



User's Manual

for the
LC1612A

Premium Series



Echelon, LonWorks, Lonbuilder, and Neuron are U.S. registered trademarks of Echelon Corporation. Nodebuilder, Neuron C, and Lonmark are trademarks of Echelon Corporation. Other trademarks are owned by their respective companies.

AP products are not designed for use in equipment or systems which involve danger to human health or safety or a risk of property damage and AP assumes no responsibility or liability for use of AP products in such applications.

AP makes and you receive no warranties or conditions, express, implied, statutory or in any communication with you, and AP specifically disclaim any implied warranty of merchantability or fitness for a particular purpose.

AP products are warranted for workmanship defects or component failures not caused by user for 1 year after date of purchase.

TABLE OF CONTENTS

1.	Introduction to the AP LC1612 DDC Controller	4
2.	The LC1612 features.....	5
3.	Hardware design overview	6
4.	Inputs and Outputs overview	7
5.	Programming and configuration	11
6.	The AP hardware Programming/configuration Tool	14
7.	Explanation of Standard Network Variable Types (SNVTs).....	19
8.	Network Management and LON Bindings with the Niagara Framework (To be added, later)	

Introduction to the AP LC16 DDC Controller

Thank you for purchasing this AP control product for deployment into or part of DDC Automation solutions.

The LC1612A is part of the AP *Premium* Products family and uses the LonWorks communication protocol for its information bus connection to the overall Automation system NCU and HMI.

Other versions of this controller are being developed and the product footprint includes the provision for the future addition a BACnet protocol stack and also, Modbus.

The AP LC1612 series contains core circuits to interface with analog inputs, analog outputs, and digital input signals (including dry contact closures), and outputs using relay contacts.

The LC1612 has been designed with the input of many years of ‘hands-on’ experience in the field, deploying integrated Automation Solutions in all types of Commercial, Public and Industrial facilities.

The key aspects of the hardware and software designs are as follows:

- Stability and reliability
- Easy to use, no complex programming and expensive tools
- High I/O point count, 28 hardware points in a small standalone electronic unit
- Low Cost
- Multiple protocol possibilities, including LonWorks, BACnet, Modbus and ITG proprietary
- Low heat emission
- Low EMC radiation emission
- Ability to use different Thermistor input types

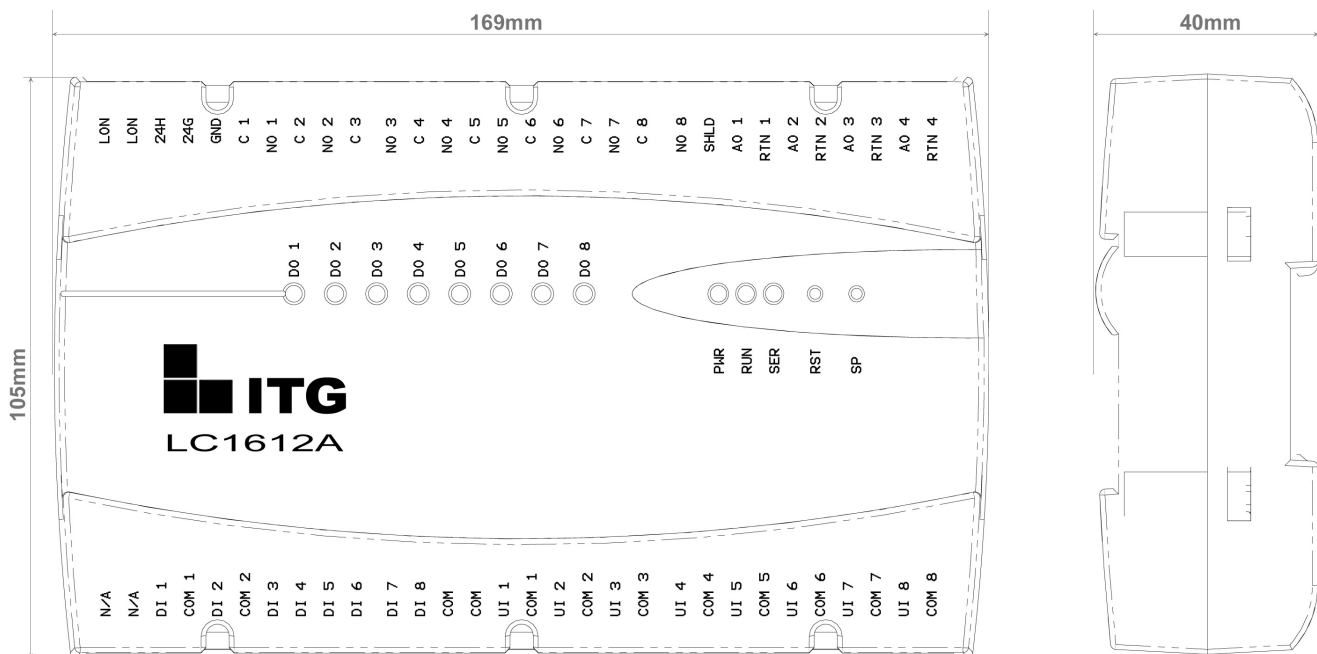
The LC1612 features

The LC1612 Multi-functional Controller uses the LonWorks Protocol and a special proprietary programming tool.

The Controller can be used to handle multiple automation applications.

In HVAC applications it is commonly used to control 4 AHU's/FCU's from 1 low cost controller.

- On board LED indication
- 28 Points with an excellent mix of inputs and outputs for optimum utilization
- 4 built in PID loops
- Easy to use RS485 link Programming tool
- Built in real time clock and time scheduler
- Additional memory and on board processing via additional chip to LON Neuron
- Customization of Analog inputs is possible with the programming Tool
- Ability to set and customize the curve for Thermistor inputs
- Compact, sleek design: 169 x 105 x 40



Hardware design overview

The LC1612 has two main processor chips.

One is the Echelon '3 in one' Neuron Processor and the other is the AP 'workhorse' proprietary processor which handles the programming a configuration of the Controller.

The Neuron LonWorks Chip has 3 built in Processors, one is the low level Network Processor, the other is a Protocol Processor and the other is an application processor.

By having 3 proc working in tandem on a single chip, the Neuron delivers the throughput with far less resources than a single conventional 16/32 bit Micro Controller (MCU).

The three processors in one Silicon are utilizing a multi thread architecture which enables high data throughput.

The Neuron splits each processor and is dedicated to a group of tasks that are mutually exclusive – if these tasks are being processed by a single processor there are inherently bottlenecks in a 16/32 conventional MCU

The DDC Controller package is designed with 4 processors, housed in only 2 Chips.

Three of the processor on single silicon is the Neuron engine and the other is the high performance I/O processor, resulting in fast real time automation of inputs and outputs.

The TMPN3150 contains three identical 8-bit central processing units (CPUs) which are dedicated to the following functions.

CPU-1 is the Media Access Control (MAC) CPU that handles layers one and two of the seven-layer LonTalk protocol stack. CPU-1 processing includes driving the communications sub-system hardware as well as executing the media access algorithm.

CPU-1 communicates with CPU-2 using network buffers located in shared memory.

CPU-2 is the Network CPU which implements layers three through six of the LonTalk protocol stack.

It handles network variable processing, addressing, transaction processing, authentication, background diagnostics, software timers and network management.

CPU-2 uses network buffers to communicate with CPU-1, and application buffers to communicate with CPU-3.

The buffers are also located in shared memory.

Access to them is mediated with hardware semaphores to resolve contention when updating shared data.

CPU-3 is the Application CPU.

It runs code written by the user, together with the operating system services called by applications code.

The programming language used by the application programmer is Neuron C, a derivative of the ANSI C language optimized and enhanced for LONWORKS distributed control applications.

The major enhancements are the following:

A built-in multi-tasking scheduler that allows the applications programmer to express logically parallel event-driven tasks in a natural way, and to control priority-based execution of these tasks.

A declarative syntax for input/output objects directly mapping into the input/output capabilities of the TMPN3150/3120.

- A declarative syntax for network variables, which are Neuron C language objects whose values are automatically propagated over the network whenever new values are assigned to them.
- A declarative syntax for millisecond and second timer objects which activate user tasks on expiration.
- A library of callable functions which can perform event checking, manage input/output activities, send and receive messages across the network, and control miscellaneous functions of the TMPN3150/3120.

The support for all these capabilities is part of the TMPN3160/3120XX firmware and does not need to be programmed by the user.

Each of the three identical CPUs has its own register set, but all three CPUs share data and address ALUs and memory access circuitry.

