LONMARK® External Interface File Reference Guide

Revision 4.0A
April 2000

Introduction

LONMARK external interface files are files that define the external interface for one or more LONWORKS® devices. The *external interface* is the interface to a device that is exposed over a LONWORKS network. The external interface does not expose the internal algorithms of a device, instead, it only exposes the inputs to the algorithms and the outputs from the algorithms. The external interface includes the device's self-documentation information, the number of address table entries, the number of message tags, and the number, types, and directions of network variables.

Much of the external interface of a device can be queried over the network by a network tool. The device manufacturer determines the completeness of a queried interface. For example, a device manufacturer may choose to embed network variable names in a device to ensure that the queried network interface includes these names.

There are two benefits to using external interface files. First, an external interface file may include information that is not included in a device such as network variable names. Second, an external interface file can be used during network engineering when the device is not accessible from the network engineering tool.

The primary external interface for a device is a text file with a .XIF extension. Some platforms such as the LNS™ network operating system may convert this file to alternate formats for performance optimization. For example, LNS uses a binary external interface file (.XFB extension) and an optimized external interface file (.XFO extension). Both of these files are created from the date contained within the text external interface file. This document describes the format of the text external interface file. The XIF32BIN External Interface File Conversion Utility is used to convert a text external interface file to a binary external interface file. The optimized external interface file is created automatically by LNS

to reduce the access time to data within an external interface file. Other network operating systems may create their own optimized versions of the external interface file.

External interface files are typically generated by LONWORKS® development tools. Many of the fields of the external interface file for a device must match the application in the device. If an external interface file is modified in such a way that it does not match the application it is documenting, installation errors may occur for the device.

Text External Interface File Format

A text external interface file consists of the following sections:

- Header
- Network variable and message tag definitions
- File definitions (added in version 4.0)
- Network variable value definitions (added in version 4.0)

All sections are optional, except for the header section. These sections must be in the specified order, and are described in the following sections. Following are a few general rules that apply to all sections:

- If the first non-blank character on a line is '#', the entire line is ignored. This means that comment lines may be inserted anywhere, since they do not count as blank lines.
- Multiple blank lines are allowed anywhere a single blank line is required, blank lines may appear between individual network variable or message tag records and at the end of the file, and blank characters are allowed at the beginning of any line.
- In general, string fields contain an asterisk if they are not applicable or they are default values. Integer fields contain zero when they are not applicable, and asterisks when they are default values.
- The maximum line length for any line is 160 characters.

Header Section

The header section is the first section of the external interface file, and is the only required section. The header describes some basic information about the capabilities of the device, such as the transceiver type and buffer configuration.

Installation tools may use the transceiver type information to determine if a device is compatible with its intended channel. This usage is optional. An installation tool may use the external interface file solely for program definition and may ignore the transceiver type information.

Following is an example of a header section. The lines are numbered for reference in this document; these line numbers are not included in the external interface file.

```
1: File: Ao10A.XIF generated by APC Revision 2.51, XIF Version 4.0
2: Copyright (c) 1990, 1999 Echelon Corporation
3: All Rights Reserved.  Run on Mon Mar 23 18:14:27 1999
```

```
4:
5: 90:00:01:05:19:8A:04:02
6: 2 15 1 52 0 3 2 0 0 2 5 11 11 9 10 0 0 16 20 1 1 128
7: 0 5 100 13 28 726 0 15 5 3 342 4
8: 1 7 1 0 4 4 4 15 200 0
9: 78125 0 0 0 0 0 0 0 0 0 0 0
10:90 0 240 0 0 0 40 40 0 5 8 5 12 14 15
11:*
12:"&3.0@0,3[2]Analog Output,20006[2]PID Controller,20002[1]Digi
13:"tal Encoder,20005[2]Analog Fn Block,20010[2]Type Translator;
14:"Echelon LonPoint AO-10 Module version 2.x. Supports two Anal
15:"og Outputs, two PID Controllers, one Digital Encoder, two An
16:"alog Function Blocks, and two NV Type Translators.
17:
```

The header section consists of the following lines:

