REAL TIME AUTOMATION

LonWorks

A Plan for Product Enhancement

PREPARED BY REAL TIME AUTOMATION

WWW.RTAAUTOMATION.COM

LonWorks

A Plan For Product Enhancement

LonWorks is a Growing Trend in Device Networking

LonWorks™ is a completely open solution for device networking in the Building Automation, Industrial, Public Utility and many other markets. Created by Echelon Corp (www.echelon.com) in 1988 LonWorks is a leading networking solution for Building Automation with a growing presence in Industrial Automation. Estimates for the number of nodes installed worldwide range into the millions. The LonMark Interoperability Association with over 300 member companies reflects the strength LON now has in the automation market.

When used in an industrial environment a LonWorks solution is very different from the open device networks like DeviceNet, Profibus and Modbus typically found on the factory floor. First unlike these popular device busses LonWorks is a completely peer-peer network. Instead of moving data through a "Master" device, any device can exchange data with any other LonWorks device on the network. Second, LonWorks is not tied a single physical communication layer. Where DeviceNet is limited to CAN and Profibus and Modbus are limited to RS485, LonWorks can use twisted pair, Ethernet or even a power line as its communication channel. Finally, network data exchanged on LonWorks is configured by a network configuration tool. This operation called "binding" ties an input of one device to an output of another device independent of the operation or application software in either device.

LonWorks is a standard technology for many of the global standards organizations including ASHRAE, IEEE, ANSI, SEMI and many others. In fact, LonWorks capability is a prerequisite for participation in a growing number of automation projects.. LonWorks is becoming a major network standard in the commercial buildings market with a number of Building Automation Systems suppliers standardizing on LON including Siemens Building Systems and Honeywell. In order to profit in this market it is important to support LON.

What is this technology? What advantages does it have over other networking technologies like Ethernet and others? How can I LonWorks-enable my industrial or building automation device? What are some of the issues and considerations that must be considered when building a LonWorks device? This paper is about getting your device LonWorks enabled and understanding the technology and how it differs from similar device networking technologies.

Important Questions You Must Ask Yourself First

Once your decide that your products must support LonWorks, the big question becomes "how can I get it done and get it done quickly"? Do you have excess Engineering resources to commit to the project? Can your existing opportunities and customers for LonWorks be put off until next year? Is your engineering staff intimately familiar with the specification? Do you thoroughly understand the different LonWorks physical standards and what is best for you and your customers?

If you're having difficulty with any of these issues, then this paper was written for you. It details the technical information you need to know and gives you practical steps for adding LonWorks to your existing product.

What is LonWorks and why do I need it in my product?

Mike Markula, one of the original founders of Apple, founded Echelon in 1983. When Mike Markula mapped out the progression of the mainframe down to the PC he noticed a significant technology gap. At each level, as integration and costs came down, volumes and applications expanded dramatically. But there was no technology that addressed the networking of devices for status and control.

Markula founded Echelon with the vision is to create a technology to allow the networking of everyday devices. Their business model is not to manufacture the silicon but license the networking firmware in silicon and provide the surrounding "glue" that helps companies quickly get to market with a control network. Echelon has successfully
implemented this strategy. They do not produce or sell the “Neuron” integrated circuits, the silicon containing all the LonWorks networking firmware. Instead they focus on the development tools, network management software, transceivers, routers etc.

Echelon—Creators of the LonWorks protocol.

One of the reasons many engineers in the controls industries are unfamiliar with Echelon and with LonWorks is that the company fought some significant challenges bringing the technology to market. With no silicon, all they had to sell was a development tool. Their integrated circuit semiconductor partners were late with the delivery of the first Neuron chips and this significantly delayed initial adoption. Also, Lonworks was not quickly adopted by the controls market much to the surprise of the company's founders. They did not realize how conservative the controls market is in contrast to the computer and telephone industries.

Fortunately for Echleon they have been in the enviable position of receiving patient venture capital on the strength of the founders reputations and have been continually growing and refining the technology and tool suite. Ten years ago Echelon initially licensed Toshiba and Motorola as silicon partners. When Motorola made a decision to shed itself of many different market segments including their LonWorks efforts, Echelon licensed Cypress Semiconductor as a second source manufacture. Echelon now receives royalties on each Neuron and a license fee from each of these companies. Beyond these silicon providers there are literally thousands of companies worldwide building products or using LonWorks technology. As this list grows, so does the acceptance of LonWorks in the various industries it serves.

Today, LonWorks is used widely in the building automation market, the market where it is both best known and a market leader. The technology is also being used in the transportation, medical, industrial, home, utility, and many other markets. The Echelon web site (www.echelon.com) lists some of these other applications.

What is LonWorks Technology?

LonWorks technology encompasses the Neuron chips from the multiple vendors, the LonTalk protocol, the various physical media which connects devices, the connectivity devices such as routers and PC interface cards, network management tools, and all the various products built around the platform.

Why is The “Neuron Chip” So Special?