Each CPU minor cycle consists of three system clock cycles, or phases.

In turn, each system clock cycle is comprised of two input clock cycles.

The minor cycles of the three CPUs are offset from one another by one system clock cycle, so that each CPU can access memory and ALUs once during each minor cycle.

The active elements for each CPU during one of the three phases of a minor cycle.

The system thus pipelines the three processors, reducing hardware requirements without affecting performance.

This allows the execution of three processes in parallel without time-consuming interrupts and context switching.

Inputs and Outputs

Universal Analog Inputs

The Analog Inputs are "universal" inputs which can interface with almost any type of common analog or digital input.

These channels can be configured for measuring the following kinds of inputs:

- 1) 0-5 Volts
- 2) 0-10 Volts
- 3) 0-20 mA Current
- 4) Resistive sensors
- 5) Dry contacts (e.g. switches), or TTL compatible inputs.

The LC1612 has 8 Universal Inputs and can accept inputs as follows:

- 1) $K\Omega$ Thermistor with 11 $K\Omega$ Shunt -40 °C to 121°C range
- 2) 10 $K\Omega$ Resistance 0 to 10.5 $K\Omega$
- 3) Voltage 0 to 10Vdc
- 4) Current 0 to 20mA requires 250 Ω Shunt resistor

Resistive Sensor Measurements

A key feature of the LC1612 is the ability for the user to configure and customize the 'curve' for Thermistor inputs.

The Controller can also accept 10K and 20K ohm type RTD Thermistor for measuring temperature.

Each Universal Input can be individually configured to resistive sensors, such as Thermistor, RTDs, etc.

There is a selection of Jumper headers that are used to configure the Analog Inputs.

Shunt jumpers on the headers define the configurations, summarized later in this section and covered in detail in this chapter.

Analog Outputs

The LC1612 has four channels of Analog Outputs which can output 0-10 Volts or 0-20mA.

Analog Output characteristics are: Voltage 0 to 10Vdc, Current 0 to 20mA, (Output load from 80 Ω to 550 Ω)

Digital Inputs

The LC1612 has eight digital input channels which can interface with dry contact (voltage-free) inputs such as relay or switch contacts or with 0-5 Volt logic sources.

Digital input characteristics:

Detection of closed switch $<300\Omega$

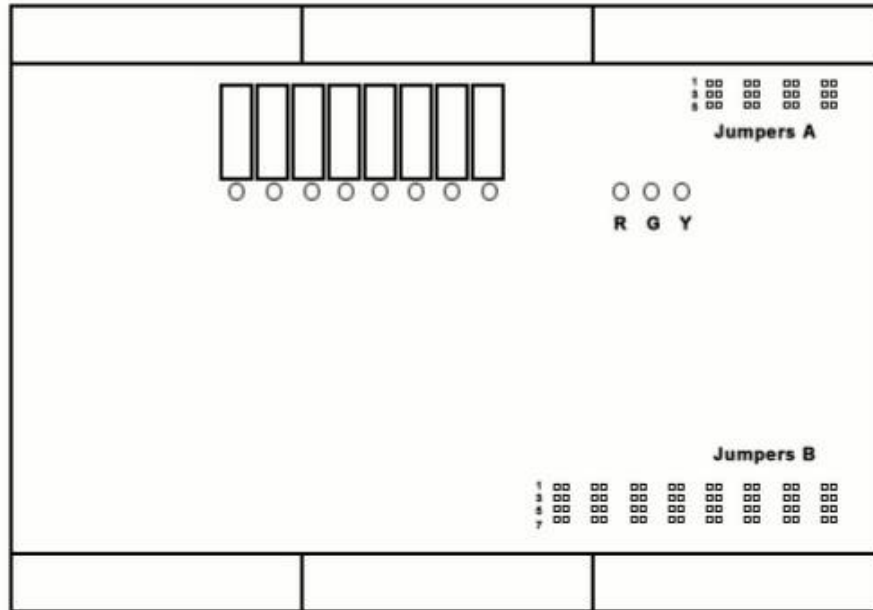
Detection of open switch $>100\Omega$

Relay Outputs

The LC1612 has eight relay output channels.

8 x Relay Contacts, 48VA at 24Vac, pilot duty

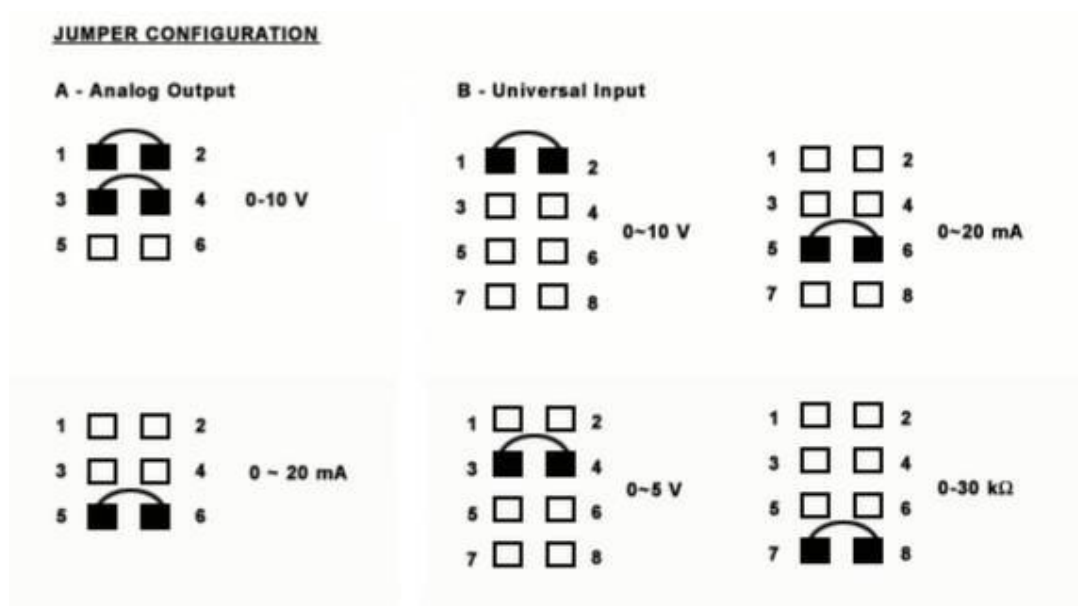
Configuration and Connectors



Jumper Summary

The table below shows a summary of the positions of the jumper shunts and their corresponding channels and configurations.

There are two separate sets of Jumpers, set A and Set B:



Programming and Configuration

There are three of programming/configuration for the LC1612.

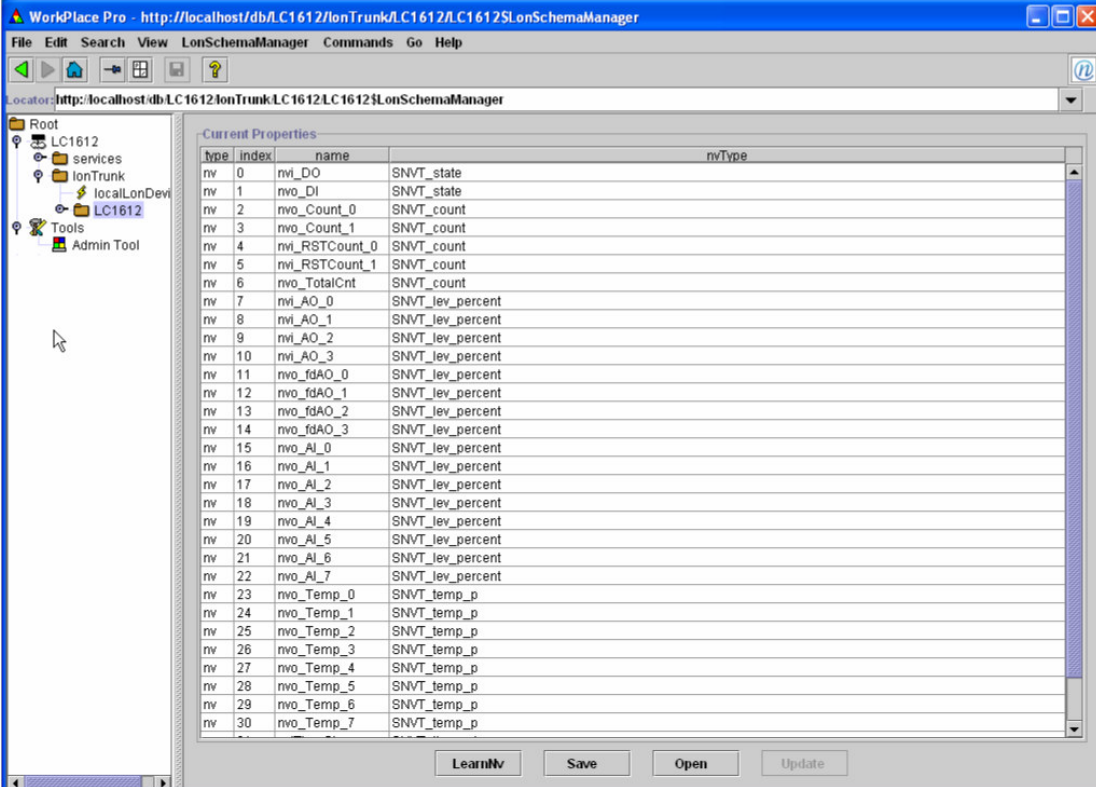
Firstly, the LC1612 needs to have its hardware configured to suit the designer and users requirements, this is done by selecting Jumpers on the I/O circuit board and also with a Niagara string object.