Line	***Contents***
Line 1	File name, source of the file, and format version number. This document describes format version 4.0.
Lines 2 – 3	Copyright information and a timestamp of when the file was created.
Line 4	Blank line.
Line 5	Program ID. This consists of eight 2-digit hex values, separated by colons (no spaces). The first hex digit identifies the program ID format. If the first digit is 7 or less, the format is an ASCII string, typically with the name of the program. If the first digit is 8 or 9, the format is the following: *FI*:*II*:*II*:*CC*:*CC*:*SS*:*TT*:*NN* The fields of the type 8 or 9 program ID are described below: • ***F – Format***. A 4-bit value defining the structure of the program ID. Program ID format 8 is reserved for LONMARK certified devices. Format 9 can be used to identify a LONMARK compliant application that has not received LONMARK certification. Formats 10 through 15 are reserved. Formats 8 or 9 must be used for devices with LONMARK objects and configuration properties. • ***I:II:II – Manufacturer ID***. A 20-bit unique ID identifying each manufacturer of interoperable LONMARK devices. The LONMARK Interoperability Association assigns this ID to a manufacturer. • ***CC:CC – Device Class***. A 16-bit ID identifying the device class. This ID is drawn from a registry of pre-defined class definitions maintained by the LONMARK Interoperability Association. • ***SS – Device Subclass***. An 8-bit ID identifying the usage of the application. This ID is drawn from a registry of pre-defined subclass definitions maintained by the LONMARK Interoperability Association. The high-order bit of the device subclass must be one if the application uses any network variables with changeable types. • ***TT – Transceiver Type***. An 8-bit ID identifying the transceiver type of the application. This ID is drawn from a registry of pre-defined transceiver

type IDs maintained by the LONMARK Interoperability Association.

- ***NN – Model Number***. An 8-bit ID identifying the specific product model. Model numbers are assigned by the product manufacturer and must be unique within the device class and subclass for the manufacturer. The same hardware may be used for multiple model numbers depending on the program that is loaded into the hardware. The model number within the program ID does not have to conform with the manufacturer's model number.

Line 6 Contains the following fields:

Field 1 Number of domains. **Must be set to 2.**

Field 2 Number of address table entries.

Field 3 Boolean that specifies whether the application handles incoming application messages. Set to 1 if the application handles incoming application messages, otherwise set to 0.

Field 4 Number of static network variable declarations (0 to 4096 for host-based applications, 0 to 62 for Neuron Chip hosted applications) in the application. Network variables arrays count as one declaration even though each array element counts as one network variable.

Field 5 Number of message tags (0 to 15).

Field 6 Number of network input buffers. Encoded as follows:

Count	***Encoded Value***
0	0
1	2
2	3
3	4
5	5
7	6
11	7
15	8
23	9
31	10
47	11
63	12
95	13
127	14
191	15

Field 7 Number of network output buffers. Encoded as described under network input buffers (field 6).

Field 8 Number of priority network output buffers. Encoded as described under network input buffers (field 6).

Field 9 Number of priority application output buffers. Encoded as described under network input buffers (field 6).

Field 10 Number of application output buffers. Encoded as described under network input buffers (field 6).

under network input buffers (field 6).

Field 11 — Number of application input buffers. Encoded as described under network input buffers (field 6).

Field 12 — Network input buffer size. Encoded as follows:

Size	*Encoded Value*
20	2
21	3
22	4
24	5
26	6
30	7
34	8
42	9
50	10
66	11
82	12
114	13
146	14
210	15
255	0

Field 13 — Network output buffer size. Encoded as described under network input buffer size (field 12).

Field 14 — Application output buffer size. Encoded as described under network input buffer size (field 12).

Field 15 — Application input buffer size. Encoded as described under network input buffer size (field 12).

Field 16 — Application type, encoded as follows. (Field added in version 2)

Value	*Type*
0	Unknown
1	MIP application without a host application; no network variables or message tags
2	Neuron Chip-hosted application; 62 network variables and 62 aliases maximum
3	Host application with host selection of network variables; 4096 network variables and 8192 aliases maximum
4	Host application with network interface selection of network variables; 62 network variables and 62 aliases maximum

Field 17 — Size of the network variable configuration table for MIP applications (not host applications) with network interface selection enabled. Set to 0 for all other applications, including host applications and Neuron Chip-hosted applications. (Field added in version 2)

Field 18 — Number of receive transaction buffers. (Field added in version 2)

Field 19 — Number of network variable alias table entries (0 to 4096) provided by the device. (Field added in version 3.1)

