# Understanding the Neuron® Firmware Address Table

April 2009 **LONWORKS® Engineering Bulletin**

## Introduction

The Neuron firmware implements an essential data structure, the *address table*, which can contain no more than 15 entries. Running short of address table space might prevent new connections from being created, resulting in less efficient solutions.

This bulletin describes the purpose of the address table, explains how this valuable resource is being used and explains how device manufacturers and integrators can control and optimize its use.

## Purpose of the Address Table

The address table is used to store addressing information, describing the destination for outgoing data. The address table also stores details about all groups the device belongs to, and might apply to incoming data, thus.

Each bound output network variable on a given device is associated with an address table entry[1] via the address table index field of the NV_config table. When this network variable needs to be propagated, the firmware consults the associated address table entry for addressing information and transport mechanism to be used.

Each used address table entry contains an indicator for the address type (subnet/node addressing, group addressing, or broadcast addressing) and the respective addressing details. These details also include transport properties such as the retry counts, tx_timer values, etc.

Multiple network variables can share an address table entry, provided all these network variables are bound to the same destination, using the same service type and transport properties.

Figure 1 illustrates a basic scenario. Device A is bound to two remote devices, devices X and Y. Device A's address table entry 0 points to device X, and address table entry 1 points to device Y. The second address table entry is shared between two network variables on device A, NV index 0 and 2. These network variables have a unique selector value each, allowing for the receiving device Y to distinguish between an update to nvi01 and nvi02, and to route the incoming NV update accordingly.

Assuming subnet/node addressing, neither device X nor Y requires an address table entry.

[1] With the exception of polled output network variables or output network variables being bound to a polling input network variable. See later in this document for more details on polling connections.

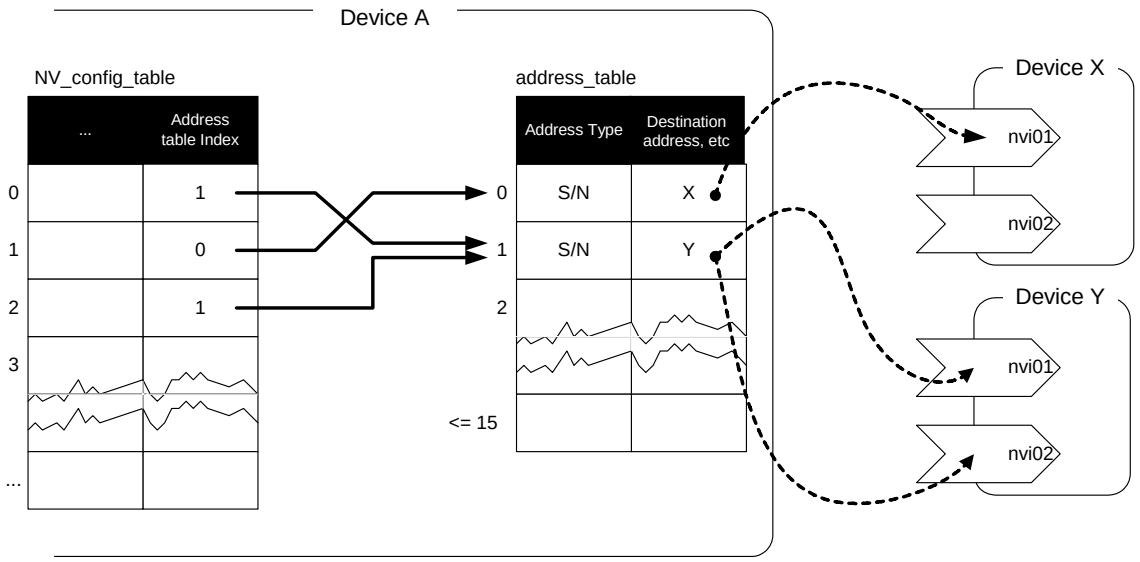


**Figure 1.** Neuron Tables and NV Updates

Typically, the address table stores the addressing information for transactions that are initiated on the local device.