The second part is to configure the PID loops and interlocks utilizing the windows based RS485 AP Configuration tools.

The third part is then to apply Network Management, Web Access and LON Binding for peer to peer binding to other LC1612 or other LonWorks controllers/devices.

Network Management and LON Binding with the Niagara Framework

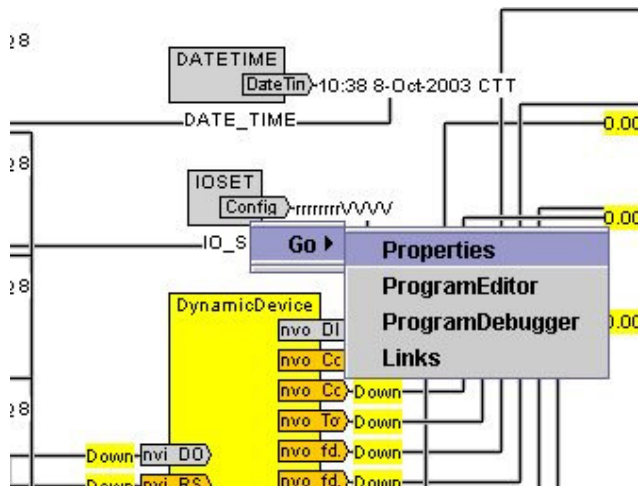
Below is the standard SNVT's Schema for the LC1612



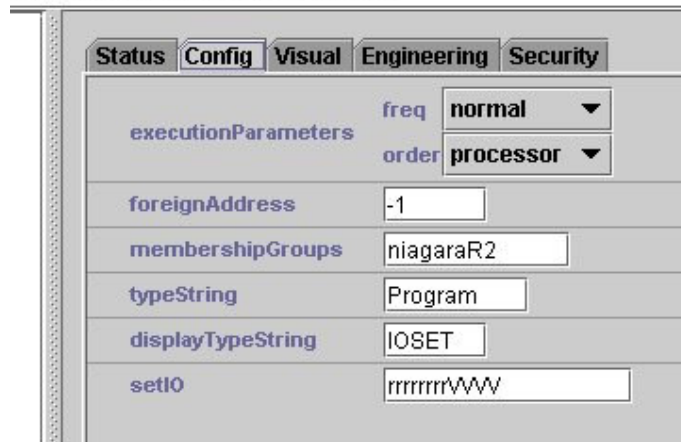
type	index	name	nvType
rv	0	rvi_DO	SNVT_state
rv	1	rvo_DI	SNVT_state
rv	2	rvo_Count_0	SNVT_count
rv	3	rvo_Count_1	SNVT_count
rv	4	rvi_RSTCount_0	SNVT_count
rv	5	rvi_RSTCount_1	SNVT_count
rv	6	rvo_TotalCnt	SNVT_count
rv	7	rvi_AO_0	SNVT_lev_percent
rv	8	rvi_AO_1	SNVT_lev_percent
rv	9	rvi_AO_2	SNVT_lev_percent
rv	10	rvi_AO_3	SNVT_lev_percent
rv	11	rvo_fdAO_0	SNVT_lev_percent
rv	12	rvo_fdAO_1	SNVT_lev_percent
rv	13	rvo_fdAO_2	SNVT_lev_percent
rv	14	rvo_fdAO_3	SNVT_lev_percent
rv	15	rvo_AI_0	SNVT_lev_percent
rv	16	rvo_AI_1	SNVT_lev_percent
rv	17	rvo_AI_2	SNVT_lev_percent
rv	18	rvo_AI_3	SNVT_lev_percent
rv	19	rvo_AI_4	SNVT_lev_percent
rv	20	rvo_AI_5	SNVT_lev_percent
rv	21	rvo_AI_6	SNVT_lev_percent
rv	22	rvo_AI_7	SNVT_lev_percent
rv	23	rvo_Temp_0	SNVT_temp_p
rv	24	rvo_Temp_1	SNVT_temp_p
rv	25	rvo_Temp_2	SNVT_temp_p
rv	26	rvo_Temp_3	SNVT_temp_p
rv	27	rvo_Temp_4	SNVT_temp_p
rv	28	rvo_Temp_5	SNVT_temp_p
rv	29	rvo_Temp_6	SNVT_temp_p
rv	30	rvo_Temp_7	SNVT_temp_p

Above is a screen shot from the Niagara Station - it can be seen from the Schema that LC1612A uses five (5) standard SNVT types.

There types are: SNVT_state, SNVT_count, SNVT_lev_percent, SNVT_temp_p and SNVT_time_stamp.



JM1000/LC1012/IO_SET Properties



FOR INPUTS

R = Resistance or temperature sensor (NVO_TEMP_X)

V = Voltage 0-5 vdc (NVO_AI_X)

V = Voltage 0-10 vdc (NVO_AI_X)

A = 0-20 ma (NVO_AI_X)

FOR OUTPUTS

V = Voltage 0-10 vdc (NVI_AO_X)

A = 0-20 ma (NVI_AO_X)

For example

UI01 – Temperature 10k Thermistor

UI02 - Temperature 10k Thermistor

UI03 – 0-10 vdc

UI04 – 0-20 ma

UI05 - 0-20 ma
UI06 – 0-r vdc
UI07 – 0-10 vdc
UI08 – 0-10 vdc

AO01 – 0-10 vdc
AO02 – 0-10 vdc
AO03 – 0-20 ma
AO04 – 0-10 vdc

The resulting IO string set in the IO configuration object will be: **rrVaavVVVVaV**

AP Windows based RS485 Configuration tool:

After installing the simple AP Windows based configuration software, and linked to the Controller via an RS 485 connection, the following configuration instructions

This document describes briefly various parameters used in the PID loops settings for LC1612A.

There are 4 PID loops each assigned permanently to the Analogue output AO[].

For e.g., the PID loop 1 is assigned to AO[1] and AO[2] for PID Loop 2 respectively.

There are 8 UI[] or UI[1..8] or also known as analogue input.

One of these UI[] can be assigned to one of the 4 PID Loops as the analogue source.

All the parameters listed as the PID Loops settings are configurable using LC1612Config software provided.

Figure 1.0 and 2.0 as shown below are two screen shots of the software for PID Loop configuration.

Whenever the input source is an RTD, “Temperature” must be configured as the Input Type. Also, be sure that an appropriate RTD Table has been assigned to this Input using the Temp. Table Data Entry Screen of this LC1612Configuration tool.

In addition, the physical Input Selection Jumper for the chosen Input on LC1612A PCBA must also be placed at the right header.

