT):		
	present_value	must be valid state
	time_delay	0 to 4294967295

Notification Class

Notification Class:		
1. Creatable?	NO	
2. Deletable?	NO	
3. Optional Properties Supported:	description	
4. Writeable Properties:	ack_required description	priority recipient_list
5. Proprietary properties:	None	
6. Range Restrictions:		
	description	limited to 50 octets in length
	recipient_list	limited to 5 BACnetDestinations

Program

Program:		
1. Creatable?	NO	
2. Deletable?	NO	
3. Optional Properties Supported:	description description_of_halt instance_of	program_location reason_for_halt reliability
4. Writeable Properties:	description program_change program_location	
5. Proprietary properties:	None	
6. Range Restrictions:		
	description	limited to 50 octets in length
	program_location	limited to 50 octets in length

Schedule

Schedule (ENUM) and Schedule (UNS):			
1. Creatable?	NO		
2. Deletable?	NO		
3. Optional Properties Supported:	priority_for_writing description	exception_schedule weekly_schedule	
4. Writeable Properties:	description effective_period exception_schedule	list_of_object_property_references present_value	priority_for_writing weekly_schedule
5. Proprietary properties:	None		
6. Range Restrictions:			
	description	limited to 50 octets in length limited to 30 BACnetSpecialEvents	
	exception_schedule	each being limited to 6 BACnetTimeValues	
	present_value	0 to 4294967295	
	weekly_schedule	limited to 6 BACnetTimeValues per BACnetDailySchedule	

Trend_log

Trend_log:			
1. Creatable?	NO		
2. Deletable?	NO		
3. Optional Properties Supported:	acked_transitions client_cov_increment cov_resubscription_interval description event_enable	event_time_stamps last_notify_record log_deviceobjectproperty log_interval notification_class	notification_threshold notify_type records_since_notification start_time stop_time
4. Writeable Properties:	buffer_size client_cov_increment cov_resubscription_interval description event_enable	log_enable log_interval notification_class notification_threshold notify_type	 record_count start_time stop_time stop_when_full
5. Proprietary properties:	None		
6. Range Restrictions:			

Trend_log:		
	description	limited to 50 octets in length
	log_interval	0 to 4294967295
	notification_class	must be valid notification_class
	notification_threshold	0 to 4294967295
	record_count	can only be written to 0

Trend_log (Non-BACnet Property)

Trend_log (of a non-BACnet property):			
1. Creatable?	NO		
2. Deletable?	NO		
3. Optional Properties Supported:	acked_transitions client_cov_increment cov_resubscription_interval description event_enable	event_time_stamps last_notify_record log_interval notification_class notification_threshold	notify_type records_since_notification start_time stop_time
4. Writeable Properties:	buffer_size client_cov_increment cov_resubscription_interval description event_enable	log_enable log_interval notification_class notification_threshold notify_type	record_count start_time stop_time stop_when_full
5. Proprietary properties:	None		
6. Range Restrictions:			
	description	limited to 50 octets in length	
	log_interval	0 to 4294967295	
	notification_class	must be valid notification_class	
	notification_threshold	0 to 4294967295	
	record_count	can only be written to 0	

Appendix (B) Modbus Protocol Implementation Conformance Statement

Date: 11/22/05

Vendor Name: **OEM**

Product Name: **I/O Flex 6126**

Applications Software Version: **HW_Exec_B DRV_IOFLEX** Firmware Revision: **2.03**

Product Description:

The I/O Flex 6126 is a general purpose building management controller with custom programmable functionality, designed for communicating through multiple protocols. Modbus registers are spawned within the device as a result of downloading graphical control programs. The I/O Flex 6126 controller speaks the Modicon Modbus RTU/ASCII Protocol as described in the Modicon Modbus Protocol Reference Guide, PI-MBUS-300 Rev.J, and acts as a Modbus Master or Slave. Further details on the Modbus supported implementation are described below.

Serial Transmission Mode:	Supported?
RTU	Master or Slave (Slave RTU is the Default Dipswitch setting)
ASCII	Master or Slave

Communication Types:	Baud rates:	Data Bits:	Parity:	Stop Bits:
2-wire EIA-485, 4-wire EIA-485, or EIA-232	9600, 19200, 38400, 76800	8	None	1

Function Codes:	Purpose:	Used with Register Numbers:
01 – Read Coil Status	Read Discrete Outputs	00001 - 09999
02 – Read Input Status	Read Discrete Inputs	10001 - 19999
03 – Read Holding Registers	Read Holding Registers	40001 - 49999
04 – Read Input Registers	Read Input Registers	30001 - 39999
05 – Force Single Coil	Write Discrete Outputs (single)	00001 - 09999
06 – Preset Single Register	Write Holding Registers (single)	40001 - 49999
15 – Force Multiple Coils	Write Discrete Outputs	00001 - 09999
16 – Preset Multiple Coils	Write Holding Registers	40001 - 49999