In case of a polling connection, the initiating device is the polling one (the device that implements the input network variable), rather than the one implementing the output network variable. A polling connection is automatically created if the input network variable is flagged as being polling. This flag is automatically set by the Neuron C Compiler as a result of applying the poll() Neuron library function to an input network variable[2]. The total address table consumption remains unchanged, however, polling changes the location of the associated address table entry.

Figure 2 illustrates this scenario: device X requires an address table entry to be able to poll the output network variable's value from device A. Device A, in return, only requires one (shared) address table entry for the connections made to device Y.

The third purpose of the address table is to denote group membership. Each LONWORKS®device belongs to one or two domains. It will be member of a subnet for each domain, and it will have a unique identifier within each subnet, the node ID.

The node might also become member of a group. Groups allow for reliable multicast transactions over the boundaries of subnets, but require all participating devices to join the group.

[2] This flag is not to be confused with output network variables declared with the "polled" attribute. The latter must be bound to a polling input network variable, whereas the polling input network variable can be bound to any output network variable.

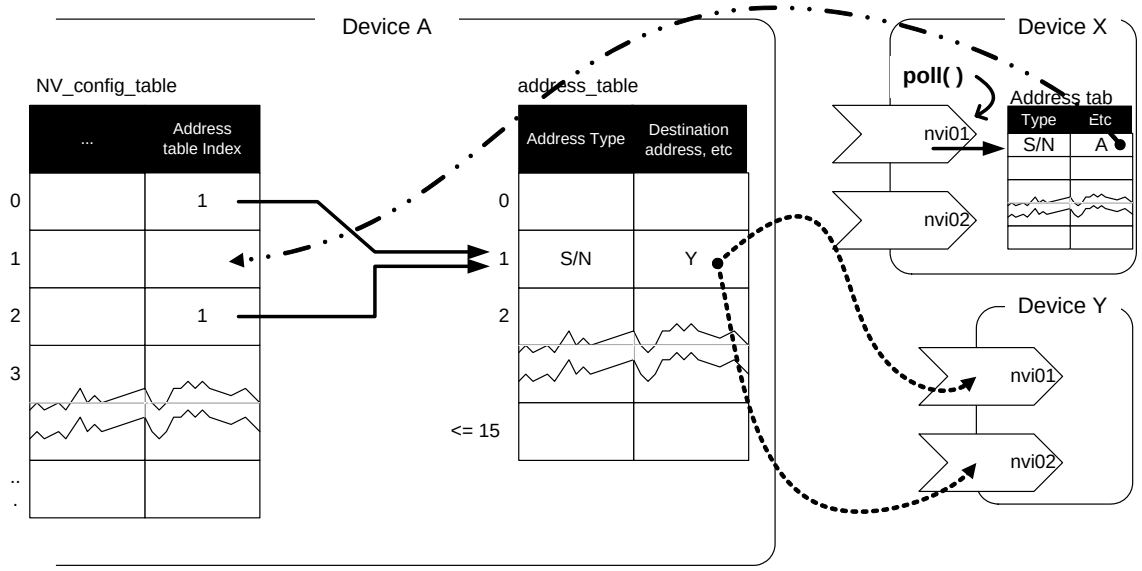


**Figure 2.** Polling Connection

An address table entry is required for each group to which the device belongs. Upon receipt of a group-addressed packet from the network, the node explores the address table, sequentially searching for an entry indicating membership to the group.

This is illustrated in figure 3: Device A is bound to both, device X and Y, using a group addressed connection. Due to the group membership information being required on all participating devices, an address table entry is required on all devices[3].

In summary, address table entries are required for the following:

- Addressing information for transactions being initiated by the local device
- Group membership information

[3] Note that this is true for all groups. The LonTalk® protocol limits the use of acknowledged service in groups to groups with no more than 64 members, whereas unacknowledged services may be used with groups of unlimited size *(open groups)*. However, the group membership information must be stored on each member device in either case.