The Neuron chip is the heart of almost all LonWorks based devices. The Neuron chip is a complete system on a chip. The Neurons contains the entire LonTalk protocol stack and is comprised of multiple CPUs, memory, I/O, communications port, firmware, and operating system. There are two basic types of Neuron chips, the 3120 and the 3150. Functionally they are identical with the exception of memory configuration and packaging. If memory is not an issue, an application that runs in a 3120 will run in a 3150. The opposite may not be true if the application is too big for the memory size of the 3120. Basically, the 3120 is a selfcontained unit, all of its memory is onboard and it cannot be expanded. The silicon providers make several different flavors of 3120s with differing amounts of memory to meet various design constraints. The 3150 has an external address and data bus to allow for expanded memory. The 3150 is typically used in larger applications. The key element to any Neuron chip is that it must implement the complete LonTalk protocol stack.

Why Are There Three CPUs In A Neuron?

The Neuron chip contains three CPU's. One CPU is the Media Access Processor (MAC). The MAC processor is responsible for the sending and receiving of messages on the network. It also checks to verify if the CRC and the message destination is correct. The Network Processor is responsible for the middle layers of the protocol. Doing things like packet routing, destination addressing, end-to-end acknowledgements, retries, duplicate message detection, etc.

The third processor is the Application processor. It is the processor that runs the users custom application. This is an 8-bit processor and does not have any hardware floating point. So it will not handle every application but if you can implement your application in a typical 8-bit micro then you shouldn't have any problem using the Neuron chip. For those more powerful applications you can either port the protocol to another high-end processor or turn the Neuron into a communications co

-processor. The Neuron can connect to another processor and let that high-end processor run the application while the Neuron handles the communications.
There are several reasons for using two processors including low cost of processors, performance, and decoupling of network traffic from application processing. With separate processors, if the application gets very busy, the network processor can still send and receive messages. If the network gets very busy, the application can still process the local control algorithms.

Within the Neurons there are basically 3 types of memory available - RAM, ROM and EEPROM. In the case of the 3120, all of its memory is onboard and cannot be changed or expanded. The 3120 has onboard ROM. This is not memory space available to the user.

This space is already pre-programmed at the factory with the LonTalk protocol, the operating system, I/O libraries etc. In addition to the ROM the 3120 also has onboard RAM and EEPROM. How much of each depends on which derivative of the 3120 you choose. The 3150 has no onboard ROM but instead relies on the external memory and memory to contain the protocol, libraries and other system software. The 3150 does have RAM and EEPROM onboard. Again, the amount of memory depends on which version of the 3150 you choose. With the external memory bus, the 3150 can have any mix of memory types you want. One restriction is that the first 16K worth of space is dedicated to the protocol, operating system etc. The remaining 48K worth of space is available for application code.

How Does The Neuron Handle Memory?

Within the Neuron, there are 4 memory images. Those images are the System image, the Network image, the Comm image and the Application image.

The System image contains the protocol, operating system, I/O libraries etc. This image is already in the 3120 on board ROM. In the case of the 3150, the user must make the system image is stored in external memory within the first 16K of memory address space.

The Comm image contains the communication parameters such as media type, speed, etc. The Neurons defaults to differential twisted pair at the highest data rate available with the available clock.

The Network image contains the devices logical address and binding information including the destination node to talk to, the message types to use, etc. Most typically this image is modified at installation time by a network management tool.

The Application image contains the users custom control algorithms. If you are buying a LonWorks device, it is likely that the application image is already developed and stored with the device.

By breaking it up into 4 different images, it gives the developer and the user a tremendous amount of flexibility to determine when and where each image gets loaded. At a minimum, for Neurons to communicate they must have the System and Comm images. The Network image and the App images can be downloaded over the LonWorks network or loaded at the factory. Another interesting benefit to this structure is that nowhere in the application do you specify what media you are using or what devices type you will be that target of communications. This provides great flexibility. That is the function of the Network image. An integrator can buy a device, containing the System, Application, and Communication images. The integrator can then modify the Network image to identify the physical communication channels for each particular application.

How Does The Neuron Connect To a Network?

Each Neuron has a 5-pin transceiver interface. The interface is configurable and the parameters are stored in the Comm image. There are three different configurations for the communications port:

Single ended Manchester encoded. This is typically used for interfacing to transceivers such as RS485 or Free Topology.

Differential Manchester encoded. This provides a 2-wire polarity independent interface typically found in standard twisted pair wired transceivers. This is the default configuration for the Neuron if nothing else is selected.

Special Purpose Mode. This interface provides a two way port to an intelligent transceiver. Echelon uses this interface with its power line transceivers.

It should be noted that the Neuron chip does have a complete transceiver onboard and is capable of communicating over twisted pair with no other additional components than some resistors. This is not recommended since the communications wires are connected directly to the silicon and have no isolation, but in certain limited cases, it can be appropriate to create a small network with no external transceiver. Another note is the introduction by Echelon of a combined Neuron and Free Topology transceiver chip. As the technology marches forward, you can expect to see more integration providing the designers with more implementation options.