Field 20 — Boolean that specifies whether relaxed binding constraints are allowed. Set to 1 if two output network variables on the same device that are not polled by an input network variable may use the same network variable selector, otherwise set to 0. For Neuron Chip-hosted applications and host-based applications using network interface selection, this should be set to match the capabilities of the Neuron Chip firmware; for host-based applications using host selection, this should in general be set to 0. (Field added in version 3.1)

Field 21 — Specifies whether the statistics-relative address references are allowed. Set to 1. (Field added in version 3.1)

Field 22 — Maximum size memory block that may be written at a time. For devices with flash memory, this is the flash sector size. For other devices, this value is 11 bytes. (Field added in version 3.1)

Field 23 — Maximum number of network variables this device supports, which is equal to the number of static network variables defined in field number 4 plus the maximum number of dynamic network variables supported by the application. This can be no greater than 4096 and must be greater than or equal to the number of static network variables given in field number 4. (Field added in version 4.0)

Line 7 — Describes the Neuron Chip configuration. Contains the following fields (fields 3 and beyond changed with version 3):

Field 1 — Protocol processor model. Encoded as follows:

Value	***Model***
0	Neuron 3150 Chip
8	Neuron 3120 Chip
9	Neuron 3120E1 Chip
10	Neuron 3120E2 Chip
11	Neuron 3120E3 Chip
12	Neuron 3120A20 Chip
13	Neuron 3120E5 Chip

Field 2 — Protocol processor clock rate. Encoded as follows:

Value	***Model***
1	625 kHz
2	1.25 MHz
3	2.5 MHz
4	5 MHz
5	10 MHz
6	20 MHz

Field 3 — System firmware revision number encoded as a decimal integer value.

Field 4 — Receive transaction block size in bytes.

Field 5 — Transaction control block size in bytes.

Field 6 — Number of bytes of on-chip RAM from the end of the system area that precedes the receive transaction blocks to the first user variable or the end of on-chip RAM, whichever comes first.

Field 7 — Number of bytes of off-chip RAM from the end of the available RAM that may be used by the Neuron Chip Firmware to the first user variable or the end of off-chip RAM, whichever comes first.

Field 8 — Domain table entry size in bytes.

Field 9 — Address table entry size in bytes.

Field 10 — Network variable configuration table entry size in bytes.

Field 11 — Number of bytes from the beginning of the domain area up to the first byte of user code in EEPROM.

Field 12 — Network variable alias table entry size in bytes. Set to 0 if aliases are not supported in the device. (Field added in version 3.1)

Line 8 — Describes the channel parameters. Contains the following fields (fields added in version 3):

Field 1 — Boolean that specifies whether a standard transceiver type is used. Set to 1 if a standard transceiver type is used, otherwise set to 0.

Field 2 — Standard transceiver type ID.

Field 3 — Reserved. **Must be set to 1**.

Field 4 — Transceiver interface type. Encoded as follows:

Value	***Model***
0	Not specified
1	Single ended
2	Special purpose
5	Differential

Field 5 — Transceiver interface rate. Encoded as follows:

Value	***Model***
0	1.25 Mbps
1	625 kbps
2	312.5 kbps
3	156.3 kbps
4	78.1 kbps
5	39.1 kbps
6	19.5 kbps

7	9.8 kbps
8	4.9 kbps
9	2.4 kbps
10	1.2 kbps
11	0.6 kbps

Field 6 — Number of priority slots on the channel (0 – 127).

Field 7 — Minimum clock rate for the channel. Encoded with the same values as the clock rate in line 7 field 2.

Field 8 — Average packet size in bytes.

Field 9 — Protocol processor oscillator accuracy in parts per million.

Field 10 — Protocol processor oscillator wakeup time in microseconds.

Line 9 — Describes the transceiver parameters. Contains the following fields (fields added in version 3):

Field 1 — Channel bit rate in bits per second.

Field 2 — Special purpose mode alternate channel bit rate in bits per second. Set to 0 for devices that do not use special purpose mode transceivers.

Field 3 — Boolean that specifies whether a special purpose mode transceiver controls the preamble. Set to 1 if the transceiver controls the preamble, otherwise set to 0. Set to 0 for devices that do not use special purpose mode transceivers.

Field 4 — Special purpose mode wakeup pin direction. Set to 0 for input, 1 for output. Set to 0 for devices that do not use special purpose mode transceivers.

Field 5 — Boolean that specifies whether the device can override the general purpose data used for special purpose mode. Set to 1 if the device can override, otherwise set to 0. Set to 0 for devices that do not use special purpose mode transceivers.