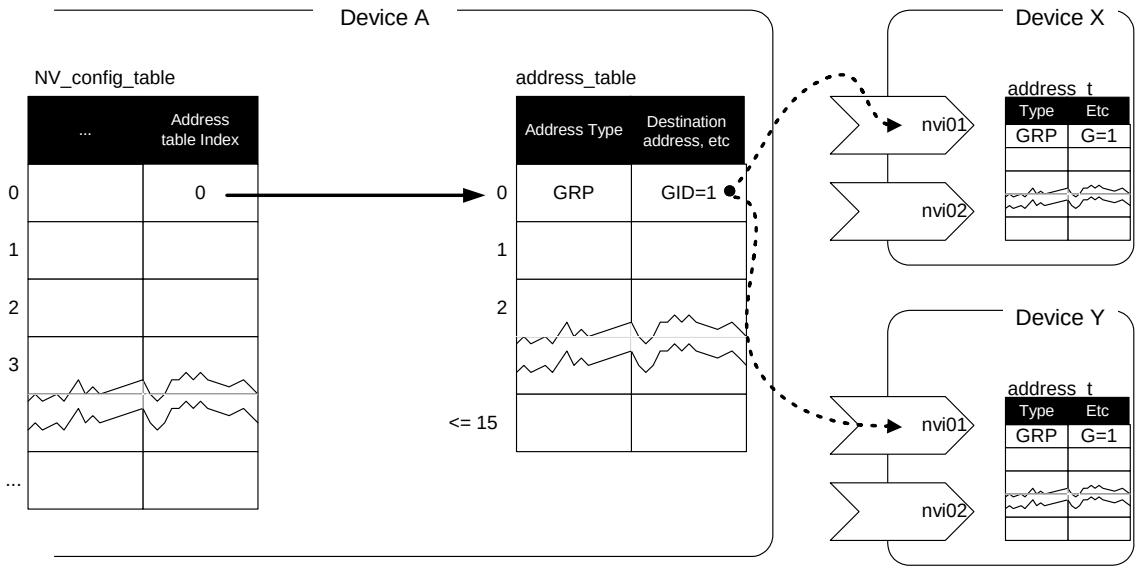


**Figure 3**. Group Connection

## Saving Address Table Consumption

In order to reduce the address table consumption on a given device without reducing the number or complexity of connections, both the integrator and the device manufacturer can assist the complex, but generic, network variable binder algorithm. Since both device manufacturer and system integrator have knowledge about the specific connection requirements as well as the future connections, that expertise should be used to prepare the devices in the best possible way, and to create connections in a resource-preserving fashion.

For the device manufacturer, the first rule should be not to limit available address table space. A maximum of 15 address table entries should be allowed if possible, allowing for a maximum number of different connection destinations.

Secondly, the device manufacturer only use the poll() function after careful consideration. In most cases, the use of the poll() function can be replaced by heartbeats in order to ensure consistent and up-to-date input network variable values after resetting or power-cycling the device. Excessive use of the poll() function can easily consume a huge amount of address table entries as well as group identifiers, not always visible to the integrator.

The integrator, too, can manage address tables consumption. As discussed above, an address table entry includes the addressing type, the destination address, and transport properties like the retry count, or the transmit timer value.

Multiple connections can only share an address table entry if the connections use equal addressing types, transport properties, and destinations. In order to achieve this, default values provided by the network management tool should be used for the transport properties if possible.

Broadcast addressing modes can also help to safe address table space being required. For example, a device using domain broadcast messages requires one address table entry to store that information. Multiple network variables might be linked to this address table entry, all using

domain broadcast addressing mechanisms. Large multicast connections with infrequent updates are likely candidates for using broadcast services.

It should be noted that broadcast addressing modes does not support acknowledged services; the only available options are unacknowledged (UACKD) and repeated (UACKD/RPT) services.