How Much I/O Does The Neuron Support?

Today's Neurons have 11 I/O pins to interface to the outside world. There is nothing magical about that number. Future Neurons could contain 1 or 100 I/O depending on the silicon providers and the market demand. Or a high end processor with the LonTalk protocol can have any configuration of I/O desirable by the designer.

Along with the I/O hardware, the Neurons contain I/O library firmware. With the combination of firmware and hardware the I/O can be configured to provide over 34 different I/O models. For example, instead of a developer having to write code to implement PWM I/O, in the application code for the Neuron you simply declare one of your I/O pins to be a PWM output. It makes development of I/O interfaces much easier for the developer.

What Is The Neuron ID?

Neuron chips each have a unique 48-bit Neuron ID; this is analogous to the MAC ID in Ethernet. Echelon manages those numbers to insure uniqueness. Communications is initiated using the Neuron ID and then logical address assignments are made for the application.

What is LonTalk?

The LonTalk protocol is the core technology providing an implemented, debugged, maintained, and proven protocol. It implements full functionality of the 7 layer OSI protocol standard. This provides a great deal of flexibility and expandability. Small networks are not required to use all the services but as that network grows, the features are there for expansion without having to upgrade software or firmware. There is nothing in the LonTalk protocol that limits the communication speed. When the first Neurons were released, they supported communications speeds as high as 1.25Mbps. The protocol supports a variety of speeds up to the maximum. This is important in order to support a variety of media. Not all media can handle 1.25Mbps. Powerline, as an example, cannot handle the higher speeds so the Neuron has to be able to handle lower speeds as well.

What is the Protocol for Accessing the LonWorks Network?

In any network, there must be a mechanism for gaining access to the bus so that your device can send a message. Echelon didn't want to re-invent the wheel but they also needed an access method that would scale well. Many of the applications Echelon targeted would involve networks of thousands or tens of thousands of devices. One access method that does scale is CSMA, which many of you may know is the access method used in Ethernet. The problem with CSMA and Ethernet is that it does not make for a good control system. The way CSMA works, is that when 2 or more devices want to communicate on the bus, they listen to the network and wait for the line to go quiet. When the line goes quiet, then the devices trying to communicate immediately start broadcasting their messages. If there are 2 or more trying at the same time a collision occurs. In the Ethernet world, the devices detect that collision and immediately stop transmission and back off. Each device then waits a random amount of time and then retries. Hopefully each device has waited a different amount of time and then the collision is resolved. The problem is, within the Ethernet protocol, there are a fixed number of randomization slots. That means, as the number of devices trying to communicate increases the probability of two or more devices picking the same randomization slot also increases. Looking at the performance curve of a typical Ethernet network, you will see a hockey stick curve. The number of collisions increases dramatically as traffic increases until the network is 100% collisions. This occurs at approximately 40% of bandwidth utilization; essentially shutting down the network.

In the office Ethernet world, having the network shut down momentarily isn't a big deal. So your print out takes a few seconds longer to print out, big deal! What if that were a brake message on a train system? Would you want your network designed such that it might not deliver the message at all? Fortunately there has been much research into CSMA protocols since the invention of Ethernet. You don't see most of those changes because of legacy issues you cannot simply go back and change everyone's Ethernet systems. Echelon relied on the research and applied some innovative techniques of their own to come up with a version of CSMA that was better suited for control applications. That version was p-Persistent CSMA with Collision Avoidance and optional Collision Detection.

While Ethernet has a fixed number of randomization slots, the LonTalk protocol utilizes information in the protocol and message headers to predict message traffic and dynamically increase or decrease the number of randomization slots. So as traffic increases, the number of slots increases. As traffic decreases, the number of slots decreases. The net result is that when traffic increases, the probability of two or more devices picking the same randomization slot remains low. Looking at the performance curve, the LonTalk protocol provides a linear response to offered traffic. Unlike the hockey stick curve of Ethernet, which has 100% collisions at 40% bandwidth utilization, the LonTalk protocol statistically has less than a 4% collision rate even at 100% bandwidth utilization. This means that collisions will increase as traffic increases but only marginally and that the network will continue to operate and deliver messages.

While this performance is remarkable, you might say that this is great but there is a 4% chance of a train brake message failing to arrive. To solve this problem the LonTalk protocol allows for priority message slots. You can define slots in which no other device can use to randomize. Those priority slots can be reserved for those key messages (train brake message) that must have access no matter how busy the network is. Over all, the LonTalk protocol is one of the most robust protocols available. It is not going to solve everyone's problems and it is certainly not appropriate for every application. But it can meet a wide a range of needs within the control industry.