Fields 6 - 12 — General purpose data used for special purpose mode. Set to 0 for devices that do not use special purpose mode transceivers.

Line 10 — Describes the channel timing parameters. Contains the following fields (fields added in version 3). All field values are in tenths of a bit time, except as noted.

Field 1 — Receive start delay.

Field 2 — Receive end delay.

Field 3 — Indeterminate time.

Field 4 — Minimum interpacket time.

	Field 5	Preamble length.
	Field 6	Turnaround time (microseconds).
	Field 7	Missed preamble time.
	Field 8	Packet qualification time.
	Field 9	Boolean that specifies whether raw data overrides the timing values. Set to 1 if raw data overrides, 0 otherwise.
	Field 10	Raw data clock rate. Encoded with the same values as the clock rate in line 7 field 2.
	Fields 11 - 15	Raw data bytes for the communications parameters.
Line 11	Contains a single asterisk indicating the end of the transceiver parameters.	
Lines 12 - N	Device self-documentation string. If the documentation string is not supplied, there is a single line containing a single asterisk. If supplied, the documentation lines each begin with a double-quote character (not part of the documentation string). Multiple lines must be concatenated without any intervening characters. There is no end double-quote, instead the line is terminated by a newline. The characters of the string must all be printable ASCII characters (this includes spaces, but not tabs). Trailing spaces are included. The line may be up to 60 characters long, not including the starting double-quote character or the newline. Any non-printable characters must be encoded using an ANSI C hex character escape sequence of "**\x***HH*" where *H* represents a single hexadecimal digit. The values A-F within a hex character escape sequence must be specified with upper case letters exclusively. If the static interface contains LONMARK Objects, the device self-documentation string should be formatted as described in *The LONMARK Interoperability Guidelines*.	
Line N+1	Blank line.	

Network Variable and Message Tag Definition Section

This section consists of zero or more network variable or message tag definitions. The number of network variable definitions that follow must be the same as the number of static network variable declarations specified in field 4 of line 7 of the header.

Network Variable Definition

Following is an example of a network variable definition. The lines are numbered for reference in this document; these line numbers are not included in the external interface file.

```
1: VAR nvo01Value 2 0 0 0
2: 0 1 63 1 0 1 0 1 0 1 0 0 0
3: "@1|2
```

```
4: 95 *   2
5: 1 0 0 0 0
6: 1 0 0 1 0
```

A network variable definition consists of the following lines:

Line	***Contents***
Line 1	A line with the following syntax: **VAR** *name index avgRate maxRate arraySize* The fields are defined as follows: *name* — The network variable name (maximum of 16 characters). *index* — The network variable index specified as a decimal string (0 - 4095). *avgRate* — The average rate estimate specified as an encoded decimal string (0 – 250). Encoded as an unsigned decimal n, where the rate estimate = $2^{(n/8)-5}$. Set to 0 if the estimate is not specified. *maxRate* — The maximum rate estimate specified as an encoded decimal string (0 – 250). Encoded as an unsigned decimal n, where the rate estimate = $2^{(n/8)-5}$. Set to 0 if the estimate is not specified. *arraySize* — The number of network variables in a network variable array, or 0 if this network variable is not an array. Each element of a network variable array is assigned a unique network variable index number. The network variable index number for the entry following that for an array must be equal to the index number of the first element of the array plus the number of elements in the array. (Field added in version 2).
Line 2	Contains the following fields: Field 1 — Specifies whether the device should be taken offline before updating the variable. Set to 0 if the variable can be updated when online or offline, or 1 if it should be updated only when offline. Field 2 — Must be set to 1. Field 3 — Must be set to 63. Field 4 — Network variable direction. Set to 0 for an input, 1 for an output. Field 5 — Default service type to use for connections containing this variable. Set to 0 for acknowledged, 1 for repeated, or 2 for unacknowledged.