Register Type:	**Range:**	**Function Codes Used with this Register Type:**
Float Value (FLOAT)	Single-Precision IEEE floating point value	3 – Read Holding Register 6 – Preset Single Register 16 – Preset Multiple Register
Unsigned Integer (UINT)	0 - 65535	3 – Read Holding Register 6 – Preset Single Register 16 – Preset Multiple Register
Signed Integer (SINT)	-32768 - 32767	3 – Read Holding Register 6 – Preset Single Register 16 – Preset Multiple Register
Discrete Input (DI)	0 = Off, 1 = On	2 – Read Input Status
Discrete Output (DO)	0 = Off, 1 = On	1 – Read Coil Status 5 – Force Single Coil 15 – Force Multiple Coils

Appendix (C) Johnson Controls N2 Protocol Implementation Conformance Statement

Date: 11/22/05

Vendor Name: **OEM**

Product Name: **I/O Flex 6126**

Applications Software Version: **HW_Exec_B DRV_IOFLEX** Firmware Revision: **2.03**

Product Description:

The I/O Flex 6126 is a general purpose building automation controller with custom programmable functionality, designed for communicating through multiple protocols. N2 network points are spawned within the device as a result of downloading graphical control programs. The I/O Flex 6126 controller speaks the Johnson N2 Open Protocol as described in the Metasys N2 System Protocol Specification (for Vendors) document, revision 6/13/96, and acts as an N2 Master or Slave. Further details on the N2 supported implementation are described below.

Serial Transmission Mode:	Supported?
N2 Open	Master or Slave (Slave is the Default Dipswitch setting)

Communication Types:	Baud rates:	Data Bits:	Parity:	Stop Bits:
2-wire EIA-485	9600	8	None	1

Network Point Types:
Analog Inputs (AI)
Binary Inputs (BI)
Analog Outputs (AO)
Binary Outputs (BO)
Internal Floats (ADF)
Internal Integers (ADI)
Internal Bytes (BD)

Protocol Commands:
Identify Device Type
Sync Time
Poll Without Acknowledge

Poll With Acknowledge
Read Analog Input
Read Binary Input
Read Analog Output
Read Binary Output
Read Internal Parameter
Write Analog Input
Write Binary Input
Write Analog Output
Write Binary Output
Write Internal Parameter
Override Analog Input
Override Binary Input
Override Internal Parameter
Override Release Request

Appendix (D) LonWorks Protocol Implementation Conformance Statement

Date: 11/22/05

Vendor Name: **OEM**

Product Name: **I/O Flex 6126**

Applications Software Version: **HW_Exec_B DRV_IOFLEX** Firmware Revision: **2.03**

Product Description:

The I/O Flex 6126 is a general purpose building automation controller with custom programmable functionality, designed for communicating through multiple protocols. LonWorks network points are spawned within the device as a result of downloading graphical control programs. The I/O Flex 6126 controller speaks the LonWorks Protocol as described by Echelon Protocol Specification. Since the controller is custom programmable it does not conform to LonMark certification. Further details on the LonWorks supported implementation are described below.

The FT 3120 Free Topology Smart Transceiver is fully compatible with the TP/FT-10 channel and can communicate with devices using Echelon's FTT-10A Free Topology Transceiver. The free topology transceiver supports polarity insensitive cabling using a star bus, daisy-chain, loop, or combination topology.

Serial Transmission Mode:	Supported?
LonWorks	Master or Slave (Slave is the Default Dipswitch setting)

Communication Types:	Baud rates:	Data Bits:	Parity:	Stop Bits:
2-wire EIA-485	variable	8	None	1

The controller supports the following SNVT listing as noted by the Echelon Protocol Specification.

SNVT_abs_humid	SNVT_elec_whr	SNVT_mass_kilo	SNVT_speed
SNVT_address	SNVT_elec_whr_f	SNVT_mass_mega	SNVT_speed_f
SNVT_alarm	SNVT_enthalpy	SNVT_mass_mil	SNVT_speed_mil
SNVT_alarm_2	SNVT_evap_state	SNVT_motor_state	SNVT_state
SNVT_amp	SNVT_ex_control	SNVT_muldiv	SNVT_state_64
SNVT_amp_ac	SNVT_file_pos	SNVT_multiplier	SNVT_str_asc
SNVT_amp_f	SNVT_file_req	SNVT_obj_request	SNVT_str_int
SNVT_amp_mil	SNVT_file_status	SNVT_obj_status	SNVT_switch
SNVT_angle	SNVT_fire_indcte	SNVT_occupancy	SNVT_telcom
SNVT_angle_deg	SNVT_fire_init	SNVT_override	SNVT_temp
SNVT_angle_f	SNVT_fire_test	SNVT_ph	SNVT_temp_diff_p
SNVT_angle_vel	SNVT_flow	SNVT_ph_f	SNVT_temp_f
SNVT_angle_vel_f	SNVT_flow_f	SNVT_pos_ctrl	SNVT_temp_p
SNVT_area	SNVT_flow_mil	SNVT_power	SNVT_temp_ror