While this performance is remarkable, you might say that this is great but there is a 4% chance of a train brake message failing to arrive. To solve this problem the LonTalk protocol allows for priority message slots. You can define slots in which no other device can use to randomize. Those priority slots can be reserved for those key messages (train brake message) that must have access no matter how busy the network is. Over all, the LonTalk protocol is one of the most robust protocols available. It is not going to solve everyone's problems and it is certainly not appropriate for every application.
But it can meet a wide a range of needs within the control industry.

How Does LonWorks Address Devices On The Network?

There are a variety of addressing mechanisms within the LonTalk protocol. One method is to use the 48-bit Neuron ID and address the node directly. While you can communicate in this manner, it is not advisable on a day-to-day basis. The Neuron ID physical address is unique to every Neuron chip. Let's assume we have a small network of devices and use the Neuron ID as the address. If one device breaks and it is replaced, then the Neuron ID of that device changes with the new Neuron. For the other devices to recognize it, you would have to change the programming of every other node on the network. This is hardly an efficient method for managing a dynamic network. More appropriately, you would like to give the Neuron a logical address. That way if one device breaks, you simply give the replacement device the same logical address as the old one. But how do you give a device a logical address if you can't address it? The Neuron ID is typically used as a bootstrap mechanism.

It gives you the means to address a device so that you can assign it a logical address. Within the Neuron that logical address is comprised of three parts - the Domain ID, Subnet ID and the Node ID. The Domain ID is the address of the overall logical network. Do you need a Domain ID? Perhaps and perhaps not, it depends on your network needs. The Domain ID is programmable in length from zero bytes to six bytes in length. Typically a Domain ID is used if you have the potential of sharing the media with other networks. For example, if I am using power line communications for home automation, it is likely that my messages could be seen on my neighbors power lines, especially if the neighbor is on the same transformer. In order to keep my control messages from interfering with my neighbor's devices, I would assign different Domain Ids. If you have a small closed network that is "never" going to share the media with another network, then you can set the domain length to zero and save some communication overhead.

Every Neuron can be assigned two logical addresses. With that, a Neuron can belong to two different logical networks. An example use of that would be to assign a meter in the house to a Utility domain and to a home domain. Or devices in a building could be assigned to a building supervisor domain and to a floor domain. The three part addressing provides flexibility in addressing schemes. You can address every device in the network by using the Domain ID. You can talk to a specific device by using its Subnet/Node ID or you can talk to a collection of devices in a subnet by using the Subnet ID.

After giving devices Domain/Subnet/Node IDs, I can then assign Group Ids. A group is a collection of devices that can be addressed as a unit. This is subtly different from using Subnet addressing. A collection of devices in a subnet cannot span a router. In other words, the devices within a subnet must be on the same physical channel. Members of a group can be anywhere within the logical Domain and are not restricted to residing on the same physical channel. Every Neuron can belong to up to 15 different groups. This allows for different device functionality based on the addressing mechanism. For example, a light can belong to group Conference Room and to group Security Lighting and to group Emergency Lighting. The behavior of the light will depend on which group address is used. One more addressing mechanism is the Explicit Message. With this method you can explicitly address any device or group using the logical addresses assigned. As you can see there is tremendous flexibility in addressing mechanisms. Where and how each is used is beyond the scope of this introduction, but suffice it to say that there are strengths and weaknesses to each. It is important to understand your network architecture so that you can pick the addressing mechanism most appropriate for your application.

What Are Network Variables?

A major goal of the LonTalk protocol is to give developers, from the same or different companies, the ability to design products that will be able to interact with one another. The LonTalk protocol provides a common applications framework that ensures interoperability using powerful concepts called network variables and Standard Network Variable Types (SNVTs). Functional device models have been developed by the LONMARK Interoperability Association to assure plug and play compatibility. Network Variables

Communication between nodes on a network takes place using the network variables that are defined in each node. The product developer defines the network variables when the application program is created as part of the Application layer of the protocol. Network variables are shared by multiple nodes. Some nodes may send a network variable while others may receive. By only allowing links between inputs and outputs of the same type, network variables enforce an object-oriented approach to product development. This greatly simplifies the process of developing and managing distributed systems. Whenever a node program writes a new value into one of its output variables, the new value is propagated across the network to all nodes with input network variables connected to that output network variable. This action is handled by the protocol within the Neuron Chip.