## Balancing Address Table Consumption

In some cases, address table consumption may be re-distributed across the network. The total number of address table entries being used in the entire system will remain unchanged, but a trade-off may be possible to move the requirement of an address table entry from one device to another. Consider the situation illustrated in figure 4: All members of the group connection require one address table entry to implement this scenario. The total number of address table entries being used is four.

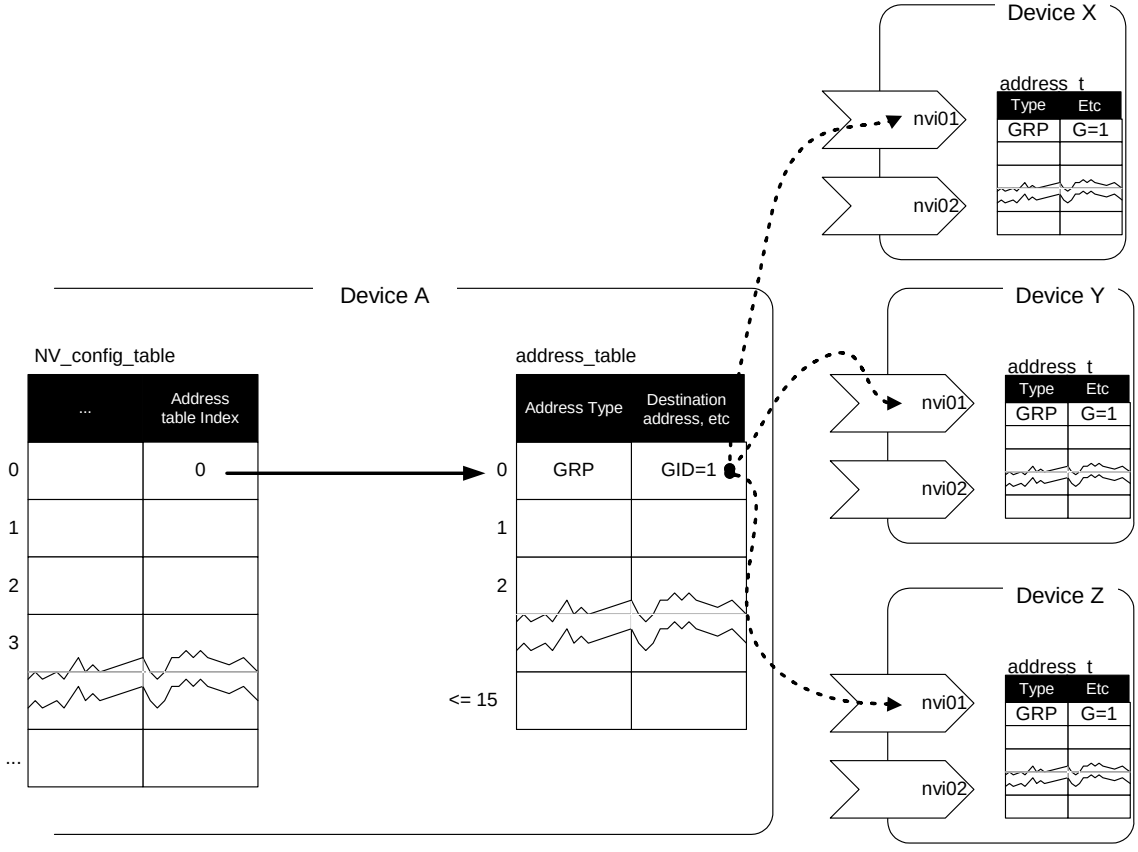


**Figure 4.** Group Multicast

Figure 5 shows a very similar solution, still requiring a total of four address table entries. In contrast to the original scenario shown in figure 4, the modified solution uses an alias to the output network variable on device A. Consequently, two address table entries are required on device A. Due to the alias being bound using subnet/node addressing, no address table entry is required on the third destination device, device Z.