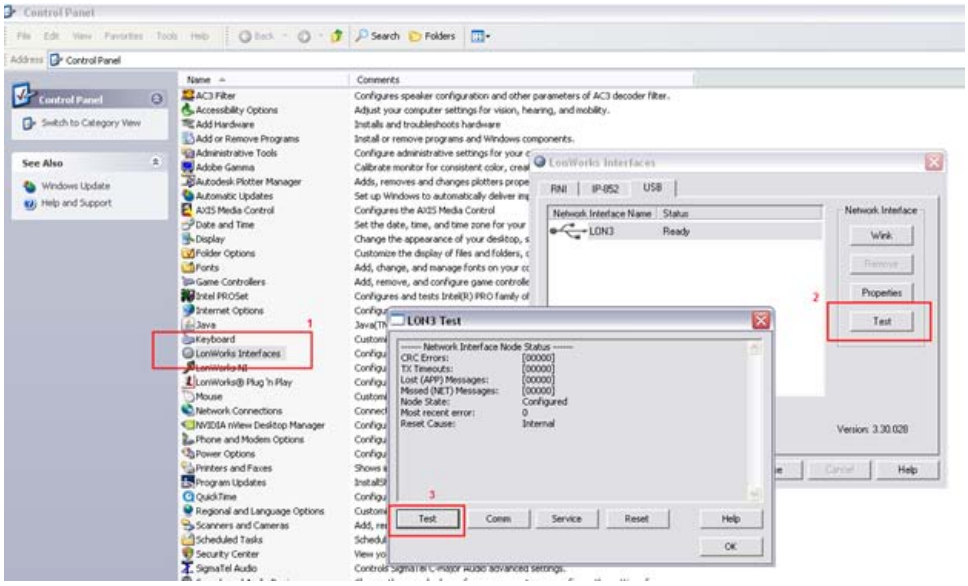
SNVT_btu_f	SNVT_flow_p	SNVT_power_f	SNVT_temp_setpt
SNVT_btu_kilo	SNVT_freq_f	SNVT_power_kilo	SNVT_therm_mode
SNVT_char_ascii	SNVT_freq_hz	SNVT_ppm	SNVT_time_f
SNVT_char_mega	SNVT_freq_kilohz	SNVT_ppm_f	SNVT_time_hour
SNVT_chlr_status	SNVT_freq_milhz	SNVT_preset	SNVT_time_min
SNVT_color	SNVT_gfci_status	SNVT_press	SNVT_time_passed
SNVT_config_src	SNVT_grammage	SNVT_press_f	SNVT_time_sec
SNVT_count	SNVT_grammage_f	SNVT_press_p	SNVT_time_stamp
SNVT_count_f	SNVT_hvac_emerg	SNVT_privacyzone	SNVT_time_zone
SNVT_count_inc	SNVT_hvac_mode	SNVT_ptz	SNVT_tod_event
SNVT_count_inc_f	SNVT_hvac_override	SNVT_pumpset_mn	SNVT_trans_table
SNVT_ctrl_req	SNVT_hvac_status	SNVT_pumpset_sn	SNVT_turbidity
SNVT_ctrl_resp	SNVT_hvac_type	SNVT_pump_sensor	SNVT_turbidity_f
SNVT_currency	SNVT_ISO_7811	SNVT_pwr_fact	SNVT_valve_mode
SNVT_date_cal	SNVT_length	SNVT_pwr_fact_f	SNVT_vol
SNVT_date_day	SNVT_length_f	SNVT_reg_val	SNVT_volt
SNVT_date_time	SNVT_length_kilo	SNVT_reg_val_ts	SNVT_volt_ac
SNVT_defr_mode	SNVT_length_micr	SNVT_res	SNVT_volt_dbmv
SNVT_defr_state	SNVT_length_mil	SNVT_res_f	SNVT_volt_f
SNVT_defr_term	SNVT_lev_cont	SNVT_res_kilo	SNVT_volt_kilo
SNVT_density	SNVT_lev_cont_f	SNVT_rpm	SNVT_volt_mil
SNVT_density_f	SNVT_lev_disc	SNVT_scene	SNVT_vol_f
SNVT_dev_c_mode	SNVT_lev_percent	SNVT_scene_cfg	SNVT_vol_kilo
SNVT_earth_pos	SNVT_lux	SNVT_setting	SNVT_vol_mil
SNVT_elapsed_tm	SNVT_magcard	SNVT_smo_obscur	SNVT_zerospan
SNVT_elec_kwh	SNVT_mass	SNVT_sound_db	
SNVT_elec_kwh_1	SNVT_mass_f	SNVT_sound_db_f	

Appendix E: Obtaining Lonworks object mapping (XIF file)

1 Install Echelon U10 network interface device using supplied drivers (or visit the *Echelon website* (http://www.echelon.com) for driver downloads).