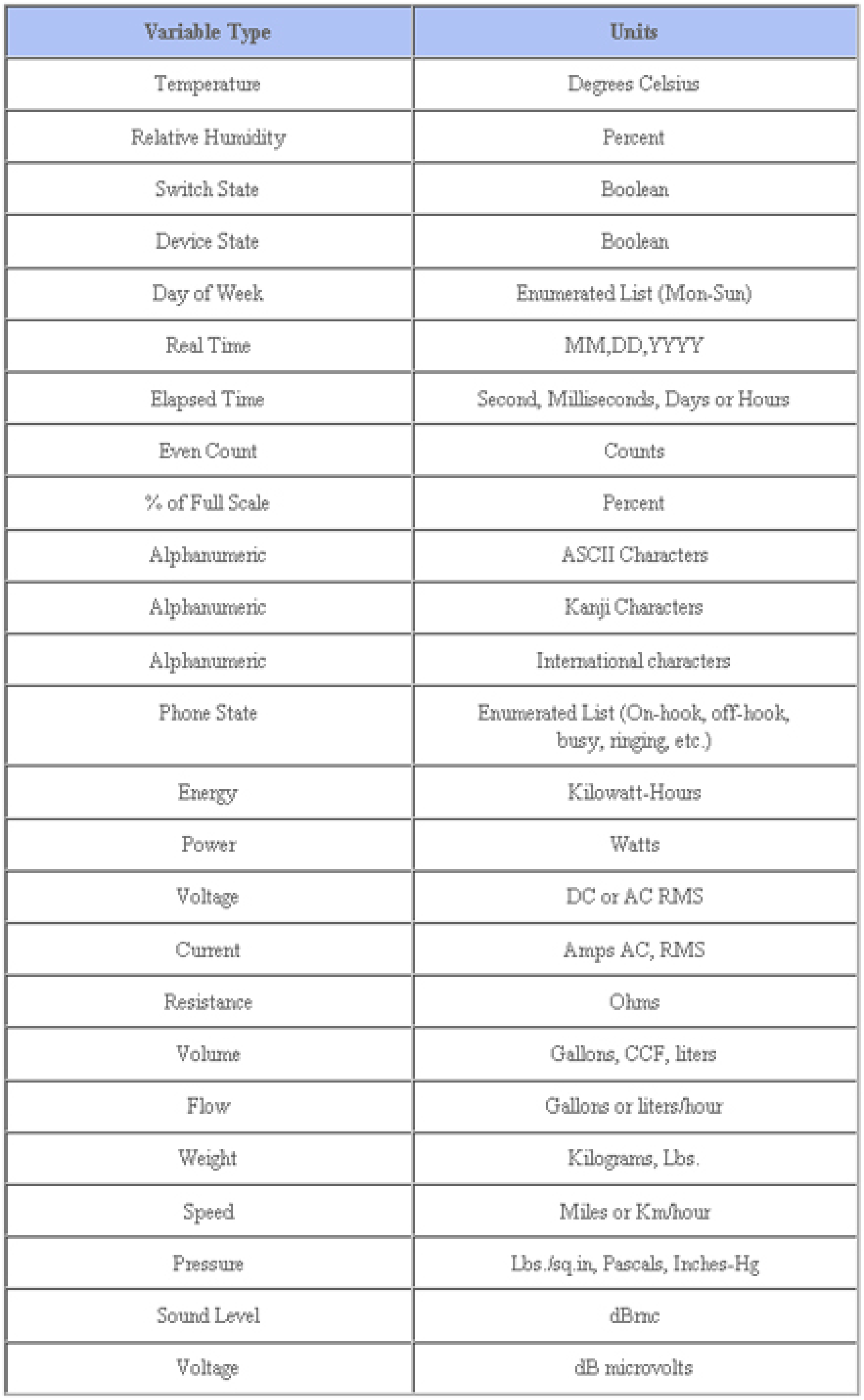
Just What are SNVTs?

The use of Standard Network Variable Types (SNVTs, pronounced "snivets") contributes to the interoperability of LONWORKS products from different manufacturers. Echelon maintains a growing list of over 100 SNVTs for nearly all physical measurement types including the type of variable such as integer or floating point. For example, a SNVT for continuous level is defined as SNVT_lev_contin.

If all manufacturers use this variable type in their application when a network variable for continuous level is defined, any device reading a continuous level can communicate with other devices on the network that may be using the variable as a sensor output to initiate an actuator. As long as a network input variable and a network output variable are defined with the same SNVT when the developer creates the applications, they can be connected together on the network through a process called binding.

When you install a node, you specify which network variables are to be connected between nodes. This is easily done by highlighting the output network variable on one node and the input network variable on the node or nodes to be connected. Only network variables of the same SNVT type can be bound together. In other words, a temperature type could not be bound to a pressure type.

The following are examples of SNVTs. A complete list of SNVTs is available from Echelon.



Variable Type	Units
Temperature	Degrees Celsius
Relative Humidity	Percent
Switch State	Boolean
Device State	Boolean
Day of Week	Enumerated List (Mon-Sun)
Real Time	MM,DD,YYYY
Elapsed Time	Second, Milliseconds, Days or Hours
Even Count	Counts
% of Full Scale	Percent
Alphanumeric	ASCII Characters
Alphanumeric	Kanji Characters
Alphanumeric	International characters
Phone State	Enumerated List (On-hook, off-hook, busy, ringing, etc.)
Energy	Kilowatt-Hours
Power	Watts
Voltage	DC or AC RMS
Current	Amps AC, RMS
Resistance	Ohms
Volume	Gallons, CCF, liters
Flow	Gallons or liters/hour
Weight	Kilograms, Lbs.
Speed	Miles or Km/hour
Pressure	Lbs./sq.in, Pascals, Inches-Hg
Sound Level	dBrnc
Voltage	dB microvolts

What Message Types Are Available?