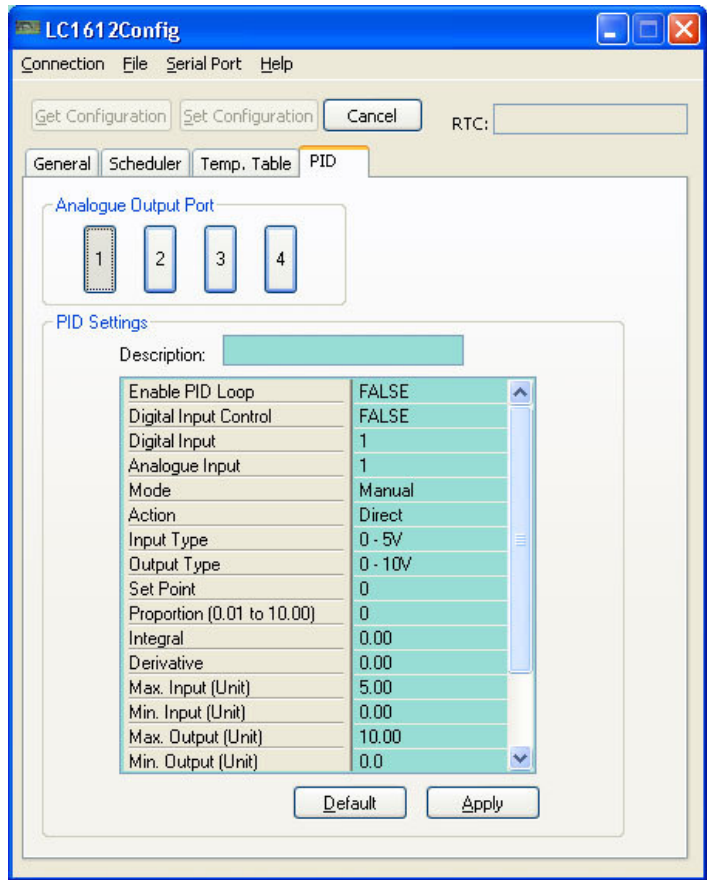


Figure 1.0 PID Configuration Window (a)

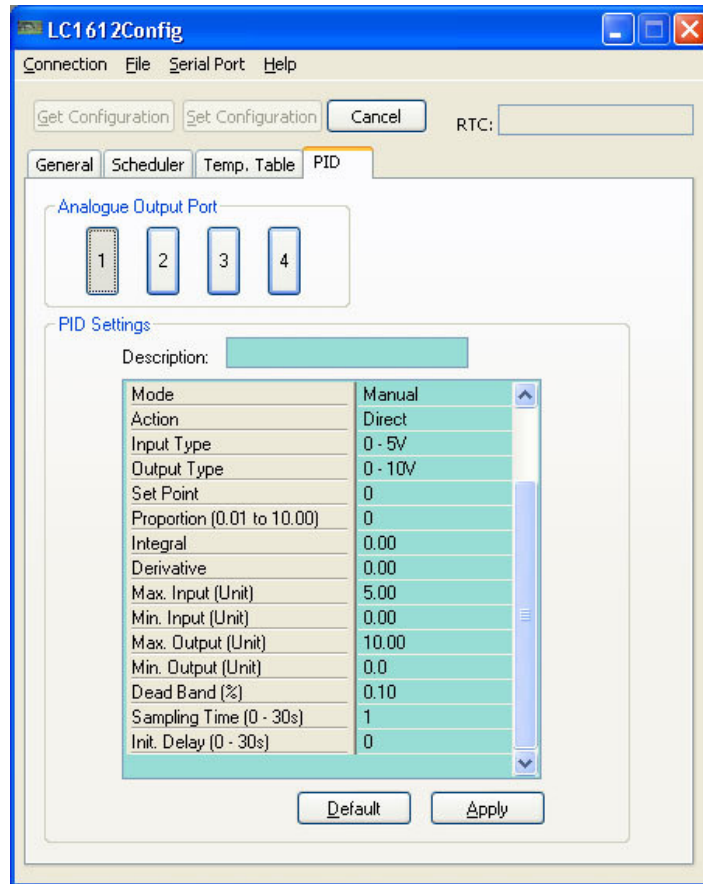


Figure 2.0 PID Configuration Window (b)

IMPORTANT NOTE:

There are 19 items in the in the PID Configuration Window.

The items marked in **BLUE** can only be configured using the LC1612Config tool whereas the items marked in **BLACK** can be configured both by JACE and the LC1612Config tool.

1. *Enable PID Loop*

TRUE: This PID Loop is enabled. In this state, the PID is made available to the system, both in the Standalone Mode or JACE driven.

FALSE: The PID Loop is disabled. Hence, no PID Loop is present. The AO[] is however controllable via Network, i.e. DDC mode.

2. *Digital Input Control*

TRUE: This PID Loop may now be enabled through one of the Digital Input as assigned in (3).

FALSE: This PID Loop does not depend on the Digital Input to start.

3. *Digital Input*

Range: DI[1..8] **Default DI[1].**
Once assigned, this Digital Input is used to start or stop the PID Loop function.

4. *Analogue Input*

Range: UI[1..8] **Default UI[1].**
Once assigned, the Universal Input is the Analogue Input of the PID Loop.

5. *Mode*

Auto: When set as Auto, the PID Loop starts immediately upon power-up. This is valid only if the *Enable PID Loop* is set as TRUE. If the *Digital Input Control* is enabled, then the configured *Digital Input* must also be ON before the PID Loop is started.

Manual: When set as Manual, the PID Loop is not started automatically upon power-up. Instead, it is enabled through the Network. Also, conditions in (1) through (4) must be valid.

6. *Action*

Direct: When set as Direct, the output increases in the same direction as the input variables. See Figure 3.0 below.

Reverse: When set as Reverse, the output is increases in the opposite direction as the input variables.

7. *Input Type*

There are 5 possible Analogue Input types.
The setting depends on the type of sensor used.

NOTE: Please be sure to make the correct Jumper Setting for LC1612A.

0 ~ 5 V [Default]

0 ~ 10V
0 ~ 20mA
0 ~ 10kOhm
Temperature in °C

8. *Output Type*

0 ~ 10V
0 ~ 20mA

9. **Set Point**

Set Point value for Input Types as shown in (7) are to be entered as percentage (%) except for Temperature which must be entered in degree Celsius ((C).

10. Proportion (0.01 to 10.00)

0.01 [Default]

For normal operation for P only or PI configuration, this value is set to 1.

11. Integral

The Integral Gain constant or commonly known as K_i .
The input range is [0 .. 300].
When set to 0, the Integral function of the PID Loop is disabled.

12. *Derivative*

The Derivative Gain constant or commonly known as K_d . The input range is [0 .. 300].
When set to 0, the Derivative function of the PID Loop is disabled.

13. *Max Input (Unit)*

Depends on the **Input Type** as in (7), this value is the maximum value within the allowed range of the selected **Input Type**.

14. *Min Input (Unit)*

Depends on the **Input Type** as in (7), this value is the minimum value is usually 0.

15. *Max Output (Unit)*

Depends on the **Output Type** as in (8), this value is the maximum value within the allowed range of the selected **Output Type**.

16. Min Output (Unit)

Depends on the **Output Type** as in (8), this value is the minimum value is usually 0.

Example for (13) ~ (16) with **Input Type** [Temperature]

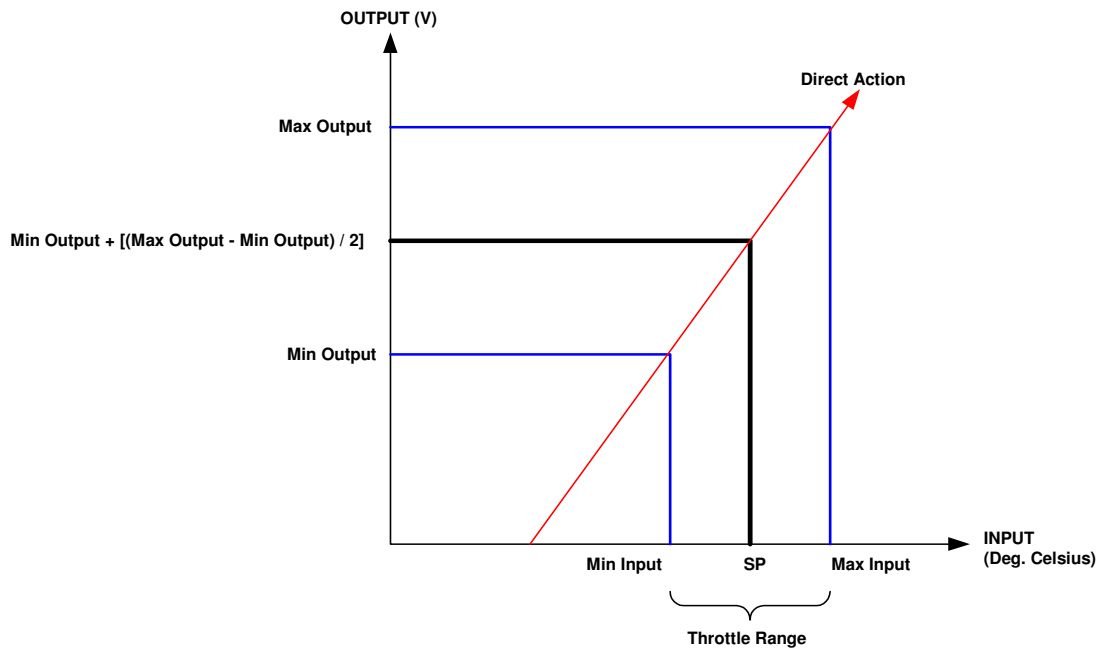


Figure 3.0 Graph

The above graph depicts several parameters in a typical **P** only control loop where direct action is taking place, for example a cooling application whereby the Cool Air Valve is widening when temperature increases.