2 Verify proper installation of Echelon U10 network interface device.

1. Navigate to **Control Panel** and select **LonWorks Interfaces**.
2. Select the **USB** tab for a list of available USB network interface devices. Take note of the **Network Interface Name** to use later (**LON3** is the network interface name in the example shown below).
3. Click **Test** in the **LonWorks Interfaces** dialog box.
4. Click **Test** in the **LON3 Test** dialog box. Correct installation and test shown below.



NOTE If U10 installation problems occur, consult your U10 documentation or visit the *Echelon website* (http://www.echelon.com) for more assistance.

3 From Windows command prompt, launch **nodeutil** from NodeUtil install directory. Include **-D** and network interface device name in syntax.

The example below shows **LON3** as the network interface device name.

```
C:\BEF\OEM\NodeUtil>nodeutil -DLON3
```

```
Echelon Node Utility Release 1.82
Successfully installed TP/FT-10 network interface.
Welcome to the Echelon Node Utility application.
Activate the service pin on remote device to access it.
Enter one of the following commands by typing the indicated letter:

A -- (A)dd device to list.
D -- Set the (D)omain of the network interface.
E -- (E)xit this application.
F -- (F)ind devices in the current domain.
1 -- Find devices in all (1)-byte domains.
G -- (G)o to device menu....
H -- (H)elp with commands.
I -- Redirect (I)nput from a file.
L -- Display device (L)ist.
M -- Change device (M)ode or state.
O -- Redirect (O)utput to a file.
P -- Send a service (P)in message.
R -- (R)eboot 3150 device.
S -- Report device (S)tatus and statistics.
T -- (T)ransceiver parameters.
V -- Control (V)erbose modes.
W -- (W)ink a device.
Z -- Shell out to command prompt.
NodeUtil>
```

4 Press service pin on the Lon device to see the **Program ID** for the device. (OEM Program ID defaults to **PROG_ID**.)

```
NodeUtil> Received an ID message from device 1.
Program ID is OEM_Demo
NodeUtil>
```

5 Type **G** to go to device menu. You may be asked to **Enter node id for Neuron data structures (0-1)** - select the Lon device you wish to access, likely node **1**.

```
NodeUtil> (G)o to device menu...
Node ID Neuron ID          Program ID
0       04 30 23 E4 01 00  USBLTA                  *** network interface
1       04 5D 1E E5 02 00  OEM_Demo
Enter node id for Neuron data structures (0-1) [1]      :1
```

6 Type **X** for **Create device interface XIF file** in the device menu for the program in the controller.

```
Enter one of the following commands:
A -- Device (A)ddress table.
B -- (B)uffer configuration.
C -- Application (C)onfiguration structures.
D -- Device (D)omain table.
E -- (E)xit this menu and return to main menu.
F -- Configuration (F)iles.
G -- (G)o to another device.
H -- (H)elp with device commands.
I -- Network variable a1(I)as table.
J -- (J)am network variable type.
K -- Chec(K) Neuron executable.
L -- (L)ist network variables.
M -- Change device (M)ode or state.
N -- (N)etwork Variable configuration table.
O -- Redirect (O)utput to a file.
P -- (P)oll network variable.
Q -- (Q)uickly send a message.
R -- (R)ead device memory.
S -- Report device (S)tatus and statistics.
T -- (T)ransceiver parameters.
U -- (U)pdate input network variable.
V -- Control (V)erbose modes.
W -- (W)rite device memory.
X -- Create device interface ((X)IF) file.
Y -- Download Neuron executable.
DEVICE:1> Create device interface ((X)IF) file
------------------------------------------------------------
Self documentation structure length = 308, version number = 0
Number of declared NVs = 17, total NVs = 17, message tags = 3
------------------------------------------------------------
Enter output filename :[OEM_Demo.XIF]
```

Type "X" for create device XIF file

7 Modify output file name as required and/or click **Enter**. The default file name is **PROG_ID.xif** and the file will be saved to the nodeutil install directory. If you changed the **Program ID** parameter on the **Protocol Setup** page in WebCTRL, the default XIF file name changes accordingly. When completed, distribute file to the controls contractor as necessary.

```
File OEM_DEMO.XIF created successfully
DEVICE:1>
```

- Rev. 9/30/2010