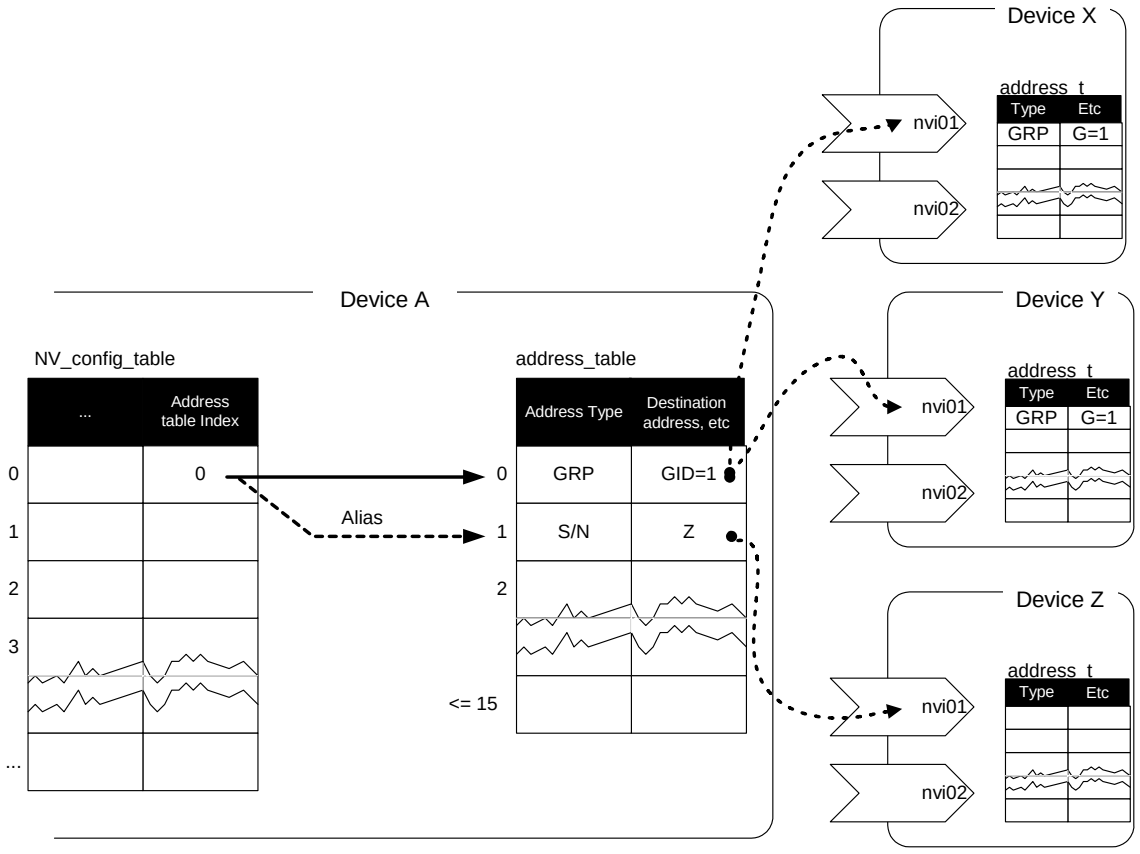


**Figure 5.** Off-loading Address Table Entries by Using Aliases

The overall effect is that one address table entry has been moved from device Z to device A by requiring the binder to use aliases rather than groups.

**Disclaimer**

Echelon Corporation assumes no responsibility for any errors contained herein. Echelon makes no representation and offers no warranty of any kind regarding any of the third-party components mentioned in this document. These components are suggested only as examples of usable devices. The use of these components or other alternatives is at the customer's sole discretion. Echelon also does not guarantee the designs shown in this document. No part of this document may be reproduced, translated, or transmitted in any form without permission from Echelon.

Part Number 005-0201-01 Rev. A

Echelon Corporation
www.echelon.com

© 2009 Echelon Corporation. Echelon, LON, Neuron, LonTalk, LONWORKS, and the Echelon logo,are trademarks of Echelon Corporation registered in the United States and other countries.