17. Dead band

It is an error range deemed acceptable.

When the Read Input oscillates around the SP (Set Point) value not exceeding the error range set forth herein, it is considered to fall within the Dead band.

Under this circumstance, the PID Loop is considered to have reached its Set Point value.

No further change in its output variable is needed. More likely, the PID Loop shall cease to apply computed results on its designated output.

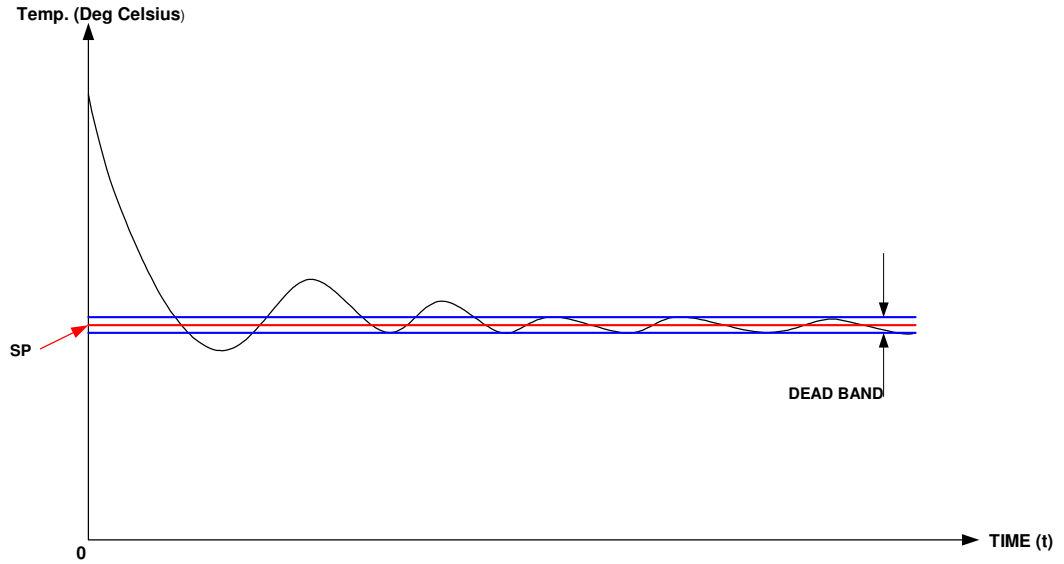


Figure 4.0 Dead band

18. Sampling Time (0 ~ 30s)

This is not the actual sampling time of the Analogue/Universal Input.
 This Sampling Time is referred to as the time intervals in which the PID Loop applies its computed result to its Analogue output.

19. Init. Delay (0 ~ 30s)

This is a time delay imposed on the PID Loop upon Power-up.
 If this value is set to 0, and the Mode is set to Auto, the PID Loop operates immediately upon power-up.

Standard Network Variable Types (SNVTs)

Echelon defines a network variable as a data item that a particular device application program expects to get from other devices on a network (an input network variable) or expects to make available to other devices on a network (an output network variable).

Network variables can be any single data item, or they can be data structures.

Examples of network variables may include temperature, switch positions (binary), or actuator position (Analog).

Applications running in LonWorks devices do not need to know anything about where input network variables come from or where output network variables go.

When the application changes a value, it simply passes the new value to firmware.

Using a process that takes place during installation called binding, the device firmware is configured to know the logical address of the other devices on the network – the device is then able to assemble a message and send it to the devices that need the information.

In a similar fashion, when the device firmware receives an updated value for an input network variable needed by the application, it passes that data to the application.

The binding process creates soft wiring between nodes that represent logical connections between an output network variable (NVO) on one node and an input network variable (NVI) on one or more other nodes.

A typical node on the network may have multiple connections, both inbound and outbound.

When the application running in the sending node writes a new value to an NVO, the value is automatically propagated to the network and received by all nodes that have a bound NVI.

In this way, network variables are continuously updated.

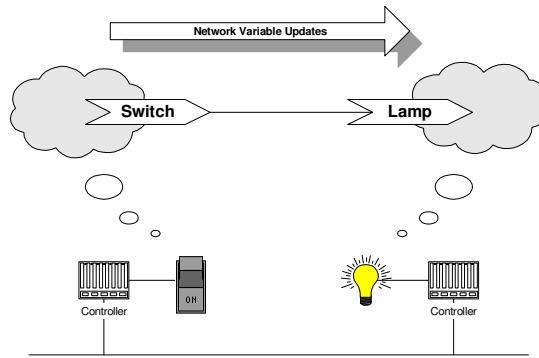
A simple example of soft wiring can be seen in Figure 1.

One node monitors the position of a switch and exposes that value using the output network variable called `switch_on_off`.

Another node controls the on/off state of a light bulb.

The second node has an input network variable called `lamp_on_off`.

A binding between `switch_on_off` and `lamp_on_off`, creates a connection between the two devices that can be used by device firmware that has the same effect as physically connecting a wire between the switch and the lamp.



SNVTs

Network variables and configuration properties of a LonMark profile or object are implemented using formal LonMark data types called Standard Network Variable Types (SNVTs).

SNVTs facilitate interoperability by providing a well-defined interface for communication between nodes made by different manufacturers.

At present, there are at least 145 SNVTs defined (version 10.00), each by name and number, definition (measurement or usage), and range of values.

For example, there are SNVTs for sharing temperature, pressure, status, and mode information.

A node can be installed in a network and logically connect to other nodes via network variables as long as their data types match.

Data Types

Each SNVT is defined as either a scalar value or a structure.

A scalar type represents a single value that is a fixed point number, a floating point numbers, or an enumeration.

A structure is a set of one or more scalar values, embedded structures, arrays, and/or unions.

Naming Convention

The naming convention for a SNVT is:

SNVT_name

The above can be extended to indicate data species or other delimiters. For example:

SNVT_name_f

denotes floating point data.

Engineering Units

Most SNVTs use System Internationale (SI) units (in other words, *metric units*) rather than their English equivalents.

Any necessary conversion from SI units to English units is performed by logic in the node or via a PC interface.

Working with LonWorks Data

Type Versus Format

Every network variable and configuration property has an associated data type and format.

The *type* specifies the units and structure of the data contained within the network variable or configuration property and the *format* specifies how the raw data contained within the network variable or configuration property is translated for display or for use by an application.

The format also specifies how data entered by a user or application is translated to the raw data to be transmitted on the LonWorks network.

The type of a network variable or configuration property is typically fixed for most devices.

However, devices may support network variables and configuration properties with changeable types as described in the *LonMark Application Layer Interoperability Guidelines*.

The format of a network variable or configuration property may include scaling and offset values to convert one type of data to another such as Celsius to Fahrenheit or kilograms to pounds.

As an example, a temperature sensor may report a temperature value with a type of SNVT_temp_f.

The SNVT_temp_f type is defined as a 32-bit signed floating point value representing a Celsius temperature with a range of -273.17 to $1E38$ degrees.

A value of SNVT_temp_f type may be displayed and entered using one of several formats including Celsius, Fahrenheit, or differential Fahrenheit (degrees F with no 32 degree offset from zero).

The format of a network variable or configuration property does not affect how the corresponding value is transmitted on the wire.

Each data type has a default format.

You can change the format at any time, but if you change data type, the format is automatically changed to the default format for that new type

Converting SNVT_lev_percent to Displayed Engineering Units

SNVT_lev_percent has the following characteristics:

- Data is signed long (-32,768 to 32,767 integer) with a resolution of 0.005%.
- The relationship between the raw value and its displayed value is linear with a given resolution. See the table below.

Displayed Value	Raw Value
0	0
.005	1
.010	2
.015	3
...	...
100	20,000

Hint: To calculate the highest raw value, divide the highest displayed value by the resolution (i.e., $100/0.005 = 20,000$).

SNVT_temp_p
SNVT temperature. Measures temperature.