Once addresses have been assigned, you can now send and receive messages or Network variables. The LonTalk protocol provides a number of options for the delivery of your messages.

One option is to send your message Un-Acknowledged (UnAcked). This is also known as the "send and pray protocol". Using UnAcked service, the message is sent on the network as required with no return acknowledgement that it was received. This can be an appropriate method for recurring, low importance messages. An example would be a device that reports the outside temperature every 5 minutes. If I miss one message, I will likely get it a few minutes later.

Another message option is Un-Acknowledged Repeat (UnAckedR). Similar to UnAcked, this service sends out the same message a fixed number of times. The number of the repeats is programmable. This mechanism is useful when you want to increase the probability that a group of devices gets a message without dealing with feedback from every device. The example here is in the building automation world. If I have a building that is LonWorks enabled and there is a fire detected, then the first priority of the control system is to sound the alarm, turn on the sprinklers and other fire related controls. As a secondary task, you would like to turn on all the lights in the building but you certainly don't want feedback coming back from every light. That would create enormous traffic on the network and impact the performance for the important messages. The repeat service increases the probability that every light gets the message without burdening the network.

The default message delivery service is Acknowledged Service with Automatic Retries (Acked). When a message goes out as Acked service, then the receiving device sends an acknowledgement (Ack) back to indicate that the message was received correctly. Acked messages can be sent to individual devices or to groups and the protocol keeps track of the Acks from the members in the group. The acknowledgements are true end-to-end acknowledgements. The Acks come from the receiving nodes and not from the routers. If an Ack is not received in a given time frame then the protocol automatically resends the message. The default is 3 retries but this is programmable. If after the retries, an Ack has not been received, then a flag is set indicating failure of message delivery. Your application can then decide how to react to that failure.

A fourth message delivery service is Request/Response. This an application to application message with the response returning data with it. This method is used in polling a device for data.

One interesting point to these message services is that they are not coded as part of your application but stored separately in the Network Image. This means at installation time you can determine which delivery method is most appropriate. Or at some later time you can come back and change the delivery method without changing your application code.

What About LonWorks Security?

The LonTalk protocol does not implement data encryption but it does implement Sender Authentication. While mathematically similar, they provide different protection. Data encryption is commonly used to hide data. Your checking account balance is an example of encrypted data. Sender Authentication is used for verifying that the sender of a message is an authorized sender. Let’s examine the home utility meter as an example. The utility would like to be able to update the utility pricing rates within the meter by sending a message to the meter with the new pricing schedules. The meter receives the new data, but how does it really know that the new schedules came from an authorized source (in this case the utility)?

The utility doesn't want to hide the data, because other devices within the home can make use of the pricing schedules to more efficiently run the house. But they do not want just anyone being able to change the pricing rates so they use Sender Authentication The way it works is that the utility places a unique 48-bit key in the meter - an authentication key. This is not the same as the 48-bit Neuron ID. The authentication key cannot be read out of the Neuron and cannot be changed without already having the key.

Once the key is in place and the utility sends the update-pricing message, the meter now sees an Authenticated message. The meter responds to the Authenticated message by sending back a challenge with a 64 bit random number. The sending device receives the random number and performs an authentication transform on it and returns the transform back to the meter. The meter performs the same transform and if it matches the response from the utility, then the "update pricing" message will be processed.

At no time is the authentication key transmitted in this exchange. A spy on the network would see the pricing schedule, followed by a 64 bit random number, followed by a transform. Mathematically, given the random number and the transform, it is virtually impossible to determine the key.

While this authentication mechanism is very secure, it should be used judiciously since it does double the amount of traffic for each authenticated message.

How Are LonWorks Messages Routed?

The LonTalk protocol includes mechanisms for routing messages. Routers can be configured as Learning Routers, Configured Routers, Repeaters or Bridges. Routers are used for several reasons. First is physical. Any media you communicate on has physical limitations. If it is wired media it might have distance, number of loads, or drive limitations. In wireless media it will have distance and interference limitations. Routers are useful for extending the network beyond those limitations, making the network go farther or adding additional devices to the network.

Another reason to use Routers is to cross over to a different media running at different communication speeds. It might be appropriate to build a network that has a wired segment, a power line segment and a wireless segment. Routers will seamlessly connect these various media and buffer messages to allow for the different speeds.

A third reason to use Routers is for traffic segmentation. A router is an intelligent device and it only passes messages from one side to the other if the message destination is on the other side of the router. This fact is useful when architecting your system. If you have a group of devices that communicate with each other on a regular basis, you can put those devices on the one channel and separate them from the rest of the network through a router. Messages among the devices on that channel will stay on that channel and not impact the performance of the rest of the system. But, any message with a destination on the other side of the router will be seamlessly be delivered. Putting some forethought into your network segmentation allows for maximum bandwidth utilization.