Field 6	Specifies whether the service type can be changed in the field. Set to1 if the type can be changed, 0 if it cannot.
Field 7	Specifies the authentication default for the network variable. Set to 1 to use authentication for the network variable by default, 0 to not use authentication by default.
Field 8	Specifies whether the use of authentication can be changed in the field. Set to 1 if the use of authentication can be changed, 0 if it cannot.
Field 9	Specifies the default use of priority for the network variable. Set to 1 to use priority for the variable by default, 0 to not use priority by default.
Field 10	Specifies whether the use of priority can be changed in the field. Set to 1 if the use of priority can be changed, 0 if it cannot.
Field 11	Specifies the polled attribute of the network variable. For an input, set to 0 if the application program does not poll using this variable, 1 if it does. For an output, set to 0 if the network variable sends unsolicited updates, 1 if the network variable must be polled for updates.
Field 12	Specifies the synchronized attribute of the network variable. Set to 0 if the network variable is not synchronized, 1 if the network variable is synchronized (i.e. all outputs are transmitted and their order is preserved).
Field 13	Specifies the configuration attribute of the network variable. Set to 0 for a configuration class network variable; 1 for a non-configuration class network variable.

Lines 3 - N

This line and the following lines define the network variable's self-documentation. If the variable has no self-documentation, the line contains a single asterisk. If supplied, one or more lines of text appear here; each line begins with a double-quote character and ends with a newline. When the lines are concatenated together without the double-quote or newline characters, this forms the self-documentation text. Each line may be up to 60 characters long not including the double-quote or newline. Any non-printable characters must be encoded using an ANSI C hex character escape sequence of "**\x***HH*" where *H* represents a single hexadecimal digit. The values A-F within a hex character escape sequence must be specified with upper case letters exclusively.

If the variable is part of a LONMARK Object, the variable's self-documentation string must be formatted as described in *The LONMARK Interoperability Guidelines*.

Line N+1

The first line after the self-documentation provides network variable type information. The line has the following syntax:

snvtIndex ***** *elementCount*

The fields are defined as follows:

snvtIndex Specifies the SNVT index (1 to 255) or 0 if this variable is a user-defined network variable type. See the *SNVT Master List* for a list of SNVT indexes.

elementCount Number of elements (1 to 256) in a network variable structure or union. Set to 1 if the network variable is not a structure or union.

Lines N+2 - M Network variable characteristics. If the network variable is not a structure or union, there is just one line. If the network variable is a structure or union, there is one line for each data element of the structure or union.

Each line has the following syntax:

type offset size signedFlag arraySize

The fields are defined as follows:

type Network variable data type. One of the following values:

Value	Data Type
0	Character
1	8-bit Integer (Neuron C “short”)
2	16-bit Integer (Neuron C “long”)
3	Bitfield
4	Union
5	Typeless. None of the remaining fields are applicable.

offset Network variable bitfield offset (0 to 7). Set to 0 if the network variable is not a bitfield.

size Number of bits in a network variable bitfield (1 to 7), or the number of bytes in a union (1 to 31). Set to 0 if the network variable is not a bitfield or union.

signedFlag Set to 0 if the network variable type is unsigned, 1 if signed. Set to 0 if not applicable.

arraySize Set to 0 if the network variable type is not an array or, if the type is an array, the size of the array (1 to 31 bytes).

Following are several example network variable declarations and the corresponding external interface file definitions. See the *Neuron C Programmer's Guide* for a description of network variable declarations.

Example 1:

```
network output polled long
   bind_info(offline ackd(nonconfig) authenticated(nonconfig)
   priority(nonconfig) rate_est(123) max_rate_est(234)) outvar;


VAR outvar 0 69 76 0
1 1 63 1 0 0 1 0 1 0 1 0 0
*
0 *   1
2 0 0 1 0
```

Example 2:

```
network input sync config int invar;


VAR invar 1 0 0 0
0 1 63 0 0 1 0 1 0 1 0 1 1
*
0 *   1
1 0 0 1 0
```

Example 3:

```
typedef struct {
   int x;
   long y;
   int array[5];
   unsigned z : 3;
   unsigned zz : 5;
   union {
      int a;
      int b;
   } u;
} group;

network input group ingroup;


VAR ingroup 2 0 0 0
0 1 63 0 0 1 0 1 0 1 0 0 0
*
0 *   6
1 0 0 1 0
2 0 0 1 0
1 0 0 1 5
3 0 3 0 0
3 3 5 0 0
4 0 1 0 0
```

Message Tag Definition

Following is an example of a message tag definition. The lines are numbered for reference in this document; these line numbers are not included in the external interface file.

```
1: TAG user_tag 0 69 76 0
2: 0 1 63 1 0 1 0 1 0 1 0 0 0
```

A message tag definition consists of the following lines:

Line	***Contents***
Line 1	A line with the following syntax: **TAG** *name index avgRate maxRate zero* The fields are defined as follows: *name* — The tag name (maximum of 16 characters). *index* — The message tag index specified as a decimal string (0 - 14). *avgRate* — The average rate estimate specified as an encoded decimal string (0 – 250). Encoded as an unsigned decimal n, where the rate estimate = $2^{(n/8)-5}$. Set to 0 if the estimate is not specified. *maxRate* — The maximum rate estimate specified as an encoded decimal string (0 – 250). Encoded as an unsigned decimal n, where the rate estimate = $2^{(n/8)-5}$. Set to 0 if the estimate is not specified. *zero* — Set to 0. (Field added in version 2).
Line 2	A line with the following syntax: **0** *bindFlag* **63 1 0 1 0 1 0 1 0 0 0** The bindFlag field specifies whether the tag is bindable. Set to 1 if it is, 0 if it is not. **In general, this should be set to 1.**

Following is an example message tag declaration and the corresponding external interface file definition. See the *Neuron C Programmer's Guide* for a description of message tag declarations.

```
msg_tag bind_info(rate_est(123) max_rate_est(234)) user_tag;


TAG user_tag 0 69 76 0
0 1 63 1 0 1 0 1 0 1 0 0 0
```

File Definition Section

This section defines the LONMARK files used for defining configuration properties. These files consist of zero or one template file definitions followed by zero, one, or two value file definitions. If a template file is defined, one or two value files must be defined; however, the contents of value files may be empty. This section was added for version 4.0 external interface files and is not present in version 3.1 and earlier files.

A file definition consists of the following lines:

Line	***Contents***
Line 1	A line with the following syntax:

FILE *name index type* [*length*]

The fields are defined as follows:

name	The filename. May be up to 16 characters without spaces.
index	The file index as defined in the LONMARK file transfer protocol. Set to 0 for the template file, or 1 or 2 for the value files.
type	The file type as defined in the LONMARK file transfer protocol. Set to 2 for the template file, or 1 for the value file.
length	The number of bytes in the file. This value is optional and is calculated from the contents of the file, but must be specified if the contents of the file are not specified. When not required, the value may be omitted or set to 0. If both the length and file contents are specified, the length value must equal the number of bytes in the file contents.

Lines 2 – N File contents. A line can be interpreted as characters or as binary data.

Character format is indicated by a double quote (") as the first non-white space character. The quote is not included in the file. In this format a subsequent double quote is considered to terminate the string and it and all subsequent characters are not included. Non printable characters can be included by using a C-style hex escape sequence. The values A-F within a hex character escape sequence may be specified with upper or lower case letters. For example, to include an 0x8A character, enter `\x8a` or `\x8A` in the string.

Binary format is assumed for any line not starting with a double quote (excluding white space). In binary format, numbers are entered using C-style hex values. Each value may optionally start with a “0x” or “\x” prefix. Values may optionally be separated with commas or spaces. If separators are not used, every pair of values represents one hex byte. Non-hex value characters are ignored. For example, the following generates a four-byte value of 0x0789abcd:

```
0x07, 0x89, 0xAB, 0xCD
0x0789abcd
0789abcd
7,89,ab,cd
\x07\x89\xab\xcd
```

N+1 Blank line.

Network Variable Values Definition Section

This section defines default values for configuration properties implemented as configuration network variables. This section was added for version 4.0 external interface files and is not present in version 3.1 and earlier files.

The network variable values definition section consists of the following lines:

Line	*Contents*
Line 1	The **NVVAL** keyword.

Lines 2 – N	A definition line for each configuration network variable defined in the external interface file. The order of the definitions must match the order of declaration of the configuration network variables in the external interface file, and there can be no more values than there are configuration network variables in the external interface file. Each line contains the default values, in hex. Each value may optionally start with a "0x" or "\x" prefix. Values may optionally be separated with commas or spaces. If separators are not used, every pair of values represents one hex byte. Non-hex value characters are ignored. For example, the following generates a two-byte value of 0x0789:
	`0x07, 0x89`
	`0x0789`
	`0789`
	`7,89`
	`\x07\x89`

Line N	Blank line.

Following is an example network variable values definition. Comments are used to identify each of the values.

```
NVVAL
# config network input long configNv1 = {5000};
0x13, 0x88
# config network input int configNv2 = {100};
0x64
# config network input long configNv3 = {2252};
0x08, 0xCC
```

Revision 4A, April 2000