SNVT_temp_p	Signed Long	Temperature	-273.17 ... +327.66 degrees C (resolution of 0.01 degrees C).	105
-------------	-------------	-------------	---	-----

The screenshot shows a window titled "SNVT Master List" with a menu bar (File, Edit, Bookmark, Options, Help) and a toolbar (Help Topics, Back, Print, <>). The main content area displays the following information for "SNVT_temp_p":

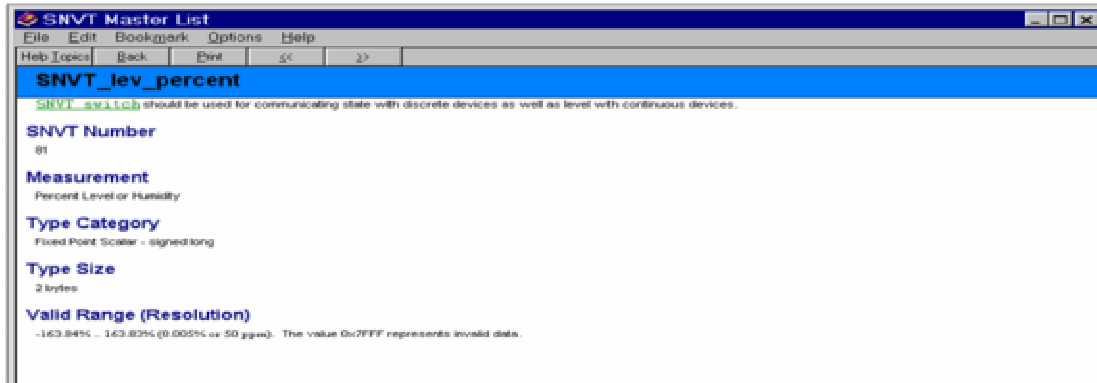
- SNVT Number:** 105
- Measurement:** Temperature
- Type Category:** Fixed Point Scalar - signed long
- Type Size:** 2 bytes
- Valid Range (Resolution):** -273.17 ... +327.66 degrees C (0.01 degrees C). The value 0x7FFF represents invalid data.

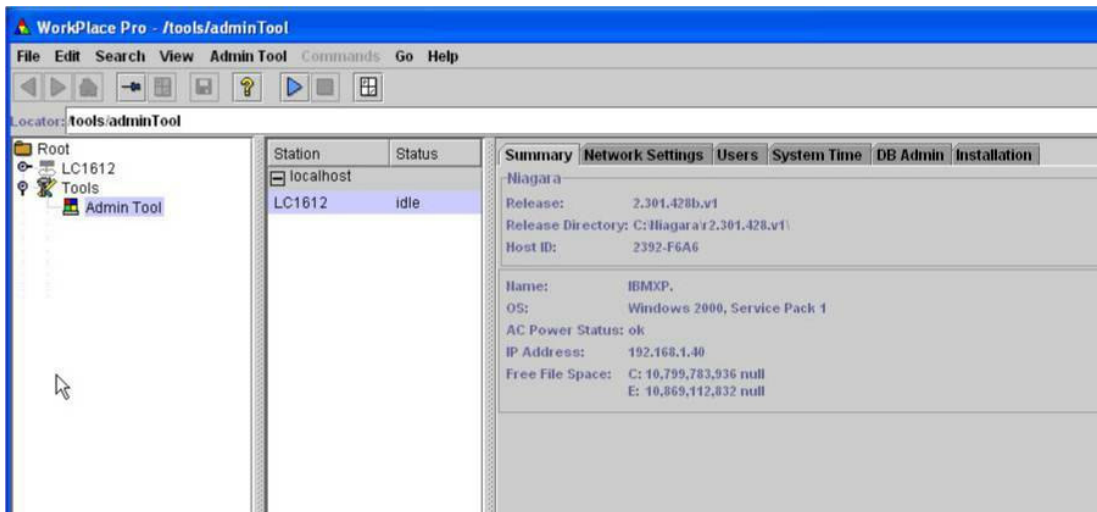
Practice Using SNVT_temp_p

SNVT_lev_percent

SNVT level percent. Measures percent level or humidity.

SNVT_lev_percent	Signed long	Humidity	-163.84% ... 163.83%	81
------------------	-------------	----------	----------------------	----





Data Types

The table below identifies the various data types that might be encountered in a LonWorks node.

Type	Number Category	Range
Char	Character	0 to 255 or ASCII character equivalent
Signed char	Character	-128 to +127 ASCII character equivalent
Unsigned char	Character	0 to 255 or ASCII character equivalent
Signed int	Discrete number	-128 to +127
Unsigned int	Discrete number	0 to 255
Signed long	Discrete number	-32,768 to +32,767
Unsigned long	Discrete number	0 to 65,535
Signed short	Discrete number	-128 to +127
Unsigned short	Discrete number	0 to 255
Float	Continuous numbers (numbers with a fractional portion)	-3.4×10^{38} to 3.4×10^{38}
S32 Type	Signed 32-bit value	-2.1×10^9 to $+2.1 \times 10^9$

How Data is Represented on the Wire

Although messages are transmitted from one node to another via a series of 1's and 0's, network variables are stored in hexadecimal format.

The NV value table displays these values.

The data displayed in this table is raw (hex data).

The communications to and from the shadow objects representing the actual nodes on the network handles conversion from hex into user-friendly data that is displayed in the workspace view – there is code that converts the raw data bytes and we can use this view to verify that that code is acting properly.

SNVT time stamp. A structure that can be used to display and/or change date and time.

SNVT_time_stamp	Structure	Time Stamp	See Structure	84

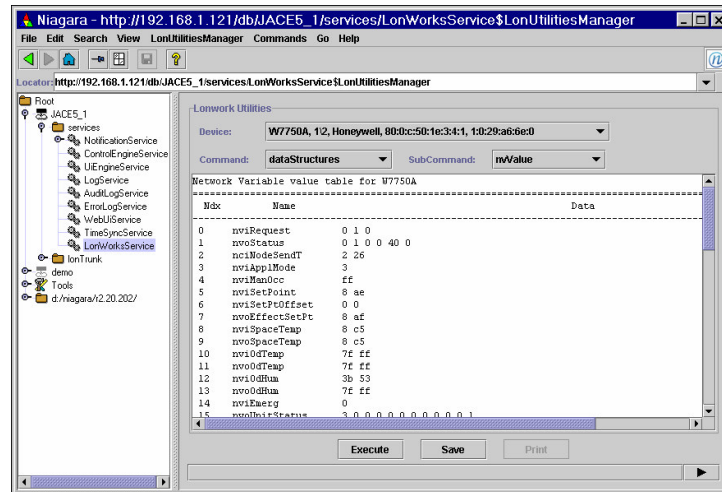
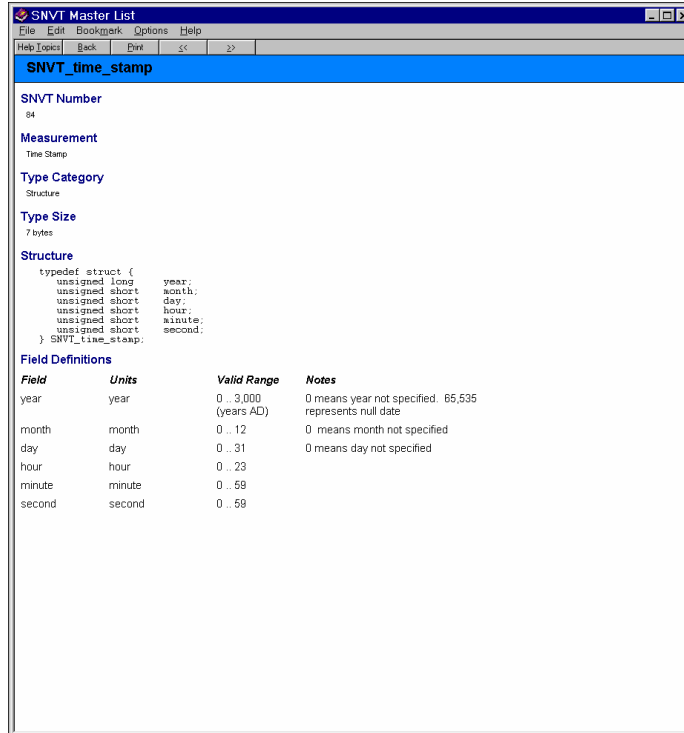


Figure 1. NV value table available in the LON Utilities Manager view.



Bindings:

This is the process that defines logical connections between LonWorks devices. Bindings define the data that devices share with one another.

LonTalk Protocol

A LON, which is a control system field bus, allows intelligent devices, such as actuators and sensors, to share data directly, using a standard protocol.