What Are The LonWorks Message Types?

There are two basic message types in the LonTalk protocol, network variables and explicit messages. Unlike message delivery services, the explicit messages must be composed ahead of time and implemented in your application code. Should you decide to make any changes to the number or types of messages, and then your application code must be modified and re-compiled. Of the two types, Network variables are the most widely used, most interoperable, and the easiest to use. Most of the discussion will be centered on Network Variables.

How are Network Variables Used?

At the beginning of your application code, you typically declare your variables. In Neuron C, the programming language of the Neuron chip, you have an option of declaring a variable as a Network Variable (NV). It can be declared as a Network Input or Network Output variable. After declaring the variable as “NV”, you declare its type, which can be any valid C data type. Then like any C variable, you give the NV variable a tag or reference name.

If a variable is declared as an input NV, then you have identified that variable to the operating system within the Neuron chip. It will expect network traffic for that variable with data of the type you specified. If the variable is declared as an output NV, the Neuron operating system watches that variable and if it is modified, it automatically propagates that variable across the network. As a programmer, you do nothing more than change the variable as part of your application code and it is automatically transmitted to consumers of that variable. There is no packet construction, frame formatting, sending and waiting for Acks etc. All communication functions are handled by the underlying protocol.

Network variables act like any other C variable. You can initialize them, evaluate them or modify them. Each NV can contain self-documentation strings, which are useful during the integration of the device.

What does “Binding” Mean?

Nowhere in your application code do you specify the destination for an output NV or the type of service used to deliver it. That is typically done during the commissioning phase of a network. At commissioning, the destination is determined through a process called "binding". Output NVs are bound to Input NVs of the same data type. One output NV can be sent to multiple input NVs or multiple Output NVs can be sent to one input NV.

One of the strengths of the technology is that this procedure is done at commissioning over the network and therefore can be changed at a later date. If you determine that additional devices need information from a specific sensor, then add those bindings to the network. There is no reprogramming or development needed.

What are LonTalk Explicit Messages?

Explicit messages require the application to build up the packet, specify the destination address and delivery mechanism etc. This technique requires significantly more programming than using NVs and it is inherently non-interoperable. So why would you use this method? Well first NVs are limited to 31 bytes of data, maximum. You might have an application that requires a structure of greater length. Perhaps you have a deliberate reason for not being interoperable. You might create your own message types this way. Perhaps the most common reason to use this method is to get around binding limitations. For example there are only 15 address table entries in the Neuron. That limits how many destination addresses you can bind to. If you have a lot of messages that must be sent to a wide range of addresses then explicit messaging might benefit you The key is to examine your needs carefully and use each message type where it is appropriate.

What Media is Supported?

The LonTalk protocol was designed to operate over any type of media. The Neuron chip provides a flexible hardware interface to facilitate the connection to a wide variety of transceiver types. Today you can get transceivers that allow you to communicate over twisted pair, free topology twisted pair (FTT), radio frequency (RF), power line communications (PLC), coaxial cable etc. If you can design a transceiver for it, then you can use the LonTalk protocol over it. There is a company in New Zealand that developed a LonWorks transceiver that communicates over an electric fence!

Of these transceiver types, the twisted pair, the free topology and the power line are available from Echelon. Other media types are available from other third party companies. You can also develop your own if you are so inclined. The decision on which media type is used is usually based on a number of criteria such as industry standards, bandwidth, distance, number of devices etc. As mentioned earlier, you don't have to change your application code to change media type. This gives the designer an opportunity to build different models with various transceiver types and re-use the application code without having to re-compile.

What has caused this enthusiasm for LonWorks?

It’s no secret that proprietary fiefdoms have dominated the automation business ever since people started networking PLC’s. Hundreds of protocols have come and gone, and the dominant ones (Remote I/O, DataHighway, Genius I/O, Modbus Plus and others) have protected the turf of the PLC manufacturers.

This has resulted in considerable cost and complexity for two groups: 1) users, who are tired of being locked into a whole series of decisions and high prices just because they bought a Programmable Controller or BAS (Building Automation System) from a particular vendor; 2) smaller vendors, who often make innovative, advanced products, but are locked *out* by proprietary protocols.

LonWorks is a very important technology, not only because it addresses most of the technical concerns of Industrial Networking, but also because it is supported by major players in the Building Automation Industry; Siemens, Honeywell, and Johnson Controls. This gives it a very important position in the marketplace.

How you can implement LonWorks quickly & effectively?

Once you’ve made a decision to add LonWorks connectivity you have a number of ways to proceed. The choices vary in time-to-market, supportability, resource requirements and price.

