Implementing Dynamic Network Variables

August 2000
Echelon Corporation

The purpose of this document is to provide correct and complete information for LonWorks® software developers who are implementing dynamic network variables (NVs). This document is not intended to be a comprehensive guide, but rather the definitive guide that includes correct information and contains references to other correct information. When the product documentation is updated, this document will no longer be distributed.

Overview

A device that implements dynamic NVs has the ability to create and delete NVs on demand, at any time. LonWorks devices that implement dynamic NVs thus are more flexible (and consequently more complicated) than devices with a fixed network interface (the "static NVs"). LonWorks devices with dynamic NVs are typically used to build gateways to legacy control systems, to build large monitoring and control devices, and for large data logging devices.

Since LNS™ fully supports devices with dynamic NVs, any LNS-based network management tool can instruct a device to add or remove any of its dynamic NVs.

Dynamic NV Protocol

The dynamic NV protocol is documented in the LonMark Application Layer Interoperability Guidelines, in the chapter about Host Based Nodes. However, the guidelines currently do not mention that the version 2 SI data structure is required in the device in order for the protocol to work. The dynamic NV protocol is also documented in the EIA 709.1-A specification, in section 13.7.16. The EIA specification does not include documentation on the version 2 SI data structure; therefore the specification too is incomplete in this regard.

The dynamic NV protocol uses an extension of the Wink command, and requires that the device implement the Self-Identification (SI) to version 2 or greater. Section 13.7.16 of the EIA specification covers the Wink extensions (called App commands or Install commands). The following is a list of the install commands:

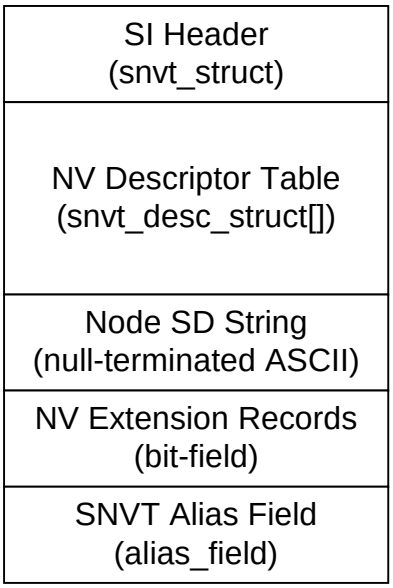
APP_WINK
APP_INSTALL
APP_NV_DEFINE
APP_NV_REMOVE
APP_QUERY_NV_INFO
APP_QUERY_NODE_INFO
APP_UPDATE_NV_INFO

These commands are sent by LNS to a device instructing it to add or remove or query information about one of its NVs.

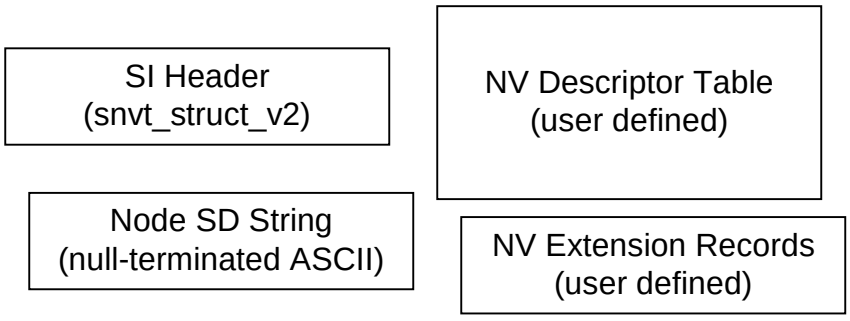
Version 2 SI Data Structure

The SI data structure diagram shown immediately below is a representation of the data structures outlined in section B.5 of the EIA 709.1-A specification. The structures outlined in the EIA specification are version 0 and version 1, and are for use with the Neuron Chip as well as host-based devices using a static NV interface (i.e. not using dynamic NVs).

The version 0 and version 1 SI data structures consists of several sections arranged as follows (from EIA 709.1-A, section B.5):



However, the version 2 SI data structure (required for dynamic NVs) is arranged as follows:



Note that each of the sections is separate from the others, and that organization and implementation is left to the device developer. This is because three of the four structures are only accessed using the dynamic NV protocol.

The Descriptor Table, SD String, and Extension records are accessed via dynamic NV protocol commands that the application must process. The device's responses to these commands are sent back to the network must follow the Dynamic NV protocol. Thus the developer is free to choose the most appropriate method of internal representation.

However, the header of the SI data containing the version of the SI data used must be formatted as follows. This structure is requested multiple bytes at a time by the Query SNVT command, and is then parsed by LNS. The data types given are NeuronC types. You should know whether your processor and environment are big-endian or little-endian, so that the correct responses are sent back to the Query SNVT command.

```
typedef struct
{
    unsigned      length_hi;       /* length of header only.  Others
                                      are read via the WINK's
                                      subcommands */
    unsigned      length_lo;       /* APP_QUERY_NV_INFO and
                                      APP_QUERY_NODE_INFO */
    unsigned      num_netvars_lo;  /* Max # of NVs which can be
                                      defined (static + dynamic) */
    unsigned      version;         /* version 2 format */
    unsigned      num_netvars_hi;  /* Max # of NVs which can be
                                      defined (static + dynamic) */
    unsigned      mtag_count;
    unsigned long static_nv_count;
    unsigned long current_nv_count;  /* Number of currently defined
                                        NVs */
    unsigned long  max_nv_in_use;    /* Maximum NV index.  0xffff
                                        indicates none defined */
    unsigned long  alias_count;
    unsigned long  node_sd_text_length;
#ifdef LITTLE_ENDIAN
    unsigned       unused      :6;
    unsigned       query_stats :1;
    unsigned       binding_II  :1;
#else
    unsigned       binding_II  :1;
    unsigned       query_stats :1;
    unsigned       unused      :6;
#endif
} snvt_struct_v2;
```