The protocol, known as LonTalk, is Echelon's open-architecture communications protocol used by all LonWorks-based devices. It allows intelligence and communication capabilities to be embedded into individual control devices that can be connected together through a variety of transmission media including twisted pair data cable, existing power lines, radio frequency (RF), coax, fiber, and others.

The core of this embedded intelligence is a mixture of hardware and firmware on a device known as a Neuron chip, which is an integrated circuit manufactured under license by Cypress, Motorola, and Toshiba.

Almost 10 million Neuron chips have been shipped since 1992 on devices from Echelon and roughly 4,000 other manufacturers worldwide.

The Neuron chip implements the LonTalk protocol and performs a variety of local sensing and control functions.

In addition, each node includes a physical interface or transceiver that couples the node to its transmission medium.

A typical node in a LonWorks control network performs a simple task, such as measuring a temperature, monitoring equipment status, or controlling a motor.

When combined together with complementing nodes, the overall network can perform rather complex control applications, such as conning a manufacturing line or automating a building.

Smart Devices

A LonWorks control network enables a group of electrical devices, or nodes, to be connected together to implement sensing, monitoring, control, and communications capabilities for a variety of applications.

Each node contains a Neuron chip, a power supply, and a communications transceiver.

The communications transceiver couples the Neuron chip to the transmission medium and the Neuron contains the firmware and software required for sending, receiving, and acting upon a variety of control signals.

In addition to these components, each node has a physical ID, a timer, a computation unit, device controllers, I/O ports, and communications and control software.

In effect, each node is a rather sophisticated computer, which is why they are often referred to as intelligent or smart devices.

Network Image

A node's network image describes how the node appears to all other nodes on the network.

It contains the address assignments of the LonWorks node, binding information connecting network variables and message tags between nodes on the network, parameters of the LonTalk protocol that may be set at installation time, and configuration variables of the application program.

Portions of a node's network image can be set by the manufacturer at production and others are downloaded over the network by a network management tool into EEPROM when the node is installed.

Network images contain the following data: Neuron ID, Mfg Data, Node Type, Node Address, Address Table, Location ID, NV Configuration Table, NV Fixed Table, Comm Data, Mode Table, and SI-SNVT/SD.

Neuron ID: This is set at the time the chip is manufactured and it cannot be changed.

It is a 48-bit identification number that is unique to the node.

Some manufacturers label their devices with a bar code label that contains the Neuron ID.

Devices can be queried for their Neuron ID (using a network management tool) or a technician can use a barcode reader to scan the ID from its label.

Network management tools use a domain/Neuron ID addressing format at installation time to assign each node to one or two domains and to assign a subnet and node address.

Wg Data: This is set at the time the chip is manufactured and it cannot be changed. It identifies the model number of the device (for example, Motorola 3120).

Node Type: This is set at application compile time. In LonMark devices, it contains the manufacturer's ID, the node type and sub-type, etc. In non-certified nodes, it contains the Neuron-C program name.

Node Address: One or two structures, each containing a domain, an authentication key, a node number, and a subnet number. There are two node addresses if the node has been assigned to two domains.

Address Table: The address table is used to send and receive messages.

Location ID: This can be set at the time the device is installed and it is intended to correspond to a physical location represented on a floor plan.

Location ID contains a channel number and location string.

NV Configuration Table: This table contains an entry for each network variable including priority bit, direction (input or output), network variable ID (assigned by the binding device), protocol service to use, authentication bit, and address table index.

NV Fixed Table: This table contains an entry for each network variable describing the compile-time attributes of the NV including SI-SNVT/SD data index, network variable length, and network variable address.

Comm Data: The number of priority slots on this channel, this node's priority slot, bit rate, and transceiver parameters.

Mode Table: This is a table of node configuration properties including the number of domains, address table entries, application buffers, network buffers, receive transaction structures, and network variables.

SI-SNVT/SD: Self-Identifying Standard Network Variable Type / Self-Documentation.

This information includes SNVT index, if standard network variable types are used, and optional self-documentation for the node.

FTT-10 Topology

As illustrated in Figure 4-3 and Figure 4-4, there are two broad categories of wiring topologies supported by FTT-10: the Free Topology approach and the Doubly Terminated approach.

The Free Topology approach allows any number of tees, stars, loops, or bus combinations in a wiring segment.

The Doubly Terminated approach, although it supports greater wiring lengths, is significantly more restrictive in terms of wiring topology.

Wiring Segment: The combined lengths of cable connecting the various devices on the LonWorks network in a Free Topology approach is limited to 500 meters and requires a single termination module installed anywhere on the segment.

In a Doubly Terminated approach, all twisted pair wiring is done in a linear bus or daisy-chain fashion that does not support wiring tees of any sort.

Each physical end of the segment must be terminated using a termination module and the maximum bus length is 2700 meters.

A single wiring segment supports up to 64 nodes (FTT- 10 transceivers), but a channel can span multiple segments that employ physical layer repeaters to expand both the number of nodes and the length of the control network.

Depending on the application and expected network performance (in terms of response time), the maximum number of nodes supported by a LonWorks network is 32,385.

The communications rate on an FTT-10 network is 78 Kbps.

Free Topology

Although free topology wiring is very flexible, there are a few restrictions including:

Restrictions: The maximum number of nodes per segment is 64.

The maximum node-to-node distance is 1640 feet (500m) when using Belden 85102 cable or 1312 feet (400m) when using Belden 8471.

The longest cable path between any possible pair of nodes on a segment must not exceed the maximum node-to-node distance.

If two or more paths exist between a pair of nodes (that is, a loop), the longest path should be considered. Note that in a bus topology, the longest node-to-node distance is equal to the total cable length.

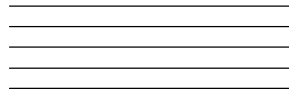
The maximum total cable length is 1640 feet (500m) unless you use TIA Category 5 cable, in which case total wire length cannot exceed 1476 feet (450m).

The total length of all cable in a segment must not exceed the maximum cable length.

One 52.3 ohm (0.25W, 1%) termination resistor is required in each segment.

If you install a repeater and another segment, you need to install a termination resistor in that segment.

The termination resistor can be located anywhere in the segment.



stub length is 9.8 feet (3m). A stub is a piece of cable that is wired between the node and the bus-if the bus is wired directly to the node, there is no stub. If you are wiring to a field terminal strip, be sure to account for any factory wiring between the terminal strip and the device.
Two 105 ohm (0.25W, 1%) termination resistors are required in each segment. One resistor must be located at each end of the bus.

Cable Specifications

The following specifications are derived from the latest information available from Echelon. Consult the Junction Box and Wiring Guideline for Twisted Pair L-Works Networks (005-0023-01) for more detailed information.

Echelon has qualified five cable types for the TP/FT-10 channel for use with the Fine Topology Transceiver. These include:

Cable type	A/B	Diameter
TIA 568A Category 5	24	05mm
Belden 8471 (PVC jacket)		3mm
Belden 85102 (Tefzel jacket)		0.65mm
Level IV	22	0.65mm
JY (st) 2x2x0.8	204	08mm

TIA 568A Category 5: For TP/FT-10 channels in a bus topology, the maximum bus length is 2951 feet (900m), with a maximum stub length of 9.8 feet (3m).

For TP/FT-10 channels in a free topology, the maximum node-to-node distance is 820 feet (250m) and 1476 feet (450m) maximum total wire length.

Belden 8471: For TP/FT-10 channels in a bus topology, the maximum bus length is 8853 feet (2700m), with a maximum stub length of 9.8 feet (3m).

For TP/FT-10 channels in a free topology, the maximum node-to-node distance is 1312 feet (400m) and 1640 feet (500m) maximum total wire length.

Belden 85102: For TP/FT-10 channels in a bus topology, the maximum bus length is 2948 feet (900m), with a maximum stub length of 9.8 feet (3m).

For TP/FT-10 channels in a free topology, the maximum node-to-node distance is 1640 feet (500m) and 1640 feet (500m) maximum total wire length.

Level IV: The Level 4 cable specification used by Echelon is as originally defined by NEMA and differs from the Category 4 specification proposed by EIA/TIA.

For TP/FT-10 channels in a bus topology, the maximum bus length is 4590 feet (1400m), with a maximum stub length of 9.8 feet (3m).

For TP/FT-10 channels in a free topology, the maximum node-to-node distance is 1312 feet (400m) and 1640 feet (500m) maximum total wire length.

JY (st) 2x2x0.8: JY (st) cable is available only in Europe and can be used only with the TP/FT-10 channel.