1. Use an off-the-shelf Serial Gateway

An off-the-shelf serial gateway is the least economical and least functional approach to LonWorks enabling of your product. Because most LonWorks devices are simple sensor/actuator devices it is rare to find an application that can support the cost, footprint and power requirements of a gateway. Additionally, most applications require a unique set of SNVTs; meaning that there really is no such thing as a completely off-the-shelf gateway that can be used in universal applications. This contrasts with networks like DeviceNet, Profibus or Ethernet where serial data can be generically converted to the target network.

For this reason, there are few, if any LonWorks off-the-shelf gateways. If your volume is high enough (250 units or more per year) a LonWorks device manufacturer like Real Time Automation may customize one of its products to work as a serial gateway in your application. Customization consists of engineering your serial communication stream and implementing the application image (SNVTs for your application). Costs for customization vary with the complexity of your serial communications protocol and your application.

2. Add-on Daughter Boards

An add-on daughter board is actually a serial gateway embedded within the footprint of your device. An add-on daughter board has all the characteristics of the gateway described previously plus redesign of you hardware to incorporate the serial gateway PCB. Real Time Automation supplies daughter boards of this kind; not only for LonWorks but also for other fieldbuses like DeviceNet, Industrial Ethernet and Profibus.

3. Custom PCBs

Contract Engineering companies can provide you with a completely custom communications card for your application usually in 90 days or less depending on the complexity of your application. Real Time Automation supplies these types of PCBs to companies all over the world.

4. Do-It-Yourself

The most costly, lengthy and risky approach is to form an internal effort to build it yourself. While this is admittedly what you'd expect to hear from a company whose business is selling custom networking hardware and software, the facts still speak for themselves.

Costs to develop LonWorks applications are considerable higher than other network development efforts. Some of the more expensive pieces that are needed include:

Echelon Nodebuilder – The development tool used to create LonWorks devices

Echelon LonMaker – The networking tool to create a LonWorks network from a group of LonWorks devices. LonMaker does the binding of SNVTs discussed previously in this document.

Training – You will not want to proceed without 4-5 days of training at Echelon's California facility.

Parts – Transceiver and Neurons can be expensive and sometimes come in large lots. Cost vary with the particular communication channel you select.

LonWorks Sniffer – Several companies make sniffers that can detect and interpret the LonTalk protocol. It may be difficult to complete your first development without a Sniffer.

Support staff to handle ongoing support.

You will also want to obtain other LonWorks devices, preferably devices for which your product must be compativle. You will need networking cable and all the other network components.

Like all complex protocol implementations there are nuances of the LonWorks specification that are not readily discernable. The internal resources required for internal build are usually much more costly than using an outside resource or buying a component.

The risk of missing seemingly innocuous nuances and missing a ship date is much higher than the cost of outside software or engineering.

Your best customer, the Guinea Pig

Anyone who has practiced engineering for any length of time has a healthy respect for Murphy's Law. On the subject of networking, Murphy's Law states that when you roll out a new network design, the odds are that you'll have the opportunity to demonstrate your proficiency. You'll demonstrate it onsite, in front of your biggest and most important customer as they wait impatiently for their network to return to life.

Just a few more things to think about before you take the leap

Another important consideration is testing and certification. Do you have the tools and resources to test, troubleshoot and certify your LonWorks implementation? A key to selecting a vendor to get "LonWorks-enabled" is to make sure that your vendor has all the right tools and can assist you with troubleshooting, locating adequate test tools and the certifying your device.

You must consider documentation. There are standard documentation formats for LonWorks devices. Your customers will expect your products to follow these standards. To get to market quickly with documentation that your customers will understand requires that these issues be addressed early in the process.

You must consider certification. Many customers simply will not purchase products unless they have passed LonMark Certification. It's not uncommon for products to be sent back to the lab a second or third time before they finally pass. If RTA does a LonWorks implementation for your product, you are guaranteed the same first time success.

You must consider Maintenance and Support. There are keys to success for any new technology. There will be inevitable corrections and revisions to the initial specifications. How you deal with these changes matters a great deal. **You must consider Factory floor LonWorks issues** What to do when devices have duplicate ID's, what to do when supposedly "compatible" devices still won't talk to each other, how to use routers on the factory floor, and how to handle device replacement are all very important factors to carefully consider.

Real Time Automation Guarantees Your Success.

Few people in the automation have the guts to guarantee anything, but Real Time Automation guarantees that your product will be LonWorks Enabled in shortest time possible. We simply require that you meet some basic requirements, which means filling out a project application and sending us a sample of your product. Within 7 days we will provide you with a quotation and a guaranteed time line. If we don't deliver the goods on time, you get half your money back! You can't lose.

What to do next

Call or Email John Rinaldi, OEM Sales Manager at (262) 439-4999 / jsr@rtaautomation.com and ask for a LonWorks Project Application. John will be happy to discuss your specific requirements and deadlines, and discuss any aspect of LonWorks that you may still be unclear about. The RTA team looks forward to serving you as you tackle one of the most important new technologies in today's automation business. Don't let a shaky economy, protocol confusion or fear of the unknown keep you from participating in new project opportunities! LonWorks can be an important part of your product success story.