For TP/FT-10 channels in a bus topology, the maximum bus length is 2948 feet (900m), with a maximum stub length of 9.8 feet (3m).

For TP/FT-10 channels in a free topology, the maximum node-to-node distance is 1050 feet (320m) and 1640 feet (500m) maximum total wire length.

Channels

The term channel is an Echelon term used to describe the physical medium to which a node is attached and the transceiver type it uses to communicate.

Although the LonTalk protocol supports networks with segments using different media (channels), all devices on a given channel must communicate using the same type of transceiver.

Every LonWorks node is physically connected to a channel.

The physical form of a channel depends on the medium: a twisted pair channel is a twisted pair wire; an RF channel is a specific radio frequency; a power line channel is a continuous section of AC power wiring, and so on. Multiple channel types are connected by routers.

Network Management Tools

About Network Management-A high-level view of network management. System Overview-A list of functions needed for a LonWorks network.

Structure of a Network Management Tool-The structure and purpose of a LonWorks network management tool and how it is used to install nodes.

Network Database-Introduces Echelon's LNS operating system and accompanying network database design and some of the tools that are available to support (LNS) network design and system management.

In addition, this section offers discussions of third party database designs and tools.

Application Programming-This section discusses some of the tools that are available to create control applications and download them into programmable controllers on the network.

About Network Management

Network management defines the architecture of the LonWorks network, including channels, routers, nodes, and more.

During the network management process, each device is queried and information about that device is recorded including its name, program ID, Neuron ID, its network variables, and binding information.

All of that information is used to construct what is known as a network database.

Network management tools typically provide graphical, applications-oriented views of the system and allow the user to develop control solutions in that environment.

This protects the user from the specific complexities of the underlying technology as they design, commission, manage, and maintain LonWorks networks.

Network management tools often provide the following:

- Device programming
- Device installation and configuration
- Device monitoring and control
- General purpose support

Device programming: Programmable devices require application software to be downloaded with a field programming tool.

Typically, this software presents control functions as function blocks and allows the user to logically connect them to create system-level capability.

The programming tool then takes the high-level representation of the system and transforms it into application code that is downloaded into the programmable device when it is installed.

Device installation and configuration: Configurable devices are typically delivered in an unconfigured state, which means they have not been assigned a network personality.

Although the devices come from the manufacturer with an application embedded in the Neuron chip, they must be assigned a unique address, they must have addresses of the other devices on the network defined, and the selectors of the network variables they use for communicating NV updates must be identified.

Device monitoring and control: In systems where supervisory functions are required, device monitoring and control software is often used.

This software provides operator interface, timed control, global data passing, data collection and archival, alarm routing, and more.

General purpose support: These tools include network diagnostics, protocol analyzers, simulation tools, and network design and system integration tools.

Binding: Binding tells each node what other nodes to talk to-it tells the node to where it can send data and from where it should expect data.

Bindings are established between network variables (network variable outputs to network variable inputs) for the purpose of passing data.

NVIs can read one or more network variable outputs.

NVOs can write to one or more network variable inputs.

Network management tools enforce type checking for each network variable binding. Standardized data types provide the basis for interoperability.

Network Database

Network Database

As a by-product of installation, network management tools create a network database.

The database contains comprehensive descriptions of all network devices including channels, routers, nodes, network variables, binding information, and more.

Since an image of the network resides in each of the nodes, the network database may not be needed for normal operation-nodes will continue to be able to share data.

But, access to the network database is necessary for network monitoring and control applications, modifications, and maintenance.

There is no single database structure in use today.

Echelon has created an operating system (and accompanying database structure) that they refer to as LNS (LonWorks Network Services), but several other vendors have created different types of database structures.

Since LonWorks technology is an evolving technology, there is no way to tell which structure will become the industry standard, if any.

LNS

Echelon has developed an architecture called LNS that provides a foundation for installation, maintenance, monitoring, and control tools.

Their design revolves around the requirement for a network database to maintain network configuration data for each node on the network.

Any tool or node that needs to change the configuration or read/write a property of a node sends a service request to the device where the network database is maintained.

LNS is a client-server operating system with a single LNS server that supports multiple interoperating client applications.

There is only one LNS network database per LonWorks network and it is always located on the PC that is running the LNS server.

The LNS server can run standalone on a PC attached to a LonWorks network, where multiple remote clients log on to access the shared LNS database.

Or, on larger systems, it may be necessary to split the database into multiple segments running on multiple servers or supervisory controllers.

Remote clients can communicate with the LNS server via the LonTalk protocol or via IT.

To speed commissioning of devices from different manufacturers, LNS defines a plug-in standard.

This standard allows sensor, actuator, and device manufacturers to provide customized applications for their products.

When those products need to be configured, an LNS-based tool presents a configuration screen that the manufacturer has tailored to the device being configured.

Regardless of the LNS tool being used, configuration of the device is consistent.

Note It is important to note that when using the LNS architecture, Echelon requires vendors to pay royalty fees on a per-node basis for each node installed.

Two issues arise with this type of (server-based) design.

Refresh rates are typically slow.

The larger the database and the more widely dispersed it is, the more sluggish response is likely to be.

LNS is limited to 40 transactions per second (20 requests-20 responses).

Server-based HRDBs (hard disk resident database) are notorious for becoming unsynchronized. In other words, when a technician makes a change to the network database (on his laptop), all subsequent modifications must have access to that copy of the database.

Otherwise, the databases become unsynchronized.

There are a number of LNS-based network management tools on the market.

LonMaker for Windows is Echelon's LNS-based integration tool that supports network engineering and installation using Visit as a graphical interface.

LonMake includes:

Network design tool, which allows you to design LonWorks networks off-line.

Network installation tool, which allows you install nodes on site once the network has been engineered (off-line).

Network documentation tool, which uses a Visio drawing to represent the installed network, making it useful for as-built reports.

Network management tool, which allows you to commission, manage, and maintain LonWorks devices.

Echelon's LonMaker network management tool uses the client-server capabilities of LNS to allow multiple LonMaker tools (running on different PCs) to simultaneously access the same LNS server.

Niagara Framework

The Niagara Framework by Tridium, Inc. The Niagara JACE station database runs within a Java virtual machine (JVM) that provides an embedded, controller-level environment in which to configure, manage, and run the nodes and services required by a control system application. In addition to its embedded design, the Niagara Framework provides network management support for both open standards devices (LonWorks and BACnet) and legacy products, as well as a comprehensive applications development environment.

The engineering environment (Java Desktop Environment or WorkPlace Pro) provides:

- Full graphical user interface
- Synchronization of controller databases and database storage and backup Password protected access
- Data collection and enterprise-level information exchange
- Synchronization of global time functions and central scheduling
- Alarm processing and routing Extensive support for all standard energy management functions Customization for non-standard control applications

The core of Tridium's patent-pending technology is the Niagara Framework object model, which allows real world devices to be modeled in a consistent manner through reusable software objects.

Then device integration services provide the glue between the physical device and its software object.

These services allow the objects to communicate with the actual devices using their native protocols and their respective networks.

The software objects use their integration services to read real-time data, send commands to the device, and provide support for reconfiguration and reprogramming.

The Java Desktop Environment (JDE), also known as WotkPlace Pro, is a comprehensive set of engineering tools combined into one, common, easy-to-use applications development environment that supports Niagara Framework solutions.

It supports rapid device integration using open industry standards, plus: Comprehensive network management and application programming tool that integrates LonWorks, BA(Thet, Modbus, third-party legacy products, and more.

Non-LNS based. An object-oriented programming tool that can be used to create supervisory application logic that resides in a supervisory controller (JACE).

Supports plug-ins from a wide variety of application-specific controllers.

The LonWorks service is the hub of LonWorks network management in the Niagara Framework. The Niagara LonWorks service includes:

The LON Device Manager, used to display status information about nodes on the network, and to perform device-level network management operations.

The LON Link Manager, which is used to manage network bindings.

The LON Utilities Manager, which includes a collection of commands and utilities for you to query and manipulate the status of nodes on the network.

LonWorks Service Properties and LonWorks Service Commands used for network configuration.

Device Manager

The Niagara LON Device Manager provides a view to manage LonWorks devices, including commands to find, commission, replace, match, learn, and upload devices.

The view also displays information about each of the nodes on the network (including the Niagara station's hosted node, known as the Local LON Device).

The Device Manager is the default view for the LonWorks service-it displays when you double-click the LonWorks service in the JDE Tree View, or you can select it from the